

Asset Pricing in Neural Network Models

Shanyan Lai

Doctor of Philosophy

University of York

Economics and Related Studies

September 2025

Abstract

The thesis investigates the application of advanced deep learning methods to predict stock returns within the framework of empirical asset pricing. This study explores how traditional and advanced neural network models can improve the predictive accuracy of stock excess returns and to what extent these models can price them. Concretely, it applies neural network frameworks, such as Multilayer Perceptron (MLP), recurrent neural network (RNN) and its variations, RNN attention models and Transformer models, as substitutes for the traditional linear framework of the asset pricing factor model. These neural network models are tested on monthly data for 420 large-cap US stocks from January 1957 to December 2021 (Chapter 2) and December 2022 (Chapter 3 and 4). This extensive sample enables a robust assessment across diverse market regimes, capturing idiosyncratic volatility clusters and structural shifts during the COVID-19 pandemic and its subsequent recovery. The 182 firm characteristics-sorted factors explored by previous researchers are selected from the literature.

Chapter 2 develops dynamic pyramidal MLP structures for factor models that learn nonlinear mappings from factors to stock excess returns. The MLP models with 2 hidden layers outperform alternative MLP and traditional linear models on predictive power. In backtesting, with the dynamic transaction cost scenario, the deep MLP models slightly outperform the buy-and-hold benchmark in both portfolios and periods. The results also prove the capability of MLP models in moderating the extreme downside risks. Chapter 3 proposed two tailored innovative pre-trained RNN attention structures for stock return prediction under the empirical asset pricing framework, which are the pre-trained RNN global self-attention model and the sliding window sparse attention model. They are benchmarked with the pre-trained RNN models and their variants (LSTM and GRU) and pre-trained RNN attention models (additive, Luong's three attentions). The out-of-sample R^2 shows the RNN model has significant advantages in modelling the stock return movements during the extreme market fluctuations, which is followed by the proposed models. Nonetheless, the backtesting results show that the proposed models outperform all alternative benchmarks on profitability in both equal-weighted and value-weighted portfolios. Chapter 4 proposes an innovative Transformer model, Single-directional representative from Transformer (SERT). It applies pre-trained Transformer models in the context of stock pricing. They are compared with standard Transformer models and encoder-only Transformer models across three periods. The best proposed SERT model achieves the highest out-of-sample R^2 during the extreme market shocks, followed by pre-trained Transformer models. Their backtesting performance proves the excellent capability for hedging extreme downside risks. It also proves that Transformer models have high capabilities to capture patterns of highly volatile temporal sparsity data in the asset pricing context. The dynamic transaction cost robustness examination shows that the profitability of these models is highly eroded by the high turnover rate.

Dedication

To my parents, for their endless love and quiet faith that shaped my strength.

To my friends, who brought light to the most difficult days.

To my colleagues, whose kindness and encouragement helped me carry on.

To those who saved my life, helped, and supported me unconditionally - you are deeply
cherished.

To those who challenged me in my darkest days: I remain grateful for the trials that forged
my strength and the hardships that taught me toughness.

To the moment love was silent, yet understood.

Acknowledgements

This thesis marks the culmination of a long and challenging journey, one that would not have been possible without the support, encouragement, and guidance of many individuals and institutions. I am deeply grateful to all those who have contributed to the completion of my doctoral studies.

First and foremost, I would like to express my sincere gratitude to my supervisor, Professor Peter N Smith, for his invaluable guidance, insightful advice, and unwavering support throughout this research. His intellectual rigour, patience, and encouragement have shaped not only this thesis but also my development as a researcher. I am also thankful to my Thesis Advisory Panel (TAP) members, Dr Mark Hallam and Prof Laura Coroneo, for their constructive feedback throughout this work. Furthermore, I would like to extend my appreciation to my internal examiner, Dr Mark Hallam, and my external examiner, Dr Chrysavalantis Vaslakis, for their meticulous work and the insightful comments they provided on this thesis.

I would also like to extend my appreciation to Professor Jia Chen (University of Macau), Professor Degui Li (University of Macau) and Professor Michael Thornton (University of York) for the advice on the model structuring. My sincere thanks also go to Dr Mark Stevenson (University of Oxford) for his insightful comments and meticulous proofreading. I am grateful for the access to computational resources provided by the Viking HPC Cluster of York University, which played an essential role in conducting the experiments and analyses presented in this work.

I owe a special debt of gratitude to my mentors and friends. Particularly, Dr Penn Rainford (University of York) and Dr Pedro Ribeiro (University of York). As my mentors from the Computer Science Department of the University of York, they provided me with patient guidance, insightful advice, and continuous encouragement throughout my doctoral journey. Their expertise and generous support have greatly shaped my research and helped me overcome many challenges along the way. Also, special thanks go to Dr Jiaxing Peng (University of York), who gave the crucial advice on the research data source for this thesis. I'm also grateful for Dr Frank Soboczinski and Dr Dimitar Kazakov's insightful advice and ideas on Transformer models. And special thanks to Qiran Lai (University of York) and Dr Jingbo Yang (University of York) for their insightful discussions, generous sharing of ideas and technical support. And Dr Naiheng Zhang (Beijing Institute of Technology), for his technical support in building GPU-based Python and PyTorch environments. Their friendship and encouragement provided strength during moments of doubt and difficulty. And special thanks to Dr Grega Smrkolj (Newcastle University) for his gentle encouragement and warm advice on this PhD journey. Together with Professor Robert Sollis (Newcastle University), Dr Xueqi Dong (Newcastle University) and other Newcastle University professors and lecturers who

taught me in my master's program. The knowledge and insights they imparted to me formed the cornerstone of my doctoral research and have continued to shape my academic journey.

Finally, and most importantly, I would like to express my heartfelt thanks to my family, especially my parents: Chengzhi Lai and Jingjuan Zhu. Their unconditional love, patience, and faith in me have been my greatest source of motivation and resilience. To my parents, who taught me the value of perseverance and curiosity.

To all those who have been part of this journey in one way or another — thank you. This thesis is as much yours as it is mine.

Shanyan Lai,
York,
September 2025.

Declaration

I declare that this thesis is a presentation of original work and I am the sole author. This work has not previously been presented for a degree or other qualification at this University or elsewhere. All sources are acknowledged as references.

Parts of this thesis have been previously disseminated in the form of working papers authored by me during the course of my doctoral research. These papers present earlier versions of the research developed in Chapters 2, 3, and 4, respectively. Each of them has been substantially revised, extended, and integrated into the present thesis.

The working papers are listed below:

Lai, S., 2025. Multilayer Perceptron Neural Network Models in Asset Pricing: An Empirical Study on Large-Cap US Stocks. Working paper, available at arXiv, SSRN, and Zenodo.

Lai, S., 2025. Is attention truly all we need? An empirical study of asset pricing in pretrained RNN sparse and global attention models. Working paper, available at arXiv, SSRN, and Zenodo.

Lai, S., 2025. Asset Pricing in Pre-trained Transformer. Working paper, available at arXiv, SSRN, and Zenodo.

Shanyan Lai,
York,
September 2025.

Contents

List of tables	viii
List of figures	x
1 General Introduction	1
2 MLP in stock pricing and factor investing	14
2.1 Introduction	14
2.2 Related works	17
2.3 Data Description	24
2.4 Models	28
2.4.1 Multilayer Perceptron Algorithms (MLPs)	29
2.4.2 Loss Function and Model Estimation	33
2.4.3 Early Stopping and Batch Normalization	35
2.4.4 Gradient Vanishing and Explosion	36
2.4.5 Benchmark Models	37
2.4.6 Model Performance Evaluation	38
2.5 Empirical application and results	42
2.5.1 Empirical Application and Results	42
2.5.2 Model performance evaluation	44
2.5.3 Evaluation of Back Testing Performance	58
2.6 Conclusion and Discussion	73
3 Attention is Truly All We Need?	76
3.1 Introduction	76
3.2 Related work	81
3.3 Data description	87
3.4 Models	88
3.4.1 MLP autoencoder for pertaining	90
3.4.2 Recurrent Neural Networks	93
3.4.3 Attention mechanisms for attention layer	93
3.4.3.1 Additive attention mechanism	95
3.4.3.2 Luong's attention mechanism	96
3.4.3.3 Global self-attention RNN model	97
3.4.3.4 Causal masks, attention weights and attention outputs for global attentions	98
3.4.3.5 Sliding window sparse attention mechanism	98
3.4.4 Linear output of the RNN attention models	101

3.4.5	Long short-term memory (LSTM) and gated recurrent unit (GRU) as benchmark models	101
3.4.6	Estimation	105
3.4.7	Forecasting	105
3.5	Empirical results	106
3.5.1	Model performance	106
3.5.2	Back-testing performance	116
3.6	Conclusion	134
4	Asset Pricing in Pre-trained Transformers	143
4.1	Introduction	143
4.2	Related work	147
4.3	Data description	158
4.4	Models	159
4.4.1	Overview of Transformer	159
4.4.1.1	Positional encoding	162
4.4.1.2	Autoencoder	164
4.4.1.3	Attention mechanism	165
4.4.1.4	Add and layer normalization	170
4.4.1.5	Estimation	171
4.4.2	The proposed pre-trained Transformer	171
4.4.3	Single directional Encoder Representations from Transformer (SERT)	172
4.4.4	Transformers in stock pricing and factor investing	175
4.5	Empirical results	177
4.5.1	Model performance of Transformer	178
4.5.2	Model performance of SERT	181
4.5.3	Backtesting performance	190
4.5.3.1	Transformer	190
4.5.3.2	Encoder-only Transformer and SERT models	195
4.5.3.3	Comparative analysis	200
4.6	Conclusion	213
5	General Conclusions and Future Work	223
	References	231
	Appendices	244
	Appendix of Chapter 2	244
	Appendix of Chapter 3	274
	Appendix of Chapter 4	278

List of Tables

2.1	Data split table for initial setting of training, validation, and testing period .	27
2.2	MLP architecture comparison between our main models and GKX2020	43
2.3	Model performance during Pre-COVID-19 and COVID-19-Inclusive periods .	46
2.4	Descriptive Statistics of R^2 for Each Model	49
2.5	DM statistics all models during Period 2112 and Period 1911	55
2.6	Equal-weighted portfolio performance with (2112) and without (1911) the COVID-19 period considering the static transaction cost.	61
2.7	Equal-weighted portfolio performance with (2112) and without (1911) the COVID-19 period considering the dynamic transaction cost.	65
2.8	Value-weighted portfolio performance with and without COVID-19 period considering the static transaction cost.	68
2.9	Value-weighted portfolio performance with and without COVID-19 period considering the dynamic transaction cost.	71
3.1	Data splits for in-sample and out-of-sample data.	88
3.2	Out-of-sample Predictive Indicators and α	109
3.3	DM test statistics.	115
3.4	Equal-weighted portfolio performance with deduction of static transaction cost.	122
3.5	Equal-weighted portfolio performance considering the dynamic transaction cost.	127
3.6	Value-weighted portfolio performance considering static transaction cost. . . .	131
3.7	Value-weighted portfolio performance considering the dynamic transaction cost.	136
4.1	Data splits for in-sample and out-of-sample data.	159
4.2	Model configurations.	177
4.3	Model performance indicators of Transformers.	179
4.4	DM test results of proposed pre-trained Transformer and benchmark models.	182
4.5	Model performance indicators of SERTs and their benchmarks.	186
4.6	DM test results of SERTs and their benchmarks.	187
4.7	The backtesting performance of buy-and-hold strategies during the examined periods considering the static transaction cost.	191
4.8	The backtesting performance for the equal-weighted portfolios of Transformer models considering static transaction cost.	192
4.9	The investing performance for the value-weighted portfolios of Transformer models considering the static transaction cost.	194
4.10	The backtesting results for the equal-weighted portfolios of Transformer models considering the dynamic transaction cost.	196
4.11	The backtesting results for the value-weighted portfolios of Transformer models considering the dynamic transaction cost.	197

4.12	The investing performance for the equal-weighted portfolios of the encoder-only Transformer models considering the static transaction cost.	198
4.13	The investing performance for the value-weighted portfolios of encoder-only Transformer models considering the static transaction cost.	199
4.14	The backtesting results for the equal-weighted portfolios of the proposed SERT and the encoder-only Transformer models considering the dynamic transaction cost.	201
4.15	The backtesting results for the value-weighted portfolios of the proposed SERT and the encoder-only Transformer models considering the dynamic transaction cost.	202
4.16	Back-testing performance of the equal-weighted portfolio between the best models in each group considering the static transaction cost.	203
4.17	Back-testing performance of the value-weighted portfolio between the best models in each group considering the static transaction cost.	207
4.18	Back-testing performance of the equal-weighted portfolio between the best models in each group considering the dynamic transaction cost.	213
4.19	Back-testing performance of the value-weighted portfolio between the best models in each group considering the dynamic transaction cost.	214
1	Detailed Definitions and Sources of Factor Variables	246

List of Figures

2.1	The mechanism of the extended window method.	27
2.2	An example of simple MLP structures with (right) and without (left) hidden layers.	29
2.3	Value-weighted price index of selected 420 stocks	46
2.4	OOS R^2 distribution comparison for fw1, fw2 and fw3.	50
2.5	OOS R^2 distribution comparison for fw4, fw5 and PCR.	51
2.6	OOS MSE distribution comparison for fw1, fw2 and fw3.	52
2.7	OOS MSE distribution comparison for fw4, fw5 and PCR.	53
2.8	VI plots for the top 20 important factors of the best proposed MLP models for the three market regimes.	56
2.9	Market-cap-weighted (MC) Price Index.	58
2.10	Cumulative excess return plot for equal-weighted portfolio considering the static transaction cost.	62
2.11	Cumulative excess return plot for equal-weighted portfolio considering the dynamic transaction cost.	66
2.12	Cumulative excess return plot for value-weighted portfolios considering the static transaction cost.	69
2.13	Cumulative excess return plot for value-weighted portfolio considering the dynamic transaction cost.	72
3.1	The workflow diagram of the RNN attention models.	89
3.2	The workflow diagram of the RNN attention models.	90
3.3	MLP autoencoder as the pre-training module.	92
3.4	The comparison of sliding window sparse attention and global self attention (with causal masks).	99
3.5	Sliding window sparse attention mechanism workflow diagram.	100
3.6	Entire workflow chart for two-layer LSTM and GRU model.	102
3.7	Market-cap-weighted (MC) Price Index.	107
3.8	Out-of-sample R^2 distribution of the best fitted models.	110
3.9	Out-of-sample MSE distribution of the best fitted models.	111
3.10	Out-of-sample α distribution of the best fitted models.	113
3.11	Correlation heatmap of top 20 extracted factors and original factors for the vanilla RNN model.	117
3.12	Correlation heatmap of top 20 extracted factors and original factors for the global self-attention model.	118
3.13	Correlation heatmap of top 20 extracted factors and original factors for the sliding window sparse attention model.	119

3.14	Annualized Sharpe ratio and Sortino ratio for Equal-weighted with static transaction cost of 50bps.	123
3.15	Cumulative return plot for Equal-weighted portfolios with deduction of static transaction cost (50bps).	125
3.16	Annualized Sharpe ratio and Sortino ratio for Equal-weighted considering the dynamic transaction cost.	128
3.17	Cumulative return plot for Equal-weighted portfolios considering the dynamic transaction cost.	129
3.18	Annualized Sharpe ratio and Sortino ratio for value-weighted considering the static transaction cost.	133
3.19	Cumulative return plot for value-weighted portfolios considering the static transaction cost.	135
3.20	Annualized Sharpe ratio and Sortino ratio for Value-weighted portfolios considering the dynamic transaction cost.	137
3.21	Cumulative return plot for Value-weighted portfolios considering the dynamic transaction cost.	138
4.1	The structure of standard Transformer	161
4.2	Sinusoidal positional encoding plots	163
4.3	General attention mechanism workflow chart.	166
4.4	Workflow of single-head attention mechanism	168
4.5	workflow of multi-head attention mechanism.	169
4.6	The structure of the proposed pre-trained Transformer.	173
4.7	The structure of the proposed pre-trained Transformer with layer normalization first.	174
4.8	single directional encoder representations from Transformer (SERT) and benchmark model.	176
4.9	OOS R^2 distribution diagrams of best-performed Transformer model in each group for each period.	183
4.10	OOS MSE distribution diagrams of best-performed Transformer model in each group for each period.	184
4.11	OOS R^2 distribution diagrams of best-performed SERT models and their benchmarks.	188
4.12	OOS MSE distribution diagrams of best-performed SERT models and their benchmarks.	189
4.13	Comparison of Sortino ratios for equal-weighted portfolios considering the static transaction cost.	204
4.14	Sign equal-weighted accumulative return plots considering the static transaction cost.	206
4.15	Value-weighted accumulative return plots considering the static transaction cost.	208
4.16	Sign value-weighted accumulative return plots considering the static transaction cost.	209
4.17	Sign equal-weighted accumulative return plots with softmax signal filter considering the static transaction cost.	211
4.18	Sign value-weighted accumulative return plots with softmax signal filter considering the static transaction cost.	212

4.19	Comparison of sortinos for equal-weighted portfolios considering the dynamic transaction cost.	215
4.20	Sign equal-weighted accumulative return plots considering dynamic transaction cost.	216
4.21	Value-weighted accumulative return plots considering the dynamic transaction cost.	217
4.22	Sign value-weighted accumulative return plots.	218
4.23	Equal-weighted with Softmax trading signal filter accumulative return plots considering the dynamic transaction cost.	219
4.24	Value-weighted with Softmax trading signal filter accumulative return plots considering the dynamic transaction cost.	220
1	Factor importance plots of fw1, fw4 and fw5.	273
2	OOS α distribution of the alternative models Part 1.	274
3	OOS α distribution of the alternative models Part 2.	275
4	Factor importance of different models.	277
5	An example of a single-layer MLP autoencoder NN structure.	286

For a long time, people lacked a clear understanding of how risks influence asset returns, even after Markowitz (1952) proposed Modern Portfolio Theory (MPT), which maximized the expected returns for a given level of risk. The question further remains on how the expected returns of individual assets are correctly derived, namely, how to price these individual assets for the portfolios. Treynor (1961) informally published manuscript proposed the capital asset pricing model (CAPM), which firstly addresses the asset pricing problem, developed and elaborated by Sharpe (1964); Lintner (1969); Mossin (1966). Equation (1.1) shows a form of CAPM in Sharpe (1964).

$$E(r_j) = r_f + \beta_j(E(r_m) - r_f) \quad (1.1)$$

where, $E(r_j)$ is the expected return of the asset j , $E(r_m)$ is expected return of the market portfolio, r_f represents the risk-free rate and the exposure of asset j to systematic market risk β_j can be derived by covariance of asset return r_j and market portfolio return r_m and the variance of the market portfolio σ_m^2 .

$$\beta_j = \frac{\text{Cov}(r_j, r_m)}{\sigma_m^2} \quad (1.2)$$

This is also known as the Security Market Line (SML). This CAPM is a theoretical equilibrium pricing model, and it is based on a series of assumptions, such as investors are mean–variance optimizers (Markowitz framework), all investors share homogeneous expectations, all assets are tradable, markets are complete and frictionless (no taxes, no transaction costs, infinitely divisible assets), and investors can borrow and lend freely at the risk-free

rate. Empirically, researchers examine CAPM in a general linear form:

$$r_{j,t} - r_{f,t} = \alpha_j + \beta_j (r_{m,t} - r_{f,t}) + \varepsilon_{j,t}. \quad (1.3)$$

where α_j is the intercept of the linear regression. Economically, if the CAPM stands, which means the excess return of an asset can be fully explained by the market portfolio, the α_j , also known as anomaly or abnormal return, should be equal to zero. However, empirical studies show that asset excess returns cannot be fully explained by market portfolios, which means, in most cases, the α_j is significantly away from zero. For addressing this issue, Ross (1978) further relaxed the assumptions of CAPM and proposed the most noted theory: Arbitrage Pricing Theory (APT). It extends the CAPM to a multivariate linear form,

$$E(r_j) - r_f = \beta_{j1}E(f_1) + \beta_{j2}E(f_2) + \cdots + \beta_{jk}E(f_k) \quad (1.4)$$

where $E(f_k)$, $k \in \mathbb{Z}^+$ represents as the risk premium of factors f_k . Thus, APT removes the conditions of investor preferences and the equilibrium. Namely, the only assumption that remains is the 'no-arbitrage' in the market. Ross (1978)'s APT theory is the foundation of the classic empirical asset pricing factor models. The most well-known factor model, the Fama-French three-factor model (FF3) (Fama and French, 1993), is built on this foundation. It employed firm characteristic-sorted portfolios (size and value) as factors, together with the market portfolio factor, to interpret excess returns. From 1963 to 1990, it consistently provided a great explanation for U.S. stock excess returns. Equation (1.5) shows the FF3 model.

$$r_{j,t} - r_{f,t} = \alpha_j + \beta_{j,m} (r_{m,t} - r_{f,t}) + \beta_{j,SMB} SMB_t + \beta_{j,HML} HML_t + \varepsilon_{j,t} \quad (1.5)$$

where $\beta_m, \beta_{SMB}, \beta_{HML}$ are factor loadings of the market portfolio factor ($r_m - r_f$), firm size-sorted portfolio factor SMB and firm book-to-market value-sorted portfolio factor HML , respectively. The success of the Fama-French three-factor (FF3) model sparks a wave of research on factor exploration of asset pricing factor models. As well-cited paradigms, Carhart (1997) utilized the momentum effect (Jegadeesh and Titman, 1993) to generate the momen-

tum factor and added it to the FF3 model. Their four-factor model significantly improves the interpretability of the multi-factor asset pricing model. Fama and French (2015) develops their FF3 model into five-factor model (FF5). It adds the profitability factor and the investment factor to the FF3 model, which further improves the interpretability of the linear empirical asset pricing model. Hou et al. (2015) proposed the Q-factor model based on investment CAPM theory, which demonstrates that firms' investment and profitability determine the cost of capital. With the Q-factor frame, they select the market factor, the size factor, the investment-to-assets factor and the return-on-equity factor for pricing the excess returns, further improving the interpretability of the asset pricing model. Although authors such as Green et al. (2017); Daniel and Titman (1997)

However, with more and more factors being discovered and developed, for example, liquidity factor (Pástor and Stambaugh, 2003), idiosyncratic volatility factor (Ang et al., 2006), investor sentiment factors (Baker and Wurgler, 2006, 2007), it becomes difficult to define which factors or factor combinations are truly interpreting the asset or portfolio returns as presented in the work of Hou et al. (2019). According to Harvey and Liu (2019)'s survey, there are more than 300 valid outstanding factors for asset pricing tasks, which organized as a 'factor zoo'. Harvey and Liu (2019) warns academia that oversizing pricing factors may lead to data snooping and raise suspicions of data mining, since statistical significance does not equate to economic significance. Moreover, the traditional empirical asset pricing factor models are under the assumption of a linear function form. When the number of factors increases, the linearity of factors and asset returns could be violated. The increasing factors may cause multicollinearity, overfitting and spurious significance in a linear model, as well as lower the economic interpretability of an asset pricing model. Thus, researchers switch their concentration from competing on factor exploration to the technique of factor selection, factor engineering and the methodology of robustness examinations. Some of them relax the linear assumption and transform the model frame from linear to non-linear. The typical structured nonlinear asset pricing frameworks include types as habit formation models (Campbell and Cochrane, 1999), long-run risks (LRR) (Bansal and Yaron, 2004; Bansal et al., 2009), rare disasters (Rietz, 1988; Barro, 2006; Gabaix, 2012), regime switching (Ang

and Bekaert, 2002) and nonlinear stochastic discount factors (SDF) (Lettau and Ludvigson, 2001b; Hansen et al., 2008). Nevertheless, the traditional non-linear models still have limitations, such as rigid function forms, low dimensionality, unobservable state variables, estimation difficulty and empirical generalization.

Fortunately, the burgeoning AI techniques, especially machine learning (ML), facilitate the development of empirical asset pricing. According to the most well-accepted concept of machine learning provided by Mitchell (1997), which is "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P , if its performance at tasks in T , as measured by P , improves with experience E ", machine learning techniques have a great potential to address tasks in different domains. It can be naturally mapped into the empirical asset pricing context as well. The task (T) corresponds to predicting or explaining asset returns, while the experience (E) refers to the vast historical information contained in the factors organized by firm characteristics, macroeconomic variables or past asset returns. And the performance measure (P) may be defined either in statistical terms, such as mean squared prediction error (MSE) (Gu et al., 2020, 2021) or in economic terms, such as Sharpe ratio or pricing error α (Chen et al., 2024; Bryzgalova et al., 2019; Kelly et al., 2025). From this definition, we know that machine learning frames latently can offer a more flexible mapping from high-dimensional factors to asset returns compared with the pre-defined statistical linear or non-linear structures. Neural network models, as a significant family of ML models, are powerful at capturing patterns and forecasting. They are developed from the inspiration of the human brain functions. In human brains, neurons connect through nerves, collect and pass the information to each other for learning patterns from multi-modal data and interact with the environment. Scientists find that a well-programmed machine can duplicate this learning process as well. Compared with individual human brains, computers have a higher searching ability and an expandable memory size, which incredibly improves the processing efficiency of massive information. Therefore, the ML models, especially neural network models, theoretically could be suitable for noisy and rambling data in finance and economics. For the ML-based empirical asset

pricing model, the mathematical representation of the factor model can be written as:

$$r_j - r_f = g(f_k; \theta) + \varepsilon_j \quad (1.6)$$

where $g(\cdot)$ is the function form of ML algorithms, θ is the parameter matrix, which is the weight matrix as the terminology of machine learning. ε_j is the residual of the equation, which contains bias and zero-mean noise in the context of machine learning. Thereby, from the angle of asset pricing, the machine learning method imposes the no-arbitrage condition directly in Equation (1.6), which is shown as the absence of α . Thus, for the ML-based empirical asset pricing, it commonly calculates residual α instead of the intercept α in the linear model to examine the significance level of the α . It examines whether the expectation of the residual ε_j equals zero to prove the existence of the α . Concretely, for the pricing error α of the traditional linear factor model, α_{trad} can be presented as:

$$\alpha_{trad} = E[R_{j,t}] - \beta'_j E[F_t] \quad (1.7)$$

If decomposing Equation (1.6):

$$r_{j,t} - r_{f,t} = \underbrace{g(f_{k,t}; \theta)}_{\text{ML structure}} + \underbrace{b_j}_{\text{Bias (ML residual alpha)}} + \underbrace{\eta_{j,t}}_{\text{Zero-mean Noise}} \quad (1.8)$$

Thus, for the empirical asset pricing factor model in the context of ML, the α is transformed into:

$$b_j = E[R_{i,t}] - E[F_t^{ML}] \quad (1.9)$$

Further, the ML residual α , b_j , can be presented as the combination of traditional α and the increment of the non-linear to the linear transformation. Therefore, the residual alpha b_j can be presented as:

$$b_j = \alpha_i^{trad} + (\beta'_j E[F_t] - E[F_t^{ML}]) \quad (1.10)$$

The seminal work of machine learning in empirical asset pricing is Gu et al. (2020). It

examines traditional ML frames, such as elastic net, random forest (RF) and multilayer perceptron (MLP) neural network, and maps the 94 firm characteristics and 8 macroeconomic indicators on approximately 30,000 U.S. stock returns during 1957 to 2016. From its empirical results, its pyramidal MLP structure with 3 hidden layers outperforms all alternative models in out-of-sample (OOS) data. However, it has limitations from the perspectives of the ML model’s structure and economic interpretability. It overthinks the limitation of data size to the ML models and enlarges the dimension of factors through the Kronecker product. Thus, it is difficult to determine which factors are the primary drivers. Moreover, it does not assess robustness under extreme market conditions, such as the turbulence caused by COVID-19. From the ML model design perspective, the early-stage ML models, such as random forest and MLP, are not specifically designed for time-series or sequential data, which means they are applied to financial or economic sequential data with an assumption of the models’ sequential recognition. Also, the seminal work does not explain how to design the model structure of MLP; instead, it uses a fixed pyramidal structure based on experience. This is only the tip of the iceberg for the issues in the early-stage research of ML in empirical asset pricing. Inspired by the work of Gu et al. (2020) (GKX2020), Chapter 2 develops their fixed structure MLP models to dynamic structure MLP models, which references the work of Coqueret and Guida (2020) and applies these MLP models to pricing and predicting large-cap U.S. stock excess returns. It employs firm characteristic-sorted factors as a substitute for Gu et al. (2020)’s firm characteristics to avoid data noisiness while retaining cross-sectional information between stocks. It also simplifies the inputs by removing the unnecessarily enlarged factor dimension via the Kronecker product. Furthermore, it applies the dynamic MLP structure design method from Coqueret and Guida (2020) to dynamically adapt the factor dimension and model depth. Innovatively, it investigates the relationship between the neural network model’s depth, predictive power, and profitability in the context of asset pricing, which requires some financial and economic interpretability within ML frameworks. It also analyses the importance of factors across the three stages of extreme market turbulence caused by COVID-19, and monitors how these factors vary over time. The MLP model itself is a type of feedforward neural network that provides a flexible framework for capturing complex, nonlinear relationships between inputs and outputs. It

consists of an input layer, several hidden layers, and an output layer, whose neurons are fully connected to the neurons in the layers next to them, with linear or nonlinear activation functions enabling the network to learn interactions between the features (input factors in this case) and labels (stock excess returns in this case). By allowing for nonlinearities and higher-order interactions, the MLP enhances the explanatory and predictive power of factor models compared to traditional statistical models. It improves the economic interpretability by utilizing the firm characteristics-sorted portfolios as factors as a substitute for the firm characteristics, thus, it highly reduces the original dimension of input factors. Additionally, it separates the testing period into two segments: the period before the COVID-19 pandemic (1911) and the period containing the COVID-19 pandemic (2112), as a robustness check for the proposed MLP models under the extreme market conditions. The empirical results of Chapter 2 show that the proposed MLP model with 2 hidden layers achieves the best performance on the model's out-of-sample (OOS) fitness, significantly outperforming all traditional statistical models (e.g. OLS, PCA, PLS) and GKX2020 MLP models. It also achieves the highest Sortino ratio in both the equal-weighted and value-weighted portfolios during the testing period with pandemic impact in the static transaction cost scenario. This proves that MLP models have strong capabilities to hedge downside risk during market turbulence.

Nevertheless, MLP models have no inherent design to capture temporal dependencies. Thus, the research in Chapter 2 is based on the assumption that the stock pricing data has a temporal dependence nature, which can be captured by the MLP models. Even with the rolling or extending window method, they are still at risk of explosion of forward-looking bias within the window. It can be solved by second-generation neural network models, which have developed mechanisms for capturing temporal relations. The second generation of ML models is designed to extend the suitability for time-series data. For example, recurrent neural network (RNN) models (Elman, 1990). RNNs incorporate lagged connections that incorporate information from previous time steps into the current output. This recurrent structure makes RNNs well-suited for modelling temporal dependencies, especially for financial and economic time series data (Dixon, 2018; Dixon and London, 2021; Wang et al., 2016; Sirignano and Cont, 2021). Hence, standard RNNs were commonly applied in both academic

research and industrial practices of financial and economic domains a decade ago. However, similar to MLP models, standard RNNs with stochastic gradient descent (SGD)-based estimation methods often suffer from vanishing and exploding gradients issues, which highly affect the parameter estimation process. It limits the ability to capture longer-term dependencies. Consequently, Hochreiter and Schmidhuber (1997) developed the long short-term memory (LSTM) neural network by configuring a memory mechanism in the standard RNN structure. While it enhances dependency capture, it also increases the number of parameters, which may cause overfitting if the data size is not sufficient for the complex models, such as LSTM. Therefore, although LSTM is still widely used in modelling financial time series (Fischer and Krauss, 2017; Bao et al., 2017a), in the empirical asset pricing domain, due to the temporal sparsity property (with longer-term dependency but fewer observations, e.g. monthly or quarterly data) of economic data, less literature is found using LSTM as the factor model framework. Lately, Cho et al. (2014) developed the gated recurrent units (GRU) model, which simplifies the memory structure of LSTM, and highly improves the overparameterization issue of LSTM, which could be a more reasonable solution for empirical asset pricing. Whereas, its parameter size is still relatively large compared with MLP models and the naive RNN model. Moreover, whether it is sufficient to capture the long-term temporal dependencies that may be introduced by macroeconomic shocks and specific market regimes, such as COVID-19, remains debatable. To further improve the predictive power of the neural network models, while simultaneously controlling or balancing the parameter size to prevent the overfitting issue, scholars proposed the attention mechanisms as an auxiliary module to neural network models. The earliest attention mechanism is the additive attention mechanism, which was proposed by Bahdanau et al. (2015). The attention mechanism is a neural network component inspired by the human brain’s attention mechanism, which allows the model to compute the relative importance (attention weights) of any two positions in a sequence, namely, any two time steps of a time series, to capture the micro pattern of the time-varying data and improve the accuracy of prediction. It is originally designed for the NLP tasks, but lately is well-applied in the financial and economic domains (Hollis et al., 2018; Shabani et al., 2022; Zhou and Wang, 2024). Subsequently, a rising number of advanced attention mechanisms are proposed, from Luong et al. (2015)’s

dot product attention, general attention and concatenate attention to Beltagy et al. (2020) sliding window sparse attention. Among the aforementioned attention mechanisms, Luong’s dot product attention is the foundation of the self-attention Vaswani et al. (2017), and the self-attention is the soul of the most prevailing large language model (LLM), Transformer, which is widely used in commercial chatbots, such as OpenAI’s ChatGPT and X’s Grok. The sliding window sparse attention of Beltagy et al. (2020) is developed from the self-attention mechanism, which shortens the attention range and highly reduces the number of parameters. Although not all attention mechanisms efficiently reduce the model’s parameter size, they concentrate on longer-term sequential dependency capture. This is suitable for monthly-based asset pricing data, particularly for studying how market impulsion influences the stock price dynamics. Less literature focuses solely on RNNs or attention mechanisms in empirical asset pricing; the most relevant work is Zhou and Wang (2024), which follows the structure of Gu et al. (2021) and transforms their MLP autoencoder module to a standard RNN structure with Bahdanau et al. (2015)’s additive attention mechanism. Cong et al. (2021b); Chen et al. (2024) conducted comparative studies for additive attention-based models, including Transformers, and provided reasonable references to the potential of attention models in the empirical asset pricing context. However, as attention mechanisms were originally designed for NLP tasks, this means they fit the sequential configuration but not the time series-specific configuration. In this sense, one position can link the information from all alternative positions, including the previous positions and future positions. This characteristic may cause future information leakage (forward-looking bias) in time-series data, which results in biased predictive performance, spurious statistical significance, and distorted economic interpretation, hence undermining the credibility of empirical findings. However, the aforementioned literature does not notice the importance of the causal relationship in time-series data, and they did not enforce causal masks in their attention mechanisms to prevent the future information leakage problem. Inspired by the RNN models and attention mechanisms, Chapter 3 investigates how attention mechanisms influence the empirical asset pricing by examining the causal mask enforced additive attention mechanism Bahdanau et al. (2015), Luong’s three attention mechanisms (Luong et al., 2015), global self-attention mechanism (Vaswani et al., 2017) and sliding window sparse attention mechanism (Beltagy et al.,

2020) on the MLP pre-trained standard RNN frame. It is the first work that proposes the context-based pre-trained RNN sliding window sparse attention model, which subtracts the sliding window sparse attention mechanism from Beltagy et al. (2020)’s Longformer model, and it is also the first work to apply the pre-trained RNN Luong’s dot product attention model (L-DotProd), the pre-trained RNN sliding window attention model (sparse_att) and the pre-trained RNN global self-attention model (self_att) for the context of empirical asset pricing. These 3-module (MLP pre-train, standard RNN base and attention layer) structures not only address the temporal recognition issue that exists in MLP for asset pricing research, but also solve the ‘dimensional disaster’ issue of asset pricing factors. The MLP pre-training module is also an advanced method for handling the missing values of input factors, extracting the information (dimensional reduction) and reducing the complexity of the neural network models. Moreover, these attention mechanisms further improve the prediction accuracy and balance the parameter size and data size to prevent potential model overfitting. The entire experiment is on the 420 top large-cap U.S. stocks and 182 firm characteristic-sorted portfolio factors, and the testing period covers the Pre-COVID-19 Period (1911), the COVID-19-Inclusive Period (2112) and the Period Including COVID-19 and One-Year After (2212) to show how these models react with extreme market turbulence. The empirical results of Chapter 3, in both static transaction cost method (50bps) and dynamic transaction cost method (turnover with 20bps for large-cap stocks), prove the contributions of the advanced attention mechanisms, where the pre-trained RNN sliding window sparse attention model outperforms all alternative models in the backtesting of value-weighted portfolios, and ranks in the second place of the equal-weighted portfolios, which closely follows the pre-trained RNN global self-attention model. However, although the pre-trained vanilla RNN model achieves the best OOS fitness with the highest R^2 values, which is right above the sparse_att model and self_att model, it fails to achieve satisfactory performance in the backtesting. This verifies that high predictive power never means high profitability of the models, due to the model’s sensitivity to the market noise (high turnover rates).

By following the work of Chapter 3, Chapter 4 further examines the most prevalent attention-based large language neural network model (LLM), the pretrained Transformer, in the con-

text of empirical asset pricing. The Transformer Vaswani et al. (2017) is the workhorse model and foundation of famous commercial chatbots, such as ChatGPT, DeepSeek, Gemini and Grok. It was originally developed for textual prediction and can be implemented in different scenes, including investors' sentiment analysis. Lately, researchers implement it and its variations directly on numerical data or multimodal data. In the asset pricing domain, Kelly et al. (2025); Fallahgoul (2025); Ma et al. (2023) are the core literature for Transformer models. They deployed the encoder-only Transformer to address the asset pricing tasks from an empirical or a theoretical angle. The standard Transformer model is built upon an autoencoder structure, which contains at least one block of encoder for compressing the original input information (factors in the asset pricing context) and one block of decoder for recovering the compressed information (lagged excess stock returns). Each encoder block of a standard Transformer contains an MLP (an extended version of the model in Chapter 2) connected to a global self-attention layer (used in Chapter 3). A decoder block of a standard Transformer has a similar structure to the encoder block. The encoder and decoder are connected through a cross-attention layer, which aligns the information from both factors and lagged stock excess returns. The entire Transformer employs the positional encoding design to impose the sequential code on the time-series, which perfectly solves the sequential recognition limitation of the MLP structure and attention mechanism within the Transformer model. Moreover, the autoencoder structure augmented with the attention mechanisms in the standard Transformer model parallels the computation and reduces the computational cost. The Transformer structure is widely applied in different contexts, such as NLP, video generation, speech recognition and investor sentiment analysis, and has proved that it has competitive capability in capturing the pattern of multimodal data and forecasting. However, from the asset pricing perspective, the encoder-only Transformer, which removes the decoder module and the cross attention and only considers the information from factors, more aligns with the traditional asset pricing factor model concept. Thus, in advanced neural network models in the asset pricing domain, Kelly et al. (2025) transformed the global self-attention mechanism of the Transformer model to Luong's general attention mechanism (introduced in Chapter 3), and applied the encoder-only Transformer model to estimate the stochastic discount factor (SDF). It shed light on the Transformer for asset pricing research, but its complicated model

structure highly reduced the economic interpretability, which made it statistically significant instead of economically significant. Moreover, due to their large data size, it is computationally intensive; hence, it is not practitioner-friendly. Fallahgoul (2025) theoretically increased the economic interpretability of the encoder-only Transformer model in the context of asset pricing to a degree. Ma et al. (2023) applied an encoder-only Transformer for pricing the Chinese stock returns and volatility for constructing the portfolio-wise trading strategy. It acquired a very high Sharpe ratio, but it ignored the limitations of the attention mechanism in the Transformer model for time series, which still requires a compulsory causal mask for preventing future data leakage, even if it is under the method of rolling window or extending window. This issue may lead to unreliable empirical results. Inspired by the core literature of the Transformer in asset pricing and the variation of the encoder-only Transformer model, Bidirectional Encoder Representations from Transformers (BERT), Chapter 4 develops an MLP pre-trained single-directional Encoder-only Representations from Transformer (SERT) models with single or multi-head attention layers, and implements the MLP pre-trained standard Transformer models with single or multi-head attention layers for empirical asset pricing. These models are enforced with the causal mask to avoid future data leakage issues. Similar to Chapter 2 and Chapter 3, these models are examined on 420 top large-cap U.S. stocks and 182 firm characteristic-sorted factors in the Pre-COVID-19 Period (1911), the COVID-19-Inclusive Period (2112) and the Period Including COVID-19 and One-Year After (2212) period. The empirical results show that the pre-trained Transformer models have noticeable advantages in handling downside risks during extreme market conditions, and interestingly, the pre-trained Transformer models perform even better during extreme market turbulence than in smooth periods. The pretrained standard Transformer models perform more stably than the SERT models, and attention heads help increase the OOS model fitness to an insignificant level. However, varying the positions of the layer normalization process, which usually improves the prediction accuracy in contexts with large data size, does not work in the empirical asset pricing context. Chapter 4 proves that complex neural network models such as Transformers are still suitable for the temporal sparsity data, for example, monthly economic or financial data. Compared with the Chapter 2 and the Chapter 3 methods, the pre-trained Transformer models consider the entire spatial-temporal

relations between factors and assets. From that perspective, Chapter 4 models are theoretically more advanced than the models in Chapter 2 and Chapter 3 in models' predictive power and capability of hedging downside risks, but models are highly sensitive to capture the market noise, thereby the profitability of these models are significantly reduced due to the high turnover rates.

The main body of this thesis is organized into three self-contained chapters. Each chapter addresses a distinct but related angle of applying neural network models as machine learning methods to empirical asset pricing. Chapter 2 develops a factor model based on a dynamic multilayer perceptron (MLP) structure. Chapter 3 extends the analysis to recurrent neural networks (RNNs) augmented with advanced attention mechanisms, with a particular focus on their robustness across three different market regimes. Chapter 4 introduces the pre-trained Transformer models in the context of empirical asset pricing and examines these models in the aforementioned market regimes for testing the robustness as well, which is followed by a general conclusion.

Multilayer Perceptron neural network models in asset pricing

2.1 Introduction

Since the beginning of the Capital Asset Pricing Model (CAPM) (Treynor, 1961; Sharpe, 1964; Lintner, 1965; Mossin, 1966), the empirical asset pricing research has been dominated by the linear factor framework. This includes the most notable Arbitrage Pricing Theory (APT) (Ross, 1978) models, such as the Fama-French three-factor model (Fama and French, 1993). Their linear factor framework modelled the multi-dimensional risk factors such as size and value. Although these models established the foundation for understanding risk premiums, their linear assumptions become a challenge later when an increasing number of factors are explored by enthusiastic researchers. The factor exploration was separated into two groups: researchers in academia were attempting to explore more factors that can give the asset excess returns a better interpretation, which shows as lower or insignificant alpha, while the practitioners were developing the factors for searching for abnormal returns (α , also known as anomaly) beyond factors, which shows as higher alpha. Due to the motivation from these two groups, the number of factors explored increased dramatically. Since then, four hundred more factors have been explored for stock pricing or factor investing (Harvey and Liu, 2019). In this work, a ‘factor zoo’ was developed. However, a challenge was that the increasing complexity of factors led to a ‘dimensionality disaster’ for the naïve linear model. Although in the early stage, researchers noticed the non-linearity of the factor models, for instance, the consumption CAPM model (Breedon, 1979) and its advanced forms (Bansal and Yaron, 2004; Lettau and Ludvigson, 2001a) or ‘disaster as a factor’ (Gabaix, 2012), none of them attempt to solve the issue of dimensional reduction for factors. Nonetheless, the development of machine learning (ML) algorithms over the past two decades has sparked a

revolution in factor models for asset pricing, shedding light on the dimensionality-reduction task in factor models. Since 2017, ML algorithms have attracted significant interest among researchers in finance and economics due to their remarkable predictive power and flexibility. For example, Baek et al. (2020); Chang et al. (2021); Sarmento and Horta (2020) all utilized the ML technique for constructing financial trading strategies. In the asset pricing domain, Gu et al. (2020) tested the most classic ML algorithms such as Elastic Net, random forest (RF), gradient-boosted regression trees and Multilayer Perceptron (MLP), benchmarked with OLS, principal components regression (PCR), and partial least square (PLS) on asset pricing tasks of U.S. stocks. It is considered the seminal work of ML application for asset pricing (AP) assignment, which made it the most cited paper in that domain. Their models are also known as ‘GKX2020’ in later papers (Gu et al., 2021). They found that the MLP model with 3 hidden layers with the fixed pyramid structure (32,16,8 neurons in each hidden layer respectively) has the highest out-of-sample average fit (R^2). The study also managed the balance of computational costs of ML algorithms and the predictive fit of the ML models, which gave a taste of ML application on asset pricing. It is a great work that inspires practitioners to use machine learning techniques with an applicable computational cost to improve their factor investing efficiency and profitability. Building on this seminal work, subsequent studies have further extended ML applications in asset pricing by incorporating alternative architectures and penalty functions (Bagnara, 2022; Giglio et al., 2022; Nagel, 2021; Wang, 2021).

Their research sparked my curiosity to pursue research on ML-based factor investing strategies and to explore the application of deep MLP models in the asset pricing domain. Specifically, how to efficiently use the ML technique to solve the ‘dimensional disaster’ of the ‘factor zoo’ and increase the predictability of the factor models via MLP structures. The main research questions in this chapter are:

- ‘Is that ML model for factor investing under suspicion of pure data mining without any economic or financial meanings, or easy to be trapped by data snooping?’
- ‘Is there any benefit for using the firm characteristics sorted portfolio factors instead

of individual firm characteristics for pricing the individual stocks?’

- ‘How can we decide the suitable MLP structure instead of the fixed pre-assumed ones?’
- ‘According to the general MLP theory of “Deeper, better”, does that still hold in asset pricing models?’
- ‘Could the MLP models’ forecasting performance maintain the stability under the extreme market conditions (e.g. extreme market fluctuation caused by COVID-19)?’

To answer these questions, the classic MLP models with dynamic structure are examined on the firm characteristics- sorted portfolio factors. As one of the developers of the GKX2020’s work, this study attempts to upgrade four aspects of the GKX2020’s MLP models. 1) increasing the economic meanings of the MLP structure for factor asset pricing models. It employs 182 factors organized by firm characteristic-sorted returns from Andrew Y.; Zimmermann (2020). They serve as predictors, or features in ML terminology, to replace GKX2020’s approximately 900 predictors, which extended from the 94 firm characteristics and 8 macroeconomic indicators via Kronecker product, and industry dummy variables. 2) I select 420 large capital stocks with the assumption of ‘going concern’ from NYSE and NASDAQ, which covers 21.38% of aggregate market capitalization. Concretely, the ‘going concern’ applies, as data are available for a full testing period to circumvent the systematic risk of abrupt delisting, which could be noticeably harmful for practitioners. 3) employing the dynamic MLP structure introduced by Coqueret and Guida (2020), where the MLP structure varies according to its depth. 4) Comparing MLP models’ performance under consideration of the COVID-19 effect to further evaluate the model suitability and investment performance for factor models.

The main contributions of this chapter appear in proposing the innovative MLP structure for asset pricing factor models, moderating the overfitting issue that existed in GKX2020 models via the firm characteristic-sorted factors, introducing a novel factor investing strategy based on classic MLP models, exploring the truth that how the COVID-19 affects the model fitness and factor investing performance on the large-cap U.S. stocks and verifying some results of the previous factor investing literature.

This chapter contains six main sections. Section 2.2 presents the literature review. Section 2.3 presents the data that have been used, while Section 2.4 discusses the models employed in this chapter. Section 2.5 shows the empirical experiment and results, and Section 2.6 gives the conclusion and further discussion.

2.2 Related works

Before Treynor (1961); Sharpe (1964); Lintner (1965); Mossin (1966) created the CAPM model, people had fuzzy notions about how the risks could affect the expected returns of assets. He proposed a single-factor linear model for interpreting expected stock returns via a factor generated from the difference between expected market return and the risk-free rate. Specifically, it can be written as:

$$E(R_i) = R_f + \beta_i (E(R_m) - R_f) \quad (2.1)$$

Where $E(R_i)$ is the expected return of asset i , R_f is the risk-free rate, β_i is the asset i 's factor loading, representing its sensitivity to the market factor, and $E(R_m)$ is the expected return of the market. Although it is the cornerstone of asset pricing theory, people cast doubt on the market factor's explainability for individual assets or portfolios. Thus, Ross (1978) insists asset returns were driven by the combined effect of factors, hence proposed a generalized factor model's framework:

$$R_i = \alpha_i + \beta_{i1}F_1 + \beta_{i2}F_2 + \cdots + \beta_{ik}F_k + \epsilon_i \quad (2.2)$$

Where F_1 to F_k are factors, β_{i1} to β_{ik} are factor loadings which measure the sensitivity of the factors. α_i is the intercept of the linear model, which represents the pricing error (namely, abnormal return for investors or anomaly in the empirical asset pricing context), and the residual ϵ_i represents the firm-specific risks. This theory suggests that if pricing errors exist, investors can take advantage of the existing market arbitrage opportunities. It is known as the Arbitrage Pricing Theory (APT). The profit from the arbitrage is equal to α_i , which

means that the returns cannot be explained fully by the factors. The model frame relaxed the single-factor assumption of CAPM, thereby improving the model's flexibility. However, with such a frame, scholars still struggle with questions such as what factors should be present and how many factors should appear in a single frame. The most well-known multi-factor model is the Fama-French 3-factor model (FF3) (Fama and French, 1993). It deployed the market factor together with size and value (book-to-market ratio) factors to the APT frame, which improved the explanation of the sorted portfolios. The FF3 model led to an increased enthusiasm for research on factor exploration afterwards. Researchers developed new factors and models based on the APT framework, and had the massive competitive models on their innovative factors and factor combinations, which pursue the insignificant α for proving a better interpretability of the factors. For example, Carhart (1997) extended FF3 to 4 factors by adding the momentum factor, while Novy-Marx (2013) proposed adding a gross profit factor to the FF3 frame. Further development led to the Fama-French 5-factor (FF5) model (Fama and French, 2015), which added profitability and investment factors to the FF3. Hou et al. (2015) optimized the FF5's investment factor by Tobin's Q theory (Tobin, 1969), which utilized the ratio of the market value of the company and replaced the cost of capital to measure the quality of investment. They offered great ideas for interpreting asset or portfolio returns. However, with continually emerging factors and countless combinations among them, it remains a key research challenge to assess the reliability of different factors. There are 400+ factors that play a role in stock pricing (Hou et al., 2020). Handpicking factors may cause the 'omitted factor' issue, where important variables are intentionally or unintentionally neglected. However, inserting all factors in the unpenalized linear APT frame causes the 'dimensionality disaster' since it leads to problems, such as overfitting and instability of estimation, which cause the false factor detection and enlarged pricing errors.

Therefore, some researchers devoted themselves to developing non-linear model forms or adding constraints to the linear model form to satisfy the increasing factor size. Stock and Watson (2002) introduced the use of principal components analysis (PCA) to extract factors from a large number of original input factors. PCA projects the predictors into the new coordinate system via a linear transformation and reorganizes the predictors into compo-

nents. It extracts components that explain the most variance as the new factors for the APT frame. The entire process is named principal components regression (PCR), and the predefined parameter, which indicates how many new factors should be retained, is known as the hyperparameter of PCR. PCA is the most widely applied technique for dimension reduction tasks in different domains, but it has limitations, such as the linear combination assumption of factors, and new factors subtracted from the original factors may not be easy to interpret and require normalization. Later, Kelly et al. (2019) utilised the instrumented PCA (IPCA), which was upgraded from the PCA for factor extraction. IPCA employs a dynamic process on PCA that analyses only a small batch of the dataset each time and updates the factors gradually, ultimately converging on the optimal factors for the full dataset. While IPCA optimizes the extraction process for large-scale factors through incremental updates, it does not resolve PCA's core theoretical weaknesses. Concretely, the extracted latent factors remain linear combinations of the original predictors, and the model still struggles with low interpretability and the loss of information that does not align with maximum variance. In contrast, the Neural Network approach, MLP models, proposed in this chapter, addresses these limitations by employing non-linear activation functions (e.g. ReLU) for a better mapping of original factors. This allows for a more flexible dimensionality reduction process that captures complex factor structures, therefore minimizing the loss of financially or economically significant information that traditional linear methods usually overlook.

Furthermore, the Least Absolute Shrinkage and Selection Operator (LASSO) and ridge regression are two commonly applied constraints for linear models, known as L1 and L2 regularization, respectively. They are employed for controlling the oversized factors in literature. Kelly and Pruitt (2013) proposed a new pricing method via market expectation, which relies on LASSO for noise reduction. Feng et al. (2020) introduced the concept of 'factor zoo' and applied an LASSO-based examination method. Chinco et al. (2019) has also verified LASSO's ability to denoise datasets. They proved that LASSO regression can significantly reduce the noise created by ineffective factors and improve model fitness and robustness. Whereas the classic LASSO has limitations, such as extracting only one factor from a group of highly correlated factors, its limitation is that it cannot jointly measure multiple corre-

lated or equally important factors. This leads to information loss. Therefore, some papers turn to the constraint of classic ridge regression. For example, Giglio and Xiu (2021) applied ridge regression to the study of how investor beliefs have an impact on portfolio decisions from a survey of wealthy retail investors. It matched the survey data to investors' portfolios, trading activities, and online behaviors data. It proved that the l_2 penalty can identify which factors should be the prioritized factors, identifying potential multicollinearity and moderating model overfitting in the dataset. Nonetheless, classic ridge regression has limitations, such as it is impossible to eliminate the ineffective factors, since it never fails to shrink the coefficients to zero, and it penalizes all correlated factors when it addresses the multicollinearity, which makes it difficult to interpret the individual effect of factors with high correlation. Moreover, for sparse factors, where only a few variables have significant effects, ridge regression shows less priority than LASSO due to the effect of non-zero coefficient shrinkage. Both classic LASSO and ridge regression face the same limitations of being computationally expensive and time-consuming for searching hyperparameters, and are therefore not suitable for non-linear data and sensitive to noise. Yuan and Lin (2006); Freyberger et al. (2020); Bertelsen (2022) later extended LASSO to group LASSO, adaptive group LASSO and prior adaptive LASSO, respectively, for moderating the aforementioned issues of classic LASSO, but it still has not been fundamentally resolved.

Fortunately, the development of machine learning (ML) methods has led to the progression of the asset pricing factor model structures. The earliest neural network (NN) models for financial applications can be traced back to the 1990s, where Hutchinson et al. (1994) used a new non-parametric NN model for simulating Black-Scholes option prices. They proved that the model recovered the Black-Scholes formula from a two-year training set of daily option prices and that it can be used on both out-of-sample price and delta-hedge options, which gave hope to subsequent research that NN models may overcome some limitations of traditional econometric methods. Later, Heaton et al. (2016) tested classic deep NN structures on financial time series prediction tasks, from MLP to Autoencoder, and found that deep neural network structures significantly improved the performance of financial series prediction. Whereas, unlike financial data, which includes higher frequency options, such as daily

data and hourly data with a larger number of observations, stock pricing typically employs monthly data, which aligns with the release frequency of financial reports. Thus, the limited data size is a challenge for the research of stock pricing via ML.

The seminal work of asset pricing in ML models is Gu et al. (2020). They examined classic ML methods, including MLP, decision trees, and random forests, on 30,000 U.S. stocks over 60 years, from March 1957 to December 2016. The work deployed 94 firm characteristics (Green et al., 2017), 8 macroeconomic indicators (Welch and Goyal, 2008) and 74 industry dummy variables as factors pricing on individual stocks' excess return series. They also applied the extending window method to their work, which means they re-estimated the parameters every 12 months after the first in-sample window. And it enlarged the factor dimension via the Kronecker product method to ensure the data size for the ML frameworks. The final factor dimension for each stock is approximately 850-900. Their fixed-neuron pyramidal MLP structures borrowed the idea of Masters (1993). Their models in later works are also named 'GKX2020' Gu et al. (2020). They found MLP with 3-4 hidden layers outperformed all alternative benchmark models (e.g. random forest, MLP with 1-2 and 5 hidden layers and Elastic net) by comparing the average out-of-sample (OOS) fitness indicator R^2 . However, as an early work of asset pricing, it has inherent limitations: they adopted the cross-sectional rank-winsorization method for denoising the noise that exists in the firm characteristics, but the method artificially truncates fat tails, stripping the model of critical signals needed to predict extreme market events or crash risks. Also, they did not consider the transaction costs, such as variable borrowing rates for short positions and liquidity risks. From a modelling perspective, they left a question of why the simple fixed pyramidal MLP structures can be the optimal selections for stock pricing, as well as it does not present the difference between the MLP models in large-cap stocks and small-cap stocks. Inspired by their work, I apply the firm characteristics-sorted stock return portfolios as factors for moderating their noisy factor data, and examining these factors on large-cap stocks to prevent liquidity risks in practice.

By following GKX2020's footpath, Wang (2021) applied GKX2020 ML models on cryptocur-

rencies, and their empirical results agree with the findings that GKX2020 MLP models, even with different asset categories and different configurations of factors. This extended investigation proved that MLP models generally improve the model fitness and forecasting performance in the context of asset pricing. However, they emphasise the excellence in projecting the factors' nonlinear influence on asset prices, but provide no backtesting to further support investment practice. Gu et al. (2021) extended the MLP models of GKX2020 by adopting an MLP autoencoder architecture for dimensionality reduction of factors, which was inspired by Kelly et al. (2019)'s instrumented principal component analysis (IPCA) model. Concretely, they employed a dual-MLP structure for pricing the U.S. stock returns, which was used in Gu et al. (2020). In their model structure, the left side is an MLP autoencoder (conditional autoencoder) structure, which maps high-dimensional lagged firm characteristics to lower-dimensional factor loadings, while the right side is another MLP autoencoder extracting the information from firm characteristic-sorted portfolios as input factors and reducing the dimension of these factors. They linearly mapped the outputs of these two MLP autoencoders to price the individual stock returns. Although their model achieved an annualized value-weighted Sharpe ratio of 1.53 in long-short value-weighted portfolio back testing, the predictive power of the model was nearly identical to pure machine learning prediction models that do not assume a factor structure, even if their model improved the model's economic interpretability in the ML context. Moreover, their model doubled the parameter size, which increases the risk of model overfitting and unnecessary computational costs. Thus, the proposed model adopts a simplified dynamic structure to improve the computational efficiency and moderate the overfitting risk.

Furthermore, Avramov et al. (2021) provided the first large-scale examination of whether ML signals in stock-return prediction retain economic significance when realistic investment frictions are imposed. By comparing four prominent ML approaches: Gu et al. (2020)'s three-layer neural network (GKX2020), Chen et al. (2024)'s adversarial deep-learning framework for stochastic discount factor estimation (CPZ), Kelly et al. (2019)'s IPCA model, and Gu et al. (2021)'s conditional autoencoder (CA), they constructed value-weighted long-short portfolios from 1987 to 2017. Although the unrestricted full-sample strategies deliver

strong raw and risk-adjusted returns (e.g., FF6 alphas of 0.92% to 1.87% per month), imposing common economic restrictions, excluding microcaps, non-rated firms, or stocks around credit-rating downgrades, dramatically attenuates profitability, especially for deep-learning methods; many α s lose statistical significance. They further document extremely high portfolio turnover (87% to 163% monthly), extreme positions in the tangency portfolio, and the concentration of gains in high limits-to-arbitrage states, yet they note that ML signals remain profitable in long-only positions, recent years, and crisis periods, and are economically interpretable as they largely aggregate well-known anomaly characteristics. This motivates backtesting long-only portfolios in this study. The paper concluded by proposing a practical protocol for future ML research that emphasized value-weighting, transaction costs, long positions, and recent-sample performance, thereby bridging the gap between statistical predictability and implementable investment value. Similarly, Giglio et al. (2022) conducted a comprehensive literature review of ML-based asset pricing research. They organized the literature around core empirical objectives, such as estimating expected returns, latent factors and risk exposures, risk premia, the stochastic discount factor (SDF), model comparison, and alpha testing. They contrasted classical low-dimensional approaches (e.g., time series and cross-sectional regressions, Fama-MacBeth procedures, and the Gibbons-Ross-Shanken test) with ML models designed for high-dimensional individual stocks or firm characteristics. They highlighted innovations such as instrumented principal components analysis (IPCA), risk-premia PCA, conditional autoencoders, penalized SDF estimation, double ML methods for factor selection, and Bayesian model comparison, while emphasizing regularization, overfitting controls, and asymptotic theory under long time series (large T), large cross-sectional (large N), or extended panel (large T and N) regimes. The survey emphasizes how ML methods address the challenges of high-dimensional predictors, time-varying loadings, weak factors and omitted variables, and provides a rigorous guide for researchers seeking to harness these instruments for asset pricing discoveries in the background of ML. It also identifies promising directions for future work at the intersection of economic theory, statistical efficiency, and practical implementation.

In addition, Kelly and Xiu (2023) also contributed a general literature review on machine

learning in financial markets. It summarized the core challenges of asset pricing, for instance, large information sets, ambiguous functional forms, and the low signal-to-noise environment that distinguishes finance from forecasting tasks in other domains. They organize the discussion around four pillars: the case for financial ML (emphasizing prices as predictions, the benefits of economic structure amid small data and weak signals, and the complementary roles of structural hypothesis testing versus atheoretical prediction models); the virtues of complex, overparameterized models (introducing tools like ridge regression with generated features and random matrix theory to show that, under realistic mis-specification, higher model complexity improves out-of-sample R^2 , expected returns, volatility, and Sharpe ratios via the ‘complexity wedge’); return prediction (covering data construction, recursive cross-validation designs, benchmarks such as Fama-MacBeth linear models, penalized linear regressions, dimension reduction, decision trees, vanilla and sophisticated neural networks, and models for alternative data); and extensions to risk-return trade-offs (unconditional/conditional/complex factor models, high-frequency settings, and alphas) and optimal portfolios (plug-in versus integrated estimation, maximum Sharpe ratio regression, SDF estimation, and reinforcement learning with trading costs). Throughout, the authors highlight empirical gains from ML methods while stressing best practices for regularization, out-of-sample evaluation, and the integration of economic theory, offering a practical guide for researchers seeking robust, implementable discoveries at the intersection of financial asset pricing and ML.

2.3 Data Description

The responses, or labels in ML terminology, used in this research contain 420 large-capital stock excess returns from January 1957 to December 2021. These stocks come from the National Association of Securities Dealers Automated Quotations (NASDAQ) and the New York Stock Exchange (NYSE), which are derived from CRSP of Wharton Research Data Services (WRDS). These 420 stocks originally are filtered from the top 15% of the largest market capitalization (MC), which covers 85% of total MC for the two markets. They satisfy the conditions of having no missing data in the testing period and less than 50% of values

missing during the training period, hence they can meet the requirement of the ‘going concern’ (Fama, 1970) and data representability. Selecting the large-cap stocks is for avoiding the liquidity risks of market trading, which happens as investors cannot close the positions to stop loss or confirm the trading profit since no counterparty takes their positions. And as a benefit of the assumption of ‘too-big-to-fail’ (Mishkin, 2006) and ‘going concern’, this setting is more realistic for practitioners since it protects the investors from systemic risks such as companies that are delisted unexpectedly due to insolvency (e.g. Lehman Brothers) or other unexpected issues. The overall coverage of market capitalization from these stocks is 21.38% in the U.S. stock markets of the NASDAQ and the NYSE. Additionally, excessive missing values can significantly compromise model reliability. They reduce the effective sample size, enlarge estimation variance, and introduce bias if the missingness is not completely at random (Little and Rubin, 2019). In this case, missing observations may disrupt the temporal dependency structure, affect predictive performance and cause the unstable out-of-sample evaluation. Thus, I remove the factors with excessively missing values. The risk-free rate for constructing the excess returns and evaluating the Sharpe ratio (SR) and Sortino ratio (SO) is derived from the Kenneth R. French Data Library.

The predictors, also known as features for the ML terminology, are organized by 182 explored firm characteristic-sorted portfolio factors provided by Andrew Y.; Zimmermann (2020) in their Open Source Asset Pricing (OSAP) database, with at least 60% of available values in the training period. The 60% data availability threshold during the training period ensures the historical continuity of the time series from 1957 to 2021. Although the full OSAP dataset provided 202 factors that authors confirmed as high-quality factors, numerous of them, especially those labeled as ‘indirect’ or ‘insignificant’ ones, suffer from massive data gaps in the early decades. The selected 182 factors provides a high-density information set that covers nearly all ‘Clear Predictors’ across six core clusters: Momentum, Value, Investment, Profitability, Intangibles, and Trading Frictions.

There are three reasons for employing the firm characteristic-sorted portfolios as factors instead of firm characteristics directly as input factors, which is adopted by literature such

as Gu et al. (2020); Ma et al. (2023). Firstly, compared with the individual firm characteristics as factors, the sorted portfolio factors consider the effects between assets, stocks in this case, and cross-sectionally evaluate the asset excess returns, which improves the economic interpretability of the factor model. Secondly, compared with the factor method of GKX2020 (Gu et al., 2020), the factor method in this study highly reduce the dimension of the input, which potentially causes model overfitting. Thirdly, if firm characteristics are used directly as factors, the data exhibit large disparities across variables, and even after standardization, such heterogeneity remains problematic for model estimation. The details of these 182 factors are presented in the appendix of Chapter 2. The entire data set is split into training (70% of the entire data length), validation (15% of the entire data length) and testing periods (15% of the entire data length for Period 2112) as shown in Table 2.1. It is consistent with the general machine learning data split rule of thumb that the training data length should normally exceed 70% of the entire data length (Goodfellow et al., 2016). And the validation window and testing period in this case are adjusted to satisfy the research target of the COVID-19 impact investigation. The training window is on an extension base with a step size of 12 months, while the validation window is fixed with 119 observations. The training window, together with a fixed validation window, moves forward 12 months at a time until it exhausts the entire data length. Thus, the out-of-sample, which contains the full testing period, has 9 years with one update per year. This follows the convention of Gu et al. (2020), and this approach ensures all models operate on the identical information set and avoids the excessive computational workload of monthly retraining. From an economic perspective, since the input predictors are organized by firm characteristics that a substantial number of them update on a quarterly or annual basis, annual re-estimation aligns the model's learning frequency with the natural turnover of these fundamental signals. This prevents the networks from overfitting to short-term monthly trading noise that often lacks long-term predictive power. Between parameter updates, the models generate monthly out-of-sample predictions by applying the most recent estimated parameters to the updated monthly factors. Figure 2.1 shows the mechanism of the extensive window method in this chapter.

As for the robustness checking, the study compares the models' testing periods depending on whether including the COVID-19 period, which is denoted as '2112' for the testing period covering COVID-19, which contains a sharp uptrend with deep drawdowns, and '1911' for the testing period without the COVID-19 period, which contains a mild uptrend. Concretely, the training extension window and validation window are identical for both methods; the only difference is the length of the testing period. It is demonstrated in Table 2.1.

Name	Start date	End date	Observation No.
Training	1/1957	1/2003	553
Validation	2/2003	12/2012	119
Testing (OOS) for '1911'	1/2013	11/2019	83
Testing (OOS) for '2112'	1/2013	12/2021	108

Table 2.1: Data split table for initial setting of training, validation, and testing period. 'Testing (OOS) for 2112' refers to the full out-of-sample evaluation period, while 'Testing (OOS) for 1911' excludes the pandemic period from December 2019 to December 2021.

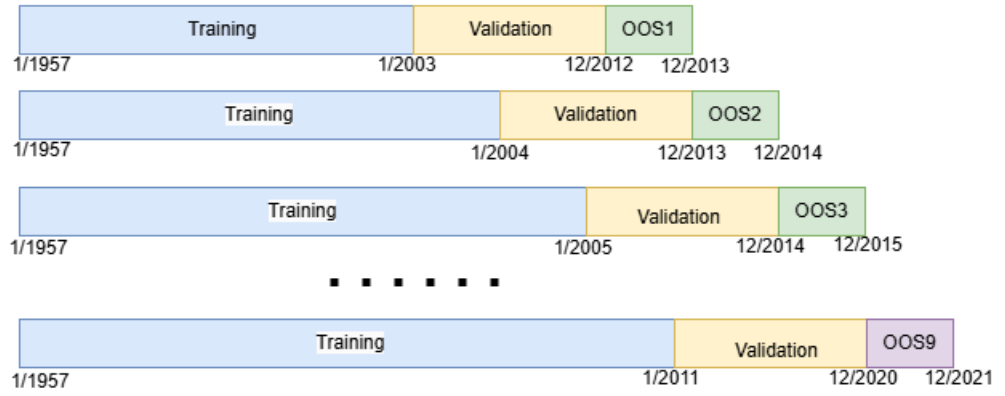


Figure 2.1: The mechanism of the extended window method. The blue bars show the sample section of the training period with an extended base. The yellow bars represent the validation period on a fixed rolling basis. The 'OOS(i)', $i = 1, 2, 3, \dots$ are the out-of-sample iterations, and each row points to one iteration. It moves forward following the timeline. The black dots between the 3rd line and the 4th line are the ellipses, which indicate that more iterations are involved.

2.4 Models

As in most ML-based asset pricing literature, it starts from the general additive prediction error model commonly used for asset pricing assignment:

$$r_{i,t+1} = \mathbb{E}_t[r_{i,t+1}] + \varepsilon_{i,t+1} \quad (2.3)$$

Where $r_{i,t+1}$ is the asset excess returns at time $t+1$. $\mathbb{E}_t[r_{i,t+1}]$ for $t \in [1, T-1]$ is the expected return, which is the ‘fair’ compensation investors demand at time t for bearing the systematic risk associated with the asset. It is determined by the asset’s sensitivity (or ‘Beta’) to broader market factors, such as the market risk premium, size, or value factors, and is calculated using all information available up to that moment. t is the time index in the range from 1 to the entire time length T . i is the asset index in the range from 1 to the entire asset number N . $\varepsilon_{i,t+1}$ is the residual (or idiosyncratic shock in asset pricing). While the expected return is the reward for staying exposed to the market’s structural risks, the idiosyncratic shock is the result of firm-specific events, such as a sudden lawsuit or a breakthrough invention, that deviate from the original forecast. Concerning the work of GKX2020 (Gu et al., 2020), the conditional expectation of $r_{i,t+1}$ can also be expressed as a function of K number of factors. If the number of factors is small, it could be suitable for traditional linear structures. Whereas if the number of factors is large, it raises the ‘factor zoo’ issue (Harvey and Liu, 2019), the additional dimension reduction process, such as principal components analysis (PCA), is required for shrinking the dimension of factors. This process is also known as ‘feature engineering or abstracting’ in ML terminology. Equation (2.4) provides the general idea of a factor model. The early literature on empirical asset pricing generally applies the ordinary least squares (OLS) linear function with fewer than 10 factors, but later, alternative non-linear function forms are applied due to the increasing number of factors. Equation (2.4) is a general function form of the asset pricing factor model in the context of machine learning, where $g(\cdot)$ represents the variable model structures in this chapter. $f_{j,t}$ is the j th factor at time t and the θ is the parameters of the factors.

$$r_{i,t+1} = g(f_{j,t}; \theta) + \varepsilon_{i,t+1} \quad (2.4)$$

2.4.1 Multilayer Perceptron Algorithms (MLPs)

From Equation (2.4), it is clear that the function form is variable for adapting the factors. In this chapter, we employ the MLP models with a pyramid structure as a substitution for the traditional linear form of the factor model. The mechanism of MLP will be presented with an example, while the application of the factor model for asset pricing will be explained in line with the work of GKX2020.

Influenced by biological brain mechanisms, vanilla neural network algorithms (alternatively multilayer perceptron) were developed in the 1960s, and a multitude of literature contributed to developing the modern concept of MLPs (Rosenbaltt, 1957; Rumelhart et al., 1986a; Werbos, 1974). A complete simple MLP structure constitutes one input layer, a single hidden layer or multiple hidden layers and one output layer. Each layer contains a single neuron or multiple neurons. Neurons are connected to the neurons from the layers next to them. Figure 2.2 demonstrates an example of fully connected MLPs with and without hidden layers. The left panel of Figure 2.2 exhibits the MLP without a hidden layer. It can

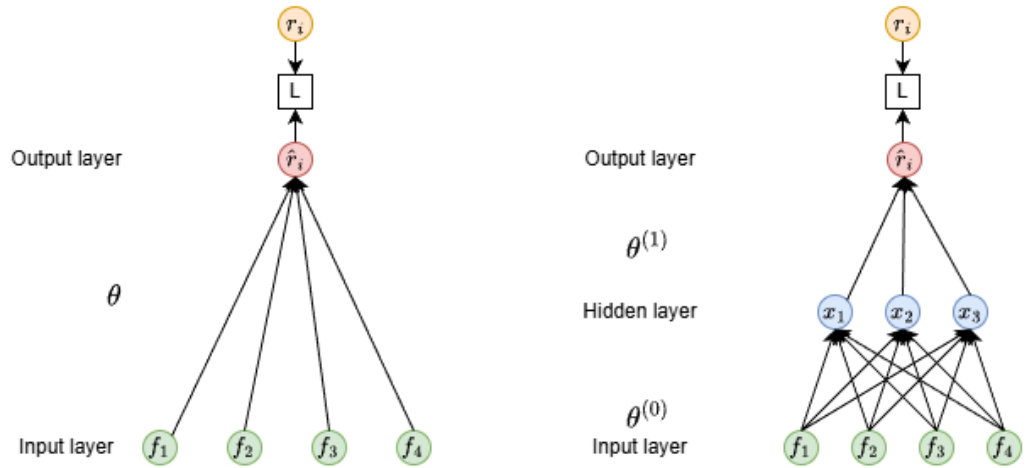


Figure 2.2: A example of simple MLP structures with (right panel) and without (left panel) hidden layers. θ are the learnable parameters for the factors $f_j (j = 1, 2, 3, 4)$ and the x_1, x_2, x_3 are values calculated from an activation function $\sigma(\cdot)$. L represents the loss function, and $\hat{r}_i, r_i$ represent the estimated and actual excess return of stock i , respectively. The layer with green neurons is the input layer, assumed with 4 factors, and the layer with 3 blue neurons is the hidden layer, which is calculated from the input factors with the activation function. The layer with one single red neuron is the output layer, which derives the estimated stock excess return $\hat{r}_i$ that is adjusted according to the actual stock excess return r_i through the loss function.

be as simple as a linear model, which can be presented as:

$$g(f_t; \theta) = \theta_0 + \sum_{j=1}^K \theta_j f_{j,t} \quad (2.5)$$

Where $g(\cdot)$ is a function form for mapping the factors to the excess returns $r_{i,t}$, f_j is the j th of the original factors. The θ_0 performs as the intercept of the linear model, and it is interpreted as the abnormal return in the context of factor investing or pricing error in the context of asset pricing. θ_j is the parameter for factor j .

As the number of factors K increases, the linear model may fail to provide reliable estimates due to its inherent limitations. Firstly, the model encounters the curse of dimensionality, where the number of parameters to be estimated approaches or exceeds the number of observations, leading to a loss of degrees of freedom and singular covariance matrices. Secondly, high-dimensional factor sets often exhibit severe multicollinearity, causing the Ordinary Least Squares (OLS) estimators to become highly unstable with inflated variances. Thirdly, the model begins to ‘fit the noise’ rather than the signal, resulting in significant overfitting, which manifests as the sharp decline in out-of-sample predictive power. Finally, a rigid linear specification is incapable of capturing non-linearities and high-order interactions between characteristics. Thus, MLP with hidden layers are the more flexible solution for the empirical asset pricing model.

The right panel of Figure 2.2 exhibits an example of a simple pyramidal MLP structure with 1 hidden layer containing 3 neurons and an input of 4 factors. The algorithm of a neuron ‘ m ’ in the first hidden layer can be presented as:

$$x_{m,t}^{(1)} = \sigma \left(\theta_{m,0}^{(0)} + \sum_{j=1}^K f_{j,t} \theta_{m,j}^{(0)} \right) \quad (2.6)$$

Where $\sigma(\cdot)$ is the activation function, $x^{(1)}$ represents the neuron m in the first hidden layer, $\theta_{m,0}^{(0)}$ and $\theta_{m,j}^{(0)}$, $j \neq 0$ are known as ‘bias’ and ‘weights’ in the context of machine learning, respectively. The number in parentheses is the sequence notation of layers, in which ‘(0)’

means the input layer and ‘(1)’ means the first hidden layer.

From the second hidden layer to the final hidden layer, the general form of mathematical presentation is shown as:

$$x_{m,t}^{(l)} = \sigma \left(\theta_{m,0}^{(l-1)} + \sum_{h=1}^{M_{l-1}} x_{h,t}^{(l-1)} \theta_{m,h}^{(l-1)} \right) \quad (2.7)$$

Where l indicates the index of the layer, $l > 1$, and M is the number of neurons in a hidden layer. Thus, M_{l-1} means the neuron number in the hidden layer $l - 1$, h is the index of neurons from the previous hidden layer. The rectified linear unit (ReLU) function is deployed as the activation function by following the work of GKX2020 (Gu et al., 2020). It shows as:

$$ReLU(x) = \max\{0, x\} \quad (2.8)$$

The economic rationale for this functional form lies in its ability to model asymmetric payoff structures and threshold-based pricing anomalies without imposing the rigid constraints of a global linear model. By nullifying negative activations, ReLU inherently introduces an internal feature (factor) selection process, which is crucial for navigating the low signal-to-noise ratio of financial data. This acceleration of convergence, coupled with the avoidance of saturation regions, enables MLP models to robustly extract non-linear predictive signals from a vast set of K input factors. Thus, it effectively improves the computational efficiency and mitigates the vanishing gradient problem during the backpropagation.

A linear output layer is assembled for the asset return value prediction context. It is mathematically presented as:

$$\hat{r}_{i,t+1} = \theta_0^{(L)} + \sum_{m=1}^{M_L} x_{m,t}^{(L)} \theta_m^{(L)} \quad (2.9)$$

Where L is the notation of the final hidden layer.

By following the work of GKX2020, the pyramid structure is applied in this chapter, which

follows the sequence:

$$\text{input dimension} > h_1 > h_2 > \dots > \text{output dimension} \quad (2.10)$$

Where the input dimension is equal to the number of features, $h(i), i = 1, 2, 3 \dots$ indicates the dimension of hidden layers. The output dimension is the number of output dimensions. GKX2020 simply used a fixed pyramid structure with $h_1 = 32, h_2 = 16, h_3 = 8, h_4 = 4, h_5 = 2$ as suggested by Masters (1993). However, the fixed structure may not be adaptive for the variable input dimensions, and Coqueret and Guida (2020) proposed a dynamic pyramid structure design method for MLP, which is:

$$h_k \approx \left\lfloor O \left(\frac{I}{O} \right)^{\frac{L+1-k}{L+1}} \right\rfloor \quad (2.11)$$

Where I is the input dimension, L is the number of hidden layers. It provides a structure with a more reasonable shrinkage of input factors. The MLP models implement the Coqueret and Guida (2020)'s method for constructing the pyramid structure, with the output dimension of one, which represents an individual stock's excess return series. A factor counts as one dimension of the input layer.

The choice of this pyramidal architecture is not merely an empirical convention inspired by the work of Masters (1993), but is grounded in two primary considerations. First, from an information-theoretic perspective, the declining neuron count functions as an informational bottleneck. Given the high dimensionality of I and the inherently low signal-to-noise ratio in financial markets, this structure forces the network to distill redundant factors into a parsimonious set of latent pricing signals, therefore filtering out idiosyncratic noise. Second, the geometric decay in h_k serves as a form of implicit regularization. By systematically reducing the parameter space in deeper layers, the model mitigates the risk of overfitting, which is a crucial failure reason for linear models when the factor number K is large. Unlike the fixed configuration in Gu et al. (2020), the dynamic approach of Coqueret and Guida (2020) ensures that the compression ratio remains proportional to the specific input dimension of the dataset, maintaining a consistent level of abstraction across different model specifications.

Thus, it is generalized for the variable dimension feature (factor) input in other economic or financial scenarios.

2.4.2 Loss Function and Model Estimation

The loss function is a function that compares the predicted stock excess return values with actual stock excess return values. By minimizing the loss function, which converges the predicted values to the actual values, the neural network models are able to detect the parameter sets for the best prediction performance. Different loss functions for different tasks. For example, classification tasks employ Cross-Entropy, Hinge, or Focal loss functions. For regression tasks, it could be Mean Absolute Error (MAE), Mean Squared Error (MSE), Huber or Mean Squared Logarithmic Error (MSLE). It is interesting that, in the mainstream of the literature on ML-based financial prediction and asset pricing domain, authors agreed to deploy the MSE as the loss function (Gu et al., 2020; Kelly et al., 2019; Freyberger et al., 2020; Bianchi et al., 2021; Chen et al., 2024). By following their pathway, the MSE with L1 regularization is adopted as the loss function in this chapter and later chapters. This loss function can be presented as:

$$\mathcal{L}_{MSE_{L1}} = \frac{1}{T} \sum_{t=1}^T (r_{i,t} - \hat{r}_{i,t})^2 + \lambda |\Theta|_1 \quad (2.12)$$

Where $\mathcal{L}_{MSE_{L1}}$ represents the loss function with L1 regularization. $r_{i,t}$ and $\hat{r}_{i,t}$ are the stock i 's actual and predicted excess returns, respectively. T is the time length of excess returns or estimated excess returns. λ is a hyperparameter named Regularization Parameter, and Θ is the estimated parameters. The term of $\lambda |\Theta|_1$ represents the L1 regularization. This configuration also follows the work of GKX2020 (Gu et al., 2020). Compared to L_2 regularization (Ridge), L_1 regularization (Lasso) is uniquely suited for asset pricing due to its ability to induce sparsity and perform automated variable selection. While L_2 merely shrinks coefficients toward zero, the L_1 penalty can force coefficients of irrelevant predictors to be exactly zero. This is essential for navigating the 'Factor Zoo' as it effectively filters out noise and identifies the most parsimonious set of factors. Consequently, L_1 enhances model

interpretability and provides more robust out-of-sample predictions by mitigating the risk of overfitting in high-dimensional financial datasets.

Intuitively, since the data used in this chapter is effectively mitigated via data pre-processing (e.g. missing value filtering, normalization), and factors are structured by firm characteristic-sorted portfolios instead of firm characteristics, this naturally reduces the noise of the data, thereby the advantages of robustness for alternative loss functions such as Huber loss or MSLE are insignificant (Gu et al., 2020). Additionally, from the statistical perspective, MSE targets the conditional mean $\mathbb{E}_t[r_{i,t+1}]$, aligning with the expected-value framework of asset pricing theory, whereas MAE targets the median. This means MSE is a smooth, second-order differentiable function with a gradient proportional to the error size, which ensures stable convergence by allowing the model to decelerate near the optimum, and avoiding the numerical oscillations inherent in the constant-gradient MAE.

Unlike traditional statistical models such as OLS, which typically rely on in-sample fitting and out-of-sample (OOS) testing, ML frameworks introduce a validation period to facilitate automated model selection and hyperparameter tuning. While the non-linear architecture of the MLP inherently relaxes classical assumptions like linearity and homoscedasticity, the validation set is crucial for preventing overfitting in such highly parameterized models. Specifically, it utilizes Stochastic Gradient Descent (SGD), or its advanced variants like the Adaptive Moment Estimation (Adam) (Kingma and Ba, 2014), to minimize the model's parameters on the training data, simultaneously monitoring performance (the lowest MSE) on the validation data (simulation OOS dataset) to determine the best model for testing data (actual OOS dataset).

Equation (2.13) and Equation (2.14) shows the GD process, where $\nabla_{\theta}\mathcal{L}(\theta)$ gradient, it is the partial derivative of the loss function and the individual model parameter, θ are the model parameters. φ is the pre-set hyperparameter named learning rate, which determines how fast or accurately to find the parameters that let the loss function reach the global minimum. Normally, when $\nabla_{\theta}\mathcal{L}(\theta)$ is approaching zero, the loss function is considered as 'reaching the

global minimum’.

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) = \left(\frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \theta_1}, \frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \theta_2}, \dots, \frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \theta_n} \right) \quad (2.13)$$

$$\boldsymbol{\theta} := \boldsymbol{\theta} - \varphi \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) \quad (2.14)$$

Concerning the computational cost of ML algorithms, the most prevailing parameter searching process is the Stochastic GD (SGD) which updates parameters for each sample in each iteration. However, it introduces unexpected noise which could trigger the unstable convergence of the loss function. That is the reason for adding the optimizer of Adaptive Moment Estimation (Adam) (Kingma and Ba, 2014) for stabilizing the SGD process. Adam introduced the first and second-moment estimation of gradients to achieve the acceleration of the loss function convergence and the self-adaptation of the learning rate. The pseudocode of Adam described in the appendix of Chapter 2.

The backpropagation algorithm is the algorithm for adjusting the weights in the network by calculating the gradient of the loss function with respect to each parameter from the output direction. It applies the “chain rule” of derivative computation to propagate gradients from the output layer backwards through the network. The process can be separated into five steps: obtaining results from forward propagation, computing the loss, backpropagation, updating weights, and iteration. The mechanism of the MLP backpropagation is exhibited in the appendix of Chapter 2.

2.4.3 Early Stopping and Batch Normalization

One parameter searching for all training data points in a batch is defined as one epoch. Epoch is a one-time gradient descent process. Normally, when the epoch number increases, the MSE of training and validation will decrease. However, as epochs are a hyperparameter of the MLP model, it is difficult to define before attempting the model. Oversize epochs will cause the overfitting issue as well because the difference of MSE for both training and validation data increases after epochs reach a critical minimum value. Early-stopping is a

process for stopping the epochs when the performance on the validation data cannot be improved further, then logging the parameters for prediction.

Batch normalization is a vertical normalization, where each observation in the batch removes the batch's mean and is divided by its standard deviation; the training data in one batch's mean is zero, and the variance is equal to 1. Batch normalization can accelerate and stabilize the training process, moderate the issue of gradient vanishing and explosion (Glorot and Bengio, 2010), which is explained in Section 2.4.4, and reduce the sensitivity towards the initial weights.

2.4.4 Gradient Vanishing and Explosion

Gradient vanishing usually happens during the backpropagation process. When the hidden layer number of the MLP model increased to a certain level, the gradient was extremely small or equal to zero. Subsequently, when it is passed back to the first hidden layer, the gradient vanishes, causing the parameters to pause updating, leading to poor performance of the model. This was especially prevalent when the activation function was using Sigmoid or Tanh. Whereas ReLU is the activation function which can prevent this issue since its derivative is most likely to be 1, batch normalization is another approach that also helps with this. Opposite to the scenario of gradient vanishing, if the initial learning rate is not properly set or an inappropriate depth to the MLP structure is deployed, the gradient could be extremely large. In that sense, if the learning rate causes excessively large parameter updating, the parameters do not converge on the loss function to the global minimum but diverge from the global minimum. This phenomenon is known as gradient explosion, and the loss function encountering gradient explosion tends to be infinite. Testing the initial learning rate and batch normalization can be applied as adjustment methods for this issue.

2.4.5 Benchmark Models

Five proposed dynamic MLP models are implemented as the function forms of the empirical asset pricing model. They are compared with six benchmark models: Ordinary Least Squares (OLS), Partial Least Squares (PLS), Principal Component Regression (PCR), and GKX2020 with 3, 4, and 5 hidden layers, respectively. GKX2020 MLP models as benchmark models utilizes their original predictor configurations. The GKX2020 MLP models are similar to the proposed MLP models, but with fixed pyramidal neuron structures. In accordance with Equation (2.4), the linear benchmark models are defined by their specific functional forms of $g(f_{i,t}; \theta)$, which include a constant intercept α to capture the mean return not explained by the factors.

The OLS benchmark assumes a global linear mapping with an intercept α :

$$g(f_{i,t}; \alpha, \theta_{OLS}) = \alpha + f_{i,t}^\top \theta_{OLS} \quad (2.15)$$

where the parameters α, θ_{OLS} are estimated by minimizing the sum of squared residuals over the training sample. The intercept α represents abnormal return or pricing error in the context of factor investing and asset pricing, respectively.

PCR (Massy, 1965) applies dimensionality reduction through an unsupervised learning framework, where the predictive function $g(\cdot)$ is defined as a linear mapping of the top K principal components:

$$g(f_{i,t}; \alpha, \theta_{PCR}, \Omega_K^{PCR}) = \alpha + (f_{i,t}^\top \Omega_K^{PCR}) \theta_{PCR} \quad (2.16)$$

The projection matrix $\Omega_K^{PCR} = [v_1, \dots, v_K]$ consists of the K eigenvectors corresponding to the largest eigenvalues of the factor covariance matrix $\Sigma_{ff} = E[(f_{i,t} - \bar{f})(f_{i,t} - \bar{f})^\top]$, where $v_1, \dots, v_K$ are the loading vectors, and $\bar{f}$ is the mean of the factors. The intercept α represents the same economic meaning as the one in OLS. The j -th principal direction v_j is

determined by maximizing the variance of the projected factors:

$$v_j = \arg \max_v \text{var}(f_{i,t}^\top v) \quad \text{s.t.} \quad \|v\| = 1, v^\top v_m = 0 \text{ for } m < j \quad (2.17)$$

Unlike PCR, the PLS model (Wold, 1966) utilizes a supervised dimensionality reduction framework, where the predictive function is defined as:

$$g(f_{i,t}; \alpha, \theta_{PLS}, \Omega_K^{PLS}) = \alpha + (f_{i,t}^\top \Omega_K^{PLS}) \theta_{PLS} \quad (2.18)$$

In contrast to PCR, the PLS model identifies the latent space Ω_K^{PLS} by accounting for both the variance of factors and their predictive relationship with future stock excess returns $r_{i,t+1}$. Specifically, the j -th direction w_j in Ω_K^{PLS} is determined by maximizing the squared covariance between the linear combination of factors and the target returns:

$$w_j = \arg \max_w \text{cov}^2(f_{i,t}^\top w, r_{i,t+1}) \quad \text{s.t.} \quad \|w\| = 1, \text{cov}(f^\top w, f^\top w_m) = 0 \text{ for } m < j \quad (2.19)$$

The inclusion of the intercept α ensures that the covariance maximization is performed on the innovations of the factors, and prevents systematic mean levels from biasing the orientation of the latent directions.

2.4.6 Model Performance Evaluation

In this section, the proposed models and benchmark models are compared from two angles: model performance (predictive power) and backtesting performance. Model performance is measured by the mean of 420 stocks' the out-of-sample (OOS) coefficient of determination (R_{OOS}^2), mean square error (MSE), residual α and its statistics, which are shown in Table 2.3. The algorithms of these indicators are presented in Equation (2.20) to Equation (2.22). The research in this chapter still employs the one-step-ahead forecasting method.

$$R_{\text{oos}}^2 = 1 - \frac{\sum (r_{i,t+1} - \hat{r}_{i,t+1})^2}{\sum (r_{i,t+1} - \bar{r}_{\text{in}})^2} \quad (2.20)$$

$$MSE = \frac{1}{TN} \sum_{t=1}^T (r_{i,t} - \hat{r}_{i,t})^2 \quad (2.21)$$

$$\alpha_{avg} := \frac{1}{N} [E(r_i) - E(\hat{r}_i)] \quad (2.22)$$

Where $r_{i,t+1}$ are the real out-of-sample excess returns, $\hat{r}_{i,t+1}$ is the predicted excess returns, which is equal to $\hat{y}_{t+1}$ in Section 3.4 and $\bar{r}_{in}$ is the mean of the excess returns in the training and validation (in-sample) period. i indicates the stock i , and TN is the time length observation number multiplied by the number of stocks. Here, α denotes the OOS average estimation error in model evaluation, but it has a specific meaning in the asset pricing context, as introduced in the general introduction section.

In the traditional asset pricing concept, α is the intercept of a linear factor model. It means the risk-adjusted return of a stock. If the $\alpha = 0$, the stock excess return is perfectly explained by the selected or constructed factors; if $\alpha > 0$, it indicates the extra capital gain exists after controlling the risks brought by factors, which is also known as the ‘arbitrage opportunities’; if $\alpha < 0$, it implies the stock underperforms the model’s prediction. The latter two circumstances indicate the existence of pricing errors, but practitioners chase the positive α as the abnormal returns beyond the factors. In traditional asset pricing literature, those with a linear model frame, factor exploration research pursues α approaching zero to verify that their factors can perfectly explain the asset returns, which satisfies the non-arbitrage hypothesis. That is slightly different in ML-based asset pricing literature, especially in this case. The linear output layer still makes the asset pricing model explainable, but forces the restriction of the non-arbitrage condition by removing the intercept term of α . The computation of α for the ML-based asset pricing is also known as residual α , which is employed by Gu et al. (2020); Ma et al. (2023) and is shown in Equation (2.22). That means if the assumption stands for the linear output layer form, the mean of both pricing error α and model bias $\mathbf{b}$ should be equal to zero; otherwise, it implies that the α , the abnormal return or the arbitrage opportunity exists, or there are unexplored factors. Exploring the structure of α , how to minimise α , and the relation to the bias of the ML frame are interesting directions for future research. In that sense, the residual α includes the traditional α and the risk premium of

the linear and non-linear model structures.

The Diebold-Mariano (DM) (Diebold and Mariano, 1995) test with the HAC estimator (preventing loss differential series autocorrelation causes spurious statistical significance) is adopted for evaluating the difference between models in this study. Equation (2.23) to Equation (2.26) presents the mechanism of the DM test. DM test here using the Mean Absolute Error (MAE) as the foundation for its robustness compared with the MSE option. MAE is less sensitive to outliers than MSE, which is particularly suitable for financial time series in AI models that contain more complicated and variable structure in their prediction errors (e.g. non-Gaussian, heavy-tailed, etc.). Therefore, MAE-DM here reflects typical difference forecasting performance and avoids being affected by a few large deviations, which causes false detection of the difference between models. Hence, MAE-DM statistics provide a more realistic and reliable assessment of model superiority in this study.

$$d_{t+1} = \frac{1}{h} \sum_{i=1}^h \left(|e_{i,t+1}^{(m)}| - |e_{i,t+1}^{(n)}| \right) \quad (2.23)$$

$$\bar{d} = \frac{1}{T} \sum_{t=1}^T d_{t+1} \quad (2.24)$$

$$DM_{\text{statistics}} = \frac{\bar{d}}{d_{\text{standard error}}} \quad (2.25)$$

$$d_{\text{standard error}} = \sqrt{\widehat{V}_{HAC}/T} \quad (2.26)$$

where $|e_{i,t+1}^{(m)}|$, $|e_{i,t+1}^{(n)}|$ are the absolute forecasting errors derived from $(r_{i,t+1} - \hat{r}_{i,t+1})$ of model m and n respectively. i indicates the stock i . h is the number of stocks, which is 420 in this case. d_{t+1} is the cross-sectional average difference between model m and model n 's absolute forecasting errors, $\bar{d}$ is the mean of d_{t+1} . The denominator, $d_{\text{standard error}}$, is the HAC-consistent standard error of the loss differential series $\{d_t\}$. $\widehat{V}_{HAC}$ denotes the heteroskedasticity and autocorrelation consistent (HAC) variance of the loss differential series d_{t+1} .

For a higher economic and financial interpretation of the model, this study also calculates the factor importance for the MLP autoencoder abstracted factors and the correlation between

the abstracted factors and the original factors. Mathematically, the permutation variable importance (VI) can be exhibit as:

$$L_{\text{baseline}} = \mathcal{L}(r_{i,t}, f(X)) \quad (2.27)$$

$$L_{\text{perm}}(X_i) = \mathcal{L}(r_{i,t}, f(X_{\text{perm}})) \quad (2.28)$$

$$\text{Importance}(X_i) = L_{\text{perm}}(X_i) - L_{\text{baseline}} \quad (2.29)$$

where L_{baseline} is baseline loss or original out-of-sample loss, X is the original feature matrix, and $f(\cdot)$ is the model forms. $L_{\text{perm}}(X_i)$ is the loss that an individual feature X_i is permuted. Then, the importance of X_i is equal to the difference between the permuted loss and the baseline loss.

This study uses the trend-following sign signal trading strategy as the market timing approach, and the transaction cost is configured as 50 base points via a traditional static transaction cost calculation method (Jegadeesh and Titman, 1993; Lesmond et al., 2004; Grundy and Martin, 2001), and employs a 20 bps (Frazzini et al., 2018; Novy-Marx and Velikov, 2016; Hou et al., 2020) concerning turnover rate (shows in Equation (2.34)) as a robustness examination for exploring how transaction cost and turnover rate affect the backtesting performance of the models in this chapter. For measuring the backtesting performance, it follows the traditional asset pricing indicators, which are annualized returns, Sharpe ratio, Sortino ratio and maximum drawdown (MDD). Equation (2.30) to Equation (2.33) presents the calculation of these indicators.

$$\text{Annualized Return} = \left\{ 1 + \left[\prod_{i=1}^n (1 + r_{i,t}) - 1 \right]^{\frac{12}{n}} \right\} - 1 \quad (2.30)$$

$$\text{Sharpe Ratio} = \frac{E(r_p) - r_f}{\sigma_p} \quad (2.31)$$

$$\text{Sortino Ratio} = \frac{E(r_p) - r_f}{\sigma_d} \quad (2.32)$$

$$\text{Maximum Drawdown} = \max_{t \in [0, T]} \left(\frac{c_{\max}(t) - c(t)}{c_{\max}(t)} \right) \quad (2.33)$$

where n is the number of the observations, $E(r_p)$ is the expected portfolio return and r_f

represents the risk-free rate. σ_p and σ_d are the portfolio and portfolio downside standard deviation, respectively. $C_{\max}(t)$ is the highest value during time t , and $C(t)$ is the value at time t . For a better comparison of models that perform under the extreme market fluctuations caused by the COVID-19 pandemic, all results are displayed in three periods: Pre-COVID-19 Period (1911), COVID-19-Inclusive Period (2112) and Period Including COVID-19 and One-Year After (2212).

The models in this chapter were implemented in Python (v3.8) using the PyTorch deep learning framework (Paszke et al., 2019). Training was conducted on NVIDIA A40 and H100 GPUs provided by the Viking HPC cluster at the University of York. Unlike traditional central processors (CPUs) that handle tasks sequentially, GPUs are engineered for massive parallel computing, allowing the model to process thousands of financial signals and factor interactions simultaneously. This high-performance hardware was essential for research in this chapter since the ‘factor zoo’ presents a high-dimensional nature in individual stock calculations. That would be computationally prohibitive, potentially requiring weeks of processing time, on standard office equipment. Leveraged by these specialized chips, the extensive iterative optimization process becomes feasible.

2.5 Empirical application and results

This section mainly presents the empirical comparison from the model function and portfolio performance perspectives. Section 2.5.1 exhibits the empirical methods, Section 2.5.2 discusses the model-wise performance, and Section 2.5.3 shows the performance of equal-weighted and value-weighted portfolios organised via the selected 420 stocks from the Pre-COVID-19 Period (1911) and COVID-19-Inclusive Period (2112) angles.

2.5.1 Empirical application

The entire experiment follows seven steps: 1) data preparation, which is described in Section 4.3; 2) modelling; 3) prediction; 4) model evaluation and comparison; 5) market timing

(trading signal generation); 6) back testing; and 7) back-testing performance assessment.

The model structures of the main models and GKX2020 MLP models are shown in Table 2.2. The input factors are 182 selected firm characteristics-sorted portfolio factors, and the output is the excess return series of individual stocks. Thus, the input dimension is 182, and the output dimension is one. The benchmark models of PLS and PCA with a hyperparameter of 0.95 indicate the components explain 95% of the variance from the original features.

	Main MLP models	GKX2020
1_layers	15	32
2_layers	36, 6	32, 16
3_layers	56, 15, 4	32, 16, 8
4_layers	73, 25, 9, 3	32, 16, 8, 4
5_layers	87, 36, 15, 6, 3	32, 16, 8, 4, 2

Table 2.2: MLP architecture comparison between our main models and GKX2020. ‘ (i) _layers’, $i = 1, 2, 3, \dots, 5$, means an MLP model contains i hidden layers. Each number defines the number of neurons in a single hidden layer. The largest number in each table row is the neuron number of the hidden layer closest to the input layer. One neuron indicates one dimension. The smallest number in the row is the one in the hidden layer closest to the output layer.

A common trading signal detection technique is applied to generate trading signals from the actual stock excess returns and estimated stock excess returns of each stock, which is named the sign signal generation system. It is utilized for market timing. Concretely, in a stock, when the sign of the actual return at time t matches the sign of the estimated return at time t in a back-testing process, it is considered a trading signal. If both the actual return and the estimated return have a positive sign, it is time to open a long position of that stock. Investors may hold this position until the opposite trading signal appears, namely, both of them have a negative sign, which is a signal for investors to close that position. This is also known as long side trading. The short side trading is in contrast to the long side trading, which searches for the negative sign matching for opening the short position, then detects the positive sign matching for closing the position. This trading signal detection technique is commonly used in the trend-following trading strategy, which allows investors to achieve profitability by following the price trend of a stock.

All selected stocks are grouped as an equal-weight portfolio or a value-weighted portfolio, since a complete practical trading strategy includes trading individual stocks according to trading signals and deciding how many positions should be held for each stock. Since the asset allocation research direction is a bit out of scope of the empirical asset pricing research, in this study, the simple asset allocation strategies are implemented to complete the backtesting process, which is commonly conducted by the relevant literature (Gu et al., 2020). The equal-weight portfolio is to distribute equal weights to each stock included in the portfolio, and the value-weighted portfolio is to distribute the weights according to the proportion of the market capitalization of each stock. Additionally, the unstable borrowing rates and limited short side trading shares provided by brokers for short side stock trading practice are used to restrict the short side trading of the stocks and bring systematic risks to investors. To prevent the systematic risks brought by the extreme cases, the backtesting of this chapter excludes the short side trading. Following the methodology in existing literature of Jegadeesh and Titman (1993); Lesmond et al. (2004); Grundy and Martin (2001), which statically configured as 50bps. For robustness examination of how turnover influences the profitability of these models, a dynamic transaction cost is applied (turnover and 20bps for large-cap stocks), following the work of Ma et al. (2023); Frazzini et al. (2018); Novy-Marx and Velikov (2016); Hou et al. (2020), which shows in Equation (2.34). The total transaction cost (TC_t) is proportional to the absolute change in portfolio weights during rebalancing, which can be expressed as:

$$TC_t = \phi \cdot \sum_{i=1}^N |w_{i,t} - w_{i,t-}^+| \quad (2.34)$$

where $\phi = 0.002$ represents the cost rate (20 basis points), $w_{i,t}$ is the target weight of asset i at time t , and $w_{i,t-}^+$ is the weight immediately before rebalancing.

2.5.2 Model performance evaluation

As discussed in Section 2.4.6, the proposed models and benchmark models are evaluated via OOS mean squared error (MSE), OOS R^2 and DM tests. Table 2.3 shows the MSE and OOS R^2 for the main models. Benchmark models for the data period include (2112) and

exclude (1911) the COVID-19 pandemic (from November 2019 to December 2021). In both periods, we observe that all proposed models (from fw1 to fw5) significantly outperform the benchmark models. The average OOS R^2 implies that the proposed MLP model with 2 hidden layers which has 4.45% as the average OOS R^2 , outperforms all alternative models and is followed by the one with 3 hidden layers, which achieved 3.65% of average OOS R^2 , but apart from the proposed MLP model with 1,2,3,4 hidden layers in Period 2112, and the proposed MLP model with 2,3,4 hidden layers, the alternative models have negative OOS R^2 , which indicates the model overfitting issue to some extent. Model overfitting refers to a situation in which a predictive model captures not only the underlying patterns in the training data set but also the random noise. As a result, it exhibits excellent in-sample performance but poor out-of-sample performance. Deep overfitting implies the possibility of model mis-specification, which means the model form mismatch. In this sense, Model OLS, PLS, PCR and GKX2020 with 3 hidden layers may not perform as expected in this case. For the testing period, excluding the pandemic, which was from January 2013 to November 2019, the best-proposed models had the highest OOS R^2 , which are 3.36% and 2.5% for 2 and 3 hidden layers, respectively. It is consistent that the proposed MLP model with 2 hidden layers is the best-performing, followed solidly by the one with 3 hidden layers, and it maintains its advantage even during the pandemic-inclusive period (2112).

Comparing the fitness of the models in pre-COVID-19 and COVID-19-Inclusive periods, the OOS fitness of MLP models increased when the COVID-19 testing period was included. This means the MLP models present higher compatibility under extreme market conditions. On the other hand, some models are insensitive to pandemic-induced market fluctuations, particularly traditional statistical models (OLS, PCR, PLS) and deep MLP models. Figure 3.7 shows the market capital-weighted price index of 420 selected stocks, which provides an intuition of the market fluctuation during the pandemic. The grey shadow-covered period in Figure 3.7 is the COVID-19 period. Together with Table 2.3, they prove the excellence of shallow dynamic MLP models' forecasting capability during the extreme market turbulence. This finding is consistent with the GKX2020's finding that shallow MLP factor models have higher performance on stock return forecasting in the background of asset pricing.

Model	Avg_R2	Avg_MSE	Model	Avg_R2	Avg_MSE
Pre-COVID-19 Period (1911)			COVID-19-Inclusive Period (2112)		
OLS	-0.2251	0.0066	OLS	-0.2464	0.0086
PCR	-0.0380	0.0058	PCR	-0.0367	0.0074
PLS	-0.2250	0.0066	PLS	-0.2464	0.0086
GKX2020_fw3	-0.3419	0.0092	GKX2020_fw3	-0.2982	0.0118
GKX2020_fw4	-0.0331	0.0062	GKX2020_fw4	-0.0310	0.0083
GKX2020_fw5	-0.0057	0.0061	GKX2020_fw5	-0.0059	0.0081
fw1	-0.0025	0.0058	fw1	0.0183	0.0075
fw2	0.0336	0.0057	fw2	0.0445	0.0074
fw3	0.0250	0.0058	fw3	0.0365	0.0076
fw4	0.0071	0.0060	fw4	0.0106	0.0079
fw5	-0.0047	0.0061	fw5	-0.0023	0.0080

Table 2.3: Model performance comparison between Pre-COVID-19 (1911) and COVID-19-Inclusive (2112) periods. The ' $fw(i)$ ', $i = 1, 2, \dots, 5$ represents the proposed MLP models with 1 to 5 hidden layers, while ' $GKX2020_fw(i)$ ' denotes the reference models. Avg_R2 and Avg_MSE represent the average R-squared and Mean Squared Error, respectively. The values of R^2 are expressed in decimals; for instance, -0.2251 in the OLS model represents -22.51% .

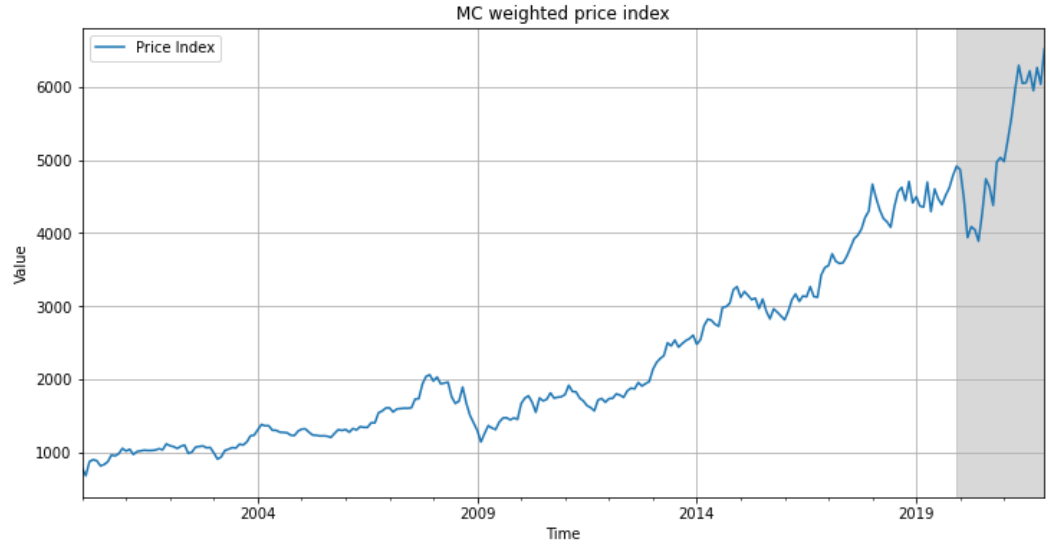


Figure 2.3: Value-weighted price index of selected 420 stocks. The grey shadow covers the COVID-19 period from December 2019 to December 2021. 'MC' means market capital.

Figure 2.4 and Figure 2.5 exhibit the distribution of OOS R^2 of the proposed MLP models and the PCR model in the Pre-COVID-19 Period (1911) and the COVID-19-Inclusive Period (2112). And Table 2.4 records the descriptive statistics of the OOS R^2 . All distri-

butions exhibit a strong right-skewed shape with a sharp peak near zero. The majority of individual stock OOS R^2 values fall within the range of -0.3 to +0.3, with very few stocks showing OOS R^2 higher than 0.5. This confirms the well-known challenge in cross-sectional asset pricing: stock return predictability is inherently weak and noisy at the individual stock level. The proposed shallow MLP models (fw1 and fw2) produce the widest distributions with the longest left tails. Model fw2, in particular, consistently shows the strongest positive OOS R^2 mass and the most heterogeneous predictions across stocks. It generates more stocks with meaningfully positive OOS R^2 (especially in the 0.1 to 0.5 range) and exhibits a moderate negative tail. This indicates that shallow networks are better at capturing differences in stock-specific predictability. The medium-depth MLP model (fw3) illustrates the transitional nature of this architecture. The distribution is sharply peaked around zero, with a high concentration of stocks near $R^2 = 0$. Compared to Model fw2, the left tail is noticeably shorter, and extreme positive or negative predictions are reduced. Model fw3 sits between the high-heterogeneity shallow models and the overly conservative deeper models. And the deep MLP models (fw4 and fw5) produce the narrowest and most concentrated distributions, with an extremely sharp peak at zero and very short tails on both sides. Predictions for the vast majority of stocks are tightly clustered around zero, indicating that deeper networks tend to generate overly smooth, conservative forecasts that lose much of the cross-sectional variation in predictability. All the phenomena indicate shallow MLP models (especially fw2) deliver more heterogeneous and informative predictions, while deeper models (fw4 and fw5) tend to “average out” signals, resulting in safer but less discriminative forecasts. Comparing the two periods, in the Pre-COVID-19 period (1911), distributions (particularly for fw1 and fw2) are relatively wider with longer left tails, and in the COVID-19-Inclusive Period (2112), all models show more concentrated distributions around zero. This suggests that the COVID-19 shock increased market noise and triggered a regime shift in pricing dynamics, making it more difficult for models to identify strong, predictable signals at the individual stock level. The PCR, as the best-performing traditional statistical benchmark, produces wider distributions than the deeper MLPs but still lags behind fw2. Shallow MLPs consistently demonstrate superior ability to generate positive predictions compared to this linear benchmark. This aligns with the findings in Table 2.3. Overall, the OOS R^2 distribution

analysis suggests that model depth significantly influences prediction heterogeneity. Shallow MLPs, especially Model fw2, yield more informative forecasts with stronger positive OOS R^2 tails, whereas deeper architectures produce overly conservative predictions clustered tightly around zero (this also applies in the GKX2020 MLP benchmarks). The COVID-19-Inclusive Period (2112) further improves predictability across the proposed MLP models, highlighting the regime-dependent nature of the forecasting. These findings reinforce the superiority of shallow MLP in this domain. In addition, the OOS MSE distribution diagrams, Figure 2.6 and Figure 2.7, agree with the findings of OOS R^2 distributions.

Furthermore, the DM test is applied to assess the level of the performance difference between the two models. Table 2.5 shows the DM test statistics and p-values-based statistical significance level. The upper panel of Table 2.5 demonstrates the DM statistics of the Pre-COVID-19 Period (1911), the lower panel reports the results for the COVID-19-inclusive period (2112). In both periods, the OLS model exhibits significantly larger forecast errors than most alternative models, confirming its inferior performance in a high-dimensional factor setting. The PCR model outperforms OLS, but it is consistently dominated by the best proposed MLP model: fw2. The PLS model performs similarly to OLS and remains inferior to most neural network specifications. The proposed MLP models (fw1 to fw5) consistently outperform the GKX2020 benchmark models, especially the proposed MLP models (fw2 to fw4), which are statistically significant. The advantage of the proposed MLP models over the GKX2020 benchmarks remains robust across both the pre-COVID and COVID-inclusive periods. Among the proposed models, fw2 and fw3 generally deliver the strongest performance in both periods, with insignificant differences between them. This pattern suggests that although increasing the depth of MLP models improves predictive accuracy, the marginal advantage diminishes after a certain level. Overall, the DM test results provide strong statistical evidence that the proposed MLP models achieve superior out-of-sample predictive accuracy compared to both traditional linear models and GKX2020 MLP models. The consistency of these findings across the pre-COVID and COVID-19-inclusive periods underscores the robustness of the proposed architecture in capturing complex nonlinear relationships in equity returns, even during periods of significant market disruption.

Pre-COVID-19 (1911)											
	OLS	PCR	PLS	GKX2020_fw3	GKX2020_fw4	GKX2020_fw5	fw1	fw2	fw3	fw4	fw5
Mean	-0.225	-0.038	-0.225	-0.342	-0.033	-0.006	-0.002	0.034	0.025	0.007	-0.005
Std. Error	0.022	0.015	0.022	0.104	0.027	0.001	0.016	0.013	0.006	0.003	0.002
Median	-0.150	-0.019	-0.150	-0.004	-0.003	-0.003	0.041	0.051	0.015	0.000	-0.003
Std. Deviation	0.455	0.317	0.454	2.123	0.558	0.019	0.327	0.259	0.130	0.061	0.037
Variance	0.207	0.100	0.207	4.506	0.311	0.000	0.107	0.067	0.017	0.004	0.001
Kurtosis	3.994	4.860	3.996	61.843	419.005	7.207	7.515	20.904	14.305	13.458	92.607
Skewness	-1.372	-1.195	-1.372	-7.703	-20.458	0.130	-2.114	-3.313	-2.024	-1.103	-6.813
Range	3.536	2.686	3.536	21.545	11.532	0.204	2.344	2.656	1.435	0.748	0.660
Minimum	-3.011	-2.132	-3.011	-21.460	-11.428	-0.100	-1.798	-2.061	-1.058	-0.489	-0.519
Maximum	0.525	0.554	0.525	0.085	0.104	0.104	0.546	0.595	0.377	0.259	0.141
COVID-19-Inclusive (2112)											
	OLS	PCR	PLS	GKX2020_fw3	GKX2020_fw4	GKX2020_fw5	fw1	fw2	fw3	fw4	fw5
Mean	-0.246	-0.037	-0.246	-0.298	-0.031	-0.006	0.018	0.045	0.036	0.011	-0.002
Std. Error	0.024	0.017	0.024	0.097	0.021	0.001	0.017	0.013	0.006	0.004	0.002
Median	-0.157	-0.006	-0.157	-0.004	-0.003	-0.003	0.072	0.058	0.020	0.001	-0.002
Std. Deviation	0.490	0.349	0.490	1.993	0.439	0.014	0.339	0.261	0.129	0.075	0.035
Variance	0.240	0.122	0.240	3.974	0.193	0.000	0.115	0.068	0.017	0.006	0.001
Kurtosis	4.781	4.716	4.780	117.900	392.707	5.764	9.909	14.585	7.469	17.430	33.292
Skewness	-1.480	-1.274	-1.480	-9.973	-19.592	-0.120	-2.450	-2.766	-1.022	-1.010	-1.291
Range	3.700	2.584	3.700	28.826	8.929	0.137	2.711	2.347	1.290	0.941	0.560
Minimum	-3.054	-1.955	-3.053	-28.777	-8.854	-0.062	-2.170	-1.814	-0.860	-0.586	-0.349
Maximum	0.647	0.629	0.647	0.049	0.075	0.075	0.541	0.533	0.430	0.355	0.210

Table 2.4: Descriptive Statistics of R^2 for Each Model. The upper panel shows the values of the Pre-COVID-19 (1911) testing period, and the lower panel shows the values of the COVID-19-Inclusive (2112) testing period.

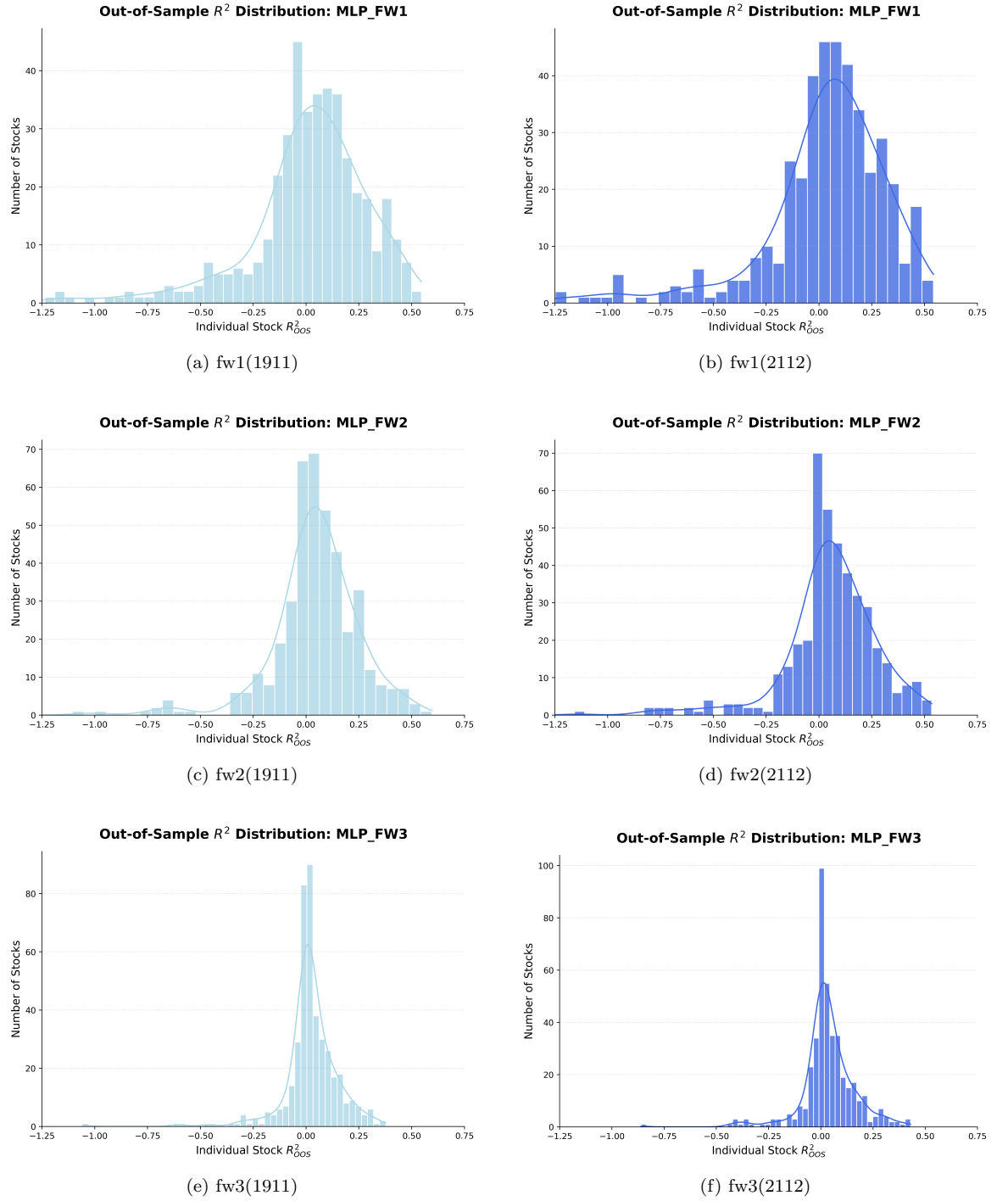


Figure 2.4: OOS R^2 distribution comparison for fw1, fw2 and fw3. The X-axis exhibits the OOS R^2 values, and the Y-axis exhibits the number of stocks. The ' $fw(i)(2112)$ ', $i = 1, 2, \dots, 5$ ' means the proposed MLP models with the COVID-19 effect, and the ' $fw(i)(1911)$ ', $i = 1, 2, \dots, 5$ means the proposed MLP models without the COVID-19 effect.

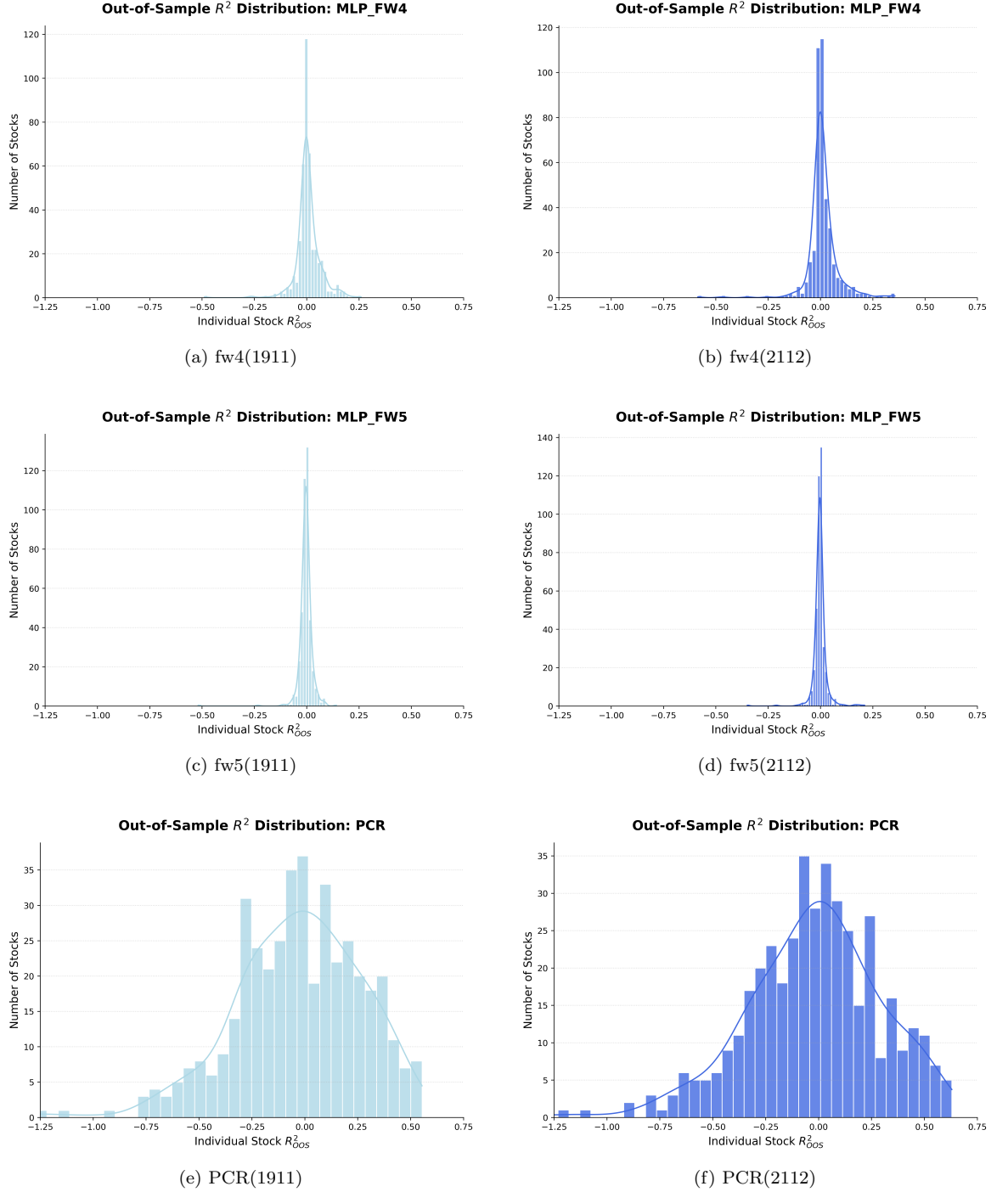


Figure 2.5: OOS R^2 distribution comparison for fw4, fw5 and PCR. The X-axis exhibits the OOS R^2 values, and the Y-axis exhibits the number of stocks. The ' $fw(i)(2012)$ and $PCR(2012)$, $i = 1, 2, \dots, 5$ ' means the proposed MLP models with the COVID-19 effect, and the ' $fw(i)(1911)$ and $PCR(1911)$, $i = 1, 2, \dots, 5$ ' means the proposed MLP models without the COVID-19 effect.

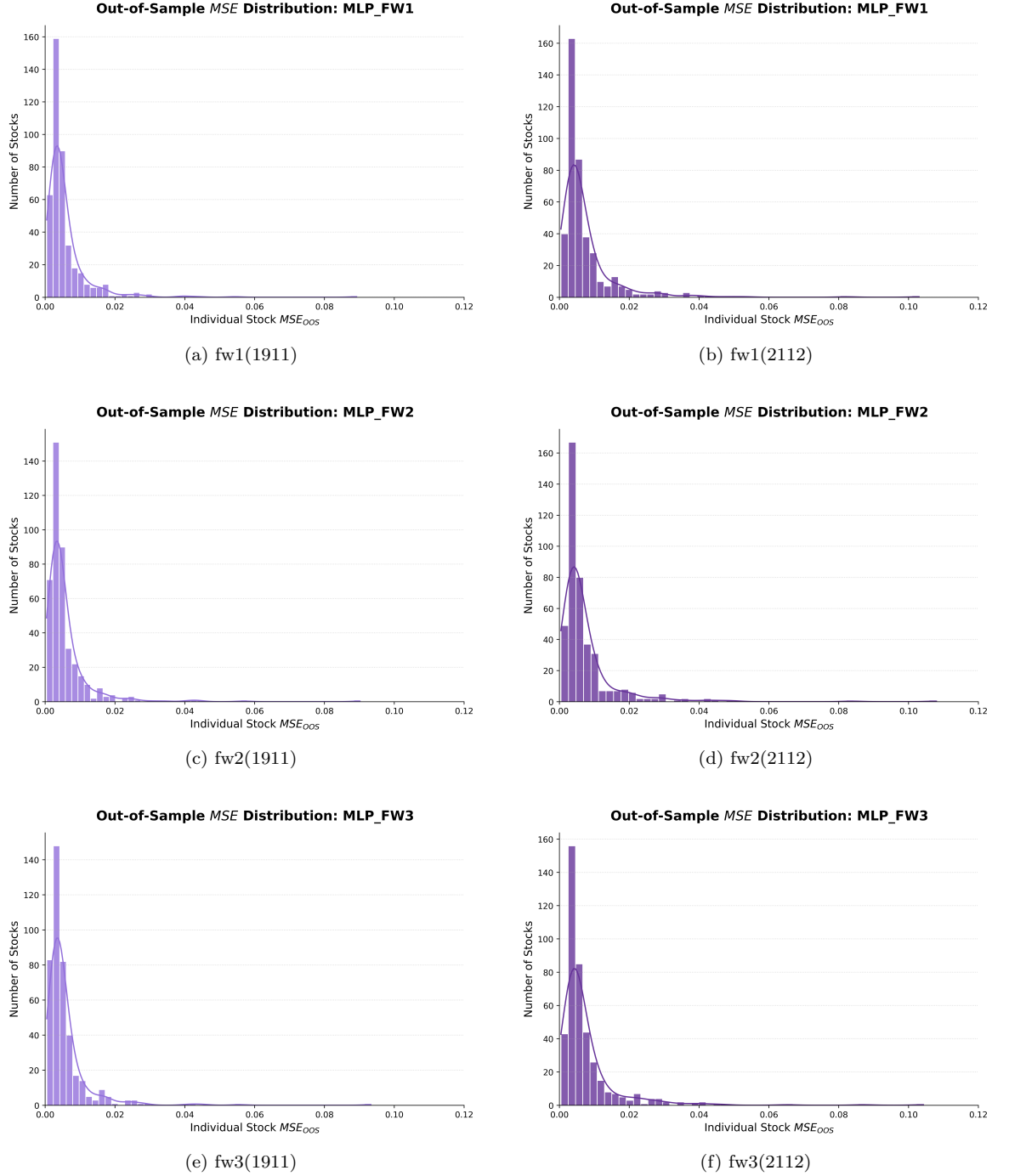


Figure 2.6: OOS MSE distribution comparison for fw1, fw2 and fw3. The X-axis exhibits the OOS MSE values, and the Y-axis exhibits the number of stocks. The ' $fw(i)(2112)$, $i = 1, 2, \dots, 5$ ' means the proposed MLP models with the COVID-19 effect, and the ' $fw(i)(1911)$ ', $i = 1, 2, \dots, 5$ means the proposed MLP models without the COVID-19 effect.

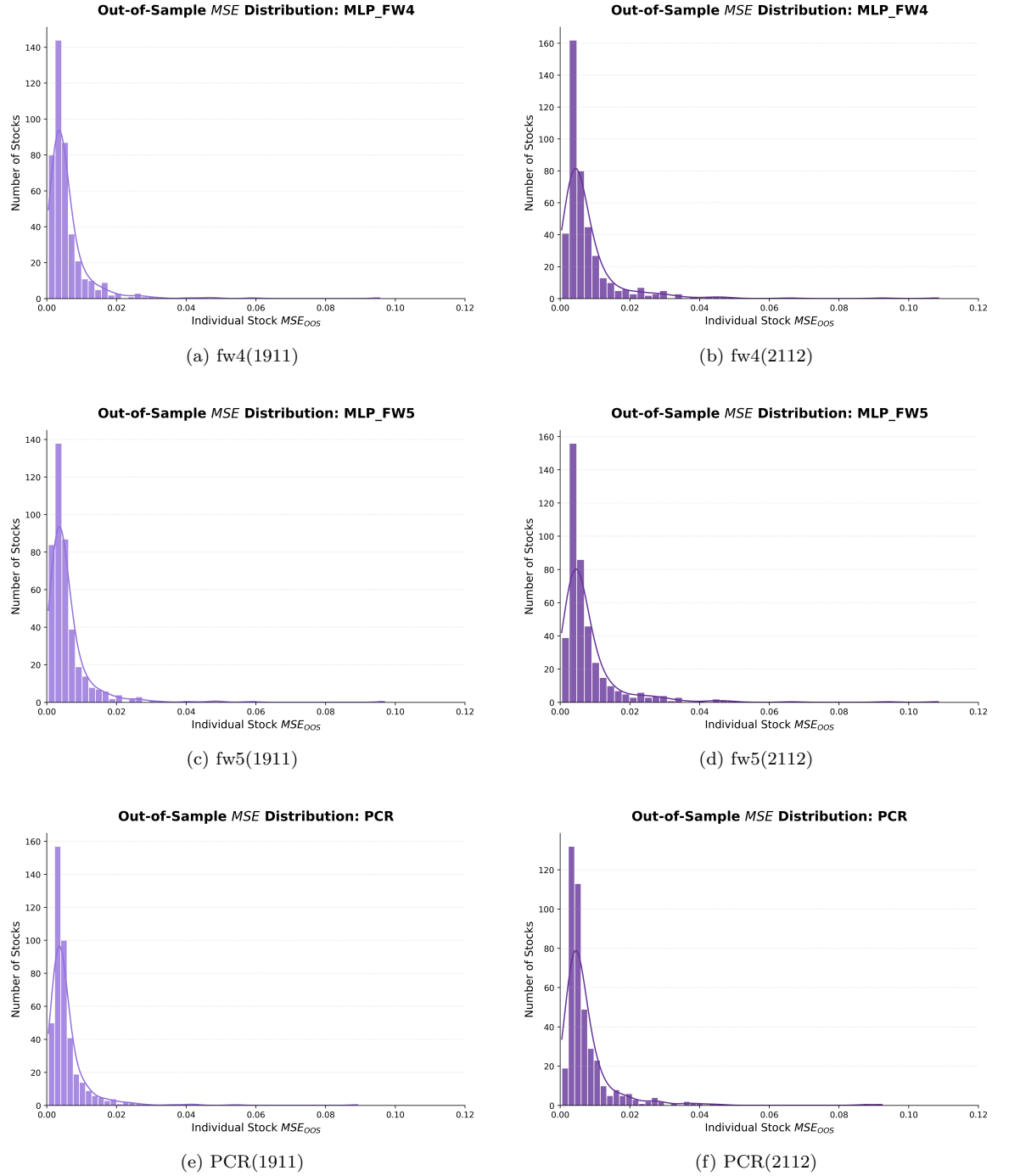


Figure 2.7: OOS MSE distribution comparison for fw4, fw5 and PCR. The X-axis exhibits the OOS MSE values, and the Y-axis exhibits the number of stocks. The ' $fw(i)(2112)$ and $PCR(2112)$, $i = 1, 2, \dots, 5$ ' means the proposed MLP models with the COVID-19 effect, and the ' $fw(i)(1911)$ ' and $PCR(1911)$, $i = 1, 2, \dots, 5$ ' means the proposed MLP models without the COVID-19 effect.

For understanding which features or factors are most important for the proposed models, variable (factor) importance examination is typically employed. As presented in Section 2.4.6, the permutation-based variable importance test is applied to the proposed MLP models for measuring the importance of the input factors. It shows that factors have strong time-varying patterns. In the pre-COVID-19 period, the importance patterns reflect relatively normal market conditions with a smooth uptrend. For Model fw2, Announcement Return (AnnouncementReturn) shows as the overwhelmingly dominant predictor, followed by short-term reversal (REV6), Accruals, and Earnings Forecast Disparity. This indicates the central role of information shocks and short-term reversal effects, with earnings quality signals, during Pre-COVID periods. In contrast, Model fw3 assigns the highest importance to Accruals. And factors of real estate (realestate), Number of Earn Increase (NumEarnIncrease), short-term reversal (REV6), and Earnings Forecast Disparity (EarningsForecastDisparity) also rank prominently. The deeper architecture thus places heavier weight on accounting-based signals and fundamental quality metrics, suggesting that additional hidden layers enable the model to capture more complex interactions among earnings management and related characteristics.

For COVID-19-Inclusive Period (2112), the factor importance shifts significantly, aligning with increased uncertainty and risk premia during the pandemic. In Model fw2, Earnings Forecast Disparity (EarningsForecastDisparity) rises to the top position, with Announcement Return (AnnouncementReturn), Leverage (BookLeverage), and Size following closely. This indicates that leverage-related risk exposures acquire substantial pricing relevance in response to the shock. For Model fw3, idiosyncratic volatility (IdioVol3F) and Size jointly dominate, accompanied by Announcement Return (AnnouncementReturn) and momentum-off-season effects. These changes highlight how crisis periods amplify the importance of volatility, size, and leverage factors, while information-related variables such as Announcement Return (AnnouncementReturn) remain robust across both architectures. This all aligns with the findings in the work of Andrew Y.; Zimmermann (2020).

	OLS	PCR	PLS	GKX_fw3	GKX_fw4	GKX_fw5	fw1	fw2	fw3	fw4
Pre-COVID-19 Period (1911)										
OLS										
PCR	-9.30***									
PLS	-1.07	9.29***								
GKX_fw3	0.06	2.66***	0.06							
GKX_fw4	-2.61***	0.61	-2.61***	-2.73***						
GKX_fw5	-2.65***	0.53	-2.65***	-2.66***	-1.27					
fw1	-8.54***	-1.49	-8.55***	-3.23***	-1.38	-1.28				
fw2	-8.27***	-2.30**	-8.27***	-3.81***	-2.49**	-2.36**	-2.79***			
fw3	-5.04***	-0.71	-5.05***	-3.58***	-2.97***	-2.74***	0.01	1.64		
fw4	-3.27***	0.25	-3.27***	-2.99***	-2.73***	-2.21**	1.05	2.33**	2.84***	
fw5	-2.72***	0.52	-2.72***	-2.69***	-1.49	-0.71	1.28	2.40**	2.82***	2.27**
COVID-19-Inclusive Period (2012)										
OLS										
PCR	-10.71***									
PLS	0.16	10.70***								
GKX_fw3	-0.52	2.09**	-0.52							
GKX_fw4	-2.54**	0.37	-2.54**	-3.33***						
GKX_fw5	-2.64***	0.25	-2.64***	-3.35***	-2.79***					
fw1	-7.81***	-1.61	-7.81***	-3.28***	-1.33	-1.16				
fw2	-7.06***	-1.96**	-7.06***	-4.29***	-2.47**	-2.23**	-2.02**			
fw3	-4.61***	-0.79	-4.61***	-4.48***	-3.21***	-2.76***	0.04	1.51		
fw4	-3.21***	-0.04	-3.21***	-3.88***	-3.54***	-2.47**	0.89	2.12**	2.77***	
fw5	-2.74***	0.21	-2.74***	-3.44***	-3.15***	-1.31	1.14	2.26**	2.85***	2.63***

Table 2.5: DM statistics all models during Period 2012 and Period 1911. The ' $fw(i)$ ', $i = 1, 2, \dots, 5$ means the proposed MLP models with 1 to 5 hidden layers, while ' $GKX_fw(i)$ ', $i = 3, 4, 5$, means GKX2020 models with 3 to 5 hidden layers. In 'DM statistics' tables, the difference of each model's OOS absolute error is computed as model m minus n in Equation (2), model n is from column labels, while model m is from row labels. For example, in the upper panel, the first number '-9.30' in the second column means 'PCR's absolute error minus 'OLS's. This indicates that Model 'PCR' performs better than 'OLS'.

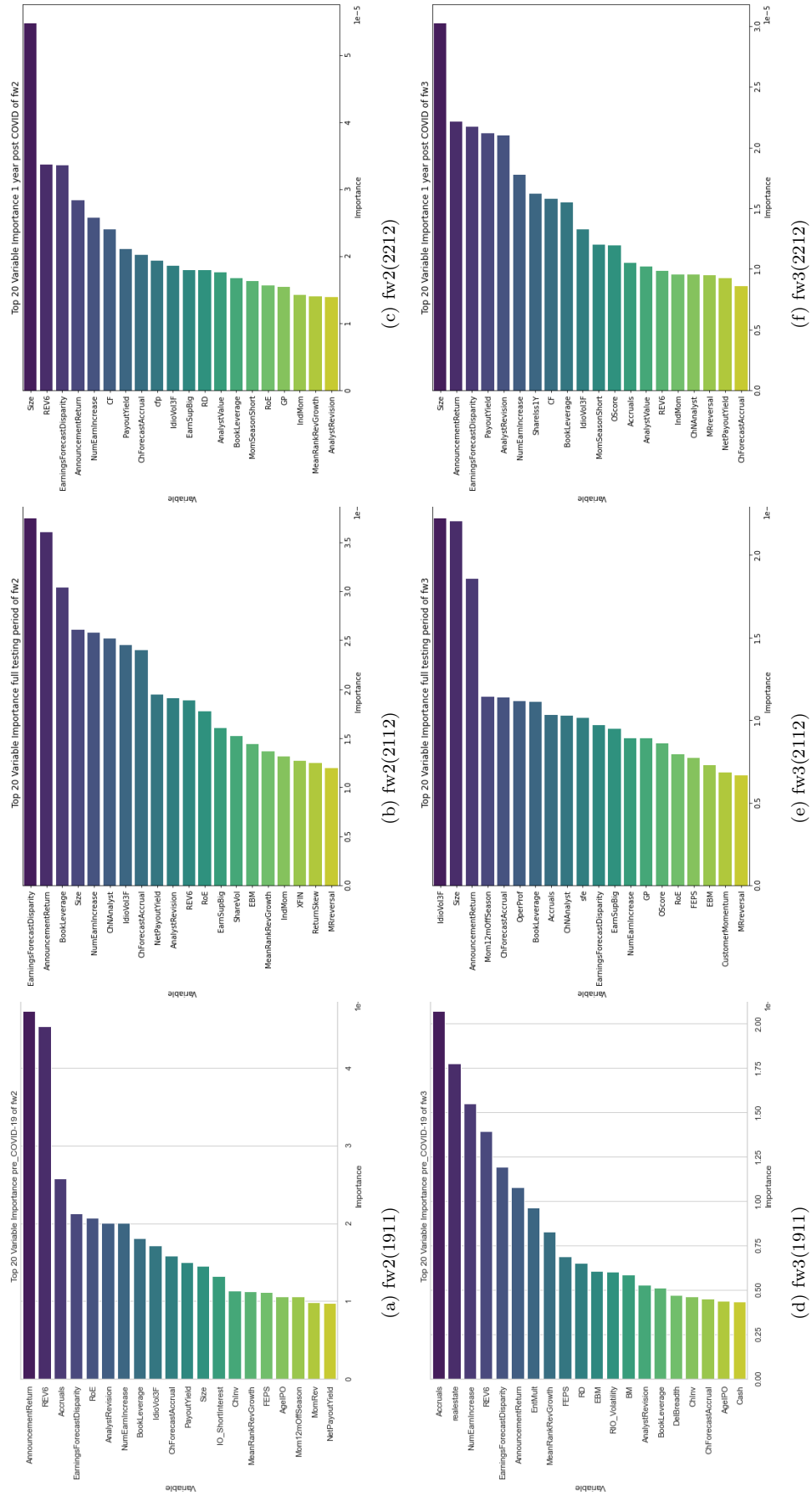


Figure 2.8: VI plots for the top 20 important factors of the best proposed MLP models for Pre-COVID-19 Period (1911), COVID-19-Inclusive Period (2112) and Period Including COVID-19 and One-Year After (2212). The Y-axis is the factor names, and the X-axis is the value of factor importance.

This study also examined the factor importance of Period Including COVID-19 and One-Year After (2212), which includes the period of one year post-pandemic (from January 2022 to December 2022), with a regime recovery pattern shown in Figure 3.7. Size surpasses all other factors by a wide margin in both fw2 and fw3. This suggests a strong resurgence of the size premium in the recovery phase, potentially reflecting market re-pricing of scale as a buffer against residual uncertainty or favoring smaller firms during rebound dynamics. Announcement Return (AnnouncementReturn) and Earnings Forecast Disparity (EarningsForecastDisparity) continue to rank highly in both models, confirming their status as the most persistent and regime-robust predictors throughout the entire sample. Additionally, payout-related variables (such as Payout Yield (PayoutYield) and Net Payout Yield (NetPayoutYield)) have notable prominence in the post-COVID period, especially in fw3, indicating that investor attention shifted toward shareholder-friendly cash distribution policies during economic normalization. Short-term reversal (REV6) also regains strength in fw2, consistent with mean-reversion effects reasserting in a recovering market.

The variable (factor) importance results illustrate some crucial insights. Firstly, cross-sectional factor premia are highly time-varying and regime-dependent, with pronounced rotations around the extreme market turbulence, such as the COVID-19 pandemic. Secondly, information-based characteristics, particularly Announcement Return (AnnouncementReturn) and Earnings Forecast Disparity (EarningsForecastDisparity), exhibit remarkable stability across all three periods and both model depths, reinforcing their role as core elements of the stock pricing. Eventually, modest yet systematic differences between fw2 and fw3 (e.g., fw3's stronger pre-crisis emphasis on accounting signals and post-crisis focus on payout policy) demonstrate that one-layer deeper network depth enables the capture of higher-order nonlinear interactions among firm-characteristics-sorted portfolio factors. In addition, high impact factors are quite consistent in alternative proposed MLP models, and they are fading in fw5, which shows in the factor importance plots in the appendix of Chapter 2.

To summarize, the proposed MLP models with 2 hidden layers outperform all alternative models with the highest OOS R^2 value in both periods, which is followed by the one with



Figure 2.9: Market-cap-weighted (MC) Price Index.

3 hidden layers. All proposed MLP models' performance is significantly improved when the COVID-19 period data is counted in the testing period, but the benchmark models show less sensitivity to the pandemic effect. Additionally, individual stock performance may have a significant impact on overall portfolio performance. Size, as one of the most important factors, has the most crucial impact on the best proposed models when the market was recovering from extreme fluctuations, as well as momentum factors and the factor of 'earnings surprise of big firms (EarnSupBig)'. The general fading variable importance in fw4 to fw5, together with results from OOS R^2 , challenges the legend 'deep, better' of MLP models. It agrees with the suggestion of Coqueret and Guida (2020) that data size should be beyond parameter size to a certain level.

2.5.3 Evaluation of Back Testing Performance

The portfolio performance is commonly evaluated via indicators such as annualized returns (annual return), Sharpe Ratio (SR), Sortino ratio (SO), maximum drawdown (MDD) and α . Table 2.6 shows the equal-weighted portfolios' performance of the selected 420 stocks with and without the pandemic period in the static transaction cost (50bps) scenario. In equal-weighted portfolios, the overall average annual return is clearly higher when the pan-

demic period is included, and the annual returns for individual models show the same trend. Although the overall standard deviation (Std) is higher for the COVID-19-Inclusive Period (2112), it does not affect the annual return much. For both periods tested, the lowest annual returns are from the models of 'fw1', 'OLS', 'PLS' and 'PCR', and the rest of the MLP-based models show a trend of 'deeper, better', but interestingly, none of them achieves higher excess returns compared with the 'buy-and-hold' strategy. Also, for the proposed models, a deeper structure comes with the higher Std, but the highest Std is shown by the 'buy-and-hold' strategy. The 'buy-and-hold' strategy, as a well-applied benchmark in backtesting of trading strategy constructions, is a passive strategy that opens a position from the first trading day, and holds the position until the final trading day. It does not need any market timing or signal generation techniques. Sharpe ratios reflect the capital gain under a certain level of risk which is measured by the standard deviation of excess returns. Period 2112's performance on SR is better than Period 1911. All models in the Period 2112 have higher Sharpe ratios than or very close to the benchmark strategy of 'buy-and-hold'. However, during the pre-COVID period (1911), all models were lower than or extremely close to 'buy-and-hold'. In terms of downside risk performance, an analogous phenomenon appears on Sortino ratios as well, which scales the gain from downside risks (e.g. the standard deviation from the negative excess returns). MDD, which measures the largest downfall from peak to the bottom of portfolio excess returns, shows the opposite trends to SR and SO, all are lower than 'buy-and-hold' in the full period but higher in the pre-pandemic period (1911). During the full testing period, our best-performing model, MLP with 2 hidden layers, showed a noticeable advantage which achieving the SR of 0.2651, Sortino Ratio of 0.4154 and the lowest MDD of 39.03% followed by MLP with 3 hidden layers. This model beat the traditional statistical models and the shallower MLP structure model as well. From the factor investing angle, alpha measures the extra gains from factors; all models have positive alphas, which are significant from zero for both periods. The proposed deep MLP models have higher alphas in the full testing period (2112), but no higher than GKX2020's MLP models. From the empirical asset pricing perspective, the insignificant α indicates a better interpretability of these factors. The positive residual OOS α of the best proposed MLP models with patterns (lower in Period 1911, a mild uptrend, and higher in Period 2112, a sharp uptrend) implies

the existence of 'omitted factors' or the arbitrage opportunities of the market. From a factor investing perspective, practitioners prefer a higher positive α , which implies their trading strategy derives higher abnormal returns beyond factors. Noticeably, the performance of GKX2020 models on the proposed dataset exhibits the similarity and approach to the 'buy-and-hold' strategy, which is detected as the trading signal capture failing.

As shown in the pre-COVID testing period (1911), the results have a mild upward trend. The overall annualized return is around 2% lower than the full testing period. The Std is around 0.01 lower as well. Shape Ratio and Sortino Ratio present that the proposed MLP models have a 'deep, better' trend, better than the 'OLS', 'PLS' and 'PCR' models, but the difference between SR and SO is less significant than the ones in the full testing period. It implies that models suffered higher downside risks in the pre-COVID period than in the pandemic period. The lowest MDD shows in the MLP with 3 hidden layers, but the best SR and SO appear in the MLP with 5 hidden layers, which also has the lowest alpha value. However, the 'fw5' and GKX2020 models' performance approaches the 'buy-and-hold' benchmark, which implies the failure to detect sign-based trading signals. The 'OLS', 'PLS' and 'PCR' models achieve similar but better α s in the pre-COVID period, which implies that practitioners have extra gain from the factor investing. Their alphas are even higher than the best alpha in the full testing period, which implies that, without high market fluctuation, the simplest models may lead to a greater gain from factors. But anyway, by comparing the full testing period and pre-COVID testing period results, it shows that the best proposed MLP models have higher profitability with relatively low risks. Figure 2.10 illustrates the cumulative excess returns of all models for the full testing period. It clearly shows the best proposed MLP model well manages the downside risks and achieves a relatively high excess return in the equal-weighted portfolio, no matter the testing period includes or excludes the pandemic period although the absolute returns are less than those in the simplest 'buy-and-hold' strategy.

For investigating how the turnover rates affect the profitability of the models, the turnover rate-based dynamic transaction cost is applied for a robust examination. Table 2.7 shows the

Equal-weighted portfolio with period of 1911										
model	AR	Std	SR	Ann.SR	SO	Ann.SO	MDD	α	Ann. α	t_stat_ α
fw5	0.10997	0.0345	0.2574	0.8917	0.3611	1.2509	-0.3745	0.0039	0.0465	10.3241
fw4	0.10045	0.0319	0.2527	0.8752	0.3463	1.1997	-0.3527	0.0040	0.0478	10.2502
fw3	0.08095	0.0273	0.2351	0.8144	0.3042	1.0539	-0.3166	0.0043	0.0521	11.3607
fw2	0.05719	0.0221	0.1996	0.6913	0.2528	0.8756	-0.3598	0.0045	0.0542	10.4123
fw1	0.04432	0.0206	0.1625	0.5630	0.1945	0.6736	-0.3952	0.0041	0.0487	8.6578
OLS	0.04286	0.0193	0.1661	0.5755	0.1961	0.6792	-0.3958	0.0064	0.0768	15.1934
PLS	0.04286	0.0193	0.1661	0.5755	0.1961	0.6792	-0.3951	0.0064	0.0767	15.1780
PCR	0.03819	0.0187	0.1504	0.5211	0.1915	0.6635	-0.5044	0.0062	0.0746	15.1854
GKX2020_fw5	0.11322	0.0354	0.2585	0.8955	0.3609	1.2502	-0.3680	0.0043	0.0511	11.5900
GKX2020_fw4	0.11346	0.0353	0.2594	0.8986	0.3622	1.2549	-0.3682	0.0042	0.0504	11.0508
GKX2020_fw3	0.11135	0.0348	0.2584	0.8950	0.3613	1.2517	-0.3684	0.0043	0.0512	4.9815
buy-and-hold (E)	0.11386	0.0356	0.2588	0.8967	0.3631	1.2577	-0.3701			

Equal-weighted portfolio with period of 2112										
model	AR	Std	SR	Ann.SR	SO	Ann.SO	MDD	α	Ann. α	t_stat_ α
fw5	0.13228	0.0434	0.2512	0.8703	0.3098	1.0731	-0.5109	0.0055	0.0664	17.0422
fw4	0.12637	0.0402	0.2565	0.8885	0.3257	1.1281	-0.4884	0.0053	0.0634	15.7385
fw3	0.10816	0.0333	0.2608	0.9036	0.3517	1.2184	-0.4319	0.0048	0.0579	14.0525
fw2	0.08841	0.0262	0.2651	0.9184	0.4154	1.4391	-0.3903	0.0039	0.0468	9.8960
fw1	0.07684	0.0239	0.2512	0.8703	0.3816	1.3218	-0.4292	0.0030	0.0358	6.7956
OLS	0.07514	0.0235	0.2573	0.8914	0.3938	1.3642	-0.4136	0.0034	0.0406	8.1689
PLS	0.07516	0.0227	0.2574	0.8917	0.3954	1.3698	-0.4139	0.0034	0.0405	8.1319
PCR	0.07151	0.0220	0.2513	0.8704	0.4155	1.4392	-0.5044	0.0050	0.0598	12.4912
GKX2020_fw5	0.13442	0.0451	0.2471	0.8558	0.2980	1.0324	-0.5315	0.0060	0.0718	18.4553
GKX2020_fw4	0.13441	0.0451	0.2470	0.8555	0.2977	1.0314	-0.5321	0.0061	0.0731	15.6569
GKX2020_fw3	0.13199	0.0446	0.2453	0.8499	0.2950	1.0221	-0.5356	0.0060	0.0715	7.5860
buy-and-hold (E)	0.13491	0.0452	0.2472	0.8564	0.2988	1.0349	-0.5310			

Table 2.6: Equal-weighted portfolio performance with (2112) and without (1911) the COVID-19 period considering the static transaction cost. Report for equal-weighted portfolio excess returns. The lower panel is the report for the full testing period, while the upper panel is for the testing period without the pandemic. Both reports include the indicators of annualized returns (AR), standard deviations (Std), Sharpe Ratio (SR), Sortino Ratio (SO), Maximum drawdown (MDD), α and the t statistic of the alpha which is listed in the last column. The 'buy-and-hold' (E) for equal-weighted and V for value-weighted) applies as a benchmark strategy for all models without any prediction technique, hence no α is presented. The notation 'Ann' means annualized.

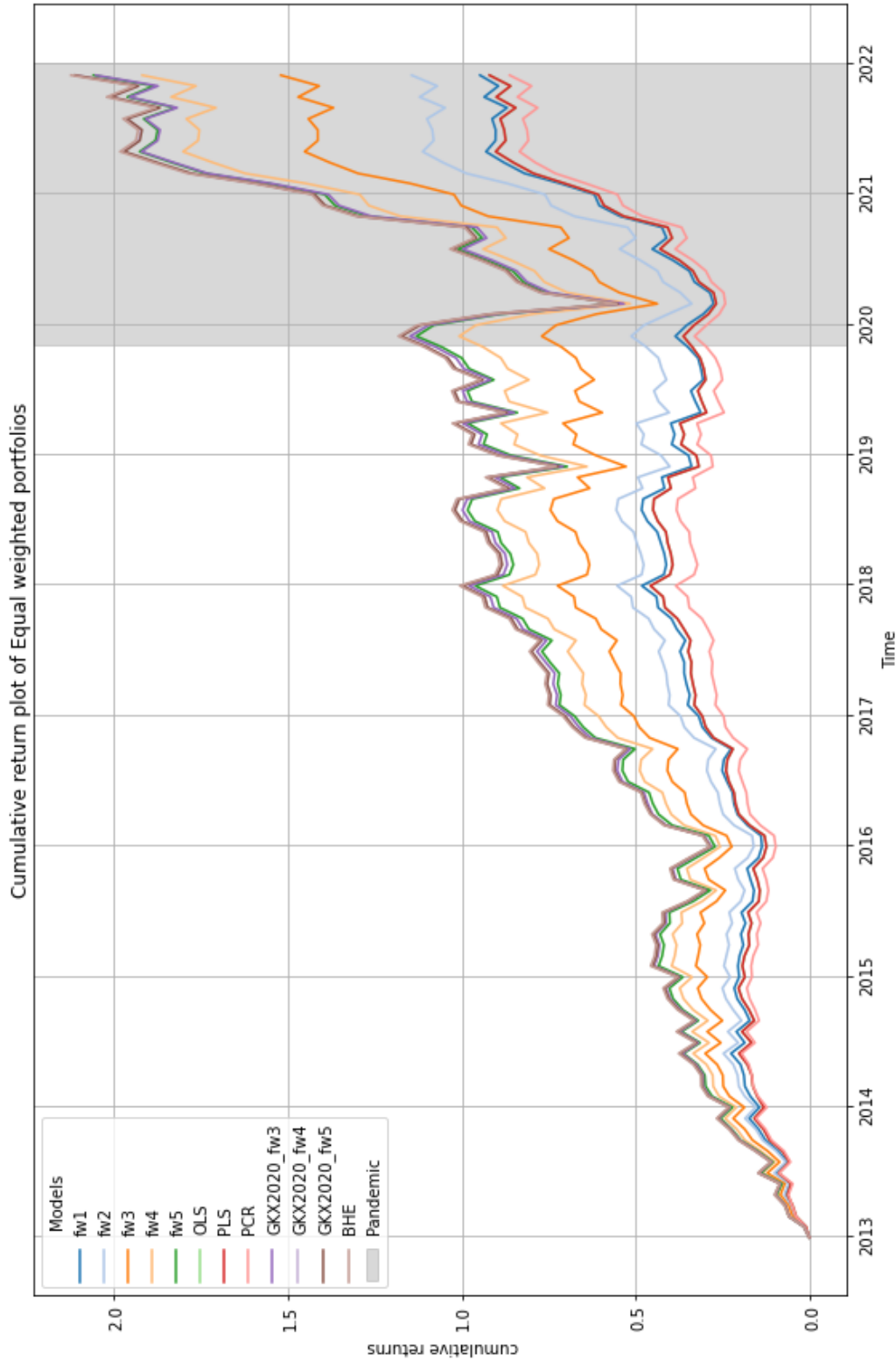


Figure 2.10: Cumulative excess return plot for equal-weighted portfolio considering the static transaction cost. The $fw(i)$, $i = 1, 2, \dots, 5$, are denoted as the proposed MLP models with i hidden layers. 'BHE' means buy-and-hold benchmark for the equal-weighted portfolio.

robust backtesting results for equal-weighted portfolios, and Figure 2.11 shows the robust cumulative return of the equal-weighted portfolios. The dynamic transaction cost robust examination in equal-weighted portfolios reveals a different picture from that in static transaction cost. The annualized returns generally improved in the dynamic transaction cost framework. This indicates that static transaction cost with an average rate may cause the over-penalization for the models' profitability in the highly variable turnover rate scene. Concretely, in the Pre-COVID-19 Period (1911), all models' annualized returns do not exceed the buy-and-hold benchmark (12.52%), and traditional statistical models (OLS, PCR, PLS) are significantly lower than the MLP models. GKX2020 MLP models slightly outperform the proposed deep MLP models (fw4 to fw5), but they significantly outperform the proposed shallow MLP models. However, when introducing the volatility as an indicator for risks (Sharpe ratio and Sortino ratio), Model fw5 and GKX2020 MLP models perform insignificantly better than the buy-and-hold benchmark. And the proposed MLP models show a tendency of 'deeper, better'. The maximum drawdown (MDD) between models is insignificant (approximately -14.8%). The turnover rates are negatively correlated with the annualized returns, where the traditional statistical models show the highest turnover rates. In the COVID-19-Inclusive Period (2112), deep MLP models slightly outperform the buy-and-hold benchmark in annualized return, and MLP models show superiority to the traditional statistical models. The proposed MLP models show a 'deeper, better' pattern as well. This is aligned with the Sharpe Ratio. Nonetheless, Sortino ratios present a different picture. Different from the Sharpe Ratio, which considers the entire volatility as risk, the Sortino ratios only introduce the downside risk. The Sortino ratios of traditional statistical models surprisingly outperform the MLP models and the buy-and-hold benchmark with the highest turnover rates. The proposed MLP models exhibit superiority over the GKX2020 models, except Model fw3, in the Sortino ratio. This provides evidence that high turnover rates are not the primary reason for the deterioration of profitability in specific conditions. In addition, the OOS α for all models in both periods is positive at a 1% significance level. The traditional statistical models outperform MLP models in the Pre-COVID-19 Period, but lag behind during the COVID-19-Inclusive Period. The OOS α of the proposed models exhibits the 'deep, better' pattern, but none of them exceeds the GKX2020 models during

the COVID-19-Inclusive Period. The cumulative return plot Figure 2.11 also supports these findings.

Apart from the equal-weighted portfolios, the value-weighted method is widely used as the asset allocation strategy. As early as Fama and French (1993) to later GKX2020 (Gu et al., 2020), all factor models have employed the value-weighted method. Specifically, value weights are generated from the ratio of an individual stock's market capitalization to the total market capitalization of all selected stocks. The portfolio returns are generated by the weighted average of the individual stocks' returns. Table 2.8 is the back-testing results of the value-weighted portfolios of the selected stocks. Compared with the equal-weighted method, the value-weighted method distributes more weight on larger-cap stocks, which increases the impact of the larger-cap stocks and decreases the impact of smaller-cap stocks. By comparing the performance of the equal-weighted portfolios and the value-weighted portfolios, we can evaluate the sensitivity of the models to the stock's market capitalization or namely, the volatility of the stock returns. With the full testing period, the conclusions drawn from the equal-weighted portfolio still hold for the value-weighted portfolio. From the table, the annualized returns of all models are quite similar to the equal-weighted portfolio's annualized returns, but standard deviations are lower than the equal-weighted portfolio's. The Sharpe Ratios are relatively higher, but the Sortino Ratios vary among models. The differences in Sortino Ratio between individual models are less pronounced when compared with the equal-weighted portfolio's Sortino Ratios. The overall maximum drawdowns are dropped considerably, and the highest Sortino Ratio value shows up on the proposed MLP model with 3 hidden layers, which comes with the second lowest MDD (33.04%). Compared to the lowest MDD in the equal-weighted portfolio with the full testing period, it dropped 15.37%. This implies that the value-weighted portfolio moderates the downside risks with a reasonable gain in the full testing period. Figure 2.12 also shows how the value-weighted portfolio moderates the downside risks. Figure 2.12 presents the full testing period cumulative returns of all models. It exhibits clearly that MLP with 3 hidden layers (green line) well reduced the downside fluctuation with a decent gain. Signal-detecting failure still exists in the deep MLP model and GKX2020 models.

Equal-weighted portfolio with Pre-COVID-19 Period (1911)										
Model	AR	Ann.SR	SR	Ann.SO	SO	MDD	Turnover	α	Ann. α	t_stats
buy-and-hold(E)	0.1252	0.8892	0.2567	1.2303	0.3552	-0.1483	0.0120	-	-	-
OLS	0.0827	0.7214	0.2083	0.9451	0.2728	-0.1456	0.7576	0.0064	0.0768	15.193***
PCR	0.0780	0.6847	0.1977	0.9508	0.2745	-0.1386	0.7587	0.0062	0.0746	15.185***
PLS	0.0827	0.7219	0.2084	0.9455	0.2730	-0.1456	0.7573	0.0064	0.0767	15.178***
GKX2020_fw3	0.1152	0.8986	0.2594	1.2437	0.3590	-0.1483	0.0215	0.0043	0.0512	4.9815***
GKX2020_fw4	0.1156	0.8994	0.2596	1.2455	0.3595	-0.1482	0.0155	0.0042	0.0504	11.051***
GKX2020_fw5	0.1153	0.8967	0.2589	1.2416	0.3584	-0.1483	0.0151	0.0043	0.0511	11.590***
fw1	0.0884	0.7553	0.2180	0.9820	0.2835	-0.1494	0.6699	0.0041	0.0487	8.6578***
fw2	0.0917	0.8035	0.2320	1.0618	0.3065	-0.1426	0.4951	0.0045	0.0542	10.412***
fw3	0.1049	0.8643	0.2495	1.1526	0.3327	-0.1458	0.2611	0.0043	0.0521	11.361***
fw4	0.1115	0.8846	0.2554	1.2164	0.3511	-0.1472	0.1018	0.0040	0.0478	10.250***
fw5	0.1140	0.8892	0.2567	1.2315	0.3555	-0.1499	0.0357	0.0039	0.0465	10.324***

Equal-weighted portfolio with COVID-19-Inclusive Period (2112)										
Model	AR	Ann.SR	SR	Ann.SO	SO	MDD	Turnover	α	Ann. α	t_stats
buy-and-hold(E)	0.1353	0.8512	0.2457	1.0181	0.2939	-0.2890	0.0093	-	-	-
OLS	0.1154	0.8117	0.2343	1.0934	0.3157	-0.2693	0.7280	0.0034	0.0406	8.1689***
PCR	0.1088	0.7940	0.2292	1.0981	0.3170	-0.2391	0.7280	0.0050	0.0598	12.491***
PLS	0.1155	0.8122	0.2344	1.0938	0.3157	-0.2693	0.7278	0.0034	0.0405	8.1319***
GKX2020_fw3	0.1357	0.8519	0.2459	1.0147	0.2929	-0.2911	0.0175	0.0060	0.0715	7.5860***
GKX2020_fw4	0.1364	0.8567	0.2473	1.0260	0.2962	-0.2890	0.0120	0.0061	0.0731	15.657***
GKX2020_fw5	0.1363	0.8564	0.2472	1.0251	0.2959	-0.2889	0.0116	0.0060	0.0718	18.455***
fw1	0.1158	0.8129	0.2347	1.0282	0.2968	-0.2684	0.6401	0.0030	0.0358	6.7956***
fw2	0.1168	0.8315	0.2400	1.0399	0.3002	-0.2495	0.4714	0.0039	0.0468	9.8960***
fw3	0.1265	0.8365	0.2415	1.0003	0.2888	-0.2752	0.2467	0.0048	0.0579	14.053***
fw4	0.1349	0.8576	0.2476	1.0386	0.2998	-0.2828	0.0948	0.0053	0.0634	15.739***
fw5	0.1358	0.8591	0.2480	1.0376	0.2995	-0.2837	0.0320	0.0055	0.0664	17.042***

Table 2.7: Equal-weighted portfolio performance with (2112) and without (1911) the COVID-19 period considering the dynamic transaction cost. Report for equal-weighted portfolio excess returns. The lower panel is the report for the full testing period, while the upper panel is for the testing period without the pandemic. Both reports include the indicators of annualized returns (AR), standard deviations (Std), Sharpe Ratio (SR), Sortino Ratio (SO), Maximum drawdown (MDD), α and the t statistic of the alpha which is listed in the last column. The 'buy-and-hold'(E for equal-weighted and V for value-weighted) applies as a benchmark strategy for all models without any prediction technique, hence no α is presented. The notation 'Ann' means annualized. ***, **, and * denote statistical significance at the 1%, 5%, and 10% levels, respectively.

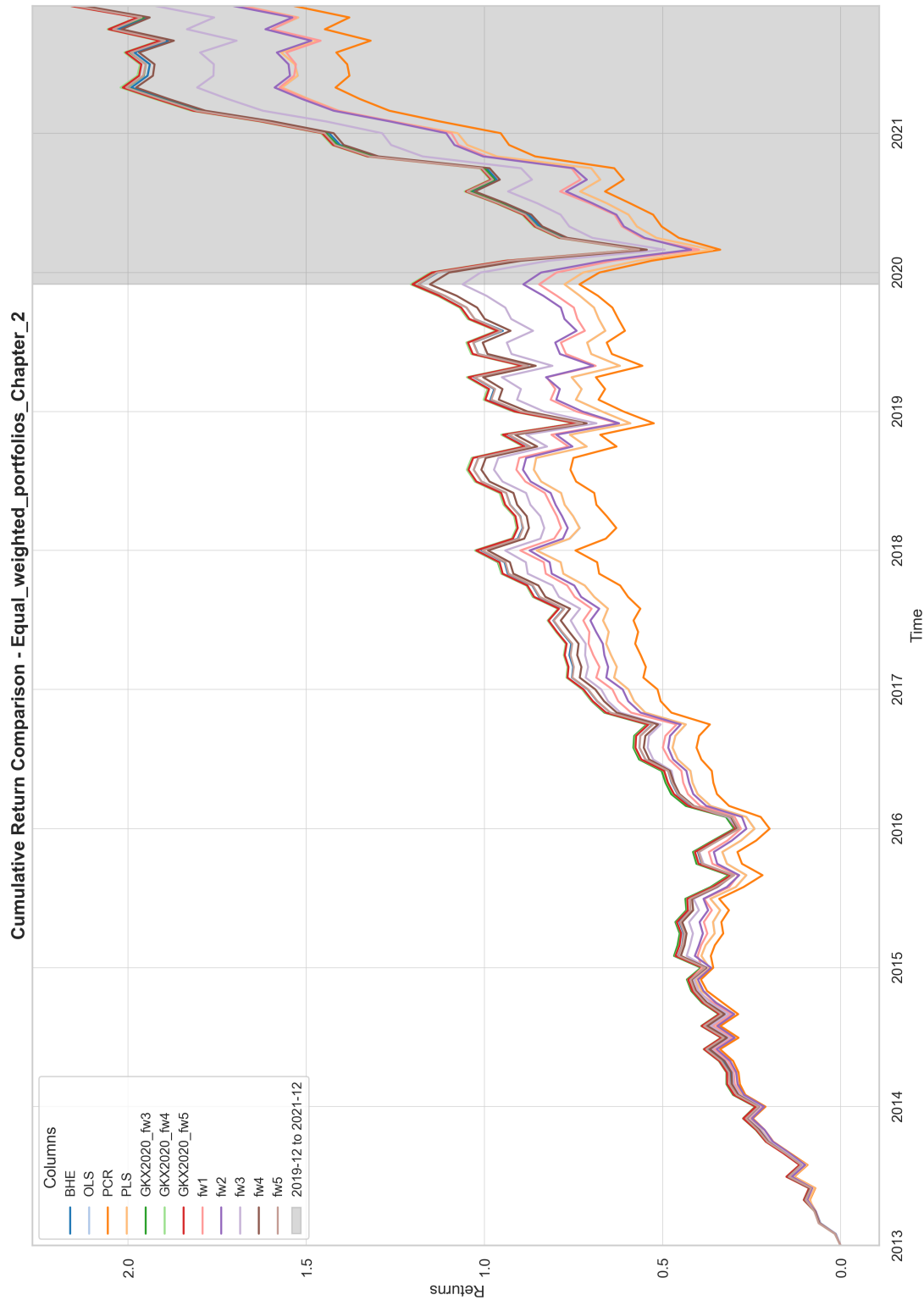


Figure 2.11: Cumulative excess return plot for equal-weighted portfolio considering the dynamic transaction cost. The $fw(i)$, $i = 1, 2, \dots, 5$, are denoted as the proposed MLP models with i hidden layers. 'BHE' represents the buy-and-hold benchmark strategy in the equal-weighted method.

In the pre-COVID testing period, compared with the equal-weighted portfolio, the annual returns for all models are slightly higher than annualized returns in the equal-weighted portfolio and the Std is noticeably decreased for MLP-based models including GKX2020, but no great change in ‘OLS’, ‘PLS’ and ‘PCR’ models. The lowest MDD took place at MLP with 3 hidden layers which is 26.50%, but the proposed MLP model with the highest SR and SO cannot surpass the ‘buy-and-hold’ strategy. Whereas the overall Sortino ratios are higher than the ones in the equal-weighted portfolio at the same period settings. The SR and SO of proposed MLP models show a ‘deeper, higher’ trend. The best proposed MLP model, with 2 hidden layers shows lower SR and SO in Period 2112 of the value-weighted portfolio compared with the ones in the equal-weighted portfolio, which indicates that shallow MLP models are more flexible and have slightly better capability in capturing the patterns in high fluctuation scenarios. Since portfolio weighting has no impact on signal detecting, the GKX2020 and deep MLP model signal detecting failure still exists under the value weighting mode in both periods. Figure 2.12 shows the cumulative return plot, which exhibits that the meaning of MLP models lies upon hedging downside risks instead of deriving the absolute returns.

The transaction cost robustness examination is conducted on value-weighted method as well. Table 2.9 is the backtesting results of the value-weighted portfolios considering the dynamic transaction cost. The conclusions drawn from the equal-weighted portfolio are consistent with the value-weighted method. From the table, the annualized returns of all models are generally higher than the equal-weighted portfolio’s annualized returns, as well as the Sharpe ratio and Sortino ratio. Noticeably, the maximum drawdown is significantly moderated in the value-weighted method, particularly in the COVID-19-Inclusive Period. In Pre-COVID-19 Period, deep MLP models’ annulized returns are superior to the buy-and-hold benchmark, as well as the Sharpe ratio and Sortino ratio. In the COVID-19-Inclusive Period, deep MLP models still show excellence in annulized returns, but it becomes complicated when considering turnover rate, volatility (SR) and downside risks (SO). Market capitalization has a higher impact on Model PCR than alternatives. Model fw1 surprisingly achieves the highest Sharpe

Value-weighted portfolio with period of 2112							
Model	AR	Std	SR	Ann.SR	SO	Ann.SO	MDD
fw5	0.1310	0.0393	0.2708	0.9382	0.3327	1.1524	-0.4345
fw4	0.1252	0.0359	0.2801	0.9704	0.3677	1.2737	-0.3913
fw3	0.1092	0.0299	0.2893	1.0023	0.3938	1.3642	-0.3304
fw2	0.0875	0.0241	0.2831	0.9808	0.3984	1.3800	-0.3564
fw1	0.0766	0.0223	0.2666	0.9236	0.3706	1.2838	-0.3488
OLS	0.0721	0.0209	0.2661	0.9219	0.3546	1.2284	-0.3221
PLS	0.0722	0.0209	0.2662	0.9223	0.3548	1.2290	-0.3219
PCR	0.0711	0.0211	0.2600	0.9006	0.3744	1.2970	-0.3627
GKX2020_fw5	0.1335	0.0407	0.2671	0.9252	0.3257	1.1283	-0.4518
GKX2020_fw4	0.1335	0.0407	0.2672	0.9255	0.3257	1.1284	-0.4518
GKX2020_fw3	0.1318	0.0405	0.2653	0.9190	0.3231	1.1194	-0.4555
buy-and-hold (V)	0.1337	0.0408	0.2673	0.9259	0.3263	1.1302	-0.4513
Value-weighted portfolio with period of 1911							
Model	AR	Std	SR	Ann.SR	SO	Ann.SO	MDD
fw5	0.1171	0.0316	0.2948	1.0213	0.4136	1.4328	-0.2982
fw4	0.1059	0.0293	0.2864	0.9922	0.3839	1.3298	-0.2827
fw3	0.0908	0.0253	0.2821	0.9773	0.3617	1.2531	-0.2650
fw2	0.0623	0.0214	0.2244	0.7774	0.2692	0.9327	-0.3564
fw1	0.0505	0.0201	0.1910	0.6617	0.2238	0.7754	-0.3374
OLS	0.0489	0.0187	0.1971	0.6827	0.2183	0.7563	-0.3209
PLS	0.0490	0.0187	0.1972	0.6833	0.2186	0.7572	-0.3208
PCR	0.0444	0.0186	0.1782	0.6174	0.2100	0.7276	-0.3627
GKX2020_fw5	0.1194	0.0322	0.2954	1.0233	0.4231	1.4656	-0.2968
GKX2020_fw4	0.1195	0.0322	0.2959	1.0250	0.4237	1.4677	-0.2946
GKX2020_fw3	0.1180	0.0319	0.2943	1.0194	0.4224	1.4633	-0.2946
buy-and-hold (V)	0.1197	0.0323	0.2956	1.0241	0.4243	1.4698	-0.2963

Table 2.8: Value-weighted portfolio performance with and without COVID-19 period considering the static transaction cost. Report for value-weighted portfolio excess returns. The upper panel is the report for the full testing period, while the lower panel is for the testing period without the pandemic. Both reports include the indicators of annualised returns (AR), standard deviations (Std), Sharpe Ratio (SR), Sortino Ratio (SO), Maximum drawdown (MDD). The ‘buy-and-hold (V)’ applies as a benchmark strategy for all models without any prediction technique, hence no α is presented. The notation of ‘Ann’ means annualized.

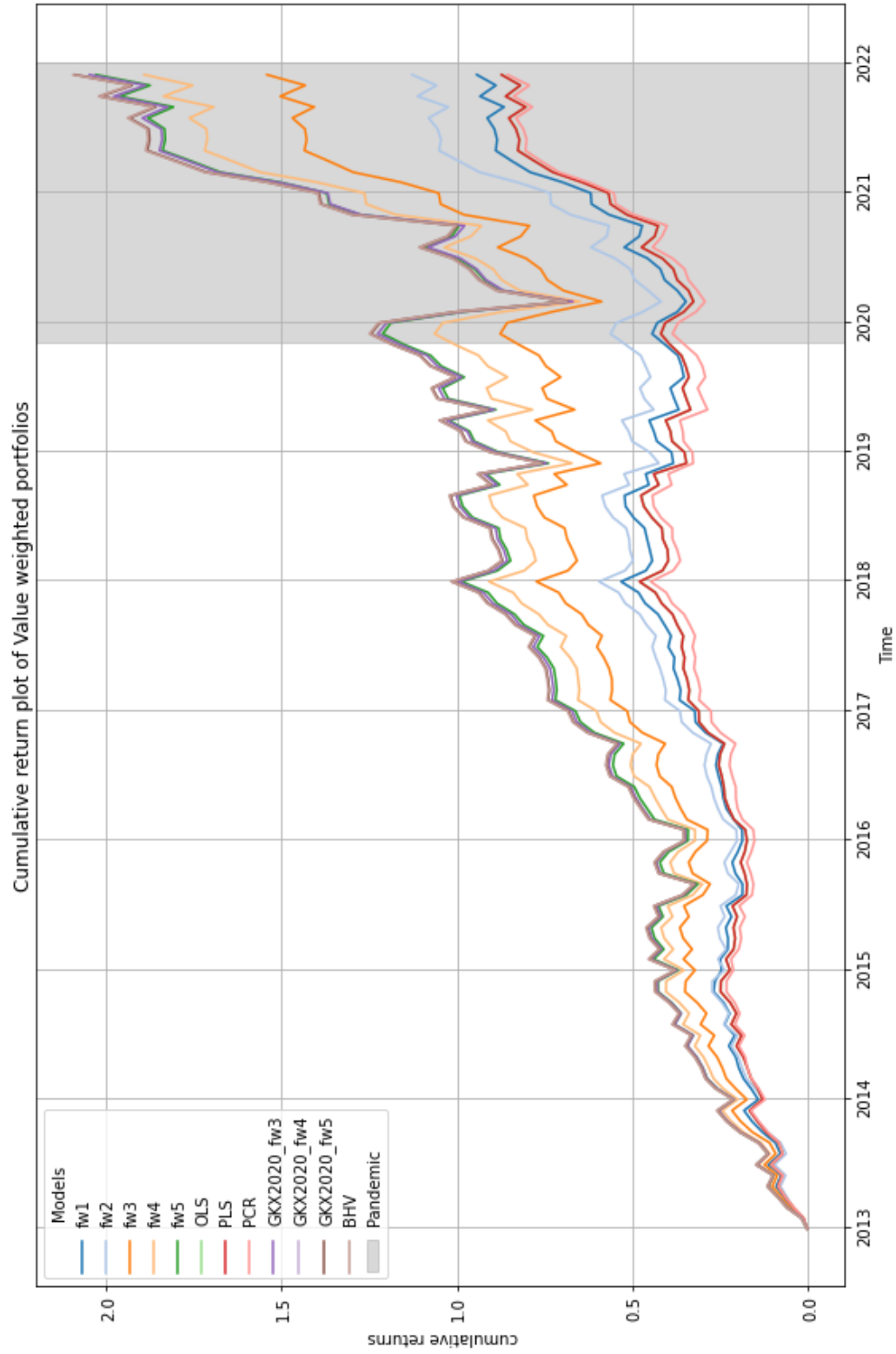


Figure 2.12: Cumulative excess return plot for value-weighted portfolios considering the static transaction cost. The $fw(i)$, $i = 1, 2, \dots, 5$ denoted for the proposed MLP models with i hidden layers. The 'PLS' and 'PCR' mean PLS and PCR with 95% variance coverage, which is equal to the notation of 'PLS' and 'PCR' in previous tables and diagrams. 'OLS' means OLS linear regression model.

ratio and Sortino ratio with a relatively high turnover rate. It proves again that the high turnover rate could affect the absolute capital returns, but not the profitability concerning the volatility and downside risks in the large-cap scenario of stocks. However, in conjunction with the report on equal-weighted portfolios, the turnover rates for the proposed deep MLP models and the GKX2020 models are approaching the buy-and-hold benchmark, suggesting a possible failure in trading signal detection. The cumulative return plot 2.13 supports the findings in the value-weighted portfolio backtesting report.

In conclusion, whether the equal-weighted portfolio or the value-weighted portfolio, the overall annualized returns from the COVID-19-Inclusive Period (2112) perform better than the Pre-COVID-19 Period (1911) due to the strong rebound trend during the pandemic in the static transaction scene. The strong upward trend with deep drawdown caused by the pandemic increases the SR and SO as well. The standard deviations for all models are smaller in the pre-pandemic period, while the MDDs are generally smaller in the pandemic as well. Between the equal-weighted portfolio and the value-weighted portfolio, the value-weighted portfolio for the COVID-19-Inclusive Period (2112) performs well for moderating the downside risks, while for the Pre-COVID-19 Period (1911), it increases the capital gain under certain risks for all models. Among the main and benchmark models, all proposed MLP models outperform the traditional statistical models, such as OLS, PLS and PCR in the scenario of static transaction cost. By considering risks and returns, the proposed MLP model with 3 hidden layers outperforms in the value-weighted portfolio for both periods and the equal-weighted portfolio for the Pre-COVID-19 Period (1911). It manages the balance of downside risk and relatively large capital gain, but for the equal-weighted portfolio COVID-19-Inclusive Period (2112), the MLP with 2 hidden layers overtakes the alternative models, achieving a better gain in terms of its risks. The robustness examination results show that high turnover rates significantly affect the models' profitability, but they have a lower impact than the transaction deduction rates (50bps for static and 20bps for dynamic). All models in all scenarios exhibit significant positive OOS α s, indicating additional gains from factors. However, in all scenarios, the GKX2020 models are suspected of failing to detect trading signals. Their performance approaches the performance of the 'buy-and-hold' strategy. The

Value-weighted portfolio with Pre-COVID-19 Period (1911)							
Model	AR	Ann.SR	SR	Ann.SO	SO	MDD	Turnover
buy-and-hold (V)	0.1388	1.1795	0.3405	1.6054	0.4634	-0.1297	0.0120
OLS	0.1095	1.0190	0.2941	1.2788	0.3692	-0.1160	0.7300
PCR	0.1069	0.9953	0.2873	1.2989	0.3750	-0.1101	0.7430
PLS	0.1096	1.0199	0.2944	1.2799	0.3695	-0.1160	0.7303
GKX2020_fw3	0.1388	1.1802	0.3407	1.6059	0.4636	-0.1297	0.0193
GKX2020_fw4	0.1398	1.1863	0.3425	1.6155	0.4664	-0.1297	0.0152
GKX2020_fw5	0.1396	1.1851	0.3421	1.6139	0.4659	-0.1297	0.0149
fw1	0.1151	1.0189	0.2941	1.3395	0.3867	-0.1190	0.6672
fw2	0.1139	1.0483	0.3026	1.3350	0.3854	-0.1244	0.4941
fw3	0.1331	1.1890	0.3432	1.5782	0.4556	-0.1153	0.2557
fw4	0.1318	1.1353	0.3277	1.5084	0.4355	-0.1348	0.1067
fw5	0.1391	1.1832	0.3416	1.6125	0.4655	-0.1294	0.0344
Value-weighted portfolio with COVID-19-Inclusive Period (2112)							
Model	AR	Ann.SR	SR	Ann.SO	SO	MDD	Turnover
buy-and-hold (V)	0.1539	1.0688	0.3085	1.3057	0.3769	-0.2352	0.0093
OLS	0.1424	1.0854	0.3133	1.5324	0.4424	-0.1678	0.7110
PCR	0.1394	1.0629	0.3068	1.5111	0.4362	-0.1773	0.7083
PLS	0.1425	1.0861	0.3135	1.5331	0.4426	-0.1678	0.7112
GKX2020_fw3	0.1535	1.0635	0.3070	1.2913	0.3728	-0.2381	0.0155
GKX2020_fw4	0.1546	1.0725	0.3096	1.3110	0.3785	-0.2352	0.0117
GKX2020_fw5	0.1546	1.0724	0.3096	1.3106	0.3783	-0.2352	0.0114
fw1	0.1487	1.1022	0.3182	1.6028	0.4627	-0.1631	0.6387
fw2	0.1423	1.0834	0.3128	1.4456	0.4173	-0.1853	0.4814
fw3	0.1491	1.0786	0.3114	1.3434	0.3878	-0.2219	0.2501
fw4	0.1515	1.0713	0.3092	1.3659	0.3943	-0.2184	0.1031
fw5	0.1545	1.0810	0.3120	1.3311	0.3842	-0.2300	0.0324

Table 2.9: Value-weighted portfolio performance with and without COVID-19 period considering the dynamic transaction cost. Report for value-weighted portfolio excess returns. The upper panel is the report for the full testing period, while the lower panel is for the testing period without the pandemic. Both reports include the indicators of annualised returns (annual return), standard deviations (Std), Sharpe Ratio (SR), Sortino Ratio (SO), Maximum drawdown (MDD). The ‘buy-and-hold (V)’ applies as a benchmark strategy for all models without any prediction technique, hence no α is presented. The notation of ‘Ann’ means annualized.

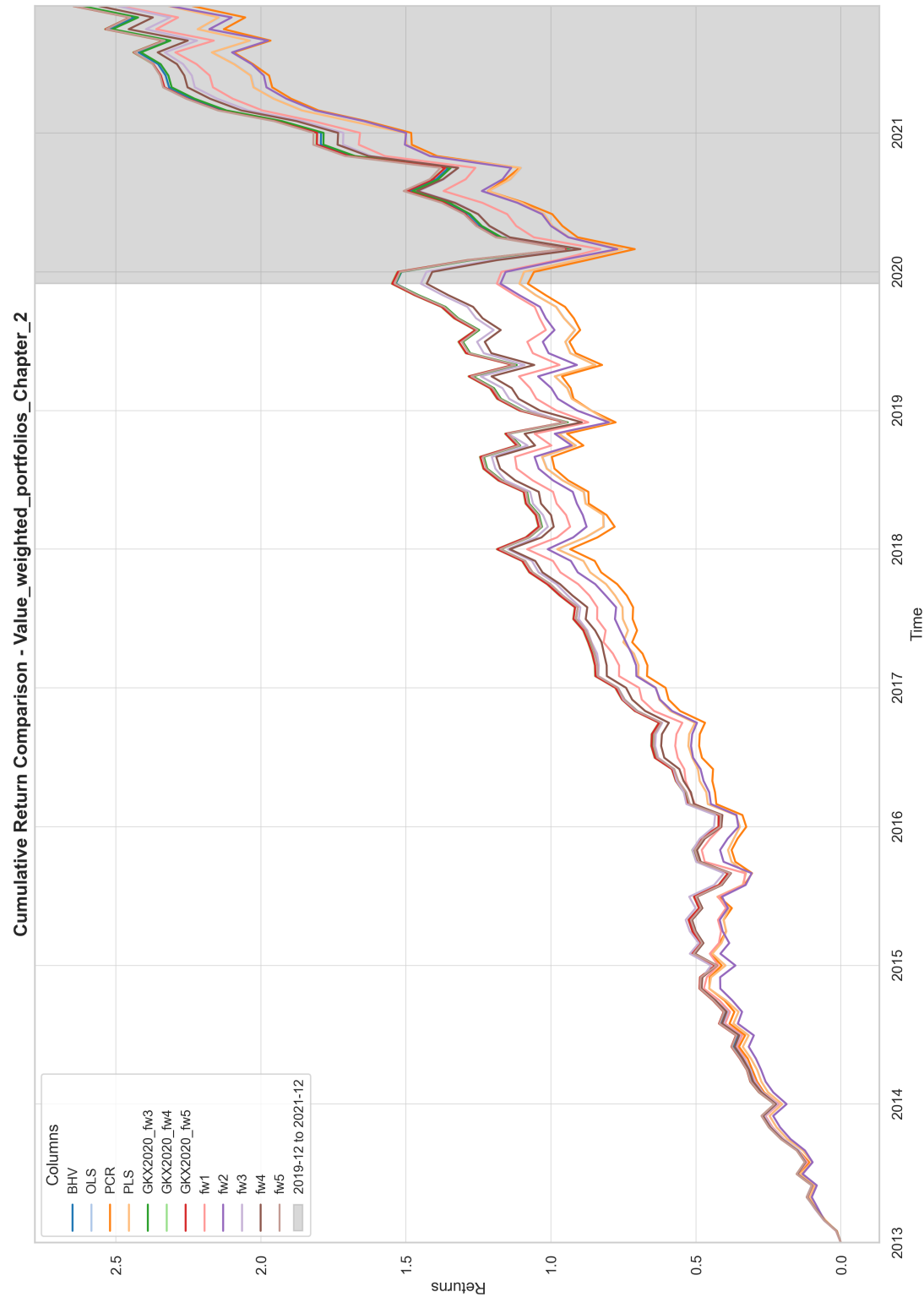


Figure 2.13: Cumulative excess return plot for value-weighted portfolio considering the dynamic transaction cost. The $fw(i)$, $i = 1, 2, \dots, 5$ denoted for the proposed MLP models with i hidden layers. 'BHV' represents the buy-and-hold strategy in the value-weighted method.

turnover rate and transaction deduction rate have a high impact on the profitability of the models. Moreover, the transaction deduction rate has a higher impact than the turnover rate in this chapter. The main findings in this section are presented as: firstly, the model overfitting issue traps the sign-based trading signal detection; secondly, when the market experiences a strong upward trend, the fundamental trading strategy such as ‘buy-and-hold’ could be the most efficient trading strategy without considering the risks. In this sense, MLP models’ meaning for factor investing is more about managing downside risks and deriving a relatively high return instead of ‘beating the market’. Finally, the value-weighted portfolio reduces downside risks compared to the equal-weighted portfolio. Finally, trading signal filtering techniques can be applied to the system to further decrease the turnover rate and optimize the profitability.

2.6 Conclusion and Discussion

This chapter revisits and develops the GKX2020’s MLP models via employing a dynamic MLP structure introduced by Coqueret and Guida (2020) and features as firm characteristics sorted factors provided by Andrew Y.; Zimmermann (2020), re-evaluating the model and portfolio performance for the testing period of COVID-19. It considers the more realistic U.S. stock trading environment and practical conditions, such as high liquidity, ‘going concern’, variable borrowing rates and available borrowing shares from the brokers. Specifically, in the real trading environment, low liquidity brings systematic risks, such as failing to close the position in time to stop loss or confirm the trading profit. The ‘going concern’ is due to the listed firms’ unexpected insolvency or delisting, which may cause severe loss to the investors. For the short positions, which borrow the securities from either brokers or companies, investors face variable borrowing rates according to the demand or estimated market risks, and sometimes the rates can be high enough to make the short trading unavailable. Moreover, brokers usually provide limited borrowing shares each day.

The variable (factor) importance analysis reveals that the predictive power of firm-characteristics-sorted portfolio factors of the best-performing MLP models (fw2 and fw3) is highly time-

varying and regime-dependent. In the pre-COVID period, AnnouncementReturn is the dominant predictor in the shallower model (fw2), whereas the deeper architecture (fw3) places greater emphasis on accounting-based signals such as Accruals. During the COVID-19-Inclusive Period, risk-related factors, including Earnings Forecast Disparity (EarningsForecastDisparity), leverage (BookLeverage), size, and idiosyncratic volatility (IdioVol3F), stably ranked in the top 20, reflecting heightened uncertainty and risk premia. In the post-COVID recovery phase (up to December 2022), Size becomes overwhelmingly dominant in both models, with the persistent strength of Announcement Return (AnnouncementReturn) and Earnings Forecast Disparity (EarningsForecastDisparity), as well as the renewed relevance of payout-related variables. These findings highlight three key insights: cross-sectional factor premia exhibit pronounced regime shifts during the extreme market turbulence, such as the one caused by COVID-19, capture the patterns that linear models such as Fama-MacBeth regressions would struggle to capture; information-based factors, particularly Announcement Return and Earnings Forecast Disparity, demonstrate considerable robustness of both models in all periods, confirming these factors as core drivers of the expected returns of the examined stocks; the modest but systematic differences between fw2 and fw3 suggest that deeper neural networks can capture higher-order nonlinear interactions among the factors that shallower models cannot capture. To conclude, variable importance analysis under the permutation method improves the economic and financial interpretability of the ML models in asset pricing.

This work also shows that a model's strong predictive power does not guarantee the profitability of a strategy built on it. The meaning of MLP models focuses on moderating the downside risks rather than achieving the absolute capital gain exceeding the buy-and-hold benchmarks, which verifies the results from Avramov et al. (2021). They achieve absolute returns during strong uptrends and deep downtrends. As the directional trading signal approach, which is commonly used in the trend-following strategy, is applied in this study, future researchers can explore alternative signal methods for improving profitability by improving the trading signal quality, such as deploying a signal filtering mechanism or configuring a restricted threshold for the correct market timing. Additionally, it is also worth

investigating the generalization of the proposed MLP models, for example, examining them in alternative markets or assets, or on small or medium-cap stocks. Also, all models examined have significant positive α , which indicates the abnormal returns for the investors, but with these frames and factor configurations, stocks are not fully explained. The pattern of α shows no sign of model mis-specification.

Overall, MLP models bring promising performance on downside risk hedging and relatively high absolute return after transaction cost as the factor investing strategy in this case, especially in the COVID-19-Inclusive Period and the value-weighted method. This means models have a higher preference for stocks with larger market capitalization. The design of the factor investing strategy with the proposed MLP models highly approaches the real stock market practical conditions, which makes it more applicable to practitioners. It highlights the new inspirations on neural network models in the empirical asset pricing and factor investing. Whereas, the existing challenges of limited data length, trading signal filtering, model hyperparameters tuning and stock selection highly influence the model and portfolio performance. Topics such as how to moderate the restriction of data length, improve the trading signal quality and tune the model hyperparameter effectively are still worth further exploration. In addition, the MLP models pre-assume the sequential configuration of the data, with no mechanism embedded for processing data sequentially. More temporally specific neural network models can be employed for the asset pricing tasks. In addition, to further improve the financial and economic interpretability of these neural network models, the DM test can be applied to individual stocks to investigate the model's preference for an industry or sector, while investigating further on MSE and OOS R^2 for the contribution of characters, for example, which industry sectors have higher predictive power in a certain framework, and what is the impact of market capitalization size. These structures can also be applied to the volatility modelling for risk control.

Attention is Truly All We Need?

3.1 Introduction

Chapter 2 proposes the dynamic MLP models for reducing the dimension of ‘factor zoo’ (Harvey et al., 2016) of the asset pricing factor model, which is inspired by the work of Gu et al. (2020) (GKX2020) and Coqueret and Guida (2020). It employs a dynamic structural algorithm from Coqueret and Guida (2020) on MLP models to generate a hidden layer structure for the proposed MLP models as a substitute for the fixed pyramidal MLP structure of GKX2020’s MLP models. It also substitutes the transformed firm characteristic factors of GKX2020’s work to the firm characteristic-sorted portfolio (Andrew Y.; Zimmermann, 2020) factors as the input factors to increase the interpretability of the asset pricing factor model in machine learning (ML) methods. The out-of-sample (OOS) fitness of the proposed MLP model with 2 hidden layers significantly outperforms traditional statistical models and GKX2020’s benchmarks in both the Pre-COVID-19 Period (1911) and COVID-19-Inclusive Period (2112). Moreover, it provides a significant advantage in mitigating downside risks caused by extreme market fluctuations (e.g. COVID-19).

However, as discussed in Zhou and Wang (2024), although early ML models present excellent capabilities in modelling non-linear structure in financial and economic time series data, fewer authors notice that early ML models, such as MLP, random forest(RF), Support Vector Machine (SVM) and MLP autoencoder, are not originally designed for time series data. They only assume that these models naturally capture the temporal dependencies in the data, or that the data follow independent and identically distributed (i.i.d.) assumptions, or attempt to enforce a temporal sequence via extending or rolling window methods. However, extend-

ing or rolling window methods face a dilemma for such tasks: the larger window size has a lower capability to capture the temporal dependency since the temporal dependency inside the window cannot be recognized; the smaller window size has a higher capability to detect the temporal dependency, but increases the possibility of model overfitting due to its small training data size. Therefore, exploring alternative ML structures originally developed the temporal mechanism, such as the recurrent neural networks RNN(Rumelhart et al., 1986a), LSTM(Hochreiter and Schmidhuber, 1997), GRU(Cho et al., 2014) as examples, which relax the aforementioned assumption, could be meaningful for ML-based asset pricing research. Moreover, GKX2020's work extends the dimension of the factors by using the Kronecker product of firm characteristics and macroeconomic indicators in the cross-sectional study and increases the data size for their ML models. This does not assist the 'factor zoo' issue but brings their research into serious doubt of pure data-mining exercises. Thus, simplifying the factor configuration and improving model financial and economic interpretability in ML asset pricing research could be another meaningful exercise, even if the study in Chapter 2 has already highly reduced the dimension of factors. Additionally, apart from capturing the sequential information of time series data, how far the model can capture the historical information away from the current time step of a time series, which is known as the 'memory' of the model, is an alternative crucial aspect for financial and economic time series modelling. Thus, searching for mechanisms that can extend the memory of recurrent neural network models, which could further improve the model performance without sacrificing the model simplicity for computational efficiency and model interpretability, can be meaningful as well. Furthermore, GKX2020, as the seminal work for the ML-based asset pricing, focuses on the predictability of the ML models instead of evaluating pricing mechanisms, for example, how to explain the pricing error α , how to improve the OOS model fitness (R^2), and how to ensure the consistency of pricing performance over time.

Building on the findings of Chapter 2, this study introduces a pre-trained recurrent neural network (RNN) architecture enhanced with self-attention mechanisms for addressing the aforementioned problems. Specifically, this study proposed two pre-trained RNN attention models: the RNN global self-attention model (self_att) and the RNN sliding window sparse

attention model (sparse_att). In the proposed structures, the MLP autoencoder is deployed as the pre-trained method for the factors' dimensional reduction task and factor information extraction, which further compresses the input factors' dimension into 70% of the original factors' dimension and reconstructs the missing values of the original factors. This process is also known as feature extraction or feature engineering in ML methods. The RNN structure (Elman, 1990) was developed to overcome the limitation of the MLP (Rumelhart et al., 1986a) models, which are the MLP models that process time series data with the sequential assumption. Unlike feedforward network models such as MLP, RNNs develop a hidden state that is updated recurrently over time, which enables the network to process input sequences. Concretely, the hidden state at each time step is calculated based on both the information of the current time step and the previous hidden state. This allows RNNs to learn patterns across time. Nevertheless, the vanilla RNN model does not overcome the issues of gradient vanishing and explosion existing in feedforward network models, especially when it is estimated by the stochastic gradient descent method (SGD). Although such issues can be mitigated through multiple approaches, such as adding L1 regularization to the loss function or using the Adam optimizer with the SGD method, the gradient vanishing issue remains more challenging. It causes the short 'memory' of the model, which prevents the RNN model from effectively capturing long-term dependencies in the sequence.

Fortunately, LSTM (Hochreiter and Schmidhuber, 1997) and GRU (Cho et al., 2014) as RNN variation models extend the memory of the classic RNN to a degree in a general perspective. However, since the temporal sparsity characteristic of the asset pricing data, which presents as a smaller data size with longer time dependency, the LSTM model and the GRU model's high parameterization nature may cause overfitting. Thus, the most prevailing attention mechanism, the dot product based self-attention mechanism (Vaswani et al., 2017) and its sparse version, the sliding window attention mechanism (Beltagy et al., 2020), which are designed for the most prevalent large language models (LLM), Transformers, could be a great substitute of LSTM and GRU for balancing the trade-off between the model parameterization complexity and longer-term temporal dependency capturing. The attention mechanism is a specific structure added to neural network models to reinforce pattern recognition. It

is inspired by the human brain’s attention mechanism. The biological design of the human brain could preferentially process impulsive information (e.g. visual items with stronger colour and louder sound). The ML attention mechanism is the mechanism that introduces heavier weights on the high-impact information and lighter weights on the low-impact information with respect to temporal correlation. The advanced attention mechanisms are confirmed for their capability to significantly improve the predictive power with relatively low computational costs (smaller parameter size) in domains such as natural language processing (NLP) and machine translation (Vaswani et al., 2017; Luong et al., 2015). Given the similar data lengths across early NLP tasks and asset pricing, attention mechanisms may be a suitable approach to improving empirical asset pricing performance. Furthermore, as the attention mechanism was originally designed for sequential data, it does not naturally align with time series data, since attention weights are computed based on correlations between any two sequential points, not on previous points alone. This causes future data leakage when attention mechanisms are applied without the imposed causal masks. However, little of the attention-based asset pricing literature has addressed this issue, for example, the papers of Zhou and Wang (2024) and Ma et al. (2023).

This study also further investigates whether the classic attention mechanisms generally improve the asset pricing and factor investing performance by constructing the pre-trained RNN additive attention model (B-Additive), the pre-trained RNN Luong’s general attention model (L-General), the pre-trained RNN Luong’s dot product attention model (L-DotProd) and the pre-trained RNN Luong’s concatenation attention model (L-Concat) as benchmark models. These attention mechanisms were originally developed for natural language processing (NLP) tasks, and some of them have been examined in asset pricing tasks, such as Zhou and Wang (2024); Kelly et al. (2025). The proposed and benchmark models are examined and backtested in three market regimes to demonstrate their ability to handle extreme market fluctuations, covering the Pre-COVID-19 Period (1911), the COVID-19-Inclusive Period (2112), and the Period Including COVID-19 and One-Year After (2212). To the best of my knowledge, this is the first large-scale study focusing on attention mechanisms in the asset pricing context. The MLP-pretrained RNN global self-attention model and the RNN

sliding-window sparse attention model, as innovative time series tailored architectures, are introduced here and examined for the first time in asset pricing research.

Therefore, the contributions of this study can be considered from five angles:

- Employed MLP autoencoder as the pre-trained method for handling the missing values in firm characteristic-sorted portfolio factors and the dimensional reduction method for improving computational efficiency and mitigating the potential model overfitting issue.
- Examined the RNN structures on firm characteristic-sorted portfolio pretrained factors as a solution for capturing the temporal dependency of the data.
- Proposed two RNN attention variation models for asset pricing and factor investing contexts, the pre-trained RNN global self-attention model and the pre-trained RNN sliding window sparse attention model.
- Investigated the necessity of RNN classic attention models, such as additive attention and Luong’s three attention models, in the context of asset pricing and factor investing.
- Tested models’ capabilities of handling the extreme market conditions during the Pre-COVID-19 Period (1911), the COVID-19-Inclusive Period (2112) and the Period Including COVID-19 and One-Year After (2212).
- Enforced the causal mask on the attention mechanism for time series to prevent the future information leaking problem, which is ignored by recent attention-based research on financial and economic time series data.

The study is organized into 6 sections. Section 2 introduces the related work of this research, while Section 3 describes the data employed in this study. Section 4 depicts the models deployed for this research. Section 5 shows the methods for evaluating the model performance and the back-testing performance, as well as an analysis of the empirical results. The conclusion and further discussion are presented in Section 6.

3.2 Related work

As mentioned in Chapter 2, researchers focus on the linear and non-linear statistical functions. Later, the burgeoning of AI algorithms provides some promising solutions for the dimensional disaster challenge of asset pricing research. The seminal work of this domain is Gu et al. (2020)'s 'empirical asset pricing in machine learning'. It examines the majority of traditional ML models, such as MLP, SVM, RF, and gradient boosting, on 94 firm characteristics and 8 macroeconomic indicators, cross-sectionally from the U.S. stock market, and shows that MLP with 3 hidden layers outperforms all alternative models on high-dimensional factors. This work is followed by Wang (2021), which tests Gu et al. (2020)'s models on cryptocurrency pricing. It agrees that the MLP with 3 hidden layers is the best model, even in the cryptocurrency market. It generalized the MLP models' application in asset pricing. Later, Gu et al. (2021) proposed a structure based on two MLP autoencoders for asset pricing tasks. One for generating β s for factors and the other for factors' dimensional reduction. By dot-producting the outputs of the two MLP autoencoders, it achieves better performance than Gu et al. (2020)'s best-performing model. Equation (3.1) shows the additive form, and Equation (3.2) shows the general form of the asset pricing factor model. Both of Gu et al. (2021, 2020) agree that in ML-based empirical asset pricing research, the concentration is on exploring the function form of $g(f_{i,t}; \theta)$. That is the foundation of later research in this direction.

$$r_{i,t+1} = \mathbb{E}_t[r_{i,t+1}] + \varepsilon_{i,t+1} \quad (3.1)$$

$$r_{i,t+1} = g(f_{i,t}; \theta) + \varepsilon_{i,t+1} \quad (3.2)$$

Avramov et al. (2021) conduct a comparative study on the MLP model with three hidden layers of Gu et al. (2020), the generative adversarial neural network model with RNN cores of Chen et al. (2024), the instrumental PCA model of Kelly et al. (2019) and the conditional autoencoder extension of Gu et al. (2021) in the U.S. stock market from a portfolio

strategy-wise perspective. This is discussed extensively in Chapter 2. Bagnara (2022) conduct a comprehensive and critical survey of recent developments at asset pricing in machine learning (ML), with a particular focus on addressing the long-standing ‘factor zoo’ problem. The review mainly investigates the classic ML methods and classifies the literature into five methodological categories: regularization, dimension reduction, regression trees and random forests, neural networks, and comparative analyses, which are applied within two main frameworks: empirical factor modelling with return prediction and stochastic discount factor (SDF) estimation. Its comparative studies find that boosted trees and neural networks outperform linear benchmarks in predictive accuracy. It identifies persistent challenges, including limited economic interpretability of ML models, the risk of data snooping, and weak integration with causal inference, and calls for hybrid approaches that combine economic theory, robust time series validation, and evaluation based on economically meaningful metrics such as portfolio-wise backtesting performance. Moreover, it aligns with the conclusion of Avramov et al. (2021) that the economic value of ML-based strategies often lies less in boosting unconditional average returns than in the ability to hedge downside risk and preserve capital during downturns or crisis periods.

Although traditional ML-based asset pricing literature provides substantial evidence that early-stage ML models have advantages in improving prediction accuracy and mitigating market downside fluctuation risks, they face the problems of the compliance of ML models and the asset pricing context. For example, MLP, SVM and RF were not originally designed for sequential or time series data, but early literature imposes the assumption of sequence recognition when they are applied to sequential data. This may affect the prediction accuracy or increase the probability of model misspecification and overfitting, even if extending or rolling window methods are deployed. Thankfully, later literature on the second generation of ML models, such as RNNs and their variants, addressed this problem due to their sequential nature. Unlike traditional empirical asset pricing models, Chen et al. (2024) propose a novel Stochastic Discount Factor (SDF) estimation approach based on a Generative Adversarial Network (GAN) (Goodfellow et al., 2014). The SDF is a generalised framework for asset pricing factor models under the non-arbitrage assumption. The linear and non-

linear factor models, ML models such as MLP, or more advanced, Transformer models, all can be transformed and fitted in the SDF framework. Specifically, their model employs an MLP autoencoder structure for the cross-sectional dimension reduction of firm characteristics, and a Long Short-Term Memory (LSTM) network to extract economic states from large-scale macroeconomic indicators. Within this framework, the generator formulates a candidate SDF, while the discriminator evaluates it. This adversarial process enforces the no-arbitrage condition by driving the pricing error (α) towards zero. Namely, it configures the no-arbitrage as the training goal of the entire model. It shows the superiority of traditional ML models and proves that the LSTM macroeconomic states promote the model performance significantly. Their work addresses the suitability for time series data through the LSTM model and statistically defends the no-arbitrage hypothesis. However, if the market exists arbitrage opportunities, which is possibly caused by price limits or liquidity fractions, even the most powerful GAN can only fit an approximate SDF rather than the true pricing kernel. Moreover, although the authors attempt to interpret factor importance, it is difficult to provide clear economic meanings comparable to traditional factors. Additionally, the model may face issues such as mode collapse and non-convergence, which cause training instability, while statistically minimizing the α may raise the suspicion of data snooping. In this sense, the proposed models on an RNN foundation and the empirical factor settings are simpler and more interpretable, thereby respecting real market conditions.

However, the application of RNNs and attention mechanisms to asset pricing is still at a very early and cutting-edge stage, and the extant literature remains relatively sparse. The most relevant literature of this study is Zhou and Wang (2024), which is developed from the work of Gu et al. (2021). It keeps the MLP structure for calculating the factor loadings, but alters the MLP autoencoder to the RNN additive attention structures for factor dimensional reduction. By dot producting the output factor loadings from the MLP structure and the extracted factors from the RNN additive attention structures, the estimated stock excess returns can be computed. It resolves the temporal dependency capturing issue by using RNN, LSTM, and GRU exchange models, while adopting the additive attention mechanism to extend memory and capture long-term temporal dependencies in data. Its LSTM additive

attention model with the extracted six factors outperforms all benchmark models, including Gu et al. (2021) and Kelly et al. (2019). Nonetheless, it does not explain how they determine the dimensionality of the extracted factors, nor how to interpret these alternative factors, which leaves their models shrouded in 'black-box' shadows. Furthermore, the additive attention mechanism extends the memory for capturing the longer-term temporal dependency of the data, which potentially improves the prediction accuracy, but the limited data size may cause overfitting due to the complication of the model with high parameter numbers. Moreover, since the attention mechanism computes attention weights across the entire sequence of RNN temporal hidden states, it risks incorporating future data into current estimates. To prevent this forward-looking information leakage, it is crucial to embed a causal mask that restricts the attention focus solely to past and present states. Otherwise, it introduces the look-ahead bias, which may cause spurious performance, loss of economic interpretability, non-tradable strategy and data snooping issues. However, it is a common issue in recent attention-based asset pricing research, including the work of Kelly et al. (2025); Ma et al. (2023); Zhang (2022) and Cong et al. (2021b). The proposed attention models notice this problem and enforce a causal mask individually to satisfy the single-direction characteristic of the time series data.

Interestingly, recent works concentrate on the application of prevailing large language models (LLMs), such as Transformers, which contain multiple attention mechanisms, instead of simplified models that focus on attention mechanisms themselves. For example, Cong et al. (2021b) implements a comparative study of ML models with the temporal dependency capture nature, such as RNN, single or bi-directional LSTM, RNN with additive and classic Transformer, for empirical asset pricing on large-scale U.S. stocks. In their study, LSTM achieves the best OOS fitness, but GRU derives the highest annualized Sharpe ratio from the long-short portfolio. Zhang (2022) conducts a similar comparative study to Cong et al. (2021b). It compares MLP and MLP residual, Convolutional Neural Network (CNN) and CNN residual, RNNs, RNN additive attention, classic Transformer model and traditional ML models. It finds that the RNN additive attention model slightly outperforms the classic Transformer model, achieving the highest OOS R^2 . Ma et al. (2023) proposes a CNN

multi-head self-attention-based encoder-only Transformer model for Chinese stock pricing and volatility forecasting from January 2000 to December 2019. It uses 72 firm characteristics and 8 macroeconomic indicators as the pricing factors and volatility forecasting results for calculating asset allocation weights. The Transformer model with three encoder blocks outperforms all alternative variants and traditional ML benchmarks. It proves that the Transformer model with the multi-head self-attention mechanism performs well in return forecasting and even better in volatility forecasting. Although these works inspire later researchers with the ML function form for asset pricing, the largest problem of these works is neglecting the single-direction characteristic of the time series data, since none of them imposes the causal mask to prevent the future information leaking, even if they are mitigated by sequential methods, such as the rolling window and the extending window. Additionally, most of these works employ the data ending in the year of 2016 (Gu et al., 2021; Avramov et al., 2021), which excludes the extreme market conditions caused by the COVID-19 pandemic. Namely, they did not examine the model performance during the market turbulence to show the stability of the models they applied. Instead, this study employs three periods that cover the stages of the Pre-COVID-19 Period (1911), the COVID-19-Inclusive Period (2112), and the Period Including COVID-19 and One-Year After (2212), and investigates how typical RNN-based attention models respond to the effects of the COVID-19 pandemic. Furthermore, most of these works use firm characteristics and macroeconomic indicators directly as input factors, which raises concerns about data snooping or data mining. In this study, the firm characteristics-sorted portfolios are applied as the original input factors, which cross-sectionally consider the information that covers the entire market and improve the economic explainability of the model.

Some of the asset pricing literature based on attention mechanisms appears to circumvent the future data leaking issue, as they apply it cross-sectionally rather than temporally. For example, Kelly et al. (2025) cross-sectionally applies an encoder-only Transformer model with Luong’s general (bilinear form) attention mechanism for generating the best SDF at each time t , which derives the highest OOS Sharpe ratio. They examine their model on large-scale U.S. stocks and 132 normalized firm characteristics as factors, while deploying

a short rolling training window of 60 observations. Their work covers the entire period of the COVID-19 pandemic. And their model outperforms the MLPs and linear benchmark models. Nevertheless, they simply substitute the missing values in the firm characteristics with zero, but ignore the input factors' data quality. For LLMs, low data quality increases the probability of model hallucination, which introduces untruthful similarity between different firm characteristics and leads to incorrect predictions. Moreover, they generate the SDF at each time t , and combine it into a time series-like sequence, but it neglects the temporal dependency to some extent. Still, the limited data size with a highly parameterized model structure increases the risk of model overfitting and lowers the model's economic interpretability. The MLP autoencoder pre-training method of this study is employed for handling the missing values in the input factors and reduces the input dimension to moderate model overfitting issues, which may also assist their asset pricing research. The simplified model structure in this study, with only an RNN and one attention layer, and the extracted factors may conquer the general problems that exist in advanced ML-based asset pricing research. Similarly, Chatigny et al. (2021) proposes a hybrid deep learning framework that combines stacked residual blocks (adapted from the N-BEATS model) for nonlinear feature extraction with a linear self-attention layer. This combined architecture cross-sectionally evaluates the importance of firm characteristics and macroeconomic variables. Specifically, the N-BEATS architecture relies on a deep stack of fully connected layers with forward and backward residual links. In the context of asset pricing, this design has the advantage since it allows the model to capture deep non-linear relationships in noisy financial data while actively preventing gradient vanishing and overfitting. The model directly estimates the SDF by minimizing squared Euler equation errors, and reports remarkably strong out-of-sample performance with annualized Sharpe ratios exceeding 2.8 at the market level, and above 1.0 even after excluding the smallest stocks or restricting the stocks to the Russell 3000 constituents. Their linear self-attention improves economic interpretability by highlighting relevant macroeconomic and firm-level features during different market conditions (e.g., the 2008 crisis). This is an innovative work, but it still faces issues as it practically neglects the transaction costs for short-selling positions, market impact, or leverage constraints, which reduces the implementability in real-world trading. Additionally, although

the missing firm characteristics are imputed with cross-sectional medians, which is a typical method commonly applied in the industry, it could induce systematic biases if the missing values are manipulated. It is also worth noting that their attention weights capture the statistical relevance of factors at each time point rather than structural causality and temporal dependencies. Thus, their high Sharpe ratio should be carefully confirmed for robustness, non-overfitting and non-data-snooping suspicion.

3.3 Data description

Similar to the data method of Chapter 2, the proposed models and benchmarks employ sorted portfolio returns as observable factors, which are reconstructed by Andrew Y.; Zimmermann (2020), to estimate individual stock returns. The 182 factors from Andrew Y.; Zimmermann (2020), which excludes the factor with missing values larger than 40% for maintaining a reasonable data quality, are selected as the original input in this study. The 420 top large-cap U.S. stocks are selected from the U.S. NASDAQ and NYSE, covering approximately 21.38% of the overall market capitalization, while following the rules of Chapter 2. Despite the proposed models deploying the MLP autoencoder as the pretrained method for dimensional reduction and information subtraction, excessive missing values can hinder model performance by reducing the effective information available for representation learning and increasing the risk of biased imputations, especially for models with attention mechanisms. The attention mechanisms may concentrate on unreliable or non-informative time steps, weakening local or global feature extractions. The biased imputations pose a risk, particularly in deep learning-based forecasting. If it is not properly constrained, it can introduce hallucinated inputs, values generated from prior correlations rather than observable causality, thereby distorting both training and evaluation. In addition, the risk-free rate for constructing the excess stock returns and evaluating the Sharpe ratio and Sortino ratio is also derived from the Kenneth R. French Data as in Chapter 2.

The entire monthly data covers from January 1957 to December 2022. The separation of the training period, validation period and testing period is listed in Table 3.1. Different from

the data method of Chapter 2, this study adopts the rolling window method instead of the prevailing extending window method in previous literature, such as Gu et al. (2020). The validation window size is approximately 20% of the training size. Thus, the ‘Initial in-sample total’ in Table 3.1 includes the training window of 553 observations and the validation window of 119 observations. The step size is 12, which means the parameters are re-estimated every 12 months, and the training window, together with the validation window, moves forward every 12 months. This configuration balances the computational efficiency and model fitness. It also improves the model’s training stability.

Name	Start date	End date	Observation No.
Initial in-sample total	1/1957	12/2012	672
Testing (OOS) for ‘1911’	1/2013	11/2019	83
Testing (OOS) for ‘2112’	1/2013	12/2021	108
Testing (OOS) for ‘2212’	1/2013	12/2022	120

Table 3.1: Data splits for in-sample and out-of-sample data. ‘1911’ means the testing period ends before the pandemic happened, ‘2112’ means the testing period contains the pandemic period and ‘2212’ means the testing period contains not only the pandemic period but also one year after the pandemic. The ‘Initial in-sample total’ includes the training observation of 553 and the validation observation of 119.

3.4 Models

The pre-trained recurrent neural network (RNN) attention models employ the MLP autoencoder as the pertaining method for the dimensional reduction of the original input factors, which is the firm characteristic-sorted portfolios provided by Andrew Y.; Zimmermann (2020), and handling the missing values. The pretrained factors (abstracted features or factors) then feed into the main models shown in Figure 3.2. The main models are organized by an RNN fundamental structure with an attention layer, which are introduced in Section 3.4.2 to Section 3.4.4. Figure 3.1 shows the entire workflow of the pretrained RNN attention models. This study investigates the RNN models with the additive attention mechanism (Bahdanau et al., 2015), Luong attention mechanisms (Luong et al., 2015), global self-attention mechanism (Vaswani et al., 2017) and the sliding window sparse attention mechanism (Beltagy et al., 2020), and compares them with the vanilla RNN model and its variations: long short-term memory (LSTM) neural networks and gated recurrent unit (GRU) neural networks. Among these RNN attention models, the RNN sliding window

sparse attention model and the RNN self-attention model are the proposed RNN attention models. The RNN sliding window sparse attention model extracts the sliding window sparse attention mechanism from the Longformer model (Beltagy et al., 2020), which is introduced in Section 3.4.3.5, while the RNN self-attention model extracts the self-attention mechanism from the classic Transformer model (Vaswani et al., 2017), which is introduced in Section 3.4.3.3. Since the sliding window sparse attention belongs to the category of self-attention, in this study, the classic self-attention mechanism from (Vaswani et al., 2017) is denoted as ‘global self-attention mechanism’ for preventing any confusion. Although the pre-trained RNN additive attention model, Luong’s attention models are still innovative in the existing literature for asset pricing research, they are configured as the benchmarks, as these attention mechanisms were developed in the early stage. The additive attention, Luong’s general attention, and Luong’s concatenate attention mechanisms have larger parameter sizes than the dot-product-based attention mechanisms, increasing the risk of overfitting and computational cost. However, Luong’s dot-product attention is a non-parameterized attention, which improves the computational efficiency but sacrifices the performance and training stability. Vanilla RNN, LSTM, and GRU models are benchmarks for evaluating whether RNN-based models benefit from attention mechanisms.

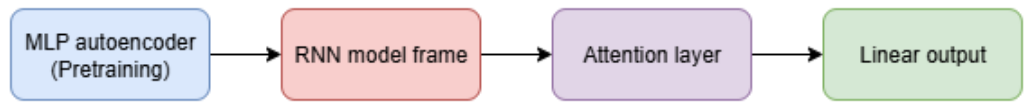


Figure 3.1: The workflow diagram of the RNN attention models.

Figure 3.2 is the workflow diagram of the main models, which assemble a two-hidden-layer vanilla RNN frame with an attention layer. An attention layer contains one attention mechanism, for example, the additive attention mechanism or one of Luong’s attention mechanisms. The left part of Figure 3.2 shows the main workflow of the RNN attention models. The input X , which is the pricing factors in this case, goes through the RNN hidden layers and the sliding window sparse attention layer, then derives the output $\hat{y}$. The loss function L measures the difference between the output derived from the RNN attention structure and the label y , which is the stock excess return series in this case. W is the learnable parameter

matrix recurrently calculated according to the timeline. The right part exhibits the unfolded 3-dimensional structure of the RNN attention models, where the RNN frame has two pyramidal hidden layers: $h^{(1)}$ and $h^{(2)}$. The first hidden layer and the second layer configure 64 and 32 neurons, respectively. From the second dimension scope, the RNN structure collects the current and the one-step previous information and passes them to the next time step as introduced in Section 3.4.2. From the third dimension angle of the figure, each time step has a pyramidal MLP network, which achieves the goal of reducing the feature's dimension. Additionally, the attention layer estimates and distributes the attention weights to each time step to depict the relevance of any two time steps in a pre-defined sight range. The attention output z_t is used for computing the estimated $\hat{y}$ with a linear connection.

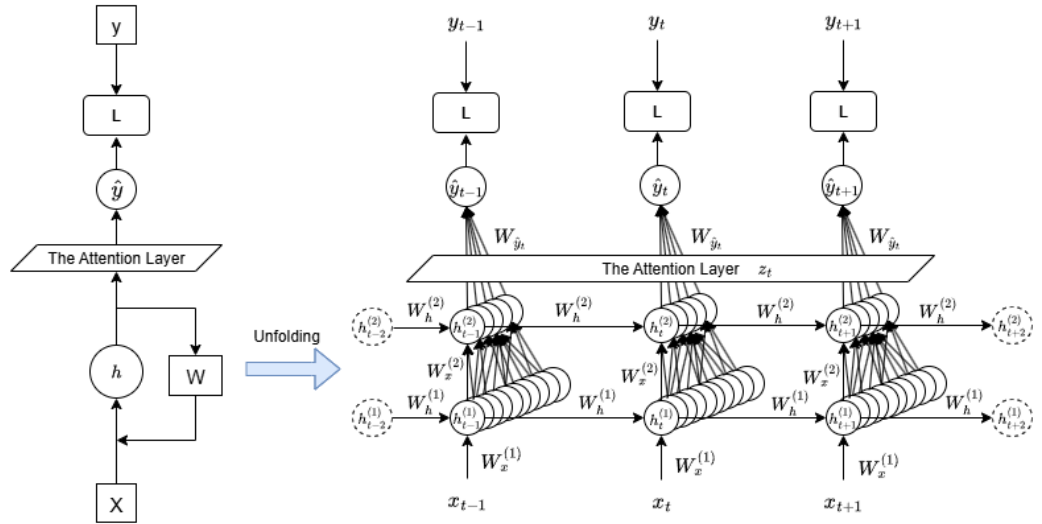


Figure 3.2: The workflow diagram of the RNN attention models. In this figure, $W_h^{(1)}$, $W_h^{(2)}$, $W_x^{(1)}$ and $W_x^{(2)}$ are learnable parameter matrices for each layer, X and y are pretrained input and labels respectively. h indicates the hidden layers of the RNN structure, $\hat{y}$ is the output derived from the attention layer after linear transformation, and L represents the loss function. The ‘Attention layer’ is the attention layer that contains an attention mechanism (e.g. additive attention mechanism or one of Luong’s attention mechanisms). The left side of the figure is the main workflow of the RNN attention models, and the right side is the 3-Dimension timeline-based unfolded workflow diagram. The circles indicate the nodes of each hidden layer. Each node is fully connected to all nodes in the next layer.

3.4.1 MLP autoencoder for pertaining

The MLP autoencoder is commonly used for feature engineering in the machine learning and financial time series forecasting context, especially for dimensional reduction and data

denoising purposes (Akhtar et al., 2017; Sagiraju and Mogalla, 2021; Bao et al., 2017b; Bieganski and Slepaczuk, 2024; Heaton et al., 2017). This study deploys the MLP autoencoder as a pre-training method for input factors' dimensional reduction and missing value filling. Figure 3.3 shows the MLP autoencoder module deployed for this study. The MLP autoencoder encompasses an input layer, an output layer (reconstructed layer in this case) and a latent layer (latent space or bottleneck layer) in between. The process from the input layer to the latent layer is named 'encoding', and from the latent layer to the reconstructed (output) layer is named 'decoding' in an autoencoder framework (Rumelhart et al., 1986b). In the asset pricing research of this chapter, the original input factors are denoted as f , $f \in \mathbb{R}^n$ and the reconstructed factors are recognized as $\hat{f}$, $\hat{f} \in \mathbb{R}^n$. Through an MSE loss function L , the reconstructed factors can be trained to closely approach the actual input factors. The layer in between with blue nodes is the latent layer, where it subtracts the new factor with a lower dimension. The parameters which lead to the minimum MSE of the loss function are used to generate new factors. The new factors are denoted as x , $x \in \mathbb{R}^m$, $m < n$. Therefore, the encoding process can be mathematically presented as:

$$\mathbf{X} = g(\mathbf{W}_{\text{en}}\mathbf{F} + \mathbf{b}_{\text{en}}) \quad (3.3)$$

where $\mathbf{F} = [\mathbf{f}_1, \mathbf{f}_2, \dots, \mathbf{f}_n] \in \mathbb{R}^{n \times T}$ is the original input factor matrix, T is the time length, while $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m] \in \mathbb{R}^{m \times T}$ is the latent factor matrix. $\mathbf{W}_{\text{en}} \in \mathbb{R}^{m \times n}$ and $\mathbf{b}_{\text{en}} \in \mathbb{R}^{m \times 1}$ are the weight matrix and bias of the encoder respectively. $g(\cdot)$ is the Rectified Linear Unit (ReLU) activation function $\text{ReLU}(x) = \max(0, x)$. The function of the encoding is to lower the dimension of the original input factors.

The function of the decoding is opposite to the function of the encoding. It attempts to reconstruct the original input factors. It utilizes a loss function to measure the accuracy of the reconstructed factors, and it can be presented as:

$$\hat{\mathbf{F}} = g(\mathbf{W}_{\text{de}}\mathbf{X} + \mathbf{b}_{\text{de}}) \quad (3.4)$$

where $\mathbf{W}_{\text{de}} \in \mathbb{R}^{n \times m}$ and $\mathbf{b}_{\text{de}} \in \mathbb{R}^{n \times 1}$ are the decoder weight matrix and bias, respectively.

$\hat{\mathbf{F}} = [\hat{\mathbf{f}}_1, \hat{\mathbf{f}}_2, \dots, \hat{\mathbf{f}}_T] \in \mathbb{R}^{n \times T}$ are the estimated original factors. Considering the balance of the information effectiveness and the noise level of the generated latent factors, the neuron number is configured 70% of the original input.

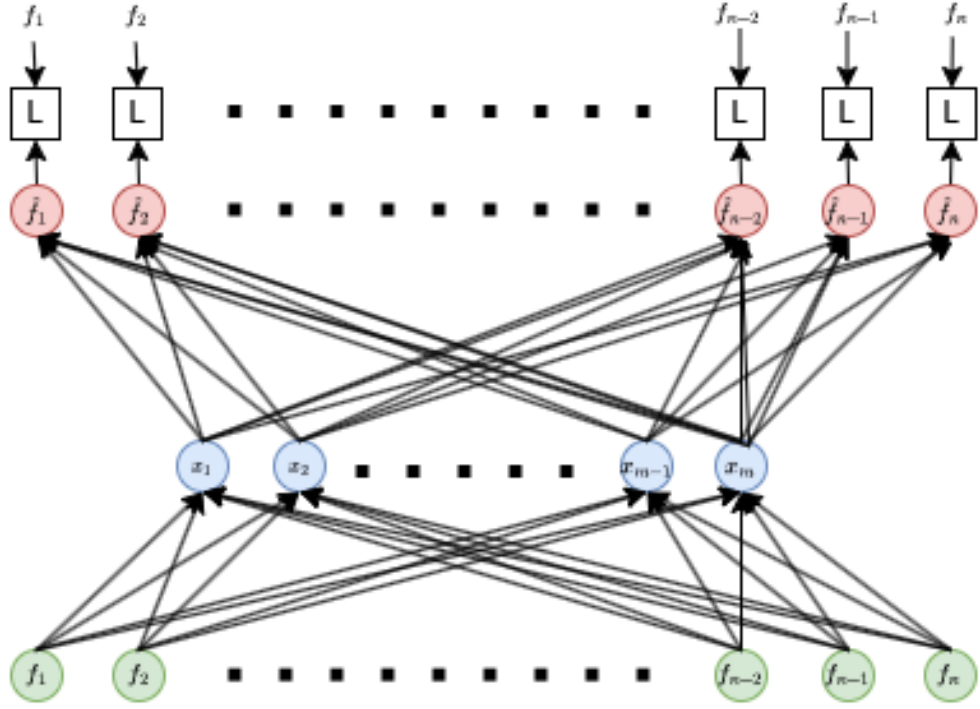


Figure 3.3: MLP autoencoder as the pre-training module. In this figure, f_1, f_2 to $f_n, n \in \mathbb{Z}^+$ are denoted as original factors. L is the loss function, MSE in this case. $\hat{f}_1, \hat{f}_2$ to $\hat{f}_n, n \in \mathbb{Z}^+$ are the reconstructed factors. The bottom layer with green nodes is the input layer, while the top layer with red nodes is the reconstructed layer. The layer in between with blue nodes named latent space of the autoencoder, which generates the new factors with a lower dimension, where x_1 to $x_m, m \in \mathbb{Z}^+$ denoted as new factors.

The mean square error (MSE) function is selected as the loss function for estimating the accuracy of factor reconstruction, which is shown in Equation (3.5). Stochastic gradient descent (SGD) algorithm with Adaptive moment estimation (Adam) optimizer is employed for the estimation, and an early-stopping method is applied for improving computational efficiency and preventing model overfitting caused by the training process. They are presented in the appendix of Chapter 3.

$$\mathcal{L}_{\text{recon}} = \frac{1}{T} \left\| \mathbf{F} - \hat{\mathbf{F}} \right\|_F^2 \quad (3.5)$$

3.4.2 Recurrent Neural Networks

The fundamental structure of the proposed model is the RNN structure (Rumelhart et al., 1986a), which is shown in the left part of Figure 3.2. The workflow chart without the sparse attention layer is the naive RNN structure. A fundamental RNN structure includes an input layer, an output layer and a single or multiple hidden layer. In this study, for a better capture of factor patterns, two hidden layers with a pyramid neuron structure are employed, specifically, the first hidden layer with 64 neurons and the second hidden layer with 32 neurons. The mathematical presentation of the multilayer RNN model is presented in Equation (3.6) to Equation (3.7), where $\mathbf{h}_t^{(1)}$ and $\mathbf{h}_t^{(2)}$ mean the output of the first hidden layer and the second hidden layer at time t . $\mathbf{h}_{t-1}^{(1)}$ and $\mathbf{h}_{t-1}^{(2)}$ are the output of hidden layers at time $t-1$, while $\mathbf{x}_t$ is the abstracted factors from pre-training module. $\mathbf{W}_h^{(1)}$, $\mathbf{W}_h^{(2)}$, $\mathbf{W}_x^{(1)}$ and $\mathbf{W}_x^{(2)}$ are the estimated parameter matrices of the previous hidden states and current inputs respectively. $\mathbf{b}_{rnn}^{(1)}$ and $\mathbf{b}_{rnn}^{(2)}$ are the bias of each hidden layers.

$$\mathbf{h}_t^{(1)} = \phi \left(\mathbf{W}_h^{(1)} \mathbf{h}_{t-1}^{(1)} + \mathbf{W}_x^{(1)} \mathbf{x}_t + \mathbf{b}_{rnn}^{(1)} \right), \quad \mathbf{h}_t^{(1)} \in \mathbb{R}^{64} \quad (3.6)$$

$$\mathbf{h}_t^{(2)} = \phi \left(\mathbf{W}_h^{(2)} \mathbf{h}_{t-1}^{(2)} + \mathbf{W}_x^{(2)} \mathbf{h}_t^{(1)} + \mathbf{b}_{rnn}^{(2)} \right), \quad \mathbf{h}_t^{(2)} \in \mathbb{R}^{32} \quad (3.7)$$

And $\phi(\cdot)$ is the hyperbolic tangent ($\tanh$) activation function.

3.4.3 Attention mechanisms for attention layer

The attention mechanism is a powerful neural network component that enables models to dynamically focus on the most relevant parts of a sequence when making predictions. It was originally developed for natural language processing (NLP), such as missing text prediction and machine translation. It employs a function to compute the attention scores to measure the relevance level between any two parts of a textual sequence, and applies the softmax function to normalize the attention scores to attention weights to measure the importance of each word. Thus, the attention mechanism can improve the textual prediction accuracy through the indicator, which shows the importance of a certain word.

The earliest attention mechanism is the additive attention mechanism from Bahdanau et al. (2015). Gradually, researchers developed different function forms for the attention mechanism. Luong’s three-form attention mechanism (Luong et al., 2015) is one of the most significant attention mechanisms. The additive attention is frequently used in recent asset pricing literature, for example, Zhou and Wang (2024) and Cong et al. (2021b). Later, the Luong’s general attention is used in the paper of Kelly et al. (2025) as a module for their Transformer model. Inspired by these works, the research in this chapter uses early-developed attention mechanisms as benchmarks for the proposed models. However, the attention mechanisms cannot directly apply to time series data such as factors and stock returns, since the textual data is sometimes bidirectional. For instance, the missing word prediction could depend not only on the previous words but also on the words behind the missing space. However, the time series forecasting only relies on historical data, which implies that bidirectional forecasting may cause the future data leakage issue. Therefore, a compulsory causal mask which can cover future data is crucial for time series attention mechanism applications. For time series attention mechanisms, the attention weights, which measure the relevance level of two time steps, are only computed between the current time step and the historical time steps, but the future time steps are covered by masks. And the attention weights are finally distributed back to the fundamental structure (e.g. RNN or LSTM) output to generate attention output, combining the information of the fundamental structure output and the importance of the information at each time step. The research on applications of attention mechanisms for finance and economics is still in the early stages. As an example, Kelly et al. (2025) deploys Luong’s general attention mechanism to their Transformer model for the asset pricing research.

Attention mechanisms have three essential elements: query, key and value. Query and key are used for computing attention scores via a certain function form, while value is used for computing the attention output. The global self-attention mechanism (Vaswani et al., 2017) and sliding window attention mechanism (Beltagy et al., 2020) have explicit values since they have a specific function form for the element of ‘value’, but alternative attention mechanisms

have hidden value equal to the historical output of the foundation neural network structure. The additive attention (Bahdanau et al., 2015), which employs a feedforward network to compute alignment scores, is introduced in Section 3.4.3.1. The multiplicative (or general) attention (Luong et al., 2015), which uses linear forms to compute the similarity between queries and keys, is introduced in Section 3.4.3.2. Scaled dot-product attention (Luong et al., 2015; Vaswani et al., 2017) further improves numerical stability and training efficiency by introducing a scaling factor, and it serves as the foundation of the most prevailing large language model named Transformer. This is introduced in Section 3.4.3.3 and Section 3.4.3.5.

3.4.3.1 Additive attention mechanism

Bahdanau et al. (2015)'s additive attention is the earliest attention mechanism originally designed for natural language processing (NLP) tasks such as machine translation. As a global attention mechanism, it measures the difference between any two time steps and distributes attention weights for every time step. It is suitable for capturing the most relevant information from context, which could increase the prediction accuracy of the missing words within a sentence. However, this causes the future information leakage if the attention mechanism is applied directly to the time series data without a causal mask. Let $\mathbf{q}_t, \mathbf{k}_j \in \mathbb{R}^{d_{RNN}}$ be the query and key of at time step of t and j , and define:

$$\mathbf{q}_t = \mathbf{W}_q \mathbf{h}_t^{(2)}, \quad \mathbf{k}_j = \mathbf{W}_k \mathbf{h}_j^{(2)} \quad (3.8)$$

where $\mathbf{W}_q, \mathbf{W}_k \in \mathbb{R}^{d_{RNN} \times d_{RNN}}$ are the learnable parameter matrices of $\mathbf{h}_t^{(2)}, \mathbf{h}_j^{(2)} \in \mathbb{R}^{d_{RNN}}$. Then the additive attention score for time t and j , $e_{t,j}$, can be presented as:

$$e_{t,j} = \mathbf{v}^\top \tanh(\mathbf{q}_t + \mathbf{k}_j) \quad (3.9)$$

where $\mathbf{v} \in \mathbb{R}^{d_{RNN}}$ is a learnable parameter vector. Considering the input is the time series data, the compulsory causal mask should be applied to force time step j to be earlier than t .

3.4.3.2 Luong's attention mechanism

Luong's attention is developed from the additive attention. Luong et al. (2015) proposes three attention weights computation methods for boosting the computational efficiency and forecasting accuracy of Bahdanau et al. (2015)'s additive attention. It employs the dot product, linear and concatenation for computing the attention weights, respectively, which are defined as 'Luong dot-product (L-DotProd) attention', 'Luong general (L-General) attention' and 'Luong concatenation (L-Concat) attention'. For the convenience of the mathematical presentations, they are denoted as 'LD', 'LG' and 'LC' respectively.

For Luong dot-product attention, the query $\mathbf{q}_t$ and key $\mathbf{k}_j$ is equal to the RNN output at time t and j , which shows in Equation (3.10)

$$\mathbf{q}_t^{LD} = \mathbf{h}_t^{(2)}, \quad \mathbf{k}_j^{LD} = \mathbf{h}_j^{(2)} \quad (3.10)$$

The Luong dot-product attention score $e_{t,j}$ is the scaled dot product of query $\mathbf{q}_t$ and key $\mathbf{k}_j$,

$$e_{t,j}^{LD} = \frac{1}{\sqrt{d_{RNN}}} (\mathbf{q}_t^{LD})^\top \mathbf{k}_j^{LD} \quad (3.11)$$

$$= \frac{1}{\sqrt{d_{RNN}}} (\mathbf{h}_t^{(2)})^\top \mathbf{h}_j^{(2)} \quad (3.12)$$

For Luong's general attention mechanism, it distributes a learnable parameter matrix $\mathbf{W} \in \mathbb{R}^{d_{RNN} \times d_{RNN}}$ to the key $\mathbf{k}_j$ but keep the query $\mathbf{q}_t$ identical to the one in Luong dot-product attention. It projects key to the query linearly by adding the parameter matrix $\mathbf{W}$. Equation (3.13) shows the query and key for Luong's general attention mechanism.

$$\mathbf{q}_t^{LG} = \mathbf{h}_t^{(2)}, \quad \mathbf{k}_j^{LG} = \mathbf{W} \mathbf{h}_j^{(2)} \quad (3.13)$$

Thus, the attention score of Luong's general attention can be:

$$e_{t,j}^{LG} = \frac{1}{\sqrt{d_{RNN}}} (\mathbf{q}_t^{LG})^\top \mathbf{W} \mathbf{k}_j^{LG} \quad (3.14)$$

For Luong's concatenation attention mechanism, the query $\mathbf{q}_t^{LC}$ and key $\mathbf{k}_j^{LC}$ have their

own learnable parameter matrix which denoted as $\mathbf{W}_q$ and $\mathbf{W}_k$ respectively, $\mathbf{W}_q, \mathbf{W}_k \in \mathbb{R}^{d_{RNN} \times d_{RNN}}$. Concretely, the query and key can be presented as:

$$\mathbf{q}_t^{LC} = \mathbf{W}_q \mathbf{h}_t^{(2)}, \quad \mathbf{k}_j^{LC} = \mathbf{W}_k \mathbf{h}_j^{(2)} \quad (3.15)$$

Different from all alternative attention mechanisms, Luong's concatenation attention applies a hyperbolic tangent function on the concatenated query and key for calculating the attention scores. The Luong's concatenation attention score can be computed as:

$$e_{t,j}^{LC} = \mathbf{v}^\top \tanh([\mathbf{q}_t^{LC}; \mathbf{k}_j^{LC}]) \quad (3.16)$$

where $\mathbf{v} \in \mathbb{R}^{2d_{RNN}}$ is the learnable parameter vector for calculating the concatenation attention scores.

3.4.3.3 Global self-attention RNN model

The global self-attention is abstracted from Vaswani et al. (2017)'s classic Transformer model. In the additive attention mechanism and Luong's attention mechanisms, the format of 'query' and 'key' varies accordingly, but the third element 'value' v is hidden and always equal to the output of the RNN model h_j . To prevent any misunderstanding, in the previous sections, the calculation of attention output z_t directly adopts the notation $h_j^{(2)}$ to multiply the attention weights $\alpha_{t,j}$. However, in the self-attention mechanism, the element v has its variable format. Specifically,

$$\mathbf{q}_t = \mathbf{W}_q \mathbf{h}_t^{(2)}, \quad \mathbf{k}_j = \mathbf{W}_k \mathbf{h}_j^{(2)}, \quad \mathbf{v}_j = \mathbf{W}_v \mathbf{h}_j^{(2)} \quad (3.17)$$

where $\mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v \in \mathbb{R}^{d_{RNN} \times d_{RNN}}$ are learnable parameter matrix for the query $\mathbf{q}_t$, key $\mathbf{k}_j$ and $\mathbf{v}_j$. And $h_j^{(2)} \in \mathbb{R}^{d_{RNN}}$ is the output at time j of the RNN structure. For each time step t and j , the attention score $e_{t,j}$ can be computed through Equation (3.18).

$$e_{t,j} = \frac{1}{\sqrt{d_{RNN}}} \mathbf{q}_t^\top \mathbf{k}_j \quad (3.18)$$

3.4.3.4 Causal masks, attention weights and attention outputs for global attentions

Causal masks are configured for preventing the future data leakage of time series data. After attention scores ($e_{t,j}$) are computed, the causal masks can be mathematically presented as:

$$e_{t,j} = \begin{cases} e_{t,j}, & j \leq t \\ -\infty, & j > t \end{cases} \quad (3.19)$$

Thus, the attention weights $\alpha_{t,j}$ can be computed via softmax function:

$$\alpha_{t,j} = \frac{\exp(e_{t,j})}{\sum_{i=1}^t \exp(e_{t,i})} \quad (3.20)$$

Furthermore, the attention outputs can be computed according to the foundation model outputs, either the hidden status from RNN structure ($h_j^{(2)}$) or the element of value (self-attention) computed from $h_j^{(2)}$ (v):

$$z_t = \sum_{j=1}^t \alpha_{t,j} h_j^{(2)} \quad (3.21)$$

or for self-attentions,

$$z_t = \sum_{j=1}^t \alpha_{t,j} v \quad (3.22)$$

3.4.3.5 Sliding window sparse attention mechanism

Sliding window sparse attention belongs to the self-attention family, and it is first proposed by Beltagy et al. (2020). Like other sparse attention mechanisms, it is developed for capturing the longer-term temporal dependency of the sequential data and lowering the computational cost of global attention mechanisms such as Bahdanau's attention (Bahdanau et al., 2015), Luong's attention (Luong et al., 2015) and Vaswani's self-attention (Vaswani et al., 2017). The difference between the sliding window attention and Vaswani's self-attention is Vaswani's self-attention calculates the difference between any two time steps, but the sliding window

attention of Beltagy et al. (2020)'s only calculates the difference between one time step and a range of time steps close to that time step. Figure 3.4 shows the difference between Vaswani's self-attention and sliding window sparse attention with causal masks for time series data. The left panel of Figure 3.4 shows an example of a sliding window sparse attention mechanism when the attention window is equal to 4, and the right panel shows the global self-attention mechanism from Vaswani et al. (2017). The global self-attention calculates the difference between the series at time step t and all available time steps previous to t , but the sliding window sparse attention only measures the difference between the series at time step t and available time steps previous to t up to the restrictions of the attention window, which is $t - 3$ in this example.

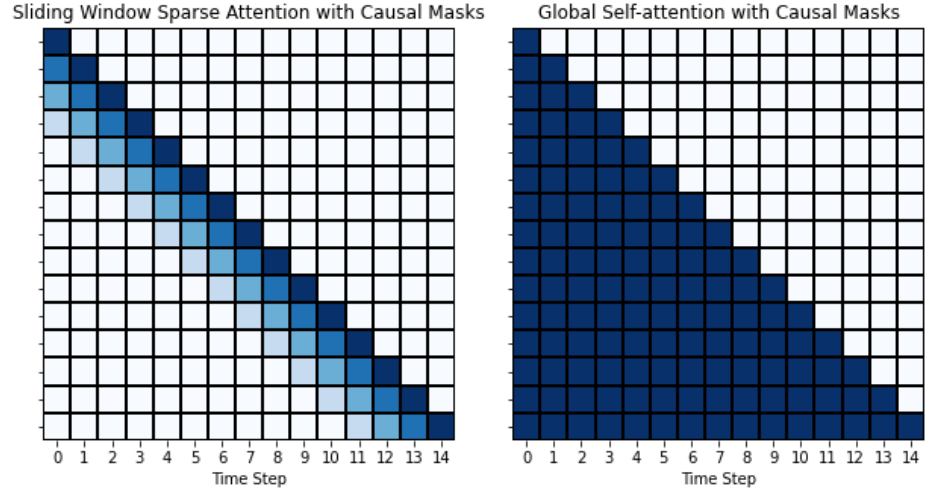


Figure 3.4: The comparison of sliding window sparse attention and global self attention (with causal masks). This is an example when the attention window is equal to 4. The white grids in both panels represent the time points with causal masks, which means they are not in the calculation of attention weights.

The sliding window sparse attention is developed from the global self-attention, which is shown in Figure 3.5. Mathematically, to each time step t , defining the attention window $\mathcal{S}_t = \{s \mid \max(0, t - w) \leq s \leq t\}$, s is the time step previous to time t . Thus, the query (q), key (k) and value (v) here of sliding window sparse attention in time step t and s are:

$$q_t = W_q h_t, \quad k_s = W_k h_s, \quad v_s = W_v h_s \quad (3.23)$$

where $h_t, h_s \in \mathbb{R}^{d_{RNN}}$ are the output from RNN second hidden layer at time t and s respec-

tively. $W_q, W_k, W_v \in \mathbb{R}^{d_{RNN} \times d_{RNN}}$ are the learnable weight matrices of q_t, k_s, v_s . Therefore, the attention score between time t and s can be:

$$\alpha_{t,s} = \frac{\exp(q_t^\top k_s / \sqrt{d_{RNN}})}{\sum_{j \in \mathcal{S}_t} \exp(q_t^\top k_j / \sqrt{d_{RNN}})} \quad (3.24)$$

Further, the attention output of time t denoted as z_t can be:

$$z_t = \sum_{s \in \mathcal{S}_t} \alpha_{t,s} v_s \quad (3.25)$$

Due to the limited data size of this study, the proposed model only considers the single-head sliding window sparse attention. With the attention window, the computational cost of this attention becomes $O(T \times \mathcal{S}_t)$.

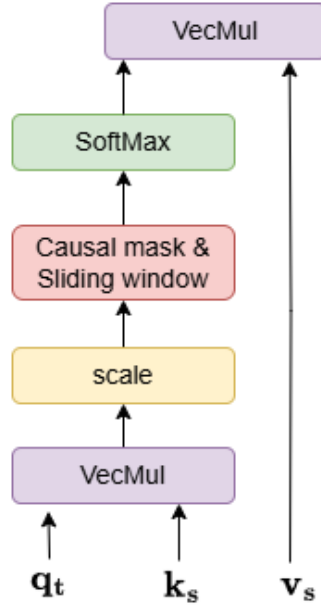


Figure 3.5: Sliding window sparse attention mechanism workflow diagram. $q_t, k_s, v_s \in \mathbb{R}^{d_{RNN}}$ are query, key, value at time t and s respectively. 'VecMul' is denoted as the vector-wise multiplication.

The main difference between the classic Transformer model (Vaswani et al., 2017) and the Longformer model (Beltagy et al., 2020) is that the latter employs the assembled attention mechanism which encompasses sparse attention of sliding window attention, dilated sliding

window attention (optional) and global attention. For simplifying the model structure for the temporal sparse data, such as the monthly stock pricing data, the proposed model only deploys the sliding window sparse attention layer from the Longformer model and assembles it on an RNN frame for a better capture of the temporal longer-term dependency that exists in the stock pricing data.

To summarize, the calculation of attention output follows the procedure of computing the attention scores, applying the softmax function on attention scores for generating the attention weights and utilizing attention weights to multiply the ‘value’ for the attention output.

3.4.4 Linear output of the RNN attention models

Finally, the attention outputs are utilized to compute the estimated label through the linear function. It is shown in Equation (3.26), where $\hat{y}_t$ is the linearly estimated label, which is the excess return of an individual stock at time t .

$$\hat{y}_t = \mathbf{W}_{\hat{y}_t} \mathbf{z}_t + b_{\hat{y}_t}, \quad \hat{y}_t \in \mathbb{R} \quad (3.26)$$

3.4.5 Long short-term memory (LSTM) and gated recurrent unit (GRU) as benchmark models

The RNN variation models, LSTM and GRU, serve as benchmark models in this study. Due to the gradient explosion and vanishing issues that exist in the naive RNN model, which was suggested by Bengio et al. (1994), Hochreiter and Schmidhuber (1997) proposes the first LSTM model for capturing a longer dependency of sequential data. They design the input gate, output gate, and cell structure, and embed them in the naive RNN framework to moderate the gradient explosion and vanishing issues. It is improved by Gers et al. (2000), who adds a forget gate in the previous LSTM structure. This LSTM is the most commonly applied LSTM in the literature of different domains. Further, people find LSTM is over-parameterized since it has three gates, one cell and one hidden state, albeit it moderates the gradient explosion and vanishing issues to a degree. Thus, Cho et al. (2014) proposes

the GRU which simplifies the structure of LSTM. GRU only configure two gates: update gate and reset gate, which highly reduces the parameters of a recurrent neural network model.

In the early stage, LSTM was developed for speech recognition, sequential labelling, time series forecasting and handwritten character recognition, while GRU was developed for machine translation originally. However, researchers have proven that GRU performs similarly to LSTM in prediction accuracy from different angles (Jozefowicz et al., 2015; Galphade et al., 2022; Pudikov and Brovko, 2020), especially in the time series forecasting context. Figure 3.6 exhibits the entire workflow of the two-layer LSTM adopted in this study from a bird's-eye view. The first LSTM or GRU layer ('LSTM/GRU 1') contains 64 neurons; thus, the dimension of the first layer is 64. And the second LSTM or GRU layer ('LSTM/GRU 2') contains 32 neurons, which indicates the dimension of the second LSTM or GRU layer is 32. For time series forecasting tasks, both LSTM and GRU frame connect a linear layer as the output layer. Briefly, a two-layer LSTM with distinctive neuron numbers can be generalized

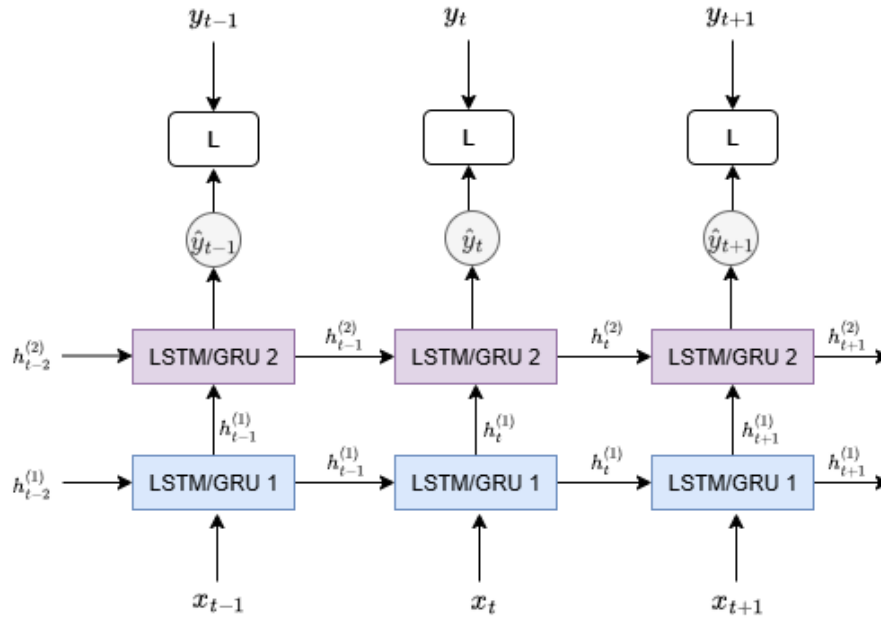


Figure 3.6: Entire workflow chart for two-layer LSTM and GRU model. x is the abstracted factors from the MLP pretraining process, while h is the output from the LSTM or GRU layer. $\hat{y}$ is the estimated output from the linear layer, and L means the loss function. y is the actual value of the label, the stock excess returns in this case.

mathematically presented as:

$$\mathbf{i}_t^{(l)} = \sigma(W_i^{(l)} \mathbf{h}_t^{(l-1)} + U_i^{(l)} \mathbf{h}_{t-1}^{(l)} + \mathbf{b}_i^{(l)}) \quad (3.27)$$

$$\mathbf{f}_t^{(l)} = \sigma(W_f^{(l)} \mathbf{h}_t^{(l-1)} + U_f^{(l)} \mathbf{h}_{t-1}^{(l)} + \mathbf{b}_f^{(l)}) \quad (3.28)$$

$$\mathbf{o}_t^{(l)} = \sigma(W_o^{(l)} \mathbf{h}_t^{(l-1)} + U_o^{(l)} \mathbf{h}_{t-1}^{(l)} + \mathbf{b}_o^{(l)}) \quad (3.29)$$

$$\tilde{\mathbf{c}}_t^{(l)} = \tanh(W_c^{(l)} \mathbf{h}_t^{(l-1)} + U_c^{(l)} \mathbf{h}_{t-1}^{(l)} + \mathbf{b}_c^{(l)}) \quad (3.30)$$

$$\mathbf{c}_t^{(l)} = \mathbf{f}_t^{(l)} \odot \mathbf{c}_{t-1}^{(l)} + \mathbf{i}_t^{(l)} \odot \tilde{\mathbf{c}}_t^{(l)} \quad (3.31)$$

$$\mathbf{h}_t^{(l)} = \mathbf{o}_t^{(l)} \odot \tanh(\mathbf{c}_t^{(l)}) \quad (3.32)$$

The l -th layer, where $l \in \{1, 2, \dots, L\}$. The $\mathbf{h}_t^{(l-1)}$ denotes the input to the l -th layer at time t . For the first layer ($l = 1$), the input is the original sequence data, meaning $\mathbf{h}_t^{(0)} = \mathbf{x}_t$. The $\mathbf{i}_t^{(l)}$, $\mathbf{f}_t^{(l)}$, and $\mathbf{o}_t^{(l)}$ are the input gate, forget gate, and output gate of the l -th layer, respectively. $\tilde{\mathbf{c}}_t^{(l)}$ and $\mathbf{c}_t^{(l)}$ are the candidate cell state and the cell state. $\sigma(\cdot)$ is the sigmoid activation function, and $\tanh(\cdot)$ is the hyperbolic tangent function. $W_i^{(l)}, W_f^{(l)}, W_o^{(l)}, W_c^{(l)} \in \mathbb{R}^{d_l \times d_{l-1}}$ and $U_i^{(l)}, U_f^{(l)}, U_o^{(l)}, U_c^{(l)} \in \mathbb{R}^{d_l \times d_l}$ are the learnable parameter matrices for each gate and cell, where d_l is the dimension of the l -th layer (the number of neurons). Note that $d_0 = d_{\text{in}}$, indicating the dimension of the original input, which are the extracted factors from the pre-training process. $\mathbf{b}_i^{(l)}, \mathbf{b}_f^{(l)}, \mathbf{b}_o^{(l)}, \mathbf{b}_c^{(l)} \in \mathbb{R}^{d_l}$ are the bias vectors for input gate, forget gate, output gate and cell status, respectively. And $\odot$ denotes the Hadamard product. Extensively, the paper Gers et al. (2000) discussed the technical details for the LSTM model.

GRU simplifies the LSTM by substituting the input gate, forget gate, output gate and cell states system with the update gate and reset gate system. A multi-layer GRU can be presented as:

$$\mathbf{up}_t^{(l)} = \sigma(W_{up}^{(l)} \mathbf{h}_t^{(l-1)} + U_{up}^{(l)} \mathbf{h}_{t-1}^{(l)} + \mathbf{b}_{up}^{(l)}) \quad (3.33)$$

$$\mathbf{re}_t^{(l)} = \sigma(W_{re}^{(l)} \mathbf{h}_t^{(l-1)} + U_{re}^{(l)} \mathbf{h}_{t-1}^{(l)} + \mathbf{b}_{re}^{(l)}) \quad (3.34)$$

$$\tilde{\mathbf{h}}_t^{(l)} = \tanh(W_h^{(l)} \mathbf{h}_t^{(l-1)} + U_h^{(l)} (\mathbf{re}_t^{(l)} \odot \mathbf{h}_{t-1}^{(l)}) + \mathbf{b}_h^{(l)}) \quad (3.35)$$

$$\mathbf{h}_t^{(l)} = (1 - \mathbf{up}_t^{(l)}) \odot \mathbf{h}_{t-1}^{(l)} + \mathbf{up}_t^{(l)} \odot \tilde{\mathbf{h}}_t^{(l)} \quad (3.36)$$

The l -th layer, where $l \in \{1, 2, \dots, L\}$ and $L = 2$ in the benchmark GRU architecture. $\mathbf{h}_t^{(l-1)}$ denote the input to the l -th layer at time t . For the initial layer ($l = 1$), the input is the original sequence data, meaning $\mathbf{h}_t^{(0)} = \mathbf{x}_t$. Where $\mathbf{up}_t^{(l)}$ and $\mathbf{re}_t^{(l)}$ represent the update gate and reset gate of the l -th layer, respectively. $\tilde{\mathbf{h}}_t^{(l)}$ denotes the candidate hidden state, and $\mathbf{h}_t^{(l)}$ is the final hidden state output of the l -th layer at time t . $\sigma(\cdot)$ is the sigmoid activation function, and $\tanh(\cdot)$ is the hyperbolic tangent function. The weight matrices $W_{up}^{(l)}, W_{re}^{(l)}, W_h^{(l)} \in \mathbb{R}^{d_l \times d_{l-1}}$ and $U_{up}^{(l)}, U_{re}^{(l)}, U_h^{(l)} \in \mathbb{R}^{d_l \times d_l}$ are the learnable parameters of the gates and hidden states for the l -th layer, where d_l is the dimension (number of neurons) of the l -th layer. It is important to note that $d_0 = d_{in}$, indicating the dimension of the input of the extracted factors. The terms $\mathbf{b}_{up}^{(l)}, \mathbf{b}_{re}^{(l)}, \mathbf{b}_h^{(l)} \in \mathbb{R}^{d_l}$ correspond to the bias vectors for the l -th layer, and $\odot$ denotes the Hadamard product.

Similar to LSTM, GRU employs a gated system to control the memory length. It removes the cell states of LSTM but embeds the candidate hidden state for a substitution of candidate cell state of LSTM which highly reduces the parameter numbers. In the first GRU layer, the updated gate and reset gate are computed according to the input factors (x_t) computed via the MLP autoencoder pretraining process and previous hidden states with the sigmoid activation function. The candidate hidden state is computed through the input and reset gated previous hidden state with the hyperbolic tangent activation function. The output hidden state is computed via update gated hidden state and candidate hidden state. Subsequently, the hidden state computed from the first layer is passed to the second layer for the process to compute the final hidden state which is for derivation of estimated label $\hat{\mathbf{y}}_t$ through a linear function shows in Equation (3.37).

$$\hat{\mathbf{y}}_t = \mathbf{W}_{\hat{\mathbf{y}}_t} \mathbf{h}_t^{(2)} + \mathbf{b}_{\hat{\mathbf{y}}_t} \quad (3.37)$$

where $\mathbf{h}_t^{(2)} \in \mathbb{R}^{d_2}$ is the output hidden state of second LSTM or GRU layer, while $\mathbf{W}_y \in \mathbb{R}^{d_{out} \times d_2}$ and $\mathbf{b}_y \in \mathbb{R}^{d_{out}}$ are the parameter matrix and bias of linear layer, where d_{out} is the output dimension. $\hat{\mathbf{y}}_t \in \mathbb{R}^{d_{out}}$ is the estimated label $\hat{\mathbf{y}}_t$ at time t . In the context of asset pricing, the estimated output $\hat{\mathbf{y}}_t$ is the individual stock excess returns $\hat{\mathbf{r}}_t$.

3.4.6 Estimation

For estimating the main module, RNN with an attention layer, of the proposed model, the MSE function is selected as the loss function, and the L1 regularization term is added to the loss function as a regularization penalty. Concretely, it can be presented as:

$$\mathcal{L} = \frac{1}{T} \sum_{t=1}^T (y_t - \hat{y}_t)^2 + \lambda \|\mathbf{W}\|_1 \quad (3.38)$$

where $\mathcal{L}$ is denoted as loss function, T is the length of the entire time window and y_t is the actual value of the label at time t . $\mathbf{W}$ represents all parameters in the model, and λ is the regularization coefficient which requires pre-setting. The best parameters for prediction can be derived from minimizing the loss function via the stochastic gradient descent (SGD) method or the adaptive moment estimation (Adam) optimizer for improving the model training efficiency and mitigating gradient explosion and vanishing problems. The SGD and Adam optimizer are recruited for the estimation task, which is shown in the appendix of Chapter 3.

$$\mathbf{W}^* = \arg \min_{\mathbf{W}} \left\{ \frac{1}{T} \sum_{t=1}^T (y_t - \hat{y}_t)^2 + \lambda \|\mathbf{W}\|_1 \right\} \quad (3.39)$$

Equation (3.39) depicts how to derive the best parameters from minimizing the MSE loss function, where $\mathbf{W}^*$ is denoted as the best parameters selected for forecasting.

3.4.7 Forecasting

One-step ahead forecasting for the RNN attention models can be derived from Equation (3.40)

$$\hat{y}_{t+1} = \mathbf{W}_{\hat{y}_t} \mathbf{z}_{t+1} + b_{\hat{y}_t} \quad (3.40)$$

And the one-step ahead forecasting for the RNN, LSTM and GRU models is

$$\hat{\mathbf{y}}_{t+1} = \mathbf{W}_{\hat{\mathbf{y}}_t} \mathbf{h}_{t+1}^{(2)} + b_{\hat{\mathbf{y}}_t} \quad (3.41)$$

It applies the best weights and biases to the outputs from the main models to derive the estimated stock excess return at time $t + 1$.

Similar to Chapter 2, the recurrent neural network models require intensive computational power; they are still infeasible in laptops or desktops, which are only equipped with CPUs. Therefore, they were implemented via Python (v3.8) using PyTorch (Paszke et al., 2019) packages, such as ‘nn.RNN’, ‘nn.LSTM’ and ‘nn.GRU’. The sliding window sparse attention mechanism is from the ‘longformer’ package of the ‘transformers’ (Wolf et al., 2020) provided by Hugging Face, specifically the function of the ‘LongformerSelfAttention’. Training was conducted on NVIDIA A40 and H100 GPUs provided by the Viking HPC cluster at the University of York.

3.5 Empirical results

The model performance and backtesting performance evaluation follow the method employed in Chapter 2. The static transaction cost (TC) of 50 bps (Jegadeesh and Titman, 1993; Lesmond et al., 2004; Grundy and Martin, 2001) is used for backtesting, and the turnover-based transaction cost of 20 bps (large-cap stocks) (Frazzini et al., 2018; Novy-Marx and Velikov, 2016; Hou et al., 2020) is used for the robustness examination to investigate how the turnover rate influences the profitability of these models. Figure 3.7 shows the price index of the three periods, which depicts the market fluctuations during the three periods. Concretely, Pre-COVID-19 Period (1911) includes a mild uptrend, COVID-19-Inclusive Period (2112) encompasses Pre-COVID-19 Period (1911) plus a volatile but sharp uptrend, and Period Including COVID-19 and One-Year After (2212) covers COVID-19-Inclusive Period (2112) plus a phase with no significant trend but extreme volatile sideways movement.

3.5.1 Model performance

Table 3.2 displays the model performance indicators for all periods. It includes the average OOS R^2 , MSE , α and its t-statistics to show whether it is significantly different from zero. From the average OOS R^2 of all periods, which indicates the out-of-sample fitness of the



Figure 3.7: Market-cap-weighted (MC) Price Index.

models, we can see that all models have positive OOS R^2 values. Surprisingly, the vanilla RNN model obtains the highest OOS R^2 , which is 11.37% in Period ‘1911’, followed by the RNN global self-attention (self.att) model (2.8%) and the RNN sliding window attention (sparse.att) model (2.45%). In Period ‘2112’, all models’ fitness is slightly lower, apart from the vanilla RNN model and GRU. The RNN model (14.01%) still outperforms the alternative models significantly, followed by the RNN global self-attention model (2.02%) and the RNN sliding window attention model (1.94%). Model GRU’s R^2 increases considerably from 0.91% to 1.71%. For Period ‘2212’, the vanilla RNN model remains the model with the highest OOS R^2 value, which is 14.32% followed by the RNN global attention model (1.64%) and the RNN sliding window attention model (1.89%). The OOS R^2 of GRU and RNN continuously increases across the three periods. They exhibit persistence in the extreme market turbulence. The lowest OOS R^2 is shown by Luong’s concatenated attention model (L-Concat) in all periods, worse than the LSTM model. The vanilla RNN exhibits the best predictive power during the extreme market conditions, and even shows strong persistence in the period with steep drawdown (Period 2212). which contains a mild uptrend. Interestingly, the proposed self-attention models (self.att and sparse.att) show great fitness in all periods, but self.att decays quickly during the extreme fluctuation of markets. Instead, the sparse.att presents significant stability during the extreme conditions, and performs better than the self.att

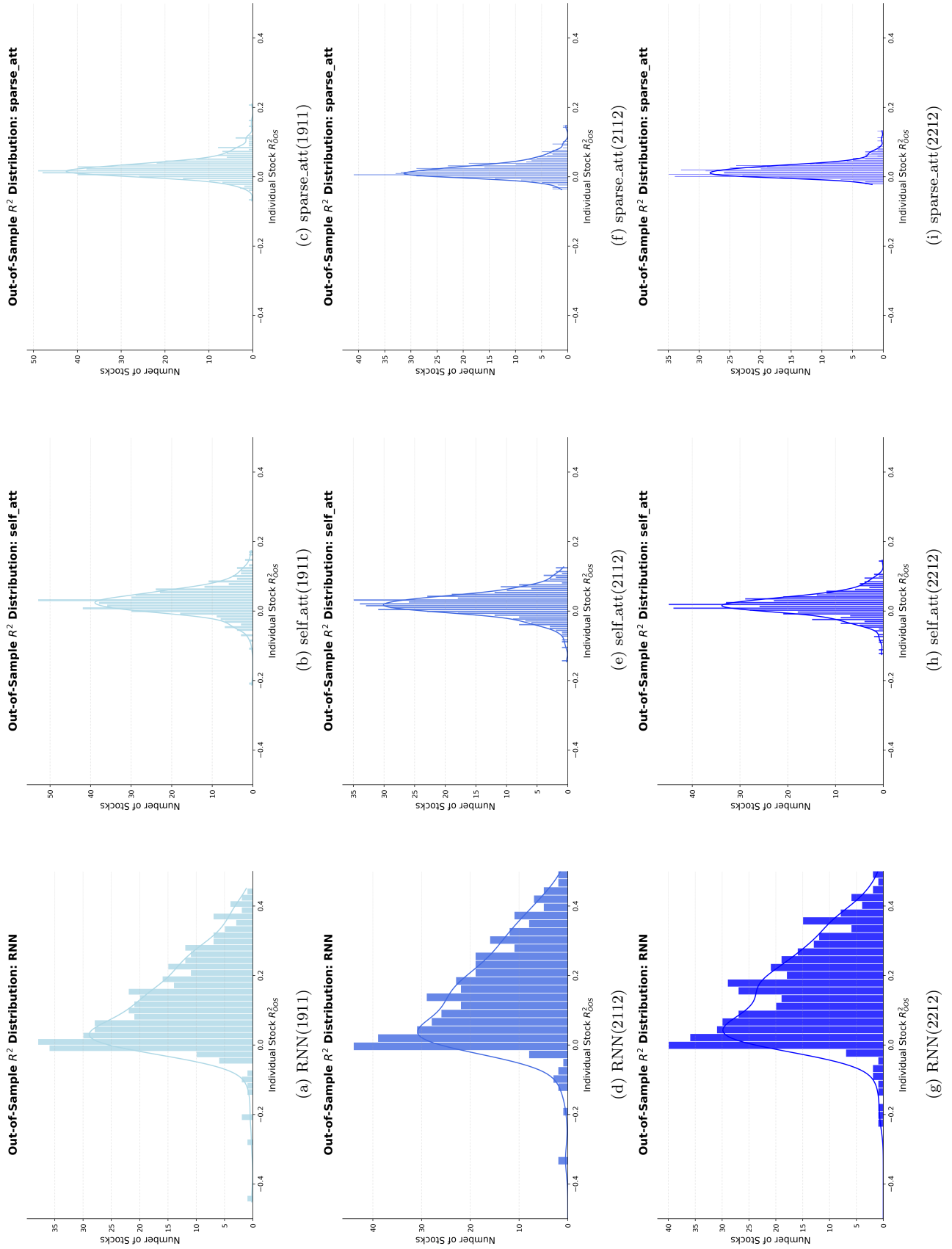
model when a sharp drawdown (Period 2212) happens. Among the rest of the benchmark attention models, Luong’s dot-product (L-DotProd) attention model is superior to the alternative benchmarks, followed by Luong’s general attention (L-General) model and additive attention (B-Additive) model, but similar to self_att, it decays slightly during the extreme market fluctuations. This provides strong evidence that higher-parameterized models have a higher risk of overfitting. It is crucial to control the parameter size in ML-based asset pricing research. Overall, the neural network models with the temporal-capturing mechanism exhibit noticeable persistence during extreme market turbulence. Average OOS MSE agrees with the findings from the average OOS R^2 .

Figure 3.8 and Figure 3.9 are the OOS R^2 and MSE distribution plots of models with the best OOS fitness. The OOS R^2 distribution plots show that the vanilla RNN model has the majority of stocks with the positive OOS R^2 during all periods, where they are distributed between 0 to 0.45. There are some similarities between the distribution of the RNN global self-attention and the RNN sliding window attention model: more than 60% of the stocks have the positive OOS R^2 values, and both of them are right-skewed in all periods. However, the RNN sliding window attention (sparse_att) model is more concentrated than the RNN global attention model (self_att), which explains why the sparse_att model performs more stably than the self_att model. The MSE distribution plots agree with the findings from the OOS R^2 distribution plots.

All models have significant positive average α s during all periods. This demonstrates the persistence of the extra capital gain beyond the factors. Economically, this indicates that there are excess returns that cannot be fully explained by the factors, which implies the unexplored or omitted factors or the existence of persistent arbitrage opportunities in the market, but it is unlikely to be caused by model misspecification. Three reasons stand for this interpretation. Firstly, the notably high α values observed during the ‘2112’ period were characterized by a sharp market uptrend, which likely contributed to the model’s tendency to underpredict returns systematically. In contrast, the relatively moderate α values in ‘1911’ and ‘2212’, which correspond to a mild uptrend and a sideways market, respectively, suggest

Model	Avg_R2	Avg_MSE	avg_ α	Ann_avg_ α	t_statistic
Pre-COVID-19 Period (1911)					
B-Additive	0.0063	0.0060	0.0035	0.0417	9.4624***
GRU	0.0091	0.0060	0.0036	0.0429	9.5687***
L-Concat	0.0016	0.0061	0.0035	0.0423	9.5809***
L-DotProd	0.0128	0.0060	0.0035	0.0426	9.4785***
L-General	0.0067	0.0060	0.0037	0.0450	10.2364***
LSTM	0.0045	0.0060	0.0038	0.0455	10.2192***
RNN	0.1137	0.0053	0.0040	0.0478	10.5572***
self_att	0.0280	0.0059	0.0028	0.0340	7.9699***
sparse_att	0.0245	0.0059	0.0030	0.0363	8.5817***
COVID-19-Inclusive Period (2112)					
B-Additive	0.0043	0.0080	0.0053	0.0632	16.5328***
GRU	0.0171	0.0079	0.0049	0.0586	14.3459***
L-Concat	0.0011	0.0080	0.0053	0.0632	15.8036***
L-DotProd	0.0102	0.0079	0.0053	0.0641	16.5363***
L-General	0.0056	0.0080	0.0054	0.0645	17.0102***
LSTM	0.0023	0.0080	0.0055	0.0660	17.2300***
RNN	0.1401	0.0068	0.0039	0.0467	11.7454***
self_att	0.0202	0.0078	0.0045	0.0540	14.2660***
sparse_att	0.0194	0.0079	0.0048	0.0581	15.3381***
Period Including COVID-19 and One-Year After (2212)					
B-Additive	0.0058	0.0084	0.0038	0.0453	13.0552***
GRU	0.0176	0.0083	0.0036	0.0429	11.7015***
L-Concat	0.0028	0.0084	0.0038	0.0456	13.1388***
L-DotProd	0.0099	0.0083	0.0038	0.0473	13.4858***
L-General	0.0069	0.0084	0.0039	0.0469	13.8141***
LSTM	0.0044	0.0084	0.0040	0.0480	14.3907***
RNN	0.1432	0.0071	0.0025	0.0303	8.5837***
self_att	0.0164	0.0083	0.0038	0.0461	13.7656***
sparse_att	0.0189	0.0083	0.0035	0.0416	11.8082***

Table 3.2: Out-of-sample Predictive indicators and α . ‘Ann’ is the notation of annualization, and ‘avg’ is short for ‘average’. *, ** and *** depict the statistical significance level of 90%, 95% and 99% respectively.

Figure 3.8: Out-of-sample R^2 distribution of the best fitted models

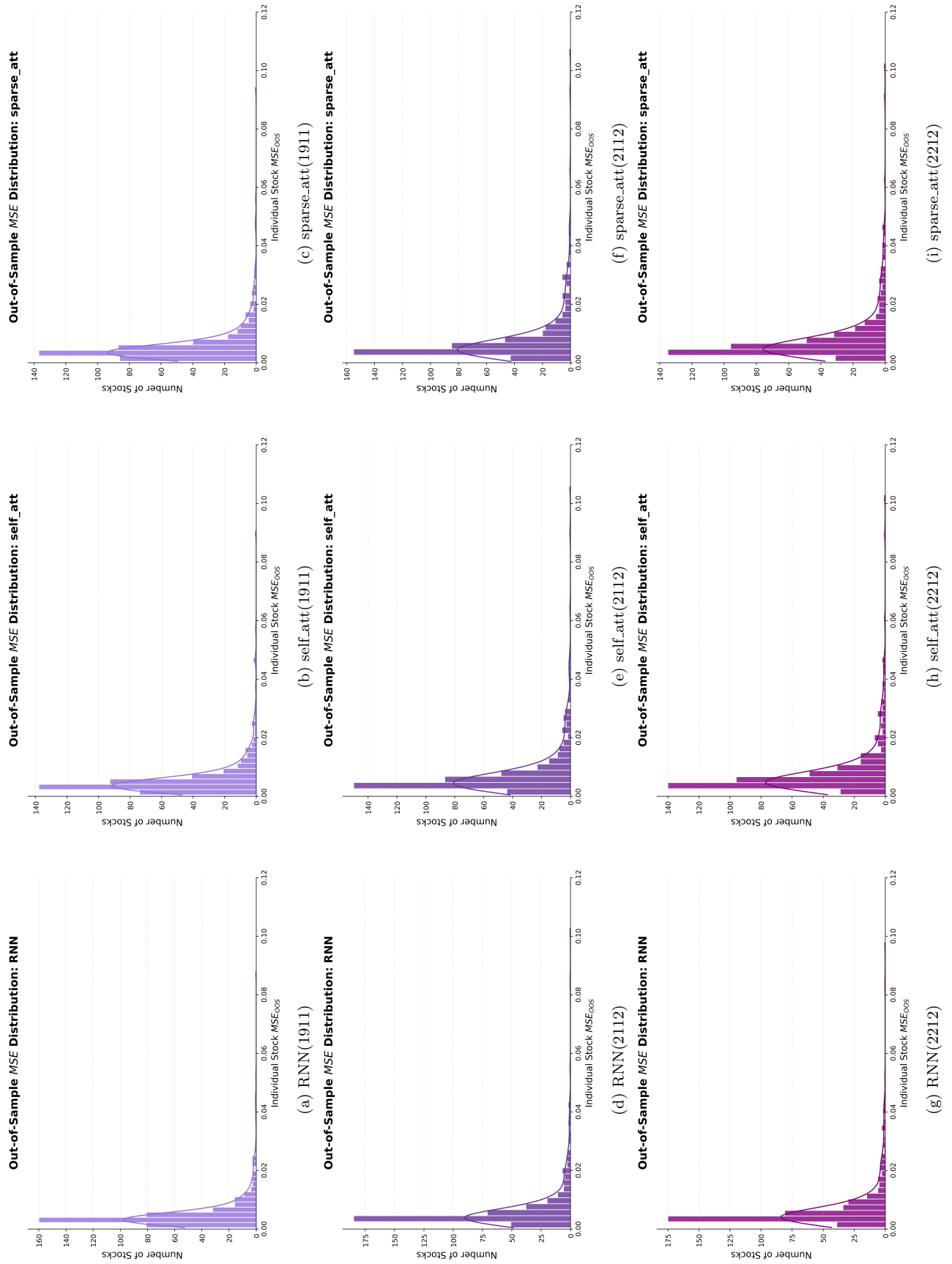
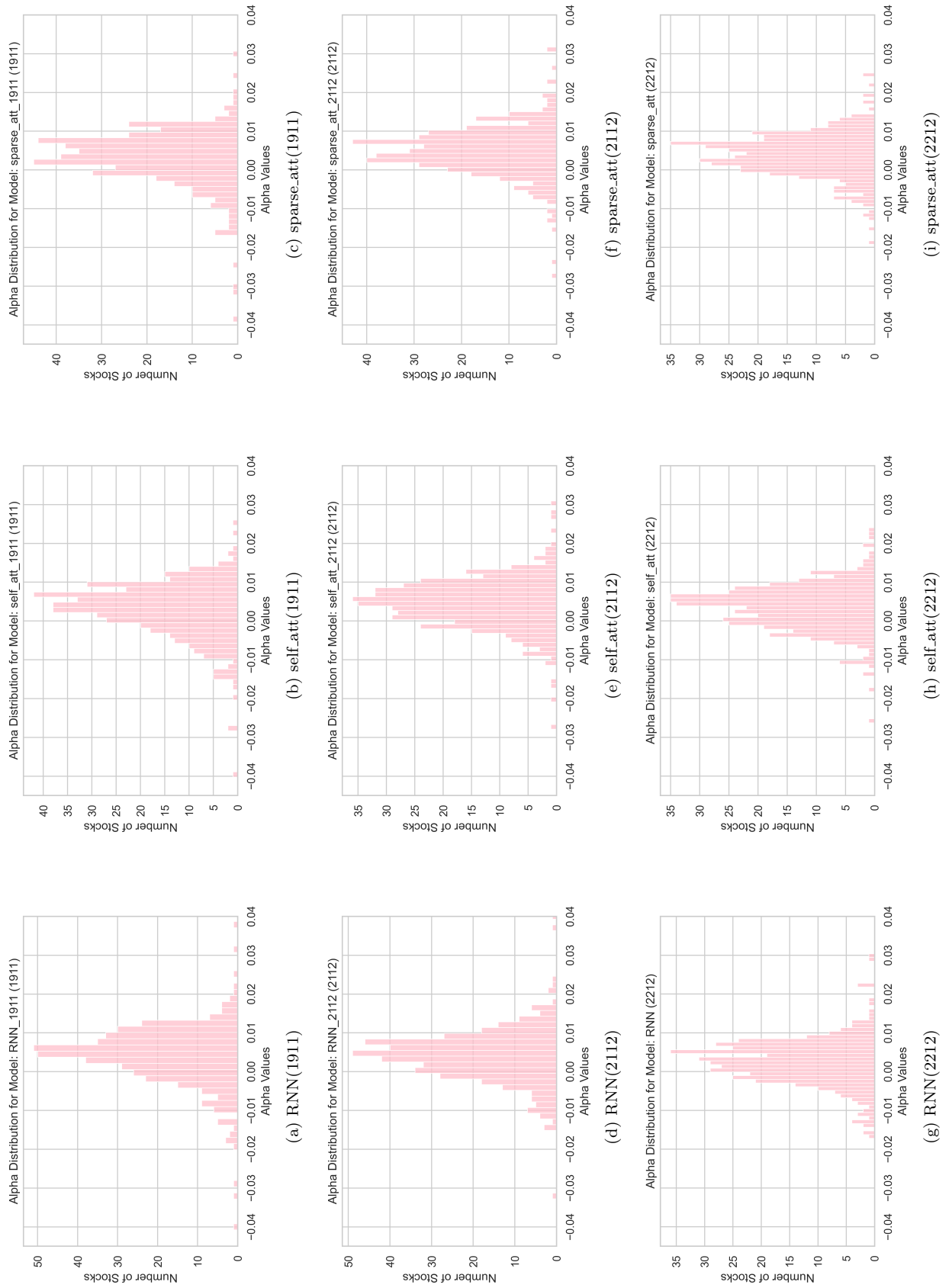


Figure 3.9: Out-of-sample MSE distribution of the best fitted models

that excess return potential tends to concentrate in highly intensive trending environments. Secondly, apart from the vanilla RNN model, there is no incredible difference between the average OOS α of models in a single period. Thirdly, at least 60% of stocks have positive OOS α individually in different models with distinctive periods. An example α distribution plot is shown in Figure 3.10. Periodically, the vanilla RNN model has the highest annualized OOS residual α (4.78%) in Period ‘1911’, while the LSTM model has the highest values in Period ‘2112’ (6.6%) and Period ‘2212’ (4.8%). Nevertheless, in each period, the difference between models is modest.

The DM test evaluates the null hypothesis that two competing forecasts have equal predictive accuracy. Table 3.3 shows the results of the DM test. A negative value between two models means the model that shows as the row label outperforms the model that shows as the column label. The DM test results agree with the OOS R^2 results from Table 3.2. The proposed attention models (self_att and sparse_att) significantly outperform alternative models in Period 1911, except for the vanilla RNN model. However, with the HAC estimator moderating the issue of autocorrelation between the differences of pair-wise models’ prediction errors, the significance of the differences between the proposed RNN global self-attention model (self_att) and alternative models is reduced significantly in COVID-19-Inclusive Period (2112) (significantly different from Model B-Additive, L-Concat and RNN) and Period Including COVID-19 and One-Year After (2212) (only RNN). In contrast, the sparse_att are significantly different from alternatives in all periods except self_att and GRU (in Period 2112 and Period 2212). The advantage of the proposed models against the GRU becomes less pronounced in Period 2112, when the market surged sharply. In Period 2212, when the market experienced a sharp decline followed by a period of sideways movement, the advantage of the proposed RNN global self-attention model (self_att) nearly vanished, but the advantage of the proposed RNN sliding window attention model (sparse_att) remains strong. On the other hand, the difference between the proposed models is insignificant. Among the benchmark models, RNN Luong’s concatenate attention model (L-Concat) achieves the worst-performing model during all periods, followed by LSTM and the additive attention model (B-Additive). The L-DotProd model, as the foundation of the proposed models, per-

Figure 3.10: Out-of-sample α distribution of the best fitted models.

forms significantly better than RNN Luong’s attention model (L-General) in Period 1911, but shows less advantage in the market turbulence periods. GRU insignificantly underperforms the RNN L-DotProd model in Period 1911, but apart from this, it outperforms all benchmark attention models (B-Additive, L-DotProd, L-Concat, L-General), excluding the proposed models, in all periods. The performance of GRU in all periods is better and more stable than that of LSTM. Overall, the robust DM test results support the findings in Table 3.2.

Since the MLP pre-training process extracted the information from the original factors and compressed the 182 factors into 128 extracted factors, which reduces the financial and economic interpretability of the models, the variable (factor) importance of the extracted factors and the correlation between the top 20 extracted factors and original factors are also computed to investigate which factors have the highest impact on the stock pricing. Figure 3.11 to Figure 3.13 illustrates the models with the highest OOS R^2 (RNN, self_att and sparse_att) of the correlation between the top 20 extracted factors and the original factors. This is a widely applied post-hoc interpretability method in ML for the asset pricing domain, for example, Gu et al. (2020); Chen et al. (2024) used the method on latent (extracted) factors and original input factors. The method uses the sum of the latent factors correlation to capture the overall economic meanings of an original factor, and the max of the latent factors correlation to measure the individual latent factor’s impact. From the sum of the correlation, the most highly impactful factors are quite concentrated; the top 5 factors are Bid Ask Spread (BidAskSpread), NetEquityFinance (NetEquityFinance), Forecast Dispersion (ForecastDispersion), Price and volatilities (IdioVol). This means under the framework of RNN, self_att, sparse_att, the market frictions (BidAskSpread), Equity Financing capability (NetEquityFinance), analyst disagreement (ForecastDispersion), lowest price stocks (Price) and Idiosyncratic volatilities (IdioVol) are the top 5 factors that drive the large-cap stock returns. These are followed by the factors of the highest point for the last 52 weeks (High52), cashflow variance (VarCF), systematic risk (Beta), intangible assets (IntanSP), forecast earnings per share (FEPS), long-term reversal (LRreversal) and operating profit (OperProf). These factors are highly consistent with classic literature, for example, Bid Ask

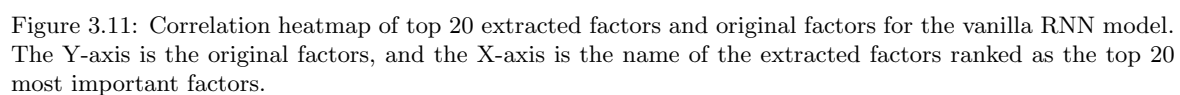
	B-Additive	GRU	L-Concat	L-DotProd	L-General	LSTM	RNN	self_att
Pre-COVID-19 Period (1911)								
B-Additive								
GRU	-0.78							
L-Concat	2.40**	2.47**						
L-DotProd	-2.62***	-1.13	-3.45***					
L-General	0.77	1.40	-1.79*	2.68***				
LSTM	0.79	1.10	-2.14**	2.83***	-0.11			
RNN	-4.62***	-4.80***	-4.73***	-4.40***	-4.61***	-4.48***		
self_att	-3.20***	-2.86***	-3.46***	-2.38**	-3.22***	-2.99***	4.32***	
sparse_att	-3.09***	-2.84***	-3.40***	-1.98**	-2.95***	-2.72***	4.57***	0.60
COVID-19-Inclusive Period (2112)								
B-Additive								
GRU	-1.91*							
L-Concat	1.51	2.18**						
L-DotProd	-1.97**	1.14	-2.12**					
L-General	-1.30	1.55	-1.92*	1.30				
LSTM	-0.10	1.58	-1.79*	1.34	0.78			
RNN	-3.90***	-4.09***	-3.88***	-3.95***	-3.86***	-3.79***		
self_att	-1.83*	-0.20	-1.98**	-1.25	-1.64	-1.51	4.04***	
sparse_att	-3.96***	-0.45	-3.68***	-2.69***	-3.66***	-2.82***	3.74***	-0.29
Period Including COVID-19 and One-Year After (2212)								
B-Additive								
GRU	-1.92*							
L-Concat	1.29	2.26**						
L-DotProd	-2.19**	1.06	-2.27**					
L-General	-1.58	1.46	-2.07**	1.00				
LSTM	-0.37	1.62	-2.19**	1.32	0.80			
RNN	-4.26***	-4.45***	-4.25***	-4.21***	-4.18***	-4.16***		
self_att	-1.32	0.63	-1.51	-0.41	-0.86	-0.99	4.42***	
sparse_att	-3.61***	-0.18	-3.38***	-2.07**	-2.96***	-2.50**	4.12***	1.08

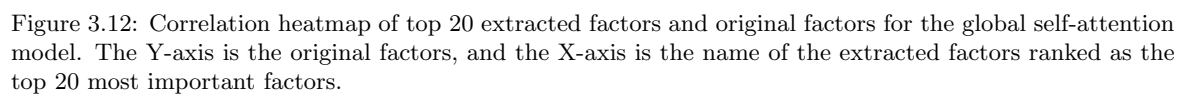
Table 3.3: DM test statistics. Values in parentheses indicate the negative values. A negative value means the model named with the row label outperforms the model with the column label. For example, the first value in the first column of the '1911' panel, which is -0.78, indicates that Model GRU outperforms Model B-Additive. *, ** and *** depict the statistical significance level of 90%, 95% and 99%, respectively.

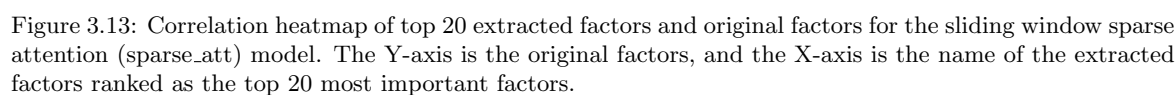
Spread (BidAskSpread) (Amihud, 2002; Brennan et al., 1998), Equity Financing capability (NetEquityFinance) (Baker and Wurgler, 2000; Fama and French, 2005), analyst disagreement (ForecastDispersion) (Diether et al., 2002; Johnson, 2004) and Idiosyncratic volatilities (IdioVol) (Ang et al., 2009). For the maximum correlation value of individual latent factors, the top 10 latent factors of these models agree that Bid Ask Spread (BidAskSpread), Equity Financing capability (NetEquityFinance), analyst disagreement (ForecastDispersion), the last 52 weeks (High52), lowest price stocks (Price), Idiosyncratic volatilities (IdioVol) and intangible assets (IntanSP) are the most important factors, which aligns with literature as well. The results of variable (factors) importance examination verified the conclusion of Andrew Y.; Zimmermann (2020) that most of the pricing ability is concentrated in a few factors, even in the context of ML neural network models. Overall, the factor importance and correlation examination highly improve the transparency of the ML models in the asset pricing context.

3.5.2 Back-testing performance

Backtesting refers to the process of evaluating a trading strategy or predictive model by applying it to OOS returns to assess its performance. The predictive models provide the predictive returns; henceforth, a market timing strategy or trading signal generating system would be deployed to the real returns and predictive returns to generate trading signals for guiding trading behaviours (open or close positions). In this study, the sign signal generation method from the typical trend-following trading strategy is adopted for the market timing assignment. Concretely, if the predictive return at a time point is positive, which matches the real return in that time point, the circumstance is considered a buy-side (long) signal. The simulated buy transaction takes place, and the real returns from the next time point would be counted as a gain or loss until the sell-side signal (negative predictive return matches negative real return) shows up as the indicator of closing position. This is also known as the long-only strategy. The short-only strategy is in contrast to the long-only strategy, which opens a short position when both real return and predictive return have negative values at one time point, while closing this position when they both have positive values. The







back-testing in this study only simulates the long-only transaction by considering the practical difficulties in real stock markets and follows the suggestion of Avramov et al. (2021). Specifically, real stock market trading has restrictions for the sell-side positions, for example, the availability of the stocks for selling, variable borrowing rates for individual stocks, as well as the liquidity of the stock. Two simple asset allocation methods are applied for portfolio-wise backtesting, which are equal-weighted and value-weighted. An equal-weighted portfolio distributes weights to 420 stocks evenly, while the value-weighted portfolio allocates the weights to stocks based on the size of the stocks' market capitalization. In this sense, in a value-weighted portfolio, stocks with larger market capitalization would be distributed with heavier weights, and the summation of all weights for these 420 stocks is equal to one. Thus, the computation of indicators in the value-weighted portfolio concentrates more on larger market capitalization stocks.

Table 3.4 presents the back testing results of the equal-weighted portfolio of each model in the three periods after deducting the static transaction cost (50bps). All models in all periods consistently outperform the buy-and-hold (BHE) strategy except for the vanilla RNN model and RNN Luong's concatenate attention model (L-Concat). The proposed models have the highest annulized returns during all periods. When considering the standard deviation risks (Sharpe Ratio) and downside risks (Sortino Ratio) of the returns, the proposed models achieve first place (RNN global self-attention model) and second place (RNN sliding window attention model) in Period 1911 and Period 2212. However, in Period 2112, the vanilla RNN model shows a marginal advantage on the Sharpe ratio (SR) and the Sortino ratio (SO). The Sharpe ratio of the vanilla RNN model lies between the RNN global self-attention model (self_att) and the RNN sliding window attention model (sparse_att), while the Sortino ratio is higher than both proposed models. This demonstrates that the vanilla RNN model has a higher risk protection capability, especially downside risks, when it faces extreme market conditions such as a volatile market with a significant trend. GRU and LSTM models perform quite similarly in all periods. For the rest of the RNN attention models, including the RNN additive attention model (B-Additive) and the RNN Luong's attention models (L-Concat, L-DotProd, L-General), their performance varies depending on the market conditions. In

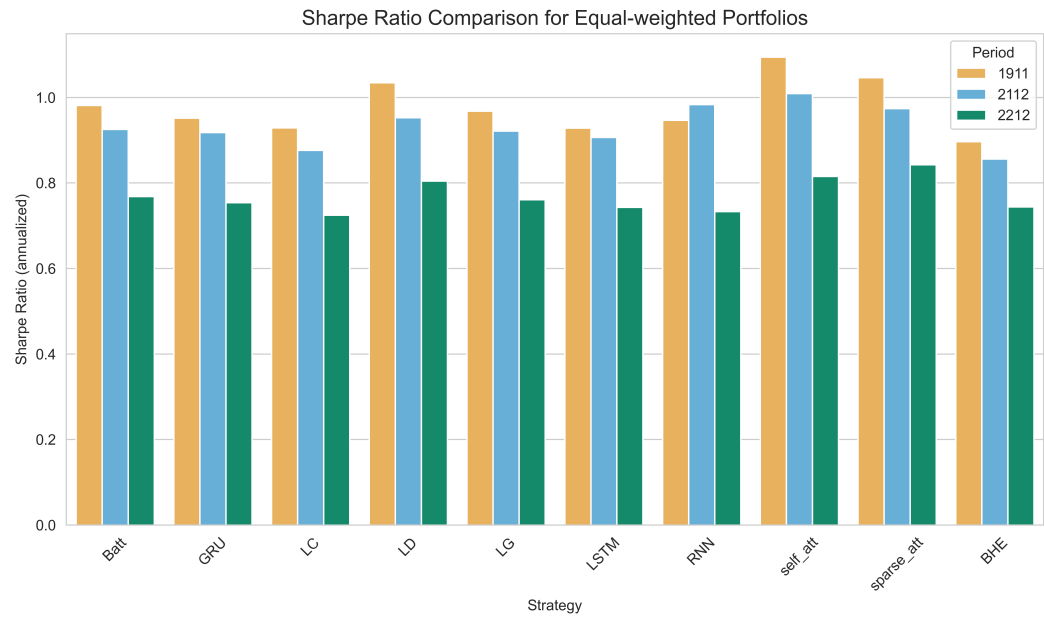
Period 1911, Model B-Additive is superior to Model L-Concat, L-DotProd, L-General, as well as GRU and LSTM models, and becomes the most profitable RNN attention model following the proposed models. However, when considering volatility risks, including standard deviation and downside risks, it falls behind the RNN Luong's dot product attention model (L-DotProd). The performance of Model B-Additive in Period 2112 and Period 2212 is not as well as in Period 1911 when comparing with alternative RNN attention benchmark models (L-Concat, L-DotProd, L-General) and RNN variation models (GRU, LSTM) in the same period. It falls behind the L-DotProd model but still outperforms RNN variation models in both periods. For RNN Luong's three attention models (L-Concat, L-DotProd, L-General), L-DotProd performs best in all periods which is followed by L-General. These phenomena imply that all RNN attention models, apart from the L-Concat model, are superior to the RNN variation models, especially when encountering extreme market conditions. Nonetheless, the vanilla RNN model shows a great capability to control downside risks during the market turbulence, but sacrifices its absolute capital gain.

Figure 3.14 is the diagram for the comparison of the annulized Sharpe ratio and Sortino ratio in the equal-weighted portfolios. The upper panel shows the Sharpe ratio, and the lower panel shows the Sortino ratio. It is apparent that the Sharpe ratio of the self_att model achieves the highest value in the pre-pandemic period (1911) and the pandemic period (2112) whereas during the period that covers one year after the pandemic (2212), which contains a rapid drop and a highly fluctuating sideways movement of the price shows in Figure 3.7, the proposed sparse_att wins the first place. It is also worth noting that the L-DotProd model follows closely to the performance of the proposed models and achieves third place. This implies that the attention mechanisms based on the dot product, including L-DotProd, self_att and sparse_att, demonstrate superior performance compared to other attention variants. The diagram of the Sortino ratios enlarges the difference between models. It supports the conclusion drawn from the diagram of the Sharpe ratios in Period 1911 and Period 2212. However, in Period 2112, the vanilla RNN model achieves the highest value with an insignificant advantage compared to the proposed models, especially the self_att model, which implies that the vanilla RNN model has a great capability to hedge the downside risks

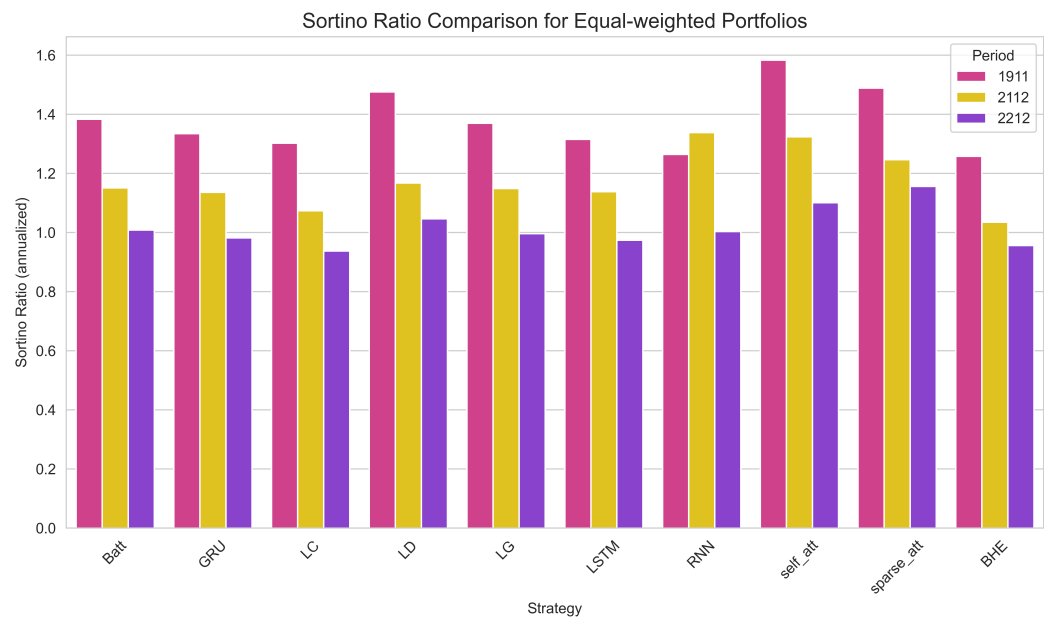
Strategy	Max Draw-down	Ann turn	Re-	Sharpe Ratio (SR)	Sortino Ratio (SO)	Ann SR	Ann SO
Pre-COVID-19 Period (1911)							
B-Additive	(0.3128)	0.1229		0.2834	0.3993	0.9816	1.3832
GRU	(0.3241)	0.1162		0.2748	0.3853	0.9520	1.3349
L-Concat	(0.3564)	0.1168		0.2681	0.3760	0.9288	1.3024
L-DotProd	(0.3077)	0.1289		0.2986	0.4259	1.0343	1.4754
L-General	(0.3180)	0.1212		0.2795	0.3954	0.9681	1.3698
LSTM	(0.3562)	0.1166		0.2680	0.3798	0.9283	1.3155
RNN	(0.2588)	0.1041		0.2733	0.3650	0.9466	1.2643
self_att	(0.3008)	0.1357		0.3159	0.4570	1.0944	1.5832
sparse_att	(0.3154)	0.1298		0.3020	0.4298	1.0462	1.4889
BHE	(0.3701)	0.1139		0.2588	0.3631	0.8965	1.2578
COVID-19-Inclusive Period (2112)							
B-Additive	(0.4953)	0.1459		0.2672	0.3323	0.9255	1.1512
GRU	(0.4932)	0.1392		0.2650	0.3279	0.9180	1.1357
L-Concat	(0.5212)	0.1367		0.2530	0.3100	0.8765	1.0739
L-DotProd	(0.4820)	0.1471		0.2751	0.3370	0.9530	1.1675
L-General	(0.4886)	0.1423		0.2660	0.3317	0.9215	1.1489
LSTM	(0.4895)	0.1395		0.2617	0.3287	0.9067	1.1385
RNN	(0.4123)	0.1300		0.2838	0.3862	0.9833	1.3379
self_att	(0.4427)	0.1540		0.2913	0.3823	1.0091	1.3242
sparse_att	(0.4584)	0.1503		0.2811	0.3598	0.9739	1.2463
BHE	(0.5310)	0.1349		0.2472	0.2988	0.8563	1.0351
Period Including COVID-19 and One-Year After (2212)							
B-Additive	(0.4960)	0.1241		0.2218	0.2911	0.7684	1.0083
GRU	(0.4968)	0.1177		0.2177	0.2835	0.7541	0.9820
L-Concat	(0.5208)	0.1165		0.2092	0.2707	0.7248	0.9377
L-DotProd	(0.4731)	0.1279		0.2323	0.3021	0.8046	1.0465
L-General	(0.4933)	0.1215		0.2197	0.2875	0.7609	0.9961
LSTM	(0.4980)	0.1181		0.2145	0.2812	0.7430	0.9742
RNN	(0.4191)	0.1012		0.2117	0.2896	0.7334	1.0033
self_att	(0.4409)	0.1334		0.2433	0.3336	0.8429	1.1558
sparse_att	(0.4545)	0.1304		0.2354	0.3178	0.8154	1.1010
BHE	(0.5308)	0.1141		0.2148	0.2760	0.7441	0.9561

Table 3.4: Equal-weighted portfolio performance with deduction of static transaction cost. ‘Ann’ is the notation for annulization, while ‘SR’ and ‘SO’ are short for Sharpe ratio and Sortino ratio respectively. ‘BHE’ means buy-and-hold strategy in Equal-weighted portfolios. The values in the parentheses indicate negative values. ‘Ann Return’ and ‘Max Drawdown’ can be transformed to a percentage presentation by multiplying by 100.

during the extreme market fluctuations.



(a) Annualized Sharpe ratio comparison



(b) Annualized Sortino ratio comparison

Figure 3.14: Annualized Sharpe ratio and Sortino ratio for Equal-weighted with static transaction cost of 50bps. 'Batt' indicates the additive attention model (B-Additive), 'LC', 'LD' and 'LG' mean L-Concat, L-DotProd and L-General attention model, respectively.

The equal-weighted portfolio cumulative return plot, Figure 3.15, agrees with the findings from Table 3.4. In Period 1911, the cumulative returns of the L-DotProd model and the RNN

sliding window attention model (`sparse_att`) are similar, and the L-DotProd model outperforms the `sparse_att` model in a short period, from late 2015 to the middle of 2018. However, since 2019, the proposed RNN global self-attention model has shown a non-negligible advantage over alternatives, closely followed by the proposed RNN sliding window attention model. All models' cumulative returns outperform the buy-and-hold benchmark (BHE) and the vanilla RNN model since 2017. Apart from the L-Concat model, all attention models show superiority to those models without attention mechanisms during the market shock periods. Model L-DotProd, B-Additive, L-General rank in third, fourth and fifth place, respectively. Although the vanilla RNN model exhibits strong OOS fitness and ability for controlling downside risks, it sacrifices its absolute capital gain during the extreme market turbulence. In this respect, the vanilla RNN model is possibly more suitable for funds that require high-capability hedging against downside risk rather than pursuing high absolute capital gains, or for modelling market volatility.

The robust examination is conducted to investigate how the turnover rate of strategies based on these models affects profitability, which is shown in Table 3.5. The self-attention (`self_att`) model consistently delivers superior risk-adjusted performance across all three periods. In both tables, `self_att` achieves the highest annualized Sharpe ratio (Ann.SR) and annualized Sortino ratio (Ann.SO), together with the competitive annualized returns (Ann Return/AR), while maintaining mild maximum drawdowns (MDD). For instance, in the pre-COVID period under static costs, `self_att` has an Ann.SR of 1.0944 and Ann.SO of 1.5832, outperforming the buy-and-hold benchmark (BHE) by substantial margins (0.8965 and 1.2578, respectively). This dominance persists even after incorporating dynamic turnover costs, where `self_att`'s moderate turnover rate (approximately 0.1156 to 0.1196) limits cost deterioration compared with higher-turnover models. The sliding window sparse attention model (`sparse_att`) ranks a close second, highlighting the capability of advanced attention-based architectures in capturing return dynamics stock pricing data.

In contrast, traditional recurrent neural network models such as LSTM, GRU, and benchmark attention models (L-Concat, L-General) exhibit moderate performance, generally trail-

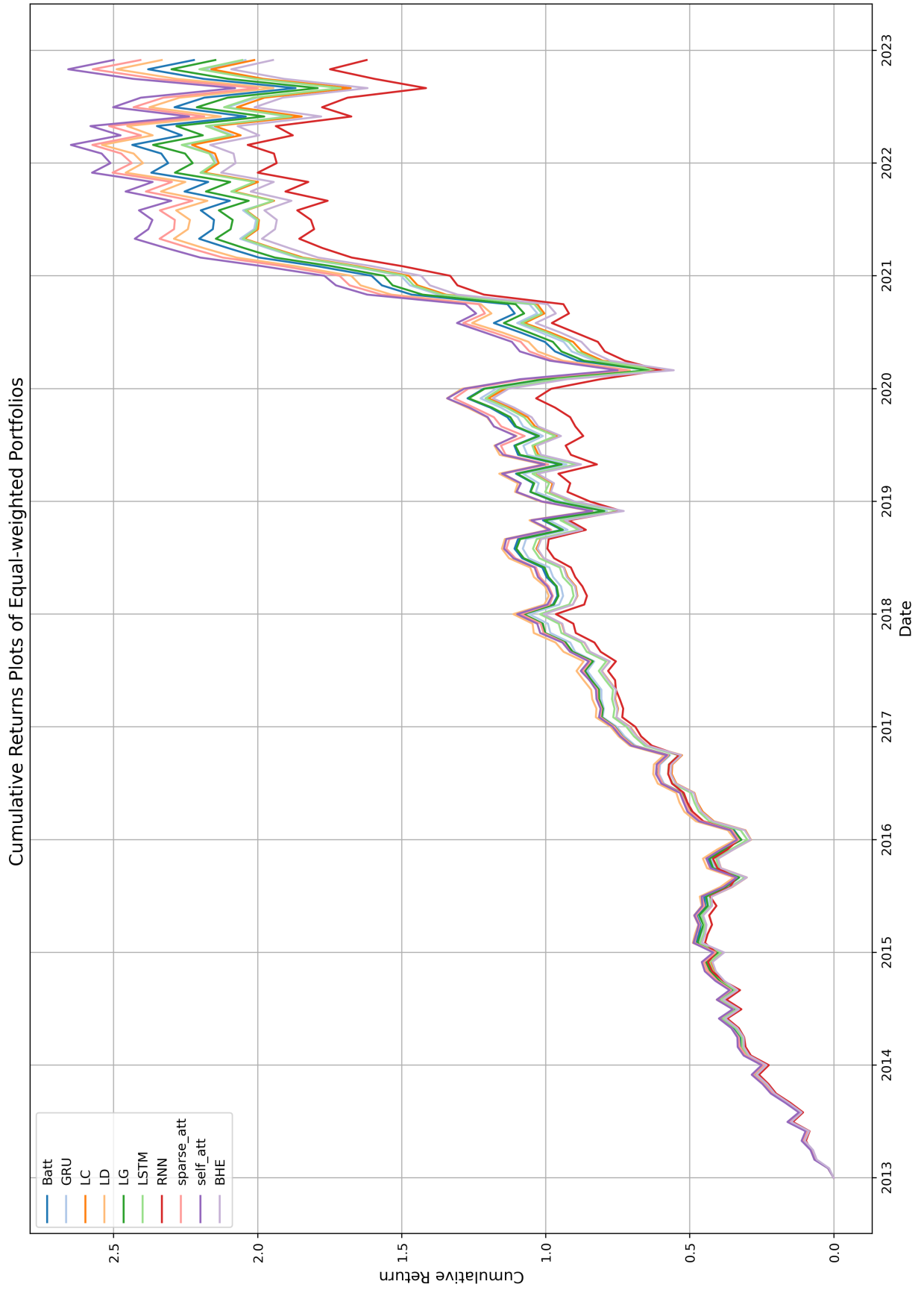


Figure 3.15: Cumulative return plot for Equal-weighted portfolios with deduction of static transaction cost (50bps). ‘Batt’ indicates the additive attention model (B-Additive), ‘LC’, ‘LD’ and ‘LG’ mean L-Concat, L-DotProd and L-General attention model, respectively.

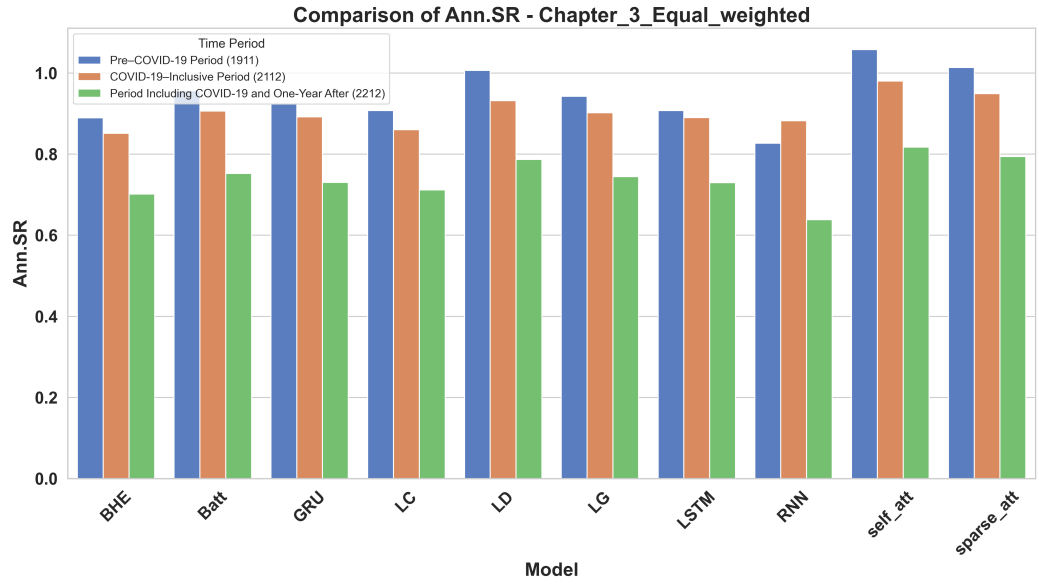
ing the attention mechanisms in risk-adjusted returns, particularly in the COVID-19-Inclusive Period (2112) and Period Including COVID-19 and One-Year After (2212), where Ann.SR and Ann.SO declines significantly across the board due to the extreme market turbulence. The Bahdanau attention model (B-Additive) performs respectably in the middle tier but is outperformed by both self_att, sparse_att and L-DotProd, highlighting the advantages of dot-product-based attention over additive attention in equal-weighted portfolio configurations. Notably, the RNN model exhibits the lowest MDD in Table 3.4, suggesting strong downside protection; however, its high turnover (exceeding 0.49 across periods) leads to severe performance deterioration once transaction costs are fully accounted for in Table 3.5, resulting in the lowest Ann.SR and Ann.SO among all strategies. All models substantially outperform the buy-and-hold benchmark apart from the vanilla RNN model in both transaction cost scenarios, confirming the presence of extra profitability beyond the benchmark.

Dynamic transaction costs (turnover with 20bps for large-cap stocks), especially when combined with high portfolio turnover, significantly enlarge the differences between backtesting performances across models. The self-attention model (self_att) and sliding-window sparse attention (sparse_att) not only demonstrate superior performance in static transaction cost environments but also exhibit the best net cumulative returns and robustness to turnover-based realistic market frictions. These findings highlight that, in equal-weighted portfolio construction, controlling turnover is critical for maintaining the profitability of these models. In this perspective, dot-product-based attention mechanisms have significant advantages in generating more stable and selective trading signals. On the other hand, although the RNN model shows persistence in risk control, its profitability significantly decays due to the high turnover. This highlights a key insight: high forecasting accuracy does not mean high real-world returns.

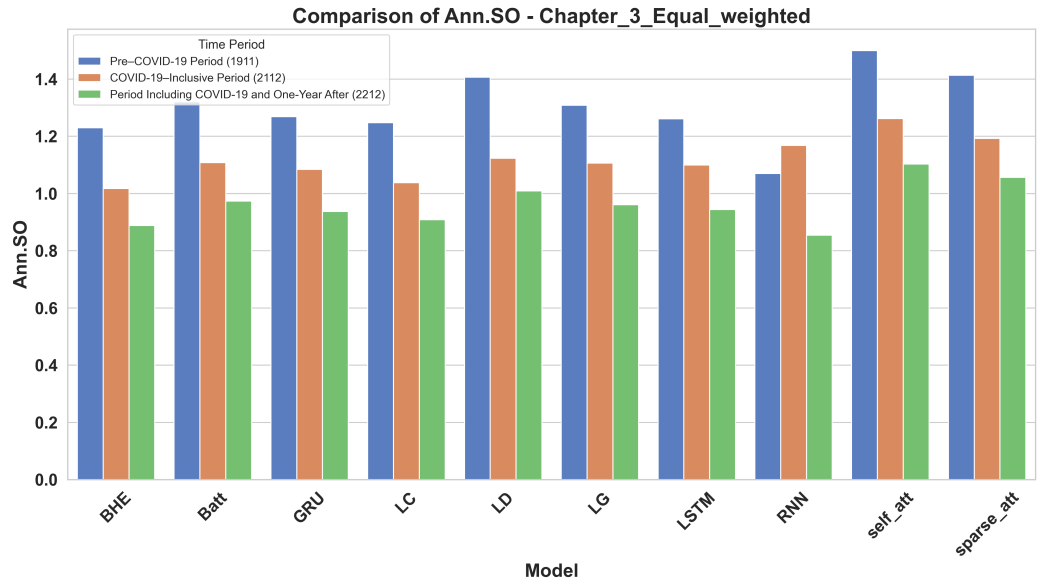
In the value-weighted portfolios, which are shown in Table 3.6, apart from the proposed models and the vanilla RNN model, the annualized returns of all models in all periods are significantly higher than those in the equal-weighted portfolio. Annualized returns of the vanilla RNN model in all periods are considerably lower than those in the equal-weighted

Model	AR	Ann.SR	SR	Ann.SO	SO	MDD	Turnover
Pre-COVID-19 Period (1911)							
BHE	0.1141	0.8892	0.2567	1.2303	0.3552	-0.1483	0.0120
B-Additive	0.1217	0.9564	0.2761	1.3208	0.3813	-0.1466	0.0672
GRU	0.1145	0.9233	0.2665	1.2691	0.3664	-0.1407	0.0833
L-Concat	0.1160	0.9075	0.2620	1.2478	0.3602	-0.1481	0.0496
L-DotProd	0.1275	1.0063	0.2905	1.4066	0.4060	-0.1366	0.0761
L-General	0.1199	0.9425	0.2721	1.3084	0.3777	-0.1478	0.0682
LSTM	0.1159	0.9077	0.2620	1.2610	0.3640	-0.1453	0.0450
RNN	0.0916	0.8266	0.2386	1.0701	0.3089	-0.1307	0.4955
self_att	0.1332	1.0577	0.3053	1.4998	0.4330	-0.1319	0.1156
sparse_att	0.1277	1.0134	0.2925	1.4132	0.4079	-0.1365	0.1011
COVID-19-Inclusive Period (2112)							
BHE	0.1353	0.8512	0.2457	1.0181	0.2939	-0.2890	0.0093
B-Additive	0.1448	0.9063	0.2616	1.1085	0.3200	-0.2806	0.0628
GRU	0.1369	0.8917	0.2574	1.0847	0.3131	-0.2743	0.1058
L-Concat	0.1360	0.8604	0.2484	1.0376	0.2995	-0.2857	0.0463
L-DotProd	0.1458	0.9320	0.2690	1.1231	0.3242	-0.2774	0.0721
L-General	0.1411	0.9019	0.2604	1.1065	0.3194	-0.2748	0.0632
LSTM	0.1388	0.8900	0.2569	1.1000	0.3175	-0.2714	0.0465
RNN	0.1174	0.8823	0.2547	1.1683	0.3373	-0.2166	0.4922
self_att	0.1514	0.9800	0.2829	1.2626	0.3645	-0.2578	0.1182
sparse_att	0.1483	0.9490	0.2739	1.1930	0.3444	-0.2632	0.0964
Period Including COVID-19 and One-Year After (2212)							
BHE	0.1143	0.7017	0.2026	0.8883	0.2564	-0.2890	0.0083
B-Additive	0.1229	0.7522	0.2171	0.9736	0.2810	-0.2798	0.0616
GRU	0.1151	0.7300	0.2107	0.9370	0.2705	-0.2760	0.1130
L-Concat	0.1158	0.7116	0.2054	0.9083	0.2622	-0.2858	0.0451
L-DotProd	0.1266	0.7869	0.2272	1.0094	0.2914	-0.2733	0.0688
L-General	0.1203	0.7446	0.2149	0.9611	0.2775	-0.2777	0.0625
LSTM	0.1174	0.7295	0.2106	0.9438	0.2725	-0.2744	0.0442
RNN	0.0881	0.6382	0.1842	0.8543	0.2466	-0.2210	0.5192
self_att	0.1308	0.8170	0.2359	1.1033	0.3185	-0.2554	0.1196
sparse_att	0.1284	0.7941	0.2292	1.0566	0.3050	-0.2608	0.0941

Table 3.5: Equal-weighted portfolio performance considering the dynamic transaction cost. 'Ann' is the notation for annulization, while 'SR' and 'SO' are short for Sharpe ratio and Sortino ratio respectively. The notation of 'MDD' means Maximum Drawdown, and 'BHV' means buy-and-hold strategy in value-weighted portfolios. The Annulized Return (AR) and 'MDD' can be transformed to a percentage presentation by multiplying by 100.



(a) Annualized Sharpe ratio comparison



(b) Annualized Sortino ratio comparison

Figure 3.16: Annualized Sharpe ratio and Sortino ratio for Equal-weighted the dynamic transaction cost. ‘Batt’ indicates the additive attention model (B-Additive), ‘LC’, ‘LD’ and ‘LG’ mean L-Concat, L-DotProd and L-General attention model, respectively.

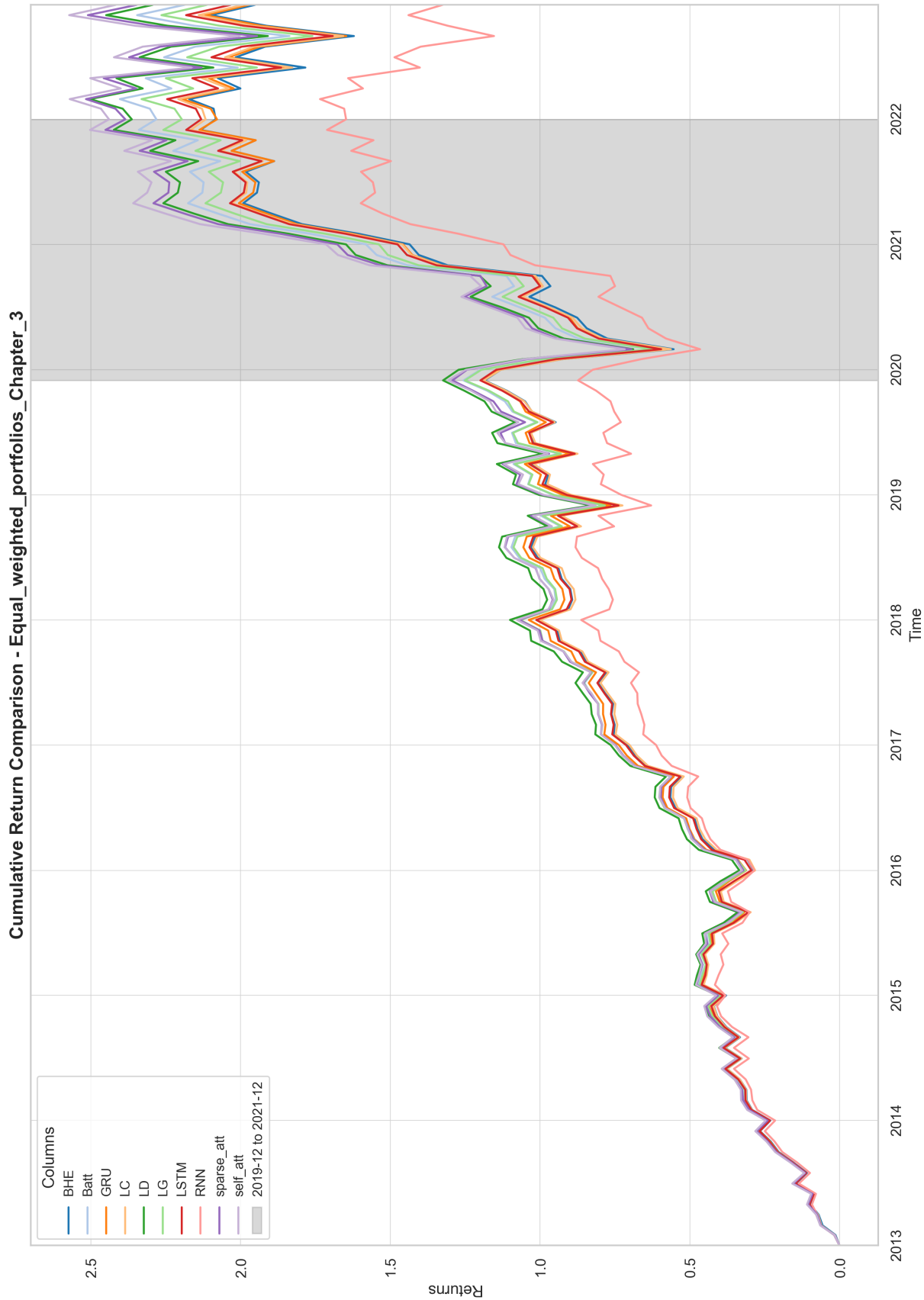


Figure 3.17: Cumulative return plot for Equal-weighted portfolios considering the dynamic transaction cost. ‘Batt’ indicates the additive attention model (B-Additive), ‘LC’, ‘LD’ and ‘LG’ mean L-Concat, L-DotProd and L-General attention model, respectively.

portfolio, but annualized returns of the proposed models vary according to the periods. In Period 1911 and Period 2212, the RNN global self-attention (self_att) model's annualized return is lower than the one in the same period of the equal-weighted portfolio, and the RNN sliding window attention model's annualized return is higher than the one in the same period of the equal-weighted portfolio. However, in Period 2212, the annualized returns of the proposed models become lower than those in equal-weighted portfolios. Specifically, the difference of the annualized return in the RNN sliding window attention model (sparse_att) is quite insignificant, but the one in self_att is significant. Nevertheless, when returns are assessed in light of volatility risks, which are shown as the Sharpe ratio and the Sortino ratio, all models in all periods illustrate significant advantages compared with those in equal-weighted portfolios. This phenomenon implies that all models work better on stocks with larger market capitalization when considering risks, but the annualized returns variation also indicates that the vanilla RNN model and self_att model are less preferred by larger market-cap stocks.

The rest of the findings from equal-weighted portfolios still stand for the value-weighted portfolios. All models in all periods outperform the buy-and-hold (BHV) strategy in value-weighted portfolios, excluding the vanilla RNN model in Period 1911, whose Sortino ratio is lower than that of BHV. From a full angle, value-weighted portfolios have a higher capability to moderate market fluctuation risks, and most of the models have higher annualized returns than equal-weighted portfolios. It is noteworthy to notice the rank change of the GRU model in Period 2112 of the value-weighted portfolio according to its Sortino ratio; it outperforms the RNN benchmark attention models (B-Additive, L-Concat, L-DotProd, L-General), demonstrating that the GRU model works more efficiently on handling the downside risk during the market turbulence for larger-cap stocks. The value-weighted portfolio cumulative return plots in Figure 3.19 support the conclusions drawn from Table 3.6.

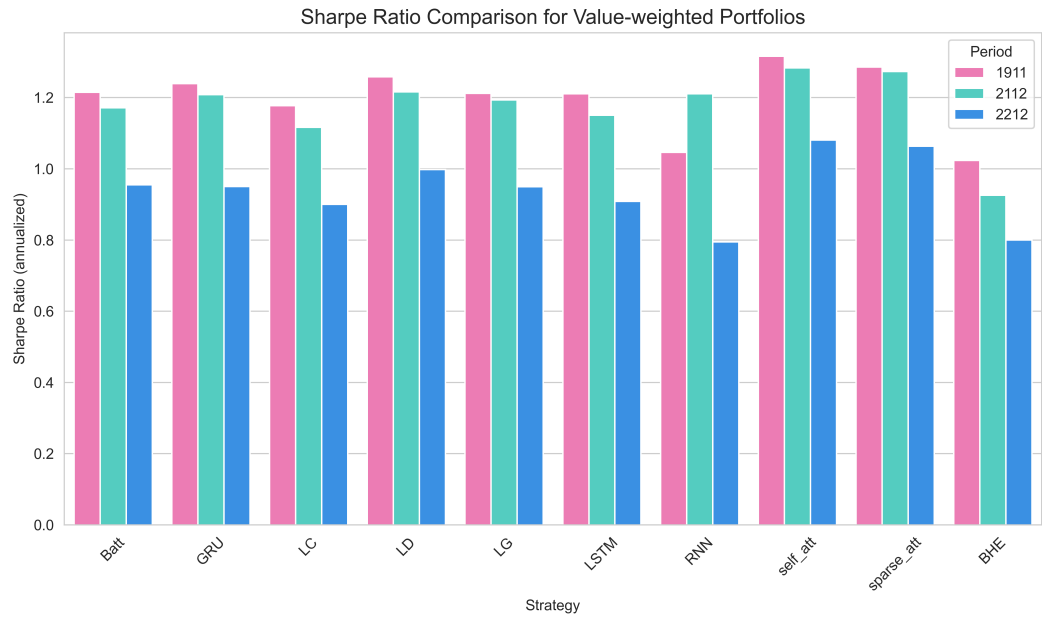
The comparison of the Sharpe ratios and the Sortino ratios for value-weighted portfolios is exhibited in Figure 3.18. The value-weighted method narrows the difference in periodic performance between models. In the Sharpe ratio diagram, the findings are consistent with those in the equal-weighted method diagram. The self_att model outperforms all alterna-

Strategy	Max Draw-down	Ann Re-turn	Sharpe Ratio (SR)	Sortino Ratio (SO)	Returns Std	Ann SR	Ann SO
Pre-COVID-19 Period (1911)							
B-Additive	(0.2526)	0.1338	0.3507	0.4787	0.0300	1.2149	1.6582
GRU	(0.2494)	0.1335	0.3578	0.4922	0.0293	1.2394	1.7049
L-Concat	(0.2639)	0.1319	0.3400	0.4699	0.0306	1.1777	1.6278
L-DotProd	(0.2196)	0.1343	0.3634	0.5135	0.0290	1.2589	1.7787
L-General	(0.2460)	0.1302	0.3499	0.4692	0.0292	1.2122	1.6252
LSTM	(0.2477)	0.1363	0.3496	0.4806	0.0307	1.2112	1.6649
RNN	(0.2848)	0.0827	0.3021	0.3955	0.0212	1.0466	1.3701
self_att	(0.2560)	0.1299	0.3800	0.5616	0.0267	1.3164	1.9455
sparse_att	(0.2258)	0.1339	0.3713	0.5048	0.0282	1.2864	1.7488
BHV	(0.2963)	0.1197	0.2956	0.4243	0.0323	1.0240	1.4698
COVID-19-Inclusive Period (2112)							
B-Additive	(0.3034)	0.1515	0.3383	0.4636	0.0355	1.1720	1.6060
GRU	(0.2684)	0.1474	0.3488	0.5016	0.0333	1.2083	1.7377
L-Concat	(0.3394)	0.1487	0.3224	0.4270	0.0367	1.1167	1.4792
L-DotProd	(0.2888)	0.1524	0.3511	0.4881	0.0343	1.2161	1.6907
L-General	(0.2895)	0.1473	0.3446	0.4656	0.0338	1.1936	1.6129
LSTM	(0.3264)	0.1539	0.3322	0.4414	0.0368	1.1509	1.5292
RNN	(0.2811)	0.1071	0.3497	0.5364	0.0238	1.2112	1.8581
self_att	(0.2829)	0.1395	0.3706	0.5785	0.0295	1.2837	2.0039
sparse_att	(0.2483)	0.1498	0.3677	0.5204	0.0320	1.2737	1.8028
BHV	(0.4513)	0.1337	0.2673	0.3263	0.0408	0.9260	1.1303
Period Including COVID-19 and One-Year After (2212)							
B-Additive	(0.2996)	0.1297	0.2757	0.3970	0.0380	0.9551	1.3754
GRU	(0.2915)	0.1241	0.2745	0.3952	0.0364	0.9507	1.3690
L-Concat	(0.3354)	0.1271	0.2599	0.3615	0.0398	0.9003	1.2524
L-DotProd	(0.2691)	0.1308	0.2883	0.4195	0.0364	0.9985	1.4533
L-General	(0.2989)	0.1268	0.2742	0.3858	0.0373	0.9500	1.3366
LSTM	(0.3410)	0.1301	0.2623	0.3585	0.0404	0.9087	1.2420
RNN	(0.3113)	0.0773	0.2295	0.3120	0.0266	0.7951	1.0807
self_att	(0.2581)	0.1240	0.3120	0.4985	0.0315	1.0808	1.7269
sparse_att	(0.2462)	0.1354	0.3070	0.4523	0.0351	1.0636	1.5668
BHV	(0.4513)	0.1156	0.2311	0.3031	0.0438	0.8006	1.0500

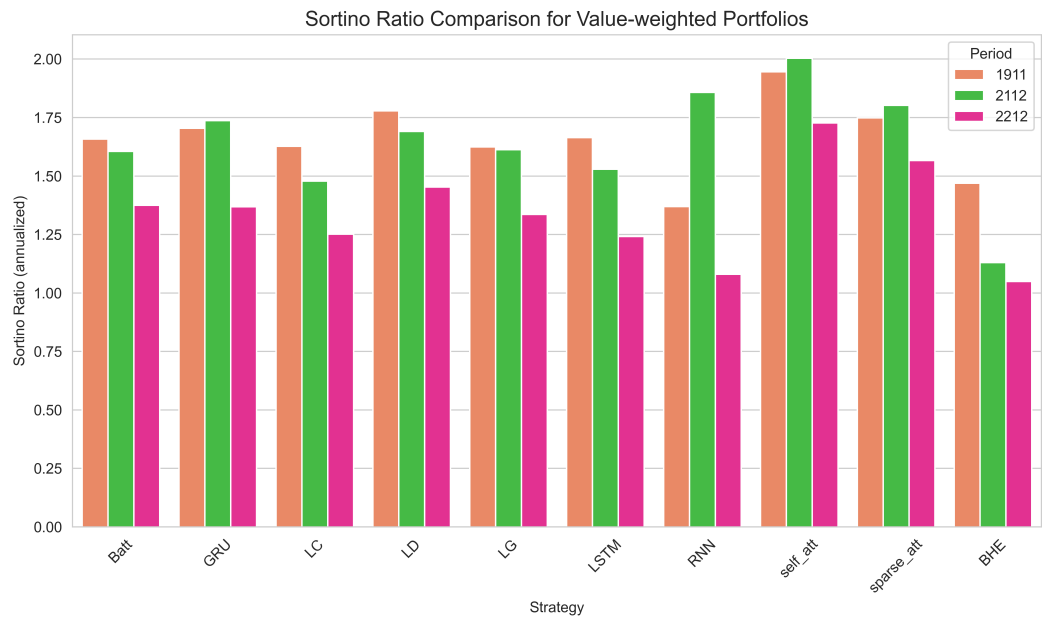
Table 3.6: Value-weighted portfolio performance considering static transaction cost. 'Ann' is the notation for annulization, while 'SR' and 'SO' are short for Sharpe ratio and Sortino ratio respectively. The notation of 'Std' means standard deviation, and 'BHE' means buy-and-hold strategy in equal-weighted portfolios. The values in the parentheses indicate negative values. 'Ann Return' and 'Max Drawdown' can be transformed to a percentage presentation by multiplying by 100.

tives, followed by the `sparse_att` model. And the difference between the proposed models is negligible. All models perform similarly in Period 1911 and Period 2112, which indicates less advantage shows up for these models on larger-cap stocks when the market has a mild or sharp uptrend. However, the Sharpe ratios are significantly reduced when the market slumps. This phenomenon is less significant in the equal-weighted method. This provides indirect evidence that these models exhibit stronger effectiveness in moderating downside risks for volatile (smaller-cap) stocks. The information from the diagrams of the Sortino ratios in both portfolio weighting methods agrees with this stance. However, the noticeably high sortino ratio of the vanilla RNN model in Period 2112 reflects the sensitivity of the vanilla RNN model towards the downside risks caused by the extreme market turbulence when larger market capitalization presents. Model GRU shows similar characteristics.

From the cumulative return plot Figure 3.19 in the value-weighted method considering static transaction cost, we can see value-weighted portfolios narrow the difference between models to a degree during the pandemic and post-pandemic period, except for the vanilla RNN model, `self_att` model and buy-and-hold (BHV) benchmark. The performance of the proposed `sparse_att` model is stable in both the equal-weighted portfolios and the value-weighted portfolios, and it ranks as the best model after 2016 due to the best model in the equal-weighted portfolios, `self_att`, dropping behind all RNN attention benchmark models and RNN variation models and losing its advantage when heavy weights are added to the larger-cap stocks. This phenomenon implies that the `self_att` model latently performs better on smaller-cap stocks, and it is sensitive to variations in time series. The variation is larger, the performance of the `self_att` is better. A similar phenomenon happens on the GRU model, while it drops right before the rank of the `self_att` model during the market turbulence in the value-weighted case. However, the other RNN variation model, LSTM, presents a reasonable capability in profitability among the larger-cap stocks, which ranks in first place before 2016 and second place after 2016. Likewise, the L-DotProd model remains in third place in the value-weighted case, which means Model `sparse_att` and Model L-DotProd are less likely to be affected by the size of stocks' market capitalization. They consistently possess great back-testing performance, which implies they are suitable for stocks with different levels of



(a) Annualized Sharpe ratio comparison



(b) Annualized Sortino ratio comparison

Figure 3.18: Annualized Sharpe ratio and Sortino ratio for value-weighted considering the static transaction cost. 'Batt' indicates the additive attention model (B-Additive), 'LC', 'LD' and 'LG' mean L-Concat, L-DotProd and L-General attention model, respectively.

volatility. It is also interesting to notice that the vanilla RNN model performs worse in cumulative return plots. It shows the capacity for hedging downside risks instead of deriving the absolute returns, but it does not work well during market turbulence when heavy weights are given to larger-cap stocks. It indicates a preference for volatile stocks.

The value-weighted method is also examined for its robustness by considering its dynamic transaction costs (turnover and 20bps for large-cap). Table 3.7 presents the backtesting results of value-weighted portfolios considering the dynamic transaction cost. Figure 3.21 shows the variation of cumulative returns over time with the consideration of turnover rate. Figure 3.20 visualizes the annualized Sharpe and Sortino ratios of three periods as a comparison of the profitability in terms of risks for each model-based strategy in three market regimes. The robustness test results support the conclusion drawn from the equal-weighted methods' robustness test.

3.6 Conclusion

This study investigates the pretrained RNN attention models with the mainstream attention mechanisms such as additive attention (Bahdanau et al., 2015), Luong's three attentions (Luong et al., 2015), global self-attention (Vaswani et al., 2017) and sliding window sparse attention (Beltagy et al., 2020) for the empirical asset pricing assignment. The research emphasises the pretrained RNN global self-attention model (self.att) and the pretrained RNN sliding window sparse attention model (sparse.att) as they are the innovative structures in different domains, especially for NLP. Less literature systematically investigates the role of attention mechanisms in the context of asset pricing. This is the first empirical asset pricing research which employs the RNN frames with the global self-attention mechanism, sliding window sparse attention mechanism and Luong's three attention mechanisms (Luong et al., 2015). They solve or moderate the issues in classic ML model-based empirical asset pricing research, such as sequence or time series data recognition, models' financial or economic interpretability and overfitting caused by over-parameterization of the ML models. It also investigates the necessity of the attention mechanisms in the context of ML-based asset pricing by construct-

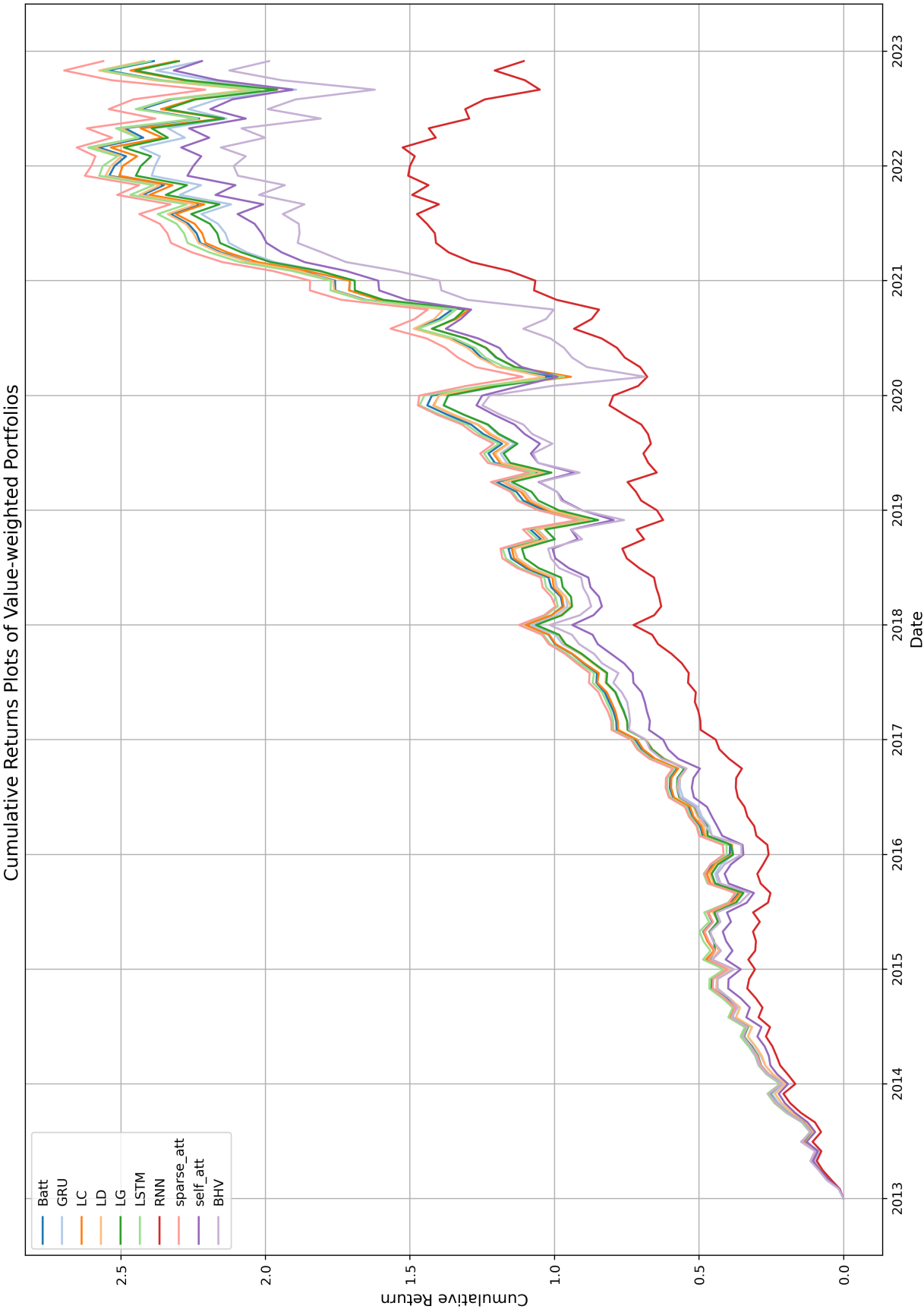
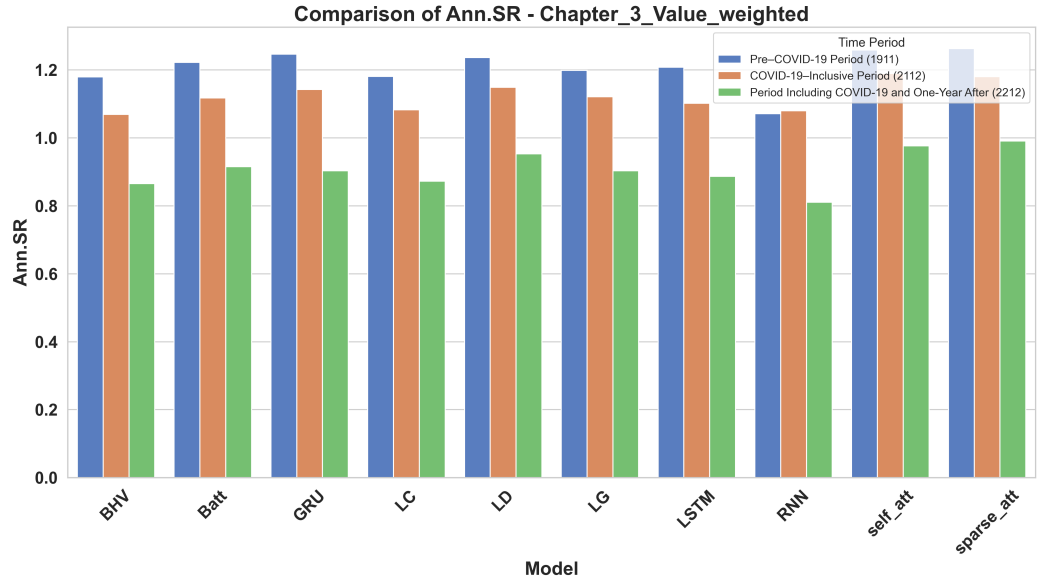


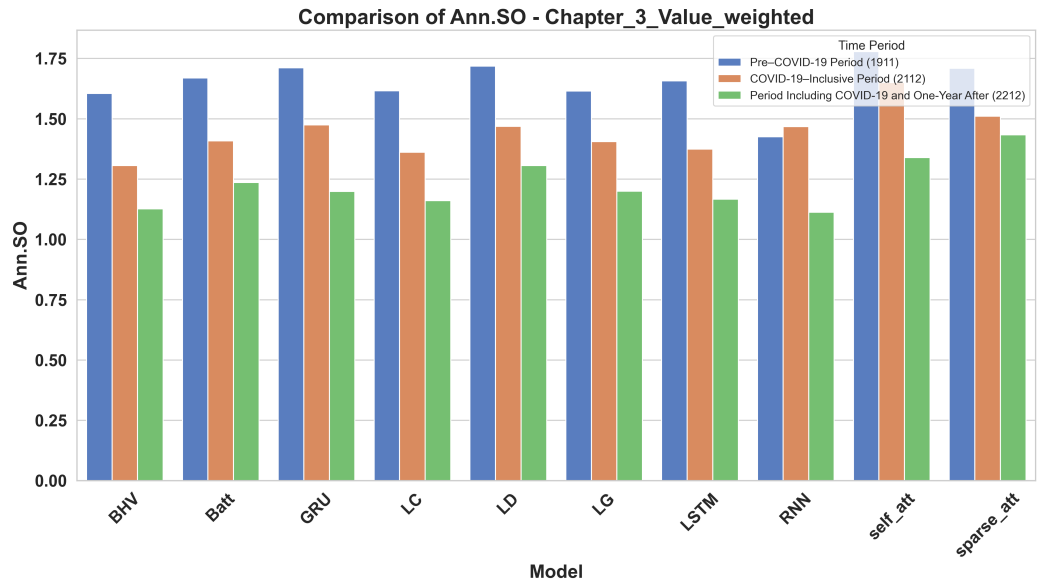
Figure 3.19: Cumulative return plot for value-weighted portfolios considering the static transaction cost. 'Batt' indicates the additive attention model (B-Additive), 'LC', 'LD' and 'LG' mean L-Concat, L-DotProd and L-General attention model, respectively.

Model	AR	Ann.SR	SR	Ann.SO	SO	MDD	Turnover
Pre-COVID-19 Period (1911)							
BHV	0.1388	1.1795	0.3405	1.6054	0.4634	-0.1297	0.0120
B-Additive	0.1445	1.2223	0.3529	1.6689	0.4818	-0.1311	0.0589
GRU	0.1446	1.2462	0.3598	1.7117	0.4941	-0.1265	0.0693
L-Concat	0.1394	1.1812	0.3410	1.6162	0.4666	-0.1317	0.0434
L-DotProd	0.1458	1.2361	0.3568	1.7188	0.4962	-0.1218	0.0747
L-General	0.1399	1.1986	0.3460	1.6147	0.4661	-0.1322	0.0656
LSTM	0.1425	1.2075	0.3486	1.6573	0.4784	-0.1286	0.0446
RNN	0.1152	1.0714	0.3093	1.4257	0.4116	-0.1046	0.4718
self_att	0.1459	1.2586	0.3633	1.7786	0.5134	-0.1150	0.1142
sparse_att	0.1475	1.2627	0.3645	1.7094	0.4935	-0.1256	0.0847
COVID-19-Inclusive Period (2112)							
BHV	0.1540	1.0688	0.3085	1.3057	0.3769	-0.2352	0.0093
B-Additive	0.1601	1.1172	0.3225	1.4088	0.4067	-0.2244	0.0516
GRU	0.1597	1.1421	0.3297	1.4750	0.4258	-0.2133	0.0922
L-Concat	0.1555	1.0829	0.3126	1.3614	0.3930	-0.2273	0.0425
L-DotProd	0.1630	1.1487	0.3316	1.4691	0.4241	-0.2182	0.0674
L-General	0.1566	1.1208	0.3235	1.4051	0.4056	-0.2151	0.0599
LSTM	0.1578	1.1020	0.3181	1.3744	0.3968	-0.2262	0.0450
RNN	0.1386	1.0797	0.3117	1.4675	0.4236	-0.1842	0.4679
self_att	0.1624	1.1891	0.3433	1.6505	0.4765	-0.1971	0.1168
sparse_att	0.1652	1.1802	0.3407	1.5108	0.4361	-0.2118	0.0901
Period Including COVID-19 and One-Year After (2212)							
BHV	0.1308	0.8654	0.2498	1.1270	0.3253	-0.2352	0.0083
B-Additive	0.1371	0.9155	0.2643	1.2361	0.3568	-0.2216	0.0540
GRU	0.1333	0.9033	0.2608	1.1988	0.3461	-0.2224	0.0990
L-Concat	0.1317	0.8727	0.2519	1.1607	0.3351	-0.2270	0.0392
L-DotProd	0.1405	0.9526	0.2750	1.3059	0.3770	-0.2086	0.0652
L-General	0.1338	0.9033	0.2608	1.1996	0.3463	-0.2182	0.0548
LSTM	0.1334	0.8867	0.2560	1.1663	0.3367	-0.2282	0.0397
RNN	0.1094	0.8103	0.2339	1.1122	0.3211	-0.2144	0.4904
self_att	0.1439	0.9905	0.2859	1.4336	0.4138	-0.1950	0.1101
sparse_att	0.1460	0.9765	0.2819	1.3391	0.3866	-0.2023	0.0810

Table 3.7: Value-weighted portfolio performance considering the dynamic transaction cost (turnover and 20bps). ‘Ann’ is the notation for annulization, while ‘SR’ and ‘SO’ are short for Sharpe ratio and Sortino ratio respectively. The notation of ‘MDD’ means Maximum drawdown, and ‘BHV’ means buy-and-hold strategy in value-weighted portfolios. The Annulized return (AR) and ‘MDD’ can be transformed to a percentage presentation by multiplying by 100.



(a) Annualized Sharpe ratio comparison



(b) Annualized Sortino ratio comparison

Figure 3.20: Annualized Sharpe ratio and Sortino ratio for Value-weighted portfolios considering dynamic transaction cost (turnover and 20bps). 'Batt' indicates the additive attention model (B-Additive), 'LC', 'LD' and 'LG' mean L-Concat, L-DotProd and L-General attention model, respectively.

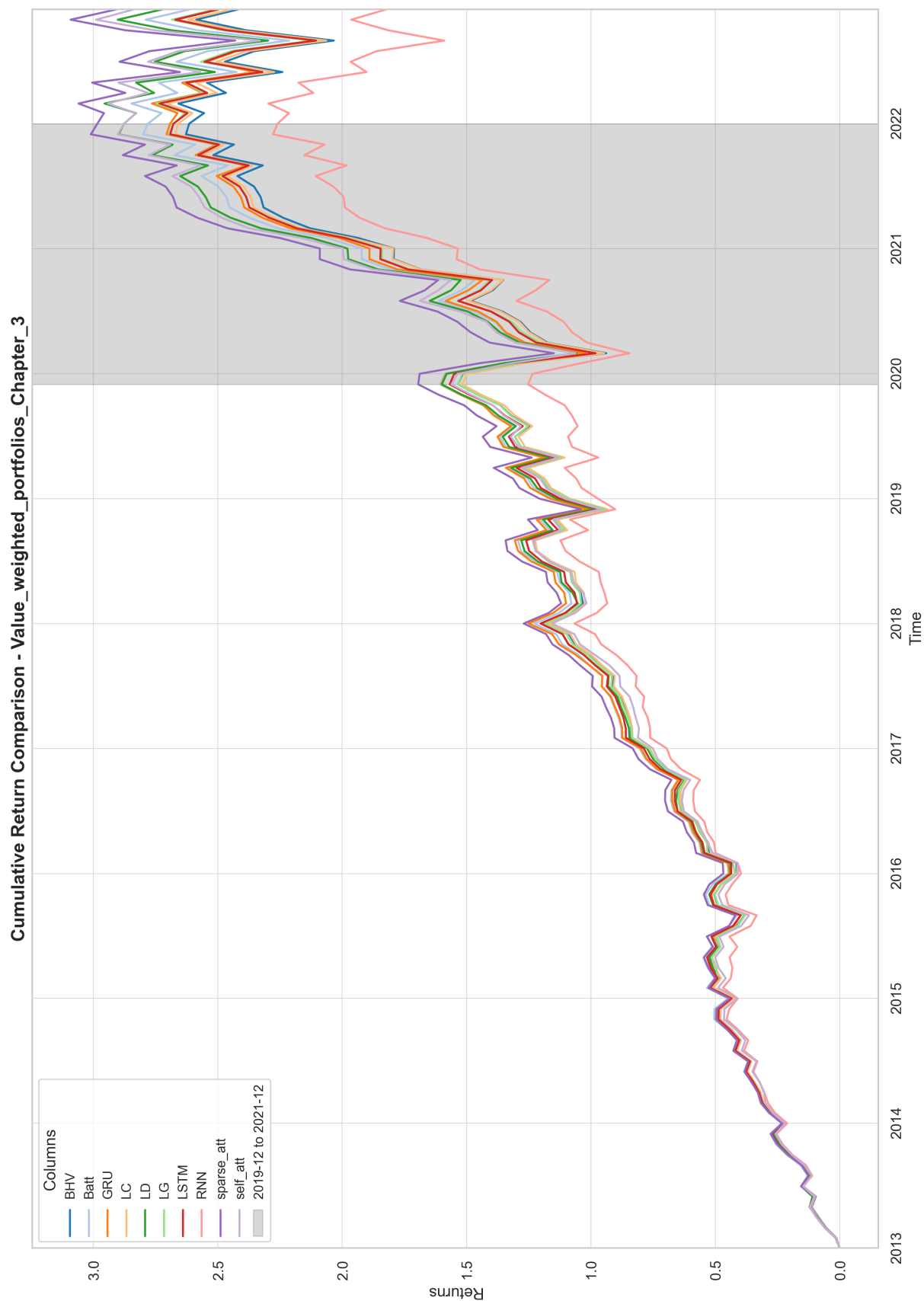


Figure 3.21: Cumulative return plot for Value-weighted portfolios considering the dynamic transaction cost (turnover and 20bps). 'Batt' indicates the additive attention model (B-Additive), 'LC', 'LD' and 'LG' mean L-Concat, L-DotProd and L-General attention model, respectively.

ing the benchmark models of the pre-trained RNN additive attention model (B-Additive), Luong’s three attention models (L-General, L-DotProd, L-Concat) and no-attention RNN models (RNN, LSTM, GRU). Moreover, all models are evaluated and portfolio-wise back-tested in three periods: Pre-COVID-19 Period (1911), COVID-19-Inclusive Period (2112), and Period Including COVID-19 and One-Year After (2212) to prove whether the model has the capability to handle extreme market conditions. These capabilities include absolute and risk-considered profitability during extreme market turbulence. In addition, research in this chapter also investigated how the turnover rate could affect the profitability of these models via the dynamic transaction cost robustness examination. In summary, this study pursues two objectives across two dimensions: 1) to examine whether the factors in the proposed and alternative structures can better explain the stock excess returns and satisfy the no-arbitrage assumption in the background of ‘factor zoo’, which is one direction of empirical asset pricing research. 2) to examine the predictability and trading strategy-wise profitability of these models, which is the goal of the factor investing studies for practitioners.

The results from model performance evaluation find that the vanilla RNN model has the highest model out-of-sample fitness, followed by Model `self_att` and Model `sparse_att`, during all periods. The proposed models present stability during the extreme market fluctuations, especially for the `sparse_att` model. This confirms the predictability of the vanilla RNN model and the proposed models to some extent. The average positive values of the residual α and the distribution plots of α provide strong evidence of market inefficiencies and excess predictability across the three periods. These anomalies suggest uncaptured non-linear risk premiums, which is where factor investors can potentially benefit from. These are anomalies caused by mispricing in empirical asset pricing literature, and it is where factor investors can benefit from. The significant increase in the average α of all models in Period 2112, which follows the trend of the price index, along with the modest differences between models within each period, further reduces the likelihood of model misspecification. This study successfully explores the meaning of the residual α in ML-based asset pricing.

From portfolio-wise backtesting, the `self_att` model outperforms all alternative models when

returns are considered in conjunction with risks in both equal-weighted portfolios and value-weighted portfolios, followed by the `sparse_att` model. However, when returns are not considered with risks, Model `self_att` is the best in equal-weighted portfolios, but deteriorates in value-weighted portfolios. Instead, Model `sparse_att` shows some stability, which is in the second place of equal-weighted portfolios but is the best in the value-weighted portfolios. This implies that the `self_att` model performs better on smaller market-cap stocks. This verifies the conclusions from the self-attention literature, which includes the classic Transformer literature, that the self-attention mechanism has higher sensitivity to the large variation between time steps, which makes it capable of modelling the highly volatile data. On the other hand, the `sparse_att` model is not affected by the stock market capitalisation. It stably provides excellent performance without considering the volatility of the stocks. Namely, it is more generalized than the `self_att` model, which is a notable advantage for both practitioners and academic scholars. Additionally, the benchmark RNN attention models in equal-weighted portfolios generally equal or outperform the models without an attention mechanism, apart from the RNN Luong's concatenate attention model (L-Concat). Nevertheless, it varies in value-weighted portfolios. The market capitalization weights seem to significantly reduce the benefit of attention mechanisms if only considering the absolute capital gain. Moreover, value-weighted portfolios narrow the difference between models. Thus, the advantage of the attention mechanism is not as significant as in the equal-weighted portfolios. The overall performance of value-weighted portfolios is higher than that of equal-weighted portfolios. Approximately, all models somehow show the capability of handling the downside risks during the period with pandemic effect, while all models outperform the buy-and-hold strategy in both portfolio groups, excluding the vanilla RNN model. And it is also interesting to notice that this model fails in the backtesting model competition, especially on the absolute returns of value-weighted portfolios. This indicates that it is easily affected by the stock volatility level. From the perspective of the Sharpe ratios, this work supports the findings from Chatigny et al. (2021) that in value-weighted portfolios, the large-cap stocks achieve the annualized Sharpe ratio above 1.0, but it does not persist during the period that the market was experiencing the sideways movement with extreme fluctuations. Furthermore, the transaction cost robustness examination illustrates that controlling turnover is critical

for maintaining the profitability of these models. The dot-product-based attention mechanisms (L-DotProd, self_att, sparse_att) have significant advantages in generating more stable and selective trading signals. However, although the RNN model shows persistence in risk control, its profitability significantly decays due to the high turnover. Thus, high forecasting accuracy does not guarantee absolute profitability.

This study also finds some interesting research questions for further discussion. Although this work covers large-scale factors, these factors are pre-selected, which indicates that the issue of omitted factors may exist. The positive α values also imply this issue. Thus, alternative methods can be proposed as a substitute for the pre-selecting factors. Moreover, literature on ML-based asset pricing neglects the discussion on how to decide the number of abstracted or transformed factors. It is worth exploring the method for determining factor numbers via the ML approaches. Furthermore, it could be interesting to explore the structure of α s and how to interpret these α s in the ML-based asset pricing research. Finally, this study only adopts the simple asset allocation strategies, equal-weighted and value-weighted, but intuitively, ML methods can be deployed for asset allocation to further improve the strategic returns of portfolios. In addition, to further improve the financial and economic interpretability of these neural network models, the DM test can be applied to individual stocks to investigate the model's preference for an industry or sector, while investigating further on MSE and OOS R^2 for the contribution of characters, for example, which industry sectors have higher predictive power in a certain framework, and what is the impact of market capitalization size. Also, for attention models, attention weights variation over time can be examined to investigate which factors are emphasized during the pandemic period, and conduct subset factors from the factor pool as factor sensitivity analysis. As future work, these advanced neural network models can also be upgraded by altering the specific modules, such as adding an additional MLP structure within the Transformer core structures. Moreover, these structures can be applied to the volatility modelling for asset allocation and risk controlling, exemplified by Ma et al. (2023).

To conclude, 'attention' is something worth exploring further, but far from 'all we need' in

the context of the empirical asset pricing.

Asset Pricing in Pre-trained Transformers

4.1 Introduction

Previous studies on machine learning (ML) models for asset pricing or factor investing tasks, such as Gu et al. (2020, 2021) have already proved that some traditional ML algorithms, for example, the multi-perception (MLP) and random forest (RF), outperform the traditional econometric models when they are applied as model frameworks of asset pricing models. The shallow MLP models show advantages in coping with extreme market fluctuations, such as those caused by the COVID-19 pandemic, with better out-of-sample performance and universal stock adaptivity among large-cap U.S. stocks. Nevertheless, the MLP model still has drawbacks such as overfitting, overlapping (under the method of GKX2020), high sensitivity towards missing values and data noises, as well as low computational efficiency, as it is the simplest neural network (NN) structure with no natural design for detecting the temporal dependency of the time series data. Specifically, inappropriate methods for handling missing values during data preprocessing may introduce significant bias. For example, simply removing the stocks or factors may cause the ‘survival bias’ and filling identical numbers, such as mean or median values, also affects the reliability of the data and the accuracy of prediction if the missing values occupy a considerable fraction of the data. On the other hand, the added complexity of MLP models compared with traditional statistical models multiplies the computational cost. Chapter 3 employed the MLP autoencoder pre-training method together with a recurrent neural network structure to address the disadvantages of MLP models for empirical asset pricing. Moreover, it employs different attention mechanisms to capture longer-term temporal dependency for improving the stock return forecasting accuracy.

However, the single-output MLP method individually calculates the stocks, which multiplies the computational cost, especially if the dataset contains thousands of stocks or is extended in length. Moreover, it does not consider cross-sectional effects among individual stock returns, which reduces the model's economic or financial meaning. Thus, the research question remains 'is there an improved NN structure that satisfies the conditions, including further moderating the overfitting issue, higher computational efficiency and balancing the trade-off between model interpretability and predictability?'. Aydogan-Kilic and Selcuk-Kestel (2023); Gu et al. (2021); Rukmana et al. (2024) attempt to utilize the alternative neural network (NN) structures, including recurrent neural networks and their variations or MLP autoencoders, to solve these issues for stock price forecasting, but the aforementioned issues still have not improved significantly.

Fortunately, the recent boom in large language models (LLMs) provides a potential solution. The standard Transformer model (Vaswani et al., 2017), as the foundation of most commercial LLMs such as openAI's ChatGPT, Google's Gemini, X's Grok and DeepSeek, is built upon an autoencoder NN structure, which is assembled with encoder and decoder blocks. Each encoder or decoder block of a standard Transformer contains an MLP autoencoder module with an auxiliary self-attention layer for capturing and processing the inputs. In between the encoder and decoder blocks, a cross-attention layer is configured to capture the spatial-temporal relationship between input and output data (observable factors and stock excess returns) to improve forecasting accuracy. Since the fundamental structure of the Transformer is the autoencoder structure, which has multiple neurons for multiple outputs, together with the dot-product-based self-attention mechanism, which highly reduces the parameters' size. They parallel the computation process of multi-stock return series through the autoencoder structure and reduce the parameter estimation workload. Thus, the design substantially increases computational efficiency. Moreover, computing stocks' returns simultaneously and jointly optimizing multiple stocks' loss functions also enables the scaling of cross-sectional effects between stocks. Both the self-attention mechanism and the cross-attention mechanism are originally designed for textual information, as the content vectors

have different lengths. Therefore, with proper embedding settings, linear in this case, they perfectly fill the missing value by projecting the original input factors. Although it was originally invented for natural language processing (NLP) tasks such as machine translation and context prediction, as a sequence-to-sequence (Seq2Seq) model, it is found highly suited for time series prediction as well (Gezici and Sefer, 2024a; Päivärinta, 2022; Zhang et al., 2022).

Furthermore, researchers have developed various Transformer-based architectures, such as BERT, Realformer, and GPT, some of which have been successfully applied to financial time series forecasting Cheng and Tang (2024). However, adapting the standard NLP-focused Transformer for time series tasks requires specific architectural modifications. For instance, strict causal masking must be incorporated into both the encoder and decoder to prevent look-ahead bias (data leakage), even when employing a rolling window approach. Additionally, the standard word embedding layer must be replaced with a linear embedding layer to process continuous numerical data effectively. Cong et al. (2021a); Ma et al. (2023); Zhang (2022) further attempt the Transformer in the context of factor investing and stock pricing on different datasets, but the results are variable according to the stock markets and data methods they are employing. Although it is a formidable challenge applying the LLMs to temporal sparsity data, which is a typical characteristic of financial and economic data that has long-term dependency but fewer observations, for instance, monthly or quarterly economic time series, their work sheds light on advanced machine learning (ML) for empirical asset pricing topics. However, compared with the financial time series, the temporal sparsity characteristic of economic data is more restricted since they are commonly low frequency and have longer time-dependency, hence fewer ML algorithms are explored successfully for economic applications such as empirical asset pricing and factor investing research, not to mention the state-of-the-art (SOTA) large language model (LLM) as Transformer. It is worth putting more effort into exploring the upgraded structure for higher suitability of economic data. In this research, the innovative Transformer model, multi-head single-directional representative Transformers (SERTs), for asset pricing and factor investing tasks, are developed and examined on large-cap U.S. stocks. Also, the innovative pre-trained Transformer models

are examined in the context of empirical asset pricing and factor investing. SERT alters the random mask pre-train module of BERT to an MLP autoencoder pre-train module for numerical data, and it enforces the causal masks for the encoder self-attention layer to prevent future data leakage. The innovative pre-trained Transformer simplifies the inside MLP structure and adds the MLP pre-training module for improving the input factors' quality.

Therefore, the contributions of this study are from four aspects:

- Developing the innovative financial economic context LLMs, SERTs and MLP autoencoder pre-trained Transformer, which enriches the scarce literature in ML for asset pricing and factor investing.
- Verifying whether general model improving methods such as multi-head attention, pre-training and layer normalization first (LNF) apply on the specific LLMs for temporal sparsity data such as stock pricing data.
- Offering an anatomy of the 'black-box' Transformer models for boosting the transparency and interpretability of the complicated LLMs in the context of stock pricing and factor investing. This increases the financial and economic meanings of the models to a degree and corrects the bias of overthinking the positive relationship between parameter size, model complexity and overfitting (Didisheim et al., 2024).
- It is the first attempt for the SERT and pre-trained Transformer models to directly work on stock pricing and factor investing. Also, it is the first comprehensive research on fundamental LLM in the context of extreme temporal sparsity data.

This study contains 6 main sections. Section 2 presents the literature review. Section 3 exhibits the data that have been used, while Section 4 discusses the Models employed in this study. Section 5 shows the empirical experiment and results, and Section 6 gives the conclusion and further discussion.

4.2 Related work

Although RNN models have highly improved the forecasting performance compared with the MLP models, as proved by Chen et al. (2024); Zhou and Wang (2024), the defects of the gradient vanishing and explosion and low computational efficiency due to the sequential gradient calculation as a substitute for parallel computation. Thus, the core of the Transformer model, the attention mechanism, was developed for efficiently capturing variable dependency. The classic Transformer models are developed for sentence translation and sentence prediction; thus, they are potentially suitable for short-length data such as monthly returns.

In the early stage, people developed soft attention to add to the RNN structure for improving the prediction accuracy in NLP tasks. The seminal work of the soft attention mechanism comes from Bahdanau et al. (2015), which also named as additive attention or Bahdanau's attention. The first soft-attention mechanism utilizes the attention-weighted sum of all input states to generate a new hidden state at each time step via a feed-forward network. This mechanism allows the model to dynamically focus on relevant features of the input sequence at each time step. Bahdanau's attention is also named additive attention. Since the attention score was calculated at every time step, it has a drawback of parameter oversizing which latently causes overfitting. It also lowers computational efficiency. Additionally, it has more hyperparameters that require manual tuning, which increases the model training time and reduces the stability of the training results. Also, since it relies on the soft alignment mechanism, it is more suitable for short-term forecasting instead of long-term trend capture, especially in sparse economic data. It was used in financial applications such as Li et al. (2018); Zhang et al. (2019). Concretely, Li et al. (2018) proposed an Attention-based Multi-Input LSTM model for stock price prediction. They used the dual-stage attention mechanism (Qin et al., 2017) to improve the LSTM capability for capturing the patterns of stock pricing and denoise. To handle the noise introduced by naively concatenating auxiliary factors, such as prices of correlated stocks and market indices, with the target stock's historical series, they separated a mainstream factor from multiple auxiliary factors. Their

MI-LSTM processed each factor stream independently and employed mainstream-controlled input gates to regulate auxiliary influence, with a dual-stage additive attention mechanism: the first stage dynamically fused candidate cell-state updates across the different factors via additive attention scores, while the second stage applied the attention to measure the importance of different time steps. The model was evaluated on historical opening price data from 261 stocks in the Chinese stock market (components of the CSI-300 index) from April 25, 2013 to May 15, 2017. They found their attention-based MI-LSTM significantly outperforms standard LSTM, simple concatenated multi-input LSTM, and alternatives in mean squared error (MSE). The additive attention-based model is validated for the effectiveness of its hierarchical factor modelling and attention-driven selective combination. This work highlighted the effectiveness of the attention mechanisms in managing noisy multivariate financial time series for the forecasting tasks. However, it has limitations such as focusing on short-term (next-day) opening price prediction in a single market, configuration on a short time window ($T=10$), and latent challenges in generalizing to alternative markets and assets, longer windows, and periods with extreme fluctuations, such as the COVID-19 period. Zhang et al. (2019) proposed AT-LSTM, a two-stage attention-based Long Short-Term Memory (LSTM) model for financial time series forecasting. In the first stage, the additive attention mechanism is applied to distribute the attention weights to input features (e.g. opening price, closing price, trading volume, and related market variables) at each time step. This mechanism computes attention weights based on the previous LSTM hidden state and a linear transformation of the input feature sequence. With the attention layer, the most relevant features are emphasized and the less informative features are suppressed. In the second stage, these attention-weighted representations are fed into a two-layer LSTM network to capture long-term temporal dependencies and predict the next day's closing price. The model was evaluated on daily data from three major U.S. stock indices: the Russell 2000, the Dow Jones Industrial Average (DJIA) and the Nasdaq from January 2, 1991, to April 30, 2018. Their results demonstrated that AT-LSTM consistently outperformed both the traditional ARIMA model and vanilla LSTM across all three datasets. Nonetheless, it acknowledged limitations, including noticeable time lags in predictions and low sensitivity to extreme market conditions. As demonstrated in Chapter 3, the modern attention mechanisms, such as

self-attention and sparse attention, potentially perform better than the additive attention in time series forecasting tasks. This work provides an early example of integrating attention with recurrent networks to enhance predictive accuracy in financial forecasting.

Later, Luong et al. (2015) simplifies the Bahdanau attention via three different functions. They proposed three methods for computing the attention score: dot product, generalizing, and concatenation. The asset pricing fitted form is introduced in Chapter 3. Luong's attentions are also known as the multiplication attention mechanism. It is a big step forward in approaching the self-attention mechanism in the Transformer model since it adopts the dot-product function proposed for computing the attention scores. The mathematical equations evidently show that Luong's attention score calculation has freed itself from the dependence of the MLP structure, while the decoder hidden status updating never requires an RNN structure. In that sense, Luong's attention highly reduces the parameter size and promotes computational efficiency. Thus, the design improves training speed significantly and extends the long-term dependency capture. Few economic or financial applications directly based on Luong's attention are found in the existing literature, but it is applied as a built-in module, for example, in Kim et al. (2019). However, it is the foundation of the self-attention mechanism, which is built in the classic Transformer models. Although additive attention offers strong expressive power and excels at capturing complex, non-linear alignments, making it effective for tasks with complicated dependencies, it is computationally heavier due to the extra feedforward neural network layer and performs unstably with long sequences. In contrast, Luong's dot-product and general attention significantly improves training speed and parameter efficiency, better supporting long-term dependency modeling, though it may exhibit reduced flexibility when dealing with highly non-linear relationships.

Subsequently, the development of attention mechanisms has ushered in a period of prosperity. Numerous attention variations were invented for different contexts. The most well-known attention variations are memory-augmented attention (Weston et al., 2014), hierarchical attention (Yang et al., 2016), dual-stage attention (Qin et al., 2017) and self-attention (Vaswani et al., 2017). Apart from the NLP tasks as they were originally designed, they are used in

different time series tasks as well, for example, Qin et al. (2017) designed the Dual-Stage Attention-based Recurrent Neural Network (DA-RNN), an encoder-decoder framework that integrates two complementary attention mechanisms to enhance multivariate time series forecasting, which is inspired by humans' dual-stage attention of Feature Integration Theory (Preattentive Stage and Focused Attention Stage) (Treisman and Gelade, 1980). Specifically, the model employs an input attention mechanism in the encoder to dynamically select and distribute the attention weights to the most relevant driving (exogenous) series at each time step, guided by the previous encoder hidden state. In its decoder, a temporal attention mechanism selectively aggregates hidden states across time steps of the encoder according to the current decoder hidden state. By combining the dual-stage attention with the LSTM foundation, it can effectively capture the longer-term temporal dependencies. The model is tested on the SML 2010 smart home sensor dataset (daily) and the NASDAQ 100 stock dataset (daily), and it consistently outperforms traditional statistical models, standard LSTM, and other attention-based RNN variants in terms of prediction error. The paper represents a widely referenced early contribution to attention-augmented recurrent networks for multivariate time series, which are verified by (Liu et al., 2020c; Peng et al., 2021; Yang et al., 2022), but due to the large parameter size, it may cause overfitting on monthly asset pricing data latently. In that sense, it cannot compete with the universally recognized one in the Transformer model: self-attention, in the asset pricing context. A detailed description of the self-attention mechanism is presented in Section 4.4.1.3.

A profound revolution occurred in 2017 when the first Transformer model was proposed. Transformer models are the core and the foundation of most LLM industrialized AI robots, for instance, OpenAI's ChatGPT, DeepSeek, X's Grok 3 and Google's Gemini. With the increasing popularity of AI robots, Transformer models have attracted great attention from researchers. People explore Transformer applications across different contexts, as well as structural variations in Transformer models. For example, Zhou et al. (2021a) developed the Informer model, which extends the length of the information that the Transformer can capture, and a review by Zhao (2024) describes the time series applications of the Informer. For handling sparse time series, (Wu et al., 2021a) also proposed the Autoformer model for

longer dependencies. However, economic time series have higher sparsity and longer dependencies than financial time series, which means there is a trade-off between the overfitting caused by oversized parameters and dependency length. These suggest that, for stock pricing factor models, simplified Transformer models may perform better than more complex variants. Wherefore, the single-directional representative from the Transformer (SERT) and pre-trained Transformer with a low-dimensional structure may be suitable for the stock pricing context.

Most existing financial or economic applications of Transformers, or more generalized, LLMs, concentrate on textual information modelling, such as sentiment analysis (Qian et al.; Rehman et al., 2022; Rizinski et al., 2024), behavioral factor exploration (Ferdus et al., 2024), financial and economic news processing (Dolphin et al., 2024; Miori and Petrov, 2024; Sharkey and Treleaven, 2024) or social media monitoring (Li et al., 2022). For example, Kaplan et al. (2023) designed CrudeBERT as a Transformer model for sentiment analysis on the crude oil future market. It identifies and classifies the news headlines that affect the supply and demand of crude oil to predict its price movement. They proved their model was superior to the prevailing FinBERT (Araci, 2019) model on the worldwide crude oil market. Ferdus et al. (2024) offer another example for the Transformer model application in financial textual information capture. They employ the Transformer model for modelling sentimental information from X (Twitter) as external factors and prove that their model excels the LSTM, random forest, and LightGBM, which derive the highest stock price forecasts among the S&P 500 components. In addition to textual data capture, Transformer models are used for speech recognition tasks in finance and economics as well. Gado (2024) innovatively applies the Transformer model (BERT) to extract the sentimental effect from Canadian and U.S. central bankers' speeches. They find that negative Federal speech increases the tail stock returns of both the U.S. and Canada, while negative BoC speech decreases the Canadian tail stock returns but has no effect on U.S. tail stocks. Federal sentimental variation causes changes in BoC sentiment but not 'vice versa', which indicates the single-directional spillover. Their long-only trading strategy based on negative sentimental Federal speeches achieves 47% higher cumulative returns than the S&P 500 and 0.61 in the Sharpe ratio. They

also found that the BERT model encompasses cross-sentence context and performs better in stock returns forecast and strategy-wise investing than the individual-sentence context for sentimental analysis. This great finding inspires us to use the sorted portfolios as factors instead of the firm characteristics for cross-sectional consideration. Li et al. (2022) design a Transformer-based Attention Network (TEA) framework for stock movement prediction. It utilizes textual social media information (Twitter, now X) together with the historical stock prices as predictors to forecast the direction of stock price movement, which excels the traditional benchmark models such as ARIMA, LSTM, StockNet and CNN-BiLSTM-AM, both in accuracy and Matthews Correlation Coefficient (MCC) scores. They find that both sentimental information from X and the price data contribute significantly to facilitating prediction accuracy. Also, the model's performance is highly improved when the price data works together with sentiment information. Miori and Petrov (2024) applied GPT-3.5 and economic network theory in the Wall Street Journal to investigate the relationship between news topic structures and financial market dislocations. They find the key economic topics (e.g., inflation, interest rates, Federal Reserve decisions) dominate market narratives, network fragmentation correlates with financial market shocks, and GPT-based narrative structures provide a higher accuracy view for economic discourse than traditional topic modelling techniques.

Recently, Nie et al. (2024) conducted a survey for the financial application of large language models (LLM), which exhibits examples of LLMs in different tasks of finance. It assists researchers in finding directions in which they are interested. Their introduction emphasizes more on specific financial Transformer variation models for individual tasks rather than the generalized models in different contexts of the financial domain. For example, FinBERT (Araci, 2019) is a model for financial sentiment analysis, and FinVIS-GPT (Wang et al., 2023) is for financial chart analysis. Although their work encourages future researchers to seek models for their assignments as well as demonstrates the challenges of applying these models to a degree, it is insufficiently comprehensive for researchers who are working on it. Fortunately, Zhao (2024) provided a more comprehensive literature review of LLMs for financial applications. Similarly, they classify the LLM for finance into five main categories,

namely, sentiment analysis, question answering, named entity recognition, time series forecasting and mathematical reasoning, but they concentrate on the mechanism of models for distinctive tasks, which furnishes readers with the blueprint of groups of models' capability to achieve. Moreover, they highlight the performance of GPT-4 for different financial tasks and indicate that a textual Financial LLM with the fundamental structure of GPT-4 has better performance than otherwise. Nevertheless, both LLM review papers agree that it is a significant challenge to apply LLMs directly to numerical data, especially time series data with pronounced temporal sparsity. Therefore, developing and adjusting LLM models and making them optimized for numerical data could be a new direction for future researchers to endeavour with.

Compared to research on processing financial or economic textual data, the direct application of Transformer models to numerical financial data is relatively immature. Korangi et al. (2023) use the Transformer model for mid-cap companies' short-term and medium-term default probability prediction. Their encoder-only Transformer model achieves the highest predictive accuracy among the benchmarks of logistic regression, XGBoost, LSTM, and Temporal Convolutional Networks models. Sun et al. (2022) applies a convolutional neural network (CNN) assembled Transformer (named Conv-Transformer) with a graph attention mechanism (GAT) unit on the portfolio optimization task of Chinese CSI 300 stocks. The model optimizes the return-risk ratio directly and improves the asset allocation decisions. Specifically, the Conv-Transformer is deployed to capture temporal dependencies in individual stocks, while the GAT reflects the cross-sectional relationship between stocks. The model dynamically learns the correlation between stocks and overcomes the limitations of static stock relationships in existing methods. Tevare and Revankar (2023) employ the Stack Transformer model for New York Stock Exchange (NYSE) stock price forecasting. They substituted the Sinusoidal Positional Encoding (SPE) positional encoding method with the Time2Vec learnable positional encoding method for a better fit for the time series data and applied their model to daily and weekly stock prices and technical analysis indicators. Their model derives the lowest mean square error (MSE), root mean square error (RMSE) and mean absolute error (MAE) compared with the benchmark LSTM models. Zhang et al.

(2024) develop a novel model named Quantformer and examine it on the Chinese stock market. It tailors the standard Transformer model for time series data via a simplified decoder block of the standard Transformer model and adopts it for building financial trading strategies based on technical analysis indicators (emphasized on the price-volume factors). They notice the effect of different temporal distances of financial time series, but removing masks of the attention layer may cause future data leakage even under the rolling window method. They also verify the Transformer models' capability for handling market fluctuations. Wheeler and Varner (2024) employs the innovative MarketGPT model based on the most prevailing commercial LLM, generative pre-trained Transformer (GPT), for modelling the order flow of the NASDAQ stock market. They invent an order generation engine within a discrete event simulator to enable the realistic replication of limit order book dynamics by building a high-fidelity market simulation environment. The model offsets the drawbacks of traditional time series models (ARIMA, GARCH, LSTM, and GANs), such as missing market microstructure, divergence between generative and real data, and neglecting long-term temporal dependencies. The empirical results of their work show that MarketGPT successfully simulates stylized facts such as heavy-tailed distribution, volatility clustering and long-range dependence. Thus, their model provides an innovative proposition of depicting the unobservable factors for asset pricing factor models. However, MarketGPT has imperfections of high computational cost (requirements of high computational power GPU), flash crash during the extreme market fluctuation period and no direct market trend predictions. This means the model possibly can solve the omitted factor problem to some degree after positive modifications. Lezmi and Xu (2023) provide another encoder-only Transformer model application on multi-step time series forecasting and asset management of the global markets. They examined their model on the daily and weekly main global stock indexes, such as the S&P500 and Euro Stoxx 50, as well as 10-year global futures markets, such as the U.S., Germany and UK. They construct a trading system based on a trend-following strategy in which trading signals are generated through the model's binary trend prediction and a portfolio optimization process based on the model's volatility forecasting. It proved the effectiveness of the model for long temporal dependency capturing in financial data among the traditional models such as ARIMA, GARCH and LSTM. They also confirm that the

self-attention mechanism allows the model to efficiently analyze both temporal and spatial relationships. Nonetheless, their model struggles to handle extreme market fluctuations during the COVID-19 period. Moreover, the short step size configuration of the rolling window method may cause overlapping problems.

Less literature on LLM model applications on numerical economic data is found, particularly ‘Transformer for asset pricing’. The most relevant works are (Cong et al., 2021a; Gabaix et al., 2024; Ma et al., 2023; Zhang, 2022). Ma et al. (2023) utilize a CNN-assembled stacked encoder-only Transformer model as a factor model framework for stock return and volatility forecasting in the Chinese stock market. Concretely, they deploy 72 firm characteristics and 8 macroeconomic indicators as the monthly observable factors to model and predict 2466 Chinese A-share stocks. In their trading system, the returns forecast is for trading signal generation, and the volatility forecast is for asset allocation (portfolio optimization). Their model with four encoder stacks achieved both the best OOS R^2 and investing performance among the benchmarks of linear regression, principal component analysis (PCA), random forest and LSTM. Cong et al. (2021a) conduct a comparative study on commonly applied NN models, which include RNN, LSTM, bi-directional LSTM, LSTM with attention mechanism, GRU and encoder-only Transformer. They applied models on the monthly returns of over 20000 stocks from 1970 to 2016 and deployed 54 firm characteristics as the factors. However, all the models they applied encountered surprisingly low OOS fitness. The highest OOS R^2 appears in the LSTM model, which is 0.45%, and the lowest one is the encoder-only Transformer, which only has 0.12%, followed by the RNN model (0.13%). This paper does not mention how the trading signals are generated, hence the investment performance indicators, such as Sharpe ratio and annualized returns have less reference significance. This work is possibly a warning for researchers who misunderstand the relationship between data size and model complexity, and evidence that a simplified Transformer structure may not successfully handle an overly large output dimension. In the factor model case, the stock number N cannot be far beyond the time length. However, Zhang (2022) conducts a similar large-scale comparative study of the U.S. stock market and reports more promising results. Some differences from Cong et al. (2021a)’s work, Zhang (2022) organizes the input factors

via Gu et al. (2020) method and adjusts the fixed time span to a dynamic time span from 5 to 40 years. They find that all their deep learning models outperform the traditional statistical models, and the RNN with attention mechanism model, and the Transformer model have the highest OOS model fitness. For investment performance, the RNN with attention mechanism achieves the highest Sharpe ratio, which is superior to all alternatives. Didisheim et al. (2024) present a clue for this phenomenon from their Artificial Intelligence Pricing Theory (AIPT), which challenges the argument that ‘sparse factors work better’ believed by the traditional Arbitrage Pricing Theory (APT). They enlarge the dimension of the input non-linear factors up to 360000 and find that it significantly reduces the pricing errors and promotes the Sharpe ratio. Consequently, they prove that large complex factor models surpass the traditional sparse models from statistical theory and further prove that even if the factor numbers are far beyond the observation numbers (time length), their conclusion still holds since large factor models considered the complex interaction between factors and have a better approximation to the true Stochastic Discount Factor (SDF). They also refute the alternative linear dimension reduction methods, such as PCA, which have better performance than the large factor models. This inspires us for the proposed pre-trained transformer models and SERT models. However, no matter the firm characteristics as factors or factors organized by the method of GKX2020, it increased the completeness of the input information but highly sacrificed the economic meanings. Thus, it is worth attempting to utilize firm characteristic-sorted portfolios as factors to improve the economic meanings. Ye et al. (2024) provided a comprehensive review of ML for empirical asset pricing. They introduce ML methods as solutions from different angles of empirical asset pricing, for example, return prediction, price direction predictions, ranking of assets and portfolio optimization. Their work discloses substantially interesting future research directions for researchers to work on. Their introduction encompasses task-specific ML models without a detailed explanation, which may struggle audiences without a computer science or machine learning background. Giesecke (2024) provides a detailed overview of ML applications in frontier economic research. They focus on five categories of ML applications for economics: data curation, sentiment analysis, clustering, classification, and function approximation. Among the five categories, data curation is the revolutionary application according to other literature. They find the ML al-

gorithm can capture textual information from government reports, which enables extracting new factors from these textual reports for economic prediction tasks, for example, Giesecke (2023) applied the LayoutLMv3 and LLM models to collect and restructure debt maturity data from more than 400,000 pages of financial disclosures, then utilizes them for predicting government bond rates. They illustrate the challenges in AI-driven economics research, such as computational cost, model interpretability and data quality, hence a hybrid approach with the deterministic (economic rules) and stochastic (ML insights) methods should be considered for addressing these challenges. Gezici and Sefer (2024b) applied the vision Transformer models (Vision Transformer, Data Efficient Image Transformers) and vision neural network models (ConvMixer) on nine volatile ETFs' prices and technical analysis indicators from January 1st, 2002, to January 1st, 2022. They visualized the ETFs' data into a two-dimensional 65×65 pixel figures, then utilized the advanced vision neural network models for generating trading signals such as 'buy', 'sell' and 'hold'. They confirmed that the ConvMixer has the highest prediction accuracy (87.94%), but since the high occupation of the 'hold' signals, the model's profitability is inferior to the vision Transformer models (achieved the annualized return of 17.19%). Although it advances the application of advanced neural network models research on alternative financial data (images), it still has limitations. For instance, high unnecessary computational cost due to the images being transformed from the numerical data. Additionally, they simplified one dollar per transaction as the static transaction cost, which can not reflect the real trading environment and how the turnover rate could affect the profitability of these advanced models. And they did not show any generalised examination results of their models on alternative ETFs or different assets, which means the sample size of nine ETFs is relatively small. Thus, compared with their work, the proposed SERT models and pre-trained Transformer models highly improve the computational efficiency and are more realistic in the transaction cost configurations. Also, the proposed models are tested on 420 large-cap U.S. stocks, which covers a larger sample. Wang (2025) verified the existence of short or long-term dependencies (autocorrelation) and seasonality (e.g. January effect) effects in U.S. stock returns, and cast a doubt on Gu et al. (2020)'s work that shallow MLP models work better on stock return prediction. They employ the Autoformer model, which was designed specifically for time series data on GKX2020's data set and data method from

1957 to 2021. They followed the rule-of-thumb ML data split, which uses 70% of the data for training, 20% for validation, and 10% for testing. The Autoformer model, which replaces the traditional self-attention mechanism in Transformer models with a Fast Fourier-based autocorrelation detection mechanism, and embeds series decomposition blocks within the encoder and decoder to isolate data into ‘seasonal’ and ‘trend-cyclical’ components. Their model also extended the neural network depth to a dual-block autoencoder structure with two attention heads. They examined their models on the step-size of 1 month, 3 months and 12 months, and found that the predictive performance is significant in the longer term due to the low-frequency structures, trend components, and seasonality being stronger over longer horizons. From a profitability perspective, their model achieves an excellent Sharpe ratio of 3.40 (peaking at 4.41). Nonetheless, they did not consider the transaction cost, turnover rate and the short-side transaction borrowing rate, as well as evaluate how the turnover rate influences the profitability. Moreover, due to the extensive factor dimension and the deep neural network structure, the large parameter size (over 40,000) requires massive computational costs, which is not quite realistic in industrial implementation. Additionally, they did not implement the vanilla Transformer as the benchmark to show the advantage of the innovative Autoformer model. Therefore, the proposed SERT and pre-trained Transformer could be a better solution for balancing predictive performance and computational cost. Furthermore, in this Chapter, the static transaction cost method is employed to evaluate the models’ profitability in a conservative penalty environment, and the dynamic transaction cost (Turnover rate with 20bps for large-cap stocks) to evaluate how turnover rate would affect the profitability. That assists the practitioners in real-time market trading.

4.3 Data description

The data approach used in this chapter mainly follows the one in Chapter 3. The entire data length covers from January 1957 to December 2022. The separation of the training period, validation period and testing period is listed in Table 4.1. I use the rolling window method instead of the prevailing extending window method in previous literature, such as Gu et al. (2020), and this is different from Chapter 2. The validation window size is 30%

of the training size. The 182 factors provided by Andrew Y.; Zimmermann (2020), which excludes the factor with missing values larger than 40% for maintaining a reasonable data quality. Although the MLP autoencoder pre-training approach can promote data quality, it is crucial to exclude series containing an inordinate number of missing values to reduce the bias or the potential ‘hallucination’ of LLMs.

The number of latent-layer neurons in the feedforward autoencoder in the main body of the Transformer model is configured to 70% of the input dimension to balance better representation of the input factors and prevent latent-layer overfitting. After examining the training window of 102, 204, 306 and 408 with the validation window of 30, 60, 90 and 120, respectively, the window size of training is configured to 102 (validation size 30). This aligns with the relatively short sight of standard Transformer model design. In other words, the standard Transformer model is originally designed for short-context prediction, for example, one phrase to predict another phrase, or one sentence to forecast another sentence. Parameters are re-estimated every 12 months (step size 12) for the balance of computational efficiency, training stability and forecasting accuracy. Therefore, the ‘in-sample total’ observation number in Table 4.1 includes 102 training, 30 validation and 45 iterations with 12 observations as step size moving forward ($102 + 30 + 45 \times 12 = 672$).

Name	Start date	End date	Observation No.
In-sample total	1/1957	12/2012	672
Testing (OOS) for ‘1911’	1/2013	11/2019	83
Testing (OOS) for ‘2112’	1/2013	12/2021	108
Testing (OOS) for ‘2212’	1/2013	12/2022	120

Table 4.1: Data splits for in-sample and out-of-sample data. ‘1911’ means the testing period ends before the pandemic happened, ‘2112’ means the testing period contains the pandemic period and ‘2212’ means the testing period contains not only the pandemic period but also one year after the pandemic.

4.4 Models

4.4.1 Overview of Transformer

The underlying structure of the Transformer is the autoencoder neural network structure, which was first proposed by Rumelhart et al. (1986a). Figure 4.1 shows the earliest version of

the Transformer from Vaswani et al. (2017). The left-side panel with ‘input’ are the encoder blocks, and the right-side panel with ‘output’ are the decoder blocks. This is a complex autoencoder structure developed from a simple autoencoder framework. N^* is denoted as the number of encoder or decoder blocks. To simplify the structure and fit it to the context of asset pricing, the N is configured to 1. For the encoder block, the standard Transformer has one multi-head attention layer assembled with a one-hidden-layer feedforward (FFN) MLP autoencoder structure. Inputs, after embedding and positional encoding processes, turn into sequences with identical dimensions of output embedding results and are fed into the encoder block as the features. These features have two forms of utilization, one goes into the add and layer normalization (Add & Norm) process through a bypass, while another form goes into the multi-head attention layer, then arrives at the Add & Norm process and is jointly calculated with the previous form. Likewise, the sum of the attention layer output and the features is either passed through the bypass and fed into the next Add & Norm process, or fed into the FFN structure after layer normalization. The output from the second Add & Norm passes through the cross-sectional attention layer, the multi-head attention layer links the encoder and decoder block, which is also known as the latent layer or latent space of an autoencoder.

In the decoder block, the original output experiences the output embedding and positional encoding, then either travels directly to the Add & Norm process or is fed through the ‘multi-head attention2’ layer and acts as the output of that layer to the Add & Norm process. Repetitively, the outcome of the first Add & Norm process either flows to the next Add & Norm process or generates the attention ‘Query’ for the cross-sectional attention layer. The ‘Key’ and ‘Value’ of the cross-attention layer (‘multi-head attention3’) are derived from the output of the connected encoder. The output of the cross-attention layer is processed directly by the Add & Norm process or passed through the final feedforward MLP autoencoder, then rejoins the ‘Add & Norm’ process. The entire procedure, after estimation and validation, yields the optimal weights for the value forecasts (linear layer) or the probability forecasts (softmax layer). In this chapter, the standard Transformer with an MLP autoencoder pre-training module (Pre-trained Transformer) and the encoder-only (with no cross-attention

and decoder block) with an MLP autoencoder pre-training module (SERT) are examined for asset pricing tasks. They are benchmarked with the layer-normalization first (LNF) transformation models and standard Transformer and encoder-only Transformer models.

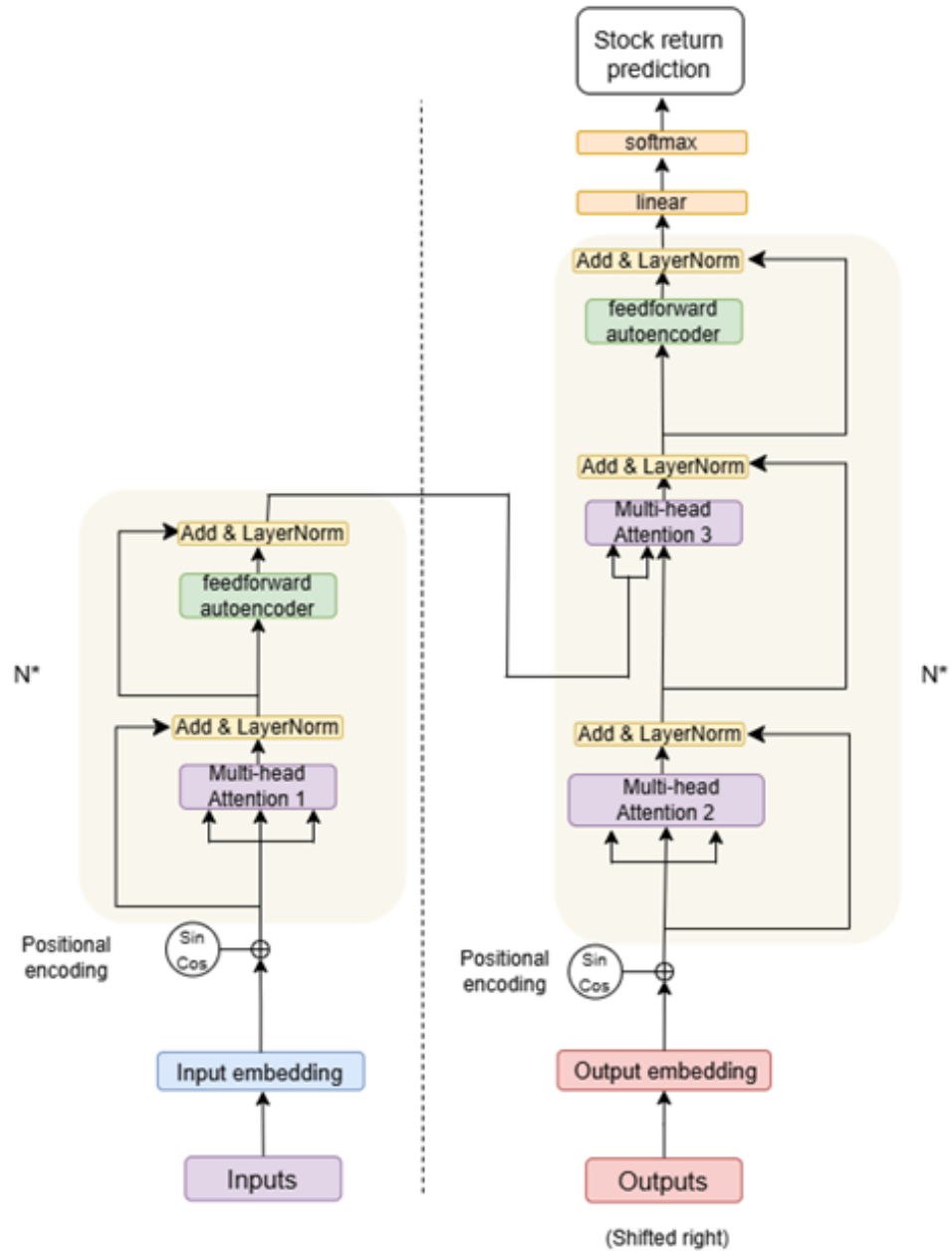


Figure 4.1: The structure of standard Transformer, source from Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. Advances in Neural Information Processing Systems, 30. The left side is the encoder, and the right side is the decoder. Both the inputs and outputs experienced the word embedding process which turned the word sequences into machine-readable vectors. $N \times$ means how many blocks are on each side. N is a hyperparameter. The structures in a rounded rectangle are defined as one block.

4.4.1.1 Positional encoding

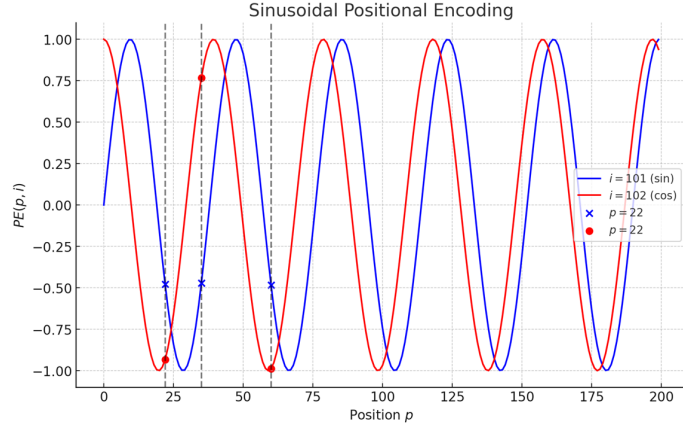
Like textual data, time series data has sequences, but unlike the recurrent neural network (RNN) models, the parallel structure improves computational efficiency, but the self-attention mechanism cannot identify the sequence unless an added function provides a guide. Also, the MLP autoencoder structure (feedforward neural network) inside the Transformer model has no mechanism designed for temporal sequence capturing. A talented positional encoding design was proposed in Transformer’s seminal paper (Vaswani et al., 2017), named Sinusoidal Positional Encoding (SPE). It can be mathematically presented as:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right) \quad (4.1)$$

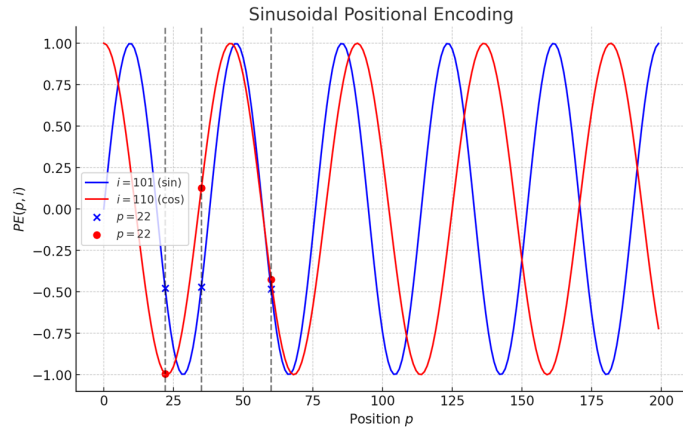
$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right) \quad (4.2)$$

Where PE is the positional encoding, pos is the position, i is the index of the dimension (features), d_{model} is the dimension of the input. These functions imply that features with odd sequential numbers are encoded with the sine function, and the ones with even sequential numbers are encoded as cosine functions. Figure 4.2 exhibits an example of how positional encoding marks the positions and notes periodic information for individual features. The sine and cosine functions originally have directionality for indicating the time steps, while the difference between the sine and cosine functions indicates the difference between two features at the identical time step. We can see the mechanism of SPE from Figure 4.2 that in the individual time step, positional codes identify different features via different wavelengths, and the distance between the round point and cross point on the identical grey dashed line demonstrates the difference between the two features. For individual features, the periodic characteristic in the sine or cosine function assists in marking the periodicity of the input features.

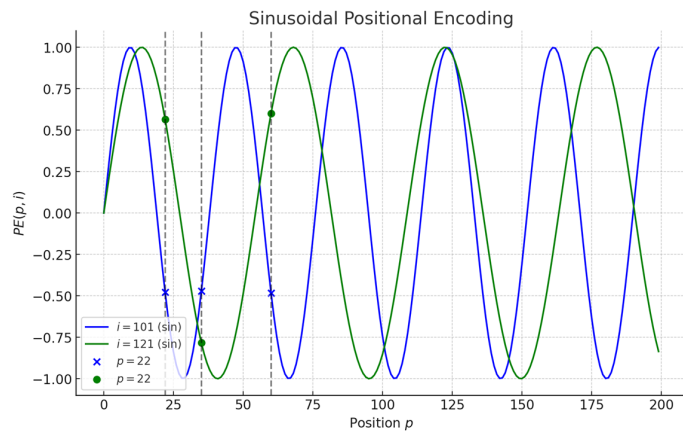
Compared with other large data applications of Transformer, parameter efficiency is a non-negligible aspect which should be considered cautiously. That is the main reason that non-parametrized SPE is preferred to Time2Vec positional encoding deployed in the works of



(a) Up



(b) Middle



(c) Bottom

Figure 4.2: Sinusoidal positional encoding plots. The X-axis shows the positions (time steps), and the Y-axis shows the positional codes. The upper panel shows the encoding values of the 101st and 102nd features; the middle panel shows the 101st and 110th features; and the bottom panel shows the 101st and 121st features. $p = 22$ means the 22nd time step.

Lezmi and Xu (2023); Ma et al. (2023); Tevare and Revankar (2023). Moreover, the summation adding positional information is superior to concatenation adding with a simplified parameter structure. In the stock pricing, if the $X' \in \mathbb{R}^{T \times d_{\text{model}}}$ is defined as the embedded factor matrix, the positional encoded input X' is equal to $X + PE$. How the positional codes work with the attention mechanism for capturing spatial-temporal variation is introduced in the appendix of Chapter 4.

The SPE fulfils all requirements for a good positional encoding solution, for example, unique for individual time steps, deterministic, consistent distance between any two time steps and better generalizability for different data lengths. It not only marks out the absolute positions for time steps but also indicates the relative positions. In addition, Vaswani et al. (2017) also mention that there is no significant difference between fixed and learnable positional encoding methods in their Transformer model.

4.4.1.2 Autoencoder

The fundamental frame of the Transformer is an autoencoder structure, which was first proposed by Rumelhart et al. (1986a). It is developed for dimensional reduction and feature learning assignments. An autoencoder has at least one encoder block and one decoder block. The left panel of Figure 4.3 shows the framework of the general autoencoder. The encoder connects the input layer, and the decoder connects the output layer. For an autoencoder, the input dimension and output dimension need not be identical if it has suitable dimension embeddings. The autoencoder mechanism is highly specialized in pivotal factor extraction. Thus, it was widely used for tasks such as data denoise, anomaly detection and prediction. Although the asset pricing data size sometimes is ‘a drop in the ocean’ compared with alternative contexts, such as video generation and speech recognition, the simplified autoencoder structure can still be fit for such tasks, which are exemplified by Gu et al. (2021).

The Transformer employs complex structures upon the autoencoder framework; the simplified Transformer model with a single encoder and single decoder has a single-layer MLP

autoencoder module linked to one multi-head attention layer, respectively. It deploys a cross-attention layer (layer in blue colour in Figure 4.3) as the latent layer connects the encoder and decoder. For stock pricing or factor investing, linear embedding is a typical substitution for the ‘Word2Vec’ embedding commonly applied in NLP tasks.

4.4.1.3 Attention mechanism

The self-attention mechanism was first proposed by Vaswani et al. (2017). It is developed from Luong et al. (2015)’s dot-product attention, which is introduced in Chapter 3. As discussed in the section on related work of this chapter, the traditional attention mechanisms rely on the output of the encoder, which means they could be highly affected by the quality of the encoder output. In addition, compressing the entire series in a fixed hidden state may cause information losses, especially for long temporal dependency data. In the self-attention mechanism, the calculation of the attention score is independent, which implies no external information, such as lagged labels (stock returns in this case), assistance. In the standard Transformer model, the self-attention layer in the encoder computes attention scores only from the input series, whereas the one in the decoder computes them only from the output series. It can lower the noise from external information. Also, the self-attention mechanism computes attention weights for all positions; it captures dependencies between any two time steps. Moreover, the self-attention in Transformer models enables parallel computing since its calculation is based on a matrix, which significantly increases computational efficiency. The right panel of Figure 4.3 provides a general picture of the attention mechanism in the autoencoder.

The cross-attention mechanism is the bridge between the encoder and decoder in the Transformer; it matches the information supported by the encoder and decoder dynamically. In other words, the joint layer of the Transformer cannot only capture the information provided by the extracted factors from the original input factors but also dynamically concentrate on the lagged stock excess returns as factors. Section 4.4 further demonstrates the mechanism of cross-attention.

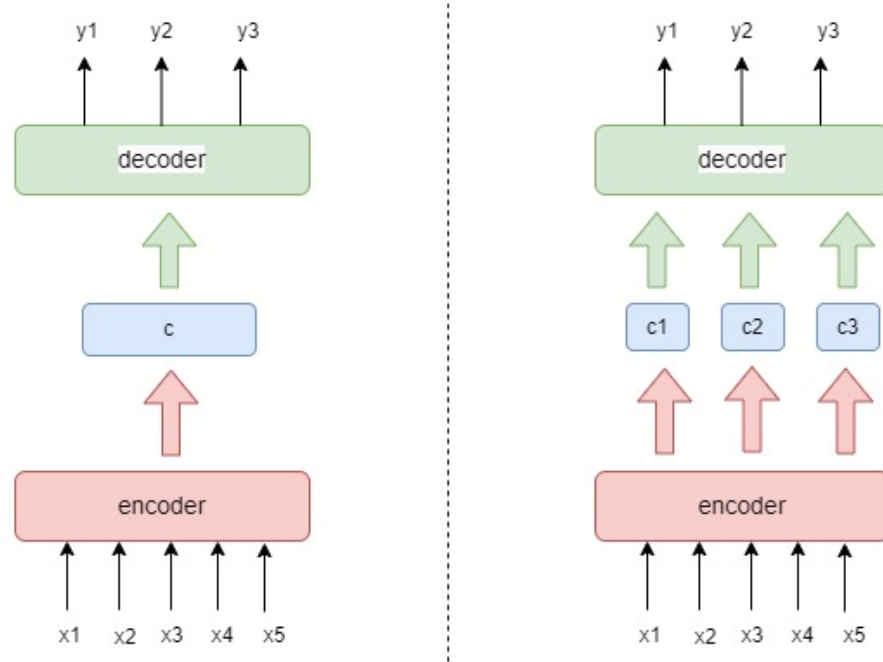


Figure 4.3: General attention mechanism workflow chart. The left panel is the general autoencoder structure, while the right panel is the autoencoder with an attention mechanism. In the left panel, c is the output vector of the encoder, which means the autoencoder without an attention mechanism distributes equal weights to the output of the encoder in each time step. In the right panel, c_1, c_2, c_3 indicate attention mechanism adds different weights on different time step t

Transformer models were originally designed for NLP tasks, it could be single-directional including generative pre-trained Transformer (GPT) or bidirectional such as BERT. The bidirectional Transformer models for NLP predictions predict the missing phrase according to the preceding and succeeding context, which implies that future data leakage may exist if the model is applied directly to time series data. Alternations for Transformer models, such as causal masks, should be taken on for a specific time series compulsorily. Section 4.4.1.3.1 also discusses how causal masks prevent future data leakage.

4.4.1.3.1 Single-head self-attention Unlike the Single-head self-attention in Chapter 3, the self-attention in the Transformer model is based on the autoencoder architecture. This means it can be computed in parallel, matrix-wise. It computes the attention score according to the entire feature group. One self-attention layer contains three input vectors. They are

Query (Q), Key (K) and Value (V). Query determines which information should be focused on. The Key is the match of Query for calculating the attention weights, while the Value represents the actual information, and it multiplies with the attention weights for the layer output. Figure 4.3 shows the workflow of single-headed attention. Concretely, the first step is generating Q,K,V via the transformation of input features X. Equation (4.3) illustrates the derivation of Q,K,V.

$$Q = W^Q X, \quad K = W^K X, \quad V = W^V X \quad (4.3)$$

Where W^Q, W^K, W^V are the weight matrices derived via estimation like MLP models. The second step is computing stabilized attention scores via dot product Q,K and dimension-wise scaling operation.

$$\text{Attention Score} = \frac{QK^\top}{\sqrt{d_{\text{model}}}} \quad (4.4)$$

For the transformer in a time series context, the third step (causal mask in Figure 4.4) is indispensable and crucial, since it is not only sequential but also temporal. Practically, due to the softmax function nature, it can be represented as '0' for visible and $-\infty$ for non-visible. In Equation (4.5), M_{ij} is the notation of the mask matrix, i is the time step that earlier than time step j initially, and it can be presented as:

$$M_{ij} = \begin{cases} 0, & \text{if } i < j \\ -\infty, & \text{if } i > j \end{cases} \quad (4.5)$$

$$M = \begin{bmatrix} 0 & \cdots & -\infty \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix} \quad (4.6)$$

Where the first column indicates the time step 1, the last column indicates the time step t . Thus, the masked attention output can be presented as:

$$\text{Masked Attention}(Q, K, V) = \text{Softmax} \left(\frac{QK^T}{\sqrt{d_{\text{model}}}} + M \right) V \quad (4.7)$$

Where the $\frac{QK^T}{\sqrt{d_{\text{model}}}} + M$ is defined as the attention score. It is weighted via the softmax function (attention weights).

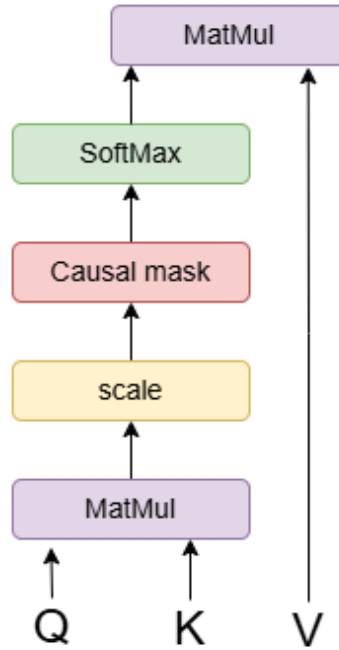


Figure 4.4: Workflow of single-head attention mechanism. ‘MatMul’ means matrix multiplication. ‘scale’ implies the division by $\sqrt{d_{\text{model}}}$.

4.4.1.3.2 Multi-head self-attention Multi-head self-attention is designed for the case where the dimension of extracted factors is large. It performs exceptionally well among the cross-sectionally oversized features in the general ML context. It separates the latent factors into subgroups and applies single-head attention within each factor group. It is an ideal approach for increasing the quality of information extraction. Concretely, for $Head_i$, the masked attention for feature group i is:

$$Head_i(Q_i, K_i, V_i) = \text{Softmax} \left(\frac{Q_i K_i^T}{\sqrt{d_k}} + M_i \right) V_i, \quad i \in \{1, 2, \dots, h\} \quad (4.8)$$

After concatenating and linearly connecting the outputs of each head, the final output of the masked multi-head attention can be computed:

$$\text{Masked MultiHead}(Q, K, V) = \text{Concat}(\text{Head}_1, \text{Head}_2, \dots, \text{Head}_h)W^O \quad (4.9)$$

Where $W^O \in \mathbb{R}^{hd_k \times d_{\text{model}}}$ is the weight matrix for linear transformation on heads outcomes. d_k is the dimension for head i , h is the number of heads, where $d_k = \frac{d_{\text{model}}}{h}$.

Compared with single-head attention, multi-head attention potentially improves the model's representability by jointly considering the relation of factors intra and inter-factor groups. It also stabilizes the model training by sharing the workload for each head. Figure 4.5 shows the workflow of multi-head attention.

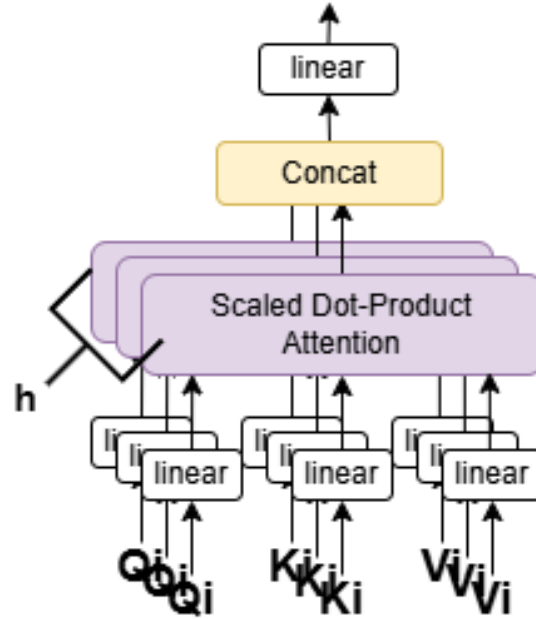


Figure 4.5: workflow of multi-head attention mechanism. Source from: Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, 30. The 'Q_i', 'K_i' and 'V_i' are the Query, Key and Value in group i , while h is the number of the heads. 'linear' is noted as the linear function.

4.4.1.3.3 Cross-attention The conjunction of encoder and decoder blocks in a standard Transformer is the cross-attention layer. Unlike the self-attention mechanism, which takes Q,K,V all from the input features, it takes Q from the decoder output and K,V from the encoder output:

$$Q_{de} = H_{de}W^Q, \quad K_{en} = H_{en}W^K, \quad V_{en} = H_{en}W^V \quad (4.10)$$

Where H_{de}, H_{en} are the outputs of the decoder and encoder, respectively. Afterwards,

$$\text{Masked Cross Attention}(Q_{de}, K_{en}, V_{en}) = \text{Softmax} \left(\frac{Q_{de}K_{en}^T}{\sqrt{d_H}} + M_{\text{cross}} \right) V_{en} \quad (4.11)$$

Where M_{cross} is the mask for the cross-attention layer, and d_H is the dimension of the hidden layer output of the encoder and decoder. Similarly, the multi-head cross attention is adding heads to handling the grouped inputs.

The feedforward MLP autoencoder in the Transformer is the same as that in the pre-training module in Chapter 3 and in this chapter. A more generalized introduction can be found in the appendix of Chapter 4.

4.4.1.4 Add and layer normalization

As introduced in the ‘overview of Transformer’ section, each outcome from the previous stage is added to the output of this stage. After layer normalization, the summation outcome is passed to the next stage. Using the ‘Add&Norm’ after single-head self-attention layer as an example, it comes:

$$X' = \text{LayerNorm} (X + \text{Masked Attention}(Q, K, V)) \quad (4.12)$$

Where X is the output of the previous layer.

4.4.1.5 Estimation

The estimation of Transformer models in this study minimises the MSE function with an L1 regularization term (to prevent latent overfitting) through the method of Stochastic Gradient Descent (SGD) with the Adaptive Moment Estimation (Adam) optimizer. The detailed derivation of estimation for the standard Transformer model is explained in the appendix of Chapter 4.

4.4.2 The proposed pre-trained Transformer

Researchers further improve the Transformer models' performance by incorporating an additional module named 'pre-training'. Clark (2020); Raffel et al. (2020), verified that pre-trained Transformers can significantly increase the predictability of NLP tasks. In this study, the pre-trained Transformer, as well as the layer normal first (LNF) pre-trained Transformer, are also examined in the context of stock pricing and factor investing. It deploys an MLP autoencoder as the pre-training module to pre-train the inputs, namely, the portfolio-sorted factors.

The advantages of the MLP autoencoder structure as a pre-training method mainly lie in denoising and pattern detection via input factor restructuring, thereby assisting the attention mechanism in the main body of the Transformer model. The proposition of structurally increasing the size of economic and financial data is supported by Didisheim et al. (2024). In the context of the factor model in this study, the noisy input factors with missing values first pass through the pre-trained autoencoder module to project the dimensionality from 182 to 420, aligning with the output dimension for the cross-attention layer. Slightly different from the MLP autoencoder in Chapter 3, the MLP autoencoder in the pre-train module is employed as the dimensional projection (extending the dimension to align with the decoder) method instead of the information extraction (dimensionality reduction). Explicitly, it can be seen as the generalized principal components analysis (PCA) process (Gu et al., 2021) or PCA without the restriction of orthogonal and linearity, which extracts the original factors with better representativity. In this case, the MLP autoencoder improves the information captured from noisy original inputs and enlarges the limited data size to some

extent via the high dimensionality of the latent space. As a byproduct, it perfectly solves the missing value existing in the input factors as well as minimizes the noise of the original input.

In addition, Liu et al. (2020b) suggest altering the ‘Add & LayerNorm’ process into layer normalization first (LNF), then ‘Add’ could stabilize the training process, moderate the ‘gradient vanish’ and ‘explosion’ issues and improve the training speed. Mathematically, it can alter the Equation (4.13) into:

$$X' = X + \text{sublayer}(\text{LayerNorm}(X)) \quad (4.13)$$

Where $\text{sublayer}(\cdot)$ is the function of output from the previous layer. Figure 4.7 shows the proposed model of the LNF version.

4.4.3 Single directional Encoder Representations from Transformer (SERT)

Kenton and Toutanova (2019) propose a simplified Transformer structure for NLP tasks based on the standard Transformer. It is named Bidirectional Encoder Representations from Transformer (BERT). Compared with the standard Transformer, BERT removes the decoder structure, which significantly reduces the number of parameters; hence, the design of BERT is more suitable for data with smaller sizes. It moderates the overfitting issue and improves computational efficiency with less sacrifice of model predictability. However, since BERT is designed for language processing, its random masking pre-training process and bidirectional mechanism make it inapplicable for single-directional numerical time series prediction. Therefore, this study also proposes an alternative model named single-directional encoder representations from Transformer (SERT) for the time series in the context of stock pricing. The SERT changes the bidirectional mechanism into a single-directional mechanism via the time series causal mask. Also, it alters the random masking pre-training process for textual prediction to the MLP autoencoder pre-training process for adapting numerical time series data with missing values and noise. In that sense, SERT also belongs to the encoder-only Transformer model family. The left panel of Figure 4.8 presents the structure of the proposed SERT model. As no decoder block is present, the encoder-only Transformer models

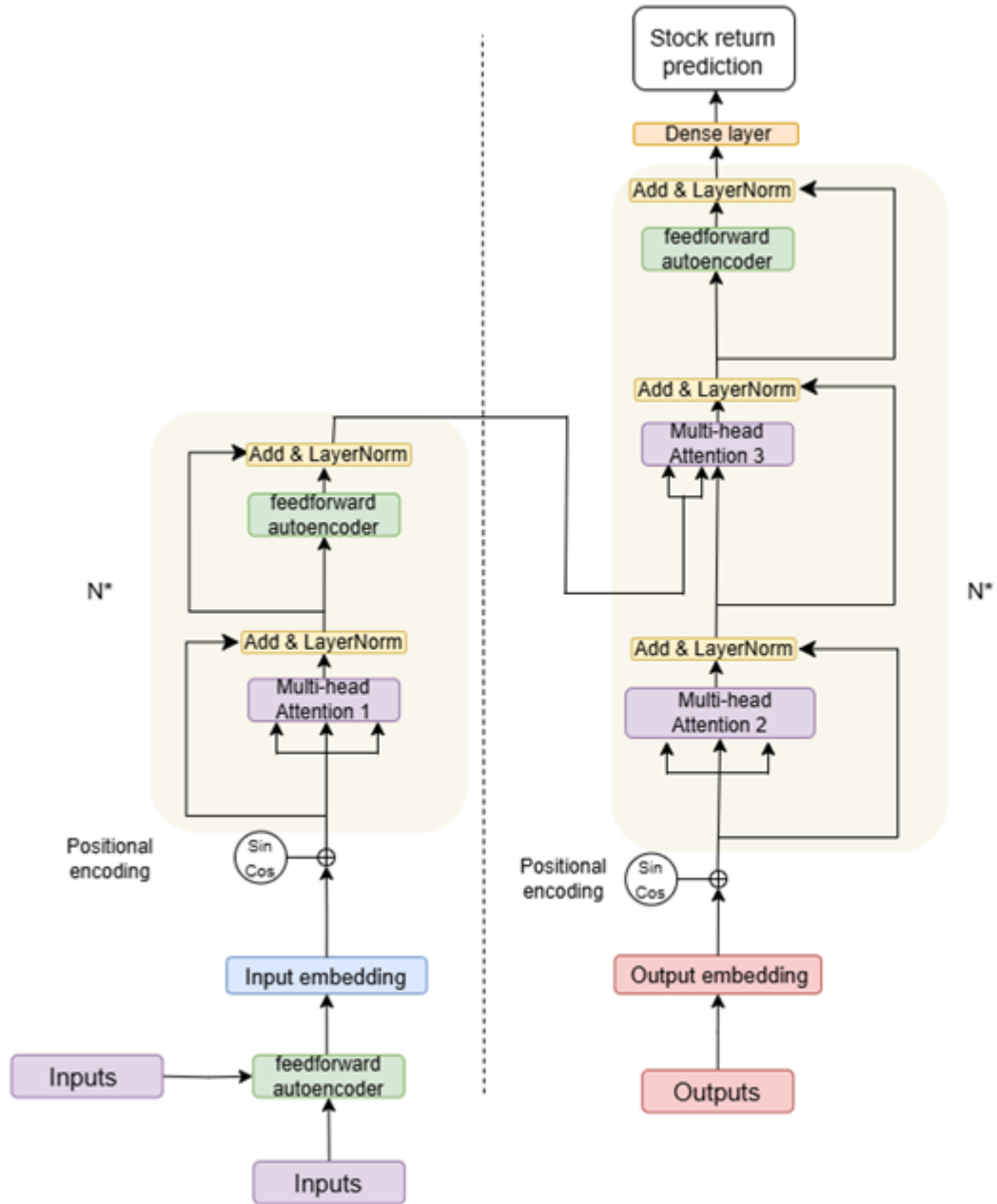


Figure 4.6: The structure of the proposed pre-trained Transformer. In the left panel, the first feedforward autoencoder is the pre-trained MLP autoencoder for enlarging the input dimension. And the input embedding is the linear embedding for numerical data. The rest of the two feedforward autoencoder modules are normal MLP autoencoders with one input and one output layer. ‘Multi-head attention 1’ and ‘Multi-head attention 2’ are self-attention modules, while ‘Multi-head attention 3’ is a cross-attention module which connects the encoder and decoder. N is the number of encoder or decoder blocks. And N^* is the number of encoder or decoder blocks.

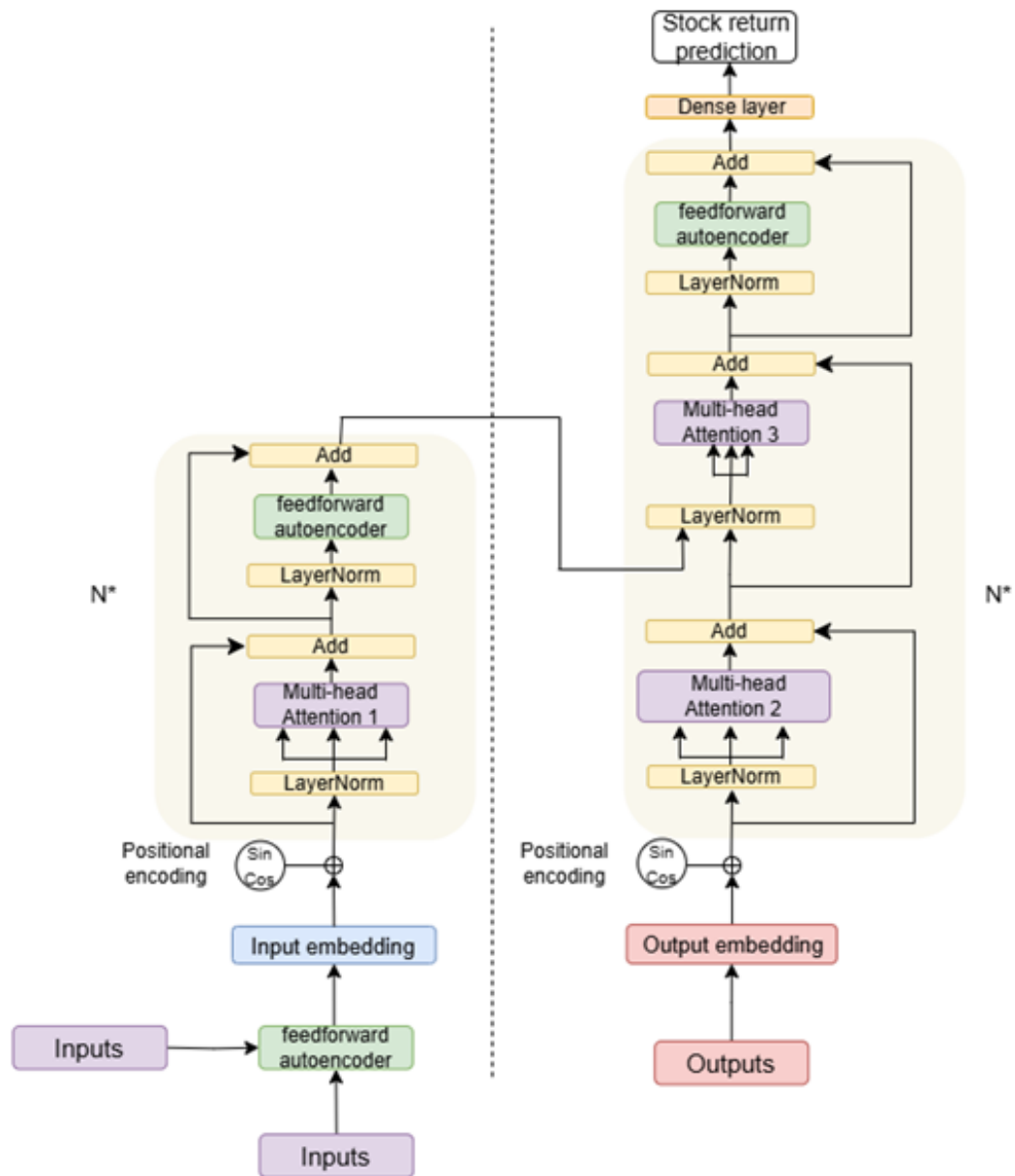


Figure 4.7: The structure of the proposed pre-trained Transformer with layer normalization first.

remove the cross-attention layer and the entire decoder block, which contains a feedforward MLP autoencoder module and a self-attention layer. Instead, it connects the feedforward MLP autoencoder module of the Transformer's encoder block linearly to the output layer. After estimation and optimization, the optimal weights are derived for stock return prediction.

Analogous to the SERT model, standard encoder-only Transformer models are widely used in different contexts. Cong et al. (2021b) employed the model as a representation of the Transformer models for comparative study on NN models in the asset pricing context, but it does not have a significant out-of-sample fitness. In this study, their model is also set as a benchmark to see whether the SERT performs better than the encoder-only Transformer model in the stock pricing or factor investing context. The right panel of Figure 4.8 shows the benchmark model employed by Cong et al. (2021b).

4.4.4 Transformers in stock pricing and factor investing

In the context of stock pricing factor models, the original input is the sorted portfolio factors (Andrew Y.; Zimmermann, 2020), while the original output is the large capital U.S. stock excess returns. Unlike the work in (Gezici and Sefer, 2024a; Päivärinta, 2022; Zhang, 2022), they deploy the entire Transformer model including the softmax layer for predicting the probability of the trading signals, this study modifies the Transformer models with a linear dense layer for predicting the out-of-sample (OOS) returns directly and using the softmax function as the trading signal filter for a comparative analysis to the sign signal generation (trend following) method. Ten Transformer models and ten encoder-only Transformer models (including SERT) are examined. They are separated into three groups: the LNF group (Group 1), the proposed pre-trained group (Group 2) and the standard group (Group 3), which are shown in Table 4.2. The difference between the models within the group is the number of heads.

The Transformer models were implemented via Python (v3.8) using the PyTorch (Paszke

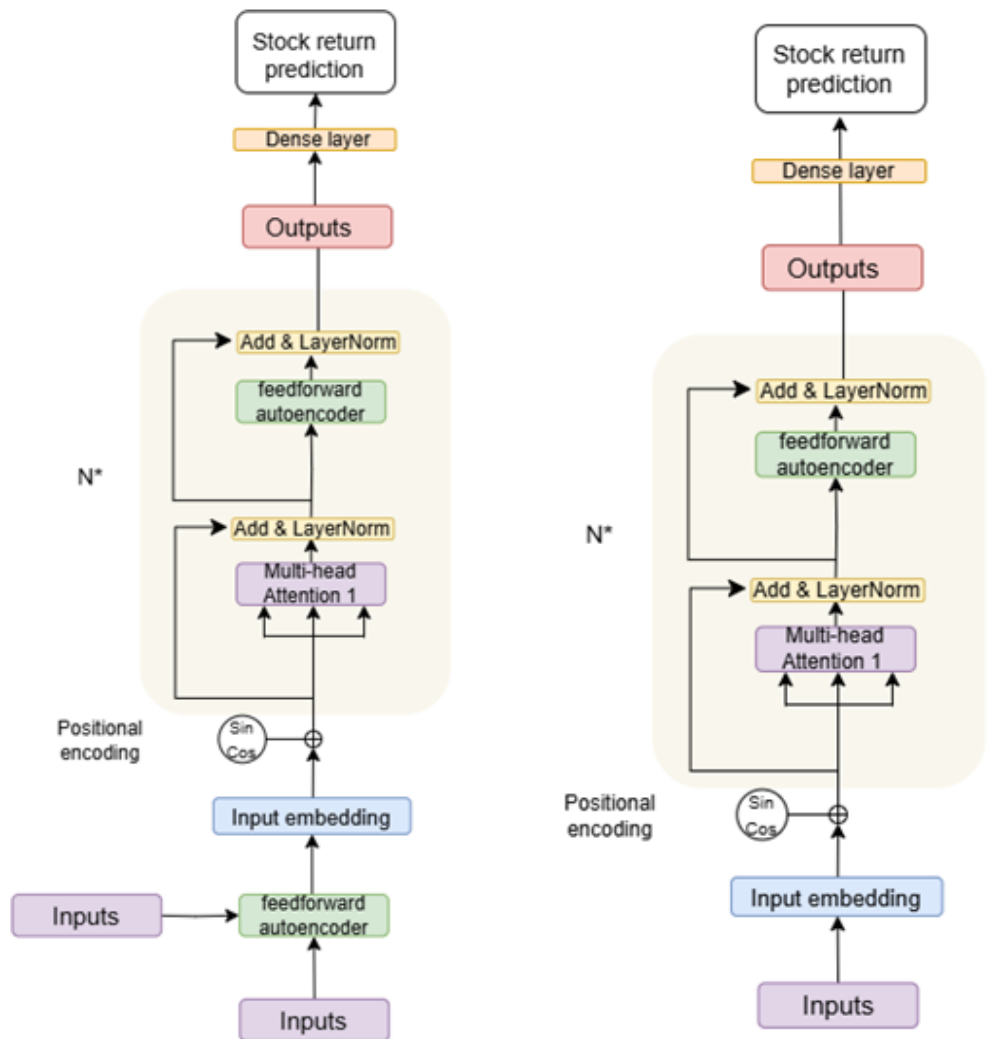


Figure 4.8: single directional encoder representations from Transformer (SERT) and benchmark model from Cong et al. (2021b). The left panel presents the structure of the proposed SERT model, and the right panel presents the standard encoder-only Transformer model in some FinTech and financial economics literature.

et al., 2019) package of ‘nn.Transformer’. Training was conducted on the NVIDIA GeForce RTX 3070 GPU. Due to the parallel computation method, it is possible to apply it on a laptop without a GPU if the data size is limited.

4.5 Empirical results

To study how the attention head affects the models’ performance in stock pricing, in this research, the standard Transformer-based models and encoder-only Transformer-based models are separated into 3 groups. ‘P_Trans_LNF’ and ‘SERT_LNF’ are denoted for proposed pre-trained models with LNF. ‘P_Trans_H1’ to ‘P_Trans_H7’ and ‘SERT_H1’ to ‘SERT_H7’ as the proposed pre-trained models with different numbers of heads, while the ‘C_Trans_H1’ to ‘C_Trans_H4’ and ‘En_Trans_H1’ to ‘En_Trans_H4’ are the benchmark models without pre-training modules with heads. The module configurations are listed in Table 4.2. The

Model	Heads	Group No.	LNF	Model	Heads	Group No.	LNF
SERT_LNF	1	1	Y	P_Trans_LNF	1	1	Y
SERT_H1	1	2	N	P_Trans_H1	1	2	N
SERT_H2	2	2	N	P_Trans_H2	2	2	N
SERT_H3	3	2	N	P_Trans_H3	3	2	N
SERT_H4	4	2	N	P_Trans_H4	4	2	N
SERT_H6	6	2	N	P_Trans_H6	6	2	N
SERT_H7	7	2	N	P_Trans_H7	7	2	N
En_Trans_H1	1	3	N	C_Trans_H1	1	3	N
En_Trans_H2	2	3	N	C_Trans_H2	2	3	N
En_Trans_H4	4	3	N	C_Trans_H4	4	3	N

Table 4.2: Model configurations. The ‘Heads’ shows the head number of the attention layers, and the ‘Group No.’ is the notation to classify model groups. The ‘LNF’ indicates the option of ‘layer normalization first’ where ‘Y’ means yes, and ‘N’ means no. ‘En’ indicates encoder-only, and ‘H’ indicates head followed by the head number. ‘C’ means classic Transformer models. All models in Group 3 are the benchmark models without the MLP pre-training process.

model performance and backtesting performance evaluation follow the methods of Chapter 2. Due to the projected large-dimension latent factors (from the pre-training process) being diluted from the original factors, the correlation between the latent factors and original factors is insignificant to evaluate the impact of the factors. Therefore, evaluating variable importance (factor impact) is of less significance in this chapter.

4.5.1 Model performance of Transformer

Table 4.3 shows the model performance of the proposed Transformer models and their benchmarks which encompass the three periods denoted as ‘1911’, ‘2112’ and ‘2212’ respectively. From the average OOS R^2 and MSE, the proposed Transformer models with different heads (P_Trans_H1 to P_Trans_H7) outperform the benchmark models (P_Trans_LNF and C_Trans_H1 to C_Trans_H4) in all periods. The advantages of the proposed models are significant, particularly during the extreme market fluctuations (sharp uptrend in ‘2112’ and sharp up-down fluctuation in ‘2212’). This means the proposed models have marked advantages in tolerating extreme market conditions. However, within each group, there are fewer differences shown between average OOS fitness indicators. This means that adding heads to attention layers helps but does not improve models’ fitness significantly. These indicators agree that Model P_Trans_H3, the proposed pre-trained Transformer with 3 attention heads, is the best-performing model in all periods. Interestingly, across all periods and within each group, the average OOS R^2 increases with head numbers, but it peaks and then declines. For example, in the proposed pre-trained Transformer group, it peaks at 3 heads and in the standard Transformer group, it peaks at 2 heads. This implies that there exists an optimal attention head number for Transformer models in a certain case.

From the factor investing perspective, all models in different periods show significant and positive mean α values. The average positive residual α values provide strong evidence of market inefficiencies and excess predictability across the three periods. These anomalies suggest the existence of the traditional α and the uncaptured non-linear risk premiums, which is where factor investors can potentially benefit from.

The three-stage Diebold-Mariano (DM) test with HAC estimator robust results shown in Table 4.4 illustrate the significance level that one model outperforms the alternatives. The negative value means the model named with the column label outperforms the model named with the row label, such as the ‘-0.44’ in the first column, which indicates that P_Trans_LNF

Model	Avg_R2	Avg_MSE	avg_α	Ann.α	α_t_statistic
Pre-COVID-19 Period (1911)					
P_Trans_LNF	0.0377	0.0057	0.0082	0.0981	21.2026***
P_Trans_H1	0.0658	0.0055	0.0116	0.1388	30.0030***
P_Trans_H2	0.0679	0.0055	0.0120	0.1439	30.9728***
P_Trans_H3	0.0763	0.0054	0.0094	0.1126	24.9751***
P_Trans_H4	0.0746	0.0054	0.0111	0.1336	29.2173***
P_Trans_H6	0.0737	0.0054	0.0107	0.1281	28.3103***
P_Trans_H7	0.0679	0.0055	0.0120	0.1442	31.0214***
C_Trans_H1	0.0536	0.0055	0.0118	0.1421	30.1773***
C_Trans_H2	0.0571	0.0055	0.0071	0.0851	18.2654***
C_Trans_H4	0.0325	0.0057	0.0088	0.1058	22.9111***
COVID-19-Inclusive Period (2112)					
P_Trans_LNF	0.0224	0.0077	0.0101	0.1207	31.3827***
P_Trans_H1	0.1029	0.0071	0.0113	0.1362	34.8214***
P_Trans_H2	0.1036	0.0070	0.0118	0.1420	35.9810***
P_Trans_H3	0.1113	0.0070	0.0098	0.1174	30.7129***
P_Trans_H4	0.1095	0.0070	0.0117	0.1399	35.6825***
P_Trans_H6	0.1052	0.0070	0.0106	0.1274	33.0697***
P_Trans_H7	0.1040	0.0070	0.0124	0.1487	37.1062***
C_Trans_H1	0.0273	0.0076	0.0127	0.1526	36.9573***
C_Trans_H2	0.0276	0.0077	0.0093	0.1115	29.2770***
C_Trans_H4	0.0123	0.0078	0.0105	0.1262	32.5920***
Including COVID-19 and One-Year After (2212)					
P_Trans_LNF	0.0234	0.0081	0.0088	0.1053	31.3703***
P_Trans_H1	0.0907	0.0076	0.0099	0.1187	35.9190***
P_Trans_H2	0.0911	0.0075	0.0104	0.1245	37.4614***
P_Trans_H3	0.0972	0.0075	0.0085	0.1017	30.9821***
P_Trans_H4	0.0960	0.0075	0.0101	0.1217	36.9343***
P_Trans_H6	0.0928	0.0075	0.0092	0.1108	33.7605***
P_Trans_H7	0.0917	0.0075	0.0108	0.1299	38.6464***
C_Trans_H1	0.0267	0.0081	0.0111	0.1332	38.6795***
C_Trans_H2	0.0266	0.0081	0.0081	0.0967	29.2690***
C_Trans_H4	0.0144	0.0082	0.0091	0.1095	33.4911***

Table 4.3: Model performance indicators of Transformers. Model P_Trans_LNF is the proposed model with layer normalization first. P_Trans_H1 to P_Trans_H7 are proposed pre-trained Transformer models and C_Trans_H1 to C_Trans_H4 are the standard Transformer models as benchmarks. In the ‘alpha_t_statistic’ column, values with three stars mean it is significant at the 99% level, two stars mean 95% significance level, and one star means 90% significance level. The notation of ‘Ann’ means annualized.

outperforms C_Trans_H4. From the panel of ‘1911’, the difference between models is insignificant. However, during the extreme market fluctuations, models in different groups perform significantly differently from each other, although models still perform insignificantly within groups. While the proposed pre-trained Transformer models demonstrate substantial improvements in overall predictive accuracy in Table 4.5, the Diebold-Mariano (DM) test with HAC standard errors suggests that the superiority is not strictly uniformly distributed across the testing period. The Diebold-Mariano (DM) test, which is based on Mean Absolute Error (MAE), yields limited statistical significance. This divergence highlights the varying sensitivities of evaluation metrics to extreme market fluctuations. Specifically, MSE disproportionately penalizes large errors, revealing that pre-trained Transformer models’ primary advantage lies in successfully capturing massive structural breaks during the highly volatile COVID-19 periods. In contrast, the MAE-based DM test is robust to outliers and primarily reflects median daily performance, where the marginal absolute differences between models are less pronounced. Furthermore, the HAC estimator strictly penalizes the strong error autocorrelation typical of volatile periods, further suppressing the test statistic. Consequently, the insignificant DM results do not imply inferior model performance; rather, they demonstrate that pre-trained Transformer models’ superiority is asymmetrically concentrated in absorbing extreme tail risks rather than yielding uniform improvements. The pre-trained Transformer models outperform benchmark models in all periods and significantly surpass the benchmarks in the extreme market conditions (‘2112’ and ‘2212’) and insignificantly in the ‘1911’. This proves that pre-trained Transformer models have a great capability to handle extreme market fluctuations.

Additionally, although Liu et al. (2020a) suggest that LNF could increase the training efficiency, stabilize the training process and improve the model performance for Transformer models, especially for deep structure, in this examination, LNF does not assist in increasing model performance in all testing periods. With the sacrifice of the FFN module dimension (20% of the input dimension) and minimization of the hyperparameters for training stability, the LNF Transformer model demonstrates suboptimal efficiency regarding the performance of standard Transformer models. In the asset pricing context, this means incomplete infor-

mation is extracted from the original input factors. In LNF cases, adding attention heads, increasing dimensions, and larger hyperparameters cause serious overfitting directly during the examinations.

Figure 4.9 exhibits the OOS R^2 distribution of the best model in each group. The distribution diagrams distinctly show that the pre-trained Transformer model with LNF significantly underperforms the best proposed Transformer model and the best standard Transformer model in all periods. The distribution charts support the conclusions drawn from other model performance indicators that the best proposed Transformer model insignificantly outperforms the best standard Transformer model in Period ‘1911’ but is noticeably superior to the benchmark models in Period ‘2112’ and Period ‘2212’ respectively. OOS MSE confirms the findings of OOS R^2 distribution chart which exhibits in Figure 4.10.

4.5.2 Model performance of SERT

Similarly, Model SERTs are examined in groups of proposed models of LNF (SERT_LNF), with different heads (SERT_H1 to SERT_H7) and benchmark standard encoder-only Transformers (En_Trans_H1 to En_Trans_H4) in three periods. Table 4.5 and Table 4.6 show the OOS model performance and DM test results, respectively. By jointly considering the OOS R^2 , MSE and DM statistics, the best model for each period is variable. In the pre-COVID period, SERT_H2 (with 2 heads) has the highest average OOS R^2 , which is 8.53%, while in ‘2112’, SERT_H7 (with 7 heads) has the highest value at 11.94%. SERT_H4 (with 4 heads) wins the model fitting competition in period ‘2212’ by a narrow margin. However, according to the DM test statistics, after controlling for the autocorrelation effect via the HAC estimator, the difference between the models in the pre-COVID period is insignificant, except for Model SERT_LNF. However, during the extreme market fluctuation periods, models in different groups perform significantly differently from each other, although models still perform insignificantly within groups. This is similar to the findings in the DM results of the standard Transformer models. The same issue of the proposed pre-trained Transformer with the LNF (P_Trans_LNF) model exists in SERT with the LNF (SERT_LNF) model and

Pre-COVID-19 Period (1911)									
	P_H1	P_H2	P_H3	P_H4	P_H6	P_H7	P_LNF	C_H1	C_H2
P_H1									
P_H2	-0.44								
P_H3	-1.37	-0.48							
P_H4	-1.44	-0.28	0.17						
P_H6	-0.66	-0.25	0.05	-0.06					
P_H7	-0.47	0.16	0.66	0.61	0.4				
P_LNF	0.83	1	1.15	1.08	1.32	0.92			
C_H1	-0.72	-1.04	-1.26	-1.06	-1.14	-0.88	0.46		
C_H2	0.13	-0.34	-0.59	-0.5	-0.65	-0.28	0.97	-0.59	
C_H4	1.12	1.34	1.46	-1.37	-1.57	-1.23	-0.51	0.95	1.41
COVID-19-Inclusive Period (2112)									
	P_H1	P_H2	P_H3	P_H4	P_H6	P_H7	P_LNF	C_H1	C_H2
P_H1									
P_H2	-0.53								
P_H3	-1.62	-0.56							
P_H4	-1.75*	-0.45	0.03						
P_H6	-0.48	-0.05	0.36	0.41					
P_H7	-0.79	-0.03	0.58	0.68	0.03				
P_LNF	1.83*	2.03**	2.11**	2.07**	2.24**	1.94*			
C_H1	-1.88*	-2.14**	-2.20**	-2.11**	-2.09**	-2.02**	0.2		
C_H2	1.65*	-1.82*	-1.94*	-1.90*	-1.95*	-1.79*	0.55	-0.41	
C_H4	2.14**	2.37**	2.40**	-2.35**	-2.45**	-2.26**	-1.05	1.05	1.33
Period Including COVID-19 and One-Year After (2212)									
	P_H1	P_H2	P_H3	P_H4	P_H6	P_H7	P_LNF	C_H1	C_H2
P_H1									
P_H2	-0.49								
P_H3	-1.56	-0.56							
P_H4	-1.68*	-0.45	0.03						
P_H6	-0.48	-0.06	0.34	0.39					
P_H7	-0.77	-0.06	0.56	0.65	0.04				
P_LNF	1.82*	2.01**	2.09**	2.05**	2.22**	1.92*			
C_H1	-1.88*	-2.13**	-2.19**	-2.10**	-2.09**	-2.02**	0.18		
C_H2	1.66*	-1.82*	-1.94*	-1.90*	-1.95*	-1.79*	0.53	-0.4	
C_H4	2.14**	2.36**	2.39**	-2.34**	-2.44**	-2.25**	-1.09	1.06	1.33

Table 4.4: DM test results of proposed pre-trained Transformer and benchmark models. Negative values demonstrate that the model with column labels as names performs worse than the one with row labels as names. 'P_H(i)' and 'C_H(i)' $i = 1, 2, \dots, 7$ is equivalent to 'P_Trans_H(i)' and 'C_Trans_H(i)' respectively, indicating the Transformer models. Significance levels: * $p < 0.1$, ** $p < 0.05$, *** $p < 0.01$.

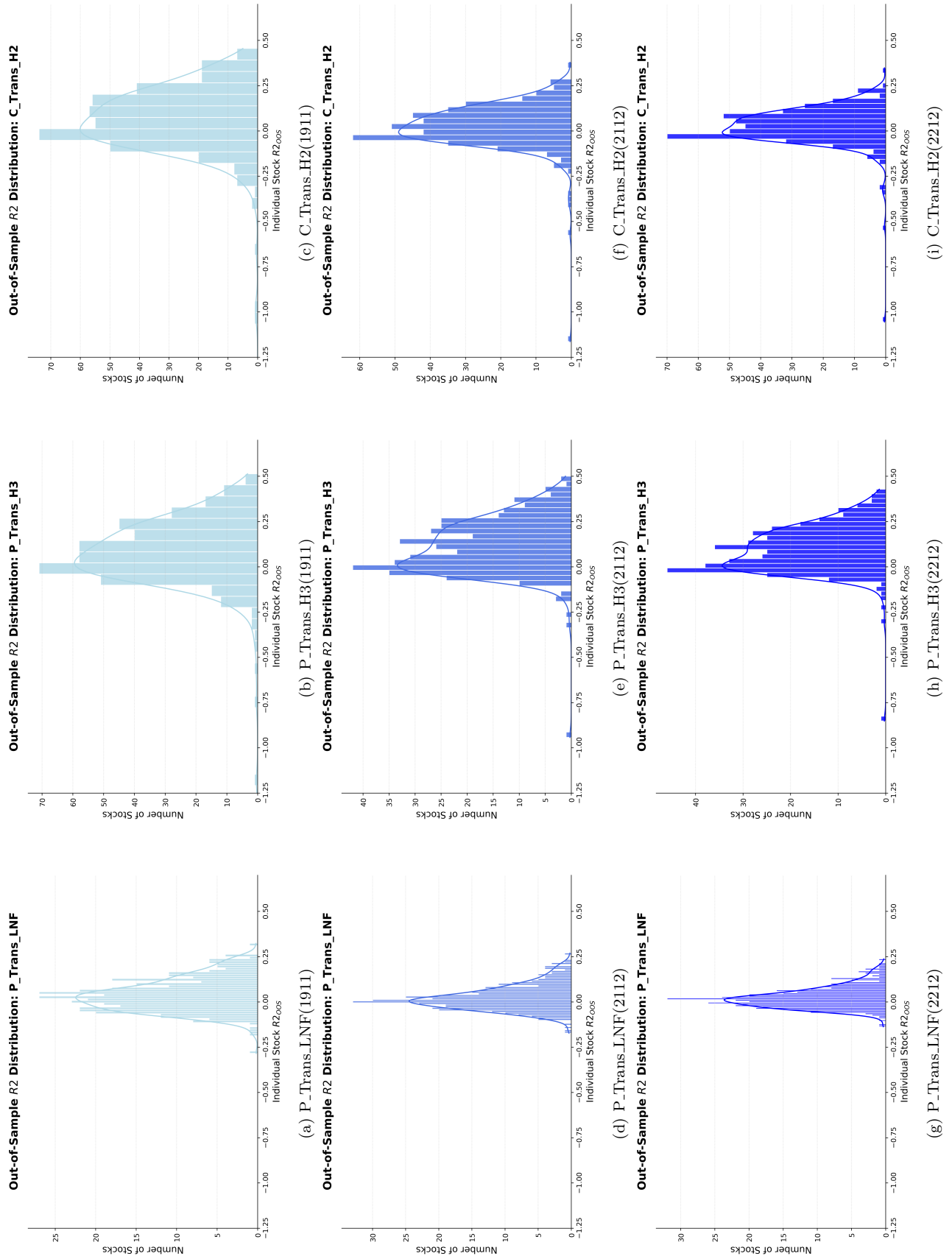


Figure 4.9: OOS R^2 distribution diagrams of best-performed Transformer model in each group for each period. The 1911, 2112 and 2212 are short for Pre-COVID-19 Period, COVID-19-Inclusive Period and One-Year After, respectively. X-axis shows the OOS R^2 value ranges in each subgraph, while the Y-axis shows the stock numbers.

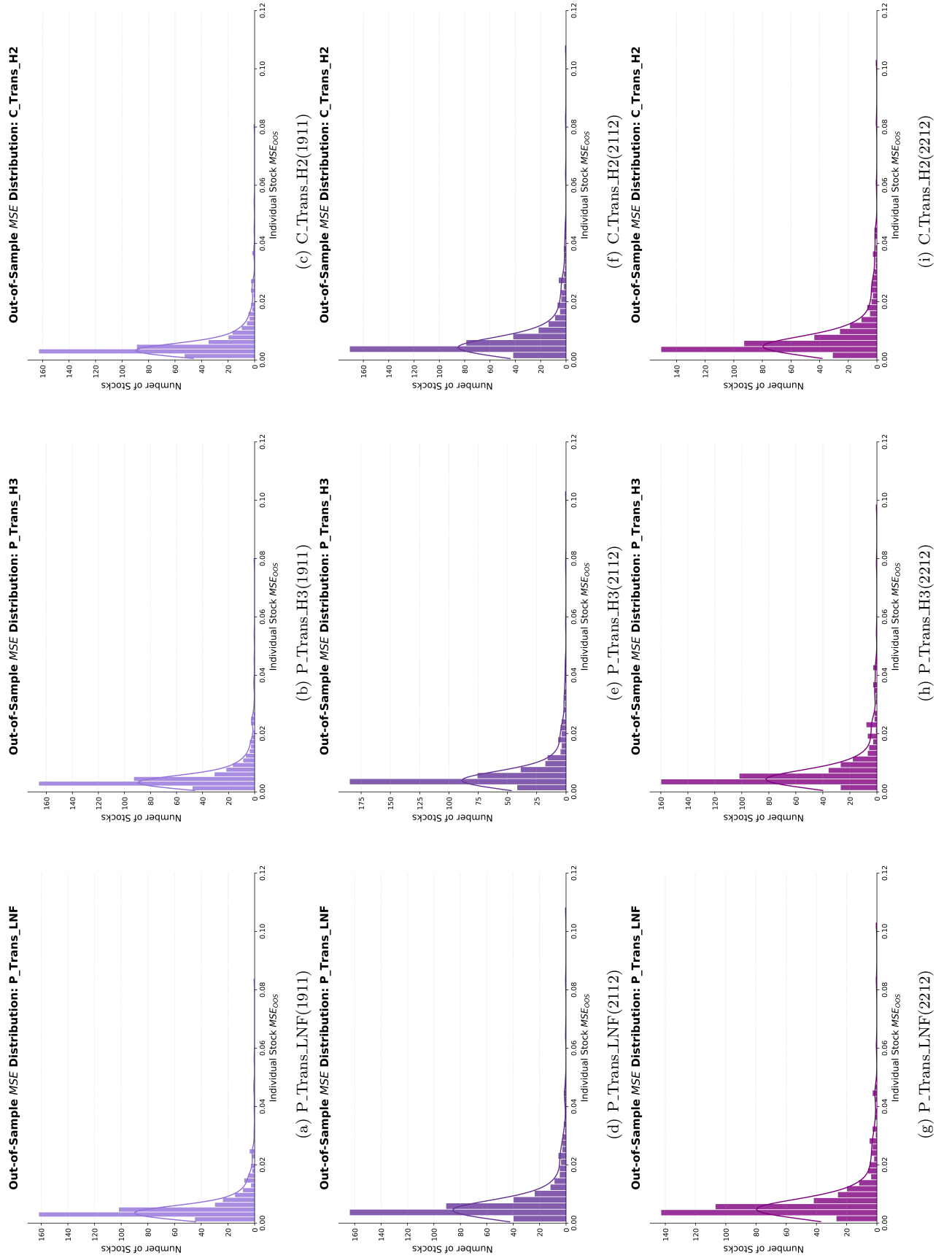


Figure 4.10: OOS MSE distribution diagrams of best-performed Transformer model in each group for each period. X-axis shows the OOS MSE value ranges in each subgraph, while the Y-axis shows the stock numbers.

has a worse impact on OOS R^2 . It fails to surpass the Transformer encoder-only models (En_Trans_H1 to En_Trans_H4).

In addition, by further investigating the OOS R^2 and DM statistics within each group, there is no obvious evidence supporting an optimal head number for SERT models in each period, as proposed by pre-trained Transformer models. This implies that the SERT models' training stability is inferior to the pre-trained Transformer models, although they reduce the parameter size for controlling the overfitting issues, whereas there is no significant difference between models with different numbers of attention heads apart from anomalies in period '2212'.

The mean α of all models in all periods is significant at a 99% significance level, which indicates Model SERTs and their encoder-only Transformer benchmarks have positive excess returns beyond factors. The conclusion drawn on mean α here still stands on those drawn on Transformer models in Section 4.5.1.

Figure 4.11 shows the OOS R^2 distribution of the best proposed SERT model and its benchmarks: LNF SERT and the best standard encoder-only Transformer. The diagrams in the pre-COVID period show no significant difference between the best models in each group, but during the high market fluctuation periods, which are '2112' and '2212', the best proposed SERT model significantly surpasses the alternatives. This agrees with the findings from other model performance indicators. Although the diagrams also present the inconsistency of the best proposed SERT model in different periods, according to DM test results, the differences between the proposed SERT models are insignificant. Thus, even if the inconsistency exists between different periods, it cannot affect the robustness of the proposed SERT models' outperformance towards benchmarks. The OOS MSE distribution diagrams support the findings from the OOS R^2 distribution diagrams, which are presented in Figure 4.12.

Compared with Transformer models, encoder-only Transformer models and SERT models have slightly higher model fitness at the expense of training stability. Similar to Transformer

Model	Avg_R2	Avg_MSE	avg_α	Ann.α	α_t_statistic
Pre-COVID-19 Period (1911)					
SERT_LNF	0.0361	0.0057	0.0138	0.1658	31.9028***
SERT_H1	0.0727	0.0055	0.0114	0.1370	29.6519***
SERT_H2	0.0853	0.0054	0.0106	0.1269	27.9494***
SERT_H3	0.0730	0.0055	0.0108	0.1297	28.4304***
SERT_H4	0.0787	0.0054	0.0085	0.1024	22.7531***
SERT_H6	0.0679	0.0055	0.0104	0.1246	27.5085***
SERT_H7	0.0805	0.0054	0.0076	0.0909	19.7284***
En_Trans_H1	0.0643	0.0055	0.0106	0.1269	27.7645***
En_Trans_H2	0.0757	0.0055	0.0077	0.0923	19.8085***
En_Trans_H4	0.0781	0.0055	0.0052	0.0629	12.9971***
COVID-19-Inclusive Period (2112)					
SERT_LNF	0.0168	0.0077	0.0147	0.1765	37.0430***
SERT_H1	0.1089	0.0070	0.0114	0.1373	34.8619***
SERT_H2	0.1066	0.0070	0.0115	0.1378	35.2856***
SERT_H3	0.1070	0.0070	0.0109	0.1303	33.6458***
SERT_H4	0.1095	0.0070	0.0091	0.1087	28.4759***
SERT_H6	0.1133	0.0070	0.0091	0.1096	28.3948***
SERT_H7	0.1194	0.0069	0.0077	0.0922	23.1957***
En_Trans_H1	0.0304	0.0077	0.0118	0.1421	35.6362***
En_Trans_H2	0.0368	0.0076	0.0097	0.1162	30.3021***
En_Trans_H4	0.0378	0.0077	0.0082	0.0987	25.5289***
Period Including COVID-19 and One-Year After (2212)					
SERT_LNF	0.0221	0.0081	0.0135	0.1618	38.4693***
SERT_H1	0.0956	0.0075	0.0100	0.1195	35.9572***
SERT_H2	0.1135	0.0074	0.0100	0.1197	36.2795***
SERT_H3	0.0939	0.0075	0.0094	0.1129	34.3404***
SERT_H4	0.1147	0.0074	0.0080	0.0959	29.1307***
SERT_H6	0.0991	0.0075	0.0079	0.0944	28.3412***
SERT_H7	0.1043	0.0075	0.0066	0.0789	22.6214***
En_Trans_H1	0.0291	0.0081	0.0103	0.1238	36.5651***
En_Trans_H2	0.0345	0.0081	0.0084	0.1009	30.6498***
En_Trans_H4	0.0356	0.0081	0.0071	0.0850	25.2777***

Table 4.5: Model performance indicators of SERTs and their benchmarks. Model SERT_LNF is the proposed model with layer normalization first. SERT_H1 to SERT_H7 are proposed SERT models, and En_Trans_H1 to En_Trans_H4 are the benchmark Transformer models. In the ‘α_t_statistic’ column, values with three stars mean it is significant at a 99% level, two stars mean a 95% significance level, and one star means a 90% significance level.

Pre-COVID-19 Period (1911)									
	SERT_H1	SERT_H2	SERT_H3	SERT_H4	SERT_H6	SERT_H7	SERT_LNF	En_H1	En_H2
SERT_H1									
SERT_H2	-0.59								
SERT_H3	0.09	0.64							
SERT_H4	-0.29	0.43	-0.37						
SERT_H6	1.29	1.05	0.87	1					
SERT_H7	-0.85	0.24	-0.96	-0.42	-1.52				
SERT_LNF	2.21**	2.32**	2.10**	2.22**	1.43	2.43**			
En_H1	-0.24	-0.55	-0.22	-0.38	0.14	-0.49	1.55		
En_H2	0.28	-0.77	-0.26	-0.42	0.14	-0.52	1.54	0.01	
En_H4	0	-0.22	-0.03	-0.12	0.38	-0.24	1.53	-0.43	-0.22

COVID-19-Inclusive Period (2112)									
	SERT_H1	SERT_H2	SERT_H3	SERT_H4	SERT_H6	SERT_H7	SERT_LNF	En_H1	En_H2
SERT_H1									
SERT_H2	-0.23								
SERT_H3	0.45	0.41							
SERT_H4	0.09	0.29	-0.21						
SERT_H6	0.05	0.22	-0.15	-0.01					
SERT_H7	-1.12	-0.43	-1.42	-1.18	-1.22				
SERT_LNF	2.97***	3.30***	2.98***	2.95***	2.87***	3.43***			
En_H1	-1.65*	-1.79*	-1.63	-1.69*	-1.61	-2.05**	1.51		
En_H2	1.74*	-2.11**	-1.73*	-1.78*	-1.71*	-2.12**	1.43	0.06	
En_H4	1.55	-0.2	1.52	-1.59	-1.52	-1.96**	1.42	-0.29	-0.2

Period Including COVID-19 and One-Year After (2212)									
	SERT_H1	SERT_H2	SERT_H3	SERT_H4	SERT_H6	SERT_H7	SERT_LNF	En_H1	En_H2
SERT_H1									
SERT_H2	-1.47								
SERT_H3	0.39	1.62							
SERT_H4	-1.61	0.29	-1.80*						
SERT_H6	0.03	1.32	-0.15	1.34					
SERT_H7	-1.15	0.83	-1.42	0.82	-1.23				
SERT_LNF	2.62***	3.82***	2.62***	3.48***	2.54**	3.05***			
En_H1	-1.65*	-2.43**	-1.64	-2.34**	-1.62	-2.05**	1.02		
En_H2	1.73*	-	-1.73*	-2.43**	-1.71*	-2.12**	0.95	0.05	
		2.79***							
En_H4	1.53	-0.21	1.51	-2.24**	-1.51	-1.95*	1.04	-0.33	-0.21

Table 4.6: DM test results of SERTs and their benchmarks. The negative values demonstrate that the model with column labels as names performs worse than the one with row labels as names. ‘En_H1’, ‘En_H2’, ‘En_H4’ are short for ‘En_Trans.H1’, ‘En_Trans.H2’, ‘En_Trans.H4’, respectively. Significance levels: * $p < 0.1$, ** $p < 0.05$, *** $p < 0.01$.

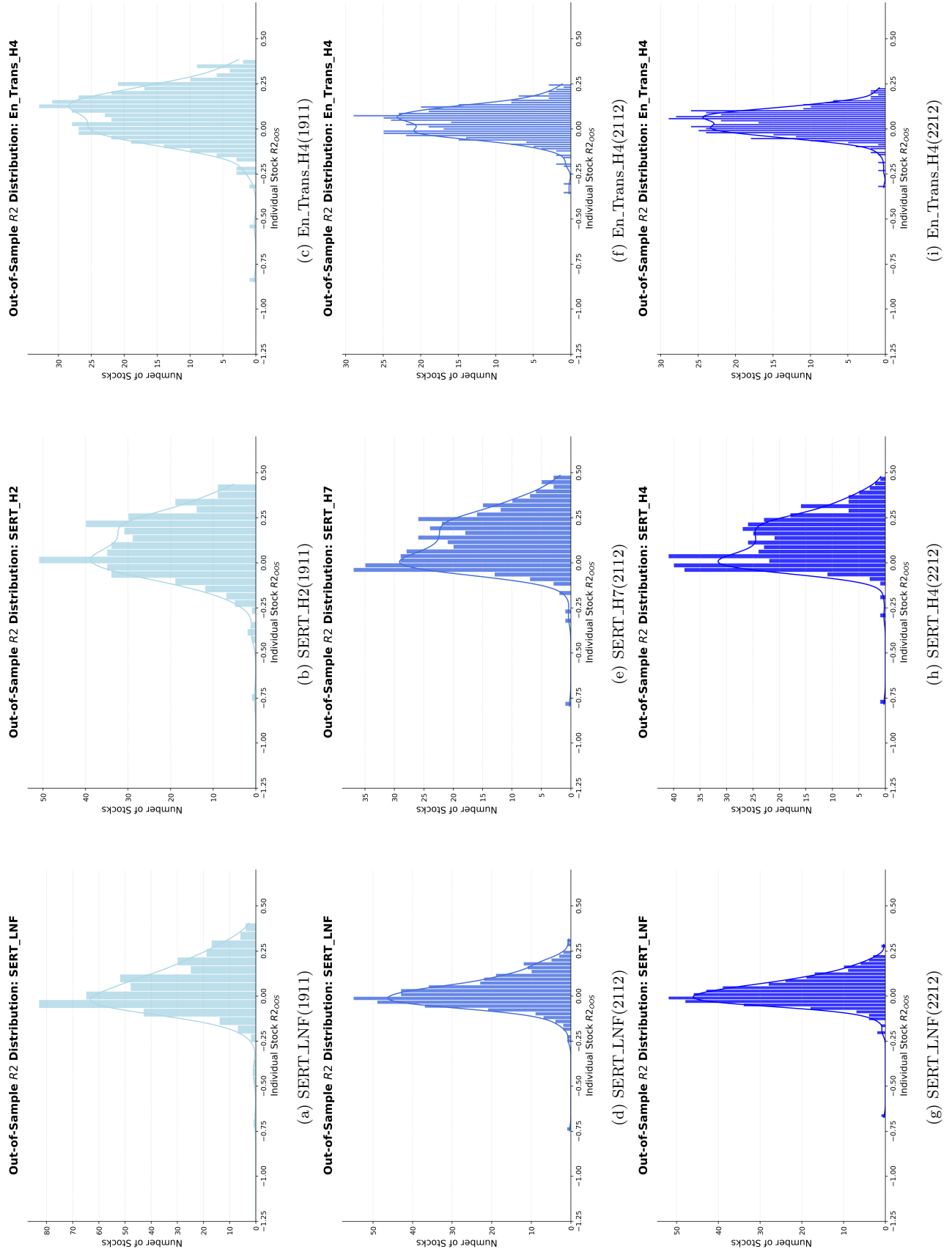


Figure 4.11: OOS R^2 distribution diagrams of best-performed SERT models and their benchmarks in each group for each period. The X-axis shows the OOS R^2 value ranges in each subgraph, while the Y-axis shows the stock numbers.

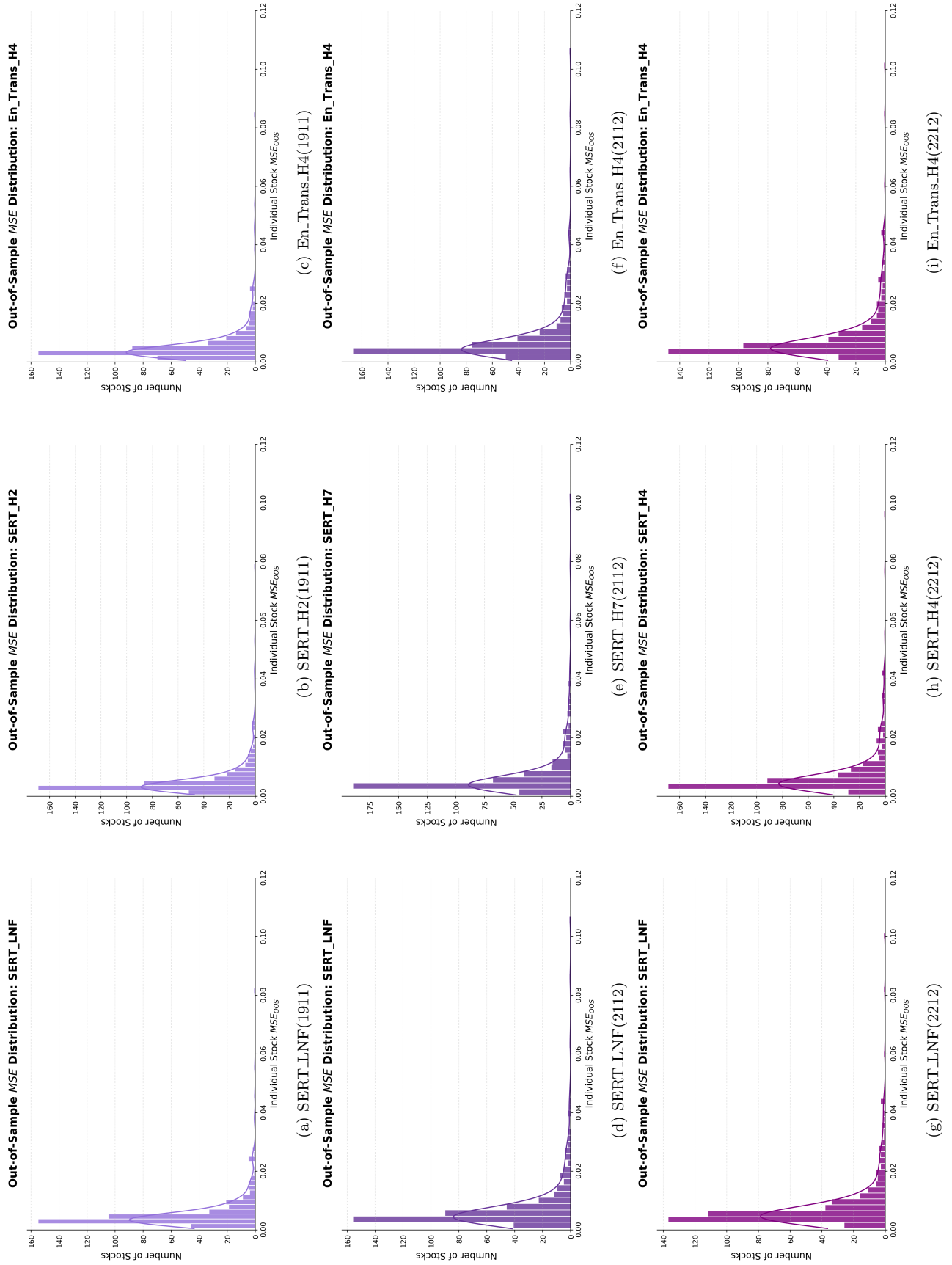


Figure 4.12: [OOS MSE distribution diagrams of best-performed SERT models and their benchmarks in each group for each period. The X-axis shows the OOS MSE value ranges in each subgroup, while the Y-axis shows the stock numbers.

models, the proposed SERT models significantly exceed the encoder-only Transformer models during extreme market fluctuation periods.

4.5.3 Backtesting performance

The Transformer models and encoder-only Transformer models are backtested with two approaches: sign signal (one of trend following trading strategies) and sign signal with softmax signal filter. They are analyzed in equal-weighted methods and value-weighted methods. Under the sign signal trading rules, the long-only strategy positions are opened at the point that both predicted return and actual return have a positive sign, which means the next value of the signal is the first value counted as the capital gain or loss. The position is closed when an identical negative sign is detected in predicted and actual returns. The softmax trading signal filter utilizes the softmax function on sign signals and filters out the worst 50% of trading signals for 420 stocks each month. The best models are selected mainly based on the indicator of the Sortino ratio and annualized returns, the maximum drawdown is the reference for the extreme downside risk. Compared with the classic Sharpe ratio, the Sortino ratio is more reasonable since investors could be more sensitive to the downside risk from the investing psychology perspective. And the annualized returns are the absolute yearly capital gain of a strategy. Table 4.7 exhibits the performance of equal-weighted (BHE) and value-weighted (BHV) buy-and-hold strategies as the most fundamental benchmarks commonly used for measuring investing performance. All models in this study are compared with these benchmarks.

4.5.3.1 Transformer

Table 4.8 presents the investing performance of equal-weighted (EW) portfolios under the static transaction cost scenario, for both sign signal and sign signal with softmax filter strategies. For the strategy without the softmax filter, none of the models has an annualized return higher than the one in the buy-and-hold strategy, and there is no significant difference between the annualized returns of each model during the Pre-COVID-19 Period

Strategy	Period	MDD	Ann. Return	Sharpe	Sortino	Ann. SR	Ann. SO
BHE	1911	-0.3701	0.1139	0.2588	0.3631	0.8965	1.2578
	2112	-0.5310	0.1349	0.2472	0.2988	0.8563	1.0351
	2212	-0.5308	0.1141	0.2148	0.2760	0.7441	0.9561
BHV	1911	-0.2963	0.1197	0.2956	0.4243	1.0240	1.4698
	2112	-0.4513	0.1337	0.2673	0.3263	0.9260	1.1303
	2212	-0.4513	0.1156	0.2311	0.3031	0.8006	1.0500

Table 4.7: The backtesting performance of buy-and-hold strategies during the examined periods considering the static transaction cost. ‘MDD’ is the maximum drawdown, ‘sharpe’ means Sharpe ratio, ‘sortino’ means Sortino ratio. BHE and BHV are denoted as buy-and-hold strategies for the equal-weighted portfolio and the value-weighted portfolio, respectively. Ann.SR and Ann.SO are the annualized Sharpe and Sortino ratios.

(1911). In addition, the Sortino ratio indicates that only Models ‘P_Trans_H2’, ‘P_Trans_H7’, and ‘C_Trans_H1’ surpass the buy-and-hold benchmark, and Model ‘P_Trans_H7’ (the proposed model with 7 heads) outperforms all other models during the pre-COVID testing period. However, the proposed pre-trained Transformer models significantly outperform the benchmark models as well as the buy-and-hold strategy (BHE) during the extreme market fluctuation (2112 and 2212) according to the Sortino ratio, which supports the findings in Section 4.5.1 that the proposed models perform better during the extreme market conditions. The ‘P_Trans_H6’ (with 6 heads) edges out Model ‘P_Trans_H7’ to become the best model in ‘2112’ and ‘2212’ in the sign signal for the EW portfolio group.

From the back testing results of the softmax filter for the EW portfolio considering the static transaction cost, the softmax filter generally increases the annualized returns for all models in all periods. With the signal filter, all models’ annualized returns perform slightly better than the buy-and-hold strategy in the testing period of ‘2112’. And most of the models’ annualized returns in the rest of the two periods excel the buy-and-hold benchmarks. Whereas it does not significantly lower the downside risks, it eliminates the advantage of the proposed models and the difference between models.

For the value-weighted (VW) portfolio considering the static transaction cost, which is shown in Figure 4.9 in the sign signal trading strategy, the annualized returns of all models are surprisingly lowered. The profitability under downside risks, which is shown in the Sortino ratio,

Sign Equal Weighted Portfolios In Static Transaction Cost						
Pre-COVID-19 Period (1911)						
Model	AR	Ann.SR	SR	Ann.SO	SO	MDD
P_Trans.LNF	0.0916	0.8421	0.2431	1.1220	0.3239	-0.2949
P_Trans.H1	0.0901	0.7985	0.2305	1.0247	0.2958	-0.3211
P_Trans.H2	0.0927	0.8726	0.2519	1.3306	0.3841	-0.3503
P_Trans.H3	0.0935	0.8293	0.2394	1.0538	0.3042	-0.3093
P_Trans.H4	0.0885	0.7777	0.2245	1.0084	0.2911	-0.3415
P_Trans.H6	0.0979	0.8851	0.2555	1.2100	0.3493	-0.3511
P_Trans.H7	0.1011	0.9131	0.2636	1.3493	0.3895	-0.3523
C_Trans.H1	0.1005	0.9166	0.2646	1.2758	0.3683	-0.2349
C_Trans.H2	0.1014	0.8691	0.2509	1.1778	0.3400	-0.2916
C_Trans.H4	0.0946	0.7850	0.2266	1.0843	0.3130	-0.4201
COVID-19-Inclusive Period (2112)						
Model	AR	Ann.SR	SR	Ann.SO	SO	MDD
P_Trans.LNF	0.1124	0.8293	0.2394	0.9789	0.2826	-0.5589
P_Trans.H1	0.1241	0.9256	0.2672	1.3216	0.3815	-0.4768
P_Trans.H2	0.1255	0.9384	0.2709	1.3645	0.3939	-0.4946
P_Trans.H3	0.1259	0.9391	0.2711	1.3101	0.3782	-0.4625
P_Trans.H4	0.1160	0.9014	0.2602	1.2495	0.3607	-0.4631
P_Trans.H6	0.1284	0.9838	0.2840	1.4341	0.4140	-0.4218
P_Trans.H7	0.1250	0.9627	0.2779	1.3735	0.3965	-0.4184
C_Trans.H1	0.1271	0.8504	0.2455	1.0212	0.2948	-0.5789
C_Trans.H2	0.1231	0.8276	0.2389	0.9727	0.2808	-0.5659
C_Trans.H4	0.1208	0.7922	0.2287	0.9644	0.2784	-0.5928
Period Including COVID-19 and One-Year After (2212)						
Model	AR	Ann.SR	SR	Ann.SO	SO	MDD
P_Trans.LNF	0.0886	0.6267	0.1809	0.7763	0.2241	-0.5589
P_Trans.H1	0.1056	0.7462	0.2154	1.0791	0.3115	-0.4768
P_Trans.H2	0.1075	0.7486	0.2161	1.0915	0.3151	-0.4946
P_Trans.H3	0.1055	0.7441	0.2148	1.0590	0.3057	-0.4625
P_Trans.H4	0.0958	0.6984	0.2016	0.9900	0.2858	-0.4631
P_Trans.H6	0.1070	0.7635	0.2204	1.1307	0.3264	-0.4218
P_Trans.H7	0.1074	0.7656	0.2210	1.1141	0.3216	-0.4184
C_Trans.H1	0.1066	0.6921	0.1998	0.8823	0.2547	-0.5789
C_Trans.H2	0.1047	0.6790	0.1960	0.8421	0.2431	-0.5659
C_Trans.H4	0.1017	0.6523	0.1883	0.8432	0.2434	-0.5928

Table 4.8: The investing performance for the equal-weighted portfolios of Transformer models considering static transaction cost. P_Trans.LNF to C_Trans.H4 are the names of models. The ‘Sign.equal.weighted’ means Sign signal strategy for equal-weighted portfolio performance. MDD is the maximum drawdown, ‘SR’ means Sharpe ratio, ‘Ann.SR’ is the annualized Sharpe ratio, ‘SO’ means Sortino ratio and ‘Ann.SO’ is the annualized Sortino ratio. ‘1911’ means the testing period ends in November 2019, which indicates the pre-COVID testing. ‘2112’ means the testing period ends in December 2021, which indicates the testing period contains the pandemic period. ‘2212’ means the testing period ends in December 2022, which indicates the testing period contains one year post-pandemic.

of all models becomes more variable between models, but during the 2112 and 2212 period, the proposed models still significantly outperform the alternatives, and most of them are better than the ‘buy-and-hold’ strategy. The Model ‘P_Trans_H2’ shows the best performance during the large market fluctuation. The value-weighted method moderates the downside risks to a degree but removes the absolute capital gain considerably. This phenomenon does not happen in MLP pricing model studies. The possible reason for that is the Transformer models have more strength for detecting and modelling data with high fluctuations, which means higher fluctuations, and higher capital gains, but under the value-weighted condition, higher weights added on the large capital stocks with lower fluctuations derive less profit. The softmax filter still rules out the difference between models and increases the annualized returns as well as downside risks, but none of them excels in the buy-and-hold (BHV) benchmark in the high market fluctuation period. However, it significantly increases performance during the pre-COVID period, which implies that it can rule out unprofitable signals and reduce the downside risks when market fluctuation is not incredibly significant. It adjusts the weight distribution bias as well.

The robustness examination is conducted to investigate how the turnover rate would affect the profitability of these models. It applies the turnover rate considering the 20bps transaction deduction rate for large-cap stocks. Table 4.10 and Table 4.11 show the dynamic transaction cost robustness backtesting results for equal-weighted and value-weighted portfolios, respectively. Comparing the backtesting results of equal-weighted portfolios in both transaction cost scenarios, the conservative penalty (the static transaction cost of 50bps) significantly lowers the absolute profitability (AR) of the models in all periods, but the high turnover rates deteriorate the models’ profitability when considering the downside risks (SO). However, the high turnover moderates the maximum drawdown (MDD). This indicates that models with high predictive power are highly sensitive to capturing the noise. In the dynamic transaction cost scenario, none of the models’ absolute capital returns is higher than the buy-and-hold (BHE) benchmarks. Similar conclusion draws in the comparison study of value-weighted portfolios, and the static transaction cost (50bps) over-penalizes the models’ profitability as well, according to annulized returns (AR) increasing and decreasing maxi-

Sign Value-Weighted Portfolios In Static Transaction Cost						
Pre-COVID-19 Period (1911)						
Model	AR	Ann.SR	SR	Ann.SO	SO	MDD
P_Trans_LNF	0.0532	0.7981	0.2304	0.9183	0.2651	-0.3061
P_Trans_H1	0.0455	0.6267	0.1809	0.6745	0.1949	-0.3289
P_Trans_H2	0.0535	0.8359	0.2413	1.0126	0.2923	-0.3260
P_Trans_H3	0.0494	0.6765	0.1953	0.7534	0.2175	-0.3044
P_Trans_H4	0.0441	0.6322	0.1825	0.6960	0.2009	-0.3202
P_Trans_H6	0.0538	0.8224	0.2374	0.9526	0.2750	-0.2939
P_Trans_H7	0.0540	0.8525	0.2461	1.0316	0.2978	-0.3242
C_Trans_H1	0.0548	0.8872	0.2561	1.0721	0.3095	-0.2883
C_Trans_H2	0.0527	0.7171	0.2070	0.8428	0.2433	-0.3649
C_Trans_H4	0.0566	0.8179	0.2361	1.0234	0.2954	-0.3061
COVID-19-Inclusive Period (2112)						
Model	AR	Ann.SR	SR	Ann.SO	SO	MDD
P_Trans_LNF	0.0648	0.8768	0.2531	1.1012	0.3179	-0.3415
P_Trans_H1	0.0691	0.8757	0.2528	1.1251	0.3248	-0.3473
P_Trans_H2	0.0801	1.0785	0.3113	1.5935	0.4600	-0.3260
P_Trans_H3	0.0748	0.9444	0.2726	1.2223	0.3529	-0.3044
P_Trans_H4	0.0617	0.8369	0.2415	1.0475	0.3026	-0.3202
P_Trans_H6	0.0785	1.0538	0.3042	1.4805	0.4274	-0.2939
P_Trans_H7	0.0709	1.0205	0.2946	1.3926	0.4020	-0.3242
C_Trans_H1	0.0744	0.8636	0.2493	1.0945	0.3160	-0.4398
C_Trans_H2	0.0658	0.7825	0.2259	0.9422	0.2720	-0.4353
C_Trans_H4	0.0745	0.8693	0.2509	1.1120	0.3210	-0.4205
Period Including COVID-19 and One-Year After (2212)						
Model	AR	Ann.SR	SR	Ann.SO	SO	MDD
P_Trans_LNF	0.0495	0.6357	0.1835	0.8268	0.2387	-0.3415
P_Trans_H1	0.0584	0.6700	0.1934	0.8715	0.2516	-0.3473
P_Trans_H2	0.0672	0.8446	0.2436	1.2308	0.3553	-0.3260
P_Trans_H3	0.0629	0.7558	0.2180	1.0022	0.2893	-0.3044
P_Trans_H4	0.0509	0.6284	0.1814	0.8089	0.2335	-0.3202
P_Trans_H6	0.0683	0.8168	0.2358	1.1411	0.3294	-0.2939
P_Trans_H7	0.0611	0.8106	0.2340	1.1179	0.3227	-0.3242
C_Trans_H1	0.0634	0.7095	0.2048	0.9412	0.2717	-0.4398
C_Trans_H2	0.0556	0.6422	0.1855	0.8186	0.2363	-0.4353
C_Trans_H4	0.0653	0.7191	0.2076	0.9692	0.2797	-0.4205

Table 4.9: The investing performance for the value-weighted portfolios of Transformer models considering the static transaction cost. P_Trans.LNF to C_Trans.H4 are model names. MDD is maximum drawdown, SR (Sharpe) means Sharpe ratio, Ann.SR is the annualized Sharpe ratio, SO (Sortino) means Sortino ratio and Ann.SO is the annualized Sortino ratio. Negative values are denoted by a minus sign.

mum drawdown (MDD) in dynamic transaction case.

4.5.3.2 Encoder-only Transformer and SERT models

Some findings from Transformer models agree with the findings of the proposed SERT models and encoder-only Transformer benchmarks in the static transaction cost scenario. The proposed SERT models show the absolute advantage according to the Sharpe ratio and Sortino ratio during the high fluctuation periods in the equal-weighted portfolios with a non-filter trading signal system. Model SERT_H4 is the best-performing model during the 2112 and 2212 periods, which outperforms the BHE benchmark in annualized return, Sharpe ratio and Sortino ratio during the pandemic. Furthermore, during the pandemic period of 2112, all proposed SERT models outperformed the encoder-only Transformer models and benchmark models concerning the Sharpe ratio and Sortino ratio. However, all models in the 2212 period have no better performance than BHE in the equal-weighted portfolio without a signal filter. The softmax filter still removes the difference between models, but it makes all models slightly exceed the BHE benchmark performance in the first two periods in most cases. In period 2212, it increases the annualized returns for most models, but not the profit under the risks. Table 4.12 shows the investment performance for the equal-weighted portfolios of the encoder-only Transformer models considering the static transaction cost (50bps).

For the value-weighted portfolios, the annualized returns of all models are reduced in all periods. Most SERT models show the advantage during the pandemic period according to the Sortino ratio. Model SERT_H1 outperforms all alternatives during the periods ‘2112’ and ‘2212’, achieving the highest returns under downside risk. The softmax signal filter moderates the difference between models as well. Similar to the function of the softmax filter in pre-trained Transformer models, the filter adjusts the weight distribution bias. With a softmax filter, SERT_H1 becomes the model with the highest Sortino ratio, which is better than BHV. It also makes SERT_LNF the best-performing model in Period 2112 and Period 2212 if only considering the Sortino ratio. SERT_LNF is also better than the BHV benchmark. Table 4.13 shows the investment performance for the value-weighted portfolios

Pre-COVID-19 Period (1911)							
Model	AR	Ann.SR	SR	Ann.SO	SO	MDD	Turnover
P_Trans_LNF	0.0733	0.6683	0.1929	0.8658	0.2499	-0.1511	0.7229
P_Trans_H1	0.0680	0.6002	0.1733	0.7497	0.2164	-0.1772	0.8748
P_Trans_H2	0.0709	0.6641	0.1917	0.9958	0.2875	-0.1481	0.8667
P_Trans_H3	0.0718	0.6344	0.1831	0.7810	0.2255	-0.1730	0.8619
P_Trans_H4	0.0653	0.5726	0.1653	0.7190	0.2076	-0.1773	0.9185
P_Trans_H6	0.0787	0.7087	0.2046	0.9390	0.2711	-0.1394	0.7589
P_Trans_H7	0.0784	0.7053	0.2036	1.0189	0.2941	-0.1368	0.8968
C_Trans_H1	0.0837	0.7586	0.2190	1.0321	0.2979	-0.1297	0.6654
C_Trans_H2	0.0828	0.7064	0.2039	0.9383	0.2709	-0.1472	0.7340
C_Trans_H4	0.0826	0.6820	0.1969	0.9079	0.2621	-0.1463	0.4798
BHE	0.1141	0.8892	0.2567	1.2303	0.3552	-0.1483	0.0120
COVID-19-Inclusive Period (2112)							
P_Trans_LNF	0.0944	0.6968	0.2011	0.8052	0.2324	-0.2956	0.6980
P_Trans_H1	0.1037	0.7726	0.2230	1.0770	0.3109	-0.2778	0.7861
P_Trans_H2	0.1053	0.7862	0.2270	1.1211	0.3236	-0.2823	0.7829
P_Trans_H3	0.1059	0.7891	0.2278	1.0688	0.3085	-0.2731	0.7729
P_Trans_H4	0.0948	0.7358	0.2124	0.9890	0.2855	-0.2674	0.8229
P_Trans_H6	0.1107	0.8454	0.2441	1.1946	0.3449	-0.2428	0.6875
P_Trans_H7	0.1041	0.8007	0.2311	1.1174	0.3226	-0.2548	0.8051
C_Trans_H1	0.1123	0.7503	0.2166	0.8926	0.2577	-0.3105	0.5733
C_Trans_H2	0.1071	0.7198	0.2078	0.8385	0.2420	-0.2883	0.6220
C_Trans_H4	0.1102	0.7204	0.2080	0.8632	0.2492	-0.2904	0.4162
BHE	0.1353	0.8512	0.2457	1.0181	0.2939	-0.2890	0.0093
Period Including COVID-19 and One-Year After (2212)							
P_Trans_LNF	0.0705	0.5036	0.1454	0.6125	0.1768	-0.2956	0.7144
P_Trans_H1	0.0866	0.6146	0.1774	0.8743	0.2524	-0.2778	0.7414
P_Trans_H2	0.0884	0.6183	0.1785	0.8892	0.2567	-0.2823	0.7498
P_Trans_H3	0.0870	0.6160	0.1778	0.8544	0.2466	-0.2731	0.7270
P_Trans_H4	0.0762	0.5585	0.1612	0.7721	0.2229	-0.2674	0.7707
P_Trans_H6	0.0906	0.6479	0.1870	0.9366	0.2704	-0.2428	0.6434
P_Trans_H7	0.0879	0.6288	0.1815	0.9015	0.2602	-0.2548	0.7634
C_Trans_H1	0.0927	0.6036	0.1742	0.7622	0.2200	-0.3105	0.5460
C_Trans_H2	0.0897	0.5838	0.1685	0.7177	0.2072	-0.2883	0.5938
C_Trans_H4	0.0917	0.5880	0.1697	0.7462	0.2154	-0.2904	0.3995
BHE	0.1143	0.7017	0.2026	0.8883	0.2564	-0.2890	0.0083

Table 4.10: The backtesting results for the equal-weighted portfolios of Transformer models considering the dynamic transaction cost. 'BHE' represents the buy-and-hold benchmark strategy under equal-weighted method.

Pre-COVID-19 Period (1911)							
Model	AR	Ann.SR	SR	Ann.SO	SO	MDD	Turnover
P_Trans_LNF	0.1125	1.0716	0.3094	1.3302	0.3840	-0.1234	0.6952
P_Trans_H1	0.0935	0.8719	0.2517	1.0950	0.3161	-0.1281	0.8439
P_Trans_H2	0.0939	0.8869	0.2560	1.1551	0.3334	-0.1463	0.8166
P_Trans_H3	0.0861	0.8096	0.2337	0.9607	0.2773	-0.1353	0.8094
P_Trans_H4	0.0904	0.8425	0.2432	1.0353	0.2989	-0.1315	0.8541
P_Trans_H6	0.0998	0.9457	0.2730	1.1738	0.3388	-0.1436	0.7247
P_Trans_H7	0.0983	0.9460	0.2731	1.3033	0.3762	-0.1342	0.8495
C_Trans_H1	0.1117	1.0511	0.3034	1.3977	0.4035	-0.0961	0.6430
C_Trans_H2	0.1017	0.9289	0.2681	1.1851	0.3421	-0.1294	0.6911
C_Trans_H4	0.1256	1.0954	0.3162	1.4692	0.4241	-0.1138	0.4638
BHV	0.1388	1.1795	0.3405	1.6054	0.4634	-0.1297	0.0120
COVID-19-Inclusive Period (2112)							
P_Trans_LNF	0.1315	1.0401	0.3002	1.2939	0.3735	-0.1959	0.6775
P_Trans_H1	0.1268	0.9590	0.2768	1.3431	0.3877	-0.1784	0.7638
P_Trans_H2	0.1213	0.9453	0.2729	1.2489	0.3605	-0.1931	0.7400
P_Trans_H3	0.1214	0.9703	0.2801	1.3288	0.3836	-0.1471	0.7300
P_Trans_H4	0.1176	0.9712	0.2804	1.3255	0.3826	-0.1479	0.7819
P_Trans_H6	0.1267	1.0279	0.2967	1.4166	0.4090	-0.1583	0.6647
P_Trans_H7	0.1235	1.0156	0.2932	1.4776	0.4266	-0.1596	0.7696
C_Trans_H1	0.1348	0.9832	0.2838	1.2351	0.3566	-0.2237	0.5539
C_Trans_H2	0.1276	0.9425	0.2721	1.1958	0.3452	-0.2142	0.5896
C_Trans_H4	0.1431	1.0245	0.2957	1.2709	0.3669	-0.2279	0.4060
BHV	0.1540	1.0688	0.3085	1.3057	0.3769	-0.2352	0.0093
Period Including COVID-19 and One-Year After (2212)							
P_Trans_LNF	0.1055	0.7924	0.2287	1.0260	0.2962	-0.1959	0.6967
P_Trans_H1	0.1060	0.7585	0.2190	1.0760	0.3106	-0.1784	0.7109
P_Trans_H2	0.0998	0.7169	0.2070	0.9831	0.2838	-0.1931	0.7053
P_Trans_H3	0.0990	0.7470	0.2157	1.0397	0.3001	-0.1756	0.6827
P_Trans_H4	0.0968	0.7447	0.2150	1.0379	0.2996	-0.1661	0.7352
P_Trans_H6	0.1086	0.8012	0.2313	1.1222	0.3240	-0.1756	0.6295
P_Trans_H7	0.1050	0.7752	0.2238	1.1152	0.3219	-0.1656	0.7357
C_Trans_H1	0.1126	0.7825	0.2259	1.0495	0.3030	-0.2237	0.5278
C_Trans_H2	0.1060	0.7367	0.2127	0.9748	0.2814	-0.2142	0.5660
C_Trans_H4	0.1232	0.8376	0.2418	1.0988	0.3172	-0.2279	0.3875
BHV	0.1308	0.8654	0.2498	1.1270	0.3253	-0.2352	0.0083

Table 4.11: The backtesting results for the value-weighted portfolios of Transformer models considering the dynamic transaction cost. 'BHV' represents the buy-and-hold benchmark of the value-weighted method.

Sign Equal Weighted Portfolios In Static Transaction Cost						
Pre-COVID-19 Period (1911)						
Model	AR	Ann.SR	SR	Ann.SO	SO	MDD
SERT_LNF	0.0968	0.9083	0.2622	1.2724	0.3673	-0.3566
SERT_H1	0.0939	0.8646	0.2496	1.2284	0.3546	-0.3066
SERT_H2	0.0839	0.7597	0.2193	0.9339	0.2696	-0.3586
SERT_H3	0.0894	0.8137	0.2349	1.1255	0.3249	-0.3604
SERT_H4	0.1066	0.9481	0.2737	1.1563	0.3338	-0.2956
SERT_H6	0.0908	0.7843	0.2264	1.0403	0.3003	-0.3651
SERT_H7	0.0890	0.7857	0.2268	1.0042	0.2899	-0.3321
En_Trans_H1	0.0940	0.8775	0.2533	1.2928	0.3732	-0.2941
En_Trans_H2	0.0964	0.8324	0.2403	1.2284	0.3546	-0.3904
En_Trans_H4	0.0960	0.8380	0.2419	1.0753	0.3104	-0.3059
COVID-19-Inclusive Period (2112)						
Model	AR	Ann.SR	SR	Ann.SO	SO	MDD
SERT_LNF	0.1263	0.9083	0.2622	1.2124	0.3500	-0.5154
SERT_H1	0.1252	0.9478	0.2736	1.3773	0.3976	-0.4498
SERT_H2	0.1162	0.8889	0.2566	1.2138	0.3504	-0.5137
SERT_H3	0.1201	0.9284	0.2680	1.3801	0.3984	-0.4354
SERT_H4	0.1374	1.0351	0.2988	1.4186	0.4095	-0.4119
SERT_H6	0.1184	0.8560	0.2471	1.1331	0.3271	-0.5083
SERT_H7	0.1214	0.9055	0.2614	1.2523	0.3615	-0.4993
En_Trans_H1	0.1222	0.8411	0.2428	1.0264	0.2963	-0.6018
En_Trans_H2	0.1252	0.8189	0.2364	1.0278	0.2967	-0.6110
En_Trans_H4	0.1197	0.8151	0.2353	0.9800	0.2829	-0.5944
Period Including COVID-19 and One-Year After (2212)						
Model	AR	Ann.SR	SR	Ann.SO	SO	MDD
SERT_LNF	0.1016	0.7070	0.2041	0.9793	0.2827	-0.5154
SERT_H1	0.1053	0.7489	0.2162	1.0863	0.3136	-0.4498
SERT_H2	0.0897	0.6495	0.1875	0.8770	0.2532	-0.5137
SERT_H3	0.1006	0.7229	0.2087	1.0697	0.3088	-0.4354
SERT_H4	0.1089	0.7791	0.2249	1.0562	0.3049	-0.4119
SERT_H6	0.0968	0.6651	0.1920	0.9104	0.2628	-0.5083
SERT_H7	0.0978	0.6876	0.1985	0.9699	0.2800	-0.4993
En_Trans_H1	0.1024	0.6772	0.1955	0.8667	0.2502	-0.6018
En_Trans_H2	0.1044	0.6641	0.1917	0.8698	0.2511	-0.6110
En_Trans_H4	0.1009	0.6637	0.1916	0.8515	0.2458	-0.5944

Table 4.12: The investing performance for the equal-weighted portfolios of the encoder-only Transformer models considering the static transaction cost. SERT_LNF to SERT_H7 and En_Trans_H1 to En_Trans_H4 are the names of the models. SERT_LNF is the proposed SERT with LNF and SERT_H1 to SERT_H7 are proposed SERTs with different attention heads. En_Trans_H1 to En_Trans_H4 are standard encoder-only Transformer models with different heads as benchmarks. MDD is the maximum drawdown, ‘SR’ means Sharpe ratio, ‘Ann.SR’ is the annualized Sharpe ratio, ‘SO’ means Sortino ratio and ‘Ann.SO’ is the annualized Sortino ratio. Negative values are denoted by a minus sign.

of encoder-only Transformer models.

Sign Value-Weighted Portfolios In Static Transaction Cost						
Pre-COVID-19 Period (1911)						
Model	AR	Ann.SR	SR	Ann.SO	SO	MDD
SERT_LNF	0.0546	0.9114	0.2631	1.0108	0.2918	-0.2475
SERT_H1	0.0540	0.8123	0.2345	0.9544	0.2755	-0.2998
SERT_H2	0.0394	0.5355	0.1546	0.5473	0.1580	-0.3715
SERT_H3	0.0445	0.6564	0.1895	0.7510	0.2168	-0.4871
SERT_H4	0.0532	0.7216	0.2083	0.7427	0.2144	-0.3211
SERT_H6	0.0499	0.6810	0.1966	0.7659	0.2211	-0.3132
SERT_H7	0.0498	0.6551	0.1891	0.7084	0.2045	-0.3096
En_Trans_H1	0.0575	0.9194	0.2654	1.0538	0.3042	-0.3684
En_Trans_H2	0.0605	0.8986	0.2594	1.1217	0.3238	-0.2395
En_Trans_H4	0.0576	0.7645	0.2207	0.8168	0.2358	-0.3000
COVID-19-Inclusive Period (2112)						
Model	AR	Ann.SR	SR	Ann.SO	SO	MDD
SERT_LNF	0.0737	0.9540	0.2754	1.2502	0.3609	-0.3382
SERT_H1	0.0760	1.0070	0.2907	1.4348	0.4142	-0.2998
SERT_H2	0.0670	0.8456	0.2441	1.0240	0.2956	-0.3715
SERT_H3	0.0714	0.9339	0.2696	1.2762	0.3684	-0.4871
SERT_H4	0.0780	0.9724	0.2807	1.1854	0.3422	-0.3211
SERT_H6	0.0751	0.8979	0.2592	1.2336	0.3561	-0.3170
SERT_H7	0.0743	0.8896	0.2568	1.1528	0.3328	-0.3096
En_Trans_H1	0.0691	0.8664	0.2501	0.9526	0.2750	-0.4391
En_Trans_H2	0.0678	0.8719	0.2517	1.0902	0.3147	-0.3838
En_Trans_H4	0.0666	0.7514	0.2169	0.8622	0.2489	-0.5050
Period Including COVID-19 and One-Year After (2212)						
Model	AR	Ann.SR	SR	Ann.SO	SO	MDD
SERT_LNF	0.0528	0.6277	0.1812	0.7773	0.2244	-0.3382
SERT_H1	0.0670	0.8196	0.2366	1.1802	0.3407	-0.2998
SERT_H2	0.0419	0.4749	0.1371	0.5466	0.1578	-0.4023
SERT_H3	0.0585	0.6876	0.1985	0.9111	0.2630	-0.4871
SERT_H4	0.0519	0.5879	0.1697	0.6873	0.1984	-0.3584
SERT_H6	0.0609	0.6772	0.1955	0.9457	0.2730	-0.3170
SERT_H7	0.0636	0.7219	0.2084	0.9893	0.2856	-0.3096
En_Trans_H1	0.0566	0.6589	0.1902	0.7756	0.2239	-0.4391
En_Trans_H2	0.0551	0.6606	0.1907	0.8525	0.2461	-0.3838
En_Trans_H4	0.0537	0.5816	0.1679	0.7077	0.2043	-0.5050

Table 4.13: The investing performance for the value-weighted portfolios of encoder-only Transformer models considering the static transaction cost. MDD is maximum drawdown, ‘SR (Sharpe)’ means Sharpe ratio, ‘Ann.SR’ is the annualized Sharpe ratio, ‘SO (Sortino)’ means Sortino ratio and ‘Ann.SO’ is the annualized Sortino ratio. Negative values are denoted by a minus sign. The values in bold font indicate values that outperform the buy-and-hold benchmark.

A robust examination of the dynamic transaction cost effect is also conducted, which is presented in Table 4.14 (equal-weighted) and Table 4.15 (value-weighted), respectively. In equal-weighted portfolios, the static transaction cost does not show significant over-penalization based on the annualized returns, but the dynamic transaction cost, which counts on the turnover rate with 20bps, highly reduces the maximum drawdown, Sharpe ratio and Sortino ratio. This indicates that high turnover rates erode the profitability of these models. The encoder-only Transformer models in the equal-weighted method, high turnover rate impact higher than the heavy static transaction cost penalty. Model SERT_H4 outperforms alternatives in profitability in both transaction cost methods. In value-weighted portfolios, the static transaction cost shows significant over-penalization by penalizing the annualized returns and decreasing the maximum drawdown. However, models, which are less sensitive to forecasting signals (lower turnover rates) derives higher Sharpe ratio and Sortino ratio, especially during the high fluctuation phrase.

4.5.3.3 Comparative analysis

For comparing the Transformer models and encoder-only Transformer models, we extract the best model in each group and evaluate them in the EW and VW portfolios in the static transaction cost scenario. Table 4.16 shows the backtesting performance of the EW portfolio between the best Transformer models and the best encoder-only Transformer models in each group, while Table 4.19 shows the backtesting performance in the VW portfolio. Interestingly, from the periodical horizontal comparison, the EW portfolio shows insignificant differences between the best pre-trained Transformer model (P_Trans_H6) and the best SERT model (SERT_H4). By only considering the absolute capital gain (Annualized return), SERT performs better than the pre-trained Transformer model, but when jointly considering the downside risks (Sortino ratio), the pre-trained Transformer model performs better, especially during the extreme market periods (2112 and 2212). Moreover, LNF SERT (SERT_LNF) performs better than the LNF pre-trained Transformer (P_Trans_LNF) in all periods, and there is no significant difference between the standard encoder-only Transformer (En_Trans_H1) and the standard Transformer model (C_Trans_H1).

Pre-COVID-19 Period (1911)							
Model	AR	Ann.SR	SR	Ann.SO	SO	MDD	Turnover
SERT_LNF	0.0782	0.7278	0.2101	1.0021	0.2893	-0.1399	0.7301
SERT_H1	0.0717	0.6578	0.1899	0.9092	0.2625	-0.1625	0.8792
SERT_H2	0.0615	0.5541	0.1600	0.6645	0.1918	-0.1939	0.8899
SERT_H3	0.0667	0.6049	0.1746	0.8163	0.2357	-0.1580	0.9013
SERT_H4	0.0856	0.7588	0.2190	0.8953	0.2585	-0.1664	0.8212
SERT_H6	0.0690	0.5940	0.1715	0.7682	0.2218	-0.1812	0.8632
SERT_H7	0.0679	0.5969	0.1723	0.7463	0.2154	-0.1825	0.8372
En_Trans_H1	0.0786	0.7247	0.2092	1.0519	0.3037	-0.1483	0.6129
En_Trans_H2	0.0785	0.6765	0.1953	0.9550	0.2757	-0.1654	0.7037
En_Trans_H4	0.0789	0.6810	0.1966	0.8479	0.2448	-0.1732	0.6734
BHE	0.1141	0.8892	0.2567	1.2303	0.3552	-0.1483	0.0120
COVID-19-Inclusive Period (2112)							
SERT_LNF	0.1088	0.7804	0.2253	1.0253	0.2960	-0.2921	0.6748
SERT_H1	0.1049	0.7932	0.2290	1.1241	0.3245	-0.2616	0.7834
SERT_H2	0.0957	0.7306	0.2109	0.9769	0.2820	-0.2914	0.7977
SERT_H3	0.0997	0.7693	0.2221	1.1181	0.3228	-0.2497	0.7925
SERT_H4	0.1182	0.8876	0.2562	1.1803	0.3407	-0.2543	0.7371
SERT_H6	0.0982	0.7099	0.2049	0.9173	0.2648	-0.2842	0.7835
SERT_H7	0.1010	0.7522	0.2171	1.0183	0.2940	-0.2841	0.7910
En_Trans_H1	0.1089	0.7456	0.2152	0.9017	0.2603	-0.3263	0.5202
En_Trans_H2	0.1095	0.7162	0.2068	0.8819	0.2546	-0.3458	0.6082
En_Trans_H4	0.1045	0.7089	0.2047	0.8377	0.2418	-0.3358	0.5893
BHE	0.1353	0.8512	0.2457	1.0181	0.2939	-0.2890	0.0093
Period Including COVID-19 and One-Year After (2212)							
SERT_LNF	0.0840	0.5875	0.1696	0.8044	0.2322	-0.2921	0.6905
SERT_H1	0.0865	0.6176	0.1783	0.8795	0.2539	-0.2616	0.7367
SERT_H2	0.0687	0.5026	0.1451	0.6659	0.1922	-0.2914	0.8296
SERT_H3	0.0818	0.5905	0.1705	0.8161	0.2487	-0.2497	0.7395
SERT_H4	0.0892	0.6410	0.1850	0.8450	0.2439	-0.2543	0.7708
SERT_H6	0.0778	0.5393	0.1557	0.7258	0.2095	-0.2842	0.7432
SERT_H7	0.0788	0.5574	0.1609	0.7762	0.2241	-0.2841	0.7460
En_Trans_H1	0.0898	0.5941	0.1715	0.7540	0.2177	-0.3263	0.4956
En_Trans_H2	0.0895	0.5725	0.1653	0.7365	0.2126	-0.3458	0.5835
En_Trans_H4	0.0865	0.5700	0.1645	0.7214	0.2082	-0.3358	0.5657
BHE	0.1143	0.7017	0.2026	0.8883	0.2564	-0.2890	0.0083

Table 4.14: The backtesting results for the equal-weighted portfolios of the proposed SERT and the encoder-only Transformer considering the dynamic transaction cost. 'MDD' is the maximum drawdown, 'SR' means Sharpe ratio, 'SO' means Sortino ratio. 'BHE' represents the buy-and-hold benchmark strategy under the equal-weighted method. 'Ann' is the notation for 'annualized'.

Pre-COVID-19 Period (1911)							
Model	AR	Ann.SR	SR	Ann.SO	SO	MDD	Turnover
SERT_LNF	0.1145	1.1008	0.3178	1.5609	0.4506	-0.1297	0.6941
SERT_H1	0.0938	0.8959	0.2586	1.1700	0.3377	-0.1481	0.8402
SERT_H2	0.1023	0.9654	0.2787	1.1489	0.3317	-0.1182	0.8240
SERT_H3	0.0824	0.7830	0.2260	0.9822	0.2835	-0.1460	0.8503
SERT_H4	0.1159	1.0813	0.3122	1.2697	0.3665	-0.1305	0.7783
SERT_H6	0.0932	0.8528	0.2462	1.0662	0.3078	-0.1523	0.8197
SERT_H7	0.0871	0.7986	0.2305	1.0021	0.2893	-0.1466	0.7932
En_Trans_H1	0.1085	1.0516	0.3036	1.5570	0.4495	-0.0983	0.5974
En_Trans_H2	0.1188	1.0585	0.3056	1.5451	0.4460	-0.1169	0.6541
En_Trans_H4	0.1190	1.0620	0.3066	1.2624	0.3644	-0.1277	0.6486
BHV	0.1388	1.1795	0.3405	1.6054	0.4634	-0.1297	0.0120
COVID-19-Inclusive Period (2112)							
SERT_LNF	0.1406	1.0701	0.3089	1.5182	0.4383	-0.1914	0.6393
SERT_H1	0.1149	0.9101	0.2627	1.2083	0.3488	-0.1920	0.7529
SERT_H2	0.1344	1.0828	0.3126	1.4585	0.4210	-0.1413	0.7468
SERT_H3	0.1147	0.9394	0.2712	1.3114	0.3786	-0.1485	0.7590
SERT_H4	0.1430	1.1621	0.3355	1.5470	0.4466	-0.1373	0.7028
SERT_H6	0.1094	0.7819	0.2257	0.8566	0.2473	-0.2653	0.7481
SERT_H7	0.1149	0.8939	0.2580	1.1837	0.3417	-0.2030	0.7530
En_Trans_H1	0.1335	0.9933	0.2868	1.2334	0.3560	-0.2300	0.5204
En_Trans_H2	0.1397	1.0041	0.2899	1.3237	0.3821	-0.2206	0.5801
En_Trans_H4	0.1368	0.9880	0.2852	1.1886	0.3431	-0.2252	0.5694
BHV	0.1540	1.0688	0.3085	1.3057	0.3769	-0.2352	0.0093
Period Including COVID-19 and One-Year After (2212)							
SERT_LNF	0.1102	0.8095	0.2337	1.1912	0.3439	-0.1963	0.6597
SERT_H1	0.0969	0.7180	0.2073	0.9830	0.2838	-0.1920	0.7052
SERT_H2	0.0981	0.7419	0.2142	0.9537	0.2753	-0.2628	0.7871
SERT_H3	0.0931	0.7081	0.2044	0.9985	0.2882	-0.1801	0.7018
SERT_H4	0.1073	0.8154	0.2354	1.0600	0.3060	-0.2485	0.7390
SERT_H6	0.0848	0.5880	0.1698	0.7022	0.2027	-0.2653	0.7063
SERT_H7	0.0927	0.6808	0.1965	0.9623	0.2778	-0.2030	0.7089
En_Trans_H1	0.1094	0.7681	0.2217	1.0088	0.2912	-0.2300	0.4994
En_Trans_H2	0.1118	0.7591	0.2191	1.0262	0.2963	-0.2206	0.5654
En_Trans_H4	0.1122	0.7826	0.2259	1.0042	0.2899	-0.2252	0.5462
BHV	0.1308	0.8654	0.2498	1.1270	0.3253	-0.2352	0.0083

Table 4.15: The backtesting results for the value-weighted portfolios of the proposed SERT and the encoder-only Transformer models considering the dynamic transaction cost. MDD is the maximum drawdown, 'SR' means Sharpe ratio, 'SO' means Sortino ratio. 'BHV' represents the buy-and-hold benchmark strategy under the value-weighted method. 'Ann' is the notation for 'annulized'.

Pre-COVID-19 Period (1911)						
Model	MDD	AR	sharpe (SR)	Ann.SR	sortino (SO)	Ann.SO
SERT_LNF	-0.3566	0.0968	0.2622	0.9082	0.3673	1.2725
SERT_H4	-0.2956	0.1066	0.2737	0.9481	0.3338	1.1561
En_Trans_H1	-0.2941	0.0940	0.2533	0.8773	0.3732	1.2928
P_Trans_LNF	-0.2949	0.0916	0.2431	0.8421	0.3239	1.1222
P_Trans_H6	-0.3511	0.0979	0.2555	0.8850	0.3493	1.2100
C_Trans_H1	-0.2349	0.1005	0.2646	0.9166	0.3683	1.2759
BHE	-0.3701	0.1139	0.2588	0.8965	0.3631	1.2578
COVID-19-Inclusive Period (2112)						
Model	MDD	AR	sharpe (SR)	Ann.SR	sortino (SO)	Ann.SO
SERT_LNF	-0.5154	0.1263	0.2622	0.9082	0.3500	1.2124
SERT_H4	-0.4119	0.1374	0.2988	1.0350	0.4095	1.4185
En_Trans_H1	-0.6018	0.1222	0.2428	0.8412	0.2963	1.0263
P_Trans_LNF	-0.5589	0.1124	0.2394	0.8294	0.2826	0.9788
P_Trans_H6	-0.4218	0.1284	0.2840	0.9837	0.4140	1.4341
C_Trans_H1	-0.5789	0.1271	0.2455	0.8505	0.2948	1.0211
BHE	-0.5310	0.1349	0.2472	0.8563	0.2988	1.0351
Period Including COVID-19 and One-Year After (2212)						
Model	MDD	AR	sharpe (SR)	Ann.SR	sortino (SO)	Ann.SO
SERT_LNF	-0.5154	0.1016	0.2041	0.7072	0.2827	0.9794
SERT_H4	-0.4119	0.1089	0.2249	0.7791	0.3049	1.0563
En_Trans_H1	-0.6018	0.1024	0.1955	0.6771	0.2502	0.8669
P_Trans_LNF	-0.5589	0.0886	0.1809	0.6268	0.2241	0.7765
P_Trans_H6	-0.4218	0.1070	0.2204	0.7636	0.3264	1.1308
C_Trans_H1	-0.5789	0.1066	0.1998	0.6920	0.2547	0.8822
BHE	-0.5308	0.1141	0.2148	0.7441	0.2760	0.9561

Table 4.16: Back-testing performance of the equal-weighted portfolio between the best models in each group considering the static transaction cost. The notation of 'Ann' means annualized. SR and SO are denoted as Sharpe Ratio and Sortino Ratio, respectively.

More specifically, the comparison of Sortino ratio for each model in different periods gives a clear picture of the trading strategy-wise model performance, which is shown in Figure 4.13. From Figure 4.13, the best proposed models and their benchmarks show no advantage or insignificant advantage over the buy-and-hold strategy for the equal-weighted portfolio (BHE) during the period of '1911' (blue bars). However, the best proposed models (SERT_H4 and P_Trans_H6) illustrate the great advantage compared with the BHE during the period of '2112'. Their Sortino ratios are approximately 47% higher than BHE, followed by the LNF SERT (SERT_LNF), 17% higher than BHE. And during the period of '2212', the best

proposed models still demonstrate superiority to the BHE benchmark. This supports the conclusion that the best proposed models achieve significantly higher profitability by considering downside risks during extreme market fluctuations.

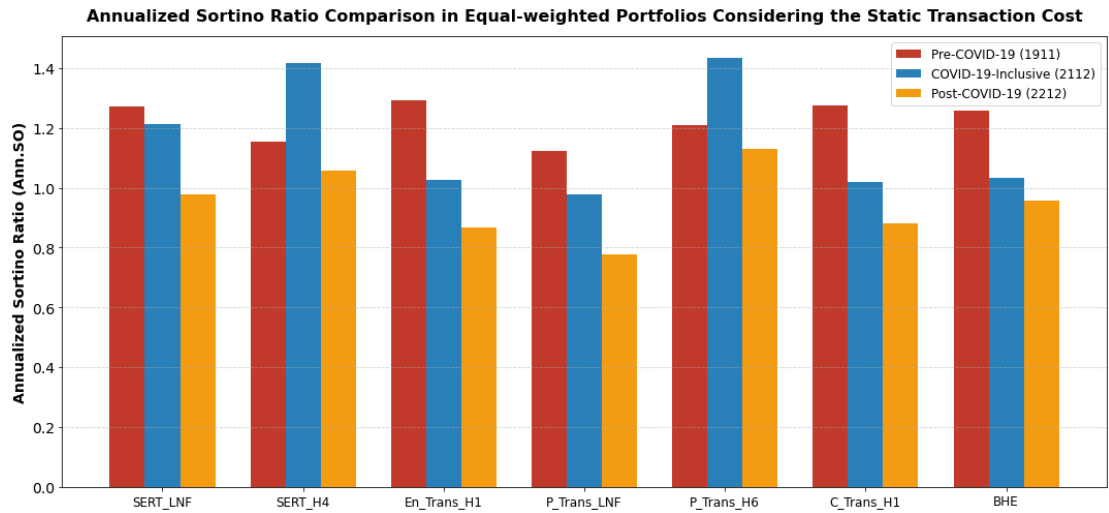


Figure 4.13: Comparison of Sortino ratios for equal-weighted portfolios considering the static transaction cost. The Y-axis presents the value of Sortino Ratio, and the X-axis presents the models. The red, blue and orange bars indicate the Sortino Ratios of each model during the period of ‘1911’, ‘2112’ and ‘2212’ respectively. ‘BHE’ is the notation of the buy-and-hold strategy for the equal-weighted portfolio.

Figure 4.14 is the cumulative return plot of selected models under the equal-weighted method. It illustrates that Model SERT_H4 outperforms the BHE strategy and achieves the highest cumulative returns in most periods. The best proposed pre-trained Transformer model, P_Trans_H6, is right below the BHE strategy and results into the second-best model for better cumulative returns. On the other hand, the LNF pre-trained Transformer model (P_Trans_LNF) performs noticeably worse than alternatives during the extreme market fluctuation period, which is followed by the single-head standard encoder-only Transformer model (En_Trans_H1). The LNF SERT model (SERT_LNF) performs slightly higher than the Model En_Trans_H1 but is still worse than the average. This concludes that the LNF models are not ideal in the factor investing context of the equal-weighted method in the static transaction cost scenario. Moreover, amid the incredible difference between SERT_H4 and En_Trans_H1, it presents the solid evidence of how the pre-trained method in the proposed SERT can improve investment performance. The similar phenomenon happens between the best standard Transformer model and the pre-trained Transformer models as well, which

confirms the advantage of pre-training structure in Transformers for economic models.

Similar findings are shown in the value-weighted portfolios in the static transaction cost method, although the value-weighted method highly reduces the absolute capital gain of all models. Nevertheless, the difference between the two models is significant for both the absolute capital gain and the gain under the downside risk during the pandemic period ‘2112’. Concretely, the performance of the pre-trained Transformer model (P_Trans_H2) is better than that of the SERT model (SERT_H1). Model P_Trans_H2 has a Sortino ratio of 0.46, which is 0.0458 higher than that of SERT_H1. Furthermore, the LNF SERT outperforms the LNF pre-trained Transformer model, while the standard encoder-only Transformer underperforms the standard Transformer model.

Figure 4.15 shows the comparison of Sortino ratios for value-weighted (VW) portfolios during the three periods. The Sortino ratios of value-weighted portfolios agree with the findings in equal-weighted portfolios that the best proposed models (SERT_H1 and P_Trans_H2) dramatically outperform the buy-and-hold (BHV) benchmark during the period of ‘2112’ (28% and 44% higher than BHV, respectively), followed by LNF SERT (SERT_LNF) and ‘2212’ (13% and 20% higher than BHV, respectively). Whereas the proposed models and their benchmarks fail to demonstrate superiority over the BHV strategy during the period of ‘1911’. It is also interesting to find that the LNF configuration performs better on the SERT model compared with the pre-trained Transformer model in both equal-weighted and value-weighted methods for capturing the extreme market fluctuations.

Figure 4.16 shows the cumulative return plot of the best models in value-weighted portfolios in the static transaction cost method. It shows all models have cumulative returns considerably lower (almost half lower) than the BHV strategy. This phenomenon may be caused by the weights distribution of the value-weighted method or over-penalization of high static transaction costs (50bps). The weight distribution is based on the market capitalization size of each stock. It means larger weights contribute to the large capital stocks which have lower volatilities. However, from the robust examination of the dynamic transaction cost method,

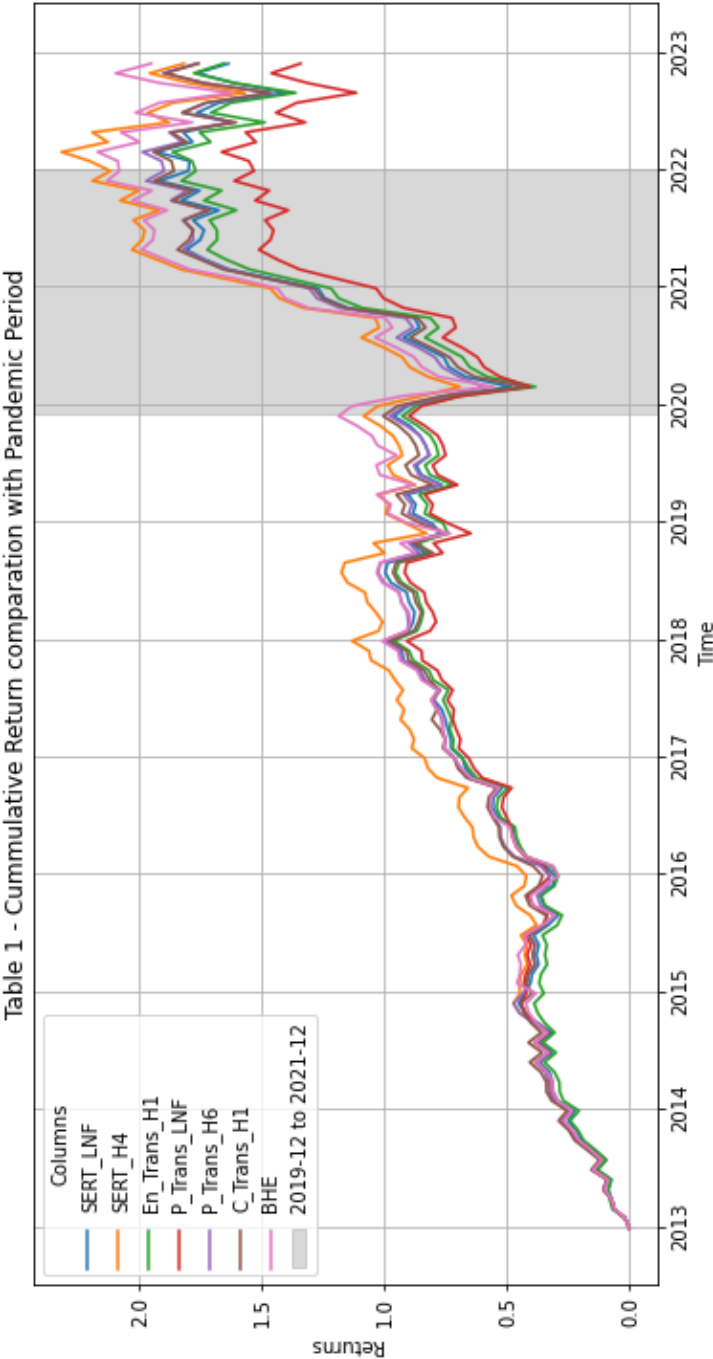


Figure 4.14: Sign equal-weighted accumulative return plots considering the static transaction cost. The area covered by a grey shadow indicates the pandemic period. ‘BHE’ is the buy-and-hold strategy for the equal-weighted portfolio.

Pre-COVID-19 Period (1911)						
Model	MDD	AR	sharpe (SR)	Ann.SR	sortino (SO)	Ann.SO
SERT_LNF	-0.2475	0.0546	0.2631	0.9115	0.2918	1.0108
SERT_H1	-0.2998	0.0540	0.2345	0.8124	0.2755	0.9542
En_Trans_H2	-0.2395	0.0605	0.2594	0.8985	0.3238	1.1216
P_Trans_LNF	-0.3061	0.0532	0.2304	0.7980	0.2651	0.9185
P_Trans_H2	-0.3260	0.0535	0.2413	0.8359	0.2923	1.0126
C_Trans_H4	-0.3061	0.0566	0.2361	0.8180	0.2954	1.0235
BHV	-0.2963	0.1197	0.2956	1.0240	0.4243	1.4698
COVID-19-Inclusive Period (2112)						
Model	MDD	AR	sharpe (SR)	Ann.SR	sortino (SO)	Ann.SO
SERT_LNF	-0.3382	0.0737	0.2754	0.9540	0.3609	1.2503
SERT_H1	-0.2998	0.0760	0.2907	1.0069	0.4142	1.4347
En_Trans_H2	-0.3838	0.0678	0.2517	0.8721	0.3147	1.0902
P_Trans_LNF	-0.3415	0.0648	0.2531	0.8767	0.3179	1.1013
P_Trans_H2	-0.3260	0.0801	0.3113	1.0785	0.4600	1.5934
C_Trans_H4	-0.4205	0.0745	0.2509	0.8693	0.3210	1.1121
BHV	-0.4513	0.1337	0.2673	0.9260	0.3263	1.1303
Period Including COVID-19 and One-Year After (2212)						
Model	MDD	AR	sharpe (SR)	Ann.SR	sortino (SO)	Ann.SO
SERT_LNF	-0.3382	0.0528	0.1812	0.6277	0.2244	0.7773
SERT_H1	-0.2998	0.0670	0.2366	0.8197	0.3407	1.1801
En_Trans_H2	-0.3838	0.0551	0.1907	0.6607	0.2461	0.8525
P_Trans_LNF	-0.3415	0.0495	0.1835	0.6357	0.2387	0.8269
P_Trans_H2	-0.3260	0.0672	0.2436	0.8440	0.3553	1.2307
C_Trans_H4	-0.4205	0.0653	0.2076	0.7191	0.2797	0.9689
BHV	-0.4513	0.1156	0.2311	0.8006	0.3031	1.0500

Table 4.17: Back-testing performance of the value-weighted portfolio between the best models in each group considering the static transaction cost. The notation of ‘Ann’ means annualized. SR and SO are denoted as Sharpe Ratio and Sortino Ratio, respectively. AR means annualized return.

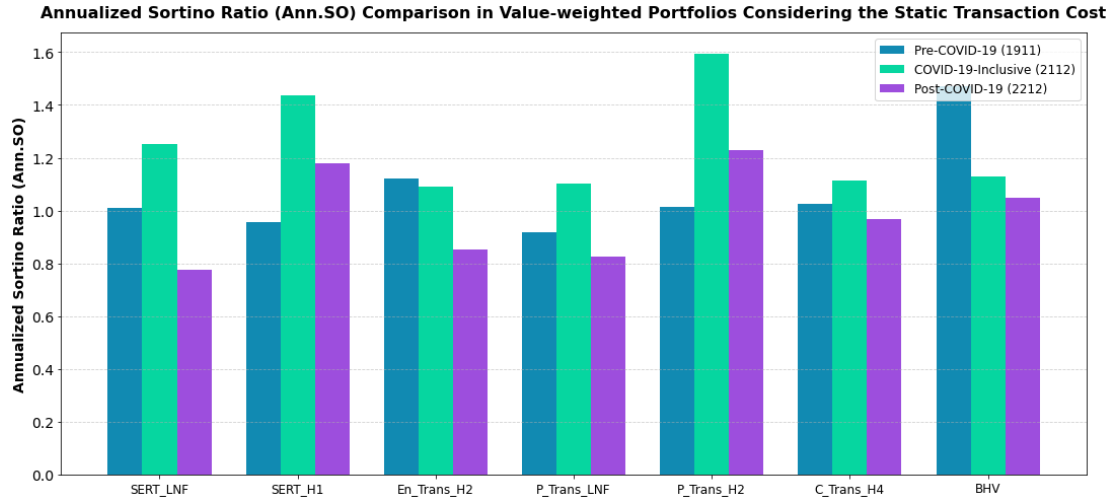


Figure 4.15: Value-weighted accumulative return plots considering the static transaction cost. ‘BHV’ is the buy-and-hold strategy for the value-weighted portfolio.

there is a high probability that the difference is caused by over-penalization of static transaction costs. Additionally, under the value-weighted conditions, the standard encoder-only Transformer model (En_Trans.H2) has the highest returns, but during the high fluctuation, the proposed models, P_Trans.H2 and SERT_H1, excel all alternative models. P_Trans.LNF, En_Trans.H2, and SERT.LNF are still the last three models in this case.

Figure 4.17 and Figure 4.18 show the accumulative return plots for the models with softmax filters of the equal-weighted and value-weighted methods, respectively considering the static transaction cost. In the equal-weighted portfolio, after filtering out bottom 50% signals, the standard encoder-only Transformer model with 2 attention heads (En_Trans.H2) outperforms all other models and gains the highest returns, while almost all models perform the BHE insignificantly apart from Model En_Trans.H1. Whereas En_Trans.H1 works better than BHE before and at the beginning of the pandemic, but not in the middle of the pandemic and one year afterwards. On the other hand, from Figure 4.18, it suggests that the softmax filter works more efficiently in the value-weighted circumstance. The softmax filter further excludes the less profitable trades after the value weighting process, significantly improving the accumulated returns of all models. Also, it does not make all models surpass the BHV strategy. However, the worst model varies in different periods, concretely, SERT.LNF before the pandemic, P_Trans.LNF during the pandemic and En_Trans.H1 after the pan-

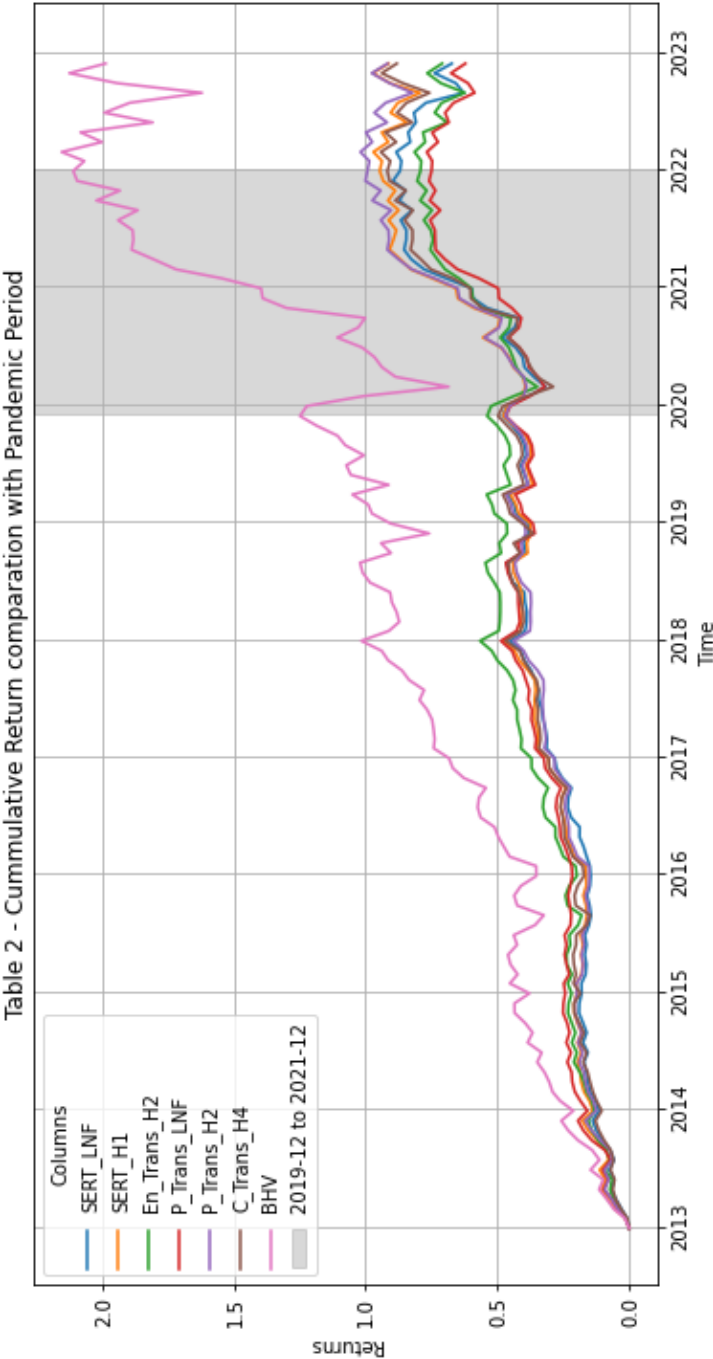


Figure 4.16: Sign value-weighted accumulative return plots considering the static transaction cost. The area covered by a grey shadow indicates the pandemic period. ‘BHF’ is the buy-and-hold strategy for the equal-weighted portfolio.

demic. Figure 4.17 and Figure 4.18 demonstrate that the softmax filter tends to eliminate the difference between models and make them perform approaching the BHV. This supports the conclusion drawn from Section 4.5.3.1 and Section 4.5.3.2 that the softmax signal filter adjusts weight distribution for value-weighted portfolios.

The dynamic transaction cost robustness results comparison analysis is conducted to assess the overall impact of turnover rates on models' profitability in the equal-weighted method and the value-weighted method. In equal-weighted portfolio, the best model in each group of standard Transformer models is consistent with the ones in static transaction cost method, but the best models of encoder-only Transformers vary. Compared with the indicators in the static transaction cost method, it shows no evidence of static transaction cost (50bps) over-penalizes the models' profitability according to annualized returns, but it enlarges the maximum drawdown. In contrast, even if the dynamic transaction cost rate is lowering to 20bps, the Sharpe ratio and Sortino ratio are highly reduced due to the high turnover rate, and no model outperforms the buy-and-hold benchmark (BHE). However, in dynamic transaction cost scenario, the proposed pre-trained Transformer model (P_Trans_H6) and single-head SERT (SERT_H1) outperform the benchmarks according to the Sortino ratio during the COVID-19 period, which shows in Figure 4.19. Non pre-training benchmark models, C_Trans_H1 and En_Trans_H1 present the resilience in the pre-COVID period, and P_Trans_H6 shows superiority lasting in Period Including COVID-19 and One-Year After (2212). In value-weighted portfolios, it is interesting to find that static transaction cost over-penalizes the profitability of the models again. The turnover adjusted transaction cost method significantly higher the annualized return, Sharpe ratio and Sortino ratio, and lower the maximum drawdown. According to Figure 4.21, the proposed pre-trained models slightly overweigh the SERT_LNF benchmark in COVID-19-Inclusive Period (2112), are the most profitable models during that period. This aligns with the value of turnover rate. The cumulative return plots, Figure 4.20 and Figure 4.22 verify the previous findings. Furthermore, the softmax signal filter narrows the difference between models. It significantly improves the models' profitability, and it optimizes the turnover rate, which leads the best proposed pre-trained models to outperform the buy-and-hold benchmarks.

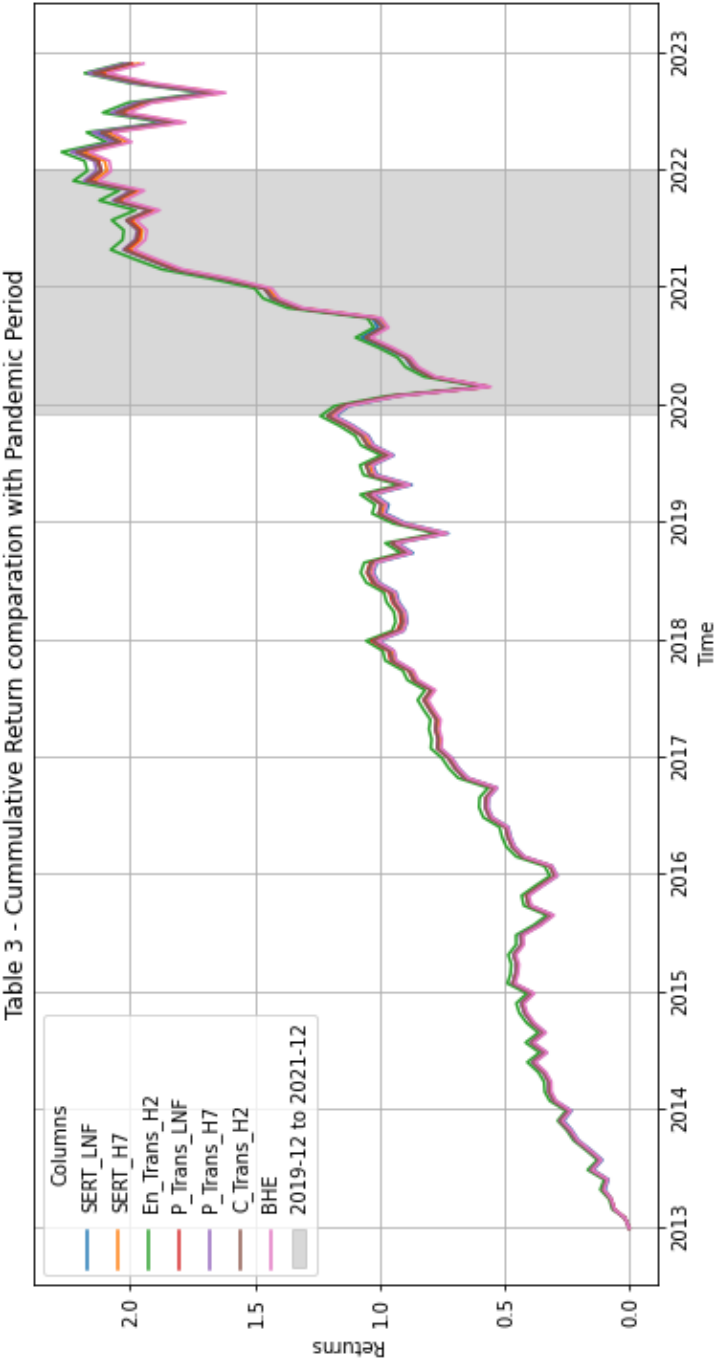


Figure 4.17: Sign equal-weighted accumulative return plots with softmax signal filter considering the static transaction cost. The area covered by a grey shadow indicates the pandemic period. ‘BHE’ is the buy-and-hold strategy for the equal-weighted portfolio.

Table 4 - Cumulative Return comparison with Pandemic Period

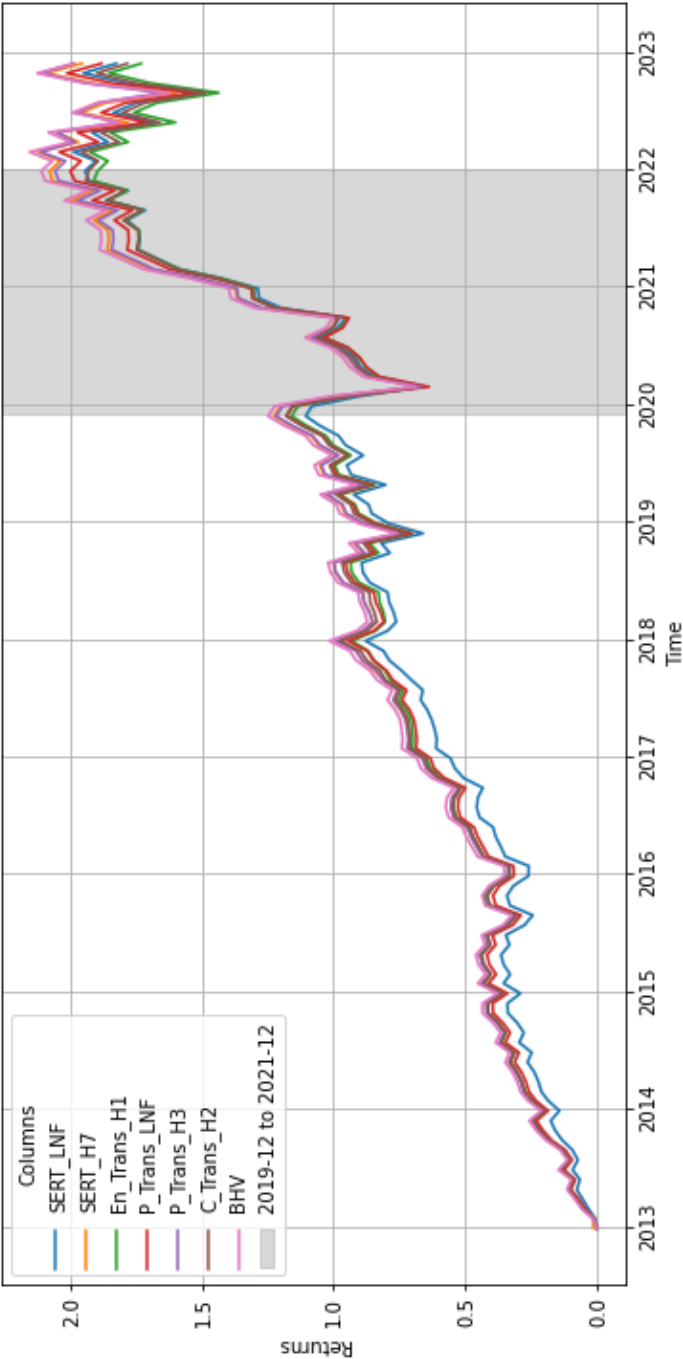


Figure 4.18: Sign value-weighted accumulative return plots with softmax signal filter considering the static transaction cost. The area covered by a grey shadow indicates the pandemic period. ‘BHE’ is the buy-and-hold strategy for the equal-weighted portfolio.

Model	AR	Ann.SR	SR	Ann.SO	SO	MDD	Turnover
Pre-COVID-19 Period (1911)							
P_Trans_LNF	0.0733	0.6683	0.1929	0.8658	0.2499	-0.1511	0.7229
P_Trans_H6	0.0787	0.7087	0.2046	0.9390	0.2711	-0.1394	0.7589
C_Trans_H1	0.0837	0.7586	0.2190	1.0321	0.2979	-0.1297	0.6654
SERT_LNF	0.0782	0.7278	0.2101	1.0021	0.2893	-0.1399	0.7301
SERT_H1	0.0717	0.6578	0.1899	0.9092	0.2625	-0.1625	0.8792
En_Trans_H1	0.0786	0.7247	0.2092	1.0519	0.3037	-0.1483	0.6129
BHE	0.1141	0.8892	0.2567	1.2303	0.3552	-0.1483	0.0120
COVID-19-Inclusive Period (2112)							
P_Trans_LNF	0.0944	0.6968	0.2011	0.8052	0.2324	-0.2956	0.6980
P_Trans_H6	0.1107	0.8454	0.2441	1.1946	0.3449	-0.2428	0.6875
C_Trans_H1	0.1123	0.7503	0.2166	0.8926	0.2577	-0.3105	0.5733
SERT_LNF	0.1088	0.7804	0.2253	1.0253	0.2960	-0.2921	0.6748
SERT_H1	0.1049	0.7932	0.2290	1.1241	0.3245	-0.2616	0.7834
En_Trans_H1	0.1089	0.7456	0.2152	0.9017	0.2603	-0.3263	0.5202
BHE	0.1353	0.8512	0.2457	1.0181	0.2939	-0.2890	0.0093
Period Including COVID-19 and One-Year After (2212)							
P_Trans_LNF	0.0705	0.5036	0.1454	0.6125	0.1768	-0.2956	0.7144
P_Trans_H6	0.0906	0.6479	0.1870	0.9366	0.2704	-0.2428	0.6434
C_Trans_H1	0.0927	0.6036	0.1742	0.7622	0.2200	-0.3105	0.5460
SERT_LNF	0.0840	0.5875	0.1696	0.8044	0.2322	-0.2921	0.6905
SERT_H1	0.0865	0.6176	0.1783	0.8795	0.2539	-0.2616	0.7367
En_Trans_H1	0.0898	0.5941	0.1715	0.7540	0.2177	-0.3263	0.4956
BHE	0.1143	0.7017	0.2026	0.8883	0.2564	-0.2890	0.0083

Table 4.18: Back-testing performance of the equal-weighted portfolio between the best models in each group considering the dynamic transaction cost. The notation of ‘Ann’ means annualized. SR and SO denote the Sharpe Ratio and the Sortino Ratio, respectively.

4.6 Conclusion

In this study, the innovative Transformer models, SERT and pre-trained Transformer, are developed specifically for financial and economic time series data. They are examined in stock pricing and factor investing assignments as the novel frameworks for the factor model. To the best of my knowledge, this is the first attempt for the pre-trained Transformer application on stock pricing and factor investing strategy building context. From the model engineering perspective, this article proves that the Transformer models, as a category of

Model	AR	Ann.SR	SR	Ann.SO	SO	MDD	Turnover
Pre-COVID-19 Period (1911)							
P_Trans_LNF	0.1125	1.0716	0.3094	1.3302	0.3840	-0.1234	0.6952
P_Trans_H7	0.0983	0.9460	0.2731	1.3033	0.3762	-0.1342	0.8495
C_Trans_H4	0.1256	1.0954	0.3162	1.4692	0.4241	-0.1138	0.4638
SERT_LNF	0.1145	1.1008	0.3178	1.5609	0.4506	-0.1297	0.6941
SERT_H4	0.1159	1.0813	0.3122	1.2697	0.3665	-0.1305	0.7783
En_Trans_H2	0.1188	1.0585	0.3056	1.5451	0.4460	-0.1169	0.6541
BHV	0.1388	1.1795	0.3405	1.6054	0.4634	-0.1297	0.0120
COVID-19-Inclusive Period (2112)							
P_Trans_LNF	0.1315	1.0401	0.3002	1.2939	0.3735	-0.1959	0.6775
P_Trans_H7	0.1235	1.0156	0.2932	1.4776	0.4266	-0.1596	0.7696
C_Trans_H4	0.1431	1.0245	0.2957	1.2709	0.3669	-0.2279	0.4060
SERT_LNF	0.1406	1.0701	0.3089	1.5182	0.4383	-0.1914	0.6393
SERT_H4	0.1430	1.1621	0.3355	1.5470	0.4466	-0.1373	0.7028
En_Trans_H2	0.1397	1.0041	0.2899	1.3237	0.3821	-0.2206	0.5801
BHV	0.1540	1.0688	0.3085	1.3057	0.3769	-0.2352	0.0093
Period Including COVID-19 and One-Year After (2212)							
P_Trans_LNF	0.1055	0.7924	0.2287	1.0260	0.2962	-0.1959	0.6967
P_Trans_H7	0.1050	0.7752	0.2238	1.1152	0.3219	-0.1656	0.7357
C_Trans_H4	0.1232	0.8376	0.2418	1.0988	0.3172	-0.2279	0.3875
SERT_LNF	0.1102	0.8095	0.2337	1.1912	0.3439	-0.1963	0.6597
SERT_H4	0.1073	0.8154	0.2354	1.0600	0.3060	-0.2485	0.7390
En_Trans_H2	0.1118	0.7591	0.2191	1.0262	0.2963	-0.2206	0.5654
BHV	0.1308	0.8654	0.2498	1.1270	0.3253	-0.2352	0.0083

Table 4.19: Back-testing performance of the value-weighted portfolio between the best models in each group considering the dynamic transaction cost. The notation of ‘Ann’ means annualized. SR and SO are denoted as Sharpe Ratio and Sortino Ratio respectively.

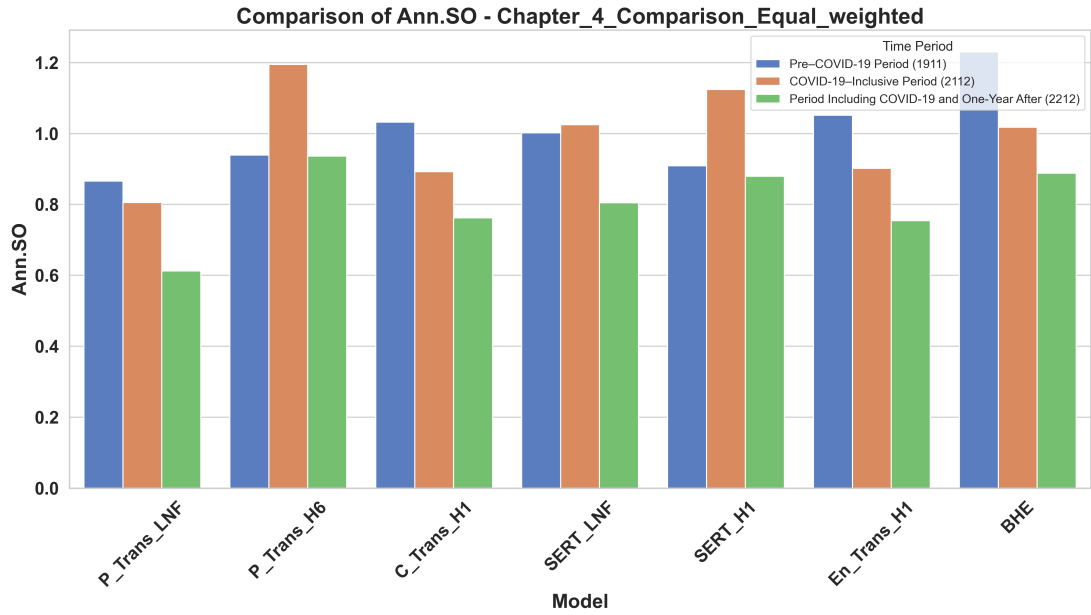


Figure 4.19: Comparison of sortinos for equal-weighted portfolios considering the dynamic transaction cost. The Y-axis presents the value of Sortino Ratio, and the X-axis presents the models. The blue, orange and grey bars indicate the Sortino Ratios of each model during the period of ‘1911’, ‘2112’ and ‘2212’ respectively. ‘BHE’ is the notation of the buy-and-hold strategy for an equal-weighted portfolio.

LLMs, are not only suitable for enormous datasets, such as datasets of NLP tasks, speech or image recognition and video generation, but also minuscule datasets such as financial and economic numerical time series data. They capture the variation of the economic data with high temporal sparsity characteristics and significantly improve the prediction accuracy under extreme market conditions, such as the price collapses and lift caused by the pandemic. In addition, the pre-training process verifies its necessity for improving model performance when the data contains the aforementioned characteristics and has missing values and noise. Moreover, the encoder-only Transformer is the simplest and most commonly used LLM in Fintech; it shows advantages in model fitness as well as prediction performance compared to the pre-trained Transformer models. Single-block SERT presents instability and inconsistency to some extent. They show sensitivity to the hyperparameters during the training process. These break the illusion that model complexity is positively linked to the model overfitting issue. This study also finds that attention head numbers affect the prediction performance at an unnoticeable level. Pre-trained Transformer models are relatively more stable than SERTs, which show a positive relationship with attention head numbers, and there is an optimal number of attention heads for each configuration. In addition, the layer

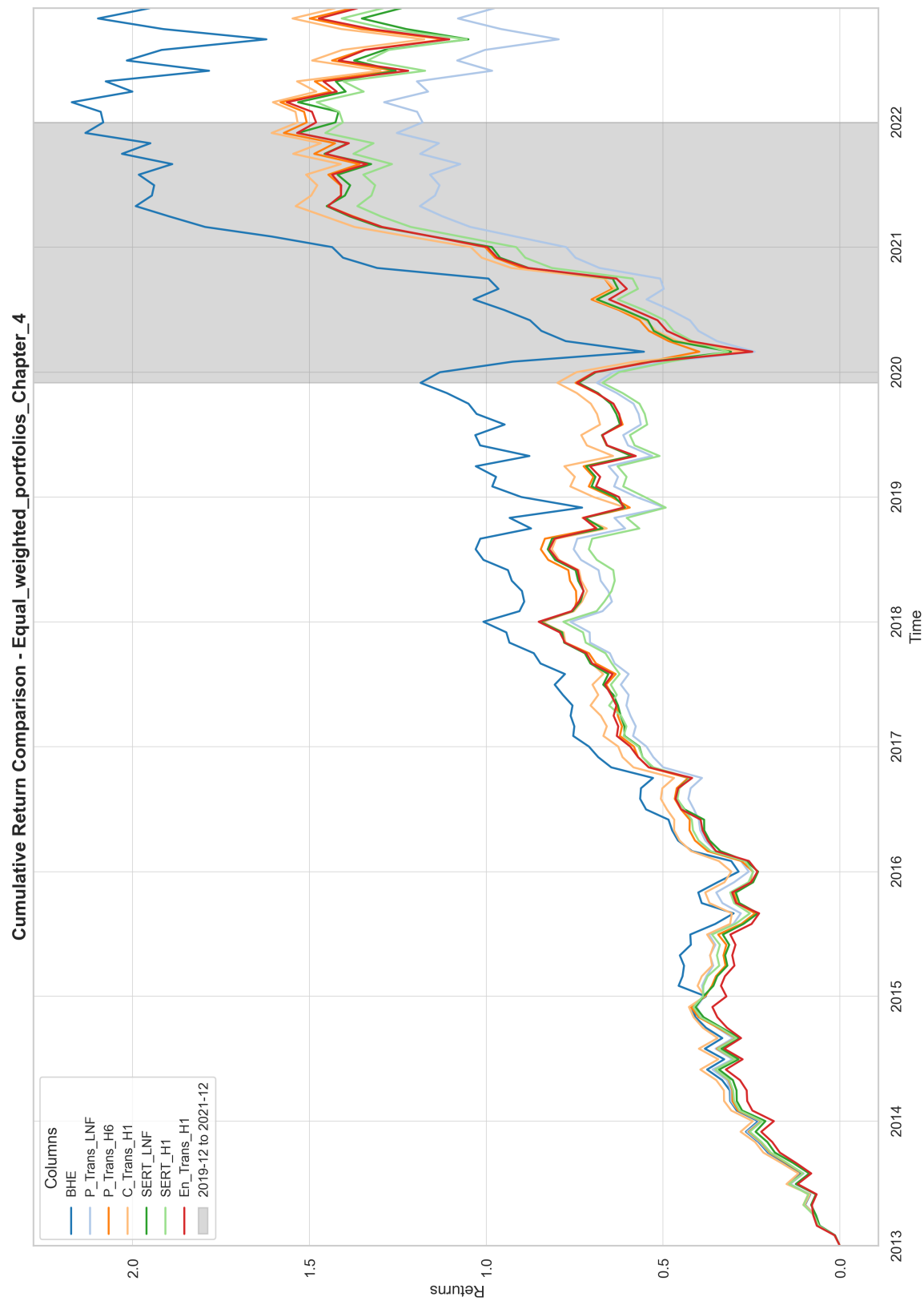


Figure 4.20: Sign equal-weighted accumulative return plots considering dynamic transaction cost. The area covered by a grey shadow indicates the pandemic period. 'BHE' is the buy-and-hold strategy for the equal-weighted portfolio.

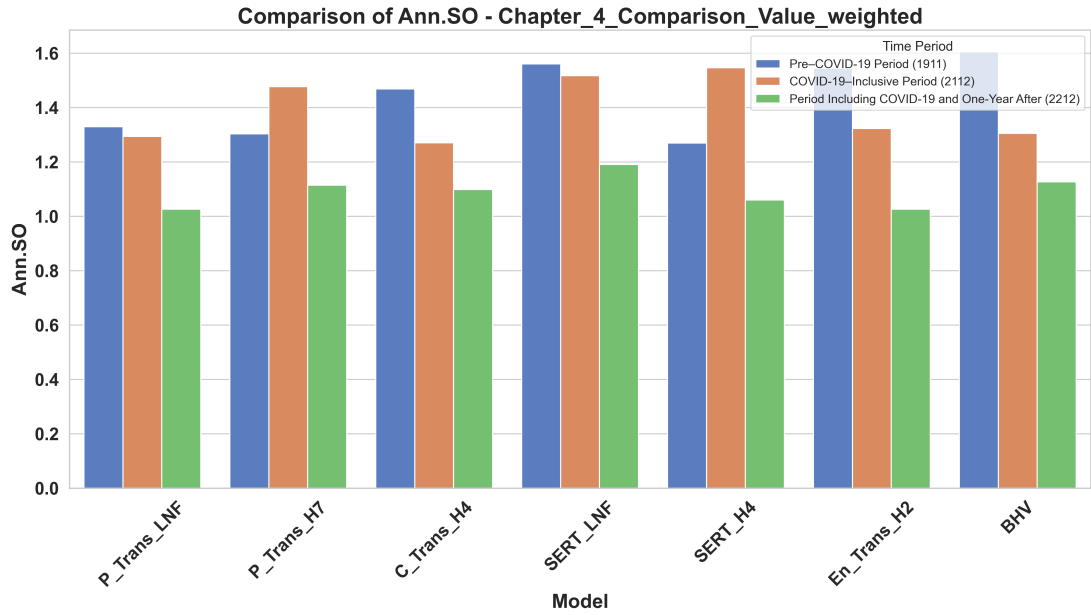


Figure 4.21: Value-weighted accumulative return plots considering the dynamic transaction cost. ‘BHV’ is the buy-and-hold strategy for the value-weighted portfolio.

normalization first setting does not perform excellent in the predictive power of this case. It could possibly work better in a deeper structure and a noisier dataset. However, they perform well in backtesting under a dynamic transaction costs configuration.

The models are examined for extreme market fluctuation adaptivity via three separate periods: the period before COVID-19, the period including COVID-19 and the period including one year after COVID-19. During these periods, the U.S. stock market experienced a moderate uptrend, an extreme uptrend and an extreme downtrend. The proposed models show magnificent adaptivity to extreme market fluctuations. In other words, they achieve excellent performance in capturing patterns in highly varied stock prices. Moreover, in the static transaction cost scenario, the factor investing strategy based on the proposed models shows high profitability, and some of them show even higher performance than the ‘buy-and-hold’ benchmarks under the equal-weighted portfolio method. They also show a fantastic capability for hedging the downside risks. It is worth noting that the softmax signal filter only removes the difference between models, making them approach the buy-and-hold benchmarks, but does not facilitate strategy-wise performance in static transaction cost scenarios. However, in dynamic transaction-cost scenarios, which are included in the turnover rate and

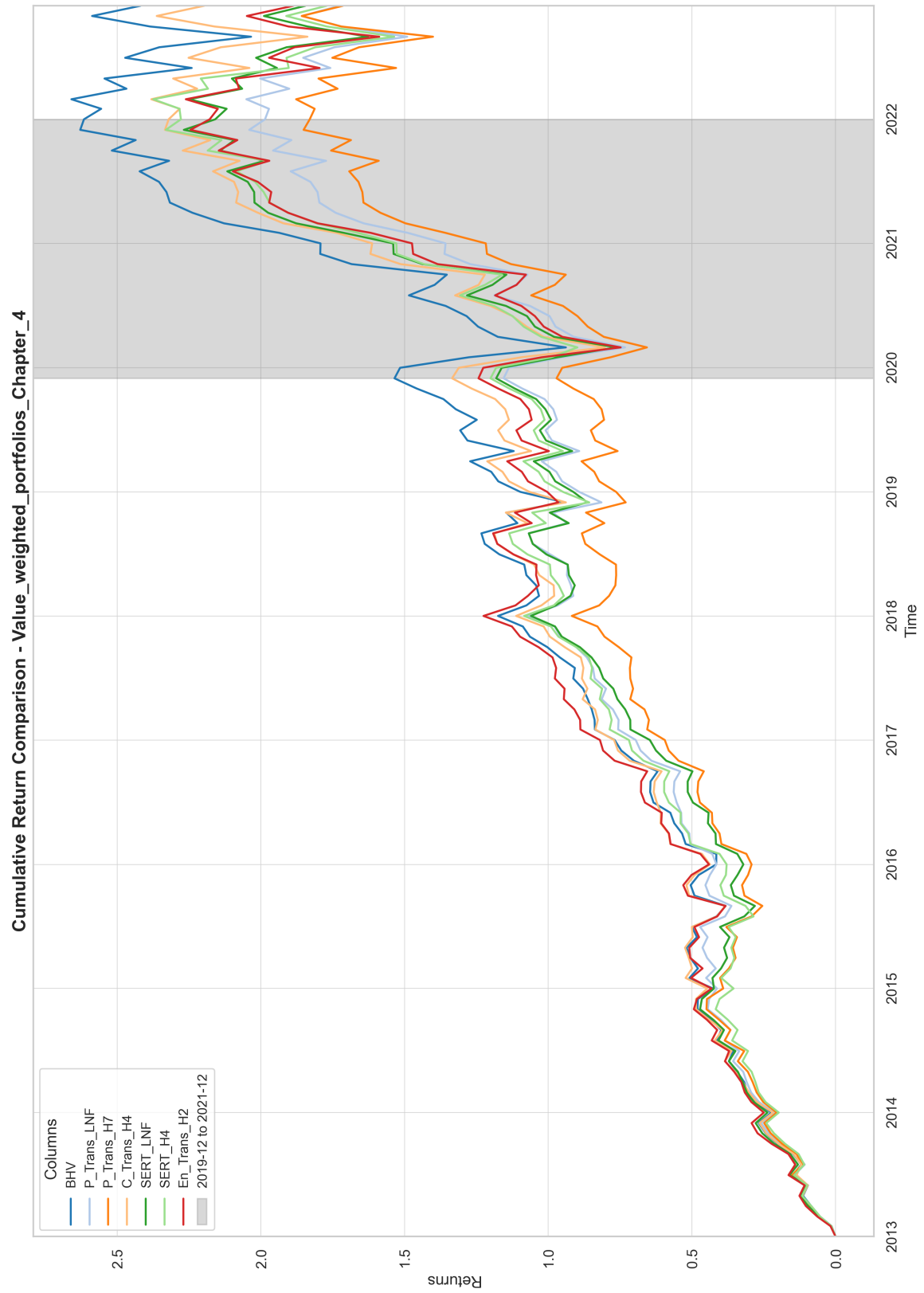


Figure 4.22: Sign value-weighted accumulative return plots. The area covered by a grey shadow indicates the pandemic period. 'BHE' is the buy-and-hold strategy in the value-weighted method.

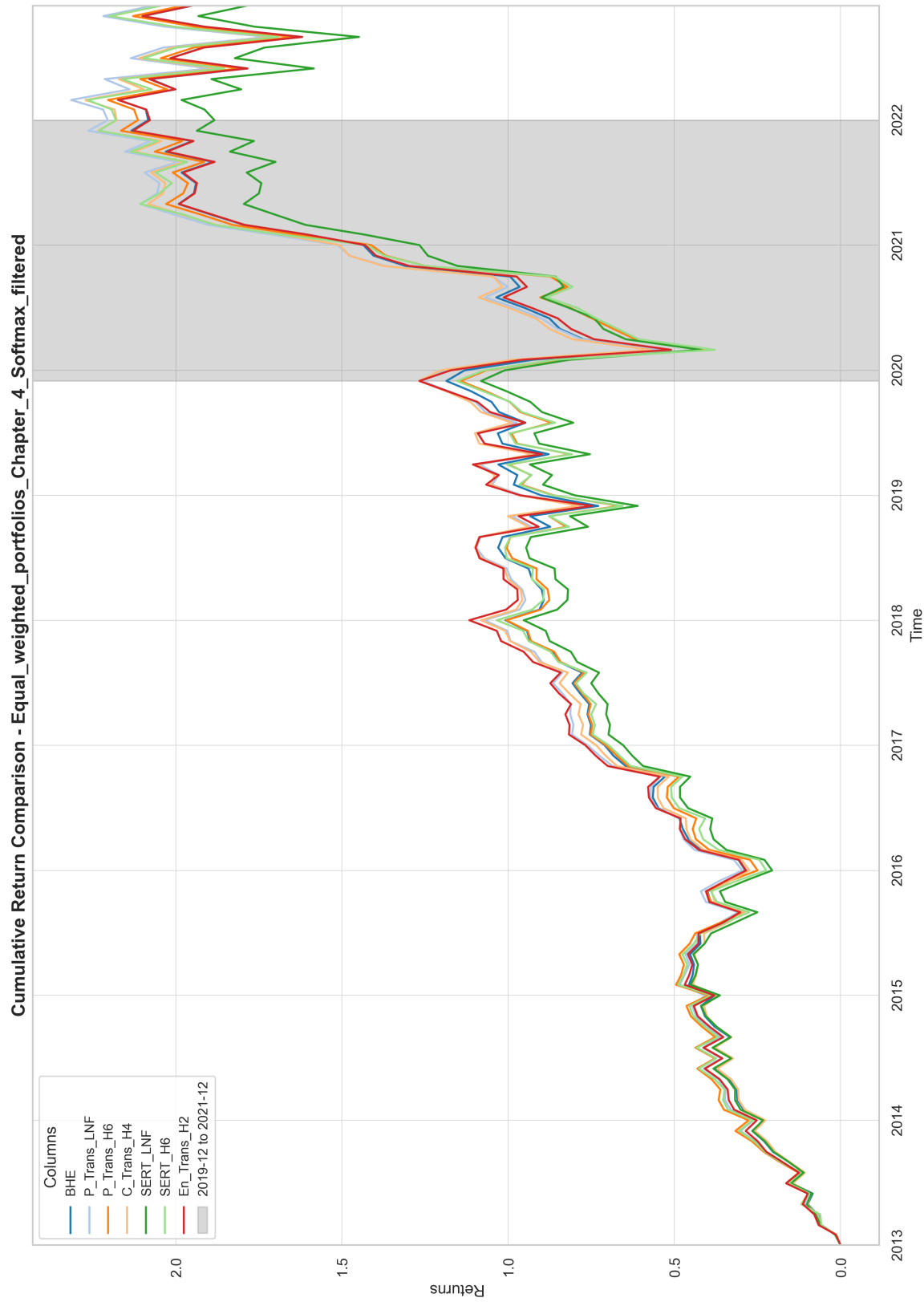


Figure 4.23: Equal-weighted with Softmax trading signal filter accumulative return plots considering the dynamic transaction cost.

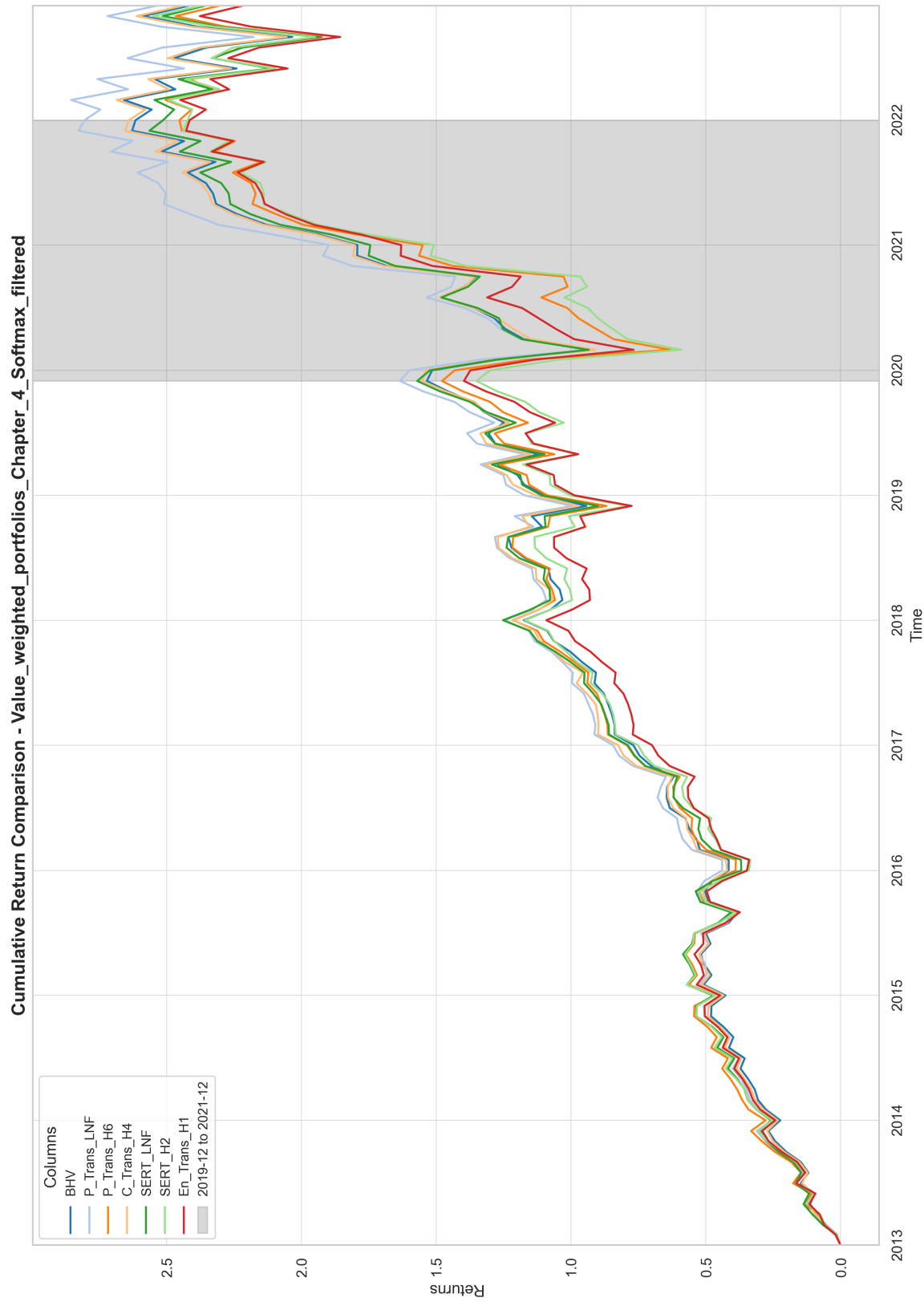


Figure 4.24: Value-weighted with Softmax trading signal filter accumulative return plots considering the dynamic transaction cost.

the adjusted large-cap-specific rate (20bps), the high turnover rate significantly erodes the profitability of these models in most cases and reduces backtesting performance to levels below buy-and-hold benchmarks. Whereas, even with the high conservative static transaction cost restrictions and high turnover rates, the best proposed models exhibit strong resilience during the COVID-19 period, which proves the profitability of the best proposed models in terms of the downside risks. Additionally, in the static transaction scenario, the absolute capital gain of the value-weighted portfolio is highly reduced, which is caused by over-penalization of the high static transaction cost. However, when considering the dynamic transaction cost with lower rates (20bps), the value-weighted portfolios show higher profitability, particularly during the pandemic period (2112). The softmax signal filter, which restricts the turnover rates, significantly improves the profitability of the models in the dynamic transaction cost scenarios, especially for value-weighted methods.

Some open questions remain for future researchers. For example, when employing the observable factors for the factor model, it may have omitted factor issues, which cause omitted variable bias or missing factor problems, although Transformer models already mitigate these issues via autoencoder structure and self-attention mechanisms to some extent. Namely, the latent factors employed in this chapter are completely derived from the observable original factors. Future researchers can attempt alternative models from the Transformer model family for extracting factors directly from the returns, for example, generative Transformer models such as generative pre-trained Transformer (GPT). Furthermore, although we study Transformer models on large capital stocks, we find that the proposed models are excellent at coping with extreme fluctuations, and higher volatility appears to enhance their performance.

This implies that Transformer models are extra sensitive to higher variations, namely, models perform better with highly volatile assets. In that sense, it is possibly worth trying the proposed models on small capital stocks, emerging markets or cryptocurrencies, which have high volatility. Also, the firm characteristic-sorted portfolios as factors increase the economic meanings to a degree, but they sacrifice some of the information originally contained in the firm characteristics. Researchers may explore innovative factor-organizing methods

which balance information completeness and economic explanations. Finally, and no less importantly, researchers can further explore the LLM-based portfolio optimization or asset allocation technique as employed in Ma et al. (2023).

General Conclusions and Future Work

This thesis attempts to develop research on empirical asset pricing in advanced machine learning (ML) approaches, specifically, advanced neural networks. It applies the dynamic MLP models, RNN-based attention models and the prevailing large language models (LLMs), Transformers, to address the ‘dimensional disaster’ issue caused by the ‘factor zoo’ (Harvey et al., 2016). The main chapters apply the neural network models as the empirical asset pricing factor model framework on 420 top large-cap U.S. stocks, which represent 21% of the entire NASDAQ and NYSE markets. They employ the firm characteristics-sorted portfolio factors provided by Andrew Y.; Zimmermann (2020) after filtering out the factors with unacceptably high missing values to maintain the data quality. Apart from Chapter 2, the main chapters are examined in three testing periods, which cover the Pre-COVID-19 Period (1911), the COVID-19-Inclusive Period (2112) and the Period Including COVID-19 and One-Year After (2212) for the models’ predictability and backtesting robustness check, and empirical asset-pricing evaluation. Period 1911 includes a mild uptrend, Period 2112 contains a sharp uptrend with a deep drawdown, and Period 2212 has a sideways movement with extreme fluctuations. Due to the GKX2020 models, as benchmark models of Chapter 2, data availability, Chapter 2 only examines the model in two testing periods, namely 1911 and 2112, to show how models perform in extreme market turbulence.

Concretely, Chapter 2 utilizes a dynamic pyramidal MLP structure to address the ‘factor zoo’ problem and revisits the GKX MLP models in the COVID-19 pandemic period. The findings of this chapter are from two angles: stock pricing and stock returns forecasting. From the stock pricing perspective, the significant and unpatterned out-of-sample (OOS) α of the best proposed models in both periods implies the potential model misspecification

issue and the omitted risk factors issue. In other words, the best proposed model with the highest OOS R^2 , which is the proposed MLP model with 2 hidden layers, has the relatively stable α in both market regimes, which indicates the sustained existing abnormal returns or omitted factors. From the stock return forecasting perspective, the best model (fw2) achieves the highest OOS R^2 , which is 3.36% in Period 1911 and 4.45% in Period 2112, significantly outperforming GKX2020 MLP models and traditional statistical models. Interestingly, the proposed MLP models perform noticeably better in the COVID-19-Inclusive Period (2112) than the Pre-COVID-19 Period (1911). This phenomenon corroborates the findings of Avramov et al. (2021) that traditional machine learning methods are notably effective at capturing patterns, especially under extreme market conditions. These models are also back-tested for the trading strategy in both equal-weighted and value-weighted portfolios. Although in both portfolios under the static transaction cost method, the absolute returns of the best proposed model are significantly lower than the buy-and-hold (BHE), deep MLP models and GKX2020 MLP models, when considering the volatility risks, the best proposed model outperforms all alternatives in the equal-weighted portfolios and value-weighted portfolios during Period 2112 according to the Sharpe ratio (SR) and Sortino ratio (SO), but not in Period 1911. Therefore, the meaning of the MLP models for building trading strategies lies more in hedging downside risk than in focusing on absolute capital gains, and they provide a considerable advantage during extreme market turbulence. However, the biggest disadvantage of MLP models for empirical asset pricing and asset return forecasting is their assumption of temporal recognition. Namely, no mechanism design in MLP models can naturally capture sequential or temporal relations between observations. Moreover, deeper MLP models show the model overfitting trend, which indicates that consideration of the trade-off between model complexity and prediction accuracy is necessary in the context of empirical asset pricing and asset return forecasting. Additionally, Chapter 2 also finds that deep MLP models and GKX2020 models are less efficient in detecting the sign-based trading signals, hence their annualized returns are convergent to the buy-and-hold strategy. Additionally, the factor importance (variable importance or VI) of the proposed MLP models varies with the development of the COVID-19 pandemic. In the proposed MLP models with the best predictive power (fw2 and fw3) prefer the information related fac-

tors, such as Announcement Return (AnnouncementReturn) and Earnings Forecast Disparity (Earnings-ForecastDisparity) as well as technical analysis indicators such as Short-term reversal (REV6), in the pre-COVID-19 period (1911) and the COVID-19-Inclusive Period (2112), which contain the main uptrend either mild or steep. However, during the Period Including COVID-19 and One-Year After (2212), size becomes the most prominent factor, which is followed by Short-term reversal (REV6) and payout-related factors. Thus, when the market experiences high volatility sideways fluctuations.

Chapter 3 resolves the temporal recognition issue of MLP models by replacing the MLP structure with the recurrent neural network (RNN) bases. Specifically, it employs an MLP autoencoder as the pre-trained method for reducing the input factors' dimensionality, compressing them to 70% of the original size to prevent latent model overfitting. The pre-trained factors, features in the context of machine learning, are subsequently fed into the standard RNN structure and the attention layers built upon the RNN structure for empirical asset pricing and asset return forecasting assignment. The RNN structure captures the temporal dependency between observations, but due to the gradient explosion or vanishing, the vanilla RNN structure encounters difficulties in modelling the long-term dependencies. And its variations, the gated recurrent unit (GRU) RNN model and the long-short term memory (LSTM) RNN model, are not absolutely suitable for the temporal sparsity data, such as asset pricing data, since the complexity of the structure causes the model to overfit. Thus, the attention mechanisms are an alternative solution for complementing the 'short-sighted' imperfections of the RNN structure. Thereby, Chapter 3 investigates prevailing attention mechanisms, such as additive attention (Bahdanau et al., 2015), Luong's attentions (Luong et al., 2015), global self-attention (Vaswani et al., 2017) and sliding window sparse attention (Beltagy et al., 2020), with a 2-layer standard RNN foundation as the factor model framework for asset pricing and asset return forecasting. Among all these pre-trained RNN attention models, the pre-trained RNN Luong's dot product (L-DotProd) attention model and the pre-trained RNN global self-attention (self_att) model are the models that are first applied in the empirical asset pricing domain. Moreover, the pre-trained RNN sliding window sparse attention (sparse_att) model is a newly proposed model, which is first applied in the empir-

ical asset pricing context as well. The empirical results show that OOS α s for all models, including the benchmark GRU and LSTM, vary in a clear pattern across market regimes. Concretely, apart from the vanilla RNN benchmark model, they notably increase in Period 2112, which contains a sharp uptrend with large fluctuations, and recover in Period 2212, which contains sideways movement with large fluctuations. This indicates the existence of abnormal returns beyond factors, namely, the stock excess returns cannot be fully explained by the selected factors, but it also implies the existence of omitted factors. However, the OOS α patterns across the three regimes exclude the possibility of model misspecification. From the stock return forecasting perspective, the vanilla RNN model achieves the highest OOS R^2 across all regimes, and even higher in the periods with extreme market conditions, which is followed by the proposed self_att model and sparse_att model. This indicates that the aforementioned three models have stable forecasting performance during the extreme market turbulence. The backtesting results considering the static transaction cost show that in the equal-weighted portfolio, the self_att model and sparse_att model in Period 1911 and 2212 have the highest Sortino ratio, but in Period 2112, the vanilla RNN model outperforms all alternatives. However, the Sharpe ratio and Sortino ratio vanilla RNN model achieves this at the cost of the absolute capital gain. In value-weighted portfolios, they outperform alternatives in all regimes. Overall, the proposed sparse_att model performs more stably than the self_att model when the stock market capitalization condition changes. These results remain in the turnover rate-based dynamic transaction cost conditions.

Nevertheless, the work in Chapter 2 and Chapter 3 does not consider the relationship between the excess returns of individual stocks. These models only measure the relationship between factors and returns for each stock. Thus, Chapter 4 attempts to jointly evaluate the spatial-temporal dependency between factors and stocks. It employs the architecture of the most prevalent large language model (LLM), the pre-trained Transformer, to capture the spatiotemporal patterns between selected stock returns and factors simultaneously. In other words, it evaluates how factors price and forecast stocks' returns, both temporally and cross-sectionally. Inspired by Kenton and Toutanova (2019), which developed the Bidirectional Encoder Representations from Transformers (BERT) model for natural language

processing (NLP) tasks, Chapter 4 develops the MLP pre-trained Single-directional Encoder Representations from Transformers (SERT) model, which is tailored for modelling numerical time-series data, and also applies the MLP pre-trained standard Transformer model for the context of empirical asset pricing and stock return forecasting. It also investigates the function of attention heads and the importance of the layer normalization process's position in the Transformer models under the context of asset pricing, and shows the meaning of the softmax function, which is the standard Transformer output configuration, in the trading strategy as a trading signal filter. Additionally, the study evaluates the advantages and disadvantages of the MLP autoencoder pre-training method.

As the first empirical asset pricing study on single-direction MLP pre-trained Transformer models, this study demonstrates that complex structures, such as Transformers, can still perform excellently on tasks involving temporal sparsity, for example, asset pricing data. The empirical results of this chapter exhibit that the multi-head attention mechanism in Transformer models improves forecasting accuracy to an insignificant extent in this case, and there is an optimal attention head number in each model. Furthermore, 'layer normalization first', namely, changing the position of the layer normalization process in each module, reduces the prediction accuracy in this case, which is not in accordance with the literature of Transformer models in research with large data size. The patterns of OOS α s for pre-trained and non-pre-trained Transformer models indicate significant abnormal returns, which confirm the presence of market arbitrage opportunities or the presence of omitted factors. No clear evidence implies the model misspecification issue. The proposed encoder-only Transformer frame, SERT, is more in line with the traditional definition of empirical asset pricing since it does not consider the impact of the lagged individual stock excess returns and embeds them as latent factors. However, the standard Transformer models consider the impact of lagged stock returns, hence they could focus more on evaluating the return forecasting instead of the economic interpretability of the factors. Similar to Chapter 3, the causal masks are imposed in the attention layers of the Transformer models to prevent future information leakage. From a forecasting accuracy perspective, the best proposed SERT models have slightly higher OOS R^2 than the best pre-trained Transformer models in all periods.

Moreover, the MLP pre-training process significantly improves the OOS R^2 , especially during the extreme market fluctuations. The backtesting results in the static transaction cost condition show that in equal-weighted portfolios, only the best SERT model outperforms the buy-and-hold benchmark in absolute returns, but with consideration of market risks, especially downside risks, the Sharpe ratio and Sortino ratio show that the best pre-trained standard Transformer model and SERT model outperform the benchmark in all periods. However, in value-weighted portfolios, apart from Period 2112, the best proposed models underperform the buy-and-hold benchmark, especially for absolute returns. The softmax signal filter only removes the difference between models, but it does not necessarily improve the trading performance in the static transaction cost method. However, it significantly improves the models' profitability in dynamic transaction cost configurations.

Comparing across three chapters, from the first generation of neural network models to the most advance Transformer models, the predictive power (according to average OOS R^2) of the best proposed models: fw2 in Chapter 2 (3.36% in 1911 and 4.45% in 2112), self_att (2.8% in 1911, 2.02% in 2112 and 1.64% in 2212) and sparse_att (2.45% in 1911, 1.94% in 2112 and 1.89% in 2212) in Chapter 3, and P_Trans_H3 (7.63% in 1911, 11.13% in 2112 and 9.72% in 2212) and SERT_H7 (8.05% in 1911, 11.94% in 2112 and 10.43% in 2212, varies depends on the periods), it shows that the advanced models have significant higher predictive powers. It is also interesting to note that the models as simple as vanilla RNN as a benchmark model in Chapter 3 achieves the highest OOS R^2 (11.37% in 1911, 14.01% in 2112 and 14.32% in 2212). Nevertheless, the high predictive power of the models never ensures a high profitability, especially considering the transaction costs. In the static transaction method-based (with the conservative deduction rate of 50bps) backtesting, models' performance in Chapter 2 hardly exceeds the buy-and-hold benchmarks in both equal-weighted portfolios and value-weighted portfolios, but after being adjusted by the turnover rate and 20bps deduction rate for large-cap stocks, the deep MLP models outperform the buy-and-hold benchmark, particularly in value-weighted portfolios. Overall, the primary significance of the MLP models lies in their ability to moderate downside risk. Models in Chapter 3 significantly reduce the turnover rate through the model mechanisms. This significantly reduces the impact of

both static and dynamic transaction costs on models' profitability. Apart from the vanilla RNN model as benchmark, all models significantly outperform the buy-and-hold strategy in both portfolios, especially during the COVID-19-Inclusive Period (2112), which proves that the high predictive power of the model does not necessarily translate into high profitability, even with the low turnover rates. Models in Chapter 4 further verify this proposition. Due to the sensitivity of the noise, the profitability of the best proposed models in Chapter 4 does not exceed the buy-and-hold benchmarks in both portfolios, especially in the dynamic transaction cost method. However, they present an excellent capability in hedging downside risks, particularly in the COVID-19-Inclusive Period (2112), according to the Sortino ratios. The softmax signal filter works efficiently on the models presented in Chapter 4, noticeably improving models' profitability and surpassing the buy-and-hold benchmarks in both equal-weighted and value-weighted portfolios. Patterned α s closely follow the market trend in the three periods, indicating the existence of abnormal returns in the market.

Additionally, there are some directions worth exploring in future research. In empirical asset pricing, it could be interesting to explore the structure of anomalies, residual α in the context of machine learning, and learn how to reduce the α with economic or financial meanings instead of violently pursuing the statistical insignificance. It is also interesting to investigate the reasons for temporal variations in factor importance. Furthermore, it could be worth investigating further how macroeconomic factors affect the stock pricing and stock return forecasting, and whether these models can be generalized to a wide range of assets, instead of the 'stocks only', for example, stocks in emerging markets, small-cap stocks or cryptocurrencies, due to the nature of attention mechanisms and the limitations of large-cap U.S. stock-only examinations, although large-cap stocks are more practicable in the real-time trading scenario. Concretely, the global self-attention mechanism, which is the original configuration in the most prevalent LLM, the classic Transformer model, is a short-sighted attention mechanism. This means it is designed with high sensitivity for capturing the variation in the data, which shows as its superiority in predictive power over the data with high volatility. Therefore, it is worth investigating further how this attention mechanism interacts with assets exhibiting high-fluctuation price patterns. In addition, recently developed

attention mechanisms, such as sparse-attention mechanisms, are also worth examining in alternative markets and assets to see how they respond to different volatilities and price patterns over time, with different market macroeconomic shocks, especially in the modern age, where markets have a high proportion of AI traders. Automatic systems can also be designed for searching for the best attention mechanisms across the different types of assets (e.g. exchange rates, commodities) or markets with varying liquidity (e.g. components of the FTSE 100, the CAC 40, Nikkei 225 and CSI 300). Alternatively, applying the newly designed attention assembled models, such as Informer (Zhou et al., 2021b), Performer (Choromanski et al., 2020) and Autoformer (Wu et al., 2021b), for the aforementioned tasks.

As factor investing strategies, it is also worth developing advanced machine learning models for asset allocation tasks as alternatives to simple strategies, such as equal-weighted and value-weighted. To further improve the financial and economic interpretability of these neural network models, the DM test can be applied to individual stocks to investigate the model's preference for an industry or sector, while investigating further on MSE and OOS R^2 for the contribution of characters, for example, which industry sectors have higher predictive power in a certain framework, and what is the impact of market capitalization size. Also, for attention models, attention weights variation over time can be examined to investigate which factors are emphasized during the pandemic period, and conduct subset factors from the factor pool as factor sensitivity analysis. As future work, these advanced neural network models can also be upgraded by altering the specific modules, such as adding an additional MLP structure within the Transformer core structures. Moreover, these structures can be applied to volatility modelling for asset allocation and risk control, as exemplified by Ma et al. (2023).

References

- Md Shad Akhtar, Abhishek Kumar, Deepanway Ghosal, Asif Ekbal, and Pushpak Bhattacharyya. A multilayer perceptron based ensemble technique for fine-grained financial sentiment analysis. In *Proceedings of the 2017 conference on empirical methods in natural language processing*, pages 540–546, 2017.
- Yakov Amihud. Illiquidity and stock returns: cross-section and time-series effects. *Journal of financial markets*, 5(1):31–56, 2002.
- Tom Chen Andrew Y.; Zimmermann. Open source cross-sectional asset pricing. *Centre for Financial Research (CFR), working paper*, 2020.
- Andrew Ang and Geert Bekaert. International asset allocation with regime shifts. *The review of financial studies*, 15(4):1137–1187, 2002.
- Andrew Ang, Robert J Hodrick, Yuhang Xing, and Xiaoyan Zhang. The cross-section of volatility and expected returns. *The journal of finance*, 61(1):259–299, 2006.
- Andrew Ang, Robert J Hodrick, Yuhang Xing, and Xiaoyan Zhang. High idiosyncratic volatility and low returns: International and further us evidence. *Journal of financial economics*, 91(1):1–23, 2009.
- Dogu Araci. Finbert: Financial sentiment analysis with pre-trained language models. *arXiv preprint arXiv:1908.10063*, 2019.
- Doron Avramov, Si Cheng, and Lior Metzker. Machine learning versus economic restrictions: Evidence from stock return predictability. *Available at SSRN 3450322*, 2021.
- Dilek Aydogan-Kilic and A Sevtap Selcuk-Kestel. Modification of hybrid RNN-HMM model in asset pricing: univariate and multivariate cases. *Applied Intelligence*, 53(20):23812–23833, 4 2023. ISSN 15737497. doi: 10.1007/s10489-023-04762-7.
- Seungho Baek, Mina Glambosky, Seok Hee Oh, and Jeong Lee. Machine learning and algorithmic pairs trading in futures markets. *Sustainability (Switzerland)*, 12(17):1–21, 2020. ISSN 20711050. doi: 10.3390/SU12176791.
- Matteo Bagnara. Asset Pricing and Machine Learning: A critical review. *Journal of Economic Surveys*, 2022. ISSN 14676419. doi: 10.1111/joes.12532.
- Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pages 1–15, 2015.
- Malcolm Baker and Jeffrey Wurgler. The equity share in new issues and aggregate stock returns. *the Journal of Finance*, 55(5):2219–2257, 2000.

-
- Malcolm Baker and Jeffrey Wurgler. Investor sentiment and the cross-section of stock returns. *The journal of Finance*, 61(4):1645–1680, 2006.
- Malcolm Baker and Jeffrey Wurgler. Investor sentiment in the stock market. *Journal of economic perspectives*, 21(2):129–151, 2007.
- Ravi Bansal and Amir Yaron. Risks for the long run: A potential resolution of asset pricing puzzles. *The journal of Finance*, 59(4):1481–1509, 2004. ISSN 0022-1082.
- Ravi Bansal, Dana Kiku, and Amir Yaron. An empirical evaluation of the long-run risks model for asset prices. Technical report, National Bureau of Economic Research, 2009.
- Wei Bao, Jun Yue, and Yulei Rao. A deep learning framework for financial time series using stacked autoencoders and long-short term memory. *PloS one*, 12(7):e0180944, 2017a. ISSN 1932-6203.
- Wei Bao, Jun Yue, and Yulei Rao. A deep learning framework for financial time series using stacked autoencoders and long-short term memory. *PloS one*, 12(7):e0180944, 2017b.
- Robert J Barro. Rare disasters and asset markets in the twentieth century. *The Quarterly Journal of Economics*, 121(3):823–866, 2006.
- Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.
- Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.
- Kristoffer Pons Bertelsen. The Prior Adaptive Group Lasso and the Factor Zoo. 2022.
- Daniele Bianchi, Matthias Büchner, and Andrea Tamoni. Bond risk premiums with machine learning. *The Review of Financial Studies*, 34(2):1046–1089, 2021. ISSN 0893-9454.
- Bartosz Bieganowski and Robert Slepaczuk. Supervised autoencoder mlp for financial time series forecasting. *arXiv preprint arXiv:2404.01866*, 2024.
- Douglas T Breeden. An intertemporal asset pricing model with stochastic consumption and investment opportunities. *Journal of financial Economics*, 7(3):265–296, 1979.
- Michael J Brennan, Tarun Chordia, and Avanidhar Subrahmanyam. Alternative factor specifications, security characteristics, and the cross-section of expected stock returns. *Journal of financial Economics*, 49(3):345–373, 1998.
- Svetlana Bryzgalova, Markus Pelger, and Jason Zhu. Forest through the trees: Building cross-sections of stock returns. *The Journal of Finance*, 2019.
- John Y Campbell and John H Cochrane. By force of habit: A consumption-based explanation of aggregate stock market behavior. *Journal of political Economy*, 107(2):205–251, 1999.
- Mark M Carhart. On persistence in mutual fund performance. *Journal of Finance*, 52(1):57–82, 1997. ISSN 00221082. doi: 10.1111/j.1540-6261.1997.tb03808.x.
- Victor Chang, Xiaowen Man, Qianwen Xu, and Ching Hsien Hsu. Pairs trading on different portfolios based on machine learning. *Expert Systems*, 38(3):1–25, 2021. ISSN 14680394. doi: 10.1111/exsy.12649.

- Philippe Chatigny, Ruslan Goyenko, and Chengyu Zhang. Asset pricing with attention guided deep learning. *Available at SSRN 3971876*, 2021.
- Luyang Chen, Markus Pelger, and Jason Zhu. Deep Learning in Asset Pricing. *Management Science*, 70(2):714–750, 4 2024. ISSN 15265501. doi: 10.1287/mnsc.2023.4695.
- Yuhan Cheng and Ke Tang. GPT’s idea of stock factors. *Quantitative Finance*, pages 1–26, 2024. ISSN 1469-7688.
- Alex Chinco, Adam D Clark-Joseph, and Mao Ye. Sparse signals in the cross-section of returns. *The Journal of Finance*, 74(1):449–492, 2019. ISSN 0022-1082.
- Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, pages 1724–1734, 2014. ISBN 9781937284961. doi: 10.3115/v1/d14-1179.
- Krzysztof Choromanski, Valerii Likhoshesterov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Re-thinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020.
- K Clark. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*, 2020.
- Lin William Cong, Ke Tang, Jingyuan Wang, and Yang Zhang. AlphaPortfolio: Direct construction through deep reinforcement learning and interpretable AI. *Available at SSRN*, 3554486, 2021a.
- Lin William Cong, Ke Tang, Jingyuan Wang, and Yang Zhang. Deep sequence modeling: Development and applications in asset pricing. *arXiv preprint arXiv:2108.08999*, 2021b.
- Guillaume Coqueret and Tony Guida. *Machine learning for factor investing: R version*. CRC Press, Boca Raton, FL, 2020. ISBN 1000176762.
- Kent Daniel and Sheridan Titman. Evidence on the characteristics of cross sectional variation in stock returns. *the Journal of Finance*, 52(1):1–33, 1997.
- Antoine Didisheim, Shikun Barry Ke, Bryan T Kelly, and Semyon Malamud. APT or “AIPT”? the surprising dominance of large factor models, 2024.
- Francis X Diebold and Roberto S Mariano. Comparing predictive accuracy. *Journal of Business and Economic Statistics*, 13(3):253–263, 1995.
- Karl B Diether, Christopher J Malloy, and Anna Scherbina. Differences of opinion and the cross section of stock returns. *The journal of finance*, 57(5):2113–2141, 2002.
- Matthew Dixon. Sequence classification of the limit order book using recurrent neural networks. *Journal of computational science*, 24:277–286, 2018.
- Matthew Dixon and Justin London. Financial forecasting with α -rnns: A time series modeling approach. *Frontiers in Applied Mathematics and Statistics*, 6:551138, 2021.

- Rian Dolphin, Joe Dursun, Jonathan Chow, Jarrett Blankenship, Katie Adams, and Quinton Pike. Extracting structured insights from financial news: An augmented llm driven approach. *arXiv preprint arXiv:2407.15788*, 2024.
- Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.
- Hasan Fallahgoul. Beyond black-box ai: A theory of interpretable transformers for asset pricing. *Available at SSRN*, 2025.
- Eugene F Fama. Efficient capital markets. *Journal of finance*, 25(2):383–417, 1970.
- Eugene F Fama and Kenneth R French. Common risk factors in the returns on stocks and bonds. *Journal of Financial Economics*, 33(1):3–56, 1993. ISSN 0304405X. doi: 10.1016/0304-405X(93)90023-5.
- Eugene F Fama and Kenneth R French. Financing decisions: who issues stock? *Journal of financial economics*, 76(3):549–582, 2005.
- Eugene F Fama and Kenneth R French. A five-factor asset pricing model. *Journal of Financial Economics*, 116(1):1–22, 2015. ISSN 0304405X. doi: 10.1016/j.jfineco.2014.10.010. URL <http://dx.doi.org/10.1016/j.jfineco.2014.10.010>.
- Guanhao Feng, Stefano Giglio, and Dacheng Xiu. Taming the Factor Zoo: A Test of New Factors. *Journal of Finance*, 75(3):1327–1370, 4 2020. ISSN 15406261. doi: 10.1111/jofi.12883.
- Mst Zannatun Ferdus, Nishat Anjum, Tuan Ngoc Nguyen, Afjal Hossain Jisan, and Md Azad Hossain Raju. The Influence of Social Media on Stock Market: A Transformer-Based Stock Price Forecasting with External Factors. *Journal of Computer Science and Technology Studies*, 6(1):189–194, 2024.
- Thomas ; Fischer and Christopher Krauss. Deep learning with long short-term memory networks for financial market predictions. In *FAU Discussion Papers in Economics*, page No. 11/2017, 2017. URL <http://hdl.handle.net/10419/157808www.econstor.eu>.
- Andrea Frazzini, Ronen Israel, and Tobias J Moskowitz. Trading costs. *Available at SSRN 3229719*, 2018.
- Joachim Freyberger, Andreas Neuhierl, and Michael Weber. Dissecting characteristics non-parametrically. *The Review of Financial Studies*, 33(5):2326–2377, 2020. ISSN 0893-9454.
- Xavier Gabaix. Variable rare disasters: An exactly solved framework for ten puzzles in macro-finance. *The Quarterly journal of economics*, 127(2):645–700, 2012. ISSN 0033-5533.
- Xavier Gabaix, Ralph S J Koijen, Robert Richmond, and Motohiro Yogo. Asset embeddings. *Available at SSRN 4507511*, 2024.
- Krisztian Gado. Transformers, Central Banker Speeches, Tail Stock Returns. *Central Banker Speeches, Tail Stock Returns (September 07, 2024)*, 2024.
- Manisha Galphade, VB Nikam, Biplab Banerjee, and Arvind W Kiwelekar. Comparative analysis of wind power forecasting using lstm, bilstm, and gru. In *International Conference on Frontiers of Intelligent Computing: Theory and Applications*, pages 483–493. Springer, 2022.

- Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. *Neural computation*, 12(10):2451–2471, 2000.
- Abdul Haluk Batur Gezici and Emre Sefer. Deep Transformer-Based Asset Price and Direction Prediction. *IEEE Access*, 12:24164–24178, 2024a. ISSN 21693536. doi: 10.1109/ACCESS.2024.3358452.
- Abdul Haluk Batur Gezici and Emre Sefer. Deep transformer-based asset price and direction prediction. *IEEE Access*, 12:24164–24178, 2024b.
- Oliver Giesecke. State Government Debt Structure. *Available at SSRN 4672561*, 2023.
- Oliver Giesecke. AI at the frontier of economic research. *Available at SSRN*, 2024.
- Stefano Giglio and Dacheng Xiu. Asset pricing with omitted factors. *Journal of Political Economy*, 129(7):1947–1990, 2021. ISSN 0022-3808.
- Stefano Giglio, Bryan Kelly, and Dacheng Xiu. Factor Models, Machine Learning, and Asset Pricing. *Annual Review of Financial Economics*, 14, 2022. doi: 10.1146/annurev-financial-101521. URL <https://doi.org/10.1146/annurev-financial-101521->.
- Xavier Glorot and Yoshua Bengio. Understanding the Difficulty of Training Deep Feedforward Neural Networks. In Yee Whye Teh and Mike Titterton, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR. URL <https://proceedings.mlr.press/v9/glorot10a.html>.
- Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, Cambridge, Massachusetts, 2016. ISBN 978-0-262-03561-3.
- Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.
- Jeremiah Green, John R M Hand, and X Frank Zhang. The characteristics that provide independent information about average u.s. monthly stock returns. *Review of Financial Studies*, 30(12):4389–4436, 2017. ISSN 14657368. doi: 10.1093/rfs/hhx019.
- Bruce D Grundy and J Spencer Martin Martin. Understanding the nature of the risks and the source of the rewards to momentum investing. *The Review of Financial Studies*, 14(1):29–78, 2001.
- Shihao Gu, Bryan Kelly, and Dacheng Xiu. Empirical Asset Pricing via Machine Learning. *Review of Financial Studies*, 33(5):2223–2273, 4 2020. ISSN 14657368. doi: 10.1093/rfs/hhaa009.
- Shihao Gu, Bryan Kelly, and Dacheng Xiu. Autoencoder asset pricing models. *Journal of Econometrics*, 222(1):429–450, 2021. ISSN 0304-4076.
- Lars Peter Hansen, John C Heaton, and Nan Li. Consumption strikes back? measuring long-run risk. *Journal of Political economy*, 116(2):260–302, 2008.

-
- Campbell R Harvey and Yan Liu. A census of the factor zoo. *Available at SSRN 3341728*, 2019.
- Campbell R Harvey, Yan Liu, and Heqing Zhu. ... and the Cross-Section of Expected Returns. *Review of Financial Studies*, 29(1):5–68, 2016. ISSN 14657368. doi: 10.1093/rfs/hhv059.
- J B Heaton, Nicholas G Polson, and Jan Hendrik Witte. Deep learning in finance. *arXiv preprint arXiv:1602.06561*, 2016.
- James B Heaton, Nick G Polson, and Jan Hendrik Witte. Deep learning for finance: deep portfolios. *Applied Stochastic Models in Business and Industry*, 33(1):3–12, 2017.
- Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- Thomas Hollis, Antoine Viscardi, and Seung Eun Yi. A comparison of lstms and attention mechanisms for forecasting financial time series. *arXiv preprint arXiv:1812.07699*, 2018.
- Kewei Hou, Chen Xue, and Lu Zhang. Digesting anomalies: An investment approach. *Review of Financial Studies*, 28(3):650–705, 4 2015. ISSN 14657368. doi: 10.1093/rfs/hhu068.
- Kewei Hou, Haitao Mo, Chen Xue, and Lu Zhang. Which factors? *Review of Finance*, 23(1):1–35, 2019. ISSN 1572-3097.
- Kewei Hou, Chen Xue, and Lu Zhang. Replicating anomalies. *The Review of financial studies*, 33(5):2019–2133, 2020. ISSN 0893-9454.
- James M Hutchinson, Andrew W Lo, and Tomaso Poggio. A nonparametric approach to pricing and hedging derivative securities via learning networks. *The journal of Finance*, 49(3):851–889, 1994. ISSN 0022-1082.
- Narasimhan Jegadeesh and Sheridan Titman. Returns to buying winners and selling losers: Implications for stock market efficiency. *The Journal of Finance*, 48(1):65–91, 1993.
- Timothy C Johnson. Forecast dispersion and the cross section of expected returns. *The Journal of Finance*, 59(5):1957–1978, 2004.
- Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. An empirical exploration of recurrent network architectures. In *International conference on machine learning*, pages 2342–2350. PMLR, 2015.
- Himmet Kaplan, Ralf-Peter Mundani, Heiko Rölke, and Albert Weichselbraun. CrudeBERT: applying economic theory towards fine-tuning transformer-based sentiment analysis models to the crude oil market. *arXiv preprint arXiv:2305.06140*, 2023.
- Bryan Kelly and Seth Pruitt. Market expectations in the cross-section of present values. *The Journal of Finance*, 68(5):1721–1756, 2013. ISSN 0022-1082.
- Bryan Kelly and Dacheng Xiu. Financial machine learning. *Foundations and Trends® in Finance*, 13(3-4):205–363, 2023. ISSN 1567-2395.
- Bryan T Kelly, Seth Pruitt, and Yinan Su. Characteristics are covariances: A unified model of risk and return. *Journal of Financial Economics*, 134(3):501–524, 4 2019. ISSN 0304405X. doi: 10.1016/j.jfineco.2019.05.001.

- Bryan T Kelly, Boris Kuznetsov, Semyon Malamud, and Teng Andrea Xu. Artificial intelligence asset pricing models. Technical report, National Bureau of Economic Research, 2025.
- Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of naacL-HLT*, volume 1, USA, 2019. Minneapolis, Minnesota.
- Taewook Kim, Ha Young Kim, and Benjamin M Tabak. Optimizing the Pairs-Trading Strategy Using Deep Reinforcement Learning with Trading and Stop-Loss Boundaries. *Complexity*, 2019, 2019. ISSN 10990526. doi: 10.1155/2019/3582516.
- Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- Kamesh Korangi, Christophe Mues, and Cristián Bravo. A transformer-based model for default prediction in mid-cap corporate markets. *European Journal of Operational Research*, 308(1):306–320, 4 2023. ISSN 03772217. doi: 10.1016/j.ejor.2022.10.032.
- David A Lesmond, Michael J Schill, and Chunsheng Zhou. The illusory nature of momentum profits. *Journal of financial economics*, 71(2):349–380, 2004.
- Martin Lettau and Sydney Ludvigson. Resurrecting the (C) CAPM: A cross-sectional test when risk premia are time-varying. *Journal of political economy*, 109(6):1238–1287, 2001a. ISSN 0022-3808.
- Martin Lettau and Sydney Ludvigson. Consumption, aggregate wealth, and expected stock returns. *the Journal of Finance*, 56(3):815–849, 2001b.
- Edmond Lezmi and Jiali Xu. Time series forecasting with transformer models and application to asset management. *Available at SSRN 4375798*, 2023.
- Hao Li, Yanyan Shen, and Yanmin Zhu. Stock price prediction using attention-based multi-input lstm. In *Proceedings of the Asian Conference on Machine Learning*, pages 454–469, Beijing, China, 2018. PMLR. ISBN 2640-3498.
- Yawei Li, Shuqi Lv, Xinghua Liu, and Qiuyue Zhang. Incorporating Transformers and Attention Networks for Stock Movement Prediction. *Complexity*, 2022, 2022. ISSN 10990526. doi: 10.1155/2022/7739087.
- John Lintner. Security prices, risk, and maximal gains from diversification. *The journal of finance*, 20(4):587–615, 1965.
- John Lintner. The valuation of risk assets and the selection of risky investments in stock portfolios and capital budgets: A reply. *The review of economics and statistics*, pages 222–224, 1969.
- Roderick J. A. Little and Donald B. Rubin. *Statistical Analysis with Missing Data*. John Wiley & Sons, Hoboken, New Jersey, 3rd edition, 2019. ISBN 978-1-119-48225-3.
- Liyuan Liu, Xiaodong Liu, Jianfeng Gao, Weizhu Chen, and Jiawei Han. Understanding the difficulty of training transformers. *arXiv preprint arXiv:2004.08249*, 2020a.

- Yang Liu, Liang Yan, Sheng Liu, Ting Jiang, Feng Zhang, Yu Wang, and Shengnan Wu. Enhancing input parameter estimation by machine learning for the simulation of large-scale logistics networks. In *Proceedings of the Winter Simulation Conference (WSC)*, volume 2020, pages 608–619, Online, December 2020b. Institute of Electrical and Electronics Engineers Inc. ISBN 9781728194998. doi: 10.1109/WSC48552.2020.9383942.
- Yeqi Liu, Chuanyang Gong, Ling Yang, and Yingyi Chen. DSTP-RNN: A dual-stage two-phase attention-based recurrent neural network for long-term and multivariate time series prediction. *Expert Systems with Applications*, 143:113082, 2020c. ISSN 09574174. doi: 10.1016/j.eswa.2019.113082. URL <https://doi.org/10.1016/j.eswa.2019.113082>.
- Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.
- Tian Ma, Wanwan Wang, and Yu Chen. Attention is all you need: An interpretable transformer-based asset allocation approach. *International Review of Financial Analysis*, 90, 4 2023. ISSN 10575219. doi: 10.1016/j.irfa.2023.102876.
- Harry Markowitz. Portfolio selection. *The Journal of Finance*, 7(1):77–91, March 1952. doi: 10.2307/2975974.
- William F Massy. Principal components regression in exploratory statistical research. *Journal of the American Statistical Association*, 60(309):234–256, 1965.
- Timothy Masters. *Practical Neural Network Recipes in C++*. Academic Press Professional, Inc., San Diego, CA, USA, 1993. ISBN 0-12-479040-2. Includes diskette with source code.
- Deborah Miori and Constantin Petrov. Narratives from GPT-derived networks of news and a link to financial markets dislocations. *International Journal of Data Science and Analytics*, pages 1–25, 2024. ISSN 2364-415X.
- Frederic S Mishkin. How big a problem is too big to fail? A review of Gary Stern and Ron Feldman’s too big to fail: the hazards of bank bailouts. *Journal of economic literature*, 44(4):988–1004, 2006. ISSN 0022-0515.
- Tom M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.
- Jan Mossin. Equilibrium in a capital asset market. *Econometrica: Journal of the econometric society*, pages 768–783, 1966.
- Stefan Nagel. Machine learning in asset pricing, 2021.
- Yuqi Nie, Yaxuan Kong, Xiaowen Dong, John M Mulvey, H Vincent Poor, Qingsong Wen, and Stefan Zohren. A survey of large language models for financial applications: Progress, prospects and challenges. *arXiv preprint arXiv:2406.11903*, 2024.
- Robert Novy-Marx. The other side of value: The gross profitability premium. *Journal of Financial Economics*, 108(1):1–28, 2013. ISSN 0304405X. doi: 10.1016/j.jfineco.2013.01.003. URL <http://dx.doi.org/10.1016/j.jfineco.2013.01.003>.
- Robert Novy-Marx and Mihail Velikov. A taxonomy of anomalies and their trading costs. *The Review of Financial Studies*, 29(1):104–147, 2016.

- Markus Päivärinta. Transformer-based deep learning model for stock return forecasting: Empirical evidence. 2022.
- L'uboš Pástor and Robert F Stambaugh. Liquidity risk and expected stock returns. *Journal of Political economy*, 111(3):642–685, 2003.
- Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.
- Jieyang Peng, Andreas Kimmig, Jiahai Wang, Xiufeng Liu, Zhibin Niu, and Jivka Ovtcharova. Dual-stage attention-based long-short-term memory neural networks for energy demand prediction. *Energy and Buildings*, 249, 4 2021. ISSN 03787788. doi: 10.1016/j.enbuild.2021.111211.
- Anton Pudikov and Alexander Brovko. Comparison of lstm and gru recurrent neural network architectures. In *International Scientific and Practical Conference in Control Engineering and Decision Making*, pages 114–124. Springer, 2020.
- T Qian, A Xie, and C Bruckmann. Sensitivity Analysis on Transferred Neural Architectures of BERT and GPT-2 for Financial Sentiment Analysis. arXiv 2022. *arXiv preprint arXiv:2207.03037*.
- Yao Qin, Dongjin Song, Haifeng Chen, Wei Cheng, Guofei Jiang, and Garrison Cottrell. A dual-stage attention-based recurrent neural network for time series prediction. *arXiv preprint arXiv:1704.02971*, 2017.
- Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020. ISSN 1533-7928.
- Tohida Rehman, Raghubir Bose, Samiran Chattopadhyay, and Debarshi Kumar Sanyal. Transfer learning and transformer architecture for financial sentiment analysis. In *Proceedings of the International Conference on Computational Intelligence, Data Science and Cloud Computing (IEM-ICDC 2021)*, pages 17–27, Kolkata, India, 2022. Springer.
- Thomas A Rietz. The equity risk premium a solution. *Journal of monetary Economics*, 22(1):117–131, 1988.
- Maryan Rizinski, Hristijan Peshov, Kostadin Mishev, Milos Jovanovik, and Dimitar Trajanov. Sentiment Analysis in Finance: From Transformers Back to eXplainable Lexicons (XLex). *IEEE Access*, 2024. ISSN 2169-3536.
- Frank Rosenbaltt. The perceptron—a perceiving and recognizing automation. *Cornell Aeronautical Laboratory*, 1957.
- Stephen A Ross. The current status of the capital asset pricing model (CAPM). *The Journal of Finance*, 33(3):885–901, 1978. ISSN 0022-1082.

- Putri Utami Rukmana, Asriana, and Hanif Fakhurroja. Application of LSTM, RNN, and Transformer in stock price prediction of information technology companies: A comparative analysis. In *2024 International Conference on Computer, Control, Informatics and its Applications (IC3INA)*, pages 249–254, Jakarta, Indonesia, 2024. Institute of Electrical and Electronics Engineers Inc. doi: 10.1109/IC3INA64086.2024.10732754.
- David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986a. ISSN 0028-0836.
- David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986b.
- Hima Keerthi Sagiraju and Shashi Mogalla. Application of multilayer perceptron to deep reinforcement learning for stock market trading and analysis. *Indonesian Journal of Electrical Engineering and Computer Science*, 24(3):1759–1771, 2021.
- Simão Moraes Sarmiento and Nuno Horta. Enhancing a Pairs Trading strategy with the application of Machine Learning. *Expert Systems with Applications journal*, 158, 2020. doi: 10.1016/j.eswa.2020.113490.
- Mostafa Shabani, Dat Thanh Tran, Martin Magris, Juho Kanninen, and Alexandros Iosifidis. Multi-head temporal attention-augmented bilinear network for financial time series prediction. In *2022 30th European Signal Processing Conference (EUSIPCO)*, pages 1487–1491. IEEE, 2022.
- Edward Sharkey and Philip Treleaven. BERT vs GPT for financial engineering. *arXiv preprint arXiv:2405.12990*, 2024.
- William F Sharpe. Capital asset prices: A theory of market equilibrium under conditions of risk. *The journal of finance*, 19(3):425–442, 1964.
- Justin Sirignano and Rama Cont. Universal features of price formation in financial markets: perspectives from deep learning. In *Machine learning and AI in finance*, pages 5–15. Routledge, 2021.
- James H Stock and Mark W Watson. Forecasting using principal components from a large number of predictors. *Journal of the American statistical association*, 97(460):1167–1179, 2002. ISSN 0162-1459.
- Jifeng Sun, Wentao Fu, Jianwu Lin, Yong Jiang, and Shudao Xia. Deep portfolio optimization modeling based on conv-transformers with graph attention mechanism. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, volume 2022-July, Padua, Italy, 2022. Institute of Electrical and Electronics Engineers Inc. ISBN 9781728186719. doi: 10.1109/IJCNN55064.2022.9892317.
- Vaishnavi Tevare and P. S. Revankar. Forecasting stock prices with stack transformer. In *Proceedings of the International Conference on Circuit Power and Computing Technologies (ICCPCT 2023)*, pages 1262–1269, Kollam, India, 2023. Institute of Electrical and Electronics Engineers Inc. ISBN 9798350333244. doi: 10.1109/ICCPCT58313.2023.10245652.
- James Tobin. A general equilibrium approach to monetary theory. *Journal of money, credit and banking*, 1(1):15–29, 1969. ISSN 0022-2879.

-
- Anne M Treisman and Garry Gelade. A feature-integration theory of attention. *Cognitive psychology*, 12(1):97–136, 1980.
- Jack L Treynor. Market value, time, and risk. *Time, and Risk (August 8, 1961)*, 1961.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
- Jie Wang, Jun Wang, Wen Fang, and Hongli Niu. Financial time series prediction using elman recurrent random neural networks. *Computational intelligence and neuroscience*, 2016(1):4742515, 2016.
- Qiyu Wang. Cryptocurrencies asset pricing via machine learning. *International Journal of Data Science and Analytics*, 12(2):175–183, 2021. ISSN 23644168. doi: 10.1007/s41060-021-00252-6. URL <https://doi.org/10.1007/s41060-021-00252-6>.
- Zhiguang Wang. Machine learning for stock return prediction: Transformers or simple neural networks. *Finance Research Letters*, page 108783, 2025.
- Ziao Wang, Yuhang Li, Junda Wu, Jaehyeon Soon, and Xiaofeng Zhang. Finvis-gpt: A multi-modal large language model for financial chart analysis. *arXiv preprint arXiv:2308.01430*, 2023.
- Ivo Welch and Amit Goyal. A comprehensive look at the empirical performance of equity premium prediction. *The Review of Financial Studies*, 21(4):1455–1508, 2008. ISSN 0893-9454.
- Paul Werbos. Beyond regression: New tools for prediction and analysis in the behavioral sciences. *PhD thesis, Committee on Applied Mathematics, Harvard University, Cambridge, MA*, 1974.
- Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. *arXiv preprint arXiv:1410.3916*, 2014.
- Aaron Wheeler and Jeffrey D Varner. MarketGPT: Developing a Pre-trained transformer (GPT) for Modeling Financial Time Series. *arXiv preprint arXiv:2411.16585*, 2024.
- Herman Wold. Estimation of principal components and related models by iterative least squares. *Multivariate analysis*, pages 391–420, 1966.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, et al. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45, 2020.
- Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in neural information processing systems*, 34:22419–22430, 2021a.
- Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in neural information processing systems*, 34:22419–22430, 2021b.

- Kuo Yang, Yugui Tang, Shujing Zhang, and Zhen Zhang. A deep learning approach to state of charge estimation of lithium-ion batteries based on dual-stage attention mechanism. *Energy*, 244, 4 2022. ISSN 03605442. doi: 10.1016/j.energy.2022.123233.
- Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 1480–1489, 2016.
- Junyi Ye, Bhaskar Goswami, Jingyi Gu, Ajim Uddin, and Guiling Wang. From factor models to deep learning: Machine learning in reshaping empirical asset pricing. *arXiv preprint arXiv:2403.06779*, 2024.
- Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 68(1):49–67, 2006. ISSN 1369-7412.
- Chen Zhang. Asset pricing and deep learning. *arXiv preprint arXiv:2209.12014*, 2022.
- Cheng Zhang, Nilam Nur Amir Sjarif, and Roslina Ibrahim. Deep learning models for price forecasting of financial time series: A review of recent advancements: 2020–2022, 4 2024. ISSN 19424795.
- Qiuyue Zhang, Chao Qin, Yunfeng Zhang, Fangxun Bao, Caiming Zhang, and Peide Liu. Transformer-based attention network for stock movement prediction. *Expert Systems with Applications*, 202:117239, 2022. ISSN 0957-4174.
- Xuan Zhang, Xun Liang, Aakas Zhiyuli, Shusen Zhang, Rui Xu, and Bo Wu. At-lstm: An attention-based lstm model for financial time series prediction. In *IOP Conference Series: Materials Science and Engineering*, volume 569, page 052037. IOP Publishing, 2019. ISBN 1757-8981.
- Zequan Zhao. Application and challenges of informer model in financial time series prediction: A review. *Applied and Computational Engineering*, 38:90–95, 2024. ISSN 2755-273X.
- Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 11106–11115, 2021a. ISBN 2374-3468.
- Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 11106–11115, 2021b.
- Xi Zhou and Yilun Wang. Learning from memory: Asset pricing via recurrent neural network and attention mechanism. *Available at SSRN 4999011*, 2024.

Appendices

Appendix A: Adam for Stochastic Gradient Descent (SGD)

Algorithm 1 Adam for Stochastic Gradient Descent (SGD)

- 1: Initialize the parameter vector θ_0 . Set $m_0 = 0$, $v_0 = 0$, $t = 0$.
- 2: **while** θ_t not converged **do**
- 3: $t \leftarrow t + 1$
- 4: $g_t \leftarrow \nabla_{\theta} \mathcal{L}(\theta; \cdot) |_{\theta=\theta_{t-1}}$
- 5: $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$
- 6: $v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t \odot g_t^a$
- 7: $\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$
- 8: $\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$
- 9: $\theta_t \leftarrow \theta_{t-1} - \alpha \hat{m}_t \oslash (\sqrt{\hat{v}_t} + \epsilon)$
- 10: **end while**
- 11: **Result:** The final parameter estimate is θ_t .

^a $\odot$ and $\oslash$ denote element-wise multiplication and division, respectively.

Appendix B: Mechanism of Backpropagation

The context shows the backpropagation mechanism of a simple MLP. Concretely, when the results are calculated from Equation (1) and Equation (2), we derive the loss from the forward round. Here, $W^{(l)}$ is the weight matrix for the l -th layer, $b^{(l)}$ is the bias, $\sigma(\cdot)$ is the activation function, and $a^{(l-1)}$ is the activation from the previous layer.

$$z^{(l)} = W^{(l)} a^{(l-1)} + b^{(l)} \quad (1)$$

$$a^{(l)} = \sigma \left(z^{(l)} \right) \quad (2)$$

After loss from the output layer $\sigma^{(L)}$ has been computed via Equation (3), backpropagation can take place. Here L is the loss function.

$$\delta^{(L)} = \frac{\partial L}{\partial a^{(L)}} \cdot \sigma' \left(z^{(L)} \right) \quad (3)$$

Concretely,

$$\delta^{(l)} = \left(\delta^{(l+1)} W^{(l+1)T} \right) \cdot \sigma' \left(z^{(l)} \right) \quad (4)$$

Where $W^{(l+1)T}$ is the transpose of the weight matrix of the $(l+1)$ -th layer, $\sigma' \left(z^{(l)} \right)$ is the derivative of the activation function from the l -th layer. Then, the new gradients can be

computed via:

$$\frac{\partial L}{\partial W^{(l)}} = a^{(l-1)T} \delta^{(l)} \quad (5)$$

$$\frac{\partial L}{\partial b^{(l)}} = \delta^{(l)} \quad (6)$$

Eventually, the weights are updated via the GD process:

$$W^{(l)} = W^{(l)} - \eta \frac{\partial L}{\partial W^{(l)}} \quad (7)$$

$$b^{(l)} = b^{(l)} - \eta \frac{\partial L}{\partial b^{(l)}} \quad (8)$$

Appendix C: Definitions of Financial Factors

Table 1: Detailed Definitions and Sources of Factor Variables

Acronym	Authors	Year	Short Description	Detailed Definition
AbnormalAccruals	Xie	2001	Abnormal Accruals	Define Accruals as net income (ib) minus operating cash flow (oancf), divided by average total assets (at) for years $t - 1$ and t . If oancf is missing, replace operating cash flow with funds from operations (fopt) minus the annual change in total current assets (act) plus the annual change in cash and short-term investments (che) plus the annual change in current liabilities (lct) minus the annual change in debt in current liabilities (dlc). For each year t and 2-digit sic code, regress Accruals on: the inverse of average total assets for year $t - 1$, the change in revenue (sale) from year $t - 1$ to t divided by total assets for $t - 1$, property plant and equipment (ppeg) divided by total assets for $t - 1$. AbnormalAccrual is the residual from this cross-sectional regression.
Accruals	Sloan	1996	Accruals	Annual change in current total assets (act) minus annual change in cash and short-term investments (che) minus annual change in current liabilities (lct) minus annual change in debt in current liabilities (dlc) minus change in income taxes (txp). All divided by average total assets (at) over this year and last year. Exclude if $abs(prc) < 5$.
AccrualsBM	Bartov and Kim	2004	Book-to-market and accruals	Binary variable equal to 1 if stock is in the highest Accrual quintile and the lowest BM quintile, and equal to 0 if stock is in the lowest Accrual quintile and the highest BM quintile. Exclude if book equity (ceq) is negative.
AM	Fama and French	1992	Total assets to market	Total assets (at) divided by market value of equity.
AnalystRevision	Hawkins, Chamberlin, Daniel	1984	EPS forecast revision	keep $fpi == "1"$, last obs each month. Signal is meanest / last month's meanest.

Continued on next page

Table 1 – Continued from previous page

Acronym	Authors	Year	Short Description	Detailed Definition
Announcement Return	Chan et al.	1996	Earnings announce- ment return	Get announcement date for quarterly earnings from IBES ($fpi = 6$). AnnouncementReturn is the sum of $(ret - mktrf + rf)$ from one day before an earnings announcement to 2 days after the announcement.
AssetGrowth	Cooper et al.	2008	Asset growth	Annual growth rate of total assets (at).
BetaLiquidityPS	Pastor and Stambaugh	2003	Stambaugh liquidity beta	Monthly excess return (ret -rf) regressed on innovations in liquidity from Pastor's website (https://faculty.chicagobooth.edu/lubos.pastor/research/liq_data_1962_2018.txt). Use 60 month rolling window regression, and require at least 36 non-missing observations.
BetaTailRisk	Kelly and Jiang	2014	Tail risk beta	Each month, compute the 5th percentile over daily returns over all firms. For all daily return observations with return below that 5th percentile, compute the average of $(\log(\text{ret}/5\text{th percentile of cross-sectional return distribution}))$. Call that average tailEX. BetaTail-Risk is the coefficient of a 120-month rolling regression of a firm's stock return on tailEX. Exclude if price less than 5 or share code greater than 11.
BM	Stattman	1980	Book to market	Log of tangible book equity over market equity matched at fiscal year end.
BMdec	Fama and French	1992	BM using Dec ME	Book-to-market using December market equity values.
BookLeverage	Fama and French	1992	Book leverage (an- nual)	Total assets (at) divided by book value of equity plus deferred taxes (txdite) and preferred stock. Equity is shareholder equity (seq) if available, or book equity (ceq) plus preferred stock (pstk, if missing pstkrv, if missing pstkl), or total assets minus total liabilities (lt).
BPEBM	Penman et al.	2007	Leverage component of BM	$BP - EBM$, where $BP = (ceq + tstk - dvpa) / (shrout * abs(prc))$, and EBM is defined above. Exclude if price less than 5.
Cash	Palazzo	2012	Cash to assets	Ratio of quarterly cash and short-term investments (cheq) and total assets (atq).

Continued on next page

Table 1 – Continued from previous page

Acronym	Authors	Year	Short Description	Detailed Definition
CashProd	Chandrashekar and Rao	2009	Cash Productivity	Calculate market value of equity (mve_c) as absolute price (prc) times number of shares outstanding (shROUT). Cash productivity is equal to the difference between mve_c and total assets (at) divided by cash and short-term investments (che).
CBOperProf	Ball et al.	2016	Cash-based operating profitability	Revenue (revt) minus cost (cogs) - (administrative expenses (xsga) - R&D expenses (xrd)) minus annual change in receivables (rect), annual change in investment (invnt) and annual change in prepaid expenses, plus annual change in current deferred revenue (drc), long-term deferred revenue (drlt), accounts payable (ap) and accrued expenses (xacc), all divided by total assets (at) in year t. Replace all variables in the numerator with 0 if they are missing. Exclude if share code is greater than 11, market value of equity, BM or total assets are missing, or if SIC code is between 6000 and 6999.
CF	Lakonishok et al.	1994	Cash flow to market	Net income (ib) plus depreciation (dp) divided by market equity. Exclude NASDAQ stocks.
cfp	Desai et al.	2004	Operating Cash flows to price	Operating cash-flow (oancf) divided by market value of equity. If operating cash-flow is missing, replace by difference between net income (ib) and level of accruals, where the latter is the annual change in current assets (act) minus the annual change in cash and short-term investments (che), minus the annual change in current liabilities (lct) plus the annual change in debt in current liabilities (dlc) plus the annual change in payable income taxes (txp) plus depreciation (dp).
ChAssetTurnover	Soliman	2008	Change in Asset Turnover	Annual change in AssetTurnover (defined above). Exclude if price less than 5.
ChEQ	Lockwood and Prombutr	2010	Growth in book equity	Ratio of book equity (ceq) to book equity in the previous year. Include only if book equity is positive this year and last year.
ChForecastAccrual	Barth and Hutton	2004	Change in Forecast and Accrual	Within upper half of Accruals distribution, equal to 1 if mean earnings estimate increased relative to the previous month. 0 if it decreased.

Continued on next page

Table 1 – Continued from previous page

Acronym	Authors	Year	Short Description	Detailed Definition
ChInv	Thomas and Zhang	2002	Inventory Growth	12-month change in inventory (inv _t) divided by average total assets.
ChInvIA	Abarbanell and Bushee	1998	Change in capital inv (ind adj)	Growth in capital expenditure (cap _x) minus average growth in capital expenditure in the same industry (two-digit SIC). If cap _x is missing, capital expenditure is defined as the annual change in property, plant and equipment (ppent). Capital expenditure growth is defined as the percentage growth of cap _x today relative to the average cap _x over the previous two years ($.5 * (\text{cap}_{x,t-1} + \text{cap}_{x,t-2})$), or as percentage growth relative to the previous year only if $t - 2$ is missing.
ChNAnalyst	Scherbina	2008	Decline in Analyst Coverage	Using IBES forecasts, keep if $fpi = 1$, fpedats not missing, and $fpedats > statpers + 30$. Binary variable equal to 1 if the number of analysts (numest) for next quarter's EPS estimate decreased relative to three months ago, and 0 if it increased. Keep if in bottom two size quintiles among all firms to match OP's table.
ChNNCOA	Soliman	2008	Change in Net Non-current Op Assets	Twelve-month change in noncurrent operating assets. Noncurrent operating assets is $((at - act - ivao) - (lt - dlc - dl\bar{t}t))/at$.
ChNWC	Soliman	2008	Change in Net Working Capital	Twelve-month change in net working capital. Net working capital is $((act - che) - (lct - dlc))/at$
ChTax	Thomas and Zhang	2011	Change in Taxes	4-quarter change in quarterly total taxes (txtq), scaled by lagged total assets (at).
CompEquIss	Daniel and Titman	2006	Composite equity issuance	5-year growth rate of market value of equity minus 5-year stock return.
CompositeDebtIssuance	Lyandres et al.	2008	Composite debt issuance	Log of long-term debt (dl $\bar{t}t$) plus debt in current liabilities (dlc) minus log of the same variable 5 years ago.
ConvDebt	Valta	2016	Convertible debt indicator	Binary variable equal to 1 if deferred charges (dc) greater than 0 or common shares reserved for convertible debt (cshrc) greater than 0.

Continued on next page

Table 1 – Continued from previous page

Acronym	Authors	Year	Short Description	Detailed Definition
CoskewACX	Ang et al.	2006	Coskewness using daily returns	Signal is the sample counterpart of $E[\tilde{r}_{it}\tilde{r}_{mt}^2]/(SD[\tilde{r}_{it}]SD[\tilde{r}_{mt}]^2)$ where $\tilde{r}_{it}$ is the de-meaned stock return and $\tilde{r}_{mt}$ is the de-meaned market excess return. Signal is computed using the past year of daily data, and using the NYSE CRSP VW index for the market (dsia), with returns continuously compounded.
CustomerMomentum	Cohen and Frazzini	2008	Customer momentum	Based on firms' principals customers from Compustat Segment data as in Cohen and Frazzini.
DelBreadth	Chen et al.	2002	Breadth of ownership	Quarterly change in the number of institutional owners (numinstowners) from 13F data. Exclude if in the lowest quintile of stocks by market value of equity (based on NYSE stocks only).
DelCOA	Richardson et al.	2005	Change in current operating assets	Difference in current operating assets (total current assets (act) minus cash and short-term investments (che)) between years t-1 and t, scaled by average total assets (at) in years t-1 and t.
DelCOL	Richardson et al.	2005	Change in current operating liabilities	Difference in current operating liabilities (total current liabilities (lct) minus debt in current liabilities (dlc)) between years t-1 and t, scaled by average total assets (at) in years t-1 and t.
DelEqu	Richardson et al.	2005	Change in equity to assets	Difference in book equity (ceq) between years t-1 and t, scaled by average total assets (at) in years t-1 and t.
DelFINL	Richardson et al.	2005	Change in financial liabilities	Difference in financial liabilities (sum of long-term debt (dltt), current liabilities (dlc) and preferred stock (pstk)) between years t-1 and t, scaled by average total assets (at) in years t-1 and t.
DelLTI	Richardson et al.	2005	Change in long-term investment	Difference in investment and advances (ivao) between years t-1 and t, scaled by average total assets (at) in years t-1 and t.
DelNetFin	Richardson et al.	2005	Change in net financial assets	Compute the sum of short-term investments (ivst) and investments and advances (ivao) minus the sum of long-term debt (dltt), debt in current liabilities (dlc) and preferred stock capital. Divide the difference between the current and one-year lagged sum by total assets (at) averaged over the current and previous fiscal years.

Continued on next page

Table 1 – Continued from previous page

Acronym	Authors	Year	Short Description	Detailed Definition
DivInit	Michaely et al.	1995	Dividend Initiation	Keep only <i>distcd</i> 2nd <i>digit</i> = 2 or 3. Define dividend initiation as having paid a dividend in month t ($divamt > 0$), and not having paid a dividend in the last 24 months. DivInit is equal to 1 if a dividend was initiated in the past 6 months, and 0 for all other stocks.
DivOmit	Michaely et al.	1995	Dividend Omission	Keep only <i>distcd</i> 2nd <i>digit</i> = 2 or 3. Define firms as quarterly, semi-annual, or annual payers based on payment history. Define a consistent payer as a firm that paid quarterly or semi-annual dividends for the past 18 months, or annual dividends for the past 2 years. An omission is the first month where a consistent payer failed to pay a dividend over the past quarter/6-months/year. Finally, DivOmit = 1 if there is an omission in the past 2 months and 0 otherwise.
DivSeason	Hartzmark and Salomon	2013	Dividend seasonality	Drop if 3rd digit of <i>distcd</i> = 2 or ≥ 6 . Assign DivSeason = 0 if there was a dividend paid in the last 12 months. Replace DivSeason = 1 if the third digit of <i>distcd</i> is 3, 0, or 1, and a positive dividend was paid 2, 5, 8, or 11 months ago, if the third digit is 4 and a dividend was paid 5 or 11 months ago, or if the third digit is 5 and a dividend was paid 11 months ago.
DivYieldST	Litzenberger and Ramaswamy	1979	Predicted div yield next month	Using CRSP distributions, keep only <i>distcd</i> beginning in 12 and if the third digit of <i>distcd</i> is 3, 4, or 5. Also keep only if stock paid a dividend in the last 12 months. Define $Ediv1 = div$ 2 months ago if <i>distcd</i> 's 3rd digit is 0, 1, or 3. Define $Ediv1 = div$ 5 months ago if the <i>distcd</i> 3rd digit is 4. Define $Ediv = div$ 11 months ago if the <i>distcd</i> 3rd digit is 5. Define $Edy1$ as $Ediv1/abs(pre)$. Finally, discretize $Edy1$ to smooth around the huge mass at 0 as follows: $DivYieldST = 0$ if $Edy1 = 0$, 1 if Edy is between 0 and 0.005, 2 if $Edy1$ is between 0.005 and 0.010, and 3 if $Edy > 0.010$.

Continued on next page

Table 1 – Continued from previous page

Acronym	Authors	Year	Short Description	Detailed Definition
dNoa	Hirshleifer et al.	2004	change in net operating assets	12-month growth in Net Operating Assets scaled by lagged total assets (at). Net Operating assets are operating assets minus operating liabilities. Operating assets are total assets (at) minus cash and short-term investments (che), operating liabilities are total assets minus long-term debt (dltt), minority interest (mib), deferred charges (dlc), book equity (ceq) and preferred stock (pstk), all items (except at and ceq) replaced with 0 if missing.
DolVol	Brennan et al.	1998	Past trading volume	Log of two-month lagged trading volume (vol) times two-month lagged price (prc).
Earnings Consistency	Alwathainani	2009	Earnings consistency	Average earnings growth over previous 48 months. Earnings growth is defined as EPS (epspx) minus EPS 12 months ago divided by average EPS 12 and 24 months ago. Exclude if price less than 5, absolute value of 12 month earnings growth greater 600%, or earnings growth and earnings growth 12 months ago have different signs.
Earnings ForecastDisparity	Da Warachka and	2011	Long-vs-short EPS forecasts	Analyst forecasted 5-year earnings growth (fgr5yr) minus 100 times the difference between mean earnings forecast (meanest) and fiscal year earnings expectations (fy0a) scaled by the absolute value of fy0a. Drop if fpedats is missing or $fpedats - statpers < 30$
EarningsSurprise	Foster et al.	1984	Earnings Surprise	EPS (epspxq) minus EPS twelve months ago - Drift, scaled by standard deviation of that expression. Drift is the average earnings growth (EPS - EPS twelve months ago) over the past two years. Exclude if price less than 5.
EarnSupBig	Hou	2007	Earnings surprise of big firms	Average monthly value of EarningsSurprise (defined above) of the 30% largest companies by market value of equity in the same Fama-French 48 industry. Exclude the largest 30% of companies for EarnSupBig (not to compute the anomaly).

Continued on next page

Table 1 – Continued from previous page

Acronym	Authors	Year	Short Description	Detailed Definition
EBM	Penman et al.	2007	Enterprise component of BM	$(ceq + che - dl tt - dlc - dc - dvpa + tstkp)/(mve_c + che - dl tt - dlc - dc - dvpa + tstkp)$. Exclude if price less than 5.
EntMult	Loughran and Wellman	2011	Enterprise Multiple	Market value of $equity + long - termdebt(dl tt) + debtincurrentliabilities(dlc) + deferredcharges(dc) - cash$ and short-term investments (che), divided by operating income ($oibdp$). Exclude if missing book equity or negative operating income.
EP	Basu	1977	Earnings-to-Price	$ib/lag(marketvalueofequity, 6 \text{ months})$. NYSE stocks only. Exclude if $EP < 0$. Lag simulates the Dec 31 market equity used in original paper.
EquityDuration	Dechow et al.	2004	Equity Duration	see code of Andrew Y.; Zimmermann (2020).
ExchSwitch	Dharan and Ikenberry	1995	Exchange Switch	Binary variable equal to 1 if a firm switched from AMEX or NASDAQ to NYSE within the past year, or from NASDAQ to AMEX within the past year.
FEPS	Cen et al.	2006	Analyst earnings per share	Using IBES unadjusted forecasts, keep $fpi == 1$, signal is meanest.
fgr5yrLag	La Porta	1996	Long-term EPS forecast	Lag long-term earnings forecast ($fgr5yr$) by 6 months. Then keep only June observations, and fill in missing with most recent obs. Exclude if book equity (ceq), net income (ib), deferred taxes ($txdi$), dividends (dvp), revenue ($sale$) or depreciation (dp) is missing.
FirmAgeMom	Zhang	2006	Firm Age - Momentum	6 month return, restricted to the bottom quintile of the cross-sectional firm age distribution. Exclude if price less than 5 or firm younger than 12 months.
ForecastDispersion	Diether et al.	2002	EPS Forecast Dispersion	Keep $fpi = 1$ and $fpedats > statpers + 30$. Standard deviation of earnings estimates ($stdev_est$) scaled by mean earnings estimate.

Continued on next page

Table 1 – Continued from previous page

Acronym	Authors	Year	Short Description	Detailed Definition
Frontier	Nguyen and Swanson	2009	Efficient frontier index	Frontier is the residual of a regression of $\log(\text{BM})$ on $\log(\text{book equity (ceq)})$, $\log(\text{long-term debt (dltt)})$ to assets (at), capital expenditures (capx) to revenue (sale), R&D expense (xrd) to revenue, advertising expense (xad) to revenue, property plant and equipment (ppent) to assets, EBIT (ebitda) to assets, and dummies for Fama-French's 48 industry definitions. Regression is updated each month with a rolling window of 60 months.
GP	Novy-Marx	2013	gross profits / total assets	Revenue (sale) - cost of goods solds (cogs), divided by total assets (at). Drop if financial.
GrAdExp	Lou	2014	Growth in advertising expenses	Log of advertising expense (xad) minus log of advertising expense last year. Exclude if price less than 5, xad less than .1 or stock in the lowest decile of market value of equity.
grcapx	Anderson and Garcia-Feijoo	2006	Change in capex (two years)	Growth rate of capital expenditures (capx) relative to two years ago. If capx is missing, replace with annual change in property, plant and equipment (ppent).
grcapx3y	Anderson and Garcia-Feijoo	2006	Change in capex (three years)	Capital expenditures (capx) divided by the sum of capital expenditures from year - 1, year -2, and year -3. If capx is missing, replace with annual change in property, plant and equipment (ppent).
Herf	Hou and Robinson	2006	Industry concentration (sales)	Three-year rolling average of the three digit industry Herfindahl index based on firm revenue (sale). Exclude regulated industries (4011, 4210, 4213 & year ≤ 1980 ; 4512 & year ≤ 1978 , 4812, 4813 & year ≤ 1982 , 4900-4999 in any year).
HerfBE	Hou and Robinson	2006	Industry concentration (equity)	Three-year rolling average of the three digit industry Herfindahl index based on firm book equity. Exclude regulated industries (4011, 4210, 4213 & year ≤ 1980 ; 4512 & year ≤ 1978 , 4812, 4813 & year ≤ 1982 , 4900-4999 in any year).
hire	Bazdresch et al.	2014	Employment growth	Change in number of employees (emp) between t-1 and t, scaled by average number of employees in t-1 and t. Replace hire with 0 if emp or lagged emp is missing.

Continued on next page

Table 1 – Continued from previous page

Acronym	Authors	Year	Short Description	Detailed Definition
IdioVol3F	Ang et al.	2006	Idiosyncratic risk (3 factor)	Standard deviation of residuals from Fama-French three factor regressions using the past month of daily data. Value weighted.
IdioVolAHT	Ali et al.	2003	Idiosyncratic risk (AHT)	Standard deviation of residuals from CAPM regressions using the past year of daily data. Require at least 100 non-missing observations.
Illiquidity	Amihud	2002	Amihud's illiquidity	Past twelve-month average of: daily return ($\text{abs}(\text{ret})$) divided by turnover($(\text{abs}(\text{prc}) * \text{vol})$).
IndIPO	Ritter	1991	Initial Public Offerings	1 if IPO in the past 6-36 months. 0 otherwise. IPO dates are taken from Jay Ritter's IPO data available at: http://bear.warrington.ufl.edu/ritter/ipodata.htm . Missing IPO dates imply IndIPO = 0.
IndMom	Grinblatt and Moskowitz	1999	Industry Momentum	Weighted average of firm-level 6 month buy-and-hold return. Average is taken over two digit industries each month and weights are based on market value of equity.
IndRetBig	Hou	2007	Industry return of big firms	Average monthly return (ret) of the 30% largest companies by market value of equity in the same Fama-French 48 industry. Exclude the largest 30% of companies for IndRetBig (not to compute the anomaly!).
IntanBM	Daniel and Titman	2006	Intangible return using BM	In each month, run a cross-sectional regression of a firm's five-year stock return on 5-year lagged BM (defined above) and a constructed regressor that is the change in BM from 5 years ago to today plus the five-year stock return. The residual from that regression is IntanBM.
IntanCFP	Daniel and Titman	2006	Intangible return using CFtoP	In each month, run a cross-sectional regression of a firm's five-year stock return on the 5-year lagged CFP = (net income (ni) plus depreciation (dp))/market value of equity and a constructed regressor that is the change in CFP from 5 years ago to today plus the five-year stock return. The residual from that regression is IntanCFP.

Continued on next page

Table 1 – Continued from previous page

Acronym	Authors	Year	Short Description	Detailed Definition
IntanEP	Daniel and Titman	2006	Intangible return using EP	In each month, run a cross-sectional regression of a firm's five-year stock return on the 5 year lagged EP = net income (ni)/market value of equity and a constructed regressor that is the change in EP from 5 years ago to today plus the five-year stock return. The residual from that regression is IntanEP.
IntanSP	Daniel and Titman	2006	Intangible return using Sale2P	In each month, run a cross-sectional regression of a firm's five-year stock return on 5-year lagged SP (defined above) and a constructed regressor that is the change in SP from 5 years ago to today plus the five-year stock return. The residual from that regression is IntanSP.
IntMom	Novy-Marx	2012	Intermediate Momentum	Stock return between months t-12 and t-6.
Investment	Titman et al.	2004	Investment to revenue	Ratio of capital investment (capx) to revenue (revt) divided by the firm-specific 36-month rolling mean of that ratio. Exclude if revenue less than \$10m.
InvestPPEInv	Lyandres et al.	2008	change in ppe and inv/assets	One-year change in property, plants and equipment (ppeg) plus one year change in inventory (inv), scaled by one-year lagged assets (at).
InvGrowth	Belo and Lin	2012	Inventory Growth	Deflate <i>inv</i> growth using <i>gnp</i> deflator. Signal is deflated <i>inv</i> growth rate from fiscal year t to fiscal year t-1. Drop if 1 digit sic code is 4 or 6, or if <i>at</i> or <i>ppent</i> ≤ 0.
Leverage	Bhandari	1988	Market leverage	Total liabilities (lt) divided by market value of equity.
LRreversal	De Bondt and Thaler	1985	Long-run reversal	Stock return between months t-36 and t-13.
MaxRet	Bali et al.	2011	Maximum return over month	Maximum of daily returns (ret) over the previous month.
MeanRank				
Continued on next page				

Table 1 – Continued from previous page

Acronym	Authors	Year	Short Description	Detailed Definition
RevGrowth	Lakonishok, Shleifer, Vishny	1994	Revenue Growth Rank	Rank firms by their annual revenue growth each year over the past 5 years. MeanRankRevGrowth is the weighted average of ranks over the past 5 years, that is, $\text{MeanRankRevGrowth} = (5*\text{Rank}_{t-1} + 4*\text{Rank}_{t-2} + 3*\text{Rank}_{t-3} + 2*\text{Rank}_{t-4} + 1*\text{Rank}_{t-5})/15$. Exclude NASDAQ stocks.
Mom12m	Jegadeesh and Titman	1993	Momentum (12-month)	Stock return between months t-12 and t-1.
Mom12mOff Season	Heston and Sadka	2008	Momentum without the seasonal part	Average return in other months over the previous year.
Mom6m	Jegadeesh and Titman	1993	Momentum (6 month)	Stock return between months t-6 and t-1.
MomOffSeason	Heston and Sadka	2008	Off season long-term reversal	Average return in other months over the preceding 2-5 years.
MomOffSeason 06YrPlus	Heston and Sadka	2008	Off season reversal years 6 to 10	Average return in other months over the preceding 6-10 years.
MomOffSeason 16YrPlus	Heston and Sadka	2008	Off season reversal years 16 to 20	Average return in other months over the preceding 16-20 years.
MomRev	Chan and Ko	2006	Momentum and LT Reversal	Binary variable equal to 1 if firm is in the highest Mom6m quintile and the lowest Mom36m quintile, and equal to 0 if firm is in the lowest Mom6m quintile and the highest Mom36m quintile. Exclude if price less than 5.
MomSeason	Heston and Sadka	2008	Return seasonality years 2 to 5	Average return in the same month over the preceding 2-5 years.
MomSeason				

Continued on next page

Table 1 – Continued from previous page

Acronym	Authors	Year	Short Description	Detailed Definition
06YrPlus	Heston and Sadka	2008	Return seasonality years 6 to 10	Average return in the same month over the preceding 6-10 years.
MomSeason 11YrPlus	Heston and Sadka	2008	Return seasonality years 11 to 15	Average return in the same month over the preceding 11-15 years.
MomSeason 16YrPlus	Heston and Sadka	2008	Return seasonality years 16 to 20	Average return in the same month over the preceding 16-20 years.
MomSeason Short	Heston and Sadka	2008	Return seasonality last year	Average return in the same month in the previous year.
MomVol	Lee and Swaminathan	2000	Momentum in high volume stocks	Define momentum as Mom6m, and volume as the rolling average of the past 6 months of monthly turnover (minimum 5 months). Independent sort stocks into 10 momentum ports and 3 volume ports. Keep if volume is in the top port, and assign signal = momentum port. Drop if less than 2 years on CRSP.
MS	Mohanram	2005	Mohanram G-score	See code of Andrew Y.; Zimmermann (2020) for details. MS is only evaluated for low BM firms and comes from combining three signals related to profitability and cash flow, two signals related to income volatility, and three signals related to investment.
NetDebtFinance	Bradshaw et al.	2006	Net debt financing	Long-term debt issuance (dltis) minus long-term debt reduction (dltr) minus current debt changes (dlcch), scaled by average total assets (at) in years t-1 and t. Replace missing values of <i>dlcch</i> with 0. Exclude if ratio is greater than 1.

Continued on next page

Table 1 – Continued from previous page

Acronym	Authors	Year	Short Description	Detailed Definition
NetDebtPrice	Penman et al.	2007	Net debt to price	Long-term debt (dltt) plus debt in current liabilities (dlc) plus preferred stock (pstk) plus preferred dividends in arrears (dvpa) minus treasury stock (tstkp) minus cash and short-term investments (che), scaled by market value of equity. Exclude if SIC between 6000 and 6999, or if missing value for total assets (at), net income (ib), common shares outstanding (csho), book value of equity (ceq) or price close fiscal year (prcc.f). Keep only 3rd B/M Quintile.
NetEquityFinance	Bradshaw et al.	2006	Net equity financing	Sale of common stock (sstk) minus purchase of common stock (prstk), scaled by average total assets (at) from years t and t-1. Exclude if absolute value of ratio is greater than 1.
NetPayoutYield	Boudoukh et al.	2007	Net Payout Yield	Dividends (dvc) plus purchase of common and preferred stock (prstk) minus sale of common and preferred stock (sstk), divided by market value of equity lagged 6 months. Exclude if NetPayoutYield is 0, financial firm based on SIC code, $ceq \leq 0$, or less than 2 years in CRSP.
NOA	Hirshleifer et al.	2004	Net Operating Assets	Difference between operating assets and operating liabilities, scaled by lagged total assets. Operating assets are total assets (at) minus cash and short-term investments (che), operating liabilities are total assets minus long-term debt (dltt), minority interest (mib), deferred charges (dc) and book equity (ceq).
OPLeverage	Novy-Marx	2011	Operating leverage	Sum of administrative expenses (xsga) and cost of goods sold (cogs), scaled by total assets (at). Use $xsga = 0$ if xsga is missing.
OrderBacklog	Rajgopal et al.	2003	Order backlog	Order backlog (ob) divided by average total assets (at) in years t-1 and t. Exclude if order backlog is 0.
OrderBacklog Chg	Baik and Ahn	2007	Change in order backlog	Define normalized order backlog as order backlog (ob) divided by average total assets (at) in years t-1 and t. Exclude if order backlog is 0. Signal is normalized order backlog minus normalized order backlog one year ago.

Continued on next page

Table 1 – Continued from previous page

Acronym	Authors	Year	Short Description	Detailed Definition
OrgCap	Eisfeldt and Papanikolaou	2013	Organizational capital	Defined recursively. Set xsga to 0 if missing. Initialize with OrgCap = 4*general expenses (xsga) in the first year, and calculate as .85*OrgCap previous year + xsga/gnpdeflator current year thereafter. Scale by total assets (at). Keep only december year ends, and non-financials, and non-missing sic code. Then winsoriz at the 1% level and standardized by subtracting the same month within-industry mean (Fama French 17 industries) and dividing by the same month within industry standard deviation of OrgCap.
OScore	Dichev	1998	O Score	OScore = $-1.32 - .407 \cdot \log(\text{at} / \text{GNP deflator}) + 6.03 \cdot (\text{lt} / \text{at}) - 1.43 \cdot (\text{act} - \text{lct}) / \text{at} + .076 \cdot (\text{lct} / \text{act}) - 1.72 \cdot \mathbf{I}(\text{lt} > \text{at}) - 2.37 \cdot (\text{ib} / \text{at}) - 1.83 \cdot (\text{fopt} / \text{lt}) + .285 \cdot (\text{ib} + \text{ib}_{t-12} + \text{ib}_{t-24} < 0) - .521 \cdot (\text{ib} - \text{ib}_{t-12}) / (\text{abs}(\text{ib}) + .\text{abs}(\text{ib}_{t-12}))$. $\text{fopt} = \text{oancf}$ if fopt is missing. Exclude if SIC code between 3999 and 4999, or greater than 5999. Exclude if price less than 5. Then exclude if OScore is in bottom quintile of OScore (original paper shows non-monotonic returns, as does our replication).
PayoutYield	Boudoukh et al.	2007	Payout Yield	Sum of dividends (dvc), purchase of common and preferred stock (prstk) and max(preferred stock redemption value (pstkrv), 0) over market value of equity lagged 6 months. Exclude if PayoutYield ≤ 0 , financial firm based on SIC code, $\text{ceq} \leq 0$, or less than 2 years in CRSP.
PctAcc	Hafzalla et al.	2011	Percent Operating Accruals	Income before extraordinary items (ib) minus net cash flow (oancf) divided by absolute value of ib. If oancf is missing, PctAcc is defined as $((\text{act} - \text{act}_{t-12}) - (\text{che} - \text{che}_{t-12}) - ((\text{lct} - \text{lct}_{t-12}) - (\text{dlc} - \text{dlc}_{t-12}) - (\text{txp} - \text{txp}_{t-12}) - \text{dp})) / \text{abs}(\text{ib})$. In either case, if ib is equal to 0, divide by .01 instead. Exclude if price less than 5.
Price	Blume and Husic	1973	Price	Log of absolute value of price (prc).

Continued on next page

Table 1 – Continued from previous page

Acronym	Authors	Year	Short Description	Detailed Definition
PriceDelayRsqr	Hou and Moskowitz	2005	Price delay r square	Each July regress stock return on day t on market return in $t, t - 1, \dots, t - 4$ using observations from July 1 of the previous year to June 30 of the current year. Then regress again with no market return lags. PriceDelay Rsqr = $1 - [\text{Rsqr from second regression}] / [\text{Rsqr from first regression}]$.
PS	Piotroski	2000	Piotroski F-score	Sum of nine indicator variables which are: 1 if net income (ib) greater 0; 1 if net cash flow (oancf) greater 0; 1 if return on assets (ib/at) increased relative to previous year; 1 if net cash flow greater net income; 1 if long-term debt to assets (dltt/at) declined over the previous year; if current assets to current liabilities (act/lct) increased over the previous year; 1 if ebit/sale (ebit = ib + txt + xint) increased over the previous year; 1 if revenue to assets increased over the previous year; 1 if shrou $\leq$ shrou last year. Include highest quintile of book-to-market only. Exclude if missing any of the input variables.
RD	Chan et al.	2001	R&D over market cap	R&D expense (xrd) over market value of equity.
RDAbility	Cohen et al.	2013	R&D ability	Regress log of sales growth (sale over sale in previous year) on log of $(1 + \text{xrd}/\text{sale})$ in 5 bivariate regressions with $(1 + \text{xrd}/\text{sale})$ lagged by 1, ..., 5 years. Run regressions over previous 8 years and require at least 6 non-missing observations. Also require at least half of past research and development observations to be non-zero. RDAbility is the mean of the coefficients on the five lags of $\log(1 + \text{xrd}/\text{sale})$. Set to missing if firm is not in the highest tercile of xrd/sale in a year or if research and development expenses are non-positive.
RDcap	Li	2011	R&D capital-to-assets	R&D capital to assets is the weighted sum of lagged R&D expenditures scaled by the assets (at). I replace xrd with 0 if xrd is missing and compute the numerator as $\text{xrd}_t + .8\text{xrd}_{t-1} + .6\text{xrd}_{t-2} + .4\text{xrd}_{t-3} + .2\text{xrd}_{t-4}$. Replace RDcap with missing before 1980 or if firm is in upper two thirds of market cap ($\text{shrou} * \text{abs}(\text{prc})$) distribution in a month.

Continued on next page

Table 1 – Continued from previous page

Acronym	Authors	Year	Short Description	Detailed Definition
RDIPO	Gou, Lev and Shi	2006	IPO and no R&D spending	Binary variable equal to 1 if R&D expense (<i>xrd</i>) = 0 and IndIPO = 1. 0 otherwise.
RDS	Landsman et al.	2011	Real dirty surplus	Define Dirty Surplus as annual change in marketable securities adjustment <i>msa</i> plus annual change in retained earnings adjustment (<i>recta</i>) + .65 times the annual change in min(Unrecognized prior service cost (<i>pcupsu</i>) - Pension additional minimum liability (<i>paddml</i>),0). Real dirty surplus is the annual change in book equity (<i>ceq</i>) minus dirty surplus minus (net income (<i>ni</i>) minus dividends preferred (<i>dvp</i>)) + dividends (<i>divamt</i>) - end-of-fiscal-year-stock-price (<i>prcc-f</i>)*annual change in common shares outstanding (<i>csho</i>).
RealizedVol	Ang et al.	2006	Realized Volatility (Total)	Standard deviation of residuals from CAPM regressions using the past month of daily data. Value weighted.
Residual Momentum	Blitz et al.	2011	Momentum based on FF3 residuals	Run a rolling regression over 36 months of excess return (<i>retrf</i>) on excess market return (<i>mktrf</i>), size and value factors (<i>smb</i> , <i>hml</i>) and compute idiosyncratic returns as the one-month lagged residual. ResidualMomentum is the rolling mean of the residual divided by the rolling standard deviation of the residual, both computed over the past 11 months.
retConglomerate	Cohen and Lou	2012	Conglomerate return	Identify conglomerate firms as those with multiple OPSEG or BUSSEG entries in the Compustat segment data (and require that at least 80% of firm's total assets are covered by segment data). Compute monthly stock return at the 2-digit SIC level for stand-alone (non-conglomerate) firms only, and match those returns to conglomerates' segments. Compute weighted conglomerate return as the industry return of stand-alone companies, weighted with a conglomerate's total sales in each industry.
ReturnSkew	Bali et al.	2015	Return skewness	Skewness of daily returns (<i>ret</i>) over previous month.

Continued on next page

Table 1 – Continued from previous page

Acronym	Authors	Year	Short Description	Detailed Definition
ReturnSkew3F	Bali et al.	2015	Idiosyncratic skewness (3F model)	Skewness of idiosyncratic returns computed as residuals from regression of daily excess returns (ret - rf) on Fama-French factors (mktrf, smb, hml) over the previous month. I require at least 15 non-missing observations.
REV6	Chan et al.	1996	Earnings forecast revisions	Define revisions as the change in the mean earnings estimate (mean-est) for the next quarter from month t-1 to t, scaled by stock price in month t-1. REV6 is the sum of that variable from months t-6 to t.
Revenue Surprise	Jegadeesh and Livnat	2006	Revenue Surprise	Define revenue per share as quarterly revenue (revtq) divided by quarterly common shares outstanding (cshprq). RevenueSurprise is the 4-quarter change in revenue per share minus the average 4-quarter change in revenue per share over the previous 2 years. RevenueSurprise is scaled by its standard deviation over the previous 2 years. Exclude if price less than 5.
RIO_Dis	Nagel	2005	Inst Own and Forecast	Dispersion Follow RIO_MB, except define Disp as the stdev of IBES forecasts where fpi == 1 divided by at, and sort on Disp instead of MB. Finally, let RIO_Dis = lagged RIO quintile if the Disp quintile >= 4.
RIO_MB	Nagel	2005	Inst Own and Market to Book	Residual institutional ownership (RIO) is defined as $\log(\text{institutional ownership} / (1 - \text{institutional ownership})) + 23.6 - 2.89 * \log(\text{market value of equity}) + .08 * \log(\text{market value of equity})^2$. Replace instown_perc with 0 if it is missing, with .9999 if it's above .9999, and with .0001 if it's below .0001. Then lag RIO by 6 months. Keep only stocks with size > 20th percentile across exchcd == 1 or 2. Define MB as market cap / (ceq + txdtc), where txdtc = 0 if missing, and MB is replaced with missing if ceq < 0. Then sort into quintiles based on the lagged RIO, and sort into quintiles based on MB. Finally, let RIO_MB = lagged RIO quintile if the MB quintile = 5.

Continued on next page

Table 1 – Continued from previous page

Acronym	Authors	Year	Short Description	Detailed Definition
RIO_Turnover	Nagel	2005	Inst Own and Turnover	Follow RIO_MB, except define turnover as vol/shrout. Let RIO_Turnover = lagged RIO quintile if the Volatility quintile == 5.
RIO_Volatility	Nagel	2005	Inst Own and Idio Vol	Follow RIO_MB, except define volatility as the rolling standard deviation of the last 12 months of the stock return. Let RIO_Disp = lagged RIO quintile if the Volatility quintile == 5.
roaq	Balakrishnan et al.	2010	Return on assets (qtrly)	Quarterly net income (ibq) divided by lagged total assets (atq).
sfe	Elgers et al.	2001	Earnings Forecast to price	Median estimate for next year eps (fpi = 1) in March, divided by stock price from December. Dec fiscal year ends only, keep only forecasts more than 90 days out. Keep only below median analyst coverage each month.
ShareIss1Y	Pontiff and Woodgate	2008	Share issuance (1 year)	Growth in number of shares between t-18 and t-6. Number of shares is calculated as shrout/cfacshr to adjust for splits.
ShareIss5Y	Daniel and Titman	2006	Share issuance (5 year)	5-year growth in number of shares. Number of shares is calculated as shrout/cfacshr to adjust for splits.
ShareVol	Datar et al.	1998	Share Volume	Let tempvol = sum of monthly share trading volume (vol) over the previous three months, scaled by 3 times common shares outstanding (shrout). Let ShareVol = 1 if tempvol $\geq$ 10%, and ShareVol = 0 if tempvol < 5%.
Size	Banz	1981	Size	Log of monthly market value of equity (abs(prc)*shrout)).
std_turn	Chordia et al.	2001	Share turnover volatility	Standard deviation of turnover (vol/shrout) over the past 36 months.
STreversal	Jegadeesh	1990	Short term reversal	Stock return (ret) over the previous month.
SurpriseRD	Eberhart et al.	2004	Unexpected R&D increase	Binary variable equal to 1 if: R&D (xrd) scaled by revenue (revt) is positive, R&D scaled by total assets (at) is positive, annual R&D growth is greater than 5%, annual growth in R&D over total assets is greater than 5%. SurpriseRD is 0 otherwise.

Continued on next page

Table 1 – Continued from previous page

Acronym	Authors	Year	Short Description	Detailed Definition
tang	Hahn and Lee	2009	Tangibility	Cash and short-term investments (che) plus .715*receivables (rect) + .547*inventory (invnt) + .535* property, plant and equipment (ppent), scaled by total assets (at). Only defined for manufacturing firms ($SIC \geq 2000$ and $SIC < 4000$). Exclude the lowest tercile of manufacturing firms by total assets.
Tax	Lev and Nissim	2004	Taxable income to income	Ratio of Taxes paid and tax share of net income. Numerator is defined as the sum of foreign (txfo) and federal (txfed) income taxes. If either one is missing, numerator is defined as total taxes (txt) minus deferred taxes (txdi). Denominator is the product of the prevailing tax rate and net income (ib). Tax rate is .48 before 1979, .46 from 1979 to 1986, .4 in 1987, .34 between 1988 and 1992 and .35 from 1993 onwards. If net income is negative, and the numerator is positive, tax is defined as 1. Exclude if price less than 5.
TotalAccruals	Richardson et al.	2005	Total accruals	Before 1988: Change in net working capital ((act - che) - (lct - dlc)) plus change in net noncurrent assets ((at - act - ivao) - (lt - dlc - dltd)) plus change in net financial assets ((ivst + ivao - (dltd + dlc + pstk))). Starting in 1988: net income (ni) minus total, operating and investment cashflows (oancf, ivncf, fincf) plus stock sales minus repurchases and dividends (sstk, prstk, dv)). Scaled by lagged total assets (at). Replace missings in ivao, ivst, dltd, dlc, pstk sstk, prstk, dv with 0.
TrendFactor	Han et al.	2016	Trend Factor	See Andrew Y.; Zimmermann (2020) section 2.1 and 2.2.
VolSD	Chordia et al.	2001	Volume Variance	Rolling standard deviation of monthly trading volume (vol) over the past 36 months (require at least 24 observations). Include only NYSE stocks.
XFIN	Bradshaw, Richardson, Sloan	2006	Net external financing	Sale of common stock (sstk) minus dividends (dv) minus purchase of common stock (prstk) plus long-term debt issuance (dltis) minus long-term debt reductions (dltr). Scaled by total assets (at).

Continued on next page

Table 1 – Continued from previous page

Acronym	Authors	Year	Short Description	Detailed Definition
zerotrade	Liu	2006	Days with zero trades 6	In each month, count the number of days with no trades. Define zerotrade as the number of days without trades plus (the sum of monthly turnover (vol/shrout) divided by 48×10^5), multiplied by 21/number of trading days per month. Zerotrade is the 6-month average of that variable.
zerotradeAlt1	Liu	2006	Days with zero trades 1	In each month, count the number of days with no trades. Define zerotrade as the number of days without trades plus (the sum of monthly turnover (vol/shrout) divided by 48×10^5), multiplied by 21/number of trading days per month. Take 1-month average.
zerotradeAlt12	Liu	2006	Days with zero trades 12	In each month, count the number of days with no trades. Define zerotrade as the number of days without trades plus (the sum of monthly turnover (vol/shrout) divided by 48×10^5), multiplied by 21/number of trading days per month. Take 12-month average.
AdExp	Chan et al.	2001	Advertising Expense	Advertising expense (xad) over market value of equity (shrout*abs(prc)).
AgeIPO	Ritter	1991	IPO and age Age	is (current year - founding year from Jay Ritter's dataset). Exclude if IndIPO == 0.
AnalystValue	Frankel and Lee	1998	Analyst Value	Value based on a three-stage dividend discount model and analyst forecasts, scaled by market value. See code of Andrew Y.; Zimmermann (2020) for details.
AOP	Frankel and Lee	1998	Analyst Optimism	AnalystValue (defined above) minus IntrinsicValue (defined above), divided by abs(IntrinsicValue).
Beta	Fama and MacBeth	1973	CAPM beta	Coefficient of a 60-month rolling window regression of monthly stock returns minus the riskfree rate on market return minus the risk free rate (ewretd - rf). Exclude if estimate based on less than 20 months of returns.

Continued on next page

Table 1 – Continued from previous page

Acronym	Authors	Year	Short Description	Detailed Definition
BetaFP	Frazzini and Pedersen	2014	Frazzini-Pedersen Beta	Using daily data, call tempRi the sum of today's return and the past 2 days, call tempRm the sum of today's mkt return and the past 2 days mkt return. Regress return on tempRi and tempRm using past 5 years of data, with a minimum of 3 years. BetaFP is the square root of the r square from this regression times the ratio of the idiosyncratic stock vol and the market vol.
BidAskSpread	Amihud and Mendelsohn	1986	Bid-ask spread	Effective bid ask spread based on Corwin-Schulz scaled by stock price.
BrandInvest	Belo, Lin and Vitorino	2014	Brand capital investment	Advertising expenses (xad) divided by BrandCapital. Brand capital is computed by the perpetual inventory method. In the first year, brand capital is advertising expense divided by (.5 + .1). In subsequent years, we let brand capital depreciate with a rate of .5 and add current advertising expenses. Brand capital is scaled by total assets (at). Set to missing if advertising expense is missing. Drop if 1st digit of sic = 4 or 6, and keep only Dec fiscal year ends.
Coskewness	Harvey and Siddique	2000	Coskewness	Signal is the sample counterpart of $E[\tilde{r}_{it}\tilde{r}_{mt}^2]/(SD[\tilde{r}_{it}]SD[\tilde{r}_{mt}]^2)$ where $\tilde{r}_{it}$ is the de-meaned stock excess return and $\tilde{r}_{mt}$ is the de-meaned market excess return. Signal is computed using the past 60 months of monthly data, and using the NYSE/AMEX CRSP VW index for the market (msic). See Andrew Y.; Zimmermann (2020) code for details.
DebtIssuance	Spiess and Affleck-Graves	1999	Debt Issuance	Equal to 1 if debt issuance (dltis) greater 0 and 0 otherwise. Exclude if share code > 11 or missing book-to-market.
FirmAge	Barry and Brown	1984	Firm age based on CRSP	Months since start of CRSP coverage.

Continued on next page

Table 1 – Continued from previous page

Acronym	Authors	Year	Short Description	Detailed Definition
FR	Franzoni and Marin	2006	Pension Funding Status	FR = (FVPA - PBO), scaled by market value of equity. FVPA is pbnaa from 1980 to 1986, pplao + pplao from 1987 to 1997, and pplao after 1997. PBO is pbnvv from 1980 to 1986, pbpro + pbpru from 1987 to 1997, and pbpro after 1997. Exclude if price less than 5 or shrcd > 11.
GrLTNOA	Fairfield, Whisenant and Yohn	2003	Growth in long term operating assets	Annual growth in net operating assets, minus accruals. Net operating assets are (rect + invt + ppent + aco + intan + ao- ap- lco- lo) / at. Accruals are (rect-l12.rect + invt - l12.invt + aco - l12.aco - (ap - l12.ap + lco - l12.lco) - dp) / ((at + l12.at)/2).
GrSaleToGrInv	Abarbanell and Bushee	1998	Sales growth over inventory growth	Percentage growth in sales (sale) relative to average sales of t-1 and t-2, minus percentage growth in inventory (invt) relative to average inventory of t-1 and t-2. Both growth terms are calculated relative to t-1 only if t-2 is missing.
GrSaleToGrOverhead	Abarbanell and Bushee	1998	Sales growth over overhead growth	GrSaleToGrOverHead = Percentage growth in sales (sale) relative to average sales of t-1 and t-2, minus percentage growth in administrative expenses (xsga) relative to average administrative expenses of t-1 and t-2. Both growth terms are calculated relative to t-1 only if t-2 is missing.
HerfAsset	Hou and Robinson	2006	Industry concentration (assets)	Three-year rolling average of the three digit industry Herfindahl index based on firm assets (at). Exclude regulated industries (4011, 4210, 4213 & year ≤ 1980; 4512 & year ≤ 1978, 4812, 4813 & year ≤ 1982, 4900-4999 in any year).
High52	George and Hwang	2004	52-week high	Price (prc/cfacshr) divided by the maximum price over the previous 12 months.
IO_ShortInterest	Asquith Pathak and Ritter	2005	Inst own among high short interest	Exclude all stocks with short interest (ShortInterest) below 99th percentile. IO_ShortInterest is institutional ownership (instown-perc). Keep NYSE Only.
MomOffSeason				

Continued on next page

Table 1 – Continued from previous page

Acronym	Authors	Year	Short Description	Detailed Definition
11YrPlus	Heston and Sadka	2008	Off season reversal years 11 to 15	Average return in other months over the preceding 11-15 years.
MRreversal	De Bondt and Thaler	1985	Medium-run reversal	Stock return between months t-18 and t-13.
NumEarnIncrease	Loh and Warachka	2012	Earnings streak length	Number if consecutive 4-quarter increases in ibq, up to 8.
OperProf	Fama and French	2006	operating profits / book equity	Revenue (revt) minus cost (cogs) - administrative expenses (xsga) - interest expenses (xint), scaled by book value of equity (ceq). Exclude smallest size tercile.
OperProfRD	Ball et al.	2016	Operating profitability R&D adjusted	Revenue (revt) minus cost (cogs) - (administrative expenses (xsga) - R&D expenses (xrd)), all divided by total assets (at) in year t. Replace all variables in the numerator with 0 if they are missing. Exclude if share code is greater 11, market value of equity, BM or total assets are missing, or if SIC code between 6000 and 6999.
PriceDelaySlope	Hou and Moskowitz	2005	Price delay coeff	Each July regress daily stock return (ret) on market return (mktrf) in $t, t-1, \dots, t-4$ with observations over the previous year. Define PriceDelaySlope as the ratio of $1*\text{beta on } \text{mktrf}_t - 1 + 2*\text{beta on } \text{mktrf}_t - 2 + 3*\text{beta on } \text{mktrf}_t - 3 + 4*\text{beta on } \text{mktrf}_t - 4$, and $[\text{beta on } \text{mktrf}_t + \text{beta on } \text{mktrf}_t - 1 + \text{beta on } \text{mktrf}_t - 2 + \text{beta on } \text{mktrf}_t - 3 + \text{beta on } \text{mktrf}_t - 4]$.
PriceDelayTstat	Hou and Moskowitz	2005	Price delay SE adjusted	Each July regress daily stock return (ret) on market return (mktrf) in $t, t-1, \dots, t-4$ using observations from July 1 of the previous year to June 30 of the current year. Trim the highest and lowest 1% of estimated coefficients, standard errors, and t-stats. Define PriceDelayTstat as the ratio of $1*\text{tstat on } \text{mktrf}_t - 1 + 2*\text{tstat on } \text{mktrf}_t - 2 + 3*\text{tstat on } \text{mktrf}_t - 3 + 4*\text{tstat on } \text{mktrf}_t - 4$, and $[\text{tstat on } \text{mktrf}_t + \text{tstat on } \text{mktrf}_t - 1 + \text{tstat on } \text{mktrf}_t - 2 + \text{tstat on } \text{mktrf}_t - 3 + \text{tstat on } \text{mktrf}_t - 4]$. Finally winsor the extreme 10 and 90 percentiles each month.

Continued on next page

Table 1 – Continued from previous page

Acronym	Authors	Year	Short Description	Detailed Definition
realestate	Tuzel	2010	Real estate holdings	Industry-adjusted value of real estate holdings. Real estate holdings are $(fatb+fatl)/ppeg$ if available, and $(ppenb+ppenl)/ppent$ otherwise. Drop firms in 2 digit sics with < 5 firm or missing at or missing both $ppent$ and $ppeg$. Subtract monthly industry-mean at the 2 digit SIC level.
RoE	Haugen and Baker	1996	net income / book equity	Net income (ni) over book value of equity (ceq). Exclude if price less than 5.
ShareRepurchase	Ikenberry et al.	1995	Share repurchases	Binary variable equal to 1 if stock repurchase indicated in cash flow statement ($prstk > 0$), and 0 if $prstk = 0$.
ShortInterest	Dechow et al.	2001	Short Interest	Short-interest from Compustat (shortint) scaled by shares outstanding (shrout). Short-interest data are available bi-weekly with a four-day lag. I use the mid-month observation to make sure data would be available in real time.
sinAlgo	Hong and Kacperczyk	2009	Sin Stock (selection criteria)	Using Compustat Segment data, sinAlgo is defined as a binary variable equal to 1 if at least one segment of a firm is listed as being in at least one of the following industries: $sic \geq 2100$ & $sic \leq 2199$, $sic \geq 2080$ & $sic \leq 2085$, NAICS in {7132, 71312, 713210, 71329, 713290, 72112, 721120}. As in the original paper, we assume that the sin stock indicator applies to the entire history and future of the identified firm. sinAlgo is equal to 0 if the firm is not identified in the CS Segment data as a sin stock and if the firm is in one of the following industries: $(sic \geq 2000 \text{ \& } sic \leq 2046) \text{ OR } (sic \geq 2050 \text{ \& } sic \leq 2063) \text{ OR } (sic \geq 2070 \text{ \& } sic \leq 2079) \text{ OR } (sic \geq 2090 \text{ \& } sic \leq 2092) \text{ OR } (sic \geq 2095 \text{ \& } sic \leq 2099) \text{ OR } (sic \geq 2064 \text{ \& } sic \leq 2068) \text{ OR } (sic \geq 2086 \text{ \& } sic \leq 2087) \text{ OR } (sic \geq 920 \text{ \& } sic \leq 999) \text{ OR } (sic \geq 3650 \text{ \& } sic \leq 3652) \text{ OR } sic == 3732 \text{ OR } (sic \geq 3931 \text{ \& } sic \leq 3932) \text{ OR } (sic \geq 3940 \text{ \& } sic \leq 3949) \text{ OR } (sic \geq 7800 \text{ \& } sic \leq 7833) \text{ OR } (sic \geq 7840 \text{ \& } sic \leq 7841) \text{ OR } (sic \geq 7900 \text{ \& } sic \leq 7911) \text{ OR } (sic \geq 7920 \text{ \& } sic \leq 7933) \text{ OR } (sic \geq 7940 \text{ \& } sic \leq 7949) \text{ OR } sic == 7980 \text{ OR } (sic \geq 7990 \text{ \& } sic \leq 7999)$.

Continued on next page

Table 1 – Continued from previous page

Acronym	Authors	Year	Short Description	Detailed Definition
SP	Barbee, Mukherji and Raines	1996	Sales-to-price	Ratio of annual sales (sale) to market value of equity.
Spinoff	Cusatis, Miles and Woolridge	1993	Spinoffs	Spinoffs are identified as all observations in the CRSP acquisition file with valid acperm entry. Spinoff is a binary variable equal to 1 if a firm is identified in the CRSP Acquisition data and if it has at most two years of history in the CRSP stock return data. Spinoff is equal to 0 otherwise.
VarCF	Haugen and Baker	1996	Cash-flow to price variance	Rolling variance of (ib+dp)/mve_c over the past 60 months (minimum 24 months data required).
VolMkt	Haugen and Baker	1996	Volume to market equity	Average monthly dollar trading volume (vol*abs(prc)) over the previous 12 months, scaled by market value of equity. Exclude if price less than 5.
VolumeTrend	Haugen and Baker	1996	Volume Trend	Rolling coefficient from regressing monthly trading volume on a linear time trend over a window of 60 months (require that at least 30 exist). Scale coefficient by 60-month average of trading volume.

Appendix D: Hyperparameters used in this chapter

Hyperparameter	fw5	fw4	fw3	fw2	fw1	GKX2020_fw5	GKX2020_fw4	GKX2020_fw3
Learning rate	[0.0007,0.001]	[0.0007,0.001]	[0.0007,0.001]	[0.0007,0.001]	[0.0007,0.001]	0.02	0.017	0.01
Batch size	Full batch (entire training sample)							
Number of epochs	8000	8000	8000	8000	8000	3000	5000	5000
Early stopping patience	500	500	500	500	500	500	500	500
Regularization	L1	L1	L1	L1	L1	L1	L1	L1
Dropout rate	0	0	0	0	0	0	0	0
Seeds	200	200	200	200	200	200	200	200

Note: As each stock has its own optimal set of hyperparameters, more detailed parameter configurations can be provided upon request.

Appendix E: Factor Importance plots of fw1, fw4 and fw5 in this chapter

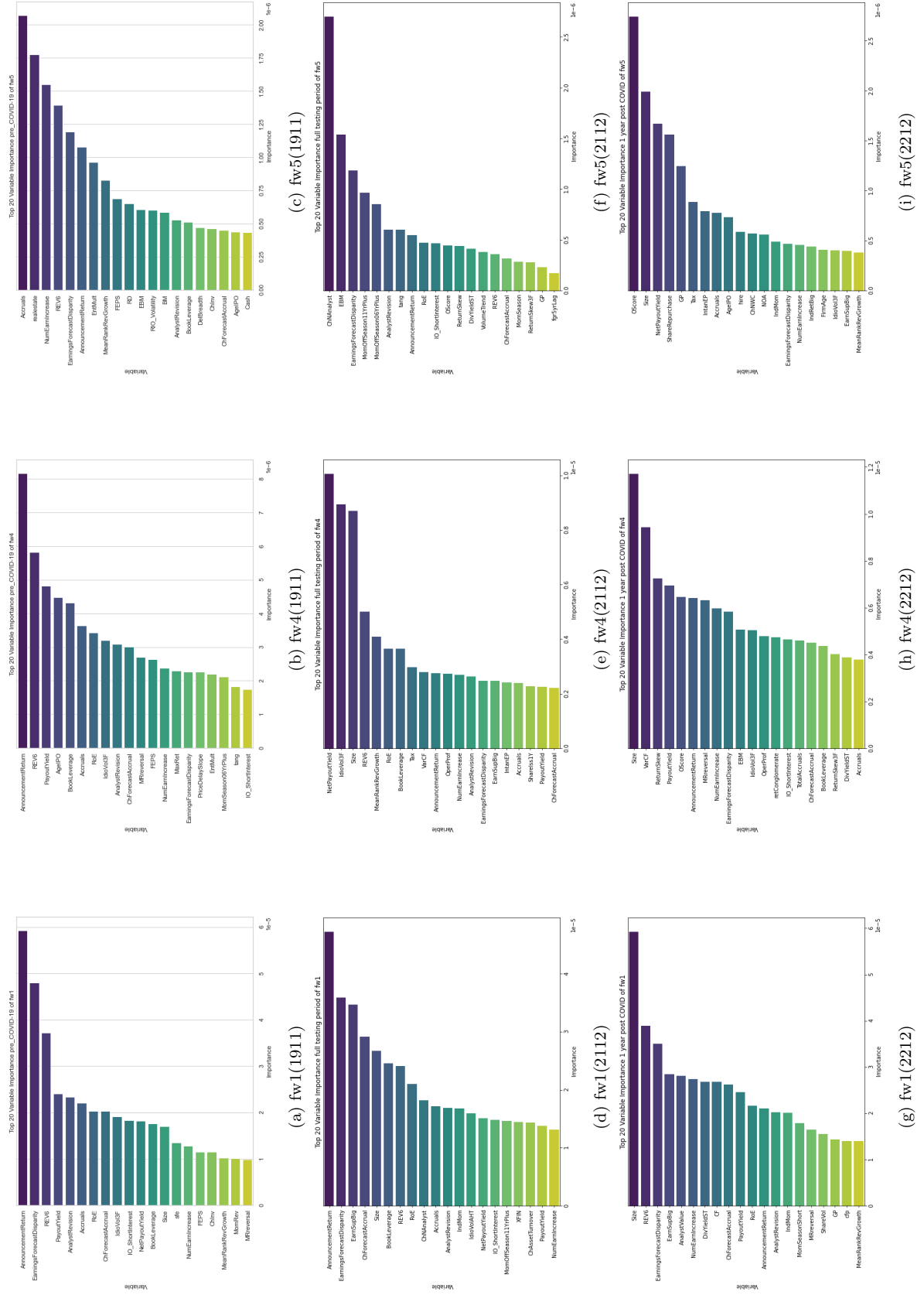


Figure 1: Factor importance plots of fw1, fw4 and fw5.

Appendix of Chapter 3

Appendix A: Out-of-sample α distributions for alternative models

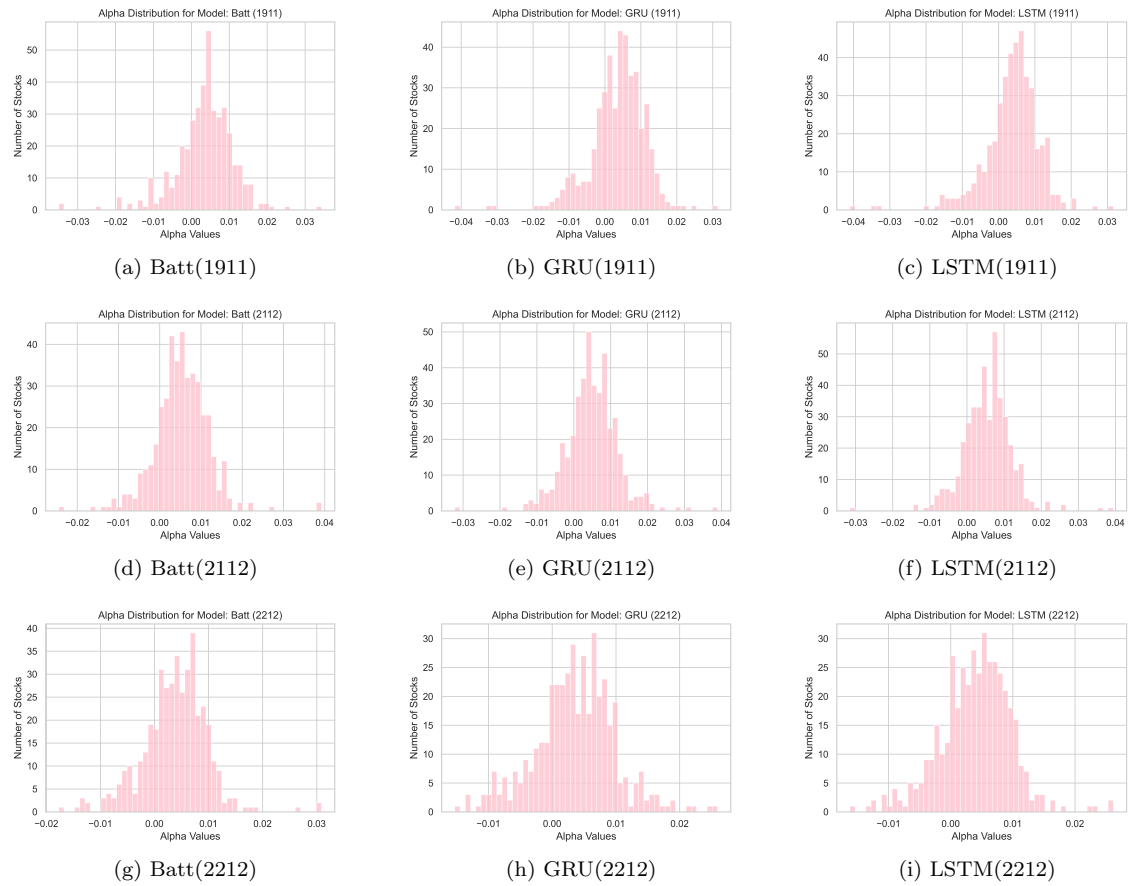
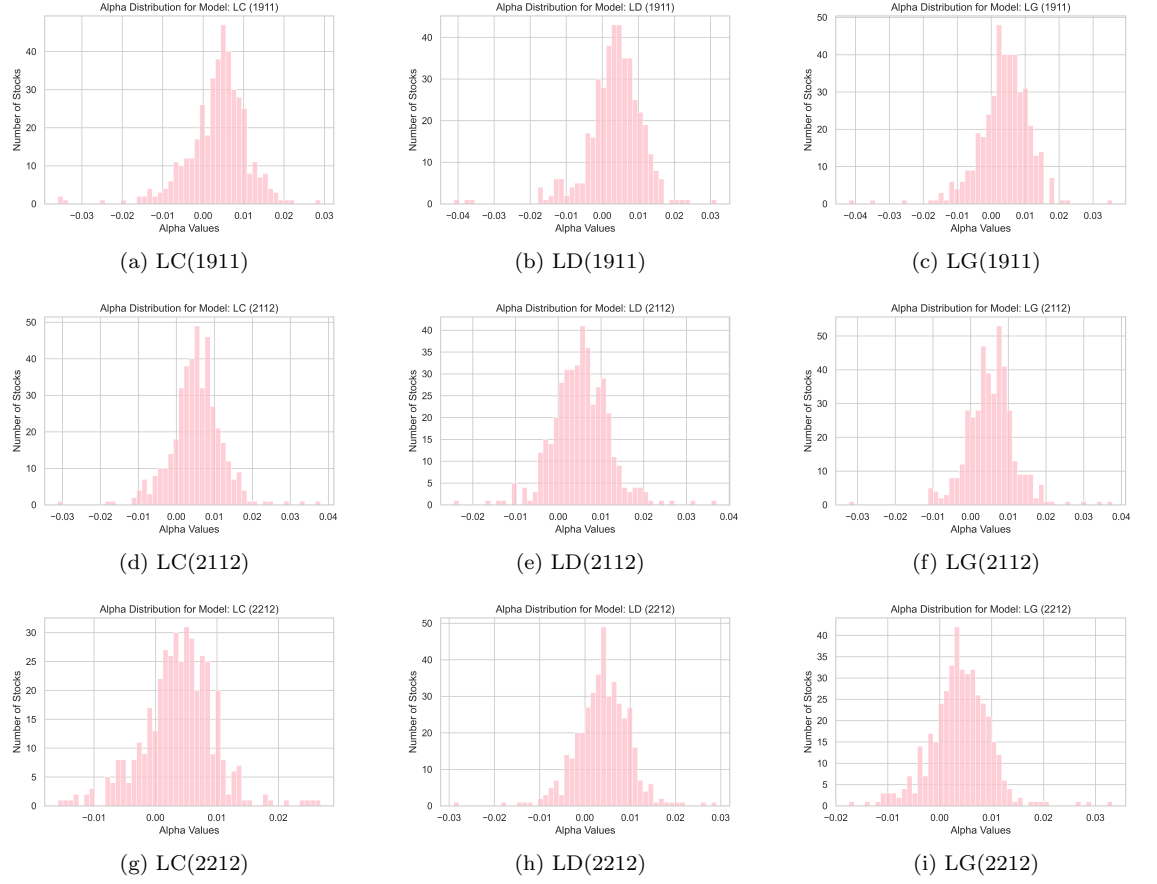


Figure 2: OOS α distribution of the alternative models Part 1

Figure 3: OOS α distribution of the alternative models Part 2

Appendix B: Adam optimizer

Adam updates parameters by maintaining the first and second moments of the gradients.

Algorithm 2 Adam Optimization

- 1: **Initialize:** $m_0 = 0, v_0 = 0, l = 0$
 - 2: **Set hyperparameters:** learning rate $\eta = 0.001, \beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$
 - 3: **while** θ_l not converged **do**
 - 4: $l \leftarrow l + 1$
 - 5: Compute gradient: $g_l = \nabla_{\theta} L(\theta_{l-1})$
 - 6: $m_l \leftarrow \beta_1 m_{l-1} + (1 - \beta_1) g_l$
 - 7: $v_l \leftarrow \beta_2 v_{l-1} + (1 - \beta_2) g_l \odot g_l$ $\triangleright \odot$: element-wise multiplication
 - 8: $\hat{m}_l \leftarrow \frac{m_l}{1 - \beta_1^l}$ $\triangleright$ Bias correction
 - 9: $\hat{v}_l \leftarrow \frac{v_l}{1 - \beta_2^l}$
 - 10: $\theta_l \leftarrow \theta_{l-1} - \eta \cdot \frac{\hat{m}_l}{\sqrt{\hat{v}_l} + \epsilon}$
 - 11: **end while**
 - 12: **Return:** θ_l
-

Source: Adapted from Gu et al. (2020)

where g_t and θ_t represent the gradients and parameters computed from previous sections respectively.

Appendix C: Early Stopping

Algorithm 3 Early Stopping

```

1: Initialize:  $j = 0$ ,  $\epsilon = \infty$ , select patience parameter  $p$ .
2: while  $j < p$  do
3:   Update  $\theta$  using the training algorithm (e.g., for  $h$  steps).
4:   Calculate the prediction error from the validation sample, denoted as  $\epsilon'$ .
5:   if  $\epsilon' < \epsilon$  then
6:      $j \leftarrow 0$ 
7:      $\epsilon \leftarrow \epsilon'$ 
8:      $\theta' \leftarrow \theta$ 
9:   else
10:     $j \leftarrow j + 1$ 
11:   end if
12: end while
13: Return:  $\theta'$ 

```

Source: Adapted from Gu et al. (2020)

Appendix D: List of Abbreviations

RNN	pretrained vanilla recurrent neural network model
GRU	pretrained RNN with Gated Recurrent Unit
LSTM	pretrained RNN with Long Short-Term Memory
Batt	pretrained RNN additive attention model
LC	pretrained RNN Luong's concatenate attention model
LD	pretrained RNN Luong's dot product attention model
LG	pretrained RNN Luong's general attention model
self_att	pretrained RNN global self-attention model
sparse_att	pretrained RNN sliding window sparse attention model

Appendix E: Variable Importance

Appendix F: Hyperparameters used in this chapter

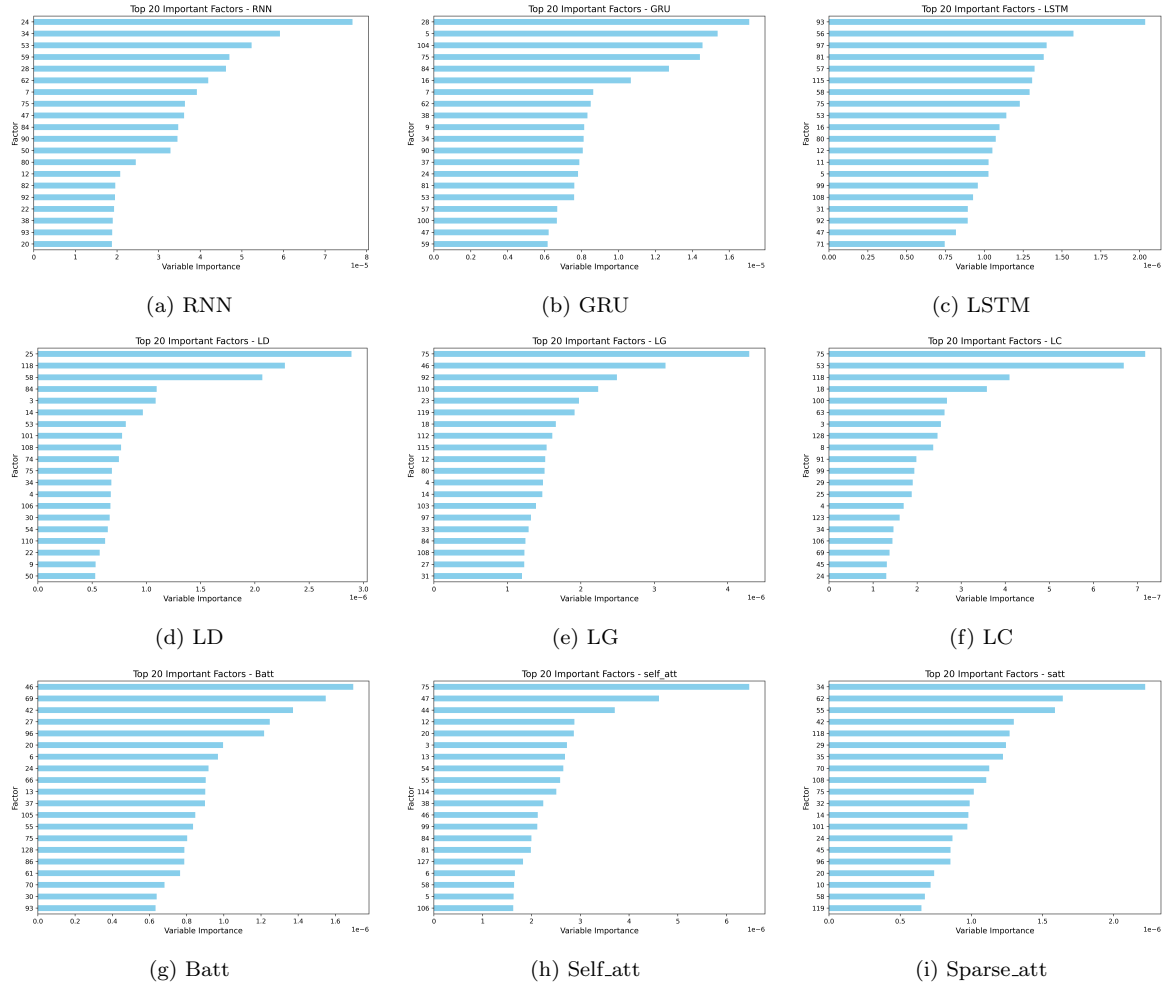


Figure 4: Factor importance of different models

Hyperparameter	RNN	LSTM	GRU	LD	LC	LG	Batt	self_att	sparse_att
Learning rate	[0.00005,0.0015]	[0.00005,0.005]	[0.00005,0.005]		[0.00007,0.0005]	[0.0001,0.0006]	[0.0002,0.0004]	[0.00005,0.0001]	[0.0001,0.0003]
Batch size	Full batch (entire training sample)								
Number of epochs	[1000,12000]	[1000,12000]	[1000,12000]		[300,5000]	[200,2000]	[350,1000]	[200,15000]	[400,1400]
Early stopping patience	200	200	200	200	200	200	200	200	200
Regularization	L1	L1	L1	L1	L1	L1	L1	L1	L1
Dropout rate	0	0	0	0	0	0	0	0	0
Seeds	200	200	200	200	200	200	200	200	200
Attention window	NA	NA	NA	full	full	full	full	full	16

Note: As each stock has its own optimal set of hyperparameters, more detailed parameter configurations can be provided upon request.

Appendix A: Transformer model Estimation

Assume the input is given by

$$X \in R^{n \times d} \quad (9)$$

where n is the length of the features and d is the dimension of features. The MSE loss function form can be presented as:

$$L = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2 \quad (10)$$

Gradients of output linear layer

$$\hat{y}_i = W_{\text{out}} h_{L,i} + b_{\text{out}} \quad (11)$$

Where $h_{L,i}$ is the outputs from previous layer, W_{out} and b_{out} are learnable parameters. For the gradient g , we have,

$$\begin{aligned} g_{W_{\text{out}}} &= \frac{\partial L}{\partial W_{\text{out}}} = \frac{\partial}{\partial W_{\text{out}}} \left[\frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2 \right] \\ &= \frac{1}{N} \sum_{i=1}^N 2(y_i - \hat{y}_i) \cdot \frac{\partial (y_i - \hat{y}_i)}{\partial W_{\text{out}}} \\ &= \frac{2}{N} \sum_{i=1}^N (\hat{y}_i - y_i) h_{L,i}^T \end{aligned} \quad (12)$$

$$\begin{aligned} g_{b_{\text{out}}} &= \frac{\partial L}{\partial b_{\text{out}}} = \frac{1}{N} \sum_{i=1}^N 2(y_i - \hat{y}_i) \cdot \frac{\partial (y_i - \hat{y}_i)}{\partial b_{\text{out}}} \\ &= \frac{2}{N} \sum_{i=1}^N (\hat{y}_i - y_i) \end{aligned} \quad (13)$$

Gradients of Attention layer

$$\begin{aligned}
Q &= hW_Q, \quad K = hW_K, \quad V = hW_V \\
S &= \frac{QK^T}{\sqrt{d_k}} \\
A &= \text{softmax}(S) \\
O &= AV \\
Z &= \text{Concat}(O_1, O_2, \dots, O_H)W_O
\end{aligned} \tag{14}$$

- $A = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) \in \mathbb{R}^{N \times N}$,
- $h \in \mathbb{R}^{N \times d}$, $W_V \in \mathbb{R}^{d \times d_k}$,
- $V \in \mathbb{R}^{N \times d_k}$, $O \in \mathbb{R}^{N \times d_k}$.

For $O = AV$,

$$\frac{\partial L}{\partial W_V} = \frac{\partial L}{\partial O} \cdot \frac{\partial O}{\partial V} \cdot \frac{\partial V}{\partial W_V}. \tag{15}$$

$$O_{ij} = \sum_k A_{ik} V_{kj}, \frac{\partial O_{ij}}{\partial V_{mn}} = A_{im} \cdot \delta_{jn}, \frac{\partial O}{\partial V} = A, \frac{\partial L}{\partial V} = \frac{\partial L}{\partial O} A^T. \tag{16}$$

$$V_{kj} = \sum_m h_{km} W_{V,mj}, \frac{\partial V_{kj}}{\partial W_{V,mn}} = h_{km} \cdot \delta_{jn}, \tag{17}$$

$$\frac{\partial L}{\partial W_{V,mn}} = \sum_{k,j} \frac{\partial L}{\partial V_{kj}} \cdot h_{km} \cdot \delta_{jn} = \sum_k \frac{\partial L}{\partial V_{kn}} \cdot h_{km}, \tag{18}$$

$$\frac{\partial L}{\partial W_V} = h^T \cdot \frac{\partial L}{\partial V} = h^T \cdot \left(\frac{\partial L}{\partial O} A^T \right). \tag{19}$$

$$g_{W_V} = \frac{\partial L}{\partial W_V} = h^T \left(\frac{\partial L}{\partial O} \right)^T A \tag{20}$$

For W_Q ,

$$g_{W_Q} = \frac{\partial L}{\partial W_Q} = \frac{\partial L}{\partial O} \cdot \frac{\partial O}{\partial A} \cdot \frac{\partial A}{\partial S} \cdot \frac{\partial S}{\partial Q} \cdot \frac{\partial Q}{\partial W_Q} \tag{21}$$

$$\frac{\partial O}{\partial A} = V \quad (22)$$

$$\frac{\partial L}{\partial A} = \frac{\partial L}{\partial O} V^T \quad (23)$$

$$\frac{\partial A_i}{\partial S_j} = \begin{cases} A_i(1 - A_i), & i = j \\ -A_i A_j, & i \neq j \end{cases} \quad (24)$$

$$\frac{\partial A}{\partial S} = \text{diag}(A) - AA^T \quad (25)$$

$$\frac{\partial L}{\partial S} = \frac{\partial L}{\partial A} \cdot [\text{diag}(A) - AA^T] \quad (26)$$

$$S_{ij} = \frac{\sum_m Q_{im} K_{jm}}{\sqrt{d_k}} \quad (27)$$

$$\frac{\partial S_{ij}}{\partial Q_{mn}} = \frac{K_{jn}}{\sqrt{d_k}} \cdot \delta_{im} = \frac{K^T}{\sqrt{d_k}} \quad (28)$$

$$\frac{\partial L}{\partial Q} = \frac{\partial L}{\partial S} \cdot \frac{K}{\sqrt{d_k}} \quad (29)$$

$$Q_{ij} = \sum_m h_{im} W_{Q,mj} \quad (30)$$

$$\frac{\partial Q_{ij}}{\partial W_{Q,mn}} = h_{im} \cdot \delta_{jn} \quad (31)$$

$$\frac{\partial Q}{\partial W_Q} = h^T \quad (32)$$

$$g_{W_Q} = \frac{\partial L}{\partial W_Q} = h^T \left(\frac{\partial L}{\partial S} \cdot \frac{K}{\sqrt{d_k}} \right)^T \quad (33)$$

For W_K ,

$$\frac{\partial L}{\partial W_K} = \frac{\partial L}{\partial O} \cdot \frac{\partial O}{\partial A} \cdot \frac{\partial A}{\partial S} \cdot \frac{\partial S}{\partial K} \cdot \frac{\partial K}{\partial W_K} \quad (34)$$

$$O_{ij} = \sum_k A_{ik} V_{kj} \quad (35)$$

$$\frac{\partial O_{ij}}{\partial A_{mn}} = V_{nj} \cdot \delta_{im} \quad (36)$$

$$\frac{\partial L}{\partial A} = \frac{\partial L}{\partial O} V^T \quad (37)$$

$$\frac{\partial A_i}{\partial S_j} = \begin{cases} A_i(1 - A_i), & i = j, \\ -A_i A_j, & i \neq j, \end{cases} \quad (38)$$

$$\frac{\partial A}{\partial S} = \text{diag}(A) - AA^T \quad (39)$$

$$\frac{\partial L}{\partial S} = \left(\frac{\partial L}{\partial O} V^T \right) \cdot (\text{diag}(A) - AA^T) \quad (40)$$

$$S_{ij} = \frac{1}{\sqrt{d_k}} \sum_m Q_{im} K_{jm} \quad (41)$$

$$\frac{\partial S_{ij}}{\partial K_{mn}} = \frac{1}{\sqrt{d_k}} Q_{im} \cdot \delta_{jn} \quad (42)$$

$$\frac{\partial S}{\partial K} = \frac{Q^T}{\sqrt{d_k}} \quad (43)$$

$$\frac{\partial L}{\partial K} = \frac{\partial L}{\partial S} \cdot \frac{Q}{\sqrt{d_k}} \quad (44)$$

$$K_{kj} = \sum_m h_{km} W_{K,mj} \quad (45)$$

$$\frac{\partial K_{kj}}{\partial W_{K,mn}} = h_{km} \cdot \delta_{jn} \quad (46)$$

$$\frac{\partial L}{\partial W_{K,mn}} = \sum_k \frac{\partial L}{\partial K_{kn}} \cdot h_{km} \quad (47)$$

$$\frac{\partial L}{\partial W_K} = h^T \cdot \frac{\partial L}{\partial K} = h^T \cdot \left(\frac{\partial L}{\partial S} \cdot \frac{Q}{\sqrt{d_k}} \right) \quad (48)$$

$$g_{W_K} = \frac{\partial L}{\partial W_K} = h^T \left(\frac{\partial L}{\partial S} \cdot \frac{Q}{\sqrt{d_k}} \right)^T. \quad (49)$$

In the case of multi-head attention,

$$Z = \text{Concat}(O_1, O_2, \dots, O_H) W_O \quad (50)$$

$$\frac{\partial L}{\partial W_O} = \frac{\partial L}{\partial Z} \cdot \frac{\partial Z}{\partial W_O} \quad (51)$$

$$(52)$$

Assume $C = \text{Concat}(O_1, O_2, \dots, O_H) \in \mathbb{R}^{N \times (Hd_k)}$,

$$Z = CW_O \quad (53)$$

$$Z_{ij} = \sum_k C_{ik} W_{O,kj} \quad (54)$$

$$\frac{\partial Z_{ij}}{\partial W_{O,mn}} = C_{im} \cdot \delta_{jn} \frac{\partial L}{\partial W_{O,mn}} = \sum_{i,j} \frac{\partial L}{\partial Z_{ij}} \cdot C_{im} \cdot \delta_{jn} = \sum_i \frac{\partial L}{\partial Z_{in}} \cdot C_{im} \quad (55)$$

$$\frac{\partial L}{\partial W_O} = C^T \cdot \frac{\partial L}{\partial Z} \quad (56)$$

$$g_{W_O} = \frac{\partial L}{\partial W_O} = \text{Concat}(O_1, O_2, \dots, O_H)^T \cdot \frac{\partial L}{\partial Z} \quad (57)$$

$$(58)$$

Gradients of MLP autoencoder feedforward NN (FFN)

$$\text{FFN}(h) = W_2 \cdot \text{ReLU}(W_1 h + b_1) + b_2 \quad (59)$$

Let $z = W_1 h + b_1$, $a = \text{ReLU}(z)$, For W_2 ,

$$g_{W_2} = \frac{\partial L}{\partial W_2} = \frac{\partial L}{\partial \text{FFN}} \cdot a^T \quad (60)$$

For b_2 ,

$$g_{b_2} = \frac{\partial L}{\partial b_2} = \frac{\partial L}{\partial \text{FFN}} \quad (61)$$

For W_1 ,

$$g_{W_1} = \frac{\partial L}{\partial W_1} = \frac{\partial L}{\partial a} \cdot \frac{\partial a}{\partial z} \cdot h^T \quad (62)$$

Where $\frac{\partial a}{\partial z} = 1$, if $z > 0$, 0 for otherwise.

Gradients of Embedding layer

In the embedding layer, x is reflected as $h_0 = W_{em}x$,

$$g_{W_{em}} = \frac{\partial L}{\partial W_{em}} = \frac{\partial L}{\partial h_0} \cdot x^T \quad (63)$$

Adam optimizer

Adam updates parameters by maintaining the first and second moments of the gradients.

Algorithm 4 Adam Optimization

```

1: Initialize:  $m_0 = 0, v_0 = 0, l = 0$ 
2: Set hyperparameters: learning rate  $\eta = 0.001, \beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$ 
3: while  $\theta_l$  not converged do
4:    $l \leftarrow l + 1$ 
5:   Compute gradient:  $g_l = \nabla_{\theta} L(\theta_{l-1})$ 
6:    $m_l \leftarrow \beta_1 m_{l-1} + (1 - \beta_1) g_l$ 
7:    $v_l \leftarrow \beta_2 v_{l-1} + (1 - \beta_2) g_l \odot g_l$  ▷  $\odot$ : element-wise multiplication
8:    $\hat{m}_l \leftarrow \frac{m_l}{1 - \beta_1^l}$  ▷ Bias correction
9:    $\hat{v}_l \leftarrow \frac{v_l}{1 - \beta_2^l}$ 
10:   $\theta_l \leftarrow \theta_{l-1} - \eta \cdot \frac{\hat{m}_l}{\sqrt{\hat{v}_l} + \epsilon}$ 
11: end while
12: Return:  $\theta_l$ 

```

Source: Adapted from Gu et al. (2020)

where g_t and θ_t represent the gradients and parameters computed from previous sections respectively.

Early stopping**Algorithm 5** Early Stopping

```

1: Initialize:  $j = 0, \epsilon = \infty$ , select patience parameter  $p$ .
2: while  $j < p$  do
3:   Update  $\theta$  using the training algorithm (e.g., for  $h$  steps).
4:   Calculate the prediction error from the validation sample, denoted as  $\epsilon'$ .
5:   if  $\epsilon' < \epsilon$  then
6:      $j \leftarrow 0$ 
7:      $\epsilon \leftarrow \epsilon'$ 
8:      $\theta' \leftarrow \theta$ 
9:   else
10:     $j \leftarrow j + 1$ 
11:   end if
12: end while
13: Return:  $\theta'$ 

```

Source: Adapted from Gu et al. (2020)

Layer Normalization

Algorithm 6 Layer Normalization

- 1: **Input:** Input vector $x \in \mathbb{R}^d$, learnable parameters γ, β
 - 2: Compute the mean: $\mu = \frac{1}{d} \sum_{i=1}^d x_i$
 - 3: Compute the variance: $\sigma^2 = \frac{1}{d} \sum_{i=1}^d (x_i - \mu)^2$
 - 4: Normalize: $\hat{x}_i = \frac{x_i - \mu}{\sqrt{\sigma^2 + \epsilon}}$ for $i = 1, \dots, d$
 - 5: Apply scale and shift: $y_i = \gamma_i \hat{x}_i + \beta_i$ for $i = 1, \dots, d$
 - 6: **Return:** Output vector y
-

Appendix B: Hyperparameters used in this chapter

Hyperparameter	SERT1	SERT2	SERT3	SERT4	SERT5	SERT6	SERT7	T-En1	T-En2	T-En3
Batch size	Full batch (entire training sample)									
Early stopping patience	200	200	200	200	200	200	200	200	200	200
Regularization	L1	L1	L1	L1	L1	L1	L1	L1	L1	L1
Dropout rate	0	0	0	0	0	0	0	0	0	0
Seeds	200	200	200	200	200	200	200	200	200	200
Encoder block	1	1	1	1	1	1	1	1	1	1
Decoder block	0	0	0	0	0	0	0	0	0	0
fw	1	1	1	1	1	1	1	1	1	1
input_dim	420	420	420	420	420	420	420	182	182	182
num_neurons	84	294	294	294	294	294	294	128	128	128
Head	1	1	2	3	4	6	7	1	2	4
Learning rate	0.00003	0.0015	0.003	0.001	0.001	0.001	0.001	0.002	0.003	0.003
LNF	y	n	n	n	n	n	n	n	n	n
Epochs	7700	500	270	500	500	500	500	325	270	270
Hyperparameter	Trans1	Trans2	Trans3	Trans4	Trans5	Trans6	Trans7	Trans8	Trans9	Trans10
Batch size	Full batch (entire training sample)									
Early stopping patience	200	200	200	200	200	200	200	200	200	200
Regularization	L1	L1	L1	L1	L1	L1	L1	L1	L1	L1
Dropout rate	0	0	0	0	0	0	0	0	0	0
Seeds	200	200	200	200	200	200	200	200	200	200
Encoder block	1	1	1	1	1	1	1	1	1	1
Decoder block	1	1	1	1	1	1	1	1	1	1
fw	1	1	1	1	1	1	1	1	1	1
input_dim	420	420	420	420	420	420	420	182	182	182
num_neurons	84	294	294	294	294	294	294	294	294	294
Head	1	1	2	3	4	6	7	1	2	4
Learning rate	0.00003	0.0015	0.003	0.0015	0.0015	0.0005	0.0025	0.002	0.001	0.001
LNF	y	n	n	n	n	n	n	n	n	n
Epochs	7700	500	270	500	500	1200	300	325	450	450

Appendix C: Extended Positional Encoding Explanation

Self-attention In addition, while the self-attention mechanism can measure the similarity between any two vectors from different time steps is not explained by the Transformer model's literature. For this part, we can dig a bit deeper into the dot product-based attention score calculation. From Equation (4.4), the attention score is calculated from the scaled dot product of Q and K matrix. Thus, we denote $S \in \mathbb{R}^{d_{\text{model}} \times d_{\text{model}}}$, then we have:

$$S = QK^T \quad (64)$$

For each $s_{i,j}$ in S,

$$s_{i,j} = q_i \cdot k_j = \|q_i\| \cdot \|k_j\| \cdot \cos \theta \quad (65)$$

Where $q_i, k_j \in \mathbb{R}^{d_{\text{model}}}$ are vectors from Q and K. From Equation (65), the dot product can measure the similarity of two vectors via directional similarity. In other words, when vectors with similar directions, the dot product value is high, which means the two vectors in different time steps have higher similarity, while when the vectors with opposite directions, the dot product value is low, which means the two vectors in different time steps are irrelevant. Two orthometric vectors dot product value is zero. $\sqrt{d_{\text{model}}}$ is the scale for the dot product value to moderate the anomalies. Since the attention weights are computed by the softmax function, higher weights are distributed to vectors in time steps that have higher similarities.

However, the dot product measures the similarity between vectors in different time steps, but without positional codes, the self-attention mechanism cannot identify sequential information. For example, it recognizes ‘I love cats’ and ‘cats love me’ as the same sentence, because it cannot detect different sequential meanings. Thus, when positional encodings are added in, the attention score is altered as:

$$Q = W^Q(X + PE), \quad K = W^K(X + PE) \quad (66)$$

$$s_{ij} = (W^Q X_i + W^Q PE_i) \cdot (W^K X_j + W^K PE_j)^T \quad (67)$$

$$= (W^Q X_i)(W^K X_j)^T + (W^Q X_i)(W^K PE_j)^T \quad (68)$$

$$+ (W^Q PE_i)(W^K X_j)^T + (W^Q PE_i)(W^K PE_j)^T \quad (69)$$

Where $(W^Q X_i)(W^K X_j)^T$ conveys the original features’ interaction information, $(W^Q X_i)(W^K PE_j)^T$ and $(W^Q PE_i)(W^K X_j)^T$ conveys features and positional interaction information which represent how the Query features at one time step are influenced by the positional information of the Key at another time step, $(W^Q PE_i)(W^K PE_j)^T$ contains pure positional interaction information. Therefore, it is more advanced and efficient than traditional statistical models since this attention mechanism simultaneously captures differences of lagged inputs, and features in this case.

Cross-attention Analogous to the self-attention mechanism introduced in Section 4.4.1.3.2, cross-attention in the Transformer is more powerful than self-attention since the Query from the decoder enables the cross-attention to capture the spatial-temporal information from a fuller angle. Concretely,

$$Q_{de} = W^{Q_{de}}(Y + PE_{de}), \quad K = W^K(X + PE_{en}) \quad (70)$$

$$s_{ij} = (W^{Q_{de}} Y_i + W^{Q_{de}} PE_{de,i}) \cdot (W^K X_j + W^K PE_{en,j})^T \quad (71)$$

$$= (W^{Q_{de}} Y_i)(W^K X_j)^T + (W^{Q_{de}} Y_i)(W^K PE_{en,j})^T \\ + (W^{Q_{de}} PE_{de,i})(W^K X_j)^T + (W^{Q_{de}} PE_{de,i})(W^K PE_{en,j})^T \quad (72)$$

Where $(W^{Q_{de}} Y_i)(W^K X_j)^T$ demonstrate the spatial-temporal information of input X and output Y . $(W^{Q_{de}} Y_i)(W^K PE_{en,j})^T$ demonstrates the interaction of stock returns and the positional information of the encoder, while $(W^{Q_{de}} PE_{de,i})(W^K X_j)^T$ demonstrates the interaction of decoder positional information and input X . $(W^{Q_{de}} PE_{de,i})(W^K PE_{en,j})^T$ logs the similarity of positional information from the encoder and decoder.

Appendix E: Extended Introduction of MLP Autoencoder

The feed-forward neural network module in the Transformer encoder block or decoder block is a single-layer MLP autoencoder NN structure. Also, the MLP autoencoder is the simplest autoencoder structure among the autoencoder family. Figure 5 shows an example of the single-layer MLP autoencoder module. It contains one encoder block and one decoder block as well. All nodes in the input layer or output layer are connected with the nodes of the latent layer or bottleneck layer (blue dots layer in Figure 5, which attaches encoder and decoder blocks). If $x_1, \dots, x_5$ are the input vectors, while $y_1, \dots, y_5$ are output vectors, the matrix of the input could be denoted as X , where $X = [x_1, \dots, x_5]$, and the output could be denoted as Y , where $Y = [y_1, \dots, y_5]$. Further, the input layer to the latent layer can be presented as:

$$Z = g(WX + b) \quad (73)$$

If Z is assumed to be the output of the latent layer and $Z = [z_1, \dots, z_3]$. Here, $W \in R^{3 \times 5}$ is the weight matrix and $b \in R^{3 \times 1}$ is the bias vector. $g(\cdot)$ is the activation function for the encoding process such as ReLU.

From the output of the latent layer to the decoder, it can be presented as:

$$\hat{Y} = g'(W'Z + b') \quad (74)$$

Where $W' \in R^{3 \times 5}, b' \in R^{5 \times 1}$ are weight matrix and bias vector for the decoder, while $g'(\cdot)$ is the activation function for the decoding process. $g'(\cdot)$ and $g(\cdot)$ can be identical or distinct.

Generally, the encoder is a method to project $X \in R^n$ to latent space (latent layer) output

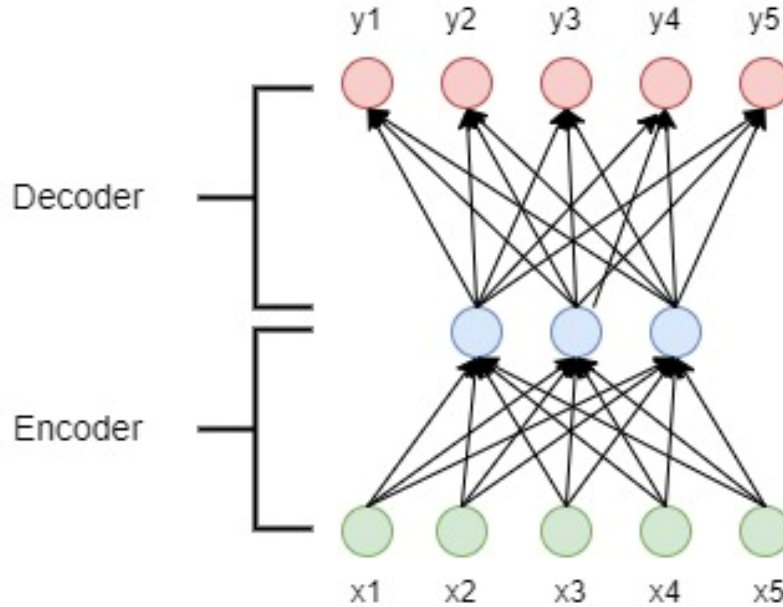


Figure 5: An example of a single-layer MLP autoencoder NN structure. The green dots represent the input vectors, while the red dots represent the output vectors. The blue dots indicate the neurons of the latent layer. $x_1, \dots, x_5$ are the inputs and $y_1, \dots, y_5$ are the outputs. The entire structure is a feedforward neural network.

$Z \in R^m$, hence a generalized encoder with multiple hidden layers can be simplified as:

$$Z = f_{\text{en}}(X; \theta_{\text{en}}) \quad (75)$$

Where $f_{\text{en}}(\cdot)$ is the projection form of the encoder, θ_{en} presents the parameters of the encoder which contain the weight matrix and bias vectors. If the projection form is MLP, the multi-hidden-layer encoder can be extended as:

$$h^{(1)} = \sigma^{(1)} \left(W^{(1)}X + b^{(1)} \right) \quad (76)$$

$$h^{(2)} = \sigma^{(2)} \left(W^{(2)}h^{(1)} + b^{(2)} \right) \quad (77)$$

$$\vdots$$

$$Z = h^{(L)} = \sigma^{(L)} \left(W^{(L)}h^{(L-1)} + b^{(L)} \right) \quad (78)$$

Where $h^{(L)} \in R^{d_L \times 1}$ is the L th hidden layer output, and $\sigma^{(L)}(\cdot)$ is the L th hidden layer's activation function. $W^{(L)} \in R^{d_L \times d_{L-1}}$ and $b^{(L)} \in R^{d_L \times 1}$ are the weight matrix and bias vector in the L th hidden layer.

Similarly, the decoder projects the encoder's output $Z \in R^m$ to $\hat{Y} \in R^n$:

$$\hat{Y} = f_{\text{de}}(Z; \theta_{\text{de}}) \quad (79)$$

The $f_{\text{de}}(\cdot)$ is assumed MLP projection, θ_{de} presents the parameters of the decoder which contain the weight matrix and bias vectors. Furthermore, the decoder with hidden layers can be generalized as:

$$h^{(L+1)} = \sigma^{(L+1)} \left(W^{(L+1)}Z + b^{(L+1)} \right) \quad (80)$$

$$h^{(L+2)} = \sigma^{(L+2)} \left(W^{(L+2)}h^{(L+1)} + b^{(L+2)} \right) \quad (81)$$

$$\vdots$$

$$\hat{Y} = h^{(2L)} = \sigma^{(2L)} \left(W^{(2L)}h^{(2L-1)} + b^{(2L)} \right) \quad (82)$$

$W^{(2L)} \in R^{d_{2L} \times d_{2L-1}}$ and $b^{(2L)} \in R^{d_{2L} \times 1}$ are the weight matrix and bias vector in the L th hidden layer. All activation function θ here employs the ReLU function.

Corresponding to naïve MLP models, the autoencoder can be estimated by minimising the loss function such as mean square error (MSE) via the method of Stochastic Gradient Descent (SGD) with the Adaptive Moment Estimation optimizer (Adam) as well. The full presentation of the simplified autoencoder can be written as:

$$\hat{Y} = f_{\text{de}}(f_{\text{en}}(X; \theta_{\text{en}}); \theta_{\text{de}}) \quad (83)$$

And the parameter θ_{en} , θ_{de} can be estimated via minimizing the loss function:

$$\min_{\theta_{\text{en}}, \theta_{\text{de}}} L(Y, \hat{Y}) \quad (84)$$