The University of Sheffield

Improving the Participation of Resource-Constrained Devices in Federated Learning

Hongrui Shi

Supervisor: Dr Po Yang; Dr Valentin Radu

A thesis submitted for the degree of Doctor of Philosophy in the School of Computer Science

23rd February 2025

To my parents.

Declaration

I, Hongrui Shi, hereby declare that this thesis has been composed solely by myself and that it has not been submitted, in whole or in part, in any previous application for a degree. Except where states otherwise by reference or acknowledgement, the work presented is entirely my own.

Acknowledgements

I would first like to thank my supervisors, Dr Po Yang and Dr Valentin Radu, for their wholehearted guidance and support throughout my PhD journey. Their dedicated mentorship has been instrumental in helping me achieve my goals and complete this research. Beyond their academic guidance, they have also served as life mentors, encouraging me to explore and practice, imparting valuable knowledge and experiences, and fostering opportunities and connections for my professional development. Their influence will undoubtedly benefit me throughout my life.

My appreciation extends to my friends and colleagues in the Pervasive Computing research group. I thoroughly enjoyed the moments spent discussing and collaborating with them, as well as the leisure times we shared. They have made my PhD journey more vibrant and enjoyable.

Lastly, I express heartfelt thanks to my beloved parents, wife and my daughter. Their unwavering love and support have been my sanctuary during the challenging moments of my PhD journey. A warm hug from my wife and daughter when I return home remains the best remedy for all my problems.

Abstract

Federated Learning (FL) has achieved huge successes in training machine learning models on remote devices. However, FL traditionally relies on devices with equal and sufficient computing capabilities. Resource-constrained devices, often termed stragglers, struggle to contribute their knowledge to FL due to limited computational resources. This issue significantly renders FL performance sub-optimal, especially when applied at large scales, as stragglers can offer unique data perspectives that other participants cannot provide. Therefore, it is important to alleviate the straggler issue by enabling the participation of resource-constrained devices that would otherwise be declared stragglers.

This research addresses the straggler issue by advancing efficient machine learning approaches to improve participation of resource-constrained devices in FL. Notably, three efficient machine learning approaches are explored in this thesis: 1) partial model training; 2) reduced-size models; 3) active data selection. Identified research gaps limiting the efficacy of existing FL works are addressed through proposed methods: 1) few-shot fine-tuning; 2) attention transfer and metadata training; 3) clustering-based and entropy-based data selection. Novel FL algorithms and worthy insights are delivered through this research, including: 1) few-shot fine-tuning not only reduces workloads on stragglers but also accelerates FL convergence, reducing energy consumption on resource-constrained devices. 2) attention transfer and metadata training enhance knowledge transfer from custom-size client models to the global model, improving the efficacy for addressing the straggler issue. 3) entropy-based data selection improves both learning efficiency and generalisation ability, achieving better FL performance with less resource consumption on small devices.

This research opens the field for other cross-strategy approaches to address the bottleneck of training large models on resource-constrained devices. In light of the emerging large language models, many current devices that generate user data are deemed stragglers if efficient machine learning approaches for FL will not be considered moving forward.

Contents

1 Introduction **1**

1.1 What is Federated Learning? 2

1.2 Straggler Issue 4

1.3 Addressing the Straggler Issue 6

1.4 Identifying the Research Gaps 8

1.4.1 Research Gaps 8

1.4.2 Major Research Questions 10

1.4.3 Proposed Methods to Approach the Identified Major Research Questions 10

1.4.4 Research Paths to Bridge the Research Gaps 12

1.4.5 Research Contributions 12

1.5 Thesis Outline 15

1.6 Publications 17

2 Preliminaries and Related Work **18**

2.1 Preliminaries 18

2.1.1 Standard Federated Learning Workflow 18

2.1.2 Federated Learning Scenarios 21

2.2 Efficient Machine Learning Methods for Tackling the Straggler Issue 25

2.2.1 Straggler Issue Deteriorates FL Performance 25

2.2.2 Partial Model Training 28

2.2.3 Custom-size Client Models 29

2.2.4 Active Data Selection 32

2.3 Related Work on Model Shift Mitigation 34

2.3.1 Model Shift Aggravates Straggler Issue 34
2.3.2 Regularising Local Updates 37
2.3.3 Advanced Federated Optimisation 40
2.3.4 Formulating Centralised Learning on the Server 41
2.4 Research Gaps in Approaches for Addressing the Straggler Issue . . 42
2.5 Summary 44

3 Joint Experimental Setup 46

3.1 Datasets 46
3.2 Baselines and Models 48
3.3 Evaluation Metrics 49
3.4 Summary 50

4 Mitigating Model Shift in Partial Model Training 51

4.1 Introduction 51
4.1.1 Research Questions to Address 51
4.1.2 Proposed Methods and Rationale 52
4.1.3 Introduction to FedFSC 53
4.2 Preliminary 56
4.3 Few-shot Learned Classifier is Less Biased 58
4.3.1 Few-Shot Learning with the Frozen Feature Extractor 58
4.3.2 Training a Less Biased Classifier with Few-Shot Updates . . 58
4.4 Federated Learning with Few-shot Learned Classifier 63
4.4.1 FedFSC+: Utilising Stragglers 64
4.5 Experiments 68
4.5.1 FedFSC with Full Client Participation 68
4.5.2 FedFSC+ with Partial Client Participation 73
4.5.3 Regularising Local Updates with Few-Shot Learned Classifier 76
4.5.4 Ablation Studies on FedFSC and FedFSC+ 80
4.5.5 Model Architecture Variation 85
4.6 Summary 87

5 Improving Knowledge Transfer to the Global Model 89

5.1 Introduction 89

5.1.1 Research Questions to Address 89

5.1.2 Proposed Methods and Rationale 90

5.1.3 Transferring Knowledge with Attention Transfer and Metadata Training 92

5.1.4 Improving Knowledge Transfer with Prototype Feature Maps 93

5.2 Preliminaries 96

5.2.1 Knowledge Distillation with Attention Transfer 96

5.2.2 Refining the Global Model with Metadata Training 100

5.3 Federated Global Updates with Attention Transfer 102

5.3.1 Local Updates of FedAT 102

5.3.2 Global Updates of FedAT 104

5.4 FedAT Experiments 109

5.4.1 Single Client Scenario 109

5.4.2 Multiple Clients Scenario 119

5.5 Federated Learning with Aggregated Prototype Feature Maps . . . 123

5.5.1 Local Updates of FedKAD 125

5.5.2 Global Updates of FedKAD 126

5.6 FedKAD Experiments 129

5.6.1 Compressing Metadata with Local Prototype Feature Maps . 131

5.6.2 Experiment Results 131

5.6.3 Mitigating Negative Knowledge Transfer with Prototype Feature Maps 132

5.7 Summary 136

6 Addressing the Straggler Issue with Active Data Selection 138

6.1 Introduction 138

6.1.1 Research Questions to Address 138

6.1.2 Proposed Methods and Rationale 139

6.1.3 Selecting Latent Feature Maps with K-means Clustering . . 141

6.1.4 Selecting Client Data with Shannon Entropy 142
6.2 Selecting Feature Maps to Train the Global Model 144
6.2.1 K-means Clustering for Data Selection 145
6.2.2 Local Updates of FedSplit with Feature Maps Selection . . . 147
6.2.3 Global updates of FedSplit with Feature Maps Selection . . 149
6.3 FedSplit Experiments 150
6.3.1 Hyperparameter Studies for K-means Clustering 151
6.3.2 FedSplit Performance with Feature Maps Selection 156
6.4 Entropy-based Data Selection with Pretrained Global Model 159
6.4.1 Pretraining a Global Model that Resists Model Shift 159
6.4.2 Federated Fine-tuning with Entropy-based Data Selection . 165
6.5 FedFT and FedFT-EDS Experiments 172
6.5.1 Evaluation on CIFAR-10 and CIFAR-100 173
6.5.2 Evaluation on GSC 183
6.6 Summary 184

7 Conclusions 187
7.1 Summary of Thesis 187
7.2 Limitations and Future Research 189
7.2.1 Limitations 189
7.2.2 Future Research 190

List of Figures

1.1 Three approaches to address the straggler issue. 7
1.2 An overview of the three sub-research paths. 13

2.1 Standard FL workflow. 19
2.2 Cross-silo FL and cross-device FL. 22
2.3 Straggler dropout 26
2.4 Degraded FL performance introduced by the straggler dropout. . . 27
2.5 Label skew and quality skew. 36
2.6 The model shift problem. 37
2.7 Degraded FL performance with heterogeneous data. 38

3.1 Non-IID local data visualisation. 48

4.1 FedFSC workflow. 55
4.2 Few-shot learning paradigm. 57
4.3 Comparing the client performance of few-shot updates and full updates. 61
4.4 Comparing the predictions of few-shot updates and full updates. . . 62
4.5 Training time of few-shot updates. 66
4.6 Learning curves of FedFSC. 72
4.7 Regularising local updates with the baseProx and FSCProx (test accuracy of the global model). 80
4.8 Regularising local updates with the baseProx and FSCProx (training losses of the global model). 80
4.9 Ablation study 1: varying $Diri(\alpha)$. 82
4.10 Ablation study 2: varying the few-shot number. 83

4.11 Ablation study 3: varying the number of clients performing few-shot updates (f_{fs}). 84

5.1 FedAT overview. 94
5.2 Visualising attention maps. 99
5.3 S-T learning with AT. 100
5.4 Training the global model with feature maps extracted from custom-size client models. 101
5.5 The global updates and local updates of FedAT. 103
5.6 Client model split and feature extraction. 104
5.7 FedAT: 3-step optimisation on the server. 105
5.8 AT from the aggregated global model to individual client models. . 109
5.9 FedAT: step 2 and step 3 performance on the server. 114
5.10 Global model performance across the 3 steps of global updates with varying metadata proportions. 118
5.11 The formulation of the global prototype feature maps for the "cat" class. 128
5.12 Comparing FedKAD and FedMD under strong data heterogeneity. . 134
5.13 The effect of prototype feature maps regularisation in KD. 136

6.1 Selecting feature maps to update the upper layers of the global model.146
6.2 Cumulative Explained Variance plot of PCA. 152
6.3 Attention map reconstruction. 153
6.4 Selected feature maps visualised through attention maps for the airplane class. 154
6.5 Visualisation of the clustering results with K-means. 155
6.6 Confusion matrices showing clustering results. 156
6.7 Heatmaps of the CKA similarity in the scenario of 10 clients and $Diri(0.1)$. 162
6.8 Heatmaps of the CKA similarity in the scenario of 10 clients and $Diri(0.5)$. 163

6.9 Averaged CKA similarity at different layer levels of the locally trained models 164
6.10 Entropy-based data selection. 168
6.11 Learning curves of FedFT-EDS and baselines. 175
6.12 Comparing the learning efficiency of FedFT-EDS and baselines. . . 176
6.13 Learning curves of FedFT-EDS and baselines. 180
6.14 Comparing FedFT-RDS and FedFT-EDS with 10% or 50% local data selection for local updates. 181
6.15 The learning efficiency of FedFT-RDS, FedFT-EDS, FedFT-ALL, and the baselines in the 100-client scenario. 182

List of Tables

4.1 Hyperparameters used by the baseline/baseline+ and FedFSC/FedFSC+. 69
4.2 FedFSC Validation 69
4.3 FedFSC with full client participation. 71
4.4 FedFSC(FedAvg) and FedAvg comparison on different seeds. 74
4.5 FedFSC and FedFSC+ with partial client participation. 77
4.6 FedFSC and FedFSC+ with partial client participation. Continue from Table 4.5. 78
4.7 Regularising local updates with FSCProx. 81
4.8 Ablation study 4: utilising the fully updated classifier. 85
4.9 Evaluating FedFSC+ with different model types. 87

5.1 Baseline performance in the single client scenario on CIFAR-10. . . 111
5.2 AT from the client model improves the global model performance trained on $\mathcal{D}_g$ at step 1. 112
5.3 The global model performance at step 1 and step 2 in the global updates. 116
5.4 Comparison of global model performance on all classes. 117
5.5 Configurations of custom-size WRN models over 20 clients in FedAT. 120
5.6 FedFSC Validation 120
5.7 Comparison of FedAT and FedAvg with varying straggler ratios. . . 122
5.8 The global model performance on absent classes in the generic data. 123
5.9 FedKAD Validation 129
5.10 Compressing metadata with prototype feature maps. 131
5.11 Comparing the performance of FedKAD and baselines. 133

5.12 Improved global model performance by adopting prototype feature maps. 134

6.1 Feature map dimensions across layers in the WRN-40-1 model. . . . 151
6.2 Test performance of the composed model in FedSplit. 157
6.3 The performance of FedSplit using feature maps extracted from different layer levels. 158
6.4 The pretrained global model improves the performance of the downstream FL task significantly. 160
6.5 The effect of pretraining. 163
6.6 FedFT-EDS Temperature Validation 173
6.7 FedFT-EDS Partial Model Validation 173
6.8 The global model performance of FedFT-EDS and baselines with 10 clients of full participation. 174
6.9 100 clients scenario with straggler simulations. 178
6.10 The performance of FedFT on GSC. 183

Nomenclature

AM	Attention Maps
ASR	Automatic Speech Recognition
AT	Attention Transfer
CEV	Cumulative Explained Variance
CKA	Centered Kernel Alignment
CM	Confusion Matrix
CNN	Convolutional Neural Network
DNN	Deep Neural Network
EM	Expectation-Maximum
EMR	Empirical Risk Minimisation
EWC	EmElastic Weight Consolidation
FFT	Fast Fourier Transform
FL	Federated Learning
FM	Foundation Model
FSL	Few-Shot Learning
FT	Fine-Tuning
GDPR	General Data Protection Regulation

GM	Global Model
GSC	Google Speech Command
IID	Identically and Independently Distributed
IoT	Internet of Things
KD	Knowledge Distillation
LLM	Large Language Model
LSTM	Long Short-Term Memory
MTL	Multi-Task Learning
NAS	Neural Architecture Search
NLP	Natural Language Processing
PEFT	Parameter-Efficient Fine-Tuning
S-T	Student-Teacher
t-SNE	t-Distributed Stochastic Neighbor Embedding
WRN	Wide ResNet

Chapter 1

Introduction

Smart devices, such as mobile phones, wearable devices, Internet of Things (IoT) have surged in recent years. They generate a wealth of user data on a daily basis, offering unique insights into various aspects into our daily lives. These on-device data can provide a variety of opportunities for machine learning applications across various domains, such as human activity recognition, automatic voice recognition, recommendation systems, health monitoring, etc, enabling businesses and organisations to develop data-driven tools and deliver enriched user experiences.

However, the widespread utilisation of on-device data in machine learning has been hindered by two primary factors, data privacy concerns and computational limitations of devices. On one hand, user-generated data commonly contains sensitive information about individuals. While traditional centralised machine learning collects and stores user-generated data, people are getting increasingly concerned that their sensitive information could be at significant risk. To protect user information, AI service providers have to invest a fortune on security measures to guarantee data safety. According to the government report of AI activity in UK businesses in 2022 (Capital Economics and DCMS, 2022), the challenges of reaching sufficient training data and the cost to follow the privacy laws with regards to personal data, such as General Data Protection Regulation (GDPR) (European Parliament and Council of the European Union, 2016), have become major obstacles for adopting machine learning techniques in small or medium-size enterprises. On the other hand, developing machine learning models, particularly deep learning models, requires substantial computational resources, which can be challenging to satisfy resource-constrained devices typically found on the user side. Consequently, machine learning applications miss out the unique perspective offered by a large amount of unique user-generated data.

Federated Learning (FL) (McMahan et al., 2017) has emerged as a promising machine learning paradigm to approach the challenge of leveraging user-generated data on resource-constrained devices to train models while preserving data privacy.

1.1 What is Federated Learning?

Federated Learning (FL) is a distributed machine learning paradigm that pushes the training of machine learning models to the edge without revealing user data to third parties. Smart devices with suitable computational resources and organisations with siloed data, such as banks, hospitals, and research institutions are typical FL participants at the edge. FL employs a server on the cloud to orchestrate the model training on participants, collect their training results and update a *global model* (GM) as the final model for future deployment. The formal notion of federated learning originates from McMahan et al. (2017): "We term our approach *Federated Learning*, since the learning task is solved by a loose federation of participating devices (which we refer to as *clients*) which are coordinated by a central *server*". In this groundbreaking work, the model training on the client is termed as *local updates*, and the update of parameters of the global model on the server is referred to as *global updates*. These originally defined terms, including "client", "server", "local updates", and "global updates", are consistently used in this thesis.

The increasing popularity of FL is mainly driven by its ability to tackle practical challenges of training machine learning models, particularly the advancement of data demands for large model training and privacy concerns associated with collecting sensitive user data (Li et al., 2020a, 2021c). Deep neural networks (DNN) with millions or even billions parameters, such as convolutional neural networks (CNN) (LeCun et al., 2015) and the Transformer (Vaswani, 2017), require vast amounts of training data to be effective in practical machine learning tasks like computer vision and natural language processing (NLP). FL enables model training to access a wealth of user-generated data while ensuring this data never leaves their devices and is only used to perform local updates. Moreover, FL broadly adopts a range of defense techniques, such as robust aggregation (Muñoz-González et al., 2019), differential privacy (El Ouadrhiri and Abdelhadi, 2022), and encryption (Liu et al., 2021b), to further safeguard client-extracted features from adversarial attacks (Kumar et al., 2023). Due to the decentralised nature of FL, even trained models and extracted features on devices are not often safe to share. Notably, leveraging model inversion techniques (Haim et al., 2022; Fang et al., 2024), malicious attackers are able to recover sensitive user information by studying model parameters and features during training, communication, and server aggregation stages (Khowaja et al., 2022; Yin et al., 2023). The integration of advanced defense techniques enables FL to further improve its privacy-preserving capability while scaling to a large population of participants. By doing so, FL encourages more users to participate in model training by addressing their privacy concerns, as it does not reveal user information to any third parties. Consequently, FL is able to transform the machine learning paradigm from data scarcity into data richness.

Impressive successes have been recently achieved by FL. Google was among the pioneers in utilising FL for training machine learning models on millions of mobile devices, including on-device item ranking, content suggestions, next word prediction on keyboards (Bonawitz et al., 2019; Chen et al., 2019a; Kairouz et al., 2021). Similarly, Amazon has employed FL to train its speech recognition system for Alexa applications (Dupuy et al., 2022). To assist large-scale federated learning in business, tech giants and startup companies have developed a number of popular FL frameworks, such as the Project Florida by Microsoft (Diaz et al., 2023), Flower (Beutel et al., 2020), FedML (He et al., 2020b). Despite its increasing popularity and significant successes, FL faces challenges inherent in decentralised devices and data.

To leverage decentralised devices and data without risking user privacy, FL sacrifices the control over training data and computing resources. In centralised machine learning, the training centre has authority over managing the training data effectively to train the model. For instance, samples collected for model training are drawn from the distribution close to the target distribution to ensure the learned model generalises well on the target task. In addition, training centres are typically equipped with extremely powerful machines assumed to have sufficient computing resources for model training.

In contrast, edge devices participating in FL display high diversity in both hardware and on-device data. The hardware variability among participants in FL is referred to as *system heterogeneity*, while on-device data variability is termed *data heterogeneity*. System heterogeneity leads FL to favour powerful devices capable of completing local updates quickly. Conversely, data heterogeneity results in the model shift problem in FL (Karimireddy et al., 2020; Wang et al., 2020a; Li et al., 2021b), where inconsistencies in local datasets across devices hinder FL convergence. Both system heterogeneity and data heterogeneity discourage resource-constrained devices from participating in FL, leading to the straggler issue and ultimately diminishing the overall performance of FL.

1.2 Straggler Issue

A critical problem that hinders the performance of federated learning is the *straggler issue*. In FL, stragglers are typically those resource-constrained devices that are less likely to contribute their learning results due to limited computational resources. Excluding contributions from stragglers leads to deteriorated FL performance in terms of global model generalisation, convergence, fairness, etc, particularly when stragglers generate and store valuable data essential for training a robust model. For instance, through their nature, battery powered devices are pervasive in places where other more powerful devices would not operate (e.g., remote sensing in agriculture, wildlife monitoring), so they can capture unique perspectives not shared by more powerful devices though critical for robust and generalisable models in FL. Formally, FL terms this problem as the straggler issue.

The straggler issue in FL is mainly attributed to the variability of hardware across devices, or the system heterogeneity. At a scale of hundreds of thousands or even millions of participants in FL, edge devices are highly diverse in terms of their hardware capabilities, making them unable to finish local updates in close time. To avoid running local updates infinitely, the FL server typically demands participants to complete their local updates within a hard-imposed deadline. For devices that cannot complete local updates before the deadline, their learning results are simply discarded by the server, making both their knowledge and computing efforts wasted and subsequently penalising the aforementioned FL performance. As such, there is a great need to tackle the straggler issue by reducing workloads on resource-constrained devices so that their participation is improved.

It is worth noting that the definition of resource-constrained depends on the learning task. Even powerful devices can become stragglers when dealing with heavily-parameterised models or large datasets. Therefore, in FL, stragglers are not necessarily resource-constrained devices. However, for simplicity, this thesis will not discriminate between stragglers and resource-constrained devices unless explicitly stated, due to the common sense that resource-constrained devices are highly likely to become stragglers.

In addition to the hardware constraints, straggler issue is further aggravated by the *model shift* problem (Li et al., 2020b). The model shift problem arises from data heterogeneity, characterised by non-Identically and Independently Distributed (non-IID) local data generated on the client side. Similar to centralised learning, the global optimisation objective of FL is to minimise the training losses on data across all participants. However, when the model is offloaded to the client side for training, it is optimised to minimise the training losses on local data. With non-IID local data, local updates could significantly deviate locally updated models from the global objective (Hsu et al., 2019; Li et al., 2020c; Wang et al., 2020a; Reddi et al., 2020). The deviation from the global objective significantly slows down FL convergence, imposing a heavier resource burden on stragglers, such as battery, bandwidth, connectivity, etc., throughout FL and consequently making them struggle more to contribute effectively to FL.

The straggler issue significantly diminishes the effectiveness and efficiency of FL compared to centralised machine learning. Recently, there are a number of

FL works studying the negative impacts of stragglers. Their findings show that FL under heterogeneous conditions, including both system and data heterogeneity, requires significantly more communication rounds to converge (Zhao et al., 2018; Hsu et al., 2019; Li et al., 2019c). Most importantly, the straggler issue introduces a substantial model performance gap between the federated learned model and the centralised learned model. Empirical studies reveal that the generalisation capability of the federated learned model can degrade by up to $4\times$ (Abdelmoniem et al., 2023) under extreme heterogeneous environments with substantial straggler dropout.

To this end, this thesis aims to contribute to research endeavours addressing the straggler issue, a primary obstacle in advancing FL applications at large scales. Particularly, this thesis focuses on advancing approaches that can effectively reduce workloads on stragglers and mitigate model shift, thereby significantly improving the participation of resource-constrained devices.

1.3 Addressing the Straggler Issue

This research aims to address the straggler issue, improving the participation of stragglers in training FL models. To address the straggler issue, it is critical to reducing the computational cost of on-device model training. Two key factors decide the computational cost: the number of trainable model parameters and the size of the training data (Cai et al., 2018; Zhou et al., 2020a). By reducing the number of model parameters, operations required for both forward and backpropagation are reduced, thereby decreasing the overall computing footprint during training. In centralised learning, several approaches have been proposed to achieve the reduction of training parameters, such as parameter-efficient fine-tuning (PEFT) (Hu et al., 2021; Han et al., 2024b), model pruning (Liu et al., 2017; Hoefler et al., 2021), and neural architecture search (NAS) (Zoph et al., 2018). These methods improve the training efficiency by minimizing the training parameters without compromising model competitiveness. Another effective strategy to reduce training cost is to train with less data. If a model can be learned using fewer but more informative sam-

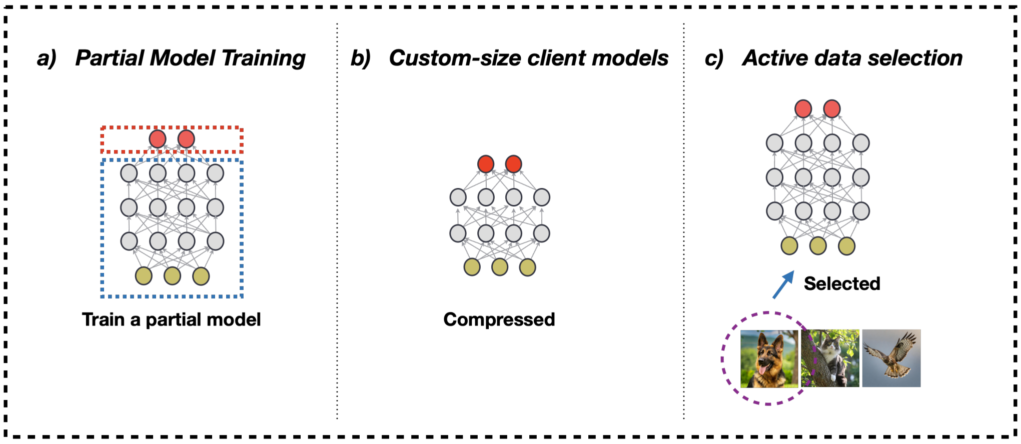


Figure 1.1: This thesis advances three efficient machine learning approaches to address the straggler issue. They are: 1) Partial model training; 2) Custom-size client models; 3) Active data selection.

ples—without sacrificing its performance—then the training budget is significantly lowered. This insight propels the advancement of active learning, where selectively querying the most valuable data can considerably reduce the volume of training data while preserving model accuracy (Mindermann et al., 2022; Yang et al., 2023; Li et al., 2022).

Through the literature review, three dominant machine learning approaches have been identified to reduce the training parameters and training data in the context of FL. And this thesis aims to fill the research gaps in these approaches. The three approaches are illustrated in Figure 1.1. They are:

- *Partial model training:* This approach reduces the computing burdens for stragglers by training a part of the model instead of the entire model. A handful of FL works, including FjORD (Horvath et al., 2021), HeteroFL (Diao et al., 2020), FedRolex (Alam et al., 2022), etc, have developed algorithms to tackle the straggler issue using this approach.

- *Custom-size client models:* This approach encourages straggler participation by deploying custom-size client models that match local computing capabilities. Given a smaller-sized model, both the forward pass and back-propagation leave less computing footprint during local updates. Typically, knowledge distillation (KD) (Hinton et al., 2015) is adopted to allow custom-size client models transfer their learned knowledge to the global model. Representative works on this track include FedMD (Li and Wang, 2019), DS-FL (Itahara et al., 2021), FedAUX (Sattler et al., 2021), etc.

- *Active data selection:* This approach alleviates workloads on stragglers by significantly reducing the size of training data for local updates. Active data selection constructs a subset of training data by identifying instances beneficial for learning. Without compromising learning performance, training efficiency is improved when feeding the model with fewer samples. Despite active data selection has been proven effective in centralised learning to reduce training costs (Mindermann et al., 2022; Yang et al., 2023; Li et al., 2022), its adoption in FL for addressing the straggler issue remains limited (Li et al., 2020a).

These three efficient machine learning methods play pivotal roles in reducing the workload of stragglers. Within them, critical research gaps are further identified in existing research. This thesis aims to bridge the research gaps by raising research questions, proposing methods to address research questions, and finally demonstrating the effectiveness and practical feasibility of proposed methods through rigorous evaluation. To this end, this thesis makes substantial contributions to the research endeavours enabling less capable devices to contribute to FL with their unique knowledge, particularly when smart devices are becoming increasingly ubiquitous and generating valuable on-device data that can be leveraged to advance machine learning applications.

1.4 Identifying the Research Gaps

1.4.1 Research Gaps

Our Chapter 2 will introduce critical research gaps identified in the approaches of partial model training, custom-size client models, and active data selection. They are summarised here as follows:

- While partial model training methods aim to reduce client workloads during local updates, they often fail to mitigate model shift. Model shift poses a significant challenge for partial model training methods, such as FjORD, HeteroFL, FedRolex, making them extremely difficult to train distributed models evenly across clients (Alam et al., 2022). Consequently, this leads to inconsistent performance and slow convergence, discouraging stragglers from participating in FL due to increased energy costs. Therefore, there is a pressing need to mitigate model shift for the partial model training approach to stabilise performance and accelerate convergence.

- The approach of custom-size client models faces challenges with knowledge transfer from client models to the global model. Notably, traditional parameter fusion methods, widely adopted in FL with homogeneous client models, become impractical when client models have varying sizes. As such, FL methods with custom-size client models, such as FedMD, DS-FL, and FedAUX, utilise KD to transfer client learned knowledge to the global model. However, these methods commonly rely on a single source of knowledge by simply aligning the output space across models. This single knowledge source is particularly unreliable given drifted client models, leading to degraded FL performance. Hence, it is necessary to advance the custom-size client models approach by improving the quality of knowledge transfer (Chen and Chao, 2020; Itahara et al., 2021), making them more practical for addressing the straggler issue.

- The efficacy of active data selection for addressing the straggler issue is understudied in FL (Li et al., 2020b). Two key research gaps are identified within limited works on this direction. First, primary studies, such as LoGo (Kim et al., 2023b) and F-AL (Ahn et al., 2024), aims to ease data labelling rather than reducing workloads on clients, thereby lacking the evaluation relevant to training efficiency. Second, methods like FLRD (Nagalapatti et al., 2022) commonly introduce substantial workloads to clients for identifying useful training instances rather than reducing workloads. Therefore, these studies fail to effectively address the straggler issue with active data selection, despite its established efficacy for improving training efficiency in centralised learning. To this end, essential research is needed to advance the approach of active data selection to effectively reduce workloads on stragglers in FL.

1.4.2 Major Research Questions

To fill the aforementioned research gaps, this thesis derives the following major research questions:

- With previous works missing to leverage partial model training to tackle both the straggler issue and model shift problem, the first major **R**esearch **Q**uestion (RQ) addressed in this thesis is "How can partial model training reduce workloads for stragglers and mitigate model shift simultaneously?"
- Given the adoption of custom-size client models to address the straggler issue, the second major RQ is "How can we improve the knowledge transfer from custom-size client models to the global model using multiple knowledge sources rather than a single source?".
- While previous data selection methods in FL less focus on addressing the straggler issue or even introduce additional workloads to clients, the third major RQ approached in this thesis is "How to leverage active data selection to address the straggler issue, and what impact does the active data selection bring to the global model performance under heterogeneous data?"

1.4.3 Proposed Methods to Approach the Identified Major Research Questions

Three methods are proposed and evaluated in this thesis to approach the research questions and bridge the research gaps. They are: 1) few-shot fine-tuning; 2) attention transfer and metadata training; 3) clustering-based and entropy-based data selection. They are summarised as follows:

- This thesis proposes few-shot fine-tuning to approach the first research question, tackling both the straggler issue and the model shift problem with partial model training. Particularly, few-shot fine-tuning allows clients to fine-tune the classifier locally in a few-shot learning manner (Chen et al., 2019b; Tian et al., 2020), facilitated by a universal feature extractor learned within FL loops. Few-shot fine-tuning can be particularly tailored for stragglers due to its computational efficiency. Most importantly, stragglers contribute few-shot updated classifiers that can effectively alleviate model shift, leading to accelerated FL convergence and reduced energy consumption on stragglers.

- To address the second research question, this thesis introduces attention transfer (AT) (Zagoruyko and Komodakis, 2016a) and metadata training as additional knowledge sources for transferring knowledge from custom-size client models to the global model. Notably, AT leverages attention maps formulated with latent representations to transfer attention styles between models. Metadata training uses feature maps extracted from custom-size client models as the metadata to refine the global model. Despite their success in centralised learning (Zagoruyko and Komodakis, 2016a; Sung et al., 2018; Wang et al., 2020b), both AT and metadata training have not been adopted in FL for enhancing knowledge transfer.

- Finally, clustering-based and entropy-based active data selection are proposed to address the third research question. They are chosen for their minimal computational burden on stragglers compared to previous active learning applications in FL (Sinha et al., 2019; Haussmann et al., 2019; Nagalapatti et al., 2022). Given selected training samples, this thesis hypothesises that we cannot only reduce workloads on stragglers but also overcome the penalisation on FL performance resulted from training data reduction. In this work, the clustering-based method performs K-means clustering (MacQueen et al., 1967) in latent representations to select the most representative samples for refine the global model. The entropy-based algorithm constructs subsets of training instances by ranking their importance with calculated Shannon entropy (Shannon, 1948). Additionally, a unique pretraining phase is introduced to initialise a global model resistant to model shift, further improving FL convergence.

1.4.4 Research Paths to Bridge the Research Gaps

This section outlines our research paths to bridge the gaps. As described in Section 1.3, three efficient machine learning approaches–partial model training, custom-size client models, and active data selection–are commonly employed in FL for easing the workload on clients. Section 1.4 identifies the research gap of each approach in existing literature. To address these gaps, this thesis establishes three sub-research paths. Each path aims to fill the corresponding research gaps by constructing research questions, proposing methods to answer these questions, and evaluating the efficacy of the proposed methods. Figure 1.2 is an overview of these three sub-research paths.

1.4.5 Research Contributions

By bridging the research gaps in the three efficient machine learning approaches in FL–partial model training, custom-size client models, and active data selection–this thesis contributes advanced algorithms and substantial findings to the research endeavours aimed at improving the participation of resource-constrained devices in FL. They are summarised as follows:

- This thesis demonstrates that the proposed few-shot fine-tuning not only reduces the training time on clients by 90% but also generates a classifier that is less biased under model shift compared to a fully updated classifier. Based on these discoveries, FedFSC (Shi et al., 2023b) is proposed to integrate few-shot fine-tuning into FL, constructing the global model with the few-shot updated classifier that is contributed by stragglers. FedFSC is observed to outperform popular FL baselines significantly on image, speech, text classifications in terms of both generalisation and convergence. FedFSC successfully fills the research gap by reducing workloads on stragglers and mitigating model shift within one framework, thereby minimising overall energy cost for stragglers. This work contributes one first-author publication at DistributedML 2023 (Shi et al., 2023b).

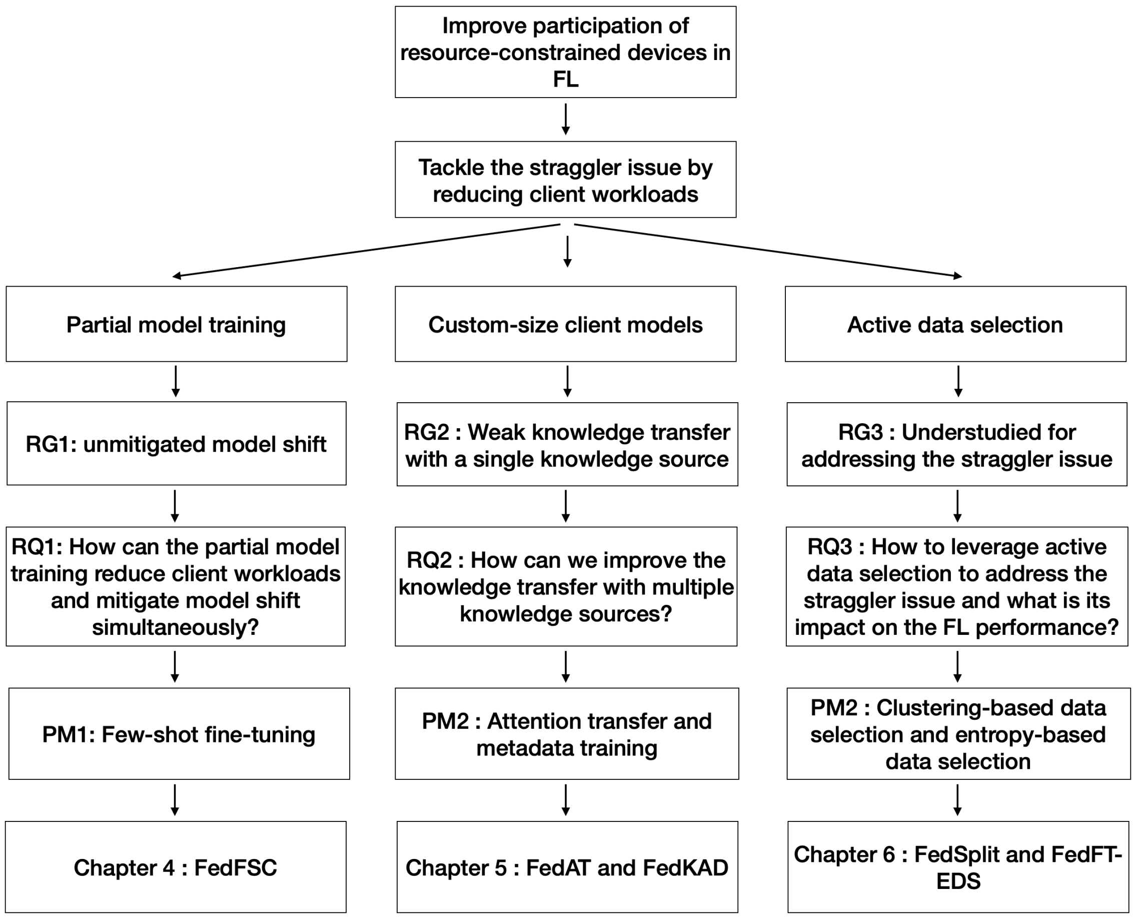

Improve participation of resource-constrained devices in FL
Tackle the straggler issue by reducing client workloads
Partial model training
Custom-size client models
Active data selection
RG1: unmitigated model shift
RG2 : Weak knowledge transfer with a single knowledge source
RG3 : Understudied for addressing the straggler issue
RQ1: How can the partial model training reduce client workloads and mitigate model shift simultaneously?
RQ2 : How can we improve the knowledge transfer with multiple knowledge sources?
RQ3 : How to leverage active data selection to address the straggler issue and what is its impact on the FL performance?
PM1: Few-shot fine-tuning
PM2 : Attention transfer and metadata training
PM2 : Clustering-based data selection and entropy-based data selection
Chapter 4 : FedFSC
Chapter 5 : FedAT and FedKAD
Chapter 6 : FedSplit and FedFT-EDS

Figure 1.2: An overview of the three sub-research paths. This research consists of three interconnected paths, all aiming to improve the participation of resource-constrained devices in FL. Each path approaches a specific research gap (RG) by formulating a research question (RQ), and proposing a method (PM) to answer it.

- With the adoption of custom-size client models, this thesis demonstrates that AT and metadata training effectively enhance the knowledge transfer from custom-size client models to the global model. Given clients with heterogeneous data, the global model is capable of generalising on unseen classes with the assistance of AT and metadata training (Shi and Radu, 2021). Moreover, it is found that AT has a regularisation effect to prevent the global model from overfitting to the generic data used for knowledge distillation, leading to improved global model performance on unseen classes. These studies contribute novel knowledge sources to improve the knowledge transfer in FL with custom-size client models, which resulted in a first-author publication at EdgeSys 2021 (Shi and Radu, 2021).

- This thesis further reveals that prototype feature maps–metadata in a compressed form, which preserve intra-class knowledge learned across custom-size client models–can be utilised as an additional source to correct negative knowledge transfer induced by model shift (Shi et al., 2023a). The delivered FL algorithm, namely FedKAD, further improves global model performance through regularising knowledge distillation with prototype feature maps. This work leads to a first-author publication at MLSys, EuroSys 2023 (Shi et al., 2023a).

- By adopting clustering-based data selection, the global model performance is only penalised by 4.35% on CIFAR-10, given that the size of metadata for refining the global model is significantly reduced by over 98%. It demonstrates that the global model can largely preserve its performance despite selecting only a small fraction of training data in order to address the straggler issue (Shi and Radu, 2022). This work contributes to a first-author publication at MLSys, EuroSys 2022 (Shi and Radu, 2022).

- With the assistance of pretraining, this thesis finds that the proposed entropy-based data selection successfully prevents the penalisation of FL performance induced by reduced training data. First, pretraining is found to generate a global model resistant to model shift, significantly accelerating FL convergence. The entropy-based data selection further demonstrates that over half of the client data is not beneficial for FL performance. It is evident that FL adopting entropy-based data selection not only triples client learning efficiency but also boosts global model performance by a large margin over baselines by training models on local instances selected based on their entropy information. With these insights, FL can leverage active data selection to effectively reduce workloads on stragglers without penalising its performance. The work is expected to be published later this year (2024).

1.5 Thesis Outline

The rest of this thesis is organised as follows:

Chapter 2 first introduces the fundamental workflow and categories of FL as preliminaries. Then related FL works that aim to solve the straggler issue and model shift problem are reviewed. Finally, research gaps in existing literature are identified in the final section.

Chapter 3 describes the joint experimental setup used in this thesis. We conduct our research on image classification and speech recognition tasks widely used in existing FL works so that our comparison with state-of-the-art methods can be straightforward.

Chapter 4 is the technical chapter for the first sub-research path. It presents the proposed method of few-shot fine-tuning and the developed FL algorithm, namely FedFSC, which is built on few-shot fine-tuning. FedFSC aims to address both the straggler issue and the model shift problem induced by the data heterogeneity. Chapter 4 answers the first research question in this thesis, “How can the approach of partial model training address the straggler issue and model shift problem simultaneously?”, and fills the research gap that previous partial model training methods fail to mitigate model shift.

Chapter 5 is the technical chapter for the second sub-research path. It introduces the proposed methods of attention transfer and metadata training to address the knowledge transfer problem in previous works that adopt custom-size client models. Particularly, two FL algorithms, FedAT and FedKAD, are delivered. In the first part of Chapter 5, FedAT demonstrates the efficacy of attention transfer and metadata training for transferring locally learned knowledge from custom-size client models to the global model. The second part develops FedKAD to mitigate negative knowledge transfer induced by model shift using prototype feature maps, which are metadata with a compressed form. Chapter 5 addresses the research question, "How can we improve knowledge transfer from custom-size client models to the global model with multiple knowledge sources?", and fills the research gap that relying on a single knowledge source at the output space of models leads to degraded FL performance.

Chapter 6 is the technical chapter for the third sub-research path. It depicts the proposed clustering-based data selection and entropy-based data selection. The clustering-based data selection leads to the design and evaluation of the proposed FL algorithm, FedSplit, which is introduced in the first part of Chapter 6. FedSplit demonstrates the penalisation on FL performance when reducing the size of training data. In the second part, FedFT-EDS, the FL algorithm that leverages entropy-based data selection is proposed and studied. FedFT-EDS shows it can prevent the performance penalisation and further boost FL performance with selected client data. Chapter 6 successfully approaches the research question, "How can active data selection addresses the straggler issue and what impact does the active data selection bring to the global model performance under heterogeneous data?", and fills the research gap that active data selection has not been proven to successfully reduce workloads on stragglers in FL.

Chapter 7 concludes the research presented in this thesis and proposes three promising future research directions.

1.6 Publications

This thesis has contributed four first-author conference papers and one journal manuscript currently being under review to this date. They are listed below in a chronological order.

- The work described in Chapter 5, Section 5.3 and Section 5.4 is published at EdgeSys '21, EuroSys conference, titled "Towards Federated Learning with Attention Transfer to Mitigate System and Data Heterogeneity of Clients", Hongrui Shi, Valentin Radu.
- The work introduced in Chapter 6, Section 6.2 and Section 6.3 is published at MLSys '22, EuroSys conference, titled "Data selection for efficient model update in federated learning", Hongrui Shi, Valentin Radu.
- The work detailed in Chapter 5, Section 5.5 and Section 5.6is published at MLSys '23, EuroSys conference, titled "Distributed Training for Speech Recognition using Local Knowledge Aggregation and Knowledge Distillation in Heterogeneous Systems", Hongrui Shi, Valentin Radu, Po Yang.
- The work depicted in Chapter 4, is published at DistributedML '23, CONEXT conference, titled "Lightweight Workloads in Heterogeneous Federated Learning via Few-shot Learning", Hongrui Shi, Valentin Radu, Po Yang.

In the next chapter, we will introduce relevant FL literature for addressing the straggler issue and identify the existing research gaps.

Chapter 2

Preliminaries and Related Work

This chapter first introduces the preliminaries of FL. The standard workflow of federated learning is described in Section 2.1.1, providing fundamental FL concepts used throughout this thesis. Section 2.1.2 outlines various FL scenarios and justifies the scenario targeted by this research. Previous works for tackling the straggler issue and model shift problem are introduced in Section 2.2 and Section 2.3 respectively. Finally, the research gaps in existing literature and the necessities to address them are described in Section 2.4.

2.1 Preliminaries

To better understand the background of federated learning, this section introduces the standard FL workflow and various FL scenarios.

2.1.1 Standard Federated Learning Workflow

In contrast to centralised learning, the training data of FL is scattered across clients without pooling in advance. For a indexed client k, its local training data is denoted as $\mathcal{D}_k$, which consists of a number of samples defined by $\{(x_k^{(i)}, y_k^{(i)})\}_1^{|\mathcal{D}_k|}$, where $x_k^{(i)}$ and $y_k^{(i)}$ are the i-th local instance and its associated label respectively. The model trained in FL is called the global model, denoted as M_g whose its parameters defined by w_g. The learning objective of FL is to find the w_g to minimise the combined

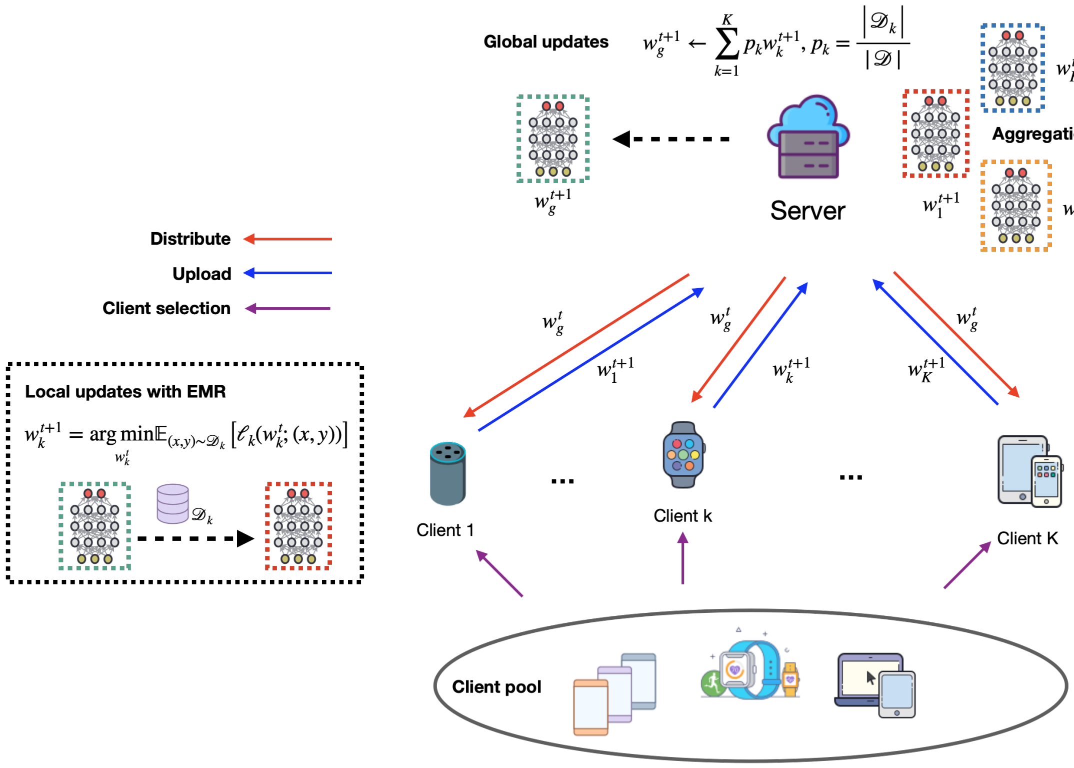

Global updates
$w_g^{t+1} \leftarrow \sum_{k=1}^{K} p_k w_k^{t+1}, p_k = \frac{|\mathscr{D}_k|}{|\mathscr{D}|}$
w_g^{t+1}
Server
w_1^{t+1}
w_K^{t+1}
w_k^{t+1}
Aggregation
Distribute
Upload
Client selection
Local updates with EMR
$w_k^{t+1} = \arg\min_{w_k^t} \mathbb{E}_{(x,y)\sim\mathscr{D}_k} [\ell_k(w_k^t; (x, y))]$
$\mathscr{D}_k$
w_g^t
Client 1
Client k
Client K
Client pool

Figure 2.1: Standard FL workflow. It comprises five steps at each iteration: 1) Client selection; 2) Model distribution;) Local updates; 4) Aggregation; 5) Global updates.

local losses across the clients described by Equation 2.1.

$$\arg\min_{w_g} \mathcal{L}(w_g) = \sum_{k=1}^{N} p_k L_k(w_g) \tag{2.1}$$

where, N is the size of the client pool, $L_k(w) = \mathbb{E}_{(x,y)\sim\mathcal{D}_k} [\ell_k(w; (x, y))]$ is the local empirical loss. $p_k = \frac{|\mathcal{D}_k|}{|\mathcal{D}|}$ is the coefficient to weight individual losses, determined by the fraction of local data over the entire client data $\mathcal{D} \triangleq \bigcup_{k\in[N]} \mathcal{D}_k$.

FedAvg (McMahan et al., 2017) is widely recognised as the standard workflow of FL (Kairouz et al., 2021; Ye et al., 2023). Particularly, a server sits on the cloud to synchronise the federated learning with iterations. For each iteration, the main steps of FL are summarised as: 1) Client selection; 2) Model distribution; 3) Local updates; 4) Aggregation; 5) Global updates. This workflow is illustrated in Figure 2.1.

Client selection. Not all clients in the client pool are supposed to participate in the model training due to eligibility requirements. Thus, the server samples a set of clients from the pool every iteration to perform the model training. The size of the set of participant clients is denoted as K.

Model Distribution. At iteration t, the server distributes copies of the global model, M_g^t, to the selected clients. Once the model is distributed to the client side. We use the set $\{M_k^t\}_1^K$ to denote the client models. For an indexed client k, M_k^t is associated with the parameters denoted by w_k^t.

Local updates. Distributed models are trained locally on clients. The local training objective is the empirical risk minimisation (ERM) over local data. For client k, the local training objective is defined by Equation 2.2.

$$w_k^{t+1} = \arg\min_{w_k^t} \mathbb{E}_{(x,y)\sim\mathcal{D}_k} \left[\ell_k(w_k^t; (x, y))\right] \tag{2.2}$$

where, $\ell_k(\cdot)$ is the local loss function on client k. More concretely, we update w_k^t to w_k^{t+1} with stochastic gradient descent (SGD) under a local learning rate λ_l.

$$w_k^{t+1} \leftarrow w_k^t - \lambda_l \nabla_{w_k^t} \ell_k(w_k^t; \mathcal{D}_k), \tag{2.3}$$

Conventionally, local updates are performed for a fixed number of epochs, which is arbitrarily instructed by the server. The number of local epochs is denoted as E.

Aggregation. The server collects updated client models before a preset deadline. In an ideal scenario, all K clients are supposed to upload their updated models to the server. However, due to realistic hardware variability, resource-constrained clients could fail to complete local updates timely. Those clients that fail to deliver their updated models in time for the iteration deadline are called stragglers and end up not contributing to the global updates.

Global updates. The global updates are performed on the server to transfer client learned knowledge to the global model, commonly known as *knowledge trans-*

fer from client models to the global model. The server updates the global model parameters by aggregating the locally updated client models. In an ideal scenario without stragglers, the global updates on the server are defined by Equation 2.4.

$$w_g^{t+1} \leftarrow \sum_{k=1}^{K} p_k w_k^{t+1} \tag{2.4}$$

where, p_k is similar to the one defined in Equation 2.1, but is calculated based on the selected clients $p_k = \frac{|\mathcal{D}_k|}{|\mathcal{D}|}$ with $\mathcal{D} \triangleq \bigcup_{k\in[K]} \mathcal{D}_k$. Finally, w_g^{t+1} will be distributed to the client side to start the next FL iteration. As the server and clients communicate models repeatedly. The iteration is commonly referred to as the *communication round.*

2.1.2 Federated Learning Scenarios

When Federated Learning (FL) is formally proposed by McMahan and Ramage; McMahan et al. (2017), it emphasises the offload of machine learning to edge devices on a large scale (Kairouz et al., 2021). Since then, academics and industry have adapted FL to a range of scenarios, significantly expanding the family of FL.

Cross-device FL and cross-silo FL

Defined by the scale and properties of participants, Kairouz et al. (2021) categorises FL into the cross-device FL and the cross-silo FL. Figure 2.2 illustrates the differences between the *cross-device FL* and *cross-silo FL*. The cross-device FL typically involves a great number of edge devices, such as mobiles or IoTs, with the total number of FL devices potentially reaching billions. However, the participation rate of devices is often extremely low, with only a small fraction, for instance 10% or less, of the device population available for federated learning at any given time. Over the entire iterations of federate learning, a device may only be able to participate in a few or even fewer iterations. Computation and communication are considered as the primary bottlenecks for cross-device federated learning as the majority of the smart devices are equipped with limited computing resources (CPU and memory) and uses wifi or slower connections such as 4G (Li et al., 2020a). Representative

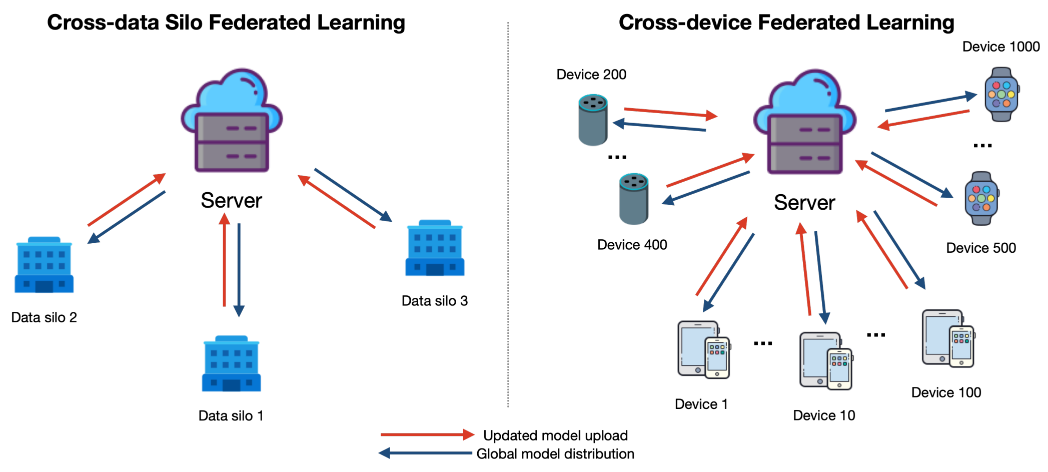


Figure 2.2: Cross-silo FL and cross-device FL.

industrial achievement from the cross-device FL are the next word prediction on the keyboard by Google (Hard et al., 2018; Chen et al., 2019a) and the Apple applications of the QuickType keyboard and "Hey Siri" vocal recogniser (Apple). Instead of collecting user-generated data to train their models, these companies utilize FL to train models directly on millions of user devices, subsequently aggregating these user-trained models to create a federated model deployed in their AI applications.

Conversely, the cross-silo FL is defined by its participants coming from a relatively small number of data centers compared to the numerous edge devices in the cross-device FL. Typical data centers are often organisations in the area of finance (Liu et al., 2023) or medical (Courtiol et al., 2019), who are poised to embrace machine learning applications to promote business or research but are not allowed to share their siloed customer data with third parties. By adopting federated learning, organisations can collectively train a machine learning model using their unique data with the privacy protected. In contrast to the cross-device FL, the computation and communication are often not major bottlenecks of the cross-silo FL as participants such as banks or medical institutes typically have much more powerful hardware than the edge devices. Some exemplary applications of the cross-silo FL in the industry are the drug discovery (Oldenhof et al., 2023), medical data segmentation (Courtiol et al., 2019), financial crime detection (IBM).

Horizontal FL, vertical FL, and federated transfer learning

Federated learning scenarios can be also categorised with respect to the data partition on the client side. Yang et al. (2019) classify FL into the *horizontal FL*, *vertical FL*, and federated *transfer* learning (Kairouz et al., 2021) based on the split of the client data. In the horizontal FL, data is assumed to be partitioned horizontally in the sample space but shares the same feature space. Training samples on different clients are typically not overlapped. On the other hand, the vertical FL emphasises the data partition in the feature space but shares overlapped sample space. Federated transfer learning is the hybrid scenario of horizontal FL and vertical FL, where the data across clients is not only split in the feature space but also in the sample space. In the cross-device FL, data split is commonly assumed to be horizontal rather than vertical. Conversely, both horizontal and vertical data split could practically be present in the cross-silo FL. Examples are businesses, such as banks and retail companies, could have non-overlapped customers and have access to different sets of customer information.

Personalised FL performance

The original FL and the main stream of follow-up FL studies focus on the generalisation of the global model on the learning task. The learned global model is deployed fore inference. However, most recently, a stem of studies put the personalised FL performance into the spotlight. The personalised FL performance highlights the generalisation of the federated learned model on individual clients rather than the global distribution (Tan et al., 2022a).

In the early works of personalised FL, Multi-Task Learning (MTL) (Zhang and Yang, 2021) and Meta Learning and (Thrun and Pratt, 2012) become particularly relevant techniques if we consider the learning problem in each client as a separate task. MOCHA (Smith et al., 2017) integrates MTL into FL framework to simultaneously learn a set of different weights for all the clients. However, the assumption of convex loss function in MOCHA limits its further applications. A potential exploration of applying MTL to learning personalised models is modelling the relationship between the tasks and clients. MTL learns a model per task. However, this does not necessarily mean each client should learn a distinct model if a subset of clients can be grouped as a learning task (Kairouz et al., 2021). On the other hand, Model Agnostic Meta Learning (MAML) (Finn et al., 2017) has stemmed from Meta Learning to learn a global model that can be used as an initialization for further learning a good model adaptive to a new task in only a few local gradient steps (Kairouz et al., 2021). Both Khodak et al. (2019) and Jiang et al. (2019) explore the relationship between MAML and FL to address the data heterogeneity challenge. The global update in FL is compared to meta-training an initial model in MAML and local updates performed by each client can be associated with the meta-testing to adapt the initial model to an individual local task, resulting in personalised client models. Recently, Per-FedAvg (Fallah et al., 2020), LG-FedAvg (Liang et al., 2020), KT-pFL (Zhang et al., 2021), FedPCL (Tan et al., 2022b), FedPAC (Xu et al., 2023), pFedHR (Wang et al., 2024a) are among the most representative works that advances personalised FL.

The Targeted FL Scenario

As the straggler issue tackled by this thesis practically exists in the cross-device FL scenario with horizontally split client data (Kairouz et al., 2021), this thesis sets up experiments to simulate this scenario. Global model performance is emphasised in this research rather than personalised FL performance. This is because the straggler issue primarily affects the global model performance due to device dropout during FL.

This thesis will solely focus on addressing the straggler issue through efficient machine learning approaches and does not intend to cover all research aspects in FL, as FL is a highly integrated research topic in machine learning (Ding et al., 2022; Ji et al., 2024). For instances, FL is intertwined with popular machine learning topics, such as unsupervised learning (Zhang et al., 2023a; Kim et al., 2023a), lifelong learning (Liu et al., 2019a; Yu et al., 2022), adversarial learning (Bhagoji et al., 2019; Zhang et al., 2019), model optimisation (Wang et al., 2021; Chen et al., 2023), and more. To protect user privacy, differential privacy (El Ouadrhiri and

Abdelhadi, 2022; Wei et al., 2020), cryptography (Mansouri et al., 2023; Zheng et al., 2023), secure aggregation (Bonawitz et al., 2016; Fereidooni et al., 2021) are also widely studied in FL. Finally, works on the device topology (Marfoq et al., 2020), communication protocols (Shahid et al., 2021) and systems (Khan et al., 2021) pave the way for delivering practical FLs. Each of these directions addresses various challenges in FL. This work specifically focuses on addressing the straggler issue.

2.2 Efficient Machine Learning Methods for Tackling the Straggler Issue

This section first demonstrates the impact of the straggler issue in FL. Then three efficient machine learning approaches for solving the straggler issue in current research are rigorously reviewed. They are: 1) partial model training; 2) custom-size client models; 3) active data selection.

2.2.1 Straggler Issue Deteriorates FL Performance

Clients selected to participate in a FL communication round train the downloaded global model for a few local epochs and are expected to finish before a deadline. But as our computing ecosystem flourishes with a wider spectrum of computing devices – such as in the Internet of Things (IoTs) space – system heterogeneity rifts a gap in the theoretical validation of FL, if more diverse devices are involved in FL.

Straggler issue. The straggler issue primarily arises from the mismatch between client hardware and the computational demand for training machine learning models, particularly deep learning models. Traditional FL distributes a uniform model for all participating clients and requires them to perform a fixed number of training epochs on the distributed model before a deadline. Some clients with limited computational resources cannot complete the required training within the deadline. Consequently, these clients are discarded by the server during the communication

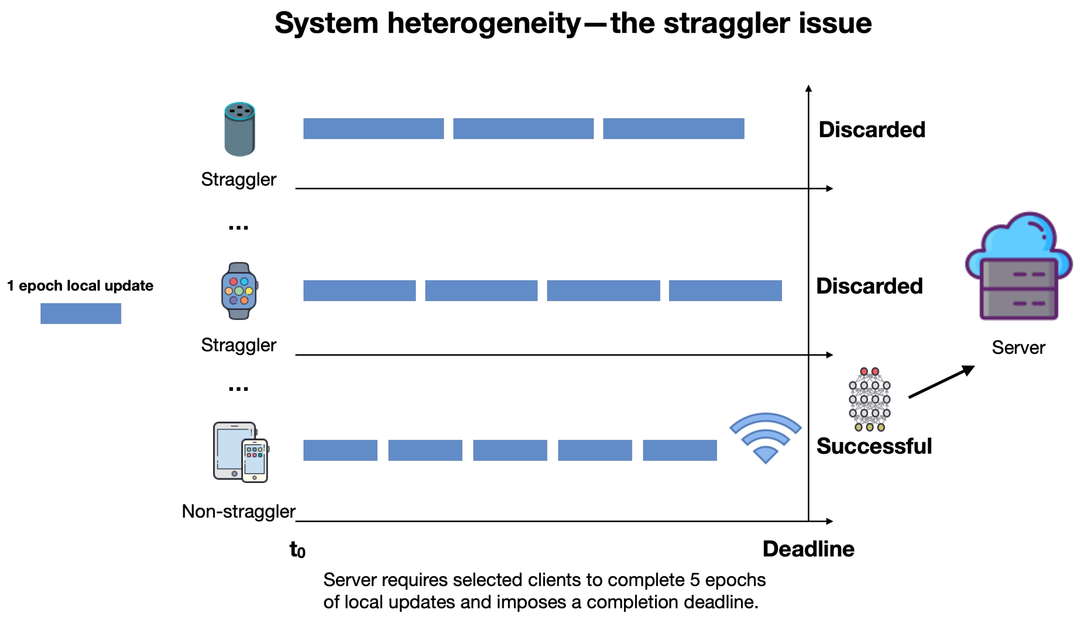
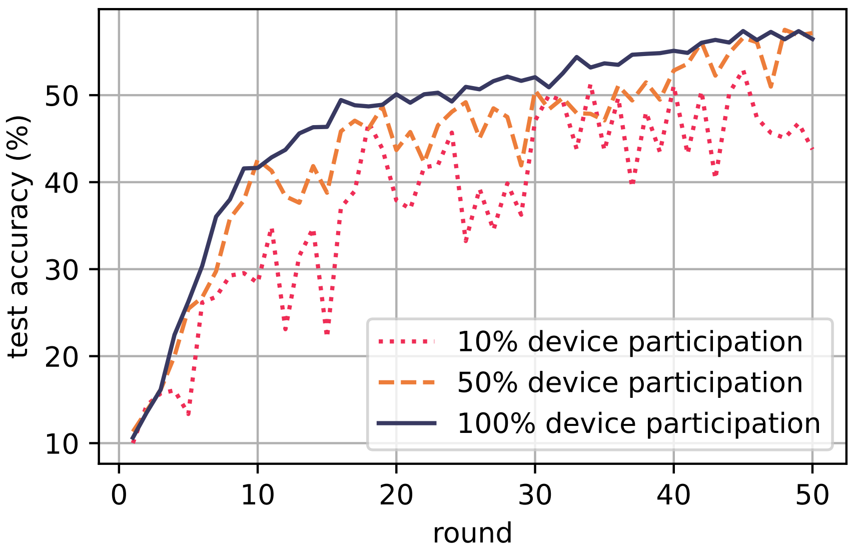

System heterogeneity—the straggler issue
1 epoch local update
Straggler
...
Straggler
...
Non-straggler
Discarded
Discarded
Successful
Server
t_0
Deadline
Server requires selected clients to complete 5 epochs of local updates and imposes a completion deadline.

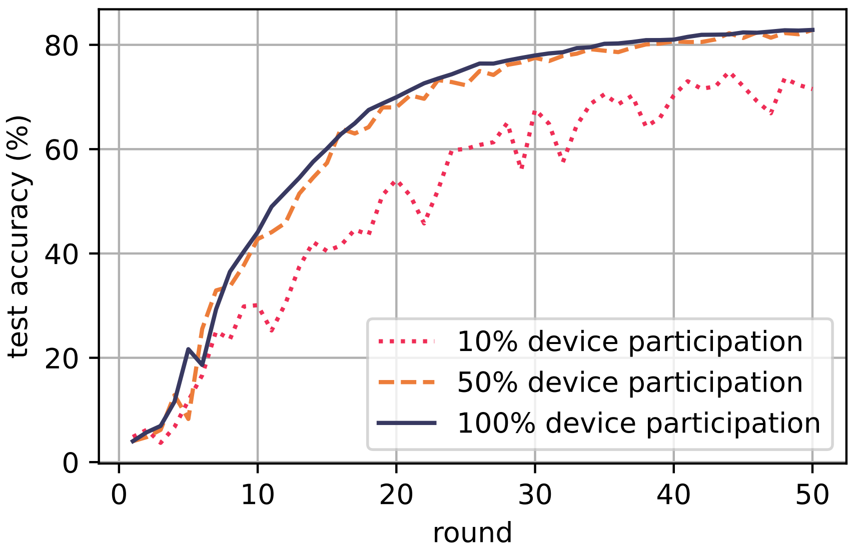
Figure 2.3: Stragglers are discarded by the server due to incomplete local updates. The diagram is a remade version of Figure 2 from Li et al. (2020a).

round, contributing no knowledge to FL. As stragglers often hold essential data for learning, their dropout compromises global model performance. Figure 2.3 illustrates the dropout of stragglers in FL. Figure 2.4 demonstrates how the straggler issue deteriorates global model performance in FL. The significant performance decrease mainly results from discarding the learning results from stragglers. With more devices being discarded, the global model performance is more heavily penalised. Further, it is observed that the straggler issue affects datasets differently in Figure 2.4. Given 50% device dropout (equivalent to 50% device participation), the GSC dataset displays more resilient performance compared to the CIFAR-10 dataset. Our hypothesis for this observation is that the GSC dataset is less complex than the CIFAR-10 dataset. More detailed discussions about the performance differences on GSC and CIFAR-10 are provided in Section 4.5.1.

Several research directions have been explored by the FL community to tackle the straggler issue. These include efficient machine learning approaches (He et al., 2020a; Diao et al., 2020; Horvath et al., 2021; Mei et al., 2022), asynchronous FL (Nguyen et al., 2022; Zhu et al., 2022), and one-round FL (Guha et al., 2019;



(a) CIFAR-10



(b) GSC

Figure 2.4: The straggler issue diminishes global model performance in FL. A total of 10 clients on CIFAR-10 and GSC datasets are simulated in FL. With more stragglers being dropped out, the learning curve of the global model is slower and more unstable.

Zhou et al., 2020b; Gong et al., 2021; Zhang et al., 2022a; Su et al., 2023).

This thesis narrows down its focus to the direction of efficient machine learning as it is one of the most well-studied and widely accepted approaches for addressing the straggler issue due to its high efficacy and competitive performance. Through the literature review, efficient machine learning in FL is summarised into three approaches: 1) partial model training: 2) custom-size client models; 3) active data selection. Related works of the three approaches are reviewed in the subsequent subsections.

2.2.2 Partial Model Training

Partial model training reduces workloads on clients by allowing them to train only a portion of the model rather than the full model. A series of FL works have proposed updating submodels of the global model on clients. These submodels are extracted in a way that fits local computational power, thereby improving the participation of less capable clients. Generally, submodels are formulated by masking units in hidden layers of the model. Once updated on clients, the updated model parameters can be fused into different parts of the global model on the server.

FjORD (Horvath et al., 2021) proposes an ordered dropout strategy to extract submodels from the global model and broadcast them to the client side. The submodel is obtained by dropping out inside units with ordered probabilities. The ordered probabilities come from a set of dropout probabilities broadcast to clients for deciding how many units of the model is pruned in accordance to their computational capabilities. Those pruned units do not participate in the local updates, thus reducing training cost on clients. FjORD compares its ordered dropout with random dropout, demonstrating better global model performance. In a very close study to FjORD, HeteroFL (Diao et al., 2020) diminishes the size of every hidden layer in the model with a fixed ratio to extract the submodel. The ratio is decided by the capability of the participating client. Server aggregates updated submodels and uses them to complete the global model. FedRolex (Alam et al., 2022) criticises that the parameters of submodels are unevenly trained in FjORD and HeteroFL, leading to a degraded global model performance under data heterogeneity. To address this problem, FedRolex introduces a rolling window for extracting submodels across communication rounds, enabling all parts of the global model to be exposed to different local data distributions, improving the global model performance in scenarios of strong data heterogeneity. Following FjORD and HeteroFL, InclusiveFL (Liu et al., 2022) adapts the submodel solution to large model learning on the edge, such as training the Transformer. FLANC (Mei et al., 2022) interprets the submodel extraction with tensor decomposition on model parameters. It proposes a more flexible tensor decomposition than FjORD and HeteroFL, where original parameters can be approximated by the multiply of two tensors, a neural basis tensor that is shared among all clients and an adjustable tensor that can adapt to client capability. Similarly to FedRolex for addressing the uneven model training for FjORD and HeteroFL, the neural basis tensor in FLANC is utilised to transfer the knowledge learned on all clients to every part of the global model.

2.2.3 Custom-size Client Models

Partial model training improves the training efficiency of local updates by reducing trainable parameters. Based on the same idea, recent FL studies alternatively reduce the trainable parameters by diminishing the model size. Specifically, these studies leverage custom-size client models to address the straggler issue.

Instead of distributing a global model uniformly to the participant clients, which is challenging for various devices to complete the local updates in the same amount of time, we can assume that the client is assigned with a model with a specific size that matches its computational capability, ensuring its local updates are completed timely. Specifically, a client with less powerful hardware is given a lightweight model requiring less effort to update. Therefore, the client can successfully complete local updates and subsequently upload its training results to the server. Given this premise, stragglers are transformed into non-stragglers in heterogeneous FL by the deployment of custom-size client models.

Approaching Custom-size Client Models with Knowledge Distillation

However, while addressing the straggler issue, the distribution of custom-size client models introduces another problem: creating knowledge barriers between the global model and client models during global updates. Recall Equation 2.4, given uniform client models, the global model is formulated by simply fusing parameters of updated client models. This is not feasible when client models have various sizes, as we cannot match the parameters of client models. Hence, alternative global updates methods are needed to transfer the learned knowledge from custom-size client models to the global model.

Knowledge distillation (KD) (Hinton et al., 2015) is widely adopted to transfer the locally learned knowledge to the global model. KD is also known as knowledge transfer (KT). It is a popular machine learning approach for model compression (Buciluă et al., 2006; Polino et al., 2018). With KD, a heavily-parameterised model with superior performance can transfer its knowledge to a lightweight model that aims to achieve similar performance to the heavily-parameterised model. KD is model structure agnostic, thereby perfectly fitting to the knowledge transfer among heterogeneous models.

Early applications of KD to FL were motivated by improving the communication efficiency rather than supporting custom-size client models to address the straggler issue. Compared to communicating full models between clients and server, KD-based FLs transmit logits (Jeong et al., 2018). Logits are model outputs of much smaller size than the model itself, significantly reducing communication costs in the network. Later works (Chang et al., 2019; Sun and Lyu, 2020) demonstrated that KD-based FL is robust to adversarial attacks and noise pollution, further advancing its application in FL. More recently, with growing research interest in addressing the straggler issue in FL, a number of KD-based FLs are proposed to facilitate custom-size client models in tackling this challenge. These methods focus on improving knowledge aggregation in the presence of data heterogeneity and relaxing the need for a public dataset to assist KD.

Advancing Knowledge Aggregation and Relaxing the Public Data

FedMD (Li and Wang, 2019) and FedDF (Lin et al., 2020) are among the most popular KD-based FLs tailored for custom-size client models. In FedMD, a public dataset is assumed to be accessible by all clients. After local updates, client models upload their logits generated on a public dataset to the server. Uploaded logits are aggregated by the server to form a consensus as the global knowledge which is used to supervise client models to learn on the public dataset. Client models approximate their outputs to the consensus, thereby learning from the global knowledge. FedMD demonstrates the efficacy of KD for transferring knowledge among custom-size client models. FedDF follows a similar approach to FedMD in terms of forming the consensus and transferring the global knowledge. However, FedDF proposes to use KD to improve the performance of the fused global model in a homogeneous client model setup. Nevertheless, its solution can adapt to custom-size client models.

Later, DS-FL (Itahara et al., 2021) and FedAUX (Sattler et al., 2021) advances FedMD by assuming the public data is not labelled, as FedMD relies on a labelled public dataset to pretrain client models in advance of FL. Particularly, FedAUX performs unsupervised representation learning on the unlabelled public dataset to learn a feature extractor before starting FL. Additionally, FedAUX advances the formulation of the consensus with certainty weights, improving the reliability of the consensus. Similarly, Fed-ET (Cho et al., 2022) introduces a weighted consensus, selecting credible logits to transfer knowledge to the global model. Also using an unlabeled public dataset, FedKEM (Nguyen et al., 2023) proposes performing mutual knowledge distillation between the global model and client models.

On another track, FedBE (Chen and Chao, 2020) proposes using the averaged prediction from ensemble models to enhance the robustness of the global performance. A number of models are sampled from an estimated conditional distribution of the global model to form the ensemble. KD is used later to distill ensemble models into a single model for faster inference. Finally, FedGEN (Zhu et al., 2021) and DENSE (Zhang et al., 2022a) completely remove the need of public data for transferring the global knowledge. Particularly, FedGEN learns a lightweight data generator using predictors learned on the client side. The data generator can yield augmented global representations for class labels and is offloaded to the client side to transfer the global knowledge by regularising the predictor in local updates. DENSE advances FedGEN by learning a generator on the server to create synthesised raw data rather than feature representations, without breaking the privacy-preserving rule of FL. The synthesised data is then used to assist knowledge transfer from client models to the global model.

Utilising Latent Feature Representations

FedGKT (He et al., 2020a) is proposed to split the learning of the global model between the server and clients. FedGKT addresses the straggler issue by having clients train small-size feature extractors, while the large part of the global model is learned on the server. Particularly, it trains local feature extractors on clients and uses them to extract feature representations for local data. Clients upload these feature representations to train the majority of the global model on the server. KD is employed to assist the split training on both server and client sides by aligning the outputs between the paired client model and global model.

FedAD (Gong et al., 2021) uses latent feature representations to enhance the knowledge transfer from custom-size client models to the global model. Notably, during the global model training on the public dataset, FedAD not only aligns the global model to the consensus in the output space but also approximates its latent output to the top-down class-specific attention maps aggregated from client models, leading to improved global model performance.

2.2.4 Active Data Selection

Rather than reducing the number of parameters to learn, another popular strategy to improve the training efficiency is to reduce the amount of training data. Active data selection is an established approach to achieve training data reduction. It has been well studied in centralised machine learning, leading to a popular research branch in machine learning known as active learning (Settles, 2009; Ren et al., 2021).

Active learning is driven by the fundamental premise that machine learning algorithms can perform better with actively selected training samples. For supervised machine learning, active learning is leveraged to reduce labeling efforts (Fu et al., 2013; Zhan et al., 2021). Conversely, works such as Mindermann et al. (2022); Yang et al. (2023); Li et al. (2022) demonstrate that active data selection can improve the training efficiency for deep learning models.

Typical FL works that adopt active learning, such as LoGo (Kim et al., 2023b) and F-AL (Ahn et al., 2024), are mainly driven by the objective of reducing labeling burdens on clients (Jeong et al., 2020), the same motivation behind the development of active learning in centralised machine learning. These works have not conducted studies on reducing the workloads on stragglers through active data selection. On the other hand, there are only a limited number of works that leverage active data selection for improving FL performance in heterogeneous environments. Inspired by the active learning method proposed in centralised learning (Katharopoulos and Fleuret, 2018), Li et al. (2021a) use the gradient upper bound norm of the global loss to calculate importance scores for selecting local instances. FLRD (Nagalapatti et al., 2022) learns a relevant data selector on the client side using reinforcement learning. The learned selector can select local samples most beneficial for global model performance. To train the data selector locally, the client is assumed to hold a public dataset for computing the reward for reinforcement learning. Sun and Lyu (2020) proposes a once-for-all data selection phase in advance of FL. It filters out relevant training instances on clients with a loss threshold calculated on the server. The threshold loss is determined using instance losses uploaded by all participating clients.

Uncertainty-based data selection (Beluch et al., 2018; Liu et al., 2021c; Yuan et al., 2021) is one of the most widely used approaches for identifying useful learning instances. By calculating the uncertainty of the instance on a proxy for the learning task, the most difficult samples are unearthed to train the model. These difficult samples are more useful for learning as more knowledge is conveyed to the model by using them. In centralised learning, entropy is among the most popular metrics for measuring the uncertainty for data points (Joshi et al., 2009; Luo et al., 2013;

Aggarwal et al., 2014; Li et al., 2019a). In the realm of FL, FedEntropy (Orlandi et al., 2023) leverages entropy to select the most useful client models for global aggregation. Each participating client uploads an averaged entropy on its local data to the server. Then the server prunes clients that reduce the overall entropy, resulting in a subset of clients that achieve maximal global entropy. FedEntropy demonstrates that fusing models uploaded by those selected clients leads to better global model performance. FedAvg-BE (Orlandi et al., 2023) utilises entropy to select batched local data for updating the client model. The batched local data is hard for the client model to learn, consisting of selected training samples having maximal entropy calculated by the client model.

2.3 Related Work on Model Shift Mitigation

2.3.1 Model Shift Aggravates Straggler Issue

Besides addressing the straggler issue, solving the model shift problem needs is essential for improving the participation of resource-constrained devices. Model shift can significantly slow down the convergence of FL, resulting in higher energy costs for clients throughout FL. The higher energy costs can discourage the participation of clients, particularly those with resource constraints. Both theoretical and empirical studies (Karimireddy et al., 2020; Wang et al., 2020a; Li et al., 2021b) have demonstrated that model shift is induced by the data heterogeneity across clients. Therefore, to understand the model shift problem, we first need to introduce data heterogeneity in FL.

Modelling Data Heterogeneity with Non-IID Client Data

Data heterogeneity poses a major challenge to federated learning. Since Federated Averaging (FedAvg) (McMahan et al., 2017), many subsequent works (Zhao et al., 2018; Hsu et al., 2019; Li et al., 2019c; Abdelmoniem et al., 2023) have revealed that data heterogeneity can negatively impact generalisation, convergence and fairness in FL. Data heterogeneity refers to the non-IID client data, reflecting the diverse distributions of user-generated data.

Mathematically, a client i is sampled from distribution $\mathcal{Q}$, denoted as $i \sim \mathcal{Q}$, where $\mathcal{Q}$ is the distribution over available clients, to participate in federated learning. Considering a supervised learning task, the local data on client i is represented by the distribution of $P_i(x, y)$, where x is the data point and y is its assigned label. Kairouz et al. (2021) and Ye et al. (2023) define the non-IID client data in federated learning with $P_i \neq P_j$ for client i and j. They distinguish non-IID client data scenarios with respect to the marginal distributions and conditional distributions, as we can decompose $P_i(x, y)$ into $P_i(y \mid x)P_i(x)$ or $P_i(x \mid y)P_i(y)$.

Feature skew. Feature skew refers to the feature differences across clients. It includes two cases: 1) feature distribution skew, denoted by the marginal feature distribution differences $P_i(x) \neq P_j(x)$; 2) feature condition skew, denoted by the conditional feature distribution differences $P_i(x \mid y) \neq P_j(x \mid y)$. An exemplary case in feature distribution skew is the handwriting styles from different people can be various even though they may share the same conditional label distribution $P_i(y \mid x) \neq P_j(y \mid x)$. With respect to feature condition skew, it describes that the features from different clients can vary significantly under the same label. For instance, horses from different countries can have different sizes. The horses from the Mongolia region are typically shorter than the horses from the European countries.

Label skew. Label skew refers to the differences of label distributions among clients, encompassing two categories: 1) label distribution skew, denoted by the marginal distribution differences $P_i(y) \neq P_j(y)$; 2) same feature and different labels, denoted by the conditional distribution differences $P_i(y \mid x) \neq P_j(y \mid x)$. The label distribution skew is commonly observed in cross-device FL, reflecting varying users preferences for categorised content. For instance, cat lovers may predominantly watch cat videos on their devices, while dog videos are favoured by dog enthusiasts. On the other hand, the same feature and different labels indicates that users may class the same item differently due to their background differences, such as cultural influences.

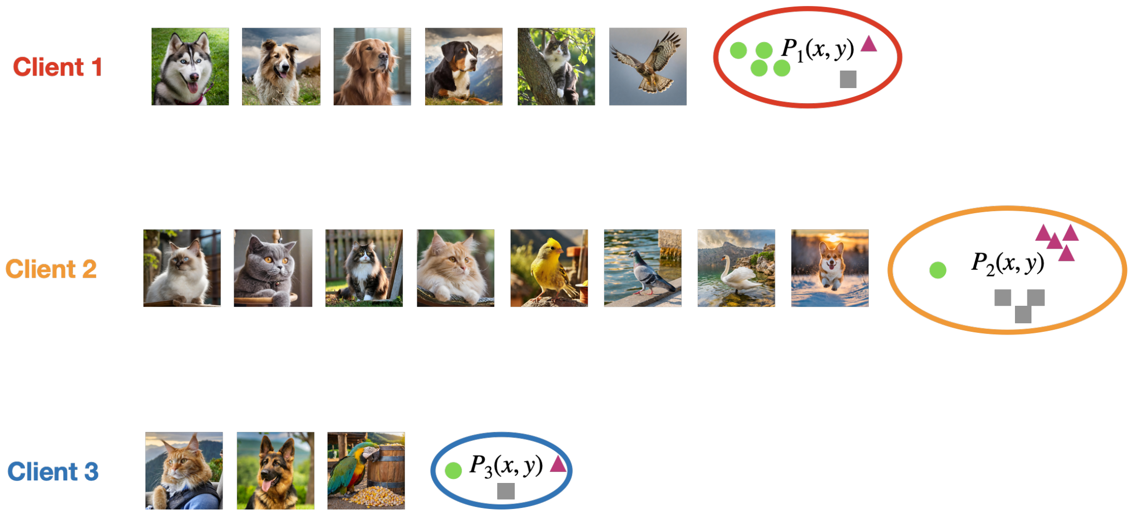


Figure 2.5: Label skew and quality skew across three clients. P_1, P_2, P_3 represents different client data distributions.

Quality skew. Quality skew suggests substantial variations in the size of client data. Active clients tend to generate a large amount of on-device data, whereas samples from idle users are relatively limited. The disparity in data volume can compound with label skew, as illustrated in Figure 2.5.

Model shift problem. Model shift occurs when locally updated models drift away from the globally optimal model (Reddi et al., 2020). This problem is closely related to the concept of dataset shift in machine learning (Quinonero-Candela et al., 2008; Moreno-Torres et al., 2012; Kairouz et al., 2021). When the model is updated through ERM on client data, it is optimised to fit client data distribution. With non-IID client data, client updated models tend to shift drastically from each other. As a result, the global model formulated during global updates deviates from the global optimum. Figure 2.6 illustrates how model shift happens. Figure 2.7 shows the negative impact of data heterogeneity on the global model performance by simulating different levels of data heterogeneity. The degraded FL performance is mainly induced by the local optimisation shifting away from the global objective given non-IID client data. In a practical FL setup, $P_i(x, y) \neq P_g(x, y)$, where $P_g(x, y)$ is the global distribution of the learning task (or the target distribution), the aggregation of deviated client models leads to slow convergence and deteriorated

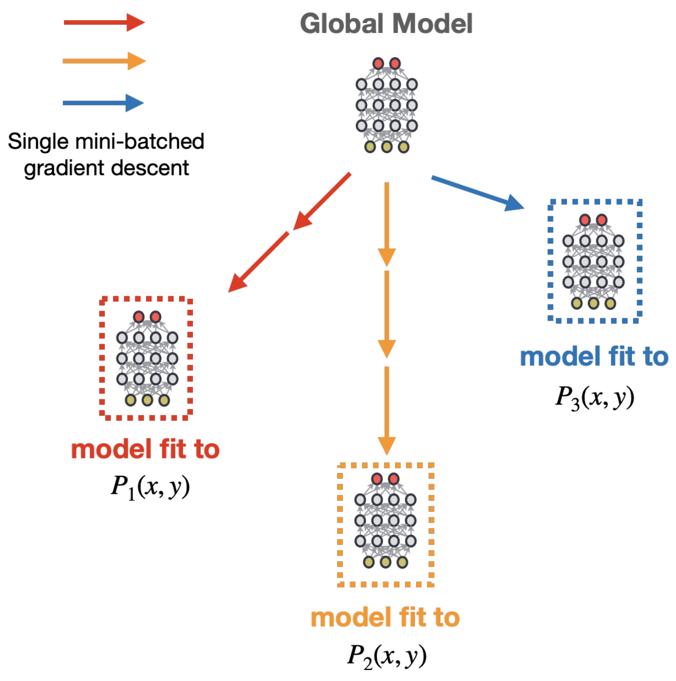


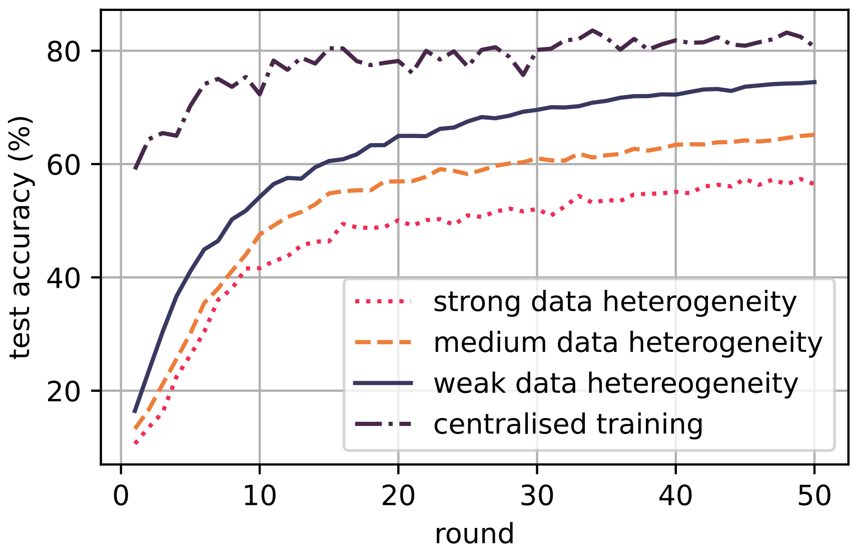
Figure 2.6: The model shift problem. Updated models fit to local data distributions.

generalisation.

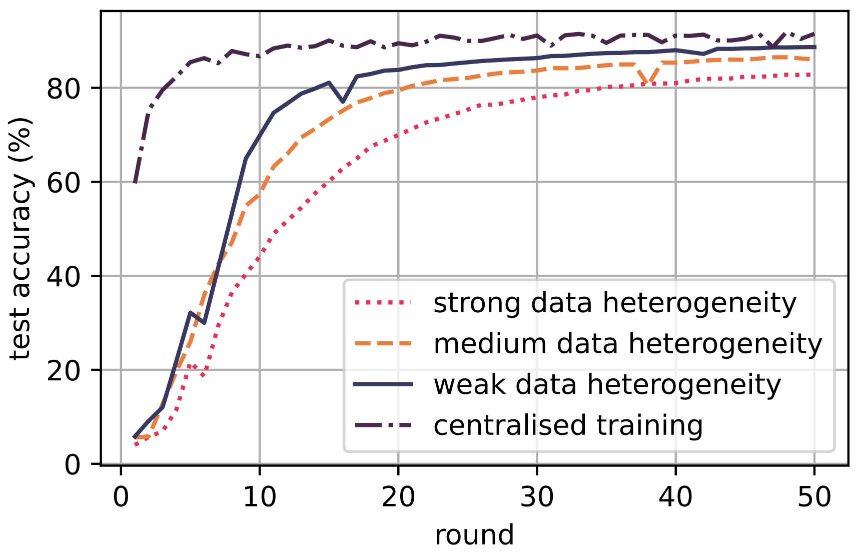
Extensive studies have investigated the model shift problem and its detrimental effects on FL performance (Zhao et al., 2018; Hsu et al., 2019; Li et al., 2019c,b; Abdelmoniem et al., 2023). In the following section, existing works that address the model shift problem will be reviewed. These approaches are categorised into three groups: regularisation on local updates, federated optimisation, and server-side centralised learning.

2.3.2 Regularising Local Updates

A common approach adopted to mitigate model shift is to regularise local updates. This approach prevents the updated model from overfitting to the non-IID client data by penalising the training loss function with a term indicating the global learning objective on the task. FedProx (Li et al., 2020c) is one of the most recognised regularisation solutions to tackle the model shift problem. FedProx regularises the local updates with a proximal term, which is essentially the global model formulated by the server every communication round. As such, the updated model on



(a) CIFAR-10



(b) GSC

Figure 2.7: Model shift induced by data heterogeneity penalises global model performance on CIFAR-10 and GSC datasets. It is apparent that both the generalisation and convergence are negatively affected by heterogeneous client data. As data heterogeneity becomes stronger, the detrimental impact on FL performance becomes more pronounced. There is a noticeable performance gap between FL and centralised learning.

the client cannot deviate much from the global model given non-IID client data. However, FedProx suffers from slow convergence due to the constant restriction on the local updates. FedDyn (Acar et al., 2021) criticises that FedProx fails to match the minima of the proximal term to the global stationary point, resulting in slow convergence. In addition to the original proximal term for aligning the model parameters, FedDyn introduces a linear penalty term, which accelerates FL convergence by restricting the deviation of local gradient descent.

Similarly, SCAFFOLD (Karimireddy et al., 2020) proposes that every client additionally calculates a gradient-based term to correct its local update direction, forcing it closer to the ideal IID local data scenario. Nevertheless, compared to FedDyn, SCAFFOLD is disadvantaged by the additional communication overhead as it transmits both model parameters and gradients in the network. The aforementioned works mainly consider the data heterogeneity as the label skew and quality skew. To tackle the model shift induced by the feature skew, FCCL (Huang et al., 2022), utilises the unsupervised representation learning on the server to maximise same class invariance in different domains and the variance among different classes, learning a class representation that is not only universal across domains but also distinguishable to other class representations to combat the domain shift. FCCL also introduces a dual knowledge distillation to clients for avoiding the knowledge forgetting problem. However, the dual knowledge distillation adds an extra training cost for updating the model.

The model shift problem in FL shares similarities with the catastrophic forgetting problem in lifelong learning. Comparing the local updates on one client to a new task in lifelong learning, FL and lifelong learning both seek to avoid the model overfitting to a single task. For instance, FedCurl (Shoham et al., 2019) adopts the lifelong learning solution, Elastic Weight Consolidation (EWC) (Kirkpatrick et al., 2017), to penalise the local loss function, aligning the client models to the global optimum. Instead of penalising the entire model, as mentioned above, another line of regularisation focuses on limiting the drift in the feature extractor during local updates. This approach is based on the premise that the feature extractor learned on the entire dataset is more generic than one learned on non-IID client data. MOON (Li et al., 2021b) introduces the contrastive loss to regularise the feature extractor. The contrastive loss is formulated by differences between the features extracted from the global model and the client model on the client data. It pushes the client model to learn representations close to the global representations but away from the representations learned in previous rounds. Penalised by the contrastive loss, the feature extractors across the clients are aligned during local updates.

FedPAC (Xu et al., 2023) also aligns the feature extractor with regularisation during local updates. However, this paper proposes utilising prototype features constructed on the server with features extracted from client models. Client subsequently download the prototype features and uses them to guide its local updates, injecting global views to avoid local overfitting. On the other hand, FedBABU (Oh et al., 2021) completely fixes the classifier across the clients to make the updated feature extractor resistant to model shift. The frozen classifier acts as a regulariser while updating the feature extractor. Similarly, FedETF (Li et al., 2023) and FedNH (Dai et al., 2023) introduces the prototypical classifier specifically designed for locally updates to avoid the classifier bias introduced by data heterogeneity.

Finally, FedAlign (Mendieta et al., 2022) examines standard regularisation methods used in local updates, particularly Mixup (Zhang et al., 2017), Stochastic Depth (Huang et al., 2016), GradAug (Yang et al., 2020), comparing them to methods adopting the proximal term like FedProx. FedAlign finds that the standard regularisation is on par with or even outperforms FedProx and MOON. They adopt GradAug, modified to consume fewer computational resources, to regularise local updates and suggest the academic community to rethink the regularisation approach by focusing on improving learning generality in local updates.

2.3.3 Advanced Federated Optimisation

The global model optimisation in vanilla FedAvg has no particular measure to mitigate model shift. As such, a branch of work addresses the model shift problem by advancing federated optimisation on the server. Some of these works further combine the advanced federated optimisation with regularisation on clients.

FedAvgM (Hsu et al., 2019) improves FedAvg to mitigate model shift by introducing a momentum term to the global updates. The adoption of the momentum is inspired by its mitigating effect on the gradient oscillations particularly induced by data heterogeneity. FedNova (Wang et al., 2020a) revises the global model optimisation in FedAvg with the normalised local gradient descent and aggregation weights decided by the times of the local updates to particular address running uneven numbers of mini-batched gradient descent across clients. FedOpt (Reddi et al., 2020) studies the application of adaptive optimisers from centralised machine learning to the federated learning setup. It points out the rigidity problem of the server optimiser in FedAvg and proposes utilising adaptive optimisers, such as ADAGRAD, YOGI or ADAM, to improve the flexibility of global model optimisation. They demonstrate that the adaptive optimiser can considerably improve FL convergence through rigorous evaluations.

On a different track, FedDC (Gao et al., 2022) introduces a local variable to track the model drift, which is then uploaded to the server. This strategy allows the server to cancel out the model shift effect while updating the global model. Most recently, Chen et al. (2023) propose to use the elastic aggregation to replace the vanilla parameter averaging in global updates. The elastic aggregation introduces a model drift mitigation strategy that measures the model output sensitivity to individual parameters. During global updates, those less sensitive parameters are updated, whereas more sensitive parameters are kept unchanged, thereby mitigating model shift on the server.

Finally, FedBN (Li et al., 2021d) proposes to mitigate the feature skew by excluding the batch normalisation layers from the model aggregation. With this simple modification to FedAvg, FedBN achieves impressive performance improvement in FL under simulated feature skew scenarios.

2.3.4 Formulating Centralised Learning on the Server

Another branch of work aimed at mitigating model shift focuses on creating a centralised learning pathway within FL. This approach commonly involves extracting information about the global data distribution from clients without breaking the privacy-preserving rule, using this information to generate a synthesised dataset on the server. The global model is refined on the synthesised dataset in a centralised learning manner to improve its performance.

CCVR (Luo et al., 2021) demonstrates that the classifier in the model is particularly susceptible to the shift problem. To address this, CCVR extracts prototype representations of local images and uploads them to the server to synthesise a universal dataset. This dataset is specifically used to train a less biased classifier for the global model in a centralised learning approach.

Similarly, DynaFed (Pi et al., 2023) synthesises a dataset that can approximate the global data distribution by tracking the dynamics of the fused global model on the server. The synthesised dataset is then used for refining the global model, mitigating the model shift effect. FedFTG (Zhang et al., 2022b), on the other hand, learns a data generator on the server to output synthesised data to assist the knowledge distillation from client models to the global model, correcting the model shift following model fusion.

In summary, previous works on addressing the model shift problem commonly introduce additional workloads for clients. While achieving huge success in mitigating model shift, they often do not encourage the participation of resource-constrained devices.

2.4 Research Gaps in Approaches for Addressing the Straggler Issue

Based on the review of the state-of-the-art FL works, we have identified three key research gaps that have not been effectively addressed in previous studies aiming to solve the straggler issue: 1) unmitigated model shift in the approach of partial model training; 2) weak knowledge transfer with a single knowledge source in the approach of custom-size client models; 3) understudied efficacy of the active data selection approach for addressing the straggler issue.

Unmitigated model shift in partial model training. In the partial model training approach, the extraction of submodel proposed in previous works, such as FjORD, HeteroFL, FedRolex, etc., has successfully reduced the workload on resource-constrained devices. However, given data heterogeneity, these methods face difficulties in extracting optimal submodels for individual participating clients, leading to the problem of uneven client training (Alam et al., 2022). This uneven training problem has an interplay with the model shift problem, resulting in slow FL convergence and increased energy consumption for stragglers. Existing partial model training methods typically miss an effective solution to mitigate model shift. Conversely, previous methods focusing on model shift mitigation, such as FedProx, FedDyn, MOON, SCAFFOLD, FedCurl, FedFTG, FCCL, etc., commonly introduce additional computational overhead for clients. Hence, these methods are not suitable for addressing the straggler issue, as they solve the model shift problem by increasing computing burdens on clients. To this end, there is a need for the partial model training approach to reduce workloads on clients as well as mitigate model shift.

Weak knowledge transfer with a single knowledge source. Although many efforts (Itahara et al., 2021; Sattler et al., 2021; Cho et al., 2022; Chen and Chao, 2020) have been made to improve the knowledge transfer in FL with custom-size client models, previous works typically employ a single knowledge source–model outputs at the top layer–to transfer the knowledge between client models and the global model, including FedMD, FedDF, DS-FL, FedAUX, Fed-ET, etc. However, with drifted client models, top outputs of client models are often inconsistent and unreliable while they are aggregated, significantly weakening knowledge transfer and compromising the global model performance. The problem of weak knowledge transfer has become a major obstacle for the approach of custom-size client models to address the straggler issue. Therefore, it is necessary to employ more credible knowledge sources that can effectively enhance knowledge transfer between clients and the server regardless of the differences in model sizes. By adding new sources for knowledge transfer and demonstrating their efficacy, we can fill the research gap in existing literature where only a single knowledge source is utilised. The study can lead to improved global model performance in FL adopting custom-size client models, significantly strengthening its application in addressing the straggler issue.

Understudied active data selection for solving the straggler issue. There are two major problems in existing FL literature that employs active data selection. First, mainstream FL methods, such as LoGo and F-AL, have not fully explored the potential of active learning to address the straggler issue, despite its proven success in centralised learning for improving training efficiency. Their methods primarily focus on alleviating the labelling efforts for client data and lack a in-depth study on the learning efficiency of clients that adopt active learning. Second, methods like FLRD introduce substantial computational workloads to clients for strategically selecting beneficial training instances, further aggravating the straggler issue. To this end, it is recognised that the potential of using active data selection for addressing the straggler issue is understudied in current research. It is crucial to adapt and transfer the success of active data selection in approaching efficient machine learning to the framework of FL. By filling this research gap, we can enrich the toolbox for addressing the straggler issue by leveraging the strengths of active data selection.

In summary, closing these critical research gaps is vital for enhancing the effectiveness of approaches aiming to tackling the straggler issue in FL. By mitigating model shift in partial model training, strengthening knowledge transfer with diverse sources in custom-size client models, and advancing active data selection methods, these approaches can significantly boost the participation of resource-constrained devices, empowering a broader range of devices to contribute valuable knowledge to FL.

2.5 Summary

This chapter systematically reviews existing FL research in addressing the straggler issue. They are categorised into three approaches: 1) partial model training; 2) custom-size client models; 3) active data selection. Despite the extensive research in these areas, specific challenges remain in these approaches, diminishing their effectiveness in addressing the straggler issue in practice. They are: 1) Unmitigated model shift in partial model training. 2) Weak knowledge transfer with a single knowledge source in the approach of custom-size client models. 3) Understudied active data selection for solving the straggler issue. This review highlights the importance of addressing these research gaps in FL. In the next chapter, we will introduce the joint experimental setup used throughout this thesis.

Chapter 3

Joint Experimental Setup

In this Chapter, we introduce the joint experimental setup used throughout this thesis. The following chapters also include specific experimental setups, which are tailored for particular methods.

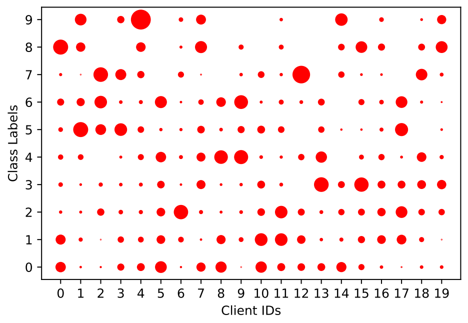
3.1 Datasets

We implement and evaluate our methods on both image and speech classification tasks. CIFAR-10 and CIFAR-100 (Krizhevsky and Hinton, 2009) are adopted for the image classification, which are very popular for evaluating FL algorithms. CIFAR-10 is an image dataset with 60,000 data points, which are evenly split among a total of 10 classes. The sizes of the training set and test set are 50,000 and 10,000 respectively. Each data point is an image of 32×32 pixels with RGB channels. CIFAR-100 is similar to CIFAR-10, except it is divided into 100 classes with each class containing 600 images. CIFAR-100 also includes coarse labels, which group 5 fine-grained labels into one coarse label. We use the fine-grained labels of CIFAR-100 in the experiments. Public CIFAR-10 and CIFAR-100 do not have exiting sets for the validation purpose. So for both of them, we uniformly draw 5000 random samples from the training set to construct the validation set for validating the hyperparameter setup for our methods.

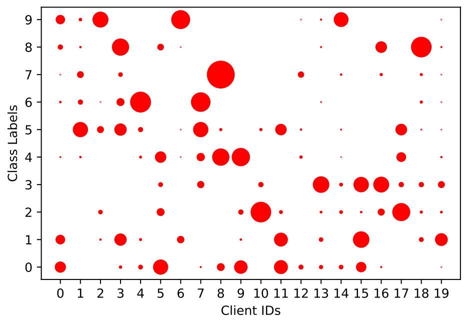
On the other hand, the application of FL to Automatic Speech Recognition (ASR) in practice is becoming increasingly important. Large-scale solutions for

ASR usually rely on audio data from multiple sources that are collected in realistic conditions. While in early-days ASR systems evolved by training on lab collected speech data, nowadays the strength of ASR systems comes from the realistic conditions under which data samples are collected – background noise, different locations and accents (Watanabe et al., 2020; Cornell et al., 2023). With data privacy becoming a greater concern in society, and with the proliferation of powerful mobile devices connected through low latency connections (e.g., 5G), distributed training for ASR is receiving increasingly more attention (Nassif et al., 2019; Cui et al., 2021; Granqvist et al., 2020; Hard et al., 2020; Leroy et al., 2019). To this end, we employ Google Speech Commands (GSC) Warden (2018) to evaluate our FL algorithms. GSC consists of 105,829 recorded utterances of 35 different words. Each recording holds an isolated word utterance lasting for one second. The training, validation and test sets of GSC are split disjointly into 84,843, 9,981 and 11,005 samples respectively. we follow the practice proposed by BalanceFL (Shuai et al., 2022) to transform the recordings from a sound wave to a 32×32 frequency map by applying Mel Spectrogram. Particularly, the sample rate of the sound waves is 16kHz. We set the number of Mel bank to 32. The FFT window size and hop window length are set to 1024 and 512 respectively. Therefore, we convert a one-second sound wave into a 32×32 single-channel image.

Heterogeneous data split. Following many prior works (Hsu et al., 2019; Lin et al., 2020; He et al., 2020b), the Dirichlet distribution, denoted by $Diri(\alpha)$, is employed to partition the non-IID local data in our experiments. Figure 3.1 visualises the non-IID local data split on 20 clients with seed 0 on CIFAR-10. Each bubble in the graph is associated with a class on a client. Its size indicates the number of data points, with a larger size suggesting a greater number of data points. The distribution created by $\alpha = 0.1$ is evidenced with a higher level of data heterogeneity than the one generated by $\alpha = 0.5$ as the bubbles have a wider range of sizes.



(a) $Diri(0.5)$.



(b) $Diri(0.1)$.

Figure 3.1: Visualisation of non-IID local data across 20 clients using CIFAR-10. Client IDs are presented on the Ox axis, CIFAR-10 classes on the Oy axis, with bubble size indicating sample counts. Higher values of $\alpha = 0.5$ suggest client data approaching an IID scenario, whereas lower $\alpha = 0.1$ demonstrates a strong non-IID case, characterised by varying bubble sizes.

3.2 Baselines and Models

Baselines. In addition to the standard FedAvg, this thesis compares its methods against popular FL methods for addressing the straggler issue and the model shift problem: FedProx, MOON, FedMD. FedAvg is one of the most widely used baselines for FL studies. As introduced in the Section 2.3, FedProx adopts a proximal term in local loss to restrict the entire model shift. Whereas, MOON limits the shift of the feature extractor in the model by adding the contrastive loss that maximises the similarity of feature representations created by the client model and the global model. FedMD is a popular KD-based baseline allowing custom-size client models in FL. It is employed to compare with our custom-size client model methods. FedProx and MOON are tuned across key hyperparameters, including the proximal term coefficient μ_{prox} and the contrastive loss coefficient μ_{con}, within the range {0.01, 0.1, 1, 5} for optimal performance.

Models. Wide ResNet (Zagoruyko and Komodakis, 2016b) is employed for both the global model and client models. In addition, MobileNetV3 (Howard et al., 2019), EfficientNet (Tan and Le, 2019), LSTM, are involved in ablation studies. Specific configurations of models are provided in the experiment sections of subsequent chapters.

Our algorithmic implementation is based on the FedML (He et al., 2020b) framework. All the experiments are conducted on a single NVIDIA GeForce RTX 2070 SUPER GPU on a desktop machine and NVIDIA Tesla V100s provided by the High Performance Clusters at the University of Sheffield. Regarding the number of FL communication rounds, different configurations are applied to various methods and datasets in the technical chapters. We determine the number of rounds by monitoring the validation performance of the global model. Notably, we focus on the improved visibility of the learning curve from changing the number of communication rounds and fixing the round number when the global model is already saturated, with just incremental gains. We apply this criterion across all methods unless otherwise stated. Detailed hyperparameter settings, such as learning rate, batch size, and number of local update epochs, etc., are also specified in the experiment sections of forthcoming chapters.

3.3 Evaluation Metrics

We choose accuracy as our primary evaluation metric. Accuracy is one of the most commonly used metrics for evaluating classification performance, as well as the main metrics used by the majority of FL works, such as FedAvg (McMahan et al., 2017), FedProx (Li et al., 2020c), MOON (Li et al., 2021b), and etc. By using the same metric to existing FL works, our comparison with the baselines will be straightforward.

Formally, accuracy is calculated by dividing the number of correction predictions made by the model with the total number of predictions as follows.

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} \tag{3.1}$$

Throughout this thesis, we use Top-1 accuracy unless stated otherwise. Top-1 accuracy counts the number of correction predictions based on the highest-probability output of the model.

3.4 Summary

This chapter describes the joint experimental setup used in this thesis to avoid repeated introductions. Our methods are evaluated on popular image classification and speech recognition tasks with data heterogeneity simulation. Accuracy is the evaluation metric employed for reporting the performance of methods. For tailored experimental setup, the details will be specified accordingly in the technical chapters. In the next chapter, we will introduce the method, few-shot fine-tuning, proposed in our first sub-research path.

Chapter 4

Mitigating Model Shift in Partial Model Training

4.1 Introduction

4.1.1 Research Questions to Address

Our literature review, Section 2.2, exposes the limitations of previous works in solving the model shift problem when adopting partial model training to reduce workloads for stragglers. This problem slows down FL convergence and limits the effectiveness of encouraging participation from resource-constrained devices. Our first sub-research path focuses on filling this research gap. Particularly, the research question to address in this sub-research path is, “How can partial model training reduce workloads for stragglers and mitigate model shift simultaneously?”

This research question can be further broken down into two more specific questions: 1) “How efficient is partial model training in FL?”; 2) “How can model shift be tackled using partial model training?”. By answering the first specific question, we will demonstrate that the proposed partial training method is computationally efficient, thus encouraging the participation of stragglers. Addressing the second question will advance the partial model training approach to not only resolve the straggler issue but also improve FL convergence by mitigating model shift, further reducing overall energy consumption for stragglers.

4.1.2 Proposed Methods and Rationale

Few-shot fine-tuning is the method proposed to approach the aforementioned research questions and bridge the research gap within partial model training. It is a partial model training method that is inspired by few-shot learning studies (Chen et al., 2019b; Tian et al., 2020).

We choose few-shot fine-tuning to tackle both the model shift problem and straggler issue because it has two main advantages over relevant state-of-the-art FL works. First, few-shot fine-tuning can effectively alleviate the model shift by freezing the majority of model parameters, avoiding additional computational overhead on clients–a critical improvement over methods like FedDyn, SCAFFOLD, MOON, FedDC, etc. Second, few-shot fine-tuning shares the entire feature extractor of the model across clients, rather than training submodels on non-IID client data proposed by previous partial model training methods, avoiding the uneven training problem that exacerbates the model shift problem.

CCVR is a close work to the proposed few-shot fine-tuning. However, CCVR performs the fine-tuning in a centralised manner requiring clients to upload additional feature maps to the server. Conversely, FedBABU, FedETF, FedNH all emphasise the important role played by the upper part of the client model, particularly the classifier, in mitigating model shift. Therefore they treat the classifier differently from the rest of the model, which is similar to few-shot fine-tuning. However, none of these works explore fine-tuning the upper part of the model for addressing the straggler issue.

To this end, the novelties of few-shot fine-tuning are twofold: 1) this work is the first to adopt few-shot learning methods for improving the participation of stragglers in FL, to our best knowledge; 2) Few-shot fine-tuning will be deployed on stragglers, enabling stragglers to contribute less shifted models with minimal computational effort. Few-shot fine-tuning uniquely unifies the mitigation of model shift with workload reduction in the approach of partial model training.

We hypothesises that few-shot fine-tuning can solve both the straggler issue and model shift problem, leading to advanced partial model training algorithms. By rigorously evaluating this method, this work will answer the research question and contribute substantial knowledge to the partial model training approach. Particularly, we designs and evaluates the proposed few-shot fine-tuning algorithm **Fed**erated Learning with the **F**ew-**S**hot Learned **C**lassifier, namely FedFSC. FedFSC will demonstrate that clients can complete few-shot fine-tuning significantly faster compared to standard local updates. Further, its efficacy in mitigating model shift will be studied. FedFSC aims to improve the convergence and generalisation of the global model significantly by allowing stragglers to contribute a less biased classifier for the global model. Therefore, the research questions outlined will be answered as follows: "Few-shot fine-tuning is computationally efficient for stragglers to perform, thereby addressing the straggler issue. The model shift is effectively mitigated by few-shot fine-tuning, resulting in a less biased classifier that can improve both the convergence and generalisation of FL."

4.1.3 Introduction to FedFSC

This section outlines our design and evaluation of FedFSC. Few-shot fine-tuning requires a generic feature extractor for parameter sharing and fine-tuning a partial model, crucial for alleviating the computational burden on stragglers. To obtain this generic feature extractor, FedFSC trains it within FL loops on non-stragglers, which have sufficient computational resources to train the entire model. Assisted by the generic feature extractor, stragglers then fine-tune only the classifier with a few local samples during local updates, enabling their participation in FL despite limited computational resources. In the context of FL, this local update method is denoted as the *few-shot updates*, contrasting with the standard local updates where the full model is updated using all local data.

FedFSC is inspired by prior studies in few-shot learning by Chen et al. (2019b) and Tian et al. (2020). In few-shot learning, to prevent overfitting deep learning models to a few training samples, MatchingNet (Vinyals et al., 2016), ProtoNet (Snell et al., 2017), RelationNet (Sung et al., 2018) redesign the metrics for generalisation by introducing similarity functions to compare the embeddings of few-shot testing samples and few-shot training samples. MAML (Finn et al., 2017) and Retile (Nichol and Schulman, 2018), on the other hand, optimise a good initial model on multiple few-shot training tasks to facilitate fast adaptation to novel few-shot testing tasks. However, Chen et al. (2019b) and Tian et al. (2020) demonstrate that simply fine-tuning the classifier atop a fixed feature extractor pretrained on a set of base tasks achieves comparable few-shot learning performance to the aforementioned state-of-the-art approaches. FedFSC leverages this insight, allowing clients to fine-tune the classifier while keeping the feature extractor fixed. This method is well-suited for addressing the straggler issue in FL as it requires minimal computational effort.

Figure 4.1 depicts the workflow of FedFSC. FedFSC fully updates the model, comprising a feature extractor and a classifier, across clients, resulting in a fully updated model termed the *base model*. The base model aids the few-shot update by providing a generic feature extractor, allowing clients to subsequently fine-tune the classifier with just a few local samples. Finally, FedFSC constructs the global model with the feature extractor from the base model and the few-shot learned classifier. FedFSC assumes all participants to update both the base model and the classifier. To enable the contributions from stragglers, we further propose a variant of FedFSC, denoted as FedFSC+. FedFSC+ allows stragglers to only perform few-shot updates that require minimal training efforts, whereas the non-stragglers in FedFSC+ are tasked to update the base model.

FedFSC and FedFSC+ are rigorously evaluated on CIFAR-10, CIFAR-100 and GSC datasets, approaching our research questions: "How efficient is partial model training?" and "How can model shift be tackled using partial model training?". Regarding the former, the few-shot updates are observed to reduce the training time on clients by 90%, encouraging the participation of resource-constrained devices. For the latter, the few-shot learned classifier shows resilience to model shift, making less biased predictions compared to a fully updated model in visualised prediction heatmaps. By performing the few-shot updates on stragglers, they are encouraged to contribute to the global model with a less biased classifier, improving the global model performance significantly by up to 6% over state-of-the-art FL baselines under simulated data heterogeneity. On learning curves, FedFSC is evident to

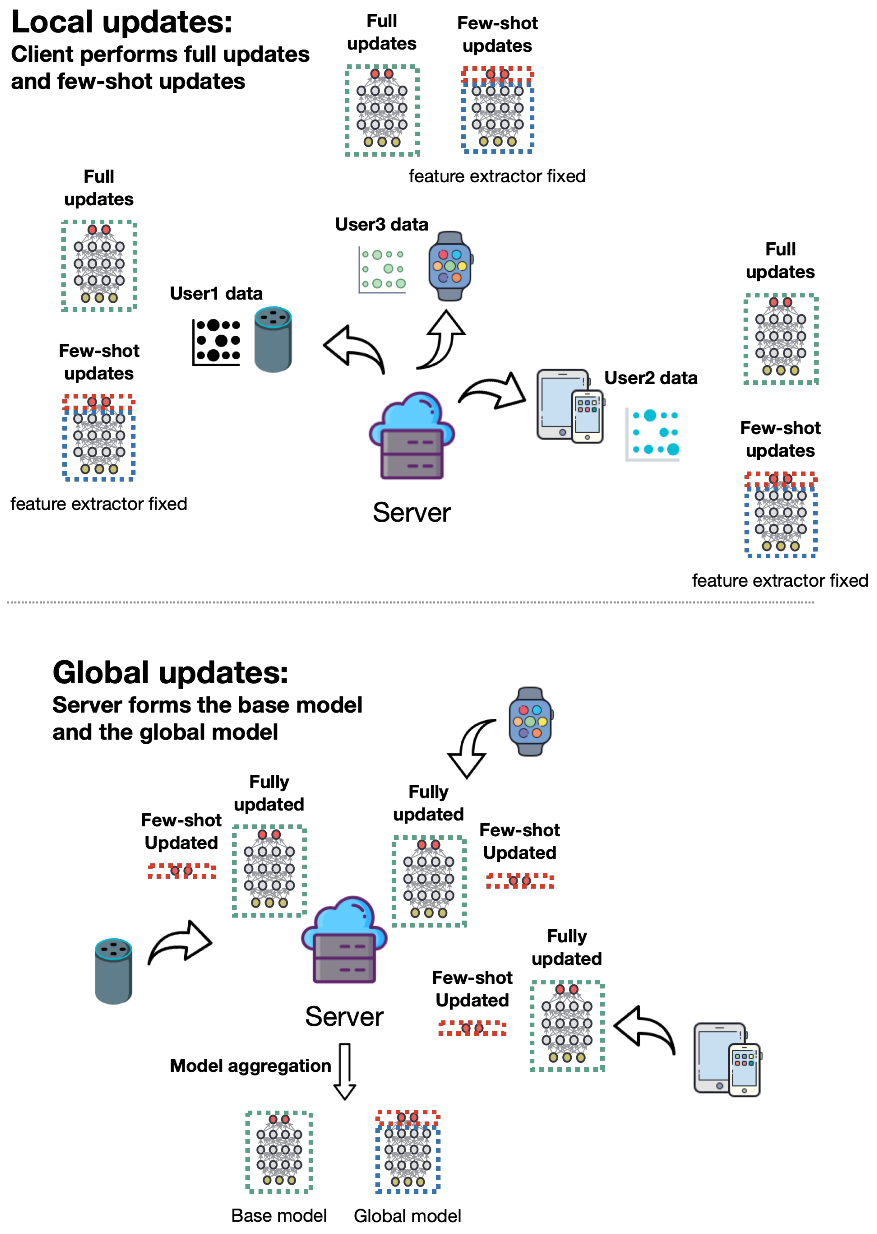


Figure 4.1: The workflow of FedFSC. This diagram illustrates a single communication round in FedFSC. In *local updates*, participants perform full model updates and few-shot updates separately. In *global updates*, the server collects the fully updated models and the few-shot updated classifiers to form the base model and the global model by fusing model parameters.

achieve faster convergence compared to the chosen baselines, reducing the overall energy cost for clients throughout FL. In the following sections, we will provide the details of the design and evaluation of FedFSC and FedFSC+.

4.2 Preliminary

Few-shot fine-tuning employs parameter sharing and fine-tuning. The parameter sharing assumption adopted by few-shot fine-tuning is the main contributor to address the straggler issue. The parameter sharing assumption originates from transfer learning. Transfer learning seeks to improve learning performance in the target domain by leveraging the knowledge learned from the source task (Zhuang et al., 2020). One key advantage of transfer learning is improved learning efficiency in target domains, based on the assumption of parameter sharing (Houlsby et al., 2019). Parameter sharing assumes a part of the learned model on the source task captures the general pattern that can be further shared with target domains. Consequently, we only need to fine-tune a part of the learned model on the target task, reducing the number of training parameters and therefore improving learning efficiency.

Conversely, few-shot learning (FSL) approaches the challenge of the increasing cost of acquiring sufficient data for training heavily-parameterised models. It is defined as training machine learning model(s) to generalise with a limited number of samples without overfitting (Wang et al., 2020c; Parnami and Lee, 2022). FSL leverages the parameter sharing assumption to achieve this goal. While training a heavily-parameterised model with a few training samples without overfitting is challenging, it becomes practical if a large part of the model is fixed. Notably, recent works (Chen et al., 2019b; Tian et al., 2020) find that simply fine-tuning the classifier atop a fixed feature extractor pretrained on a set of base tasks achieves state-of-the-art few-shot learning performance. Figure 4.2 details their approaches. Parameter sharing and few-shot learning are also used in training Large Language Models (LLM) with over billions of parameters (Brown et al., 2020), further demonstrating their efficacy in reducing the training cost.

Formally, few-shot fine-tuning splits the model into a lower part, the feature

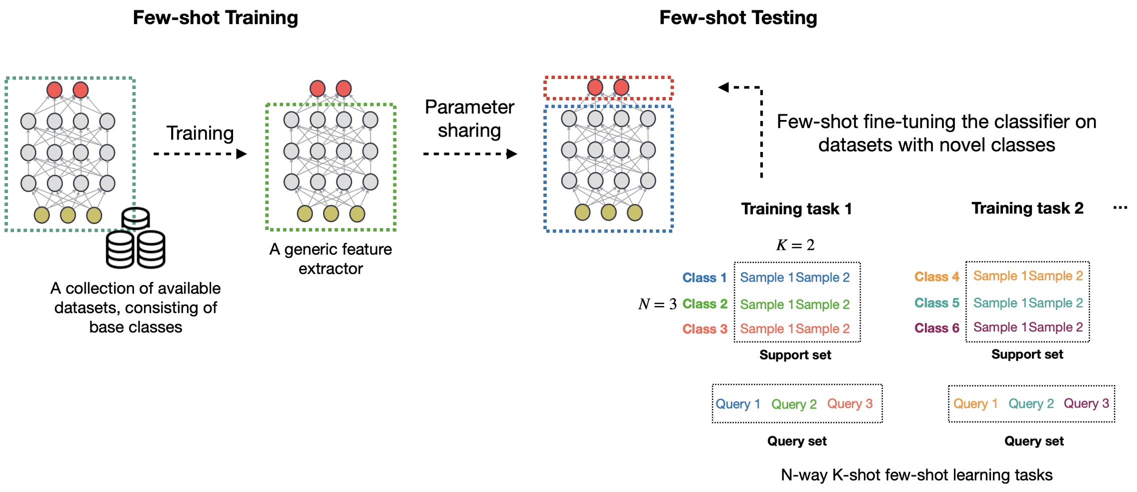

Few-shot Training
Few-shot Testing
Training
Parameter sharing
Few-shot fine-tuning the classifier on datasets with novel classes
A collection of available datasets, consisting of base classes
A generic feature extractor
Training task 1
Training task 2
...
K = 2
N = 3
Class 1 Sample 1 Sample 2
Class 2 Sample 1 Sample 2
Class 3 Sample 1 Sample 2
Class 4 Sample 1 Sample 2
Class 5 Sample 1 Sample 2
Class 6 Sample 1 Sample 2
Support set
Query 1 Query 2 Query 3
Query set
N-way K-shot few-shot learning tasks

Figure 4.2: The few-shot learning method proposed by Chen et al. (2019b); Tian et al. (2020). In the few-shot training phase, the model is pretrained on a collection of available datasets, consisting of the base classes. In the few-shot testing phase, the feature extractor in phase one is reused and fixed. The model further adapts itself to few-shot learning tasks comprising novel classes by fine-tuning its classifier. The illustration of N-way K-shot few-shot learning tasks is remade based on the article by Sultanov (2020).

extractor ϕ, and the upper part θ. In FL, parameters of the global model are expressed with $w_g = \{\phi_g, \theta_g\}$. Similarly, parameters of the model on client k are denoted by $w_k = \{\phi_k, \theta_k\}$.

Local updates on stragglers adopt the few-shot fine-tuning to achieve training efficiency. Formally, in the i-th local training epoch of client k, few-shot fine-tuning fixes ϕ_k and fine-tunes θ_k on a subset of local data, which contains a few local training instances randomly selected from the entire local data $\mathcal{D}_k$, defined by $d_{k,\,i} \sim \mathcal{D}_k$.

$$\theta_{k,\,i+1} \leftarrow \theta_{k,\,i} - \lambda_l \nabla_{\theta_{k,\,i}} \ell_k(\theta_{k,\,i}; \phi_k, d_{k,\,i}), \tag{4.1}$$

Notably, training instances in $d_{k,\,i}$ are not fixed during the local updates as they are resampled at the start of each local training epoch.

Compared with the standard local updates shown in Equation 2.3, the few-shot fine-tuning is more efficient as it only updates θ_k instead of all parameters w_k, and uses a very small subset of client data for training. Most importantly, the parameter sharing alleviates model shift by allowing clients to fix the parameters in the feature extractor during local updates, without adding any computational cost.

4.3 Few-shot Learned Classifier is Less Biased

This section introduces how few-shot fine-tuning alleviates model drift in FL. Our empirical study demonstrates the classifier updated by few-shot fine-tuning is less biased than a fully updated model with non-IID local data.

4.3.1 Few-Shot Learning with the Frozen Feature Extractor

We adopt the few-shot learning approach presented by Chen et al. (2019b); Tian et al. (2020). The model proposed for few-shot learning is split into a feature extractor ϕ and a classifier θ, where $\theta = \{\boldsymbol{W}, \boldsymbol{b}\}$ is the top fully connected layer, which sits right before the softmax layer to produce the model predictions. The remaining lower part of the model ϕ is seen as a feature extractor on the input data.

The chosen few-shot learning approach is performed in two stages. The first stage, defined as few-shot *training*, is the process of pretraining the model from scratch on a pool of base datasets. Base datasets are those available datasets in advance to deploy the model for few-shot learning on a target dataset. Trained on base datasets, the feature extractor learns to encode the data into features in the subspace. The second stage, few-shot *testing*, aims to generalise the pretrained model on novel datasets given only a few training samples are available. To achieve this, the feature extractor obtained in the few-shot training is frozen during the few-shot testing whereas a new classifier is retrained on novel datasets.

4.3.2 Training a Less Biased Classifier with Few-Shot Updates

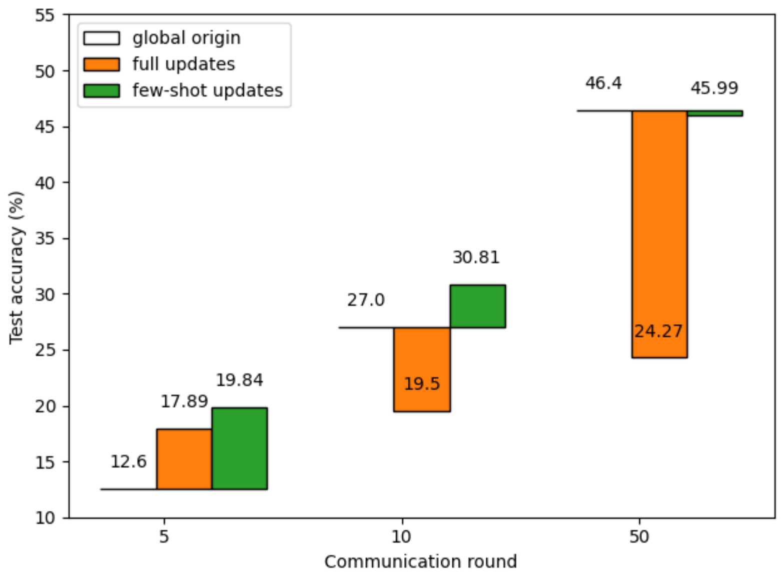
We transform the aforementioned few-shot learning solution from centralised training to the distributed training scenario. We alike the local updates on the client side to few-shot learning tasks and leverage the feature extractor trained inside the global model to perform the few-shot learning.

To this end, we propose the *few-shot update* for the client in FL. The few-shot update borrows the training strategy in the few-shot testing stage, training the distributed model with its feature extractor frozen on a few randomly selected local samples. To approach the research question, we broadcast the global model obtained at the end of a FL round to 100 clients with non-IID data to perform few-shot updates. Similar to a model pretrained in the few-shot training phase, the global model acts as a *base model* that provides few-shot updates with a feature extractor that is well-trained during FL. We also make an alteration to the few-shot testing setup introduced in their original work – instead of training the classifier from scratch, parameters of the classifier are directly copied from the trained global model.

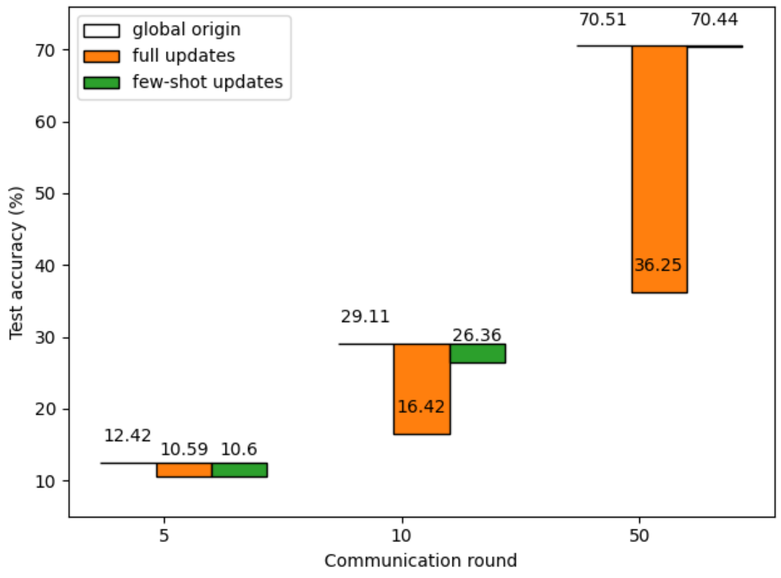
We employ the image classification dataset, CIFAR-10 and the speech recognition dataset, GSC, to evaluate the performance of few-shot updates on clients. Data heterogeneity is created across clients using the Dirichlet distribution, denoted as $Diri(\alpha)$. The hyperparameter α controls the degree of local data heterogeneity. The smaller α is, the stronger of local data heterogeneity. α is set to 0.1 in the evaluation, indicating a high level of data heterogeneity. For comparison, all 100 clients also perform full model updates on the global model separately. Sampling the global model trained at 5, 10, 50 FL round, Figure 4.3 compares the mean accuracy achieved by fully updated models and few-shot updated models on the test set of CIFAR-10 and GSC over all clients. It is obvious that the few-shot updated model that only trains its classifier performs significantly better than the fully updated model. Compared with the original global model, the few-shot updated model on average achieves either a marginal performance decrease or even a slight performance improvement, benefiting from the strategy of freezing feature extractor without incurring significant model shift on heterogeneous data. In contrast, fully updating the model yields noticeably lower performance, which is actually diverging in the later rounds of FL. Few-shot updates and full updates behave differently on CIFAR-10 and GSC at early rounds of FL. Notably, after local updates on clients, both full updates and few-shot updates improve the global model performance at communication round 5 on CIFAR-10. In contrast, few-shot updates and full updates slightly decrease the global model performance on GSC. This difference indicates local updates can impact the global model performance differently at early FL rounds on different datasets. Our insight to this observation is that FL learns faster on a less complex dataset such as GSC and thus more local updates introduced in later communication rounds have reduced benefits for improving the global model performance. Moreover, the negative impact from the model shift starts to signify at early FL rounds on GSC, overweighting the benefits of more local updates and leading to decreased global model performance. In Section 4.5.1, we will discuss the details that GSC is a simpler dataset than CIFAR-10, and further provide the insight that the complexity of dataset also affects the effectiveness of few-shot updates.

We hypothesise that fully updating the global model on non-IID local data leads to a more pronounced model shift, forgetting of its global knowledge. This in the end leads to a model with stronger biases in its predictions due to data heterogeneity. To validate this statement, Figure 4.4 displays the prediction heatmaps (mean logits per class) produced on the test set by the global model updated on one picked client. High data heterogeneity causes the heatmaps produced with the fully updated model to indicate highlighted strips on certain local classes, whose values significantly outweigh others. This is a clear signal of biased predictions towards local dominant classes, while forgetting the other classes that are under-represented on that client. In contrast, the few-shot updated model produces more symmetrical heatmaps with more predictions matching the correct class, which indicates that the learning is more robust to forgetting given non-IID local data. A similar pattern is also observed on other clients.

This study demonstrates that the proposed few-shot update is an effective solution to mitigate model shift induced by non-IID local data by showing the few-shot updated client model has better generalisation and less biased predictions compared to a fully updated client model.

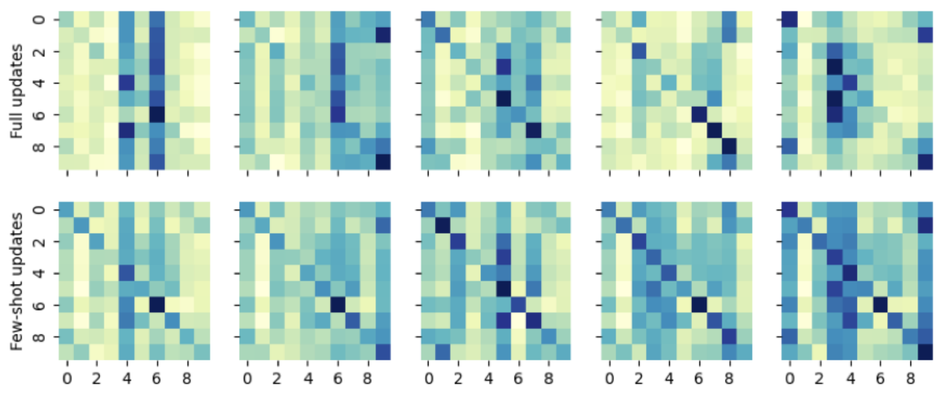


(a) CIFAR-10

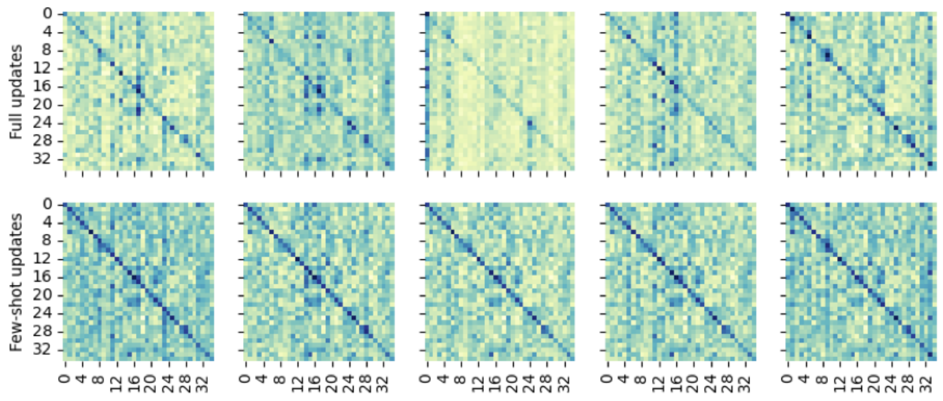


(b) GSC

Figure 4.3: Few-shot updates and full updates comparison: client test performance. Mean test accuracy of the fully updated models and the few-shot updated models on 100 clients. Global models sampled at 5, 10, 50 FL rounds are used. For reference, the test accuracy of the global model before updating is illustrated as the solid line in the left of each comparison group.



(a) CIFAR-10



(b) GSC

Figure 4.4: Few-shot updates vs. Full updates: the prediction heatmaps created by the updated global model. The numbers in the x-axis and y-axis represent the class number in CIFAR-10 and GSC. The global model updated by the few-shot approach has less biased predictions as its prediction heatmap has a stronger diagonal line.

4.4 Federated Learning with Few-shot Learned Classifier

The previous section discusses the advantageous client performance with the few-shot updates. This section formulates the FedFSC algorithm, integrating the few-shot updates into the FL workflow so that the few-shot updated classifier is utilised to boost the global model performance.

FedFSC constructs the global model with the fully updated feature extractors and the less biased few-shot updated classifiers. More concretely, each participant selected for FL performs both full model updates and few-shot updates on the broadcast model using their local data. As such, the participant not only sends a fully updated model to the server, but also uploads a few-shot updated classifier every communication round. The server collects the fully updated models to form the body of the global model, while also provides the few-shot updates with a well-trained feature extractor. In return, the few-shot updates utilise this feature extractor to specify the classifier while alleviating the model shift caused by data heterogeneity. Finally, the classifier of the global model is constructed by aggregating the few-shot updated classifiers contributed by the clients with a naive averaging defined by Equation 4.2.

$$\theta_{fs} = \sum_{k \in K} p_k \theta_{fs,\,k}, \;\; p_k = \frac{1}{|K|} \tag{4.2}$$

where $\theta_{fs,k}$ is the few-shot updated classifier on client k. K is the total number of the clients selected by the server to participate federated learning.

Algorithm 1 details the proposed FedFSC, using the notations introduced earlier in Chapter 2. Lines 3 describes client sampling, where a subset of clients is randomly selected from the client pool to participate in the communication round t. The *base model*, $\{\theta^t,\ \phi^t\}$, formed from fully updated models in the previous round is distributed by the server to all participants at round t. It should be noted that the base model is essentially the global model trained by a standard FL algorithm such as FedAvg. Line 4-14 present the local updates on the client side. Each selected client first fully updates the base model $\{\theta^t,\ \phi^t\}$ for E epochs using the entire $\mathcal{D}_k$ with SGD to $\{\theta_k^{t+1},\ \phi_k^{t+1}\}$. Then the client few-shot updates the classifier θ^t of the base model with the feature extractor ϕ^t fixed for E_{fs} epochs on a few randomly selected samples per class from $\mathcal{D}_k$, $\theta^t_{fs,\ k}$. We use t instead of $t+1$ to superscript $\theta^t_{fs,\ k}$ because this few-shot updated classifier is not distributed to the client side in round $t+1$. To initialise ϕ for the few-shot updates, the first communication round is performed with only full updates on clients. Line 15-18 depict the server averaging the fully updated models according to Equation 2.4 to form the new base model $\{\theta^{t+1},\ \phi^{t+1}\}$ that will be distributed in round t+1, as well as according to Equation 4.2 averaging the few-shot updated classifiers $\{\theta^t_{fs,;\ k}\}^K$ to create θ^t_{fs}. Finally, FedFSC returns w_g^t at the end of round t as the global model, being constructed of the averaged few-shot updated classifier θ^t_{fs} and the fixed feature extractor ϕ^t.

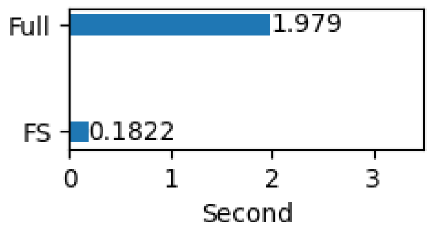
Notably, FedFSC is compatible other FL solutions than standard FedAvg. As suggested by Algorithm 1, different methods, such as regularisation approaches, can be applied to full updates as long as it can train the base model for the proposed few-shot updates. As such, we explore the performance of FedFSC on top of FedAvg and more advanced FL approaches in the evaluation.

4.4.1 FedFSC+: Utilising Stragglers

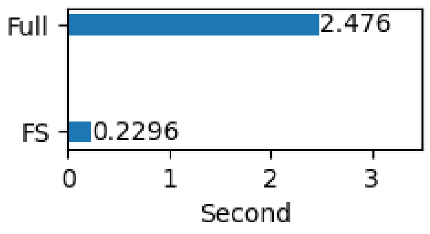
While the model shift problem is tackled, FedFSC is advanced to FedFSC+ to solve the straggler issue. Since the adopted few-shot update only trains the classifier part of the model with a few samples, it requires minimal training effort and can fit within the system constraints of those clients that would otherwise be stragglers. Figure 4.5 visualises the superior computational efficiency achieved by the few-shot updates. Running on a NVIDIA GeForce RTX 2070 SUPER GPU for 10 epochs, the averaged training time over 100 clients is reported, with $Diri(0.1)$ to split local data, achieved by 10-shot updates and full updates. On both CIFAR-10 and GSC datasets, 10-shot updates take less than 10% of time needed for a full model update. If stragglers can be identified during a communication round, we can avoid their

Algorithm 1 FedFSC: **Fed**erated learning with the **F**ew-**S**hot learned **C**lassifier.

```
1: Input: total $T$ rounds, $E$ full updates epochs, $E_{fs}$ few-shot updates epochs, $\theta$
   classifier, $\phi$ feature extractor, total $N$ clients
2: for $t = 1, \ldots, T$ do
3:     $K$ random clients are available for training and they download the base
       model $\{\theta^t, \phi^t\}$ from the server.
4:     Clients:
5:     for Client $k \in [1, K]$ do
6:         $\theta_k^{t+1}, \phi_k^{t+1} \leftarrow$ Full updates $(\theta^t, \phi^t; \mathcal{D}_k, E)$ according to Equation 2.3.
7:         if $t \geq 2$ then
8:             $\theta_{fs,\,k}^t \leftarrow$ Few-shot updates $(\theta^t; \phi^t, \mathcal{D}_k, E_{fs})$ according to Equa-
               tion 4.1.
9:         end if
10:        Client $k$ uploads $\theta_k^t$, $\phi_k^t$ and $\theta_{fs,k}^t$ to the server.
11:    end for
12:    Server:
13:    Collect fully updated models to form the full model set $B = \{\theta_k^{t+1}, \phi_k^{t+1}\}^K$
       and few-shot updated classifiers to define the classifier set $F = \{\theta_{fs,\,k}^t\}^K$.
14:    Form new base model $\{\theta^{t+1}, \phi^{t+1}\} = \sum_{i \in B} p_i \{\theta_i^{t+1}, \phi_i^{t+1}\}$, where $p_i =$
       $\frac{|\mathcal{D}_i|}{\sum_{i \in B} |\mathcal{D}_i|}$.
15:    Form few-shot updated classifier $\theta_{fs}^t = \sum_{j \in F} p_j \theta_{fs,\,j}^t$, where $p_j = \frac{1}{|F|}$
16:    Return Global model $w_g^t = \{\theta_{fs}^t, \phi^t\}$
17: end for
18: Return Global model $w_g^T = \{\theta_{fs}^T, \phi^T\}$
```



(a) CIFAR-10



(b) GSC

Figure 4.5: Training time of few-shot updates. The average training time (s) needed for updating the full model (Full) vs few-shot (FS) updates on CIFAR-10 and GSC with 100 clients.

dropout by allowing them to perform the computationally efficient few-shot updates rather than full model updates, thus enabling their contribution to FL.

FedFSC+ improves upon FedFSC by enabling contributions from stragglers. Algorithm 1 essentially designates K non-straggler to handle all the workload during a FL round, performing both full updates and the additional few-shot updates. There, stragglers are effectively discarded. However, FedFSC+ modifies the client operation in Algorithm 1 (line 3-11) to allow the identified stragglers to perform the few-shot updates and contribute less biased classifiers. Meanwhile, non-stragglers are tasked only with training the base model. FedFSC+ selects stragglers to perform the few-shot updates randomly at each communication round to reflect the realistic situation where identified stragglers could be different every time.

FedFSC+ operates under the assumption that the stragglers can be identified during a communication round. In practice, this assumption could be resolved by including active client selection, disclosing local resource information and workload capacity, keeping a history of client participation, etc. For instance, FedCS Nishio and Yonetani (2019) proposes a client selection protocol where clients share their resource information with an operator. Following this protocol, FedCS selects clients in a manner that prioritises clients capable of successfully completing local updates during a synchronised communication round. FedFSC+ could utilise such protocols to identify its stragglers.

Algorithm 2 FedFSC+ with reduced workloads for stragglers.

```
1: Input: total $T$ rounds, $E$ full updates epochs, $E_{fs}$ few-shot update epochs, $\theta$
   classifier, $\phi$ feature extractor, total $C$ clients
2: for $t = 1, \dots, T$ do
3:     $K$ random non-stragglers and additional $S$ random stragglers are available
       for training. They download the base model $\{\theta^t, \phi^t\}$ from the server.
4:     Clients:
5:     for Client $k \in [1, K]$ do
6:         $\theta_k^{t+1}, \phi_k^{t+1} \leftarrow$ Full updates $(\theta^t, \phi^t; \mathcal{D}_k, E)$ according to Equation 2.3.
7:     end for
8:     if $t \geq 2$ then
9:         for Client $s \in [1, S]$ do
10:            $\theta_{fs,\,s}^t \leftarrow$ Few-shot updates $(\theta^t; \phi^t, \mathcal{D}_s, E_{fs})$ according to Equa-
               tion 4.1.
11:        end for
12:    end if
13:    Non-straggler client $k$ sends $\theta_k^{t+1}$, $\phi_k^{t+1}$, and straggler client $s$ uploads $\theta_{fs,\,s}^t$
       to the server.
14:    Server:
15:    Identical to Algorithm 1.
16: end for
```

4.5 Experiments

FedFSC and FedFSC+ are compared with FedAvg, FedProx and MOON. In addition to the general experimental setup introduced in the Section 3, the specific setup in these experiments is detailed as follows.

FedFSC implementation details. CIFAR-10, CIFAR-100 and GSC datasets are employed for the evaluation. We exercise $Diri(\alpha)$ with $\alpha = 0.1$ for strong data heterogeneity, and $\alpha = 0.5$ for weak data heterogeneity. The Wide ResNet with the depth of 16 (WRN-16-1) is used as the distributed model. Local updates adopt the SGD optimiser. For full local updates, the optimiser has a learning rate of 0.1 with momentum of 0.5 and weight decay of 0.00001. Regarding the optimisation for the few-shot updates, we refer to the setup used in the original few-shot learning work (Chen et al., 2019b). Grid search among 0.001, 0.01 and 0.05 is applied for tuning the learning rate for the few-shot updates on the validation set of CIFAR-10 as shown in Table 4.2. It is found that the learning rate used in the original work performs best. So we borrow their setup for our few-shot updates, setting learning rate to 0.01 with a momentum of 0.9 and weight decay of 0.001 for the optimiser. We also follow the practice of the original work where the same optimisation setup is applied universally to several datasets by transferring the few-shot optimiser of CIFAR-10 to CIFAR-100 and GSC. The shot number for the few-shot update is set to 10, meaning each class in the local data contributes 10 randomly selected samples to train the classifier. Unless otherwise stated, the full updates epochs E is fixed to 10 for FedAvg, FedProx and MOON. Similarly, the few-shot update epochs E_{fs} is set 10 as well for a fair comparison. Table 4.1 summarises the specific experimental setup for FedFSC and its compared baselines.

4.5.1 FedFSC with Full Client Participation

A simple FL scenario with 10 clients is set up to compare FedFSC with the baselines. All clients are assumed to participate in FL at each round without being dropped out.

Hyperparameter	CIFAR-10		CIFAR-100		GSC	
	full	fs	full	fs	full	fs
Optimiser	SGD	SGD	SGD	SGD	SGD	SGD
Learning rate	0.1	0.01	0.1	0.01	0.1	0.01
Momentum	0.5	0.9	0.5	0.9	0.5	0.9
Weight decay	1e-5	1e-3	1e-5	1e-3	1e-5	1e-3
Batch size	64	4	64	4	128	4
Local epochs	10	10	10	10	10	10
Reduced local epochs	1	na	1	na	1	na
Few-shot number	na	10	na	10	na	10
$\mu_{prox}(Diri(0.1\&0.5))$	0.1	na	0.01	na	0.1	na
$\mu_{MOON}(Diri(0.1))$	0.1	na	0.01	na	0.1	na
$\mu_{MOON}(Diri(0.5))$	1	na	0.01	na	0.1	na

Table 4.1: Hyperparameters used by the baseline/baseline+ and FedFSC/FedFSC+. “fs” is the abbreviation for “few-shot”.

Learning Rate	0.001	0.01	0.05
Val. Acc. (%)	60.99	62.94	62.16

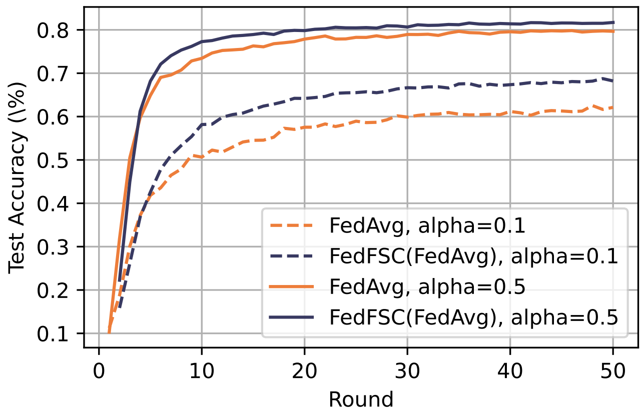
Table 4.2: The validation performance on CIFAR-10 by varying learning rates for the few-shot updates in FedFSC.

On all three datasets, we compare the test accuracy of the global model achieved on the test set between FedFSC and the baselines. The details of the accuracy metrics have been introduced in Section 3.3. Recall that FedFSC collaborates with a FL solution to train the base model inside FL loops, as the feature extractor of the base model is subsequently utilised by the few-shot updates. For a fair comparison, pairwise comparisons between FedFSC and their corresponding baselines are conducted. For example, FedAvg is compared with FedFSC using FedAvg to train the base model, denoted as FedFSC(FedAvg). Table 4.3 reports the test accuracy of the global model. In the scenario of FedFSC with full client participation, all experiments are run for 50 communication rounds as we observe that the global model learns very fast without the dropout of clients and starts showing saturated performance beyond 40 rounds. Each configuration performs 4 repeated runs using different seeds to generate non-IID local data. The mean and standard deviation of the test accuracy are reported. In addition, the centralised training results are reported for reference, showing the gap between heterogeneous FL and ideal centralised training.

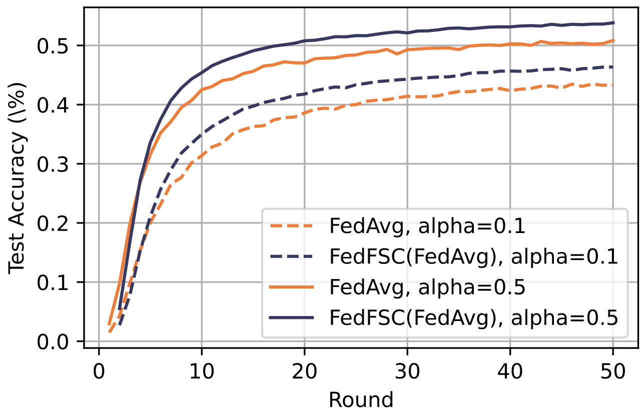
Results and analysis. Owing to the model shift mitigation introduced by the few-shot update, FedFSC is witnessed with advantageous performance across all scenarios. It outperforms the employed baselines by up to 6%, 2.9%, 1.5% on CIFAR-10, CIFAR-100, GSC respectively. In terms of the relative improvement, the performance gap from FL to the centralised learning is reduced by up to 47%, 23%, and 65% on the three datasets. On the other hand, the performance boost from FedFSC is less pronounced on GSC compared to CIFAR-10 and CIFAR-100, particularly given a weak data heterogeneity with $Diri(0.5)$. We hypothesis the minor improvement observed on GSC is due to the potential for FedFSC to mitigate model shift. Particularly, GSC is a simpler dataset than CIFAR-10 and CIFAR-100 with respect to the information containing in the data. GSC instances are one-channel images transformed by Mel spectrogram. In contrast, images in CIFAR-10 and CIFAR-100 have 3 channels. The strong performance achieved on GSC, with over 91% test accuracy, further suggests GSC is simpler than CIFAR-10 and CIFAR-

$Diri(\alpha)$	Method	CIFAR-10	CIFAR-100	GSC
0.1	FedAvg	62.95±2.84	43.64±1.19	83.73±2.09
	FedFSC(FedAvg)	68.98±2.75 ↑	46.56±0.74 ↑	84.90±1.81 ↑
	FedProx	58.76±5.04	43.76±0.99	77.46±3.90
	FedFSC(FedProx)	63.21±4.75 ↑	45.82±1.03 ↑	78.63±3.30 ↑
	MOON	64.34±1.89	43.59±0.75	83.69±2.39
	FedFSC(MOON)	69.75±2.10 ↑	46.45±0.99 ↑	85.11±1.84 ↑
0.5	FedAvg	80.48±1.79	51.00±0.98	91.27±0.22
	FedFSC(FedAvg)	81.94±1.03 ↑	53.95±0.59 ↑	91.71±0.15 ↑
	FedProx	76.03±1.18	50.49±0.11	87.13±0.32
	FedFSC(FedProx)	77.66±0.76 ↑	53.32±0.36 ↑	87.62±0.25 ↑
	MOON	80.89±1.21	50.45±0.60	91.23±0.38
	FedFSC(MOON)	81.39±1.03 ↑	53.55±0.29 ↑	91.59±0.26 ↑
IID	Centralised	83.57	64.00	91.95

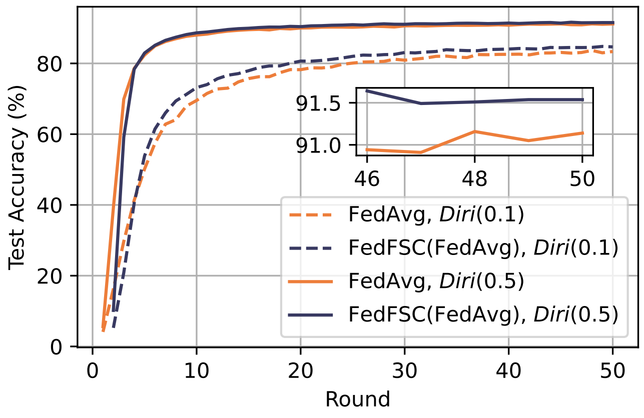
Table 4.3: FedFSC with full client participation. Top test accuracy achieved by FedFSC paired with FedAvg, FedProx, MOON. 10 clients with 100% participation are assumed. The bracketed solution attached to FedFSC indicates it is used by FedFSC to train the base model.



(a) CIFAR-10



(b) CIFAR-100



(c) GSC

Figure 4.6: Learning curves of FedFSC. Experiments assume 10 clients with 100% participation.

100 for the model to learn. With a simple dataset, the feature extractor could be overly parameterised for encoding the raw data. Consequently, the feature extractor is capable of capturing high-level representations of simple data at its bottom layers. And the model shift at lower layers is less significant with the presence of data heterogeneity, as show by the study of CCVR (Luo et al., 2021). As FedFSC mitigates the model shift by freezing the feature extractor during local updates, the insignificant model shift of the feature extractor on a simpler dataset reduces the effectiveness of FedFSC for mitigating the model shift, resulting in to marginal improvement. Although the improvement on GSC is marginal, FedFSC still achieves a test performance just 0.2% below the ideal centralised training on GSC with $Diri(0.5)$. Figure 4.6 further visualises the test performance of FedFSC with baseline FedAvg during training. After the feature extractor is initialised in the first few rounds, it becomes evident that FedFSC consistently outperforms the baseline by noticeable margins throughout the entire training period. In terms of FedProx and MOON, similar patterns to Figure 4.6 are observed. Finally, the advantageous performance achieved by FedFSC over its paired baseline, shown in Table 4.3, is consistent across experiments using different seeds to simulate heterogeneous client data. As showcased in Table 4.4, FedFSC(FedAvg) consistently outperforms FedAvg on our three chosen datasets across 4 runs with different seed numbers in the Dirichlet distribution to generate hetergeneous data, confirming that the observed performance gains from FedFSC are not due to randomness.

4.5.2 FedFSC+ with Partial Client Participation

FedFSC+ utilises stragglers to perform the few-shot updates. To evaluate it, the total number of clients is increased from 10 to 100 so that client participation and stragglers can be substantially simulated. The specific experiment setup for FedFSC+ is detailed as follows.

Partial client participation. We follow the widely used client participation setup introduced by FedAvg. A fraction of the clients, considered non-stragglers, are randomly selected from the client pool every round to perform full updates.

Dataset	Method	Seed 0	Seed 1	Seed 2	Seed 3
CIFAR-10	FedAvg	65.98	58.72	62.06	65.05
	FedFSC(FedAvg)	71.22 ↑	65.85 ↑	66.70 ↑	72.16 ↑
CIFAR-100	FedAvg	42.81	44.36	42.21	45.20
	FedFSC(FedAvg)	46.90 ↑	47.04 ↑	45.28 ↑	46.97 ↑
GSC	FedAvg	81.17	85.96	82.17	85.63
	FedFSC(FedAvg)	82.75 ↑	86.93 ↑	83.48 ↑	86.45 ↑

Table 4.4: The test accuracy (%) of FedFSC(FedAvg) and the paired baseline FedAvg across 4 runs with different seed numbers for splitting client data. $Diri(\alpha = 0.1)$ is used to simulate the data heterogeneity.

The remaining clients are regarded as stragglers. With respect to FedFSC+, non-stragglers are tasked solely with full updates. Additionally, a fraction of clients are randomly selected from stragglers to perform the few-shot updates. Based on the client pool, we use the notation f_n to define the fraction of non-stragglers and f_{fs} to determine the fraction of stragglers that perform few-shot updates.

Baseline+ for fair comparisons. For a fair comparison with FedFSC+, all baselines are allowed to leverage the stragglers as well. The original FedProx allows the stragglers to train the distributed model with a reduced number of epochs and upload the training results to the server instead of being discarded. However, FedAvg and MOON do not include any solution to address the straggler issue. Inspired by FedProx, the reduced local training epochs is introduced to FedAvg and MOON as well, letting the stragglers defined by f_{fs} in the baselines to perform E_r epochs of local updates. E_r is set to be smaller than the E full updates epochs imposed on non-stragglers. By this, all baselines leverage the stragglers just like FedFSC+. We use *baseline+* to denote that a baseline is altered, including FedAvg+, FedProx+ and MOON+. With E being set to 10, the reduced epoch E_r is set to 1 in the experiments. This is because Figure 4.5 empirically shows that the few-shot updates take approximately 90% less time than for full model updates. In terms of utilising the computational resources on stragglers, E_r is 10% of E for a fair comparison with few-shot updates.

FedFSC+(baseline+): no straggler identification needed. Based on baseline+, FedFSC+(baseline+) is introduced as an advanced variant of FedFSC+ for comparison. We incorporate baseline+ into FedFSC+ to indicate it also allows stragglers to perform full updates with reduced local update epochs, similar to baseline+. FedFSC+(baseline+) can effectively relax FedFSC+ from identifying the stragglers in advance. Specifically, a potential straggler participant in FedFSC+(baseline+) initiates its local updates by fully training the model like non-stragglers. Using the training information, the client can dynamically estimate whether it can complete the full updates before the deadline or not. If negative, the less powerful client can opt to perform the few-shot updates on the classifier. To this end, it provides the server with a model fully trained with a reduced number of epochs, along with a valuable few-shot updated classifier. For a fair comparison with baseline+, the setup from the baseline+ is reused by setting E_r to 1, which assumes straggler participants transition to few-shot updates after completing 1 epoch of full updates.

Both f_n and f_{fs} are set to 0.1 for FedFSC+ and baseline+. f_n is set to 0.1 to simulate a low participation rate as commonly used in prior works. f_{fs} is set to 0.1 to assume that not all the stragglers can be saved even by the reduced computations by introducing the condition of $f_n + f_{fs} < 1$. This last condition is considered to cover the cases when devices go offline for a prolonged period of time, or they simply run out of battery during the update round. Experiments are run on CIFAR-10 and GSC for 250 rounds and CIFAR-100 for 400 rounds until convergence. Since only a fraction of clients participate in FL during each round in the scenario of partial client participation, we observe significantly slower learning for the global model. Consequently, the number of communication rounds is considerably increased compared to scenarios with full client participation. Moreover, CIFAR-100 is the most complex dataset among the three chosen datasets as it has more unique classes. As a result, the global model learns even slower on CIFAR-100 compared to CIFAR-10 and GSC, leading us to increase the communication rounds more drastically on this dataset. Regarding the model, WRN-40-1 is employed for the scenario of partial client participation.

Results and analysis. Table 4.5 and Table 4.6 reports the test accuracy of the global model by comparing the baseline with FedFSC(baseline), baseline+ with FedFSC+(baseline) and FedFSC+(baseline+). With the inclusion of stragglers and partial client participation, our observations are twofold. First, both FedFSC and FedFSC+ outperform their paired baselines in all scenarios, without any exceptions. The superior global performance achieved by FedFSC and FedFSC+, up to over 2% improvement on the global test accuracy, demonstrates the efficacy of the few-shot learned classifier in approaching the data heterogeneity challenge given stragglers are taken into account. Second, FedFSC+ performs on par with FedFSC, suggesting the few-shot updated classifier is effective regardless of whether the few-shot update is applied to non-stragglers or stragglers. This means that FedFSC+ can effectively adapt to practical FLs where the straggler issue is combined with data heterogeneity, without compromised performance.

4.5.3 Regularising Local Updates with Few-Shot Learned Classifier

The base model trained in FedFSC+ provides the few-shot updates with the feature extractor. In this section, we approach a question, *Can the base model training benefit from utilising the few-shot learned classifier?* Inspired by the regularisation method proposed by FedProx, the few-shot learned classifier is introduced to regularise local updates.

Formally, θ_{fs} is distributed to the client side to formulate the regularisation term, minimising the differences between the fully updated classifier and the few-shot updated classifier in the local empirical loss, as described by Equation 4.3.

$$L_k(\theta, \phi; \theta_{fs}) = L_k(\theta, \phi) + \mu_{fsc} \left\| \theta - \theta_{fs} \right\|^2 \tag{4.3}$$

$Diri(\alpha)$	Method	CIFAR-10	CIFAR-100	GSC
0.1	FedAvg	67.91±0.87	46.69±0.63	88.25±0.29
	FedFSC(FedAVg)	70.14±1.12 ↑	48.25±0.22 ↑	88.85±0.39 ↑
	FedAvg+	68.70±1.08	47.96±0.82	88.21±0.67
	FedFSC+(FedAVg)	70.29±0.54 ↑	48.49±0.41 ↑	88.81±0.31 ↑
	FedFSC+(FedAVg+)	70.55±1.34 ↑	48.77±0.49 ↑	88.81±0.46 ↑
	FedProx	66.48±0.61	46.68±0.14	87.78±0.58
	FedFSC(FedProx)	68.49±0.85 ↑	47.63±0.36 ↑	88.34±0.39 ↑
	FedProx+	68.61±1.34	47.85±0.59	87.71±0.87
	FedFSC+(FedProx)	68.74±0.47 ↑	48.20±0.30 ↑	88.38±0.42 ↑
	FedFSC+(FedProx+)	69.53±1.08 ↑	48.61±0.61 ↑	88.32±0.59 ↑
	MOON	68.60±0.83	46.93±0.10	88.16±0.49
	FedFSC(MOON)	70.90±0.79 ↑	47.86±0.40 ↑	88.96±0.20 ↑
	MOON+	68.11±0.96	47.89±0.58	88.53±0.20
	FedFSC+(MOON)	70.73±0.82 ↑	48.36±0.24 ↑	88.78±0.26 ↑
	FedFSC+(MOON+)	70.61±1.02 ↑	48.51±0.49 ↑	88.75±0.31 ↑

Table 4.5: FedFSC and FedFSC+ with partial client participation. With a total of 100 clients, $f_n = 0.1$, and $f_{fs} = 0.1$, the top test accuracy achieved by FedFSC and FedFSC+ and their paired baseline and baseline+.

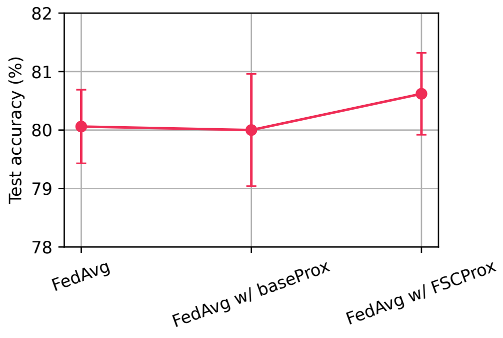
$Diri(\alpha)$	Method	CIFAR-10	CIFAR-100	GSC
0.5	FedAvg	80.06±0.63	52.98±0.31	91.40±0.17
	FedFSC(FedAVg)	80.96±0.27 ↑	54.11±0.47 ↑	91.46±0.16 ↑
	FedAvg+	80.05±0.63	53.26±0.23	91.13±0.11
	FedFSC+(FedAVg)	81.23±0.35 ↑	53.99±0.14 ↑	91.55±0.16 ↑
	FedFSC+(FedAVg+)	81.21±0.31 ↑	54.00±0.33 ↑	91.35±0.16 ↑
	FedProx	78.44±0.31	52.86±0.26	90.78±0.14
	FedFSC(FedProx)	79.60±0.12 ↑	53.90±0.31 ↑	90.96±0.06 ↑
	FedProx+	79.55±0.09	53.10±0.26	90.69±0.08
	FedFSC+(FedProx)	79.48±0.29	53.61±0.11 ↑	91.00±0.11 ↑
	FedFSC+(FedProx+)	80.12±0.24 ↑	53.77±0.19 ↑	90.86±0.03 ↑
	MOON	80.60±0.20	53.46±0.95	91.29±0.17
	FedFSC(MOON)	81.11±0.37 ↑	54.45±0.19 ↑	91.47±0.11 ↑
	MOON+	80.56±0.24	53.45±0.26	91.18±0.08
	FedFSC+(MOON)	81.34±0.18 ↑	54.44±0.63 ↑	91.45±0.13 ↑
	FedFSC+(MOON+)	81.37±0.24 ↑	54.06±0.33 ↑	91.33±0.09 ↑

Table 4.6: FedFSC and FedFSC+ with partial client participation. Continue from Table 4.5.

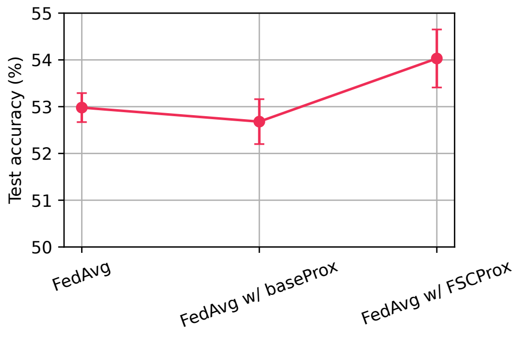
where the regularisation term is $\|\theta - \theta_{fs}\|^2$, which is weighted by μ_{fsc}. As the regularisation term adopted above has a similar form to the proximal term proposed by FedProx, it is therefore denoted as *FSCProx*. To demonstrate the efficacy of FSCProx, we compare it with the regularisation method utilising the classifier obtained directly from the base model, denoted as *baseProx*. We tune μ_{fsc} for the best global performance of FL.

Our experiments show that global model performance in FedAvg is improved by utilising the few-shot learned classifier. This observation equivalently demonstrates that the base model learned in FedFSC+(FedAvg) can be improved using the few-shot learned classifier, as the global model learned in FedAvg is essentially the base model learned in FedFSC+(FedAvg). On CIFAR-10 and CIFAR-100 with $Diri(0.5)$, Figure 4.7 and Figure 4.8 compare FedAvg, FedAvg with baseProx, and FedAvg with FSCProx in terms of the test accuracy and training losses of the global model. Experiments are run for 250 and 400 communication rounds for CIFAR-10 and CIFAR-100 respectively. The introduction of baseProx is observed to bring no benefit to global model performance in terms of either metrics. For training losses, while FedAvg with baseProx is on par with FedAvg, the utilisation of FSCProx effectively reduces the training losses of the global model. With respect to the test accuracy, the adoption of baseProx even diminishes global model performance, whereas FSCProx boosts the performance by a noticeable margin. Thus, this study shows that global model performance is improved when its classifier is adjusted to resemble a less biased classifier updated by few-shot learning in local updates. Conversely, its performance deteriorates when approaching a biased classifier that is fully updated.

Finally, on all three datasets, Table 4.7 further reports the test accuracy of the global model achieved by FedAvg with FSCProx, FedAvg and FedProx. Additionally, FedFSC+(FedAvg) is compared with its variant using FSCProx. With data heterogeneity of $Diri(0.1)$ and $Diri(0.5)$, it is observed that FSCProx can effectively boost the performance of FedAvg, outperforming FedProx across all scenarios. However, FSCProx has only a minor impact on FedFSC+(FedAvg). Given $Diri(0.5)$, the introduction of FSCProx slightly improves FedFSC+(FedAvg) on all

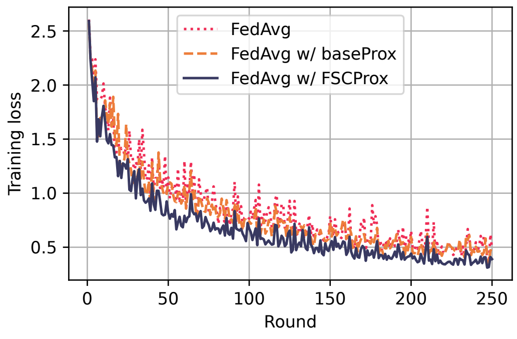


(a) CIFAR-10

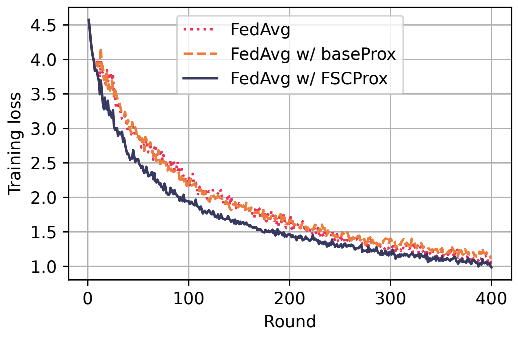


(b) CIFAR-100

Figure 4.7: Regularising local updates with the baseProx and FSCProx (test accuracy of the global model). FedAvg uses baseProx or FSCProx to regularised its local updates. $Diri(alpha = 0.5)$ is used for client data partition.



(a) CIFAR-10



(b) CIFAR-100

Figure 4.8: Regularising local updates with the baseProx and FSCProx (training losses of the global model). FedAvg uses baseProx or FSCProx to regularised its local updates. $Diri(alpha = 0.5)$ is used for data partition.

three datasets. When $Diri(0.1)$ is applied, FSCProx improves FedFSC+(FedAvg) on GSC only.

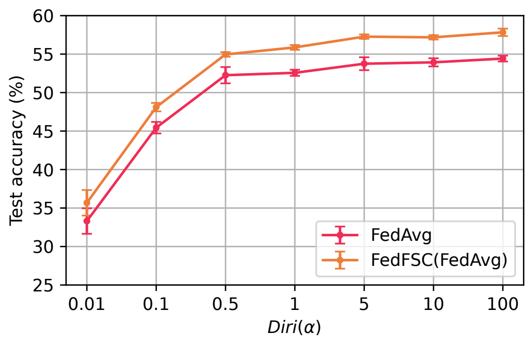
4.5.4 Ablation Studies on FedFSC and FedFSC+

The impact of important hyperparameters in FedFSC and FedFSC+ is further studied by varying their values.

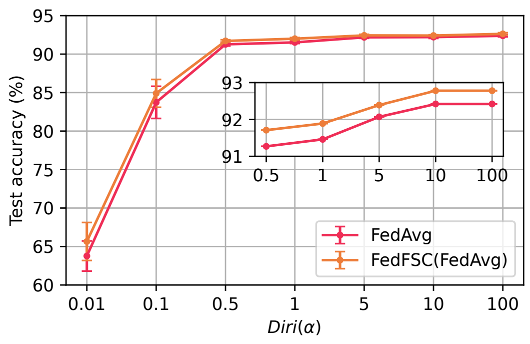
Data heterogeneity with $Diri(\alpha)$ In addition to $Diri(0.1)$ and $Diri(0.5)$, various α values are used to examine the impact of data heterogeneity on the proposed

$Diri(\alpha)$	Method	CIFAR-10	CIFAR-100	GSC
0.1	FedAvg	67.91±0.87	46.69±0.63	88.25±0.29
	FedProx	66.48±0.61	46.68±0.14	87.78±0.58
	FedAvg **w/** FSCProx	69.39±0.55↑	47.90±0.44↑	88.56±0.41↑
	FedFSC+(FedAvg)	70.29±0.54	48.49±0.41	88.56±0.41
	FedFSC+(FedAvg) **w/** f	70.05±0.91	48.02±0.38	89.01±0.31↑
0.5	FedAvg	80.06±0.63	52.98±0.31	91.40±0.17
	FedProx	78.44±0.31	52.86±0.26	90.78±0.14
	FedAvg **w/** FSCProx	80.62±0.70↑	54.03±0.62↑	91.54±0.16↑
	FedFSC+(FedAvg)	81.23±0.35	53.99±0.14	91.55±0.16
	FedFSC+(FedAvg) **w/** f	81.42±0.43↑	54.13±0.49↑	91.68±0.18↑

Table 4.7: Regularising local updates with FSCProx. The upper bracket compares the global model performance of FedAvg, FedProx and FedAvg+FSCProx. The lower bracket compares the global model performance of FedFSC+(FedAvg) and its variant using FSCProx.



(a) *Diri*(α): CIFAR-100



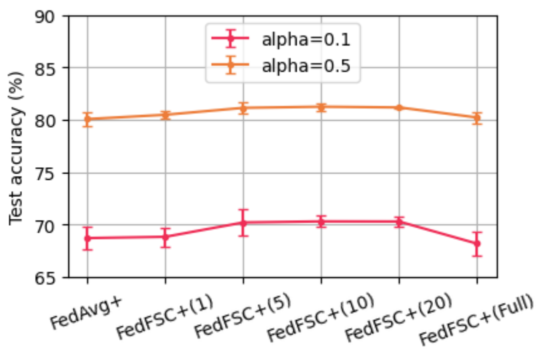
(b) *Diri*(α): GSC

Figure 4.9: Ablation study 1: varying *Diri*(α). Global model test performance of FedAvg and FedFSC(FedAvg) on CIFAR-100 and GSC using various α values for *Diri*(α).

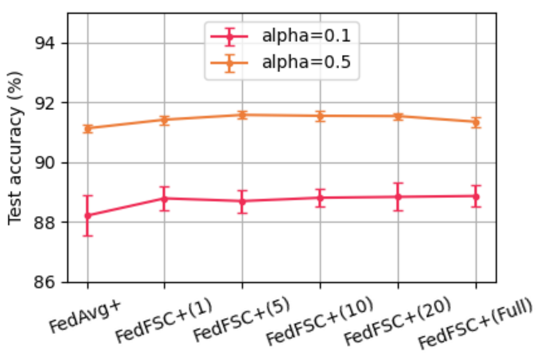
solution. This ablation study focuses solely on varying data heterogeneity without the simulation of stragglers. Therefore, FedFSC(FedAvg) is evaluated using the previous experimental setup involving 10 clients, with 100% client participation. CIFAR-100 and GSC datasets are employed for this study. Figure 4.9 illustrates the test accuracy of the global model trained with α ranging from 0.01 to 100, simulating extreme data heterogeneity cases and approaching IID scenarios.

In terms of CIFAR-100, it is observed that the improvement of the global performance made by FedFSC(FedAvg) is consistent given different data heterogeneity levels, ranging between 2.3% and 3.4%. The performance gap between FedAvg and the advantageous FedFSC(FedAvg) slightly narrows with smaller α values, such as 0.01 and 0.1. We suspect this is due to the slow convergence induced by strong data heterogeneity. With α values greater than 1, FedFSC(FedAvg) consistently outperforms FedAvg by around 3.3%.

On the other hand, FedFSC shows minor boosts in global model performance on GSC, especially as local data approaches IID scenarios. With α values greater than 0.5, the global test accuracy plateaued at round 92%. This finding aligns with our earlier experiment results, suggesting the model shift has a minor impact on the global performance with GSC.



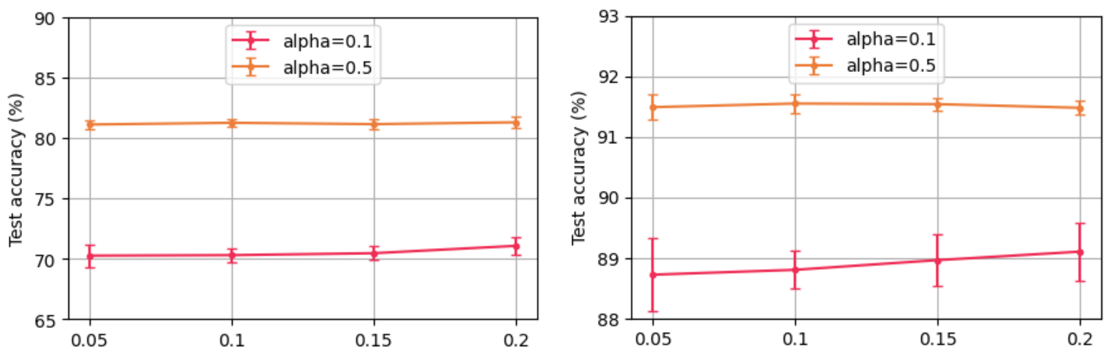
(a) Shot number: CIFAR-10



(b) Shot number: GSC

Figure 4.10: Ablation study 2: varying the few-shot number. The performance of FedFSC+ using various few-shot numbers on CIFAR-10 and GSC.

Few-shot numbers. Choosing an appropriate few-shot number is critical for few-shot learning. As such, different few-shot numbers are investigated to understand its impact on FedFSC. Figure 4.10 depicts the performance of FedFSC+(FedAvg) with 1, 5, 10, 20, full shots on CIFAR-10 and GSC. Full shots training suggests all local data points are used to train the classifier instead of randomly selecting training samples. In summary, slight performance decreases are noticed in FedFSC+(FedAvg) with both 1 shot and full shots on CIFAR-10. It is unsurprising that performance degrades when fewer training samples are used. However, utilising more samples for few-shot updates does not come with further performance increase. Full shots updates even incur a performance drop. We suspect the performance decrease with full shots is due to the reenactment of model shift in the classifier. Unlike few-shot updates, where each class selects an equal number of training samples, training the classifier on the entire non-IID local data results in classes contributing uneven numbers of training samples. Therefore, full-shots updates shift the classifier to make predictions biased to dominant classes in the local data. Finally, FedFSC+ on GSC is relatively more resilient to changes in the shot number compared to CIFAR-10. This aligns with the observation that FedFSC+ achieves the least improvement over the baselines on GSC.



(a) f_{fs}: CIFAR-10 **(b)** f_{fs}: GSC

Figure 4.11: Ablation study 3: varying the number of clients performing few-shot updates (f_{fs}). The performance of FedFSC+ using various numbers of clients performing the few-shot updates on CIFAR-10 and GSC.

The number of clients performing the few-shot updates. It is reasonable to assume that quite a few clients may not even complete the efficient few-shot updates in the real-world scenario. As such, a substantial ablation study is to understand how the number of clients performing few-shot updates affects FedFSC+. To do this, we vary f_{fs} with values of $\{0.05, 0.1, 0.15, 0.20\}$, which accounts for 5, 10, 15, 20 clients performing few-shot updates out of a total of 100 clients. Figure 4.11 presents the performance trend as f_{fs} changes on CIFAR-10 and GSC. It is observed that FedFSC+ maintains its advantageous performance even when the number of clients performing few-shot updates are reduced to just 5. Therefore, we can confidently rely on a small number of clients to perform the few-shot updates to boost the global performance of FL. Such a relaxed criterion is substantially valuable in practical applications. On the other hand, it is noticed that increasing f_{fs} cannot further improve the performance of FedFSC+ significantly.

Utilising the fully updated classifier. FedFSC and FedFSC+ use the few-shot updated classifier as a component to form the global model. Alternatively, we can create the classifier of the global model by combining both the few-shot updated classifier θ_{fs}^t and the fully updated classifier θ^{t+1}. To investigate this variant, we perform a naive averaging of the classifier from the fully updated model, θ^{t+1} and the few-shot updated classifier, θ_{fs}^t, both formed on the server, with equal weights.

Dataset	α	w/o θ^{t+1}	w/ θ^{t+1}
Cifar10	0.1	70.29±0.54	69.89±0.48↓
	0.5	81.23±0.35	80.46±0.17↓
GSC	0.1	88.81±0.31	88.75±0.38↓
	0.5	91.55±0.16	91.54±0.17↓

Table 4.8: Ablation study 4: utilising the fully updated classifier. The performance of FedFSC+ with or without the fully updated classifier in the global model.

Then the averaged classifier is used to construct the global model.

This variant is compared with the original FedFSC+ in Table 4.8 in terms of the test accuracy of the global model on CIFAR-10 and GSC. It is apparent that combining the fully updated classifier introduces a negative impact on FedFSC+, resulting in up to 0.8% performance decrease on CIFAR-10. Since the fully updated classifier is shown to be more biased than the few-shot updated classifier in this study, the observation indicates that its inclusion into the global model of FedFSC/FedFSC+ damages its performance.

4.5.5 Model Architecture Variation

This section extends the evaluation of FedFSC+ to various model architectures beyond Wide ResNet. Specifically, MobileNetV3 (Howard et al., 2019) and EfficientNet (Tan and Le, 2019) are employed. MobileNetV3 and EfficientNet are computationally efficient CNNs that are particularly developed for running on resource-constrained devices (Rosero-Montalvo et al., 2024). By demonstrating the effectiveness of FedFSC+ on them, we can bridge FedFSC+ to the practical FL applications where resource-constrained devices are prevalent, such as FL on IoTs (Zhang et al., 2022c; Choudhry et al., 2024; Han et al., 2024a). MobileNetV3 and EfficientNet are improved with network architecture search (NAS). MobileNetV3 is particularly designed for model training on mobile phones with CPUs, while EfficientNet is a collection of CNNs that are known for their faster inference speed compared to some of the most powerful existing CNNs.

In addition to the image classification and speech recognition tasks, we further investigate the performance of FedFSC+ in a Natural Language Processing (NLP) task using the AG news dataset (Gulli, 2005). AG news is a corpus of news articles collected from more than 2000 news sources. Articles are categorised by different topics. We follow the practice proposed by Zhang et al. (2015) to select the four largest classes, namely "World", "Sports", "Business", "Sci/Tech", to compose the dataset. This results in a total of 120,000 training samples and 7,600 test samples. This NLP task aims to train the model to categorise the news articles into correct topics. In this experiment, the pretrained 100D Glove embeddings (Pennington et al., 2014) are adopted to build the vocabulary and encode the tokenised samples from the AG news dataset. All embeddings remain frozen during training.

MobileNetV3(small) and EfficientNet-B0 are trained with FedFSC+(FedAvg) on the CIFAR-10 dataset. The local batch size is tuned to 32, while the rest of the hyperparameters are borrowed directly from previous FedFSC+ experiments. For AG news classification, a two-layer Long Short-Term Memory (LSTM) model with a hidden dimension of 256 is used as the feature extractor and the classifier is a fully-connected layer that is placed on top of the feature extractor. The local batch size is set to 64, the full local epochs, E, is set to 5, and the learning rate is tuned to 1. As shown in Table 4.10, few-shot updates are observed to need a few samples, such as 10 and 20, drawn from each class for fine-tuning the classifier to achieve optimal performance for FedFSC+ on CIFAR-10 and GSC. Detailed explanation about the effect of the few-shot number on FedFSC+ can be referred to our second ablation study in Section 4.5.4. As individual class in AG news has a significantly larger number of samples (30000) than CIFAR-10 and GSC, we increase the few-shot number used by FedFSC+ on AG news. We set the shot number to 60 and the batch size for few-shot updates to 32 in this study.

Table 4.9 summarises the model details and compares the test accuracy of the global model between FedFSC+(FedAvg) and FedAvg. All experiments are run for 100 communication rounds with $Diri(0.5)$ to split the client data. With the simulation of partial client participation, FedFSC+ is observed to consistently out-

Model	#Para	Dataset	FedAvg	**FedFSC**+(FedAvg)
MobileNetV3(small)	1.7m	CIFAR-10	52.49±1.19	54.20±0.48↑
EfficientNet-B0	4.0m	CIFAR-10	69.66±0.59	70.51±0.63↑
2-layer-LSTM	0.9m	AG News	75.93±2.53	77.56±0.78↑

Table 4.9: Evaluating FedFSC+ with different model types. Top test accuracy achieved by FedAvg and FedFSC+(FedAvg) with MobileNetV3 (small), EfficientNet-B0, LSTM, are reported.

perform the baseline FedAvg across all model architectures by nearly 2% in global test accuracy.

4.6 Summary

In this chapter, FedFSC is proposed to close the research gap of tackling the model shift problem in the partial model training approach. By adopting few-shot fine-tuning on stragglers, FedFSC successfully answers the research questions: "How efficient is partial model training?" and "How can the model shift be tackled using partial model training?".

This work demonstrates that the proposed few-shot update is an efficient partial model training method, reducing the training time by 90% compared to the standard full model update. Moreover, the few-shot learned classifier is shown to be less biased compared to the fully updated classifier on non-IID local data. FedFSC is constructed based on these profound discoveries, introducing few-shot updates into FL to compose a global model that benefits from the contributions of stragglers and is resilient to model shift. The only condition that FedFSC comes with is to have at least a few computationally capable devices (non-stragglers) to train a generic feature extractor that assists few-shot updates.

Our experiments demonstrate that FedFSC is compatible with popular FL solutions. FedFSC consistently boosts the global model performance by up to 6% in various scenarios, including image, speech and text data. It is also observed that FedFSC achieves faster convergence compared to baselines, further reducing the overall energy cost across clients.

FedFSC reduces the workloads on resource-constrained devices by allowing them to train a part of the models with a reduced number of parameters. Client models in FedFSC are still assumed to have a homogeneous architecture. In the next chapter, we will focus on improving FL performance in the approach of custom-size client models, which also reduces the trainable parameters in client models by allowing weak clients to design their own lightweight models rather than training a partial model.

Chapter 5

Improving Knowledge Transfer to the Global Model

5.1 Introduction

5.1.1 Research Questions to Address

The approach of custom-size client models allows clients to design models that match their hardware capabilities, rather than the server distributing a uniform model without discrimination. Notably, this model customisation enables stragglers with limited computing power to use lightweight models, reducing training workloads and ensuring timely completion of local updates without being discarded by the server. However, while custom-size client models solve the straggler issues, traditional model fusion methods for updating the global model on the server are no longer feasible due to size differences of updated client models.

Previous works, as reviewed in Section 2.2, commonly distill knowledge from the output of custom-size client models and transfers the distilled knowledge to the global model. However, given data heterogeneity, the effectiveness of knowledge distillation is compromised due to the reliance on a single knowledge source, which is negatively impacted by model shift. To this end, this sub-research path approaches the research question "How can we improve the knowledge transfer from custom-size client models to the global model with multiple knowledge sources?".

To address this research question, we aim to design and evaluate methods that leverage multiple knowledge sources extracted from client models to strengthen the knowledge transfer for the approach of custom-size client models. This work will improve the performance of FL with custom-size client models, thereby increasing its efficacy in addressing the straggler issue.

5.1.2 Proposed Methods and Rationale

Attention transfer and *metadata training* are the proposed methods to enhance knowledge transfer in FL with custom-size client models and answer the research question derived in the previous section. How client models are designed to meet the client computing requirements is outside the scope of this thesis. The literature reveals solutions for producing such models via hardware-aware Neural Architecture Search (NAS) (Wu et al., 2019; Cai et al., 2018; Tan et al., 2019), reducing the block size from teacher models (Turner et al., 2019) and specialising the computing model to match the computing budget (Wen et al., 2020).

Attention transfer (Zagoruyko and Komodakis, 2016a) (AT) is a machine learning technique to enhance knowledge distillation in centralised learning. AT teaches the learner which parts of internal feature maps should it focus on during learning. This improves the performance of the learner by not only learning the representations in the output space but also the internal space. In the original study, AT has been shown to significantly boost the performance of knowledge distillation across a range of learning tasks.

Inspired by the success of AT in centralised learning, this thesis leverages AT as an additional knowledge source to improve the knowledge transfer between custom-size client models and the global model. In addition to its proven efficacy, AT is ideal FL with custom-size client models because it can be formulated to train the global model regardless of the size differences between client models and the global model. Although FedAD also introduces AT to FL, it was published six months after this work on leveraging AT to enhance knowledge transfer in FL.

On top of AT, metadata is further introduced as another knowledge source to enhance knowledge transfer. Metadata consists of latent feature maps extracted from custom-size client models. After local updates, client models extract latent feature maps locally and upload them as metadata to the server. There, the upper part of the global model is refined on metadata, obtaining client learned knowledge.

Metadata is advocated for transferring knowledge in this work for two primary reasons. First, similar to AT, metadata can be also constructed independently of model size differences. Second, the utilisation of feature maps is an established approach in centralised learning for transferring learned knowledge across domains and models. Feature maps are widely adopted in a range of machine learning research areas, including transfer learning (Xie et al., 2016; Li et al., 2019d; Neyshabur et al., 2020; Zhuang et al., 2020), multi-task learning (Misra et al., 2016; Liu et al., 2019b; Gao et al., 2020), few-shot learning (Sung et al., 2018; Wertheimer et al., 2021; Li and Bian, 2022), knowledge distillation (Srivastava et al., 2015; Yim et al., 2017; Wang et al., 2020b), just to name a few, demonstrating impressive success. However, the use of feature maps in heterogeneous FL, particularly with custom-size client models, remains very limited. FedGKT is a close study to this work in terms of training the global model with feature maps extracted from client models. However, this method adopts a different strategy for addressing the straggler issue, transferring the training a major part of the global model on the server to alleviate workloads for stragglers rather than employing custom-size client models.

The novelties of introducing AT and metadata training are twofold: 1) To the best of our knowledge, this work is the first to leverage the attention transfer technique for improving the knowledge transfer in FL; 2) Unlike previous works, metadata training and attention transfer will be integrated and evaluated in the FL framework with custom-size client models rather than unified client models, simulating an effective strategy for reducing workloads on stragglers, where stragglers are allowed to design their own models.

The adoption of AT and metadata training delivers two novel FL algorithms and contributes profound findings that successfully address the research question. Notably, **Fed**erated Global Updates with **A**ttention **T**ransfer (FedAT) and **Fed**erated Learning with Local **K**nowledge **A**ggregation and Knowlede **D**istillation (FedKAD) are presented in this chapter. In FedAT, AT and metadata are hypothesised to enable the global model on the server to generalise on unseen client data, demonstrating the efficacy of AT and metadata for transferring the knowledge. By enabling the contributions of stragglers and improving knowledge transfer with multiple knowledge sources, FedAT is expected to close the performance gap of the global model to the ideal FL baseline, where no clients are discarded as stragglers. FedKAD, on the other hand, will demonstrate that prototype feature maps, a compressed form of metadata, can alleviate negative knowledge transfer to the global model induced by model shift. To this end, the research question is answered with "We can effectively improve the knowledge transfer from custom-size client models to the global model through attention transfer and metadata training."

5.1.3 Transferring Knowledge with Attention Transfer and Metadata Training

FedAT is named after **Fed**erated Global Updates with **A**ttention **T**ransfer. FedAT facilitates knowledge transfer with AT and metadata training. Regarding AT, the latent feature maps extracted from models are fused into attention maps. By aligning the attention maps between the global model and client models, the global model learns the attention styles of client models in its latent space. With respect to metadata training, the upper part of the global model is further refined by the metadata, which is the set of feature maps extracted from clients on their local data.

Figure 5.1 displays an overview of the proposed FedAT. A generic dataset is assumed available on the server side to facilitate the attention transfer. This generic dataset does not break the FL paradigm because it does not hold any of the private data of clients. On the generic dataset, the attention maps are constructed by feature maps extracted from the global model and client models. The attention transfer in FedAT aligns the global model and client models in their latent spaces, leading to a more effective knowledge transfer. On the client side, after the client completes its local updates, the updated client model extracts latent feature maps locally, comprising the local data representations at a predetermined level of the network. The set of these feature maps alongside their associated labels is defined as *metadata* in FedAT, encoding the locally learned knowledge on the client data. The server aggregates metadata from all participating clients to train the upper layers of the global model, further enhancing the knowledge transfer in heterogeneous environments.

In a simple scenario with just one client, rigorous ablation studies are conducted to demonstrate the effects of attention transfer and metadata training for transferring knowledge, finding that the metadata training imparts the global model new knowledge learned from the client side and the attention transfer prevents the global model overfitting on the generic data with regularisation. These experiments reveal that the global model can effectively improve its generalisation using attention transfer and metadata training, by up to 6% in the single client scenario. Finally, the evaluation of FedAT is extended to a multiple client scenario on CIFAR-10, comparing it with the hypothetical FedAvg where all the clients are non-stragglers and the realistic FedAvg where some clients are stragglers and consistently discarded by the server. It is observed that FedAT can significantly close the performance gap between the realistic FedAvg and the hypothetical FedAvg, narrowing the gap from over 19% to just 0.23% in a strong data heterogeneity case. To this end, FedAT successfully demonstrates the efficacy of AT and metadata training in enhancing the knowledge transfer between custom-size client models and the global model, answering the research question.

5.1.4 Improving Knowledge Transfer with Prototype Feature Maps

Previous FL works adopting custom-size client models show that knowledge distillation on the server is susceptible to negative knowledge transfer induced by client model shift (Itahara et al., 2021; Cho et al., 2022). Typical FL works that allow custom-size client models, such as FedMD (Li and Wang, 2019), FedDF (Lin et al., 2020), DS-FL (Itahara et al., 2021), FedAUX (Sattler et al., 2021), etc, use knowledge distillation assisted by a generic dataset to transfer knowledge from client

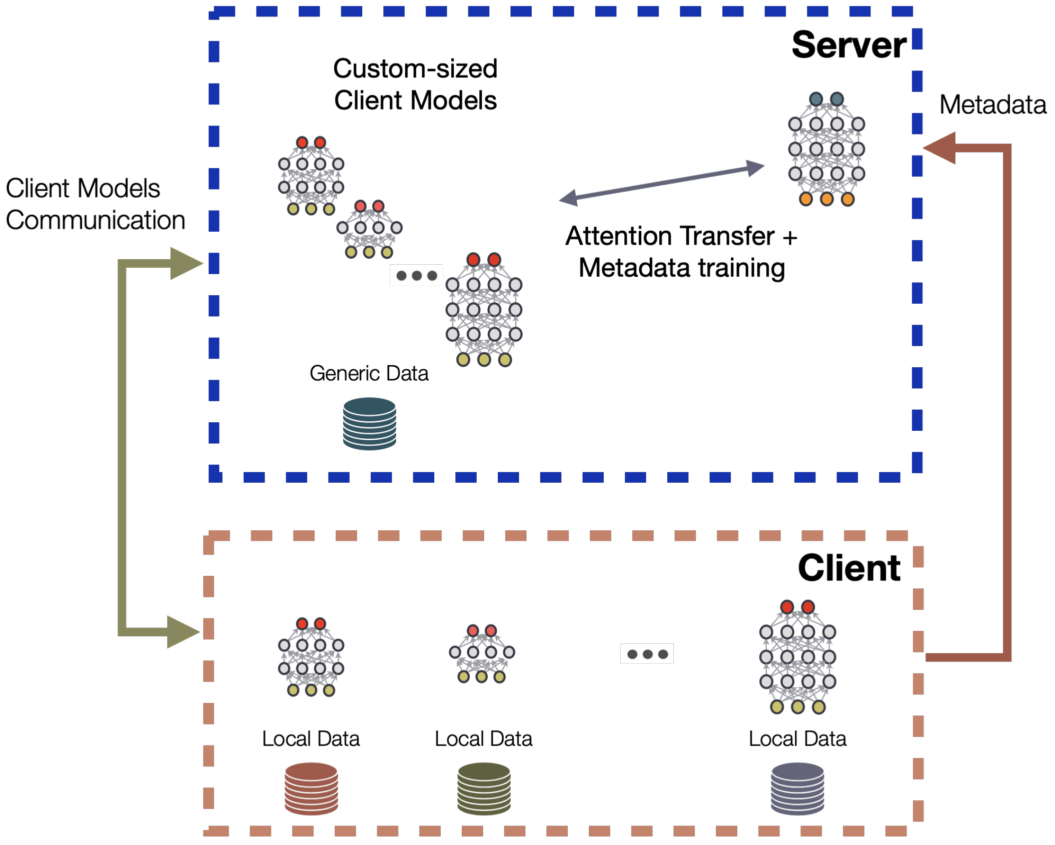

Server
Custom-sized Client Models
Metadata
Client Models Communication
Attention Transfer + Metadata training
Generic Data
Client
Local Data
Local Data
Local Data

Figure 5.1: FedAT overview. Each client uploads their updated client models and metadata to the server. Global updates are performed with attention transfer and metadata training, updating the global model and client models.

models to the global model on the server.

Notably, the server constructs a client consensus on the generic dataset by aggregating client logits. The logits are the top layer outputs of client models. The client consensus is used to supervise the global model and client models for learning on the generic dataset, thereby transferring the knowledge. However, the client consensus can be unreliable, particularly when client models are shifted drastically due to strong client data heterogeneity. This can lead to the transfer of negative knowledge to the global model, subsequently decreasing its performance.

We hypothesise that metadata can be leveraged to correct negative knowledge transfer, thereby improving the global model performance when distilling knowledge from custom-size client models to the global model. To justify this hypothesis, **Fed**erated Learning with Local **K**nowledge **A**ggregation and Knowledge **D**istillation, or FedKAD, is proposed. Concretely, FedKAD leverages prototype feature maps, which are metadata in a compressed form. These prototype feature maps are learned on clients to preserve intra-class information. FedKAD proposes to regularise KD with prototype feature maps, using them as a trustworthy knowledge source to prevent the global model from overfitting to the unreliable client consensus induced by model shift.

FedKAD is evaluated on both CIFAR-10 and GSC, and compared with uniform model baselines, FedAvg (McMahan et al., 2017) and FedProx (Li et al., 2020c), as well as a KD-based solution allowing custom-size client models across clients, FedMD. They are assessed under realistic conditions of client dropping from synchronisation rounds due to limited computational resources in uniform model distributions. It is observed that FedKAD outperforms uniform model methods with low participation rates. Compared with FedMD, FedKAD achieves advantageous performance by leveraging trustworthy knowledge from prototype feature maps, improving global model performance by up to 3% under strong data heterogeneity. Moreover, FedKAD demonstrates that client model performance is further improved by using prototype feature maps. The major contribution of FedKAD is the revelation that prototype feature maps can correct the negative knowledge transfer in knowledge distillation when the client consensus is compromised due to model shift.

This strengthens knowledge transfer quality and global model performance for the approach of custom-size client models, enhancing its application in addressing the straggler issue.

This chapter is organised into the following sections. Section 5.2 introduces the basics of knowledge distillation, attention transfer, and metadata. Section 5.3 details the FedAT algorithm, followed by Section 5.4 to present the evaluation of FedAT. The FedKAD algorithm is then described in Section 5.5, with its evaluation discussed in Section 5.6. Finally, section 5.7 concludes this chapter.

5.2 Preliminaries

Before diving into the details of FedAT and FedKAD, this section introduces the fundamental concepts employed in custom-size client models for knowledge transfer, including knowledge distillation, attention transfer, and metadata training.

5.2.1 Knowledge Distillation with Attention Transfer

Knowledge Distillation (KD) is widely accepted in previous works for facilitating knowledge transfer in FL with custom-size client models (Li and Wang, 2019; Lin et al., 2020; Sattler et al., 2021), as described in Section 2.2.3.

In centralised learning, KD is originally proposed for model compression (Buciluă et al., 2006; Hinton et al., 2015; Polino et al., 2018), which is commonly adopted for deploying computationally cheap models at large scale. KD is built on the Student-Teacher (S-T) learning framework. The fundamental idea of S-T learning is to train a lightly-parameterised model, which is referred to as the student, under the supervision of a heavily-parameterised model which is called the teacher (Wang and Yoon, 2021; Gou et al., 2021). By distilling the knowledge learned by the teacher to the student, the student model is able to achieve comparable performance to the teacher model on the target task, but with a compressed model size.

To achieve knowledge transfer, Hinton et al. (2015) proposes to align the student and teacher at their output space, comparing their softmax outputs parameterised with a hyperparameter termed temperature, as described by Equation 5.1.

$$p_i = \frac{\exp\left(\frac{z_i}{\rho}\right)}{\sum_j \exp\left(\frac{z_i}{\rho}\right)} \tag{5.1}$$

where p_i is the calculated probability of the i-th class, z_i is the logit for the i-th class, model output before it is translated into probability by the softmax, and ρ is the temperature. The temperature is introduced to the softmax activation to strengthen the information conveyed by the output.

On the target task, the student is optimised to minimise the training loss as follows:

$$L_s\left(p\left(z_s,\rho\right), p\left(z_t,\rho\right), y\right) = \mathcal{L}_{CE}\left(p\left(z_s\right), y\right) + \beta * \mathcal{L}_{KD}\left(p\left(z_s,\rho\right), p\left(z_t,\rho\right)\right) \tag{5.2}$$

where z_s and z_t are the logits from the student and teacher, y is the label of the training data, $\mathcal{L}_{CE}$ is the conventional cross entropy loss the student achieved on the target task, $\mathcal{L}_{KD}$ is the knowledge distillation loss introduced by aligning the outputs between the student and the teacher. Notably,

$$\mathcal{L}_{KD} = -\sum p\left(z_t,\rho\right)\log\left(p\left(z_s,\rho\right)\right) \tag{5.3}$$

where x is the input of the training data. N is the number of training samples. With custom-size client models, model size differences prohibit the server from using direct model fusion to transfer the learned knowledge. In contrast to direct model fusion, KD is model structure agnostic, exploiting the knowledge from client models through their outputs without requiring alignment of their structures. Thus, KD is commonly adopted in the approach of custom-size client models.

Attention Transfer (Zagoruyko and Komodakis, 2016a) (AT) extends the knowledge source of KD by focusing on transferring attention styles in the latent space from a teacher model to a student model. Unlike traditional KD, which compares final outputs, AT exploits the knowledge containing in the attention maps inside the model.

Formally, let us consider the feature maps output by a layer in the Convolutional Neural Network (CNN) when it is fed with a sample. We denote the set of feature maps as $F = \{f_i\}^C$, where $F \in R^{C\times H\times W}$. F consists of feature maps from C channels with each feature having a dimension of $H \times W$. AT fuses the feature maps from multiple channels to form an attention map, denoted as A.

$$F : R^{C\times H\times W} \to A : R^{H\times W} \tag{5.4}$$

$$A = \sum_{i=1}^{C} |f_i|^p \tag{5.5}$$

where the power p and absolute $|\cdot|$ operations on the feature map f_i are element-wise. Figure 5.2 illustrates the examples of attention maps extracted on the CIFAR-10 dataset by a wide ResNet. It shows that the attention map from the lower layer has high resolution. As the layers go deeper, attention becomes more focused on specific areas. This observation aligns with the classic finding in Krizhevsky et al. (2012).

Once feature maps are fused into the attention map, the student compares its attention maps with the teacher's on training samples to form the attention transfer loss as follows:

$$\mathcal{L}_{AT} = \sum_{l\in\mathcal{I}} \left\| \frac{Q_s^l}{\|Q_s^l\|_2} - \frac{Q_t^l}{\|Q_t^l\|_2} \right\|_p \tag{5.6}$$

where we follow the original notation defined in (Zagoruyko and Komodakis, 2016a), use $\mathcal{I}$ to denote all layer indices that the student and teacher compare their attention maps. A_s and A_t are the attention map from the student and teacher, and $Q = vec(A)$ is the vectorised form of the attention map, p is the norm type, we use $p = 2$ unless stated otherwise. Finally, the AT loss replaces the KD loss depicted in Equation 5.3 to form the student training loss. Figure 5.3 depicts the S-T learning with AT.

$$L_s\left(p\left(z_s,\rho\right), p\left(z_t,\rho\right), y\right) = \mathcal{L}_{CE}\left(p\left(z_s\right), y\right) + \beta * \mathcal{L}_{AT}\left(A_t, A_s\right) \tag{5.7}$$

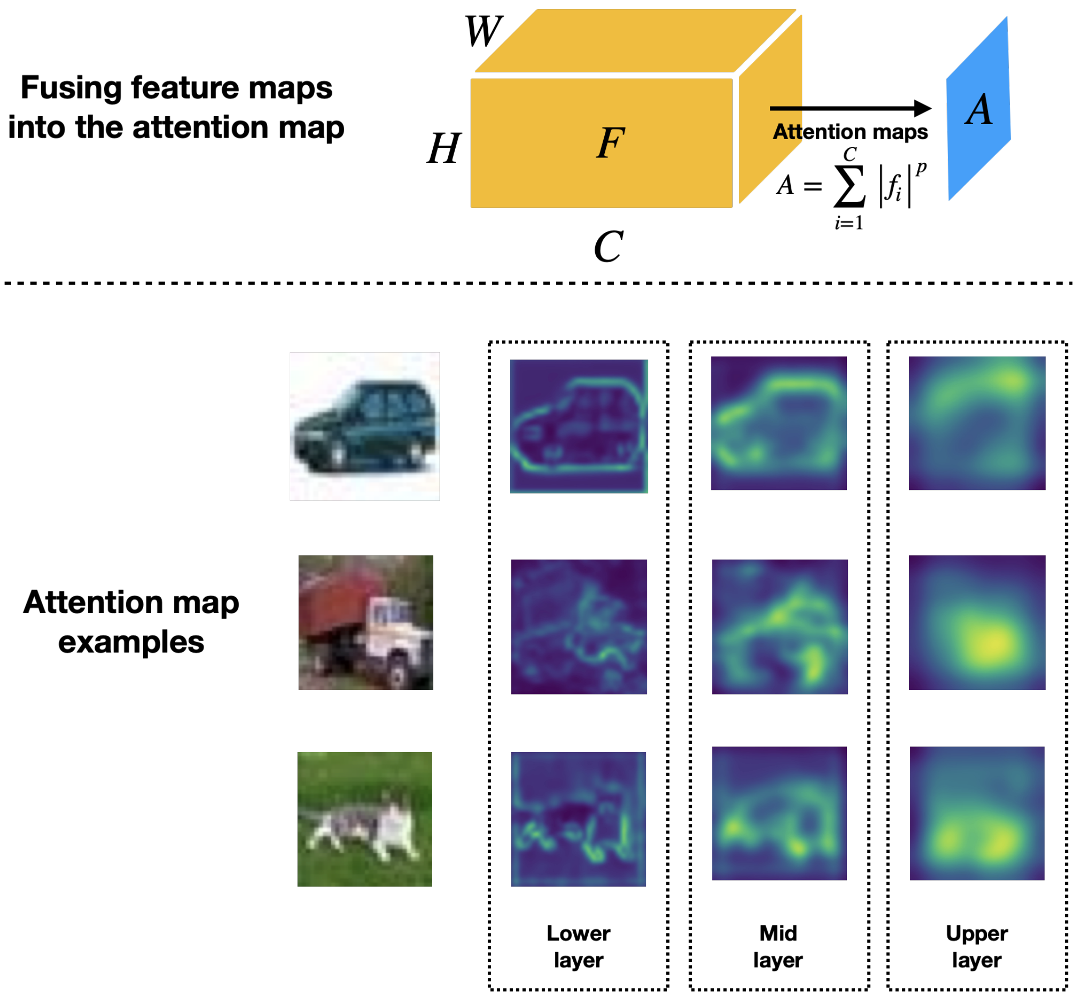


Figure 5.2: Visualising attention maps. Feature maps are extracted from a wide ResNet at the lower layer, mid layer, upper layer on samples from CIFAR-10 to form corresponding attention maps. Specifically, attention maps from lower layers highlight object outlines and basic features, whereas upper layer attention maps combine these into more abstract representations.

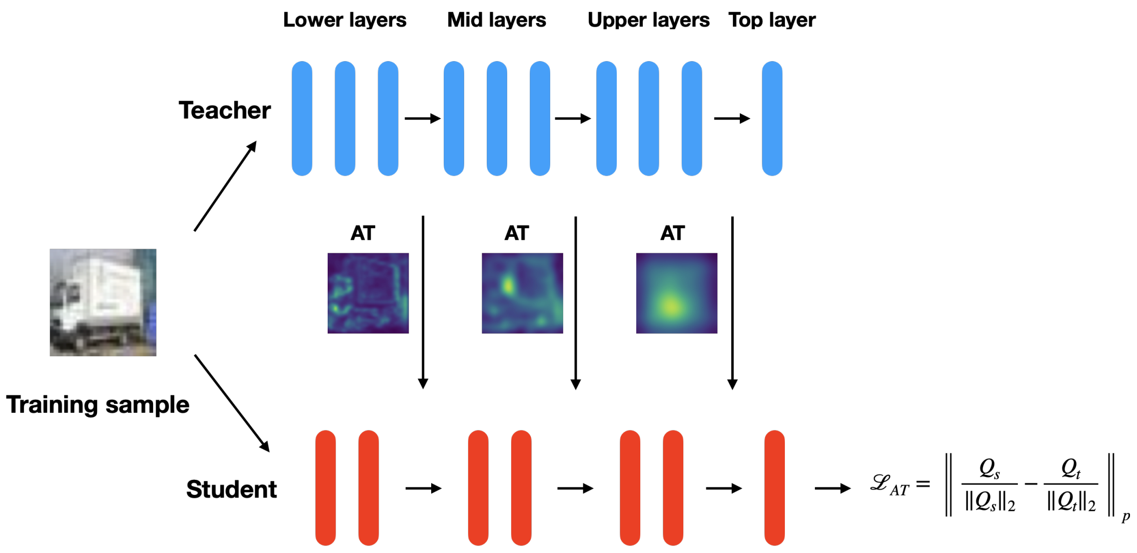

Lower layers
Mid layers
Upper layers
Top layer
Teacher
AT
AT
AT
Training sample
Student
$\mathcal{L}_{AT} = \left\| \frac{Q_s}{\|Q_s\|_2} - \frac{Q_t}{\|Q_t\|_2} \right\|_p$

Figure 5.3: S-T learning with AT. The student model is supervised with AT by the teacher model on one single training sample from CIFAR-10. The teacher model transfers its attention knowledge to the student model across three distinct levels defined in the set $\mathcal{I}$.

Supervised by attention maps from the teacher model, the student model is forced to align its attention in the latent space with that of the teacher. The original work of AT demonstrates its effectiveness in improving the performance of KD. Inspired by the success of AT in centralised learning, we hypothesise that AT can be leveraged to improve knowledge transfer in FL.

5.2.2 Refining the Global Model with Metadata Training

FL works, such as FedGKT (He et al., 2020a) and CCVR (Luo et al., 2021), have highlighted the efficacy of training the upper part of the global model using feature representations extracted from client models. On top of AT, this work is further inspired to utilise feature maps extracted on client models to enhance knowledge transfer between custom-size client models and the global model. As detailed in Section 5.1.2, this work is orthogonal to the previous works, being specifically tailored to custom-size client models. Figure 5.4 illustrates this method.

Concretely, feature maps are extracted from client models on local data after local updates. The same notations defined in Section 4.2 are reused. On client k,

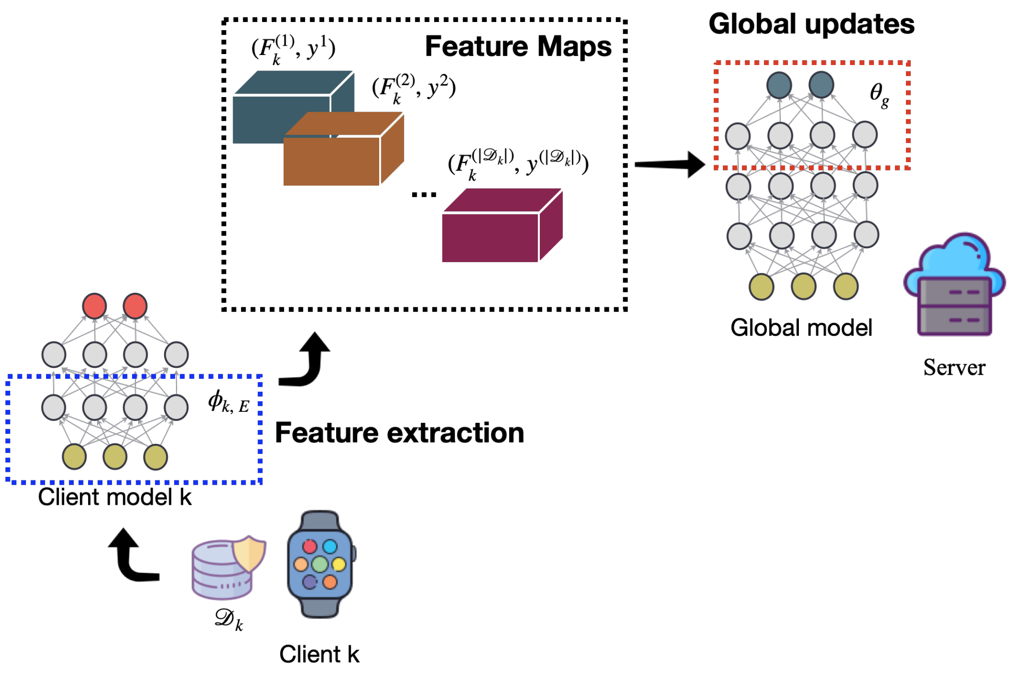


Figure 5.4: Training the global model with feature maps extracted from custom-size client models. A single client model is used for demonstration.

its local model parameterised by $w_k = \{\phi_k, \theta_k\}$, where ϕ_k is the feature extractor and θ_k is the upper part of the client model, is updated to $w_{k,\,E} = \{\phi_{k,\,E}, \theta_{k,\,E}\}$ after E local training epochs. Then, the client extracts feature maps $F_k^{(i)}$ for the client data $x_k^{(i)} \in \mathcal{D}_k,\ i = 1, \cdots, |\mathcal{D}_k|$ as follows:

$$F_k^{(i)} = f_{\phi_{k,\,E}}(\phi_{k,\,E}; x_k^{(i)}) \tag{5.8}$$

where $f_{\phi_{k,\,E}}$ is the mapping function of the updated feature extractor. Client k then forms the *metadata* with all the feature maps alongside their corresponding labels in $\mathcal{D}_k$.

$$\mathcal{D}_{k,\,meta} = \bigcup_{i=1}^{|\mathcal{D}_k|} (F_k^{(i)}, y_k^{(i)}) \tag{5.9}$$

$\mathcal{D}_{k,\,meta}$ is then uploaded to the server for supervised training the upper part of the global model, θ_g, as follows:

$$\theta_g \leftarrow \theta_g - \lambda_g \nabla_{\theta_g} \ell_g(\theta_g; D_{k,\, meta}), \tag{5.10}$$

where λ_g and ℓ_g are the learning rate and loss function for the global updates. The global model M_g is split into θ , ϕ in a way that the dimensions of the feature maps can match the input dimensions of θ. Therefore, θ can be trained with the feature maps. The feature maps are the representations of the local data encoded by the updated client model. They convey essential local knowledge learned by the client model. By training the global model with the feature maps, the global model learns local knowledge without accessing raw user data.

5.3 Federated Global Updates with Attention Transfer

This section details the algorithm of FedAT. Figure 5.5 depicts the federated learning loop in FedAT using a single client model, which applies to all other participants as well. Conventionally, FedAT includes both global updates and local updates. They are introduced separately in the following sections.

5.3.1 Local Updates of FedAT

FedAT assumes the size of the client model is customised to match its local computing capability such that all clients finish their local updates within a given budget of time. Therefore, all participants in FedAT are non-stragglers, performing E local training epochs and successfully sending their training results back to the server.

There are two operations for the custom-size client model on the edge: the conventional model training on the local data and the extraction of feature maps. Following the standard local updates defined by Equation 2.3, the client model M_k parameterised by $w_k = \{\phi_k,\, \theta_k\}$ is first updated to $w_{k,\, E} = \{\phi_{k,\, E},\, \theta_{k,\, E}\}$ after E local training epochs with Equation 5.11.

$$w_{k,\, E} \leftarrow w_k^t - \lambda_l \nabla_{w_k^t} \ell_k(w_k^t; \mathcal{D}_k), \tag{5.11}$$

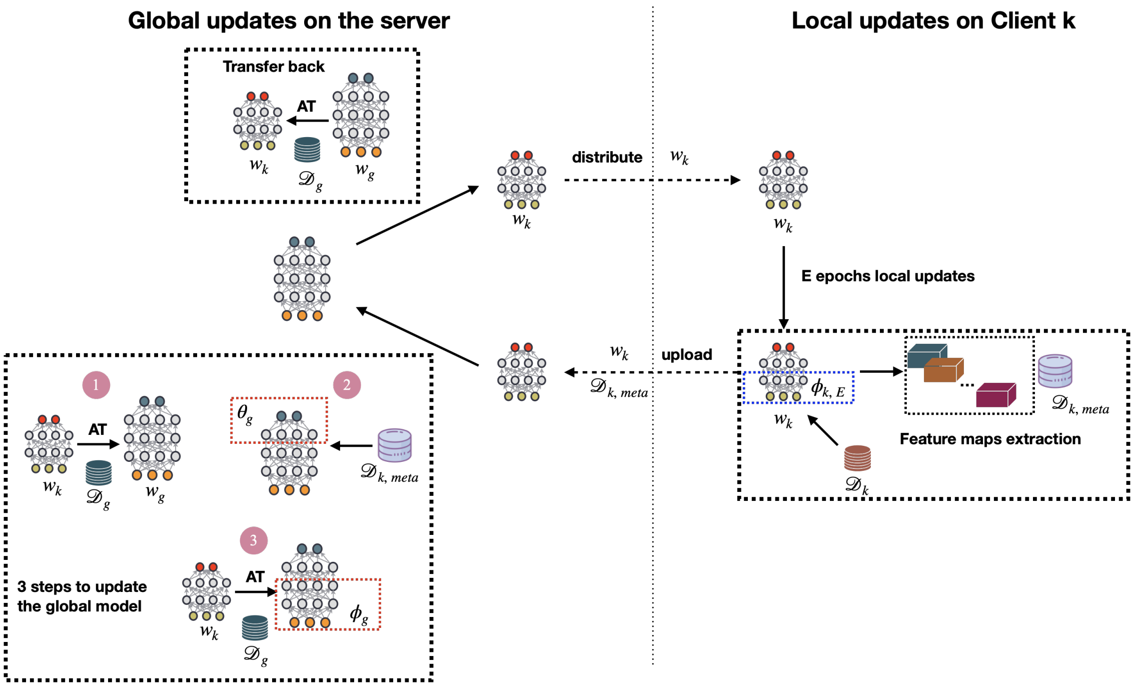


Figure 5.5: The global updates and local updates of FedAT. This graph showcases a single client scenario, which is applicable to all other participating clients.

Unlike Equation 2.3, Equation 5.11 uses $w_{k,\,E}$ instead of w_k^{t+1} because w_k^{t+1} will be obtained on the server side with attention transfer in FedAT, as depicted in the following Section 5.3.2.

To facilitate knowledge transfer to the global model, FedAT asks the updated client model to extract feature maps for the local data. As stated in Equation 5.8 in Section 5.2.2, we propose to utilise the feature extractor $\phi_{k,\,E}$ to encode the local data that are not allowed to be shared originally with the global model. The partition of $w_{k,\,E}$ into $\{\phi_{k,\,E},\, \theta_{k,\,E}\}$ can be determined by a predefined layer level on the client model. In this work, the client model is split into three groups, using notations of $l \in \{low, mid, up\}$ to indicate the three layer levels, lower layers $l = low$, mid layers $l = mid$, and upper layers $l = up$. Figure 5.6 visualises the model partition and extracted feature maps. $F_k^{(i)}$ can be extracted from one of these three layer levels.

With the completion of feature extraction, the set of the extracted feature maps is uploaded to the server together with their corresponding labels in $\mathcal{D}_k$, defined as

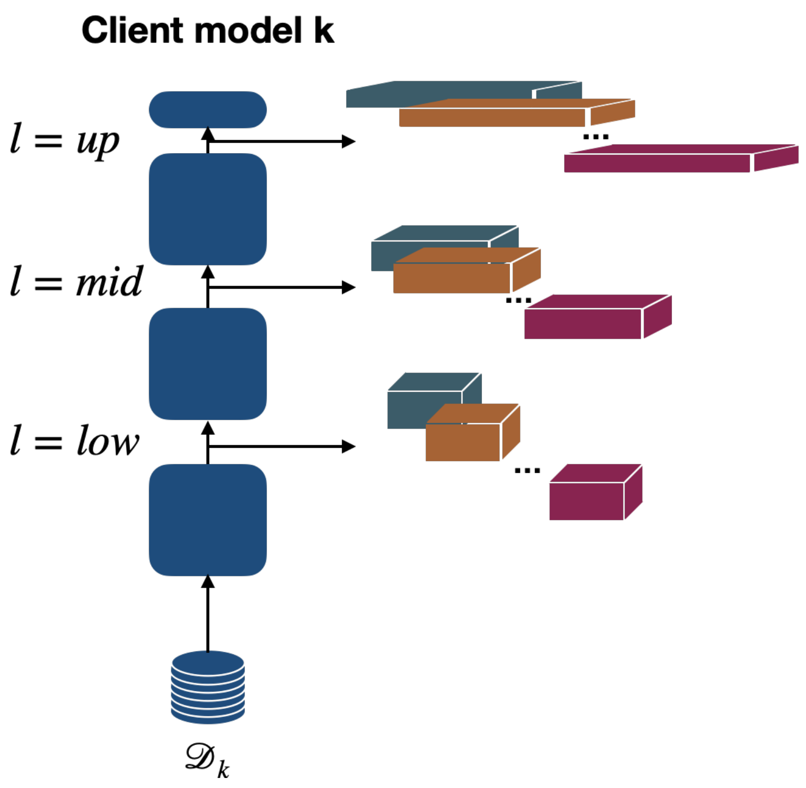


Figure 5.6: Client model split and feature extraction. The client model is split into lower, mid and upper layer levels. The dimensions of feature maps vary at distinct layer levels.

$\mathcal{D}_{k,\, meta}$ by Equation 5.9.

5.3.2 Global Updates of FedAT

FedAT trains a global model on the server for future deployments. To facilitate AT, a generic dataset $\mathcal{D}_g$, is assumed available on the server side. Same to client models, here FedAT defines the upper, mid, lower levels for the global model. Even though the sizes of the global model and client model are different, we can still ensure all models have the same input dimensions at each corresponding layer level. After the client models get updated and send their updated models and metadata back to the server, the global optimization objective is formulated by averaging K sub-optimisation objectives.

$$\arg\min_{w_g} \mathcal{L}(w_g) = \sum_{k=1}^{K} p_k L_k(w_g; w_{k,\, E},\ \mathcal{D}_g,\ \mathcal{D}_{k,\, meta}) \tag{5.12}$$

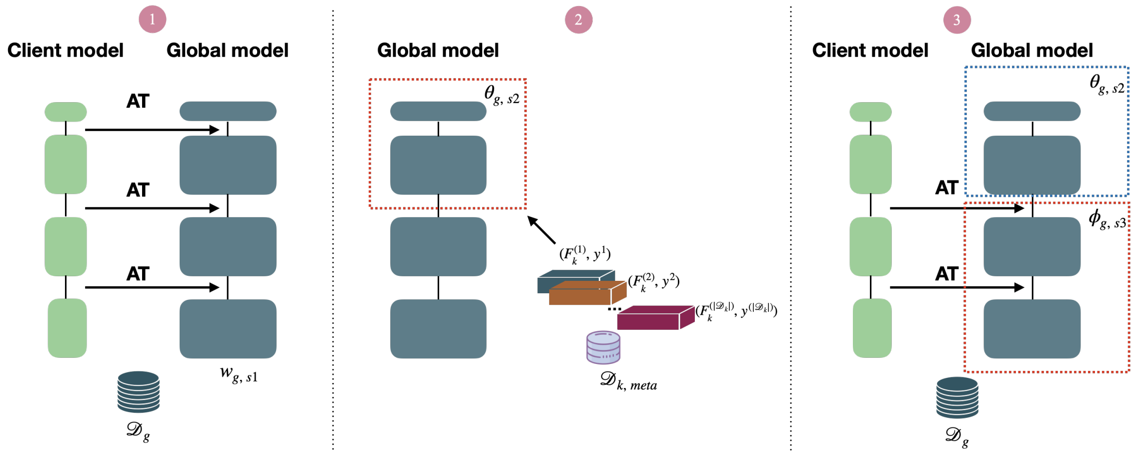


Figure 5.7: The 3-step optimisation on the server to update the global model with a custom-size client model and its metadata.

where p_k uses the same definition described in Equation 2.1.

FedAT proposes a unique 3 steps to achieve the sub-optimisation objective, transferring locally learned knowledge from client models to the global model on the server. Each sub loss function $L_k(w_g; w_{k,\,E},\ \mathcal{D}_g,\ \mathcal{D}_{k,\,meta})$ optimises a copy of w_g by first training the copy on $\mathcal{D}_g$ with the attention transfer from the updated client model $w_{k,\,E}$, and then training its upper layers on metadata, $\mathcal{D}_{k,\,meta}$, collected from the corresponding client. Finally, the copy refines its lower layers on $\mathcal{D}_g$ with attention transfer from $w_{k,\,E}$ again. Figure 5.7 shows the 3-step optimisation in the global updates.

Step 1. The global update starts with AT from $w_{k,\,E}$ to w_g on the generic data $\mathcal{D}_g$ data by minimising the loss as follows.

$$w_{g,s1} = \operatorname*{argmin}_{w_g} \left(\sum_{(x^{(i)},y^{(i)})\in\mathcal{D}_g} \ell_{ce}\left(f_{w_g}(x^{(i)}),\ y^{(i)}\right) + \beta\mathcal{L}_{AT}(Q_{w_g},\ Q_{w_{k,E}}) \right) \tag{5.13}$$

$$\mathcal{L}_{AT}(Q_{w_g},\ Q_{w_{k,E}}) = \sum_{i\in\mathcal{D}_g} \sum_{l\in\mathcal{I}} \left\| \frac{Q_{w_g}^{l,(i)}}{\|Q_{w_g}^{l,(i)}\|_2} - \frac{Q_{w_{k,E}}^{l,(i)}}{\|Q_{w_{k,E}}^{l,(i)}\|_2} \right\|_2 \tag{5.14}$$

Recall the attention maps described in Section 5.2.1, $Q_{w_g}^{l,(i)}$ and $Q_{w_{k,E}}^{l,(i)}$ are the vectorisied attention maps formed at the j level layer of global model and client model on the i-th instance of $\mathcal{D}_g$. FedAT sets p defined in Equation 5.6 to 2.

Step 2. FedAT exercises the upper layers of M_g using metadata from clients. As stated earlier in Section 5.2.2, the global model is split into θ_g, ϕ_g in a way to match the upper part θ_g with the dimensions of metadata. To this end, the global model updated in step 1, $w_{g,s1}$, is split into $\{\theta_{g,s1}, \phi_{g,s1}\}$. The feature maps from $\mathcal{D}_{k,\, meta}$ are propagated through the upper part of the global model, $\theta_{g,s1}$, starting from the predetermined level. We minimise the loss presented in Equation 5.10 to formulate the following optimization problem described by Equation 5.15.

$$\theta_{g,s2} = \underset{\theta_{g,s1}}{\operatorname{argmin}} \sum_{(F_k^{(i)}, y_k^{(i)}) \in \mathcal{D}_{k,\, meta}} \ell_{ce}(f_{\theta_{g,s1}}(\theta_{g,s1}\ F_k^{(i)}),\ y_k^{(i)}) \tag{5.15}$$

The role of this step is to specialise the features in the upper part of the network, including for the under-represented classes due to non-IID data distribution, but captured in $\mathcal{D}_{k,\, meta}$ as higher level feature maps.

Step 3. The global model updated at the end of step 1 and step 2 is $w_{g,s2} = \{\theta_{g,s2}, \phi_{g,s1}\}$. Finally, its feature extractor $\phi_{g,s1}$, is refined with AT by freezing the upper part $\theta_{g,s2}$ previously trained at step 2. As $\theta_{g,s2}$ is frozen, we do not need to transfer the attention from levels in the upper part. Hence, only attention maps from a subset of $\mathcal{I}$, denoted as $\mathcal{I}_{sub}$ are compared between the global model and the client model at step 3. Except for the fixed upper part $\theta_{g,s2}$ and attention transfer inside the trainable feature extractor $\phi_{g,s1}$, step 3 is essentially the same as step 1.

$$\phi_{g,s3} = \underset{\phi_{g,s1}}{\operatorname{argmin}} \left(\sum_{(x^{(i)}, y^{(i)}) \in \mathcal{D}_g} \ell_{ce}\left(f_{w_{g,s2}}(x^{(i)}),\ y^{(i)}\right) + \beta \mathcal{L}_{AT}(Q_{w_{g,s2}},\ Q_{w_{k,E}}) \right) \tag{5.16}$$

$$\mathcal{L}_{AT}(Q_{w_g},\ Q_{w_{k,E}}) = \sum_{i \in \mathcal{D}_g} \sum_{l \in \mathcal{I}_{sub}} \left\| \frac{Q^{l,(i)}_{w_{g,s2}}}{\|Q^{l,(i)}_{w_{g,s2}}\|_2} - \frac{Q^{l,(i)}_{w_{k,E}}}{\|Q^{l,(i)}_{w_{k,E}}\|_2} \right\|_2 \tag{5.17}$$

FedAT only updates the feature extractor at step 3, therefore the global model

$w_{g,s3}$ updated at the end of step 3 is composed of $\{\theta_{g,s2}, \phi_{g,s3}\}$. The refinement of the feature extractor at step 3 aims to adapt it to the upper part updated at step 2. As $\theta_{g,s2}$ is the training results on the metadata, which is substantially different from the generic data utilised for updating the feature extractor $\phi_{g,s1}$ at step 1, it is critical to align $\phi_{g,s1}$ and $\theta_{g,s2}$ on a unified dataset to ensure the global model be trained correctly. By fixing $\theta_{g,s2}$, the training at step 3 can force $\phi_{g,s1}$ to output features further forwarded by $\theta_{g,s2}$ to consistently minimise the loss on the generic data. If the two parts of the global model are not aligned, the training loss would be extremely high.

After each sub-optimisation is finished by completing all 3 aforementioned steps, the server obtains a total of K updated global models. Finally, FedAT forms a global model that is distributed at $t+1$-th communication round using the naive model fusion, same to Equation 2.4.

$$w_g^{t+1} \leftarrow \sum_{k=1}^{K} p_k w_{g,s3}^{(k)} \tag{5.18}$$

where $w_{g,s3}^{(k)}$ is the k-th updated global model.

Upon obtaining the global model aggregated knowledge from custom-size client models, FedAT updates client models on the server by transferring its knowledge back to them. This is the analogy to the distribution of the updated global model w^{t+1} to the client side introduced in Section 2.1.1. Again, FedAT uses AT to achieve this, as illustrated by Figure 5.8.

$$w_k^{t+1} = \underset{w_{k,\,E}}{\operatorname{argmin}} \left(\sum_{(x^{(i)}, y^{(i)}) \in \mathcal{D}_g} \ell_{ce}\left(f_{w_{k,\,E}}(x^{(i)}),\ y^{(i)}\right) + \beta \mathcal{L}_{AT}(Q_{w_{k,E}},\ Q_{w_g}) \right) \tag{5.19}$$

$$\mathcal{L}_{AT}(Q_{w_{k,E}},\ Q_{w_g}) = \sum_{i \in \mathcal{D}_g} \sum_{l \in \mathcal{I}} \left\| \frac{Q_{w_g}^{l,(i)}}{\|Q_{w_g}^{l,(i)}\|_2} - \frac{Q_{w_{k,E}}^{l,(i)}}{\|Q_{w_{k,E}}^{l,(i)}\|_2} \right\|_2 \tag{5.20}$$

Finally, client models w_k^{t+1} for $k \in \{1, \cdots, K\}$ are sent back to the client side to start round $t+1$. The FedAT algorithm is summarised in Algorithm 3

Algorithm 3 FedAT: **Fed**erated Global Updates with the **A**ttention **T**ransfer.

```
1: Input: total $T$ communication rounds, a generic dataset on the server $\mathcal{D}_g$, $E$ epochs of local updates, $w_g = \{\theta_g,\ \phi_g\}$, $w_k = \{\theta_k,\ \phi_k\}$ for all $k \in [1, K]$, the predefined layer level $l$ to extract feature maps.
2: for $t = 1, \ldots, T$ do
3:     $K$ participant clients download client model $w_k = \{\theta_k,\ \phi_k\}$ from the server.
4:     Clients:
5:     for Client $k \in [1, K]$ do
6:         $w_{k,\,E} \leftarrow$ Local updates $(w_k^t; \mathcal{D}_k, E)$ according to Equation 5.11.
7:         $\mathcal{D}_{k,\,meta} \leftarrow$ Metadata extraction $(; \phi_{k,\,E}, \mathcal{D}_k)$ according to Equation 5.8 and Equation 5.9.
8:         Client $k$ uploads $w_{k,\,E}$ and $\mathcal{D}_{k,\,meta}$ to the server.
9:     end for
10:    Server:
11:    Server makes $K$ copies of $w_g^t$
12:    for $k \in [1, K]$ on each of the copied $w_g^t$ do
13:        $w_{g,\,s1}^t \leftarrow$ AT from client to global $\big(w_g^t; w_{k,\,E}, \mathcal{D}_g\big)$ according to Equation 5.13 and Equation 5.14.
14:        $\theta_{g,\,s2}^t \leftarrow$ Metadata training $\big(\theta_{g,\,s1}^t; \mathcal{D}_{k,\,meta}\big)$ according to Equation 5.15.
15:        $\phi_{g,s3}^t \leftarrow$ AT from client to global $\big(\phi_{g,s1}^t; \theta_{g,\,s2}^t, w_{k,\,E}, \mathcal{D}_g\big)$ according to Equation 5.16 and Equation 5.17.
16:        Construct the updated global model copy $w_{g,\,s3}^{(k)} = \{\theta_{g,\,s2}^t,\ \phi_{g,\,s3}^t\}$.
17:    end for
18:    Aggregation of the updated global model to form $w_g^{t+1}$ according to Equation 5.18.
19:    for $k \in [1, K]$ on client model $w_{k,\,E}$ do
20:        $w_k^{t+1} \leftarrow$ AT from global to client $\big(w_{k,\,E}; w_g^{t+1}, \mathcal{D}_g\big)$ according to Equation 5.19 and Equation 5.20.
21:    end for
22: end for
23: Return Global model $w_g^{T+1}$, client models $w_k^{T+1}$ for $k \in [1, K]$.
```

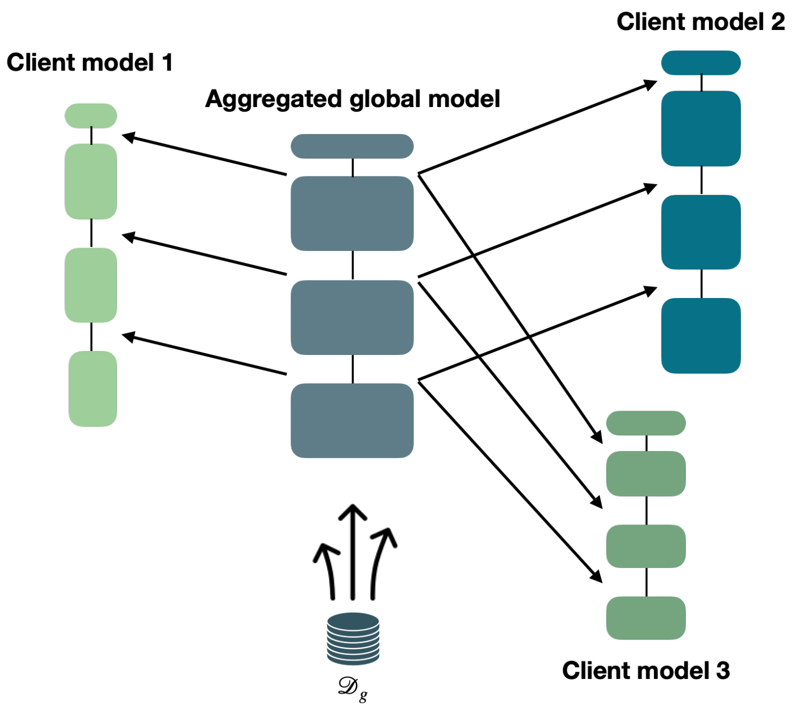


Figure 5.8: AT from the aggregated global model to individual client models.

5.4 FedAT Experiments

Specific experimental setup for FedAT. The evaluation of FedAT uses a WRN model with a depth of 40 and width of 1 (WRN-40-1) as the global model, whereas each client model has individual structure with a smaller size than the global model. FedAT is evaluated on CIFAR-10, with each class referred to based on its label (a number between 0 to 9). A simple FL scenario where there is only one client is first studied. This case is used to demonstrate the effectiveness of AT and metadata training for knowledge transfer. Then FedAT is evaluated in a realistic FL scenario where multiple clients participate.

5.4.1 Single Client Scenario

FedAT proposes the unique 3-step global updates described in Section 5.3.2 for transferring the knowledge locally learned by the client model to the global model. To evaluate its efficacy, ablation studies are conducted to study the effects and limitations of the 3 steps in a simply single client scenario. The generic data $\mathcal{D}_g$

held by the server and the client data $\mathcal{D}_1$ are partitioned with an IID scenario and a non-IID scenario in the ablation studies to simulate applications in the wild.

The client model involves two WRNs with reduced depths, WRN-16-1 and WRN-10-1, compared to the global model, WRN-40-1. It is observed that they produce very close results, so only the results of using WRN-16-1 as the client model is reported here. Attention maps are extracted from 3 paired layer levels with the same output feature dimensions as illustrated earlier in Figure 5.7. A SGD optimiser with a learning rate of 0.1 is applied to all the training steps in global updates and local updates. Borrowed from the original setup in Zagoruyko and Komodakis (2016a), the value of β that weighs the AT loss is set to 1,000. The training epochs for the attention transfer, local training, and metadata learning are tuned to 2, 1, 2 respectively.

The training set of CIFAR-10 is split into generic data $\mathcal{D}_g$ and the client data $\mathcal{D}_1$. Notably, by assigning more data points or classes to $\mathcal{D}_1$ than $\mathcal{D}_g$, the client data is used to train the client model to achieve advantageous performance over the global model trained on $\mathcal{D}_g$. Therefore, the knowledge gap between the global model and client model enables us to study whether we can narrow this gap using attention transfer and metadata training.

In the IID data scenario, $\mathcal{D}_g$ consists of 20% of training samples randomly selected from the training set and $\mathcal{D}_1$ holding the other 80% of the training set disjointly. The random selection is uniform, making both $\mathcal{D}_g$ and $\mathcal{D}_1$ have the IID distribution over. To simulate the non-IID data, the training instances of classes 0 to 8 (45,000 images) are allocated to the generic data $\mathcal{D}_g$, and the local data $\mathcal{D}_1$ holds training images of class 9 on top of samples from the other classes (0..8). This is intended to expose the performance of our FL training method in data heterogeneity, where some classes are unbalanced in representation across server and clients (non-IID data). Training only on $\mathcal{D}_g$, the test accuracy of the global model is 84.53%, because it is never exposed to images of class 9.

Table 5.1 shows the baseline performance of the global model and the client model, which is set up by training the global model with just the generic data $\mathcal{D}_g$ and training the client model on the client data $\mathcal{D}_1$. As the client data has more samples

Scenario	Model	Model config.	Training data	Test Acc (%).
IID	M_g	WRN-40-1	20% training set	85.30
	M_1	WRN-16-1	80% training set	88.18 $\uparrow$
non-IID	M_g	WRN-40-1	samples from classes 0-8	84.53
	M_1	WRN-16-1	samples from classes 0-9	91.61 $\uparrow$

Table 5.1: Baseline performance in the single client scenario on CIFAR-10. There are performance gaps between the global model trained on $\mathcal{D}g$ and the client model trained on $\mathcal{D}1$, as $\mathcal{D}1$ has more training samples or classes than $\mathcal{D}g$.

or classes than the generic data, it is obvious that the client model has advantageous performance. The client model performance is defined in Table 5.1 as the *upper baseline*, opposed to the global model performance as the *lower baseline*. The 3-step global updates in FedAT aim to close their performance gap by transferring the client knowledge to the global model.

Ablation Studies for Step 1

Table 5.1 shows that the client model, despite being smaller, outperforms the global model because it is trained on $\mathcal{D}_1$, which has more samples or classes than $\mathcal{D}_g$, where the global model is trained. To this end, we approach a question: "How effective is AT at step 1 for transferring knowledge learned by a small-size model to a model with a larger size."

Table 5.2 reports the global model performance after step 1, trained with AT from the client model. It shows AT at step 1 improves the global model performance, by up to 1.5% in the IID scenario due to the knowledge gap is created by the differences of the training data size ($\mathcal{D}_1$ 4$\times$ larger than $\mathcal{D}_g$). However, in the non-IID scenario, the improvement is very minimal as the knowledge gap exists in the missing class in $\mathcal{D}_g$, in our case is class 9. With attention transfer on $\mathcal{D}_g$, the client model can only supervise the global model on training samples in $\mathcal{D}_g$. The global model is unable to increase its performance significantly without learning knowledge

Scenario	Training strategy	Test Acc (%).
IID	trained on $\mathcal{D}_g$	85.30
	trained with AT on $\mathcal{D}_g$	86.82 ↑
non-IID	trained on $\mathcal{D}_g$	84.53
	trained with AT on $\mathcal{D}_g$	84.74 ↑

Table 5.2: AT from the client model improves the global model performance trained on $\mathcal{D}_g$ at step 1.

from class 9.

Ablation Studies for Step 2 and Step 3

This section studies the impact of step 2 and step 3 in the global updates on the global model performance. It addresses the questions: “Which layer’s metadata is more effective for knowledge transfer?” and “How important is the alignment of θ_g and ϕ_g with AT from the client model for the global updates?”.

Figure 5.9 compares the global model performance after the global updates by varying layer levels for extracting metadata, $\mathcal{D}_{1,\,meta}$ and whether performing the alignment of θ_g and ϕ_g. The bar charts to the left in Figure 5.9 describe the test performance of the global model updated on the IID generic data. Compared to the global model performance shown in the Table 5.1, 85.30%, it is observed that using metadata training alone is ineffective to boost the global performance. Whereas applying the model alignment with AT at step 3 increases the global performance up to 86.59% (using metadata extracted from the upper layer level), outperforming the baseline by 1.3% and closing the performance gap by almost 50% (from 2.9% to 1.6%). This observation generally holds true for the non-IID data partition scenario, where the global model performance is improved by over 6%. However, the difference is, with the IID data partition, metadata extracted from 3 layer levels are all effective to boost the global performance when the model alignment at step 3 is adopted.

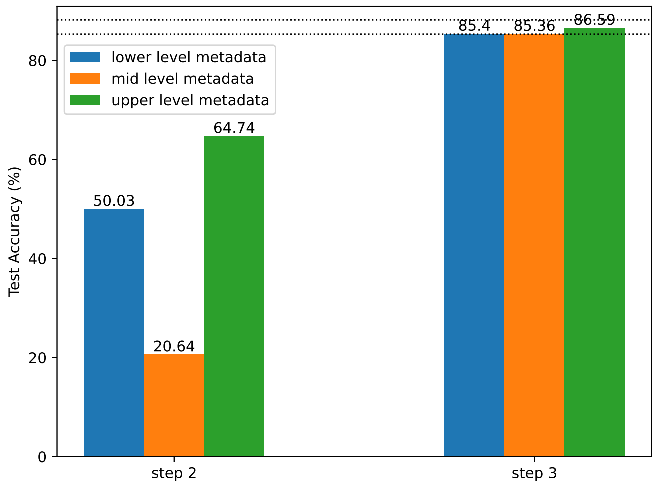
In the non-IID data scenario, only metadata extracted from the lower layer level effectively improves the global model performance from the baseline, by 6%. This observation suggests that the metadata at lower layers are more informative for transferring knowledge between models trained on non-IID data. Finally, by comparing the improved global performance using AT and metadata training to the baseline performance achieved by the client model. It is apparent that the performance gap between the global model and the client model is narrowed owing to the proposed 3-step knowledge transfer. However, the student (the global model in this case) still cannot achieve an equal or higher performance than its teacher (the client model).

Aligning the Global Model Updates with AT

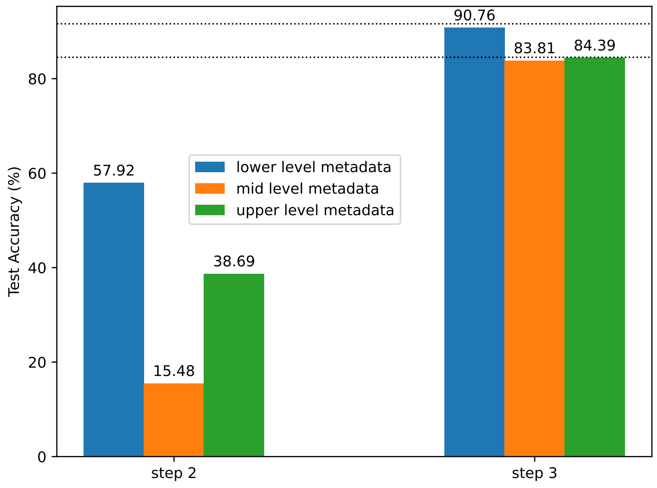
Figure 5.9 shows that metadata training negatively impacts global model performance, decreasing the baseline performance after metadata training. This section provides insights into this observation by exploring the effects of metadata training in the global updates under the non-IID scenario.

In the non-IID scenario predetermined in our setup, $\mathcal{D}_{1,\,meta}$ is the only knowledge source containing knowledge about class 9 available to the global model. This knowledge is in the form of feature maps from the output of the feature extractor $\phi_{1,\,E}$ updated locally on the client data. Metadata training aims to teach the global model how to recognise the missing class not available in the server data $\mathcal{D}_g$, class 9 in this case.

To understand metadata training, Table 5.3 reports the global model performance on all 10 classes and specifically class 9 in CIFAR-10 at the end of step 1 and step 2. We use feature maps extracted from the upper layer level of the client model to form the metadata for training $\theta_{g,\,s1}$. The results demonstrate that metadata training at step 2 significantly degrades the global performance with increasing training epochs. The table examines two distinct cases. In the first, the metadata only contains feature maps from samples in class 9. It is observed the global model gradually learns to recognise class 9 but forgets how to generalise on other classes, showing the catastrophic forgetting effect typically found in the model trained on multiple tasks. The network specialises on the features characteristic to the current



(a) IID scenario



(b) non-IID scenario

Figure 5.9: The global performance at step 2 and step 3 under both IID data scenario and non-IID data scenario is compared, using metadata formed at different layer levels. Metadata training at step 2 does not effectively boost the global model performance. After step 3, FedAT improves the global model performance using AT. In the non-IID scenario, metadata extracted from the lower layer level most significantly boosts the global model performance.

task while forgetting representations crucial for the previous task. In the second case, metadata includes feature maps sampled from all classes. It is evident the generalisation on all classes quickly diminishes, consistent with observations in Figure 5.9. We suspect the performance decline stems from the misalignment between ϕ_g and θ_g due to their updates on two distinct dataset $\mathcal{D}_g$ and $\mathcal{D}_{1,\,meta}$ at step 1 and step 2.

In summary, Table 5.3 demonstrates that while the global model effectively learns from metadata, changing training data is detrimental to its overall performance.

The AT at step 3 can mitigate the negative impacts brought by changing training data. Figure 5.9 empirically justified the benefits of AT step 3, as it restores global model performance by aligning ϕ_g and θ_g on the same dataset $\mathcal{D}_g$. Furthermore, the global model achieves better performance than the baseline, demonstrating the client knowledge is successfully transferred to the global model with AT and metadata training.

The Regularisation Effect of AT at Step 3

Another critical question to approach in ablation studies is "What is the purpose of using AT in step 3?" or "Can we just align ϕ_g and θ_g on $\mathcal{D}_g$ without AT?". AT adopted at step 3 aims to enhance the representation learning on the unseen data in $\mathcal{D}_g$.

After step 2, the upper part θ_g learns to generalise on new classes with feature maps. While aligning the feature extractor ϕ_g with θ_g on $\mathcal{D}_g$ at step 3, AT from the client model can act as a regularisation approach to avoid ϕ_g overfitting to $\mathcal{D}_g$. Table 5.4 reports the findings from ablation studies on using AT at step 3 in global updates. Results show that the global model performance on all classes is nearly identical between with AT and without AT, only 0.04% differences. However, the performance on the missing class in $\mathcal{D}_g$ has a noticeable gap between using AT and not using AT. When AT is applied at step 3, the performance on the missing class is improved by over 2%, supporting the hypothesis that AT can regularise the global model to learn representation for the missing class without overfitting to

Training step	Epochs	Test Acc. (%) on all classes	Test Acc. (%) on class 9
Step 1	2	84.74	0
Step 2 with $\mathcal{D}_{1,\,meta}$ sampled from class 9	1	84.47 ↓	0
	2	84.14 ↓	0.1 ↑
	3	77.01 ↓	19.8 ↑
	4	36.43 ↓	86.7 ↑
Step 2 with $\mathcal{D}_{1,\,meta}$ sampled from all classes	10	70.72 ↓	0
	20	65.91 ↓	0
	30	45.27 ↓	0
	40	25.70 ↓	0.6 ↑

Table 5.3: The global model performance at step 1 and step 2 in the global updates, with step 1 serving as the baseline. $\mathcal{D}_{1,\,meta}$ is derived from two sources: exclusively from class 9, and from all ten classes. In both cases, metadata training decreases the performance. The first case demonstrates the catastrophic forgetting, specifically lowering accuracy for other classes except for class 9. In the second case, we hypothesise the performance drop is due to the misalignment between ϕ_g and θ_g. Figure 5.9 shows that the performance can be restored when ϕ_g and θ_g are aligned through joint training on $\mathcal{D}_g$.

Training step	Test Acc. (%) on all classes	Test Acc. (%) on class 9
Step 1	84.74	0
Step 2	57.92	80.0
Step 3 w/ AT	90.76	92.20
Step 3 w/o AT	90.80	89.80 ↓

Table 5.4: Comparison of global model performance on all classes and specifically on class 9 in $\mathcal{D}_g$ at step 3, with or without AT, using metadata from the lower layer of the client model. AT prevents overfitting to $\mathcal{D}_g$. Applying AT at step 3 improves the performance on the missing class without compromising overall performance.

other classes.

Training the Global Model with Reduced Metadata

This ablation study explores the minimal metadata required for effective knowledge transfer from the client model to the global model, without compromising the global performance. This study is substantial for reducing the communication overhead arising from transmitting metadata in the network.

These experiments use feature maps, $\mathcal{D}_{1,\,meta}$, extracted at the lower layer level of the client model, which Figure 5.9 shows improves the global model performance the most. The client uploads varying proportions of metadata, ranging from 100% to 1% of $\mathcal{D}_{1,\,meta}$ by randomly selecting feature maps from the entire $\mathcal{D}_{1,\,meta}$, to train the global model on the server at step 2.

Figure 5.10 illustrates the global model performance at each step in the global updates. It is observed that the best global performance, over 90% on the test set, is achieved when the entire set of $\mathcal{D}_{1,\,meta}$ is transferred. Notably, this performance remains robust until only 10% of $\mathcal{D}_{1,\,meta}$ is used, accomplishing 89.65%. However, reducing the uploaded $\mathcal{D}_{1,;meta}$ to 1% significantly decreases the global model performance to 73.42%.

Interestingly, in terms of the global performance at step 2, there is an initial

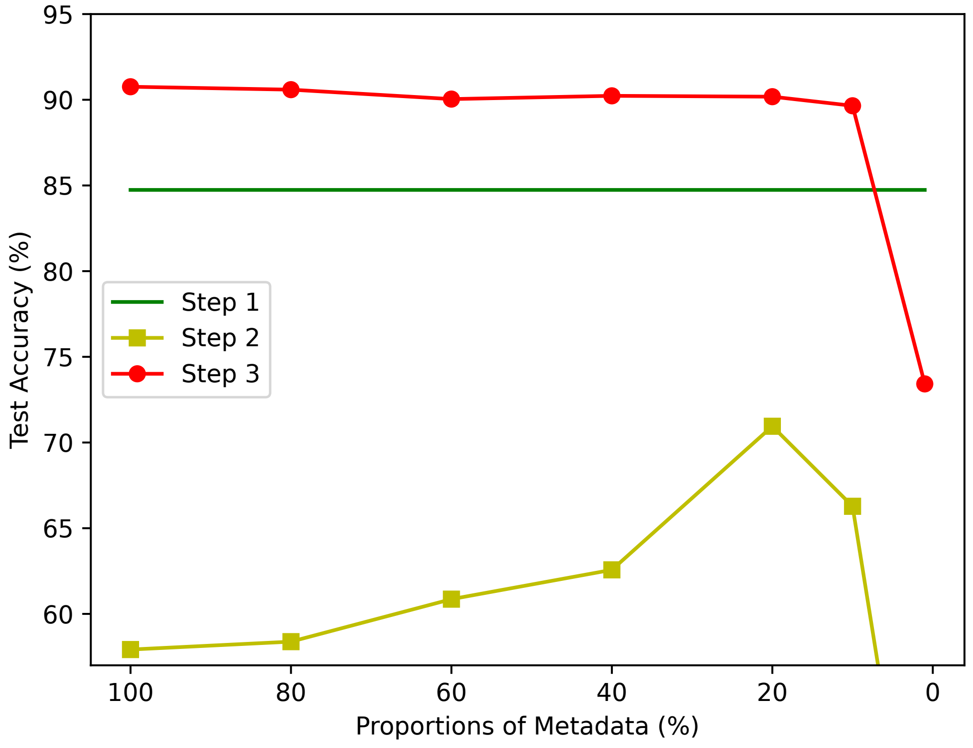


Figure 5.10: Global model performance across the 3 steps of global updates with varying metadata proportions. The green line represents step 1 without metadata. Significant performance impact occurs only when metadata used at step 2 is reduced to 1%. Removing 90% of metadata leads to only a 1% performance drop, crucial for reducing communication overhead. Consistent performance gaps among steps align with previous observations.

performance increase with less metadata transferred. This observation aligns with our earlier understanding that metadata training alone without model alignment at step 3 degrades performance. Introducing step 3 clearly improves global performance beyond the green line, which is the global performance at step 1. Since the metadata is not associated with the AT in step 1, the green line serves as a baseline to measure performance boosts introduced by metadata training at step 2 and AT at step 3.

Conclusions of ablation studies in the single client scenario. In the single client scenario, ablation studies are conducted on the global updates in FedAT, notably the 3-step knowledge transfer, providing insights and justification on each step. It is observed that both AT and metadata learning are essential for transferring knowledge learned by a small-size model to a large-size model. AT scales the weights of the larger model at the first step and fine-tunes the feature extractor with regularisation at the third step, while metadata training at step 2 adjusts the features of the upper layers of the network where representations from the missing are otherwise absent. AT and metadata training collectively enhance the generalisation of the global model in FedAT. Furthermore, it is found that using just 10% of the entire metadata still achieves comparable knowledge transfer to the global model, reliving the communication overhead introduced by uploading the metadata in the network.

5.4.2 Multiple Clients Scenario

This section experiments FedAT in a realistic scenario by using multiple clients with non-IID client data.

Experimental setup for multiple clients. The client number for the multiple clients FedAT is set to 20, with each client storing non-IID local data from a fraction of CIFAR-10 training set partitioned by $Diri(\alpha)$. Similar to the single client scenario, the generic data used in the multiple client scenario is assumed not to include all CIFAR-10 classes. Notably, the server only collects a small amount of generic data to facilitate AT in practice. $\mathcal{D}_g$ is confined to 2,500 CIFAR-10 training images randomly drawn from 7 classes (1, 2, 3, 4, 6, 7, 9). The metadata consists of feature maps extracted from the lower level layer of the client model. All experiments are conducted over 300 FL rounds. SGD optimiser is adopted again for this scenario. Learning rates are tuned to 0.1, 0.01, 0.1 for AT, local updates, metadata learning respectively, for optimal performance. Table 5.6 reports the validation performance on CIFAR-10 from a grid search over learning rates for AT and metadata learning, ranging by an order of magnitude from 0.01 to 1.0. Extremely small or large learning rates significantly degrade the overall performance of FedAT. Training epochs are set to 2, 5, 2 for AT, local updates, metadata learning respectively. The AT loss coefficient β is set to 1,000. A WRN-40-1 is used as the global model on the server.

Method	Participant ID	Depth	Width	#Para
FedAvg	Client 0-19	40	1	567k
	Server	40	1	576k
FedAT	Client 0-5	10	1	79k
	Client 6-9	16	1	177k
	Client 10-13	22	1	274k
	Client 14-17	28	1	372k
	Client 18, 19	34	1	469k
	Server	40	1	567k

Table 5.5: The configuration of custom-size WRN models over 20 clients in FedAT, with varying depths. FedAvg baseline assumes distributing a homogeneous model across the clients. FedAT assumes custom-size models to match local computational power.

Learning Rate	0.01	0.1	1.0
Val. Acc. (%)	24.76	63.58	23.68

Table 5.6: The validation performance on CIFAR-10 by varying learning rates for the attention transfer and the metadata learning in FedAT.

The 20 custom-size client models using WRNs with varying depths are described in Table 5.5.

Straggler simulation. The key advantage of FedAT over homogeneous model solutions, such as FedAvg, is that all clients are assumed as non-stragglers to fully perform local updates and successfully upload their training results. In contrast, FedAvg assumes a proportion of the clients are stragglers being dropped out by the server. Specifically, we assume the stragglers in FedAvg are constant, simulating a complete exclusion of some clients that cannot meet the computational requirements in any communication round due to the distribution of a homogeneous model in FedAvg. The same 8 clients, 12 clients, 16 clients, equivalent to 40%, 60%, 80% of all the clients, are assigned as stragglers, never finishing their local updates before the deadline. As a result, their unique local knowledge learned by the models are never utilised by the global updates in FedAvg. For FedAT, all 20 clients send their updated client models and corresponding metadata back to the server in every communication round. In addition to the realistic straggler simulation, we also include an ideal baseline, where all the clients are able to complete their local updates in FedAvg.

Experiment results. Table 5.7 compares FedAT and FedAvg under $Diri(0.1)$ and $Diri(0.5)$, reporting their best test performance achieved by the global model on CIFAR-10. For the weaker data heterogeneity level with $Diri(0.5)$, the hypothetical FedAvg achieves by far the top performance with 82.36%. Subsequently, in the realistic scenarios with 40%, 60%, 80% of clients being stragglers, FedAvg decreases its performance to 81.60%, 76.09%, 65.86% respectively. FedAT can outperform FedAvg with 60% stragglers, achieving more than 10% improvement over the FedAvg with 80% stragglers and just 6% below the ideal FedAvg. In terms of the stronger data heterogeneity with $Diri(0.1)$, it is also evident that FedAvg decreases its performance significantly as more clients become stragglers, reducing from 68.98% with no stragglers to 49.13% with 80% stragglers. The deteriorated performance of FedAvg gives FedAT a good gap to fill. It is observed that the performance gap between the hypothetical FedAvg and FedAT is marginally 0.24%, indicating FedAT outperforms realistic FedAvg baselines with stragglers by up to 19%. The experiment results show that FedAT with custom-size client models is able to improve the performance of the global model by having all the clients contribute to the learning, in contrast to a large number of clients being dropped out by the server due to system constraints in standard FedAvg.

Generalisation on the missing classes. Similar to Table 5.3 and Table 5.4 in the single client scenario, we evaluate knowledge transfer from client models and the global model by assessing the generalisation of the global model to missing classes in the generic data. Table 5.8 reports the test performance of the global

Scenario	Method	Straggler Ratio (%)	Test Acc. (%)	
			$Diri(0.5)$	$Diri(0.1)$
Hypothetical	FedAvg	0	82.36	68.98
Realistic	FedAvg	40	81.60	65.89
	FedAvg	60	76.09	52.99
	FedAvg	80	65.86	49.13
Realistic	**FedAT**	0	**76.34**	**68.75**

Table 5.7: Comparison of FedAT and FedAvg with varying straggler ratios. FedAT closes the performance gap between realistic scenarios and the hypothetical scenario, particularly under strong data heterogeneity with $Diri(0.1)$. The hypothetical case of FedAvg with no stragglers is used as the ideal performance to reach. The other cases with stragglers indicate more realistic approaches where stragglers are discarded. All the clients in FedAT are assumed to be non-stragglers.

model on the classes 0, 5, 8, which are absent in the generic data but present in the client data. It is observed that the global model achieves test performance ranging from 20% to 84%, despite not having direct access to raw samples from these classes. These results demonstrate that FedAT effectively transfers client learned knowledge to the global model through AT and metadata training, while following the privacy-preserving rule of FL.

Method	$Diri(\alpha)$	Test Acc. (%) Class 0	Test Acc. (%) Class 5	Test Acc. (%) Class 8
FedAT	0.5	70.6	61.4	75.3
	0.1	84.7	62.4	20.0

Table 5.8: The performance of the global model on classes 0, 5, 8, absent in the generic data but present in client data. The obtained performance on these missing classes demonstrates successful transfer of client learned knowledge to the global model.

5.5 Federated Learning with Aggregated Prototype Feature Maps

As stated in Section 5.1, FedKAD focuses on alleviating the negative knowledge transfer from custom-size client models to the global model induced by model shit. At a high level, FedKAD achieves this objective by regularising KD with prototype feature maps which preserve intra-class knowledge.

FedKAD setup. FedKAD allows the distribution of custom-size client models. It learns a global model on the server using client consensus and aggregated prototype feature maps from clients under the coordination of a centralised server. The global model M_g parameterised with w_g is trained on the server. There are a total number of K non-straggler clients, each using a custom-size client model M_k parameterised with w_k to perform the local updates on its local data $\mathcal{D}_k$, $k \in [1, K]$. $\mathcal{D}_k$ has class labels, denoted by c with ($c \in [1, C]$). Similarly to earlier setup, the global model and these custom-size client models are constructed of a backbone with a feature extractor ϕ and the upper part θ. Again, FedKAD assumes a generic dataset $\mathcal{D}_g$ that is available on the server to facilitate knowledge distillation. The global updates and the local updates of FedKAD are described in the subsequent sections. Algorithm 4 provides an outline of FedKAD.

Algorithm 4 Federated Learning with Local **K**nowledge **A**ggregation and Knowlede **D**istillation.

```
1: Input: total $T$ communication rounds, a generic a dataset on the server $\mathcal{D}_g$, $E$ epochs of local updates, $w_g = \{\theta_g,\ \phi_g\}$, $w_k = \{\theta_k,\ \phi_k\}$ for all $k \in [1, K]$.
2: for $t = 1, \ldots, T$ do
3:     $K$ participant clients download the custom-size client model $w_k = \{\theta_k,\ \phi_k\}$ from the server.
4:     Clients:
5:     for Client $k \in [1, K]$ do
6:         $w_{k,\,E} \leftarrow$ Local updates $(w_k^t; \mathcal{D}_k, E)$ according to Equation 5.11.
7:         $\{F_{k,c}\}_{c=1}^{C} \leftarrow$ Local prototype feature maps $(; \phi_{k,\,E}, x_{k,c})$ according to Equation 5.21.
8:         Client $k$ uploads $w_{k,\,E}$ and $\{F_{k,c}\}_{c=1}^{C}$ to the server.
9:     end for
10:    Server:
11:    $\{(F_{g,\,c},\ c)\}_{c=1}^{C} \leftarrow$ Global prototype feature maps $(\{\{F_{k,\,c}\}_{k=1}^{K}\}_{c=1}^{C})$ according to Equation 5.26.
12:    $w_g^{t+1} \leftarrow$ Global updates $\left(w_g^t; z_{con,g}, \mathcal{D}_g, \{(F_{g,\,c},\ c)\}_{c=1}^{C}\right)$ according to Equation 5.29.
13:    for $k \in [1, K]$ on each of the uploaded $w_{k,\,E}$ do
14:        $w_k^{t+1} \leftarrow$ Global updates $\left(w_{k,E}; z_{con,g}, \mathcal{D}_g, \{(F_{g,\,c},\ c)\}_{c=1}^{C}\right)$ according to Equation 5.30.
15:    end for
16: end for
17: Return Global model $w_g^{T+1}$, client models $w_k^{T+1}$ for $k \in [1, K]$.
```

5.5.1 Local Updates of FedKAD

FedKAD first follows standard FL to update the client model on local data. Similar to FedAT, each client in FedKAD is assumed to be non-straggler with the adoption of custom-size client models. Participants are capable of performing E local training epochs and successfully uploading their training results to the server. At communication round t, the client model is updated from w_k^t to $w_{k,\,E}$ by following Equation 5.11 previously described in FedAT.

Constructing Local Prototype Feature Maps

In addition to standard model updates, clients in FedKAD construct local prototype feature maps, which can be used to correct the negative knowledge transfer during the global updates.

Local prototype feature maps preserve intra-class information. These prototype feature maps for a class are formulated by fusing latent representations of local samples sharing the same class label. Using the updated feature extractor $\phi_{k,\,E}$ in the client model, feature maps associated to the same class label c are first extracted, then they are averaged to form the local prototype feature maps for class c on client k as follows:

$$F_{k,\,c} = \frac{1}{|\mathcal{D}_{k,c}|} \sum_{\{x_{k,c}\} \in \mathcal{D}_{k,c}} f_{\phi_{k,\,E}} \left(\phi_{k,\,E};\; x_{k,c}\right) \tag{5.21}$$

where $f_{\phi_{k,\,E}}$ is the mapping function of the updated feature extractor. $x_{k,c}$ is the instance attached with the ground-truth label c in $\mathcal{D}_k$ and $\mathcal{D}_{k,c}$ is a subset of $\mathcal{D}_k$ with all instances belonging to label c. If a class is completely absent in the local data $\mathcal{D}_k$, FedKAD sets its corresponding feature map $F_{k,\,c}$ to 0. By aggregating feature maps locally, client k composes the set of local prototype feature maps, $\{F_{k,c}\}_{c=1}^{C}$, of fixed size of feature maps times the number of task classes C, regardless of the size of the local data.

Finally, client k sends the updated client model parameterised by $w_{k,\,E}$ and the set of local prototype feature maps $\{F_{k,c}\}_{c=1}^{C}$ to the server. The server is unable to relate local prototype feature maps to any particular data point in local data $\mathcal{D}_k$, thus user privacy is also well protected in FedKAD.

5.5.2 Global Updates of FedKAD

Following local updates across clients, the global updates transfer the knowledge learned by client models to the global model using knowledge distillation and local prototype feature maps.

Distilling Client Consensus for Knowledge Transfer

Recall Equation 5.13, FedAT relies on the ground-truth labels of the generic data to perform the knowledge distillation. In a more realistic scenario, the generic data collected in advance on the server could be unlabelled (Chang et al., 2019; Lin et al., 2020; Itahara et al., 2021), therefore FedKAD employs the knowledge distillation using the client consensus to supervise the training of the global model instead of using the labels in the generic data.

Client consensus. The client consensus is formed from the ensemble of predictions made by individual client models on instances from the generic data. Each unique client model predicts on a given instance, and their predictions are averaged to form a consensus. This consensus can serve as a pseudo label for supervising the learning of the global model on the server. Particularly, the predictions are the logits of the model, which are the outputs before the softmax activation layer. For an instance i in $\mathcal{D}_g$, each uploaded client model performs a forward pass to generate logits as follows:

$$z_{k,\,g}^{(i)} = f_{w_{k,\,E}}\left(w_{k,\,E};\ x_g^{(i)}\right) \tag{5.22}$$

where $x_g^{(i)}$ is the i-th instance of $\mathcal{D}_g$ and $z_{k,\,g}^{(i)}$ are the corresponding logits output by the client model. Then the client consensus on this instance is formulated with Equation 5.23.

$$z_{con,g}^{(i)} = \frac{1}{K}\sum_{k} z_{k,g}^{(i)} \tag{5.23}$$

To transfer the knowledge, knowledge distillation teaches the global model with the client consensus by minimising the difference between logits from the global model and the consensus. During the global updates of FedKAD, the consensus is used to supervise both the global model and uploaded client models. The knowledge distillation loss therefore for the global model and uploaded model from client k are defined by Equation 5.24 and Equation 5.25.

$$\mathcal{L}_{KD}\left(w_g;\ \mathcal{D}_g, z_{con,g}\right) = \sum_{x^{(i)} \in \mathcal{D}_g} \|f_{w_g}(w_g,\ x^{(i)}) - z_{con,g}^{(i)}\|_1 \tag{5.24}$$

$$\mathcal{L}_{KD}\left(w_{k;\ E};\ \mathcal{D}_g, z_{con,g}\right) = \sum_{x^{(i)} \in \mathcal{D}_g} \|f_{w_{k;\ E}}(w_{k;\ E},\ x^{(i)}) - z_{con,g}^{(i)}\|_1 \tag{5.25}$$

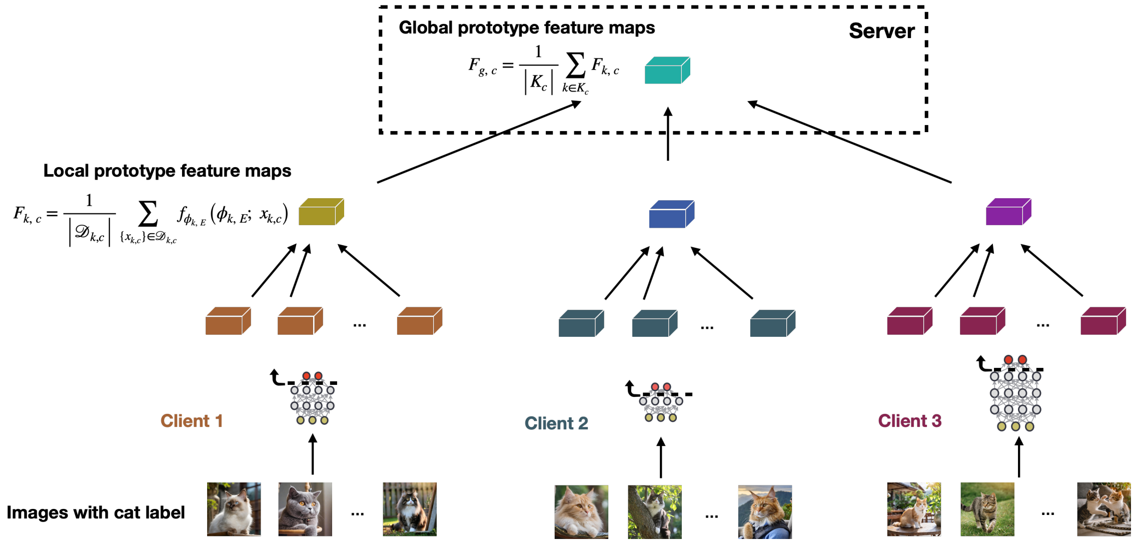
FedKAD employs L1 loss unless otherwise stated. However, other loss functions such as the cross entropy loss suggested by Equation 5.3 or $\mathcal{L}_2(\cdot)$ can also be applied. Optimised with Equation 5.24 and Equation 5.25, the global model and client models are optimised to approximate the client consensus on $\mathcal{D}_g$, learning the global knowledge distilled from all clients. The ground-truth labels of generic data are no longer needed in the knowledge distillation.

Constructing Global Prototype Feature Maps to Regularise KD

With local prototype feature maps, the server further constructs the global prototype feature maps class-wise. Specifically, the server averages all local prototype feature maps belonging to the same label c into the global prototype feature maps for label c as follows:

$$F_{g,\ c} = \frac{1}{|K_c|} \sum_{k \in K_c} F_{k,\ c} \tag{5.26}$$

where K_c is a subset of clients whose $F_{k,c}$ are non-zero, $F_{g,\ c}$ is the global prototype feature maps for class c on the server. $\{(F_{g,\ c},\ c)\}_{c=1}^{C}$, The global prototype feature maps are utilised to supervise the training of θ_g and $\theta_{k,E},\ \ k \in [1, K]$ using the cross-entropy loss.



Global prototype feature maps
Server
$F_{g,c} = \frac{1}{|K_c|} \sum_{k \in K_c} F_{k,c}$
Local prototype feature maps
$F_{k,c} = \frac{1}{|\mathscr{D}_{k,c}|} \sum_{\{x_{k,c}\} \in \mathscr{D}_{k,c}} f_{\phi_{k,E}}(\phi_{k,E};\ x_{k,c})$
Client 1
Client 2
Client 3
Images with cat label

Figure 5.11: The formulation of the global prototype feature maps for the "cat" class.

$$\mathcal{L}_{\theta_g}\left(\theta_g,\ \{(F_{g,\,c},\ c)\}_{c=1}^{C}\right) = \sum_{c=1}^{C} \ell_{ce}\left(f_{\theta_g}(\theta_g,\ F_{g,c}),\ c\right) \tag{5.27}$$

$$\mathcal{L}_{\theta_{k,E}}\left(\theta_{k,E},\ \{(F_{g,\,c},\ c)\}_{c=1}^{C}\right) = \sum_{c=1}^{C} \ell_{ce}\left(f_{\theta_{k,E}}(\theta_{k,E},\ F_{g,c}),\ c\right) \tag{5.28}$$

Finally, the global updates are performed by integrating the supervision from prototype feature maps with the knowledge distillation. At communication round t, the server updates the global model and client models to w_g^{t+1} and $w_k^{t+1},\ k \in [1, K]$ by adding the KD loss defined by Equations 5.24, 5.25 and cross-entropy loss defined by Equations 5.27, 5.28.

$$w_g^{t+1} = \underset{w_g^t}{\operatorname{argmin}} \left(\mathcal{L}_{KD}\left(w_g^t;\ \mathcal{D}_g, z_{con,g}\right) + \beta\mathcal{L}_{\theta_g^t}\left(\theta_g^t,\ \{(F_{g,\,c},\ c)\}_{c=1}^{C}\right)\right) \tag{5.29}$$

$$w_k^{t+1} = \underset{w_{k,E}}{\operatorname{argmin}} \left(\mathcal{L}_{KD}\left(w_{k,E};\ \mathcal{D}_g, z_{con,g}\right) + \beta\mathcal{L}_{\theta_{k,E}}\left(\theta_{k,E},\ \{(F_{g,\,c},\ c)\}_{c=1}^{C}\right)\right) \tag{5.30}$$

where β is the coefficient for the cross-entropy loss $\mathcal{L}_{\theta_g^t}$ and $\mathcal{L}_{\theta_{k,E}}$.

While KD aligns the global model and custom-size client models with client consensus, their upper parts are refined simultaneously by the global prototype feature maps encoding intra-class knowledge. The client models and the global model are split at the penultimate layer for FedKAD unless otherwise stated. The upper part, θ, is the classifier, and the feature extractor, ϕ, comprises the rest of the model. The feature maps are extracted at the penultimate layer.

5.6 FedKAD Experiments

Specific experimental setup for FedKAD. FedKAD is evaluated on CIFAR-10 and GSC datasets. The generic datasets used to facilitate KD are CIFAR-100 and the validation set of GSC respectively. The preprocessing of GSC is described in Section 3. Again, data heterogeneity is simulated by partitioning client data with $Diri(\alpha)$. FedKAD inherits the client setup from FedAT, with a total of 20 clients. The configuration of the global model and custom-size client models is described in Table 5.5. SGD optimiser with a learning rate of 0.1 is used for local updates and global updates. We randomly draw 5000 training samples from the generic data every communication round for KD. The training epochs for KD is set to 5. On CIFAR-10, the batch sizes for KD and local updates are set to 64 and 128 respectively. On GSC, the batch sizes for KD and local updates are set to 16 and 128 respectively. We tune the hyperparameter β in Equation 5.29, 5.30 for weighting the loss on prototype feature maps in the range of $\{1, 0.5, 0.1, 0.01\}$. Table 5.9 reports the validation performance from the grid search. It is found that using $\beta = 1$ and $\beta = 0.5$ achieve the best results for CIFAR-10 and GSC respectively.

β for FedKAD	0.01	0.1	0.5	1.0
Val. Acc. on CIFAR-10 (%)	59.64	60.78	61.02	61.32
Val. Acc. on GSC (%)	76.68	76.99	77.50	76.92

Table 5.9: The validation performance on CIFAR-10 and GSC by varying β values for FedKAD.

Baselines. FedKAD is compared with FedAvg McMahan et al. (2017) and FedProx Li et al. (2020c), which distribute the same size model to all clients. In addition, FedKAD is compared with FedMD Li and Wang (2019), which also enables custom-size client models and utilises KD in the global updates.

FedAvg does not introduce proper measures to mitigate system and data heterogeneity. The stragglers in local updates are simply discarded by the server and non-IID local data can cause significant performance degradation Hsu et al. (2019); Abdelmoniem et al. (2023). FedProx advances FedAvg to tackle system and data heterogeneity by adopting a proximal term in local updates and practising a flexible epoch number of local updates. Following the straggler setup proposed in FedAT, FedAvg and FedProx are set to consider the case of clients running on devices with limited computational resources. Particularly, we assume 0%, 40%, 60% of clients are stragglers, being consistently dropped out by the server due to being unable to finish the round before the synchronization deadline. Considering that original FedAvg was designed to operate with as low as 10% client participation McMahan et al. (2017), our baseline is not impairing FedAvg. FedProx also considers its variant allowing flexible local updates, denoted as *FedProx+*. FedProx+ assumes that only 60% or 40% of clients are non-stragglers to perform full training epochs, the rest clients are supposed to train randomly for 1 epoch to full local epochs E. We compare FedKAD with FedProx+ as they both target the straggler issue, but by adopting different approaches, distributing custom-size models or a universal model updated with flexible training epochs.

FedMD is the KD-based baseline that allows custom-size models. The original FedMD does not include the global model, only focusing on the averaged client model performance. As FedKAD focuses on the global model performance, the global model is also introduced to FedMD for a fair comparison. Concretely. This global model on the server is trained with KD in FedMD. Its performance is reported for comparison. We denote the original FedMD and the FedMD with the global model as *FedMD (clients)* and *FedMD (global)*. Similar to our earlier straggler setup for FedAT, FedMD and FedKAD both assume all client participants are non-stragglers. Additionally, to accelerate the convergence of FedMD and FedKAD,

Method	Cifar-10	GSC
Raw metadata	819.20	1390.07
Compressed metadata	3.28 ↓	11.47 ↓

Table 5.10: Compressing metadata with prototype feature maps. Float32 data type is assumed with Megabyte (MB) as the unit. Using prototype feature maps as metadata can greatly reduce its size, thereby alleviating the communication overhead.

the practice of FedMD is followed by initialising client models, training each client model only on its client data until convergence before starting FL.

5.6.1 Compressing Metadata with Local Prototype Feature Maps

The feature maps are extracted from the penultimate layer of the client model. With the WRN models, regardless of the depth of the custom-size client models. The feature maps all have an identical shape of 64×8×8, which are channel numbers, height and width respectively. By applying Equation 5.21 to formulate the local prototype feature maps, each client only contributes a set of features with a fixed size, which is the number of unique labels in the learning task. Uploading the local prototype feature maps reduces the size of metadata significantly. Table 5.10 compare the uploaded sizes of raw feature maps and local prototype feature maps. The size of metadata decreases from 819.20MB to only 3.28MB on CIFAR-10, dropping by 99.6%. Similarly, the size of metadata on GSC is reduced by 99.1%. The compression of metadata with prototype feature maps can greatly alleviate the introduced communication overhead.

5.6.2 Experiment Results

All experiments are run for 50 global rounds. The local training epochs E is set to 10. Table 5.11 reports the global model performance on test set. In summary, baselines that fuse a homogeneous model on the server, FedAvg and FedProx, still outperforms KD-based approaches, FedMD and proposed FedKAD.

However, FedKAD outperforms FedMD in three out of four scenarios. Particularly, with a high level of non-IIDness with $Diri(0.1)$, FedKAD improves the global performance by up to 3% on CIFAR-10. The only case where FedMD beats FedKAD by 0.68% is on CIFAR-10 with $Diri(0.5)$. Conversely, the performance improvement of FedKAD over FedMD on GSC is marginal, indicating that the introduction of prototype feature maps has a minor effect on GSC.

Compared with the strongest model fusion baseline FedProx and FedProx+, FedKAD is able to achieve up to 93% of its top performance given $\alpha = 0.1$ and up to 87% with $\alpha = 0.5$. In more realistic scenarios, where a significant proportion of clients are often stragglers and subsequently discarded by the server–such as 40% on CIFAR-10 and 60% on GSC–and high levels of data heterogeneity like $Diri(0.1)$, FedKAD is evident to outperform both FedAvg and FedProx. Thus, FedKAD demonstrates increasing effectiveness for knowledge transfer in extremely heterogeneous environments.

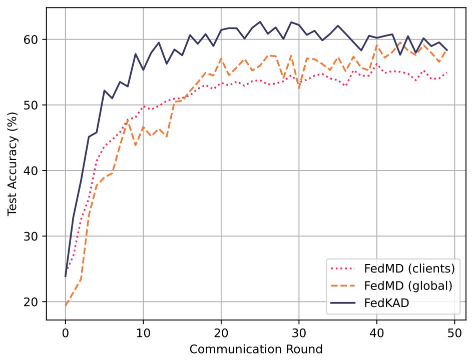
FedKAD is observed to have accelerated convergence compared to FedMD. Figure 5.12 illustrates the learning curves of the global model. Regarding the total FL rounds needed to reach top global model performance, FedKAD cuts the number to less than half with $\alpha = 0.1$. Even with weak data heterogeneity under $\alpha = 0.5$, it is clear that FedKAD hits its best performance much earlier than both FedMD and FedMD (global).

5.6.3 Mitigating Negative Knowledge Transfer with Prototype Feature Maps

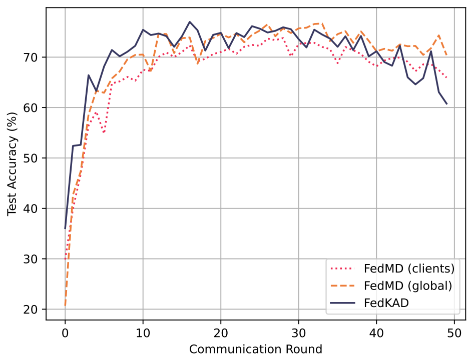
The adoption of prototype feature maps for regularising KD can mitigate the negative knowledge transfer during global updates. KD uses client consensus to supervise the global model training. The consensus is constructed by the outputs of client models, which have divergent parameters due to local updates on heterogeneous client data. Therefore, the reliability of the client consensus is compromised, particularly under high levels of client data heterogeneity (Chen and Chao, 2020;

Method	Straggler ratio (%)	CIFAR-10 $Diri(0.1)$	CIFAR-10 $Dira(0.5)$	GSC $Dir1(0.1)$	GSC $Diri(0.5)$
FedAvg	0	66.84	79.15	87.43	91.08
FedAvg	40	60.29	76.00	81.63	90.16
FedAvg	60	49.16	71.97	75.25	87.22
FedProx	0	66.18	**80.28**	**87.57**	**91.11**
FedProx	40	61.09	76.51	81.05	89.60
FedProx	60	48.02	72.09	76.04	86.55
FedProx+	40	**67.29**	79.11	87.34	90.87
FedProx+	60	65.46	79.34	86.82	90.48
FedMD (clients)	0	56.27	66.01	73.79	77.48
FedMD (global)	0	59.53	**68.74**	76.68	81.01
FedKAD	0	**62.67**	68.16	**77.00**	**81.13**

Table 5.11: Comparing global model performance (%) on CIFAR-10 and GSC test sets for baselines and FedKAD. The upper bracket of the table are model fusion methods that broadcast a homogeneous model across clients and the lower bracket are KD-based methods distributing custom-size client models. FedProx(+) achieves the highest performance overall. Among KD-based methods, FedKAD beats FedMD in three out of four scenarios, particularly pronounced when $\alpha = 0.1$. FedKAD achieves up to 93% of the best performance achieved by model fusion methods and show superior performance under high straggler ratios and data heterogeneity.



(a) CIFAR-10



(b) GSC

Figure 5.12: Comparing the global test performance between FedKAD and FedMD with strong data heterogeneity $Diri(0.1)$. It is obvious that FedKAD achieves faster convergence on both CIFAR-10 and GSC.

Party	Proto. regularisation on clients	E=5		E=10	
		$Diri(0.5)$	$Diri(0.1)$	$Diri(0.5)$	$Diri(0.1)$
Server	w/o	66.97	59.39	67.85	59.75
	w/	68.99 ↑	61.25 ↑	68.16 ↑	62.67 ↑
Client	w/o	64.46	54.37	64.08	55.13
	w/	65.58 ↑	57.43 ↑	65.83 ↑	59.13 ↑

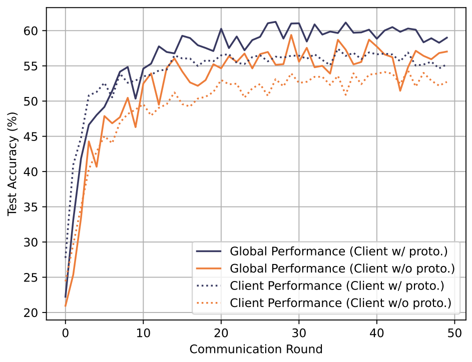
Table 5.12: The adoption of prototype feature maps to regularise KD improves client model performance in global updates, leading to improved global model performance.

Itahara et al., 2021). Unreliable client consensus can lead to negative knowledge transfer in KD. The substantial decrease in global performance observed on both CIFAR-10 and GSC, moving from $Diri(0.1)$ from $Diri(0.5)$ as shown in Table 5.11 underscores the impact of such negative knowledge transfer.

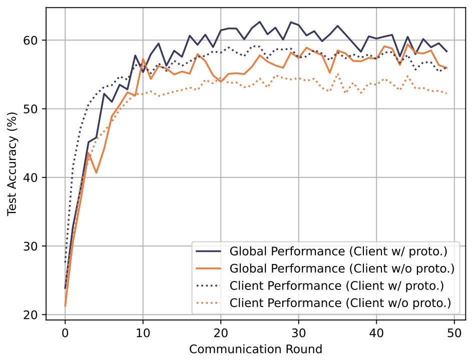
FedKAD integrates the upper part refinement with prototype feature maps into the KD to train the client models as described by Equation 5.30. The training loss on the prototype feature maps regularises knowledge distillation to prevent the upper part of the client model overfitting to the compromised client consensus induced by model shift. In this way, the negative knowledge transfer is mitigated. To validate this hypothesis, we compare the standard FedKAD with the variant that abandons the regularisation term, $\mathcal{L}_{\theta_{k,E}}$ in Equation 5.30. Table 5.12 reports both the client model performance (averaged) and the global model performance in global updates on CIFAR-10.

The inclusion of regularisation from prototype feature maps in KD proves crucial for FedKAD, benefiting both global and client performance. By leveraging prototype feature maps in KD for client models, the averaged client performance improved by up to 4%, particularly in the strong data heterogeneity scenario with $Diri(0.1)$. This observation empirically supports our hypothesis that training the upper part of the client model with prototype feature maps formulates a regularisation term to mitigate the negative knowledge transfer induced by model shift.

In contrast, the global model performance decreases by nearly 3% without the regularisation, causing FedKAD to lose its performance advantage over FedMD. Figure 5.13 further illustrates the learning curves of both client models and the global model, demonstrating that incorporating prototype feature maps enhances both global and client performance consistently.



(a) CIFAR-10, E=5



(b) CIFAR-10, E=10

Figure 5.13: The effect of prototype feature maps regularisation in KD. Without leveraging prototype feature maps, both client and global performance decrease.

5.7 Summary

This chapter introduces methods that improve knowledge transfer in the approach of custom-size client models for addressing the straggler issue. The adoption of custom-size client models converts stragglers into non-stragglers by reducing model size for resource-constrained clients. However, the knowledge transfer from custom-size client models to the global model needs to be improved due to the limitations of aligning outputs of models.

In the first part, FedAT is introduced to enhance the knowledge transfer between custom-size client models and the global model. Two knowledge sources, attention transfer (AT) and metadata training, are leveraged to transfer the learned knowledge from clients to the global model. Rigorous ablation studies show that AT is beneficial for transferring knowledge learned from a smaller-size client model to a larger global model. In addition, AT can regularise global model training, preventing overfitting on the generic dataset used for knowledge transfer. Metadata training demonstrates its efficacy in teaching the global model to generalise on classes absent from the generic data.

FedAT is further evaluated in a multi-client scenario. Compared with FedAvg with varying straggler ratios, FedAT performs closely to the hypothetical FedAvg where all clients are non-straggler in the strong data heterogeneity case, with only a marginal difference of 0.23%. FedAT improves the performance of realistic FedAvg baselines by up to 19%. These results demonstrate FedAT is highly effective to transfer locally learned knowledge to the global model. FedAT successfully solves the research question derived in the approach of custom-size client models, demonstrating that AT and metadata training are effective knowledge transfer methods in heterogeneous FL with custom-size client models. This advances previous works that employ model outputs as the sole transfer knowledge source.

In the second part, FedKAD is further proposed to mitigate negative knowledge transfer induced by model shift and strengthen user privacy protection by constructing prototype feature maps–metadata in a compressed form–and leveraging them to regularise knowledge distillation (KD) during the global updates. Prototype feature maps are created to preserve intra-class knowledge. By leveraging them to regularise KD, they effectively prevent the global model from overfitting to unreliable client consensus, which is the source of negative knowledge transfer. In extreme data heterogeneity scenarios, FedKAD improves global model performance and accelerates FL convergence compared to another popular KD-based FL method, FedMD, by noticeable margins, thereby reducing the overall energy cost for clients.

FedAT and FedKAD contribute novel and effective methods to the approach of custom-size client models by improving the quality of knowledge transfer between custom-size client models and the global model, thereby enhancing the potential of this approach to address the straggler issue.

Chapter 4 and Chapter 5 fill the research gaps in the approaches of reducing the number of trainable parameters in client models for resource-constrained devices. In the next chapter, we will introduce the approach of reducing the data for training client models, thereby improving training efficiency for resource-constrained devices.

Chapter 6

Addressing the Straggler Issue with Active Data Selection

6.1 Introduction

6.1.1 Research Questions to Address

While active data selection is a well-established approach for improving the training efficiency (Mindermann et al., 2022; Yang et al., 2023; Li et al., 2022) in centralised learning, its application in FL for addressing the straggler issue remains very limited (Kairouz et al., 2021). Reviewed in Section 2.2.4, previous FL works that adopt active data selection either focus on easing the labeling effort (Kim et al., 2023b; Ahn et al., 2024) for client instances or introduces significant computational overhead, thereby aggravating the straggler issue for clients in order to identify training samples beneficial for FL performance under heterogeneous data environments (Nagalapatti et al., 2022). Employing active data selection to address the straggler issue has not been sufficiently studied in the context of FL. To this end, the research question approached in this sub-research path is, "How to leverage active data selection to address the straggler issue and what impact does active data selection bring to the global model performance under heterogeneous data?"

By addressing this research question, this chapter aims to fill the aforementioned research gap in existing literature by developing efficient on-device data selection methods that can lead to reduced workloads for clients and improved FL performance. Particularly, with the developed data selection methods, We hypothesise workloads on clients can be reduced by updating models with fewer training instances, thereby addressing the straggler issue. The tradeoff of using fewer training samples is that the performance of the global model could be penalised. However, if the selected training samples are highly effective for learning, it is possible to minimise the performance penalty. Moreover, the contribution from stragglers is enabled by alleviating their workload. The additional contributions from them can eliminate the negative impact on the global model performance induced by reduced training samples. Therefore, the hypothesis is the global model performance can be improved if the active data selection is effective and the contribution from stragglers is enabled.

6.1.2 Proposed Methods and Rationale

The computational efficiency of data selection is the decisive factor for its success in our work. Particularly, our data selection strategies shall not introduce heavy computing efforts to clients for identifying useful learning instances, as contrast to some active learning works where a significantly amount of computing is added to the learning system due to the optimisation of additional deep learning models for querying data for annotations (Sinha et al., 2019; Haussmann et al., 2019). Otherwise, the data selection could aggravate the straggler issue rather than addressing it. To this end, two computationally efficient active data selection methods, the clustering-based data selection and entropy-based data selection are introduced for selecting client data in this work.

Notably, clustering-based data selection utilising K-means clustering (MacQueen et al., 1967) and entropy-based active data selection employing the Shannon entropy (Shannon, 1948) with a hardened softmax activation function are proposed to actively select useful local instances for FL. K-means clustering is arguably the most popular clustering method due to its simplicity and computational efficiency (Yuan and Yang, 2019; Kodinariya et al., 2013; Course). In terms of entropy-based data selection, it only requires an additional forward pass through the client model on all client data to calculate their associated entropy once for all. The process adds minimal computational overhead to the client, enabling resource-constrained clients to strategically select useful training data for FL. FedEntropy and FedAvg-BE are previous works close to this research in terms of adopting the entropy for selecting client data in FL. However, their studies mainly target to address the data heterogeneity challenge rather than the straggler issue. Therefore, they lack a comprehensive evaluation of the learning efficiency of clients using the entropy-based data selection. This thesis advances their works by not only tailoring the entropy-based data selection to solve the straggler issue but also improving FL convergence by mitigating model shift.

To conclude, the novelties of clustering-based data selection and entropy-based data selection are threefold: 1) The introduction of K-means clustering and Shannon entropy in this thesis is based on their advantage of selecting data with minimal computing efforts. Both methods have not been applied to FL particularly for addressing the straggler issue so far; 2) In entropy-based data selection, the hardened softmax activation function adopted for calculating the Shannon entropy is unique to this work. It is introduced to improve the effectiveness of data selection; 3) The learning efficiency across clients will be evaluated for demonstrating its efficacy in addressing the straggler issue. This metric has not been utilised in previous FL works adopting active data selection.

By bridging the research gap in existing studies, this work contributes to the knowledge of introducing active data selection to address the straggler issue. The following sections provide overviews of the developed FL algorithms that adopt clustering-based data selection and entropy-based data selection. They are **Fed**erated learning setting by introducing a **Split** model training paradigm (FedSplit) and **Fed**erated **F**ine-**T**uning with **E**ntropy-based **D**ata **S**election (FedFT-EDS). FedSplit will reveal the penalisation on global model performance when the size of local training data is significantly reduced. FedFT-EDS will show its efficacy in overcoming the performance penalisation by identifying the most beneficial local instances for learning. On top of entropy-based data selection, FedFT-EDS proposes a unique pretraining phase to initialise a good global model that can mitigate model shift, further boosting FL performance in terms of both the convergence and generalisation.

This chapter conducts a comprehensive evaluation of client learning efficiency to demonstrate the proposed entropy-based data selection method can significantly reduce workloads on clients, as well as improve global model performance by large margins. Therefore, the research question is answered as follows: "Assisted with pretraining, clients can leverage entropy-based active data selection to significantly improve their learning efficiency, thereby effectively reducing their workloads. The global model performance is further improved rather than being penalised by selecting the most beneficial client instances for learning."

6.1.3 Selecting Latent Feature Maps with K-means Clustering

In Chapter 5, FedAT and FedKAD have demonstrated that latent feature maps are an effective knowledge source for transferring locally learned knowledge to the global model. Based on this proven discovery, this chapter first focuses on using active data selection to select the most representative feature maps by applying K-means clustering to the latent representations. The upper layers of the global model are trained using these representative feature maps rather than all feature representations.

Concretely, we introduce a new paradigm for Federated learning called FedSplit, which alters the transitional federated learning setting. In FedSplit, the federated learned model is composed of two parts. The lower part serves as a feature extractor to extract generic features across clients, while the upper part is more sensitive to feature maps extracted.

To train the feature extractor, a process similar to federated averaging is employed, aggregating results of local updates into the lower part of the composed model. Meanwhile, the upper part is trained using metadata consisting of feature maps actively selected from centroids of clusters formed by K-means clustering. FedSplit allows for examining the effectiveness of feature map selection based on the performance achieved by the composed model.

Evaluated on CIFAR-10, the performance of the composed model shows a significant penalty when using selected feature maps that account for only 1.6% of all feature maps extracted from clients. This performance gap is the biggest when feature maps are extracted from lower layer levels of client models, with a gap of up to 22%. However, this gap can be considerably narrowed by extracting feature maps from higher layer levels. Notably, when feature maps are extracted from the penultimate layer and selected using K-means clustering, the performance of the composed model decreases by only 4%. This moderate performance penalty occurs despite a significant reduction of 98% in the feature maps used to update the upper part of the composed model. This result demonstrates that a large portion of the global performance can be preserved by training the upper part of the model with a very small number of representative feature maps selected through K-means clustering.

6.1.4 Selecting Client Data with Shannon Entropy

The promising results obtained in the study of FedSplit encourage this work to further seek an effective client data selection solution that mitigates the penalising effect on the global model performance associated with reduced training data.

To achieve this objective, transfer learning is employed in data selection, leveraging its capability to achieve competitive learning performance on target tasks using less training data, assisted by prior learning on source tasks (Khan et al., 2019; Alzubaidi et al., 2021). Particularly, this work proposes a pretraining phase for FL, where the global model is pretrained on a source domain and is offloaded to clients to perform a one-round FL. The pretraining phase learns a generic feature extractor that effectively translates raw client data into representations in the latent space. Consequently, the feature extractor can be shared across all clients without retraining. By keeping the pretrained feature extractor fixed during local updates, clients can fine-tune a partial model using a reduced set of selected samples, significantly enhancing the learning efficiency without compromised performance. This method is termed **Fed**erated **F**ine-**T**uning (FedFT).

Building on FedFT, entropy-based data selection is further introduced to guide the selection of data for fine-tuning the partial model. Participating clients select the most useful training samples by evaluating their Shannon entropy. The proposed method is denoted as FedFT-EDS, **Fed**erated **F**ine-**T**uning with **E**ntropy-based **D**ata **S**election. FedFT-EDS allows clients to perform local updates efficiently and intelligently.

Based on Centered Kernel Alignment (CKA) (Kornblith et al., 2019; Nguyen et al., 2020) analysis, we demonstrate that the global model initialised the pre-training phase can resist model shift. With mitigated model shift, FedFT greatly improves FL convergence. Evaluated on CIFAR-10 and CIFAR-100, FedFT achieves superior performance in terms of both top test performance of the global model and learning efficiency–a metrics to measure client training time relative to the global model performance–compared to the FedAvg baseline without pretraining.

Furthermore, FedFT-EDS demonstrates that entropy-based data selection is an effective method for identifying the most useful client samples in heterogeneous FL. FedFT-EDS achieves better performance than FedFT without data selection or with random data selection. It not only improves the global model performance of chosen baselines, including FedAvg, FedProx, and MOON, by up to 5% but also at least triples the learning efficiency, reducing energy costs for resource-constrained clients throughout FL.

FedFT-EDS is observed to consistently narrow the performance gap between heterogeneous FL and ideal centralised learning by 30% to 75%. When compared to FedFT using all local training samples, FedFT-EDS achieves even better global model performance and faster convergence. We observe that FedFT-EDS trained on 50% of client data outperforms FedFT trained on all client data, indicating that 50% local instances bring no performance gains for FL if we include them for training in this study case. Additionally, FedFT-EDS consistently demonstrates faster convergences and superior global model performance compared to FedFT with random client data selection.

The superior performance of FedFT-EDS is even more pronounced with an increased size of the client pool, demonstrating its scalability in large-scale FL settings. Moreover, FedFT-EDS proves effective in the scenario of drastic domain shift in transfer learning. It is observed that FedFT-EDS initialised with a global model trained on Small ImageNet outperforms FedAvg and FedFT by up to 4% on GSC.

To this end, FedFT-EDS contributes valuable insights that address the the research question derived in Section 6.1.1: "How to leverage active data selection to addresses the straggler issue and what impact does the active data selection bring to the global model performance under heterogeneous data?" Particularly,

- CKA analysis demonstrates that the pretraining can mitigate model shift, thereby significantly accelerating FL convergence.
- Assisted with the pretrained global model, clients adopting entropy-based active data selection improve their learning efficiency in FL, effectively solving the straggler issue.
- Entropy-based data selection shows that over half of the client data is not beneficial for FL performance under heterogeneous data. The global model performance is further improved by selecting the most beneficial client instances for learning rather than being penalised due to reduced training data.

With these findings, this study bridges the research gap by demonstrating that active data selection can effectively address the straggler issue, making it an effective approach to encourage the participation of resource-constrained clients in FL.

6.2 Selecting Feature Maps to Train the Global Model

The proposed FedSplit partitions the global model into two parts: the lower part, which is the feature extractor parameterised with ϕ, and the upper part parameterised with θ. The feature extractor is trained with the standard federated averaging approach, where locally updated models are aggregated on the server to form the feature extractor. Conversely, the upper part is trained exclusively on the server using feature maps extracted and uploaded by clients. Figure 6.1 and Algorithm 5 outline FedSplit. The subsequent sections detail K-means clustering data selection, local updates and global updates of FedSplit.

6.2.1 K-means Clustering for Data Selection

Clustering (Jain and Dubes, 1988) is commonly used for data analysis as an unsupervised machine learning technique to group data points without available labels on the basis of their properties (Seif, 2018). Under a certain similarity criterion, data points grouped in the same cluster share common characteristics, whereas data points from different clusters have discriminating features. Clustering is widely involved in feature selection and labelling in the unsupervised learning (Caron et al., 2018; Hancer et al., 2020). By clustering features based on their similarity, it helps to select the most useful features and remove redundant ones (Chormunge and Jena, 2018).

As this thesis centres on reducing workloads on resources-constrained clients, the computational efficiency of clustering becomes a critical factor for its adoption in selecting features on clients. K-means clustering (MacQueen et al., 1967) is arguably one of the most popular clustering methods due to its simplicity and computational efficiency (Yuan and Yang, 2019; Kodinariya et al., 2013; Course). However, its adoption to select feature representations for FL is underexplored in previous research.

K-means clustering measures the similarity of the features or data points with the Euclidean distance. It iteratively assigns data points into clusters based on the calculation of their Euclidean distances to the cluster centroids. K-means initially assumes there is a total number of C disjoint clusters, denoted as $S = \{S_1, S_2, \cdots, S_k, \cdots, S_C\}$, to allocate the a set of data points $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$. Formally, the objective of K-means clustering is to minimise the within-cluster sum of squares (MacQueen et al., 1967) as follows:

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{C} \sum_{\mathbf{x} \in S_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2 \tag{6.1}$$

where $\boldsymbol{\mu}_i$ is the centroid of data points in S_i defined by Equation 6.2.

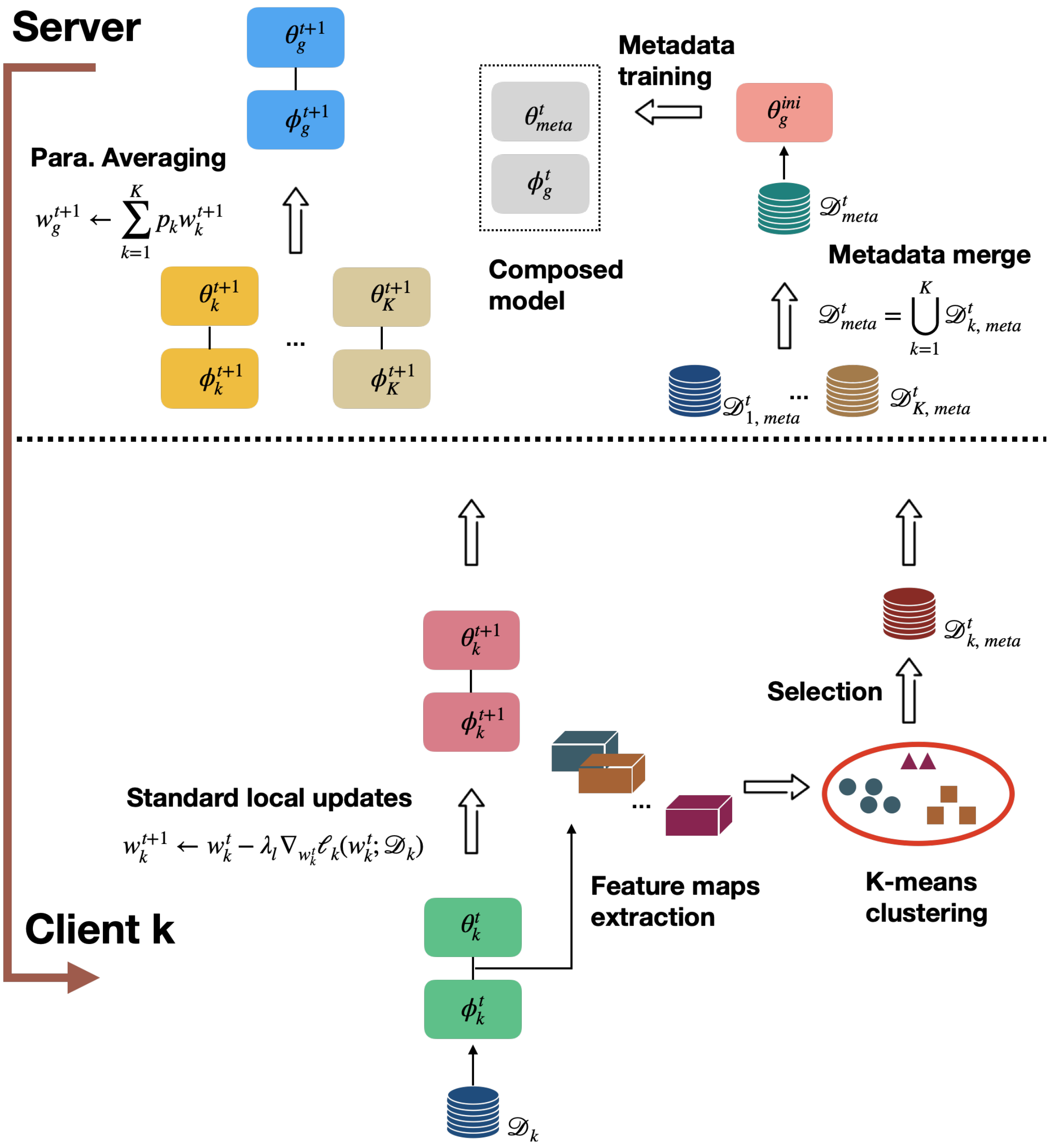


Figure 6.1: Selecting feature maps to update the upper layers of the global model.

$$\boldsymbol{\mu_i} = \frac{1}{|S_i|} \sum_{\mathbf{x} \in S_i} \mathbf{x} \tag{6.2}$$

K-means clustering solves the above optimisation with Expectation-Maximum (EM) algorithm. Given an initial set of C centroids, K-means repeats the following two steps until the assignment of data points stops.

- **Assignment step:** Assign each data point to its closest centroid, based on the Euclidean distance between the data point and the centroid.
- **Update step:** Update the centroid for each cluster by calculating the mean of all data points within the cluster.

In practice, there is no need to implement the K-means clustering algorithm from scratch. Notably, popular machine learning libraries such as Scikit-Learn and Faiss can perform the K-means clustering with just a few lines of code.

Once all data points are grouped, a subset of data points closest to each centroid is selected to represent all instances within that cluster. These representative data points are used for model training instead of the entire dataset, thereby reducing training burdens.

6.2.2 Local Updates of FedSplit with Feature Maps Selection

Local updates on the client side start with the selection of feature maps. At the start of t-th communication round, the participant client k downloads the global model from the server as the client model parameterised by $w_k^t = \{\phi_k^t, \theta_k^t\}$ from the server. Then, with Equation 6.3, feature maps are extracted by forward passing all the local samples through the feature extractor, ϕ_g, forming the set of feature maps F_k^t with element of $F_k^{t,(i)}$ corresponding to the client data $x_k^{(i)} \in \mathcal{D}_k,\ i = 1, \cdots, |\mathcal{D}_k|$.

$$F_k^{t,(i)} = f_{\phi_k^t}(\phi_k^t; x_k^{(i)}) \tag{6.3}$$

Similar to the study conducted in FedAT, three different layer levels are exercised to split the model, visualised in Figure 5.6. Notations of $l \in \{low, mid, up\}$ are used to indicate the three different layer levels, lower layers $l = low$, mid layers $l = mid$, and upper layers $l = up$. $F_k^{t,(i)}$ can be extracted from any of the layer levels.

After extracting the feature maps, the client k forms the metadata, $\mathcal{D}_{k,\,meta}^t$, by selecting the most representative feature maps from F_k^t rather than using all feature maps to construct the metadata.

K-means clustering is adopted to identify the most representative feature maps, as described in Section 6.2.1. Feature maps are clustered into groups, and within each cluster, samples closest to its centroid are selected as the most representative samples for the group. Practically, the three-dimension feature maps are flatten into a vector of single dimension. Principal Component Analysis (PCA) (Jolliffe, 2002) is then applied to reduce the dimension of vectors before clustering. K-means clustering is performed on the dimension-reduced activation maps to group feature maps with identical class labels.

The insight behind using K-means clustering is that feature maps sharing similar characteristics can be projected into the same cluster. Within each cluster, we choose the sample that is closest to the cluster centre determined by the Euclidean distance as the most representative sample for its group. This selected sample is assumed to include the most common characteristics of all other samples in its cluster. Finally, the metadata for client k, $\mathcal{D}_{k,\,meta}^t$, consists of the selected feature maps identified as the most representative samples from F_k, along with their assigned labels.

$$\mathcal{D}_{k,\,meta}^t = \bigcup_{i=1}^{|F_{k,select}^t|} (F_k^{t,(i)}, y_k^{(i)}) \tag{6.4}$$

where, $F_{k,select}^t$ is the set of the selected feature maps.

The number of expected clusters in K-means is a crucial hyperparameter (Pham et al., 2005). Choosing a smaller number of clusters can greatly reduce the size of metadata. Ideally, with an appropriately tuned cluster number, the selected feature maps are assumed to be capable of updating the global model effectively without a significant loss of generalisation. Details regarding the setup of the cluster number in K-means clustering are described in Section 6.3.

After clustering and selecting feature maps, the client follows standard FL to update its local model. For client k, with Equation 2.3, the parameters of client model are updated to $w_k^{t+1} = \{\phi_k^{t+1},\ \theta_k^{t+1}\}$ after E local training epochs. Client k sends its metadata $\mathcal{D}_{k,\ meta}^t$ and the updated model w_k^{t+1} to the server for the global updates.

6.2.3 Global updates of FedSplit with Feature Maps Selection

The server collects all uploaded metadata from clients and merge them into a single metadata set, defined by $\mathcal{D}_{meta}^t = \bigcup_{k=1}^{K} \mathcal{D}_{k,\ meta}^t$. The upper part of the global model is then trained using $\mathcal{D}_{meta}^t$. The layer level for partitioning the global model is predetermined by l, and the same value is used for creating the feature maps on clients as introduced in Section 6.2.2. Hence, metadata from clients is able to match the input dimensions of the upper part of the global model. By selecting feature maps with K-means clustering, the size of the metadata set is significantly reduced.

To strengthen knowledge transfer using metadata, we train an initialised upper part of the global model, denoted by θ_g^{ini}, from scratch for a specified number of training epochs using metadata. This strategy ensures that the global model effectively learns the knowledge conveyed in the metadata. θ_g^{ini} is updated to θ_{meta}^t with Equation 6.5.

$$\theta_{meta}^t = \underset{\theta_g^{ini}}{\operatorname{argmin}} \sum_{(F_k^{(i)}, y_k^{(i)}) \in \mathcal{D}_{meta}^t} \ell_{ce}(f_{\theta_g^{ini}}(\theta_g^{ini},\ F_k^{(i)}),\ y_k^{(i)}) \tag{6.5}$$

Conventional federated averaging is used to update the lower part of the global model. The server fuses the uploaded w_k^{t+1} from all participant clients with Equation 2.4 to form the model w_g^{t+1} distributed at the $t+1$-th communication round. Finally, the server composes the federated learned model, with its upper part from θ_{meta}^t and lower part from ϕ_g^t. Consequently, w_g^{t+1} encompasses the updated parameters of the lower part, ϕ_g^{t+1}, which will be used to compose the model for distribution in the $t+1$-th communication round. The composed model performance is evaluated to analyse the efficacy of the proposed clustering-based data selection.

Algorithm 5 FedSplit with K-means clustering to select feature maps.

```
1: **Input:** total $T$ rounds, $E$ full updates epochs, $\theta$ classifier, $\phi$ feature extractor,
   an initialised upper part of the global model $\theta_g^{ini}$.
2: for $t = 1, \ldots, T$ do
3:     $K$ clients are available for training and they download the global model
       $w_g^t = \{\theta_g^t, \phi_g^t\}$ from the server.
4:     Clients:
5:     for Client $k \in [1, K]$ do
6:         $F_k^t \leftarrow$ Feature maps extraction $(\phi_k^t;\ \mathcal{D}_k)$ according to Equation 6.3.
7:         $\mathcal{D}_{k,\,meta}^t \leftarrow$ Selection with K-means clustering $(F_k^t)$ according to
           Equation 6.4.
8:         $w_k^{t+1} \leftarrow$ Standard local updates $(w_k^t;\ \mathcal{D}_k)$ according to Equation 2.3.
9:         Client $k$ uploads $w_k^{t+1} = \{\theta_k^t, \phi_k^t\}$ and $\mathcal{D}_{k,\,meta}^t$ to the server.
10:    end for
11:    Server:
12:    Aggregates all uploaded client metadata $\mathcal{D}_{meta}^t = \bigcup_{k=1}^{K} \mathcal{D}_{k,\,meta}^t$.
13:    $\theta_{meta}^t \leftarrow$ Metadata training $\left(\theta_g^{ini};\ \mathcal{D}_{meta}^t\right)$ according to Equation 6.5.
14:    Averages all uploaded $w_k^{t+1},\ k \in [1, K]$ into the global model $w_g^{t+1}$ dis-
       tributed in communication round $t + 1$ with Equation 2.4.
15: end for
16: Return Composed model $\{\theta_{meta}^T, \phi_g^T\}$
```

6.3 FedSplit Experiments

Special Experimental Setup for FedSplit. Experiments on CIFAR-10 are conducted for evaluating FedSplit. A uniform WRN with a depth of 40 and width of 1 (WRN-40-1) is used as the distributed model. The number of clients is set

Model	Layer level	Feature maps dimensions
WRN-40-1	low	$16 \times 32 \times 32$
	mid	$32 \times 16 \times 16$
	up	$64 \times 8 \times 8$

Table 6.1: Feature map dimensions across layers in the WRN-40-1 model vary: higher layers generates lower-resolution feature maps with more channels.

to 20 and all clients are assumed to be non-stragglers, completing feature maps selection and local updates in time before finishing the round. To make a fair comparison, chosen baselines also assume full client participation. We follow the setup used (Liang et al., 2020) to partition the non-IID local data, setting up an extreme data heterogeneity scenario, where each client hosts only 2500 images randomly selected from two random classes of CIFAR-10. SGD optimiser with a learning rate of 0.1 is used for both local updates and metadata training. The batch size for local updates is set to 50. Training epochs are set to 1 and 100 respectively for local updates and metadata training. Experiments are run for 100 communication rounds.

6.3.1 Hyperparameter Studies for K-means Clustering

The efficacy of the proposed data selection method relies on PCA and K-means clustering. As such, a series of experiments is conducted to shed insights into selecting important hyperparameters for PCA and K-means clustering.

PCA projects a high-dimensional vector into a lower-dimensional space (sklearn). The key hyperparameter of PCA is the number of components, denoted as n_{com}, which determines the dimensionality of the projected space. For image inputs from CIFAR-10, the dimensions of feature maps vary depending on the layer level from which they are extracted in the model. Table 6.1 details the sizes of feature maps extracted at three layer levels in WRN-40-1.

A commonly used tool used to decide the n_{com} is the Cumulative Explained

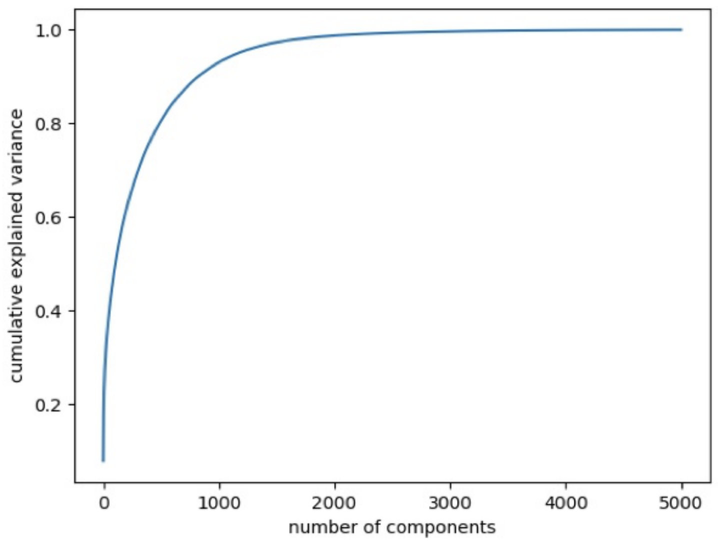
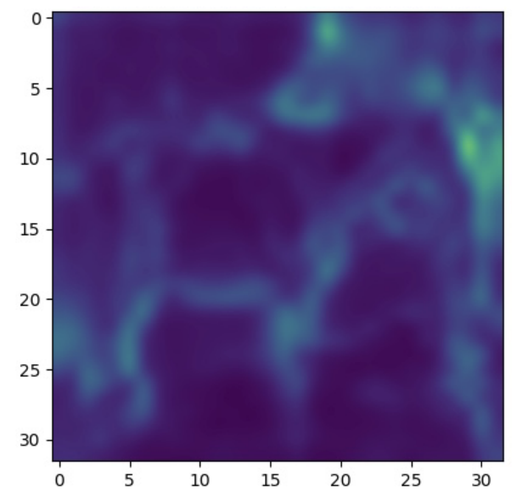


Figure 6.2: Cumulative Explained Variance plot of PCA on feature maps extracted from the low layer level. The y-axis, scaled from 0 to 1, indicates the percentage of variance preserved by PCA. With $n_{com} = 2166$, more than 99% of variance is preserved.

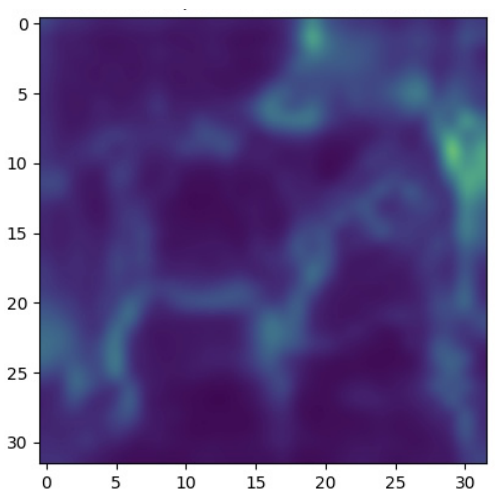
Variance (CEV) plot (Natarajan). Generally, the greater n_{com} is, the better the compressed feature maps capture the variance in the original feature maps. In other words, more original information of the data is retained with a larger n_{com} (VanderPlas, 2016). CEV shows how much percentage of variance can be preserved with a given n_{com}. It helps in choosing an appropriate n_{com} to effectively compress the data without losing substantial information. Using the feature maps extracted from the low layer level, Figure 6.2 describes the variance preserved against n_{com}. With $n_{com} = 1000$, PCA can retain more than 90% variance. By setting n_{com} to 2166, more than 99% variance is preserved.

Figure 6.3 illustrates the attention map of a horse from CIFAR-10 reconstructed by the feature maps compressed by PCA using various n_{com} values. With a smaller n_{com}, such as 500, it is observed that the reconstructed attention map is blurred slightly. In contrast, the attention map recovered with $n_{com} = 2166$ is nearly indistinguishable from the original.

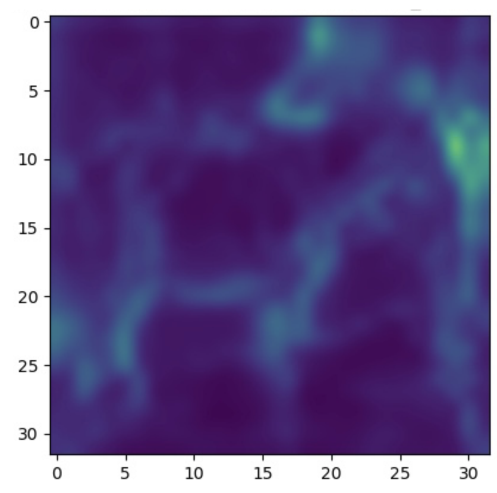
By observing Figure 6.3 and Figure 6.2, we set n_{com} to 500 for PCA to compress feature maps. This choice significantly reduces the dimension of feature maps to a



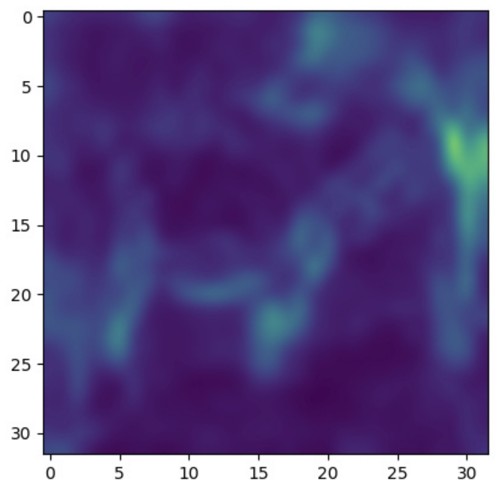
(a) Original AM



(b) Reconstructed AM, $n_{com} = 2166$

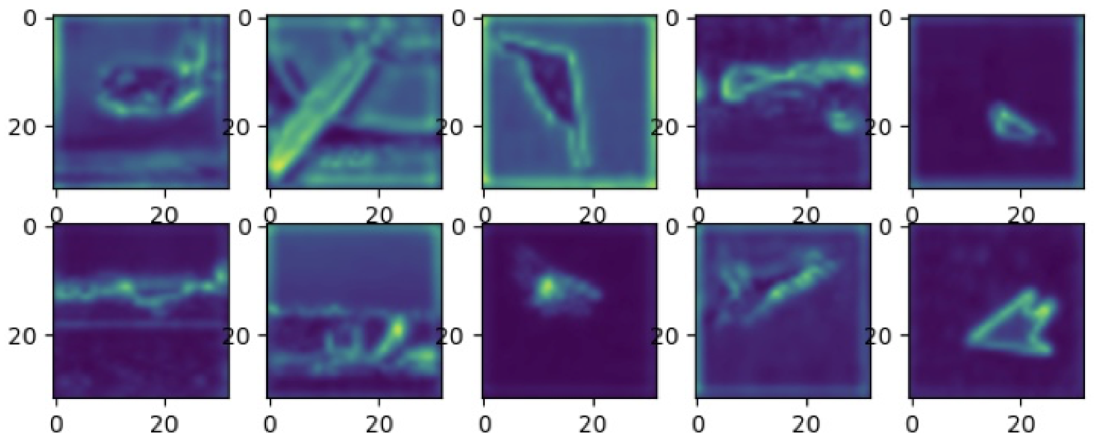


(c) Reconstructed AM, $n_{com} = 1000$



(d) Reconstructed AM, $n_{com} = 500$

Figure 6.3: Attention map reconstruction. Smaller n_{com} leads to more information loss. The original attention map of a horse from CIFAR-10 and its reconstructions using PCA-compressed feature maps with different n_com values are shown. The feature maps from the low layer level of the WRN model are visualised on a 32×32 grid.


(a) Attention map of the selected feature maps

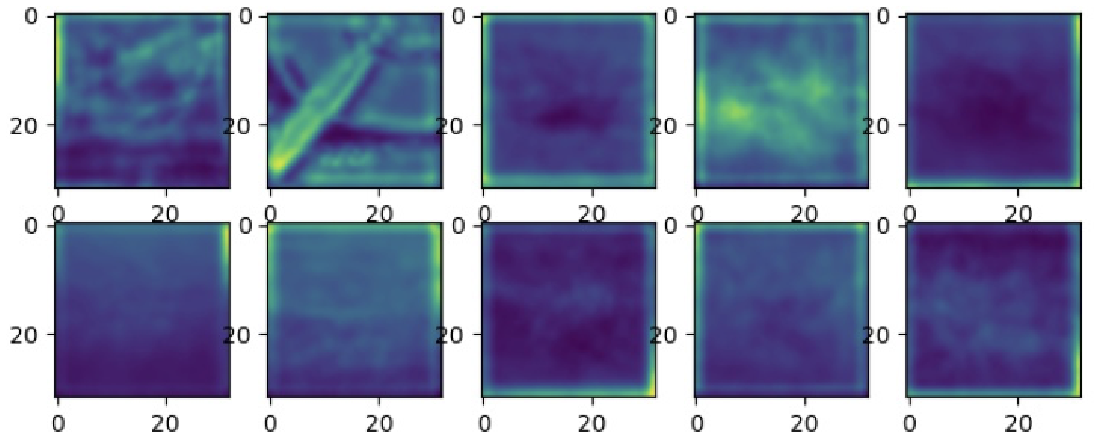

(b) Attention map of the centroids

Figure 6.4: Selected feature maps visualised through attention maps for the airplane class. The number of clusters in K-means clustering is set to 10. The top 10 represent selected feature maps, while the bottom 10 represent cluster centroids.

vector of size 500, thus improving the computational efficiency of K-means clustering. Additionally, it preserves more than 80% of the variance, and the reconstructed attention map is observed to retain most of the details.

On top of the compressed feature maps, we further showcase the selected feature maps using K-means clustering with an exemplary study. Specifically, we visualise the attention maps constructed from the feature maps selected within the airplane class in the CIFAR-10 training set. The clustering is based on 500 samples randomly selected from the airplane class, simulating the feature maps selection performed on one client.

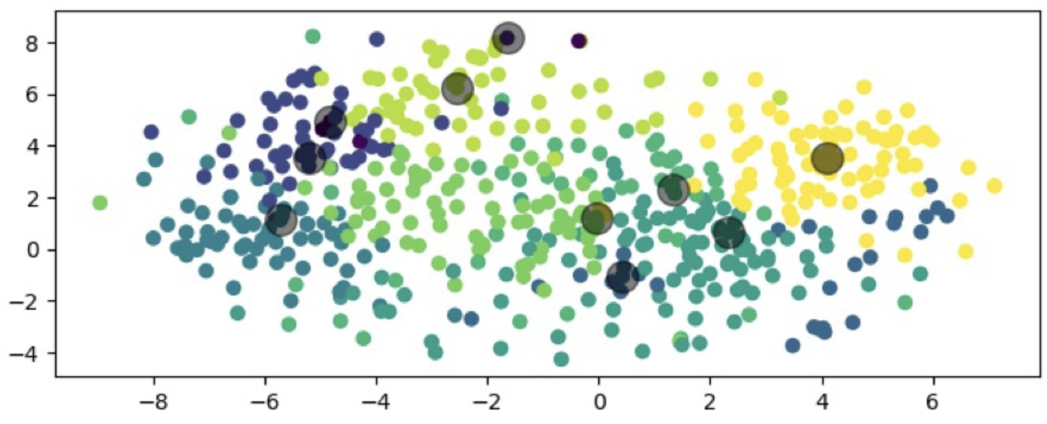
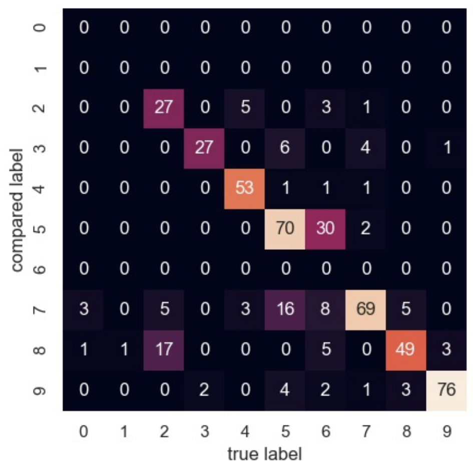


Figure 6.5: Visualisation of the clustering results with K-means. Dots with the same colour are clustered in the same group. The large grey dot represents the selected feature maps that are closest to the cluster centroid.

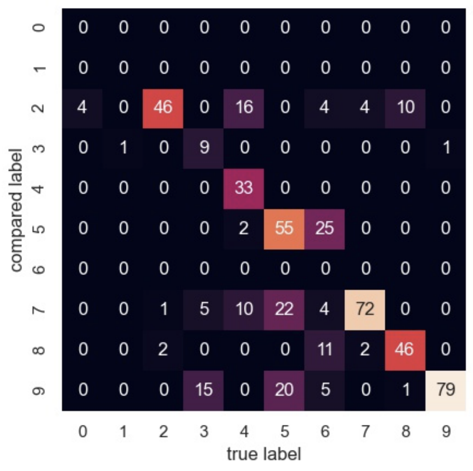
The clustering number in K-means is set to 10, grouping the feature maps representing the 500 airplane samples into 10 clusters. For each cluster, we select the feature maps closest to its centroid for visualisation. Figure 6.4 displays the attention maps formulated by the selected feature maps and centroids. The selected feature maps represent the airplane samples with varying shapes and positions within the images. However, the attention maps associated with centroids are mostly unidentifiable because the centroid is an average of the feature maps in the cluster. Therefore, instead of using the feature maps of the centroid directly, the feature maps closest to the centroid are selected.

Figure 6.5 uses t-distributed Stochastic Neighbour Embedding (t-SNE) (Van der Maaten and Hinton, 2008) to visualise the clustering results with K-means in a 2D plate. t-SNE is a widely used statistical tool to project high-dimensional features into a two-dimensional map for visualisation purposes.

Finally, we investigate how the choice of n_{com} in PCA influences the clustering results. Specifically, each group clustered by K-means is identified by a unique label. The confusion matrix (CM) (Stehman, 1997) is used to assess the consistency of cluster assignments using PCA with different n_{com} values. Figure 6.6 displays the CMs for PCA with $n_{com} = 500$ and $n_{com} = 1000$. The seed for the random



(a) CM with PCA ($n_{com} = 500$)



(b) CM with PCA ($n_{com} = 1000$)

Figure 6.6: Confusion matrices showing clustering results with different n_{com} values in PCA.

initialisation of K-means clustering is fixed for reproducibility. Notably, using K-means labels with PCA ($n_{com} = 2166$) as ground truth, PCA ($n_{com} = 500$) achieves 67.3% labelling correctness, and PCA ($n_{com} = 1000$) achieves 73.5%. Moreover, PCA with different n_{com} values for compressing feature maps leads to noticeable differences in cluster assignments, up to 30% of the labels being reassigned.

In conclusion, n_{com} and K_m are critical hyperparameters for PCA and K-means. Their choice significantly impacts clustering results, influencing the final selection of feature maps.

6.3.2 FedSplit Performance with Feature Maps Selection

The layer level in the network to extract feature maps is initially set to $l = low$. The number of PCA components is set to 200, reducing the dimension of activation maps from $16 \times 32 \times 32$ to just 200 after flattening. K-means clusters the PCA-compressed feature maps of each class into 20 groups by setting K_m to 20. By selecting the feature maps closest to the cluster centroid, each class in $\mathcal{D}_k$ contributes 20 representative samples towards the metadata $\mathcal{D}_{k,\, meta}$.

For comparison, we employ a baseline method using all feature maps to formulate the set of metadata without applying any data selection strategy. Table 6.2 reports the test accuracy of the composed model using the clustering-based data selection

Composed model	Data selection	Cluster number	Test accuracy (%)
WRN-40-1	w/o	na	70.03
	w/	20	48.47 $\downarrow$

Table 6.2: Test performance of the composed model in FedSplit. Each client selects 40 feature maps from layer level $l = low$, reducing the metadata to just 1.6% of its original size. The performance of FedSplit significantly decreases with the adoption of clustering-based data selection.

method versus the baseline. When the upper part of the composed model is trained on all 50,000 available feature maps without selection, it achieves a higher test accuracy of 70.03%. In contrast, when the size of $\mathcal{D}_{meta}$ is reduced to 800 by selecting feature maps, the test accuracy diminished significantly to 48.47%.

Further, Table 6.3 compares FedSplit using feature maps from different layer levels to train the composed model. In addition to the baseline without using feature maps selection, we include a hypothetical case where the WRN-40-1 model is trained on the entire CIFAR-10 training set in a centralised manner, as well as the ideal FedAvg assuming full client participation.

The results show narrowed performance gaps between FedSplit with feature maps selection and FedSplit without feature maps selection when $l = mid$ and $l = up$ are chosen. Particularly, with $l = up$ for feature maps extraction, the composed model performance decreases by less than 5%, even though the number of training instances is reduced by over 98% (selecting 800 feature maps from 50,000). However, with $l = mid$ for feature maps extraction, the performance gap enlarges to over 13%.

It is apparent that if all the feature maps are used for metadata training, FedSplit outperforms the ideal FedAvg when feature maps are extracted from layer level $l = mid$ and $l = low$, notably by over 4% with $l = mid$. This observation is consistent with FedAT and FedKAD, demonstrating that metadata consisting of feature maps can enhance knowledge transfer to the global model. On the other

Method	Test Accuracy (%)			
	Global	$l = low$	$l = mid$	$l = up$
Centralised	90.79	na	na	na
FedAvg	69.35	na	na	na
FedSplit (all meta)	na	70.03	73.47	68.55
FedSplit (selected meta)	na	48.47 ↓	60.11 ↓	64.20 ↓

Table 6.3: FedSplit using feature maps extracted from different layer levels. Missing values indicate inapplicable notations. WRN-40-1 is used as the global model. FedSplit outperforms the ideal FedAvg when all feature maps from either level *low* or *mid* are used. When using feature maps selected by K-means clustering, FedSplit shows the least performance drop.

hand, with $l = up$, the performance of FedSplit slightly falls behind the ideal FedAvg by 0.8%. This is because the knowledge carried by feature maps is only transferred to a small part of the composed model and the rest of the composed model can not benefit from metadata training.

In conclusion, FedSplit demonstrates consistent advantages of using metadata to transfer locally learned knowledge to the global model, similar to FedAT and FedKAD. Conversely, restricting the training of the composed model to only a few feature maps selected by K-means clustering significantly penalises its generalisation. Nevertheless, the performance gap resulting from the reduced feature maps can be mitigated by utilising feature maps extracted from higher layer levels of client models.

The second part of this chapter introduces transfer learning and entropy-based data selection to address the performance degradation caused by reduced training data in FL, realising the potential of active data selection in addressing the straggler issue.

6.4 Entropy-based Data Selection with Pretrained Global Model

Despite selecting representative training samples are selected, FedSplit shows compromised FL performance when significantly reducing the size of the training data. This section introduces entropy-based data selection to address the decrease in FL performance when fewer training instances are used for training client models. The proposed entropy-based data selection method leverages the transfer learning approach. Concretely, a unique pretraining phase is proposed to learn a generic feature extractor that can be shared across clients.

6.4.1 Pretraining a Global Model that Resists Model Shift

This section details the method of pretraining a global model prior to FL to aid the proposed entropy-based data selection during FL. We demonstrate that the pretrained global model is resistant to model shift with Centred Kernel Alignment (CKA) (Kornblith et al., 2019; Nguyen et al., 2020) analysis.

Concretely, the pretraining phase consisting of two steps. In the first step, the server trains the global model on a source domain closely related to the FL task, obtaining valuable knowledge from the source domain. The second step involves a one-round FL. During the one-round FL, the server does not impose a hard deadline for local updates on clients. There, the pretrained model acquired at step 1 is broadcast by the server to all participating clients in FL. Then each client trains the distributed model locally until convergence. The server allow sufficient time for client updated models to be returned, regardless of their computational capabilities.

Such FL strategy is studied in the one-shot[1] FL works (Guha et al., 2019; Gong et al., 2022), demonstrating its efficacy for FL. Finally, the server aggregates locally updated models using federated averaging to form the pretrained global model, which initiates the FL process by distributing it to the clients.

[1]The term of one-shot in this particular context means there is only one communication round in FL. All clients only update the distributed model once and upload them to the server without any further actions.

Method	Model	Pretraining	$Diri(0.1)$	$Diri(0.5)$
FedAvg	WRN-16-1	na	67.46	79.53
		CIFAR-100	70.46 ↑	80.70 ↑
		Small ImageNet	75.18 ↑	81.73 ↑

Table 6.4: The pretrained global model improves the performance of the downstream FL task significantly. Between the source domains experimented, Small ImageNet proves more effective than CIFAR-100.

Our preliminary experiments demonstrate that the proposed pretraining phase significantly boosts the performance of FL. Specifically, we conducted experiments on CIFAR-10 with 10 clients using FedAvg with full client participation. The global model employed is a WRN-16-1. To access the effectiveness, the global model is pretrained on two source domains separately: CIFAR-100 and Small ImageNet (Chrabaszcz et al., 2017), which are domains close to CIFAR-10. Small ImageNet is a downsampled version of the original ImageNet (Krizhevsky et al., 2012; Russakovsky et al., 2015). It contains over 1.28 million training samples with 1000 unique labels. The Small ImageNet downsampled to the size of 32×32 is used in the experiments.

Table 6.4 reports the test accuracy of the global model in FedAvg when the pretraining phase is adopted. A comparison with FedAvg without pretraining clearly shows that both pretraining on CIFAR-100 and Small ImageNet boost FedAvg performance by noticeable margins. Notably, pretraining on Small ImageNet displays greater efficacy in improving the performance compared to pretraining on CIFAR-100. This is attributed to the Small ImageNet being a more comprehensive dataset than CIFAR-100, exposing the global model with richer knowledge in image classification, which is transferable for learning on CIFAR-10.

Further, observing the performance differences under different levels of data heterogeneity, it is evident that the pretrained global model has more pronounced impact given strong data heterogeneity. It improves FedAvg by around 8% with $Diri(0.1)$, compared to a 2% improvement with $Diri(0.5)$. Our insight into this observation is the pretrained model is robust to model shift caused by heterogeneous client data.

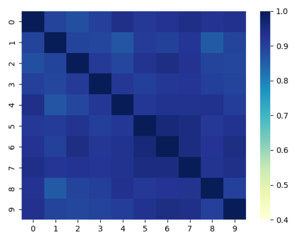
To validate this hypothesis, the Centred Kernel Alignment (CKA) (Kornblith et al., 2019; Nguyen et al., 2020) is adopted to study the feature similarity among pretrained models on heterogeneous client data. CKA is a widely used tool for accessing latent feature representations of neural networks, calculating similarity scores for features from different models. A higher CKA similarity score indicates that the compared models learn similar representations for the same instances.

As such, CKA similarity can serve as a metric to measure the magnitude of model shift. Specifically, during the proposed pretraining phase, the pertained global model is distributed to clients for local updates on heterogeneous data. Given conditions of strong data heterogeneity, locally updated models deviate from each other significantly, as described in Section 1.2. This deviation results in low CKA similarity, indicating these updated models generate highly distinct representations for the same instances.

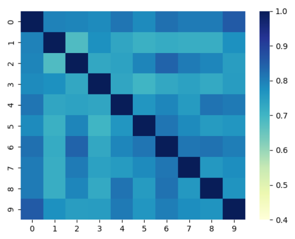
In the scenario with 10 clients set up earlier, we pair the locally trained models obtained from the one-round FL with all possible combinations and calculate their CKA similarity. The pretrained WRN-16-1 model on Small ImageNet is used for this analysis. The CKA similarity is measured at different layer levels of the WRN model on all instances in the CIFAR-10 test set. Similar to the model partition described in Section 5.3, CKA similarity is compared at levels $l \in \{low, mid, up\}$, respectively.

Figure 6.7 and Figure 6.8 visualise heatmaps of CKA similarity for the 10 locally trained models with data heterogeneity of $Diri(0.1)$ and $Diri(0.5)$. Entry (i, j) in the heatmap indicates the averaged CKA similarity score over CIFAR-10 test set between locally trained model i and locally trained model j.

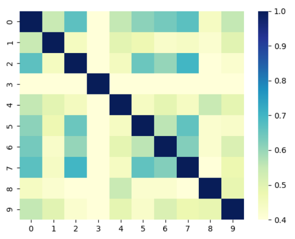
The similarities among locally trained models are notably increased when employing the pretrained model across all three layer levels, indicating successful mitigation of model shift. Figure 6.9 shows averaged CKA similarity scores in scenarios of $Diri(0.1)$ and $Diri(0.5)$. It is evident that the similarity gap between using the pretrained model and not using the pretrained model is more revealing in the case of $Diri(0.1)$ compared to $Diri(0.5)$. For instance, at the most deviated layer



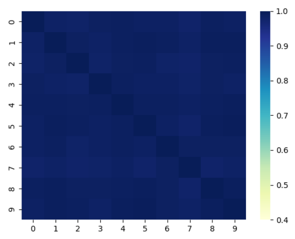
(a) no pretrain, layer low



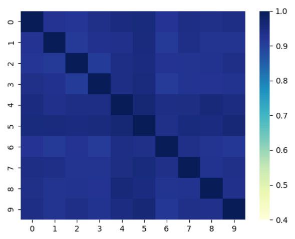
(b) no pretrain, layer mid



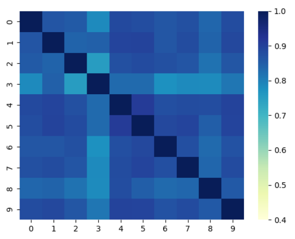
(c) no pretrain, layer up



(d) pretrain, layer low



(e) pretrain, layer mid

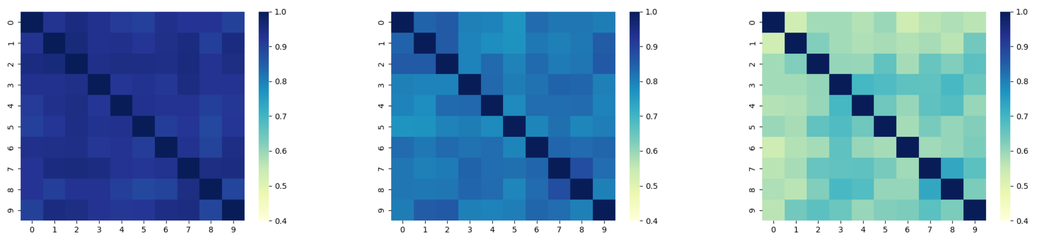


(f) pretrain, layer up

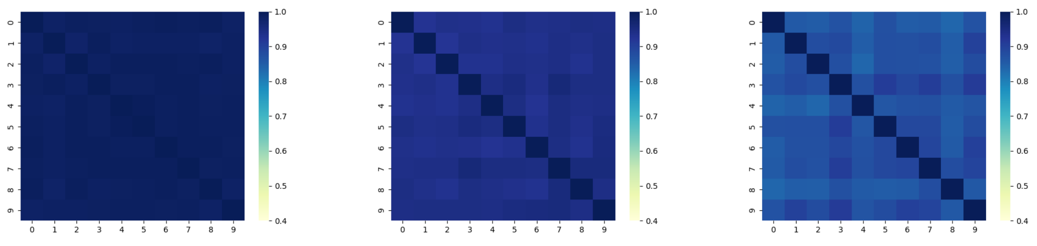
Figure 6.7: Heatmaps of the CKA similarity in the scenario of 10 clients and $Diri(0.1)$. A darker entry implies a higher similarity between the paired models indexed by the coordinate, suggesting they are less deviated from each other on heterogeneous data. The pretrained model resists model shift at all layer levels.

level, which is the *up* level, the pretrained model improves the overall similarity by over 0.4 given $Diri(0.1)$. In contrast, the improvement is less than 0.3 at the same level with $Diri(0.5)$. This observation suggests that the pretrained model can resist model shift more effectively under high levels of data heterogeneity, consistent with the results demonstrated in Table 6.4.

While pretraining on a source domain followed by one-round FL, we can initialise a robust global model resistant to model shift to start FL. However, if only the one-round FL is performed in the pretraining phase, the performance of the initialised global model is much weaker. Table 6.5 compares the performance of the initialised global model on CIFAR-10 when pretraining on Small ImageNet is included versus excluded in the pretraining phase. The results show a substantial improvement in the performance of the initialised global model , increasing from 10% to over 54% with the pretraining on on Small ImageNet and subsequently leading to overall better FL performance demonstrated in Table 6.4.



(a) no pretrain, layer low **(b)** no pretrain, layer mid **(c)** no pretrain, layer up

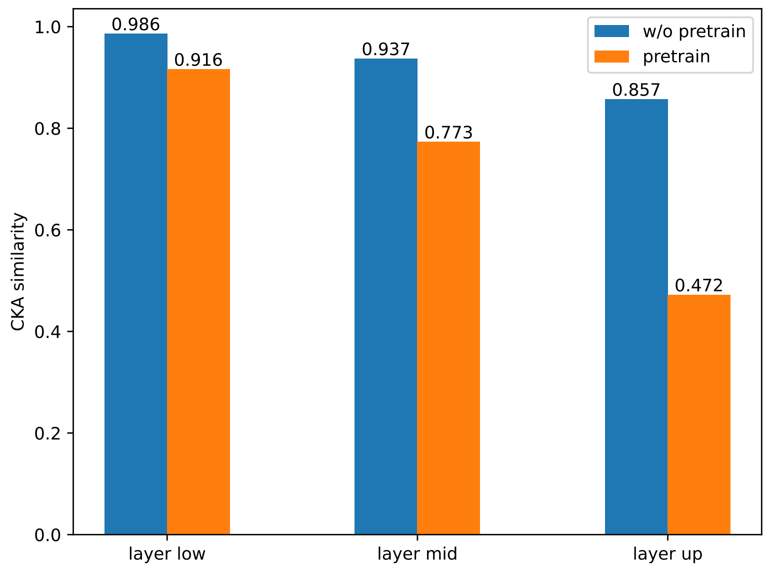


(d) pretrain, layer low **(e)** pretrain, layer mid **(f)** pretrain, layer up

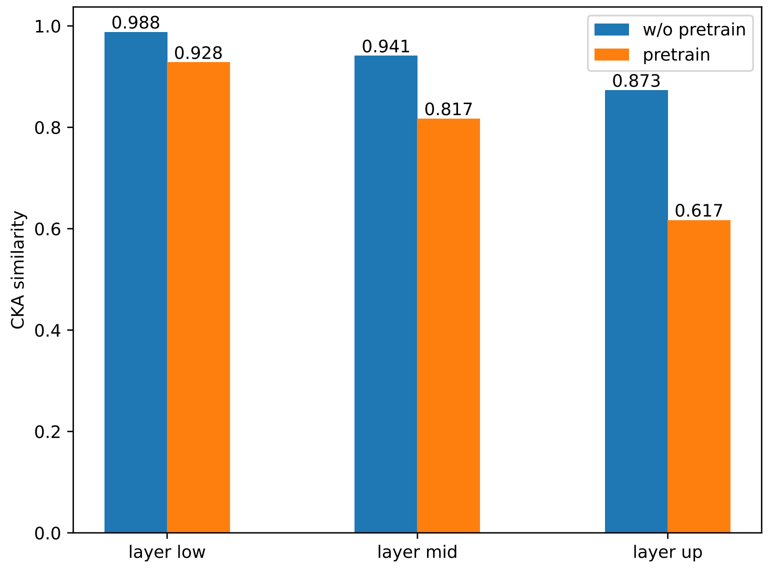
Figure 6.8: Heatmaps of the CKA similarity in the scenario of 10 clients and $Diri(0.5)$. Consistent with the observation in the $Diri(0.1)$ scenario, the pretrained model resists model shift across all layer levels.

Model	Pretraining	$Diri(0.1)$	$Diri(0.5)$
WRN-16-1	w/o	10.01	9.99
	w/	54.62 ↑	60.43 ↑

Table 6.5: The performance of the initialised global model is significantly improved if the pretraining on Small ImageNet is included in the pretraining phase.



(a) $Diri(0.1)$



(b) $Diri(0.5)$

Figure 6.9: Averaged CKA similarity at different layer levels of the locally trained models. The pretrained model displays greater robustness to model shift under conditions of stronger data heterogeneity.

6.4.2 Federated Fine-tuning with Entropy-based Data Selection

The adoption of the pretraining phase can not only yield an initialised global model resistant to model shift but can also substantially reduce the training time of local updates and assist the proposed entropy-based data selection. This section describes the algorithm of **Fed**erated **F**ine-**T**uning, abbreviated as FedFT, as well as **Fed**erated **F**ine-**T**uning with **E**ntropy-based **D**ata **S**election (FedFT-EDS), namely FedFT-EDS, which integrates entropy-based data selection and enables the data selection approach to address the straggler issue.

At a high level, clients in FedFT perform efficient fine-tuning by fixing the pretrained feature extractor obtained in the pretraining phase. FedFT-EDS leverages entropy information to actively select beneficial local instances, thereby improving the efficiency of fine-tuning and FL performance.

Preliminary: Entropy-based Active Data Selection

In addition to selecting the most representative training samples with clustering methods, we can alternatively select the most useful instances, as explored in previous studies on active learning (Beluch et al., 2018; Liu et al., 2021c; Yuan et al., 2021; Li et al., 2019a). This section proposes leveraging the Shannon entropy (WIKIPEDIA) to actively select those most useful instances for FL, reducing the training burden on stragglers through forming a small training set for local updates.

In information theory, Shannon entropy measures the uncertainty associated with a probability vector. While training a neural network, every instance sampled in the task distribution can be associated with a probability vector output by the softmax activation layer in the model. The probability vector indicates how confidently the model categorises the instance for the task. For example, when a sample is fed into the neural network, if the softmax activation layer in the network outputs a much greater value in the i-th element of the probability vector than the other elements, it suggests the model is highly certain in classifying this sample as the

i-th category.

Formally, given a possibility mass vector $P(X)$, produced by the softmax activation layer in neural networks for instance X, its Shannon entropy is calculated as follows:

$$H(X) = \mathbb{E}\left[-\log(P(X))\right] = -\sum_{i=1}^{n} p_i \log p_i \tag{6.6}$$

where p_i is the probability to label the instance X as the i-th class in the possible outcomes $\{c_1, c_2, \ldots, c_n\}$. Since the log function is concave, the Shannon entropy has an upper bound which is derived by the Jensen's inequality (Jensen, 1906) as follows:

$$H(X) = \mathbb{E}\left[-\log(P(X))\right] \leq \log\left(\mathbb{E}\left(\frac{1}{P(X)}\right)\right) = \log(n) \tag{6.7}$$

Take CIFAR-10 classification with $n = 10$ for instance, the maximal entropy for a probability vector output by the softmax activation in the neural network is $\log(10) \approx 2.30$. This is an extreme case where the model weights equally for all 10 classes, suggesting the model is completely uncertain about which class the instance belongs to. In this scenario, the model essentially labels the instance randomly with a uniform distribution. Hence, higher entropy associated with an instance signifies greater uncertainty in its classification by the model.

In active learning, instances that the model finds difficult (i.e., those with high entropy) to classify are considered more informative as they can provide new knowledge to improve the model (Settles, 2009). In contrast, instances that the model confidently classifies contribute relatively less new knowledge to the model. To this end, we select instances with the highest entropy to perform local updates on clients. Figure 6.10 a) illustrates the process to actively select the most useful instances for local updates based on entropy. To implement this method, the client model only needs to perform an additional forward pass on all client data to calculate the entropy associated with each instance. Instances are then ranked based on their entropy to identify the most useful ones. Unlike previous work such as FLRD (Nagalapatti et al., 2022), the proposed entropy-based data selection adds minimal computational overhead on clients, ensuring efficient local updates for resource-constrained devices.

Mitigate model shift with entropy-based data selection. On the other hand, we hypothesise that performing local updates on instances with the highest entropy can mitigate model shift. These selected samples not only expose the model with more informative knowledge about the learning task, but also alleviate inconsistencies between the global learning objective and local learning objectives caused by data heterogeneity in FL. Notably, consider a scenario where one client has significantly more data points with stronger label skew than others, the global model formulated at the server can become biased towards the dominant class in this client's data. However, when the entropy-based data selection is applied, the global model would output low entropy for the dominant class once it is offloaded to this client for local updates. Consequently, the local updates would avoid selecting samples in the dominant class because the model is already highly confident in classifying them correctly. Instead, samples from the underrepresented classes are selected for local updates due to their higher entropy. Therefore, using entropy-based data selection can help local updates avoid deviating the model from the global objective, which is ideally to generalise well across all possible classes rather than the dominant class on a single client.

Using hardened softmax to distinguish the most useful samples. The effectiveness of selected training samples depends on the ability to accurately distinguish those most uncertain samples using the entropy. However, a critical property of the Shannon entropy is that a small change in p_i in the probability vector results in only a small change in the entropy (Li et al., 2019a). This property can potentially limit the effectiveness of Shannon entropy in selecting those most uncertain instances. Intuitively, when p_i increases slightly, it suggests that the model slightly increases its confidence in classifying the instance to category i. Since the model learns less on this instance it is more confident about, it is preferable that a small increase in p_i leads to a significant entropy decrease, thereby excluding the instance

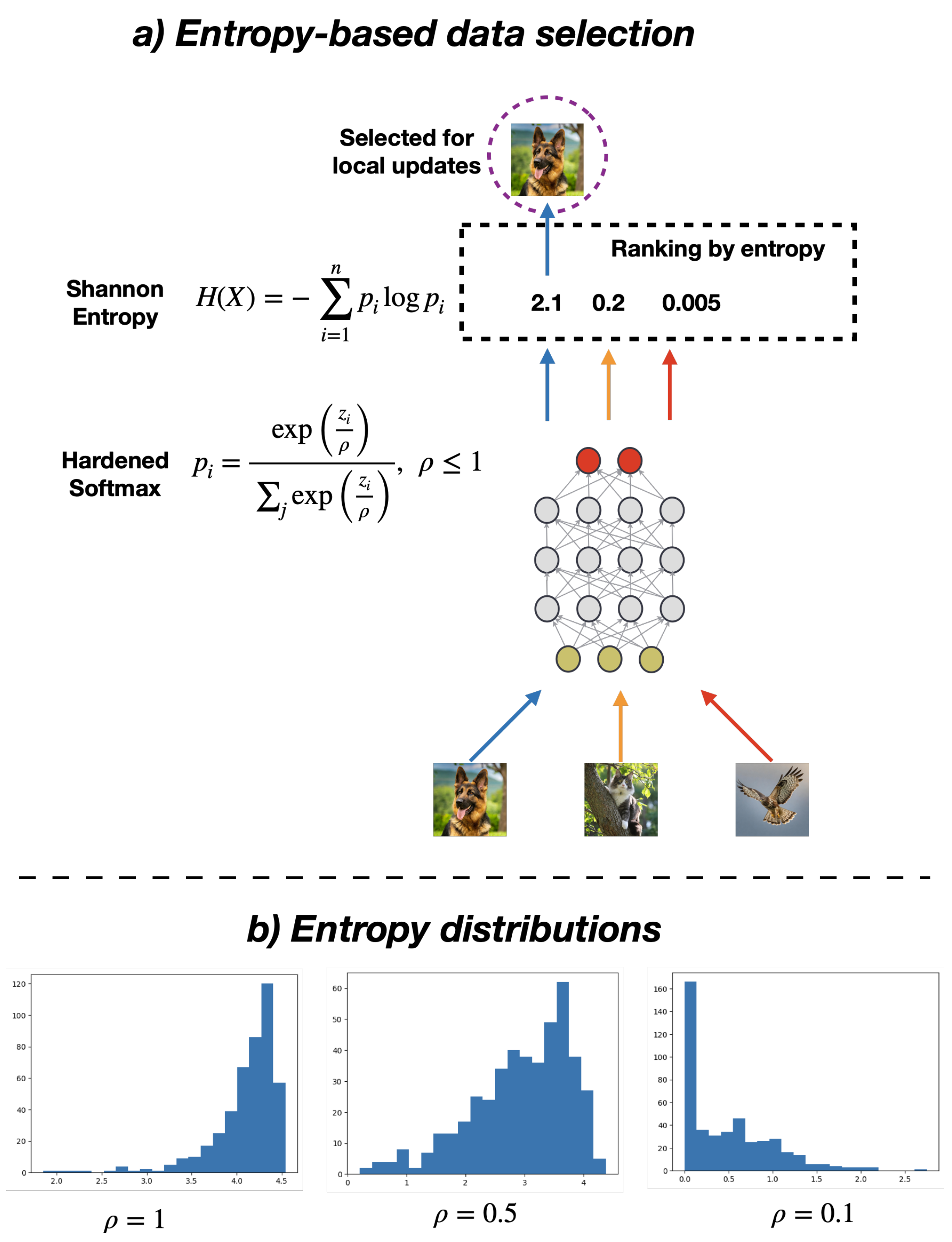

a) Entropy-based data selection
Selected for local updates
Ranking by entropy
2.1 0.2 0.005
Shannon Entropy
$H(X) = -\sum_{i=1}^{n} p_i \log p_i$
Hardened Softmax
$p_i = \frac{\exp\left(\frac{z_i}{\rho}\right)}{\sum_j \exp\left(\frac{z_i}{\rho}\right)}, \ \rho \leq 1$
b) Entropy distributions
$\rho = 1$
$\rho = 0.5$
$\rho = 0.1$

Figure 6.10: Entropy-based data selection: a) shows how the entropy is used to select the most useful instances for local updates. b) illustrates how the temperature in the introduced hardened softmax changes the entropy distribution.

from the training set for local updates. While we cannot alter the original Shannon entropy defined by Equation 6.6, we can reshape the probability distribution in the probability vector by amplifying the change in p_i. To this end, we propose using a hardened softmax activation to achieve this objective. Essentially, the hardened softmax is the softmax parameterised by a temperature value, as defined by Equation 5.1. The term "hardened" implies the temperature is set to a value less than 1 rather than greater than 1. Therefore, even a tiny change in z_i, the logit for the i-th class, can lead to a huge change to p_i, changing the entropy significantly.

The distribution of entropy shifts significantly when the temperature ρ is adjusted in the hardened softmax. Figure 6.10 b) visualises this shift by setting ρ to 1.0, 0.5 and 0.1. The entropy distribution is obtained by feeding a locally trained neural network with its local instances sampled from CIFAR-100. As ρ decreases to 0.1, it becomes obvious that a majority of the distribution shifts towards lower entropy values, forming a narrow tail in the high entropy region. Data points with the highest entropy are considered the most beneficial ones for learning and are selected to form the local training set. Conversely, as ρ approaches 1, the high entropy region becomes densely populated with data points, making it less effective to distinguish useful instances.

Local updates of FedFT-EDS

The pretraining phase proposed earlier provides FL with an well-initialised global model, including a generic feature extractor. Leveraging this feature extractor, FedFT-EDS can efficiently and strategically perform local updates. Particularly, local updates in FedFT-EDS comprises two steps, active data selection and partial model fine-tuning.

Active data selection with entropy. At communication round t, client k first downloads the global model as M_k^t, which is parameterised with w_k^t, it performs the entropy-based active data selection described in Section 6.4.2. Concretely, the client model is first fed with client data $x_k^{(i)} \in \mathcal{D}_k,\ i = 1, \cdots, |\mathcal{D}_k|$ to make predictions as follows:

Algorithm 6 FedFT-EDS: **Fed**erated **F**ine-**T**uning with **E**ntropy-based **D**ata **S**election.

```
1: Pretraining Phase: Pretrain the global model on the source domain and
   perform one-round FL, initialise the global model to $w_g^1 = \{\phi,\ \theta_g^1\}$, every
   client and the server keeps a copy of $\phi$.
2: Input: total $T$ rounds, $E$ local updates epochs, initialised $w_g^1$, total $N$ clients.
3: for $t = 1, \ldots, T$ do
4:     $K$ random clients are available for training and they download the upper
       part of the global model $\theta^t$ from the server.
5:     Clients:
6:     for Client $k \in [1, K]$ do
7:         $\mathcal{D}_{k,select}^t \leftarrow$ Data Selection $(;\theta^t, \phi, \mathcal{D}_k)$ according to Equation 6.8 and
           Equation 6.9.
8:         $\theta^{t+1} \leftarrow$ Local updates $\left(\theta^t;\ \phi, \mathcal{D}_{k,select}^t, E\right)$ according to Equation 6.10.
9:         Client $k$ uploads $\theta_k^{t+1}$ to the server.
10:    end for
11:    Server:
12:    Updates the upper part of the global model to $\theta_g^{t+1}$ with Equation 6.11.
13:    Form the global model $w_g^{t+1} = \{\phi,\ \theta_g^{t+1}\}$ to start the next iteration.
14:    Return Global model $w_g^t = \{\phi,\ \theta_g^t\}$
15: end for
16: Return Global model $w_g^T = \{\phi,\ \theta_g^T\}$
```

$$P_k^{t,(i)} = f_{w_k^t}(w_k^t; \rho,\ x_k^{(i)}) \tag{6.8}$$

where $P_k^{t,(i)}$ is the probability vector output by the softmax activation layer and ρ is the temperature in the softmax function. Recall the hardened softmax introduced in Section 6.4.2, a temperature value smaller than 1 is applied to adjust the values in $P_k^{t,(i)}$. Then the entropy associated with $x_k^{(i)}$ is calculated with Equation 6.9.

$$H_k^{t,(i)} = -\sum_{p_j \in P_k^{t,(i)}} p_j \log p_j \tag{6.9}$$

Finally, client k ranks the usefulness of $x_k^{(i)} \in \mathcal{D}_k,\ i = 1, \cdots, |\mathcal{D}_k|$ based on their entropy scores. Higher entropy scores indicate instances that are more challenging yet valuable for updating M_k^t. FedFT-EDS decides the number of selected instances arbitrarily, as detailed in the experiment section. Consequently, the client constructs a subset of local data, $\mathcal{D}_{k,select}^t$, containing these selected instances to update w_k^t.

Fine-tuning the upper part of the client model. In the second step, FedFT-EDS fine-tunes the upper part of M_k^t on $\mathcal{D}_{k,select}^t$ while keeping its feature extractor frozen. The parameters of the client model are denoted as $w_k^t = \{\phi,\ \theta_k^t\}$, where ϕ is the feature extractor in the lower part of the model and θ_k^t denotes the upper part. ϕ is not indexed with communication round t and client index k because it has been well trained with the initialised global model in the pretraining phase and each client keeps an unmodified copy of it. ϕ can effectively represent local data in the latent space without requiring further training in FL. Finally, on $\mathcal{D}_{k,select}^t$, θ_k^t is updated to θ_k^{t+1} for a total of E local epochs as follows:

$$\theta_k^{t+1} \leftarrow \theta_k^t - \lambda_l \nabla_{\theta_k^t} \ell_k(\theta_k^t; \phi, \mathcal{D}_{k,select}^t), \tag{6.10}$$

Client k uploads the training results of local updates–upper part of the model θ_k^{t+1}–to the server.

Global updates of FedFT-EDS

FedFT-EDS follows standard FedAvg to update the global model. Specifically, FedFT-EDS fuses the updated $\theta_k^{t+1}, k \in \{1, \ldots, K\}$ as follows:

$$\theta_g^{t+1} \leftarrow \sum_{k=1}^{K} p_k^t \theta_k^{t+1} \tag{6.11}$$

where, p_k^t is similar to the one defined in Equation 2.4, but is calculated based on the selected client data $p_k^t = \frac{|\mathcal{D}_{k,select}^t|}{|\mathcal{D}^t|}$ with $\mathcal{D}^t \triangleq \bigcup_{k \in [K]} \mathcal{D}_{k,select}^t$. The server constructs the global model for the next communication round $t+1$ with $w_g^{t+1} = \{\phi,\ \theta_g^{t+1}\}$ and distribute it to the client side.

As discussed in the previous section, since the feature extractor ϕ is not updated in the FL iterations, the server and clients only need to communicate the upper part of the global model, θ_g^t. Algorithm 6 describes the details of the proposed FedFT-EDS.

6.5 FedFT and FedFT-EDS Experiments

Special Experimental Setup. We evaluate FedFT-EDS on CIFAR-10, CIFAR-100 and GSC dataset. The global model is pretrained on the ImageNet Small 32×32 dataset for image classification before being pushed into the FL workflow. The number local update epochs E is set to 5. As usual, SGD optimiser with a learning rate of 0.1 and momentum of 0.5 is used for optimising the client model. We tune the temperature for the hardened softmax activation within the range {0.01, 0.05, 0.1, 0.5, 1} to achieve the best global model performance. The layer level to separate the training part and frozen part of the client model in FedFT-EDS is set to mid, as it results in the best global model performance in our experiments. Table 6.6 and Table 6.7 report the validation performance for tuning the temperature and layer level on CIFAR-10 and CIFAR-100. To reduce computational costs, we conduct a partial grid search: first varying the temperature with the layer level fixed at "mid", then fixing the temperature at 0.1 while exploring "lower", "mid" and "upper" layer levels. We find comparable and strong performance for temperatures at or below

Temperature in Softmax	0.01	0.05	0.1	0.5	1.0
Val. Acc. (%) on CIFAR-10	78.71	78.91	78.39	76.76	74.70
Val. Acc. (%) on CIFAR-100	52.64	52.58	52.74	49.10	46.15

Table 6.6: The validation performance on CIFAR-10 and CIFAR-100 by varying the temperature values for the hardened softmax activation adopted in FedFT-EDS. The layer level to split the client model is fixed to "mid" in this set of experiments.

Layer to Split the Model	Lower	Mid	Upper
Val. Acc. (%) on CIFAR-10	77.94	78.39	77.86
Val. Acc. (%) on CIFAR-100	51.00	52.74	52.36

Table 6.7: The validation performance on CIFAR-10 and CIFAR-100 by setting different layer levels to split the client model for the partial model training in FedFT-EDS. Temperature in the hardened softmax is set to 0.1 for this set of experiments.

0.1 (i.e., 0.01 and 0.05), and observe that "mid" layer level consistently performs best on both datasets.

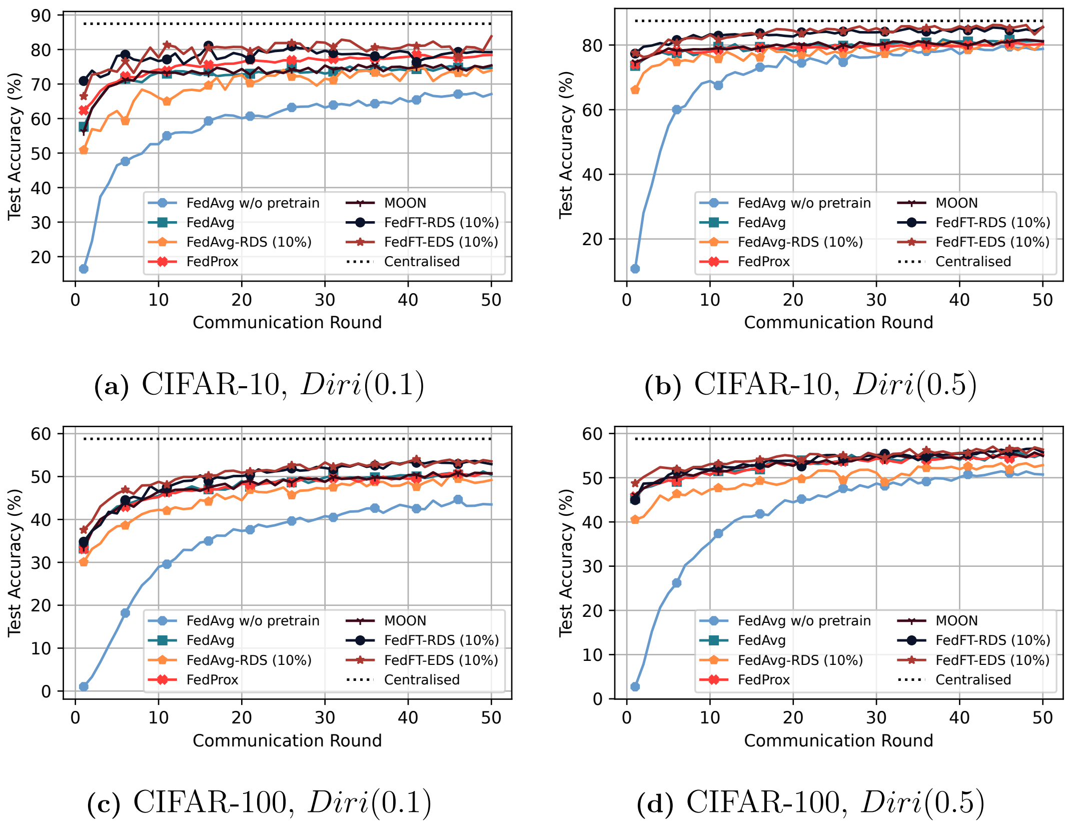
6.5.1 Evaluation on CIFAR-10 and CIFAR-100

FedFT-EDS is first evaluated on CIFAR-10 and CIFAR-100, which are in close domains to the pretraining dataset, Small ImageNet.

In a simple scenario where a total of 10 clients are assumed to be all non-stragglers, Table 6.8 compares the global model performance of FedFT-EDS with baselines. Additionally, the variant of FedFT with **R**andom **D**ata **S**election, denoted as FedFT-RDS, is also included for comparison. For FedFT-EDS, we select the local instances in proportion to the size of the entire local data, defined by P_{ds}, which is smaller or equal to 100%. When P_{ds} is set to 100%, it indicates no data selection is performed and all local data points will be used for local updates. In the baseline set, FedAvg w/o pret., FedAvg, FedAvg-RDS, FedProx and MOON are used for comparison. The FedAvg w/o pret. is the baseline without using the pretrained global model on Small ImageNet. FedAvg-RDS is the FedAvg baseline with

Method	P_{ds}	CIFAR-10		CIFAR-100	
		$\alpha = 0.1$	$\alpha = 0.5$	$\alpha = 0.1$	$\alpha = 0.5$
FedAvg w/o pret.	100%	67.46	79.53	44.66	51.44
FedAvg	100%	75.18	81.73	51.18	55.83
FedAvg-RDS	10%	75.05	81.37	50.22	53.27
FedProx	100%	78.48	80.96	50.80	55.43
MOON	100%	75.56	81.66	50.98	55.71
FedFT-RDS	10%	81.11	85.51	53.66	56.57
FedFT-EDS	10%	83.82 ↑	86.24 ↑	54.04 ↑	57.03 ↑
Centralised	100%	87.47		58.79	

Table 6.8: The global model performance of FedFT-EDS and baselines with 10 clients of full participation (%). P_{ds} is the proportion of local data selected for local updates. All methods, except FedAvg w/o pretraining, employ the proposed pretraining phase. "pret." is the abbreviation for pretraining. Centralised learning performance is attached for comparison. FedFT-EDS significantly narrows the performance gap between FL and centralised learning by up to 30% to 75% (in proportion to the performance differences between the strongest baseline and centralised training).



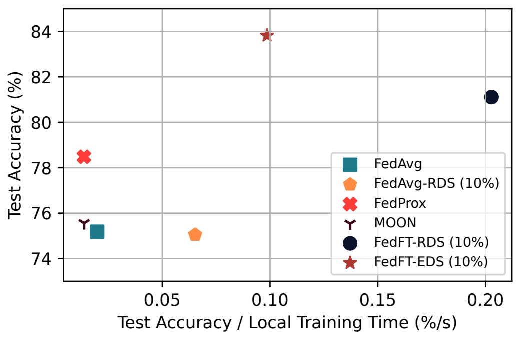
(a) CIFAR-10, $Diri(0.1)$ **(b)** CIFAR-10, $Diri(0.5)$

(c) CIFAR-100, $Diri(0.1)$ **(d)** CIFAR-100, $Diri(0.5)$

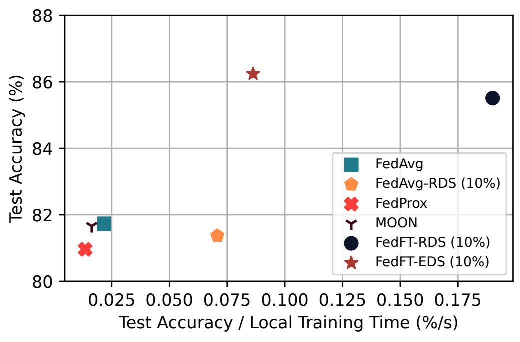
Figure 6.11: Learning curves of FedFT-EDS and baselines, showing test accuracy of the global model. The pretrained global model greatly accelerates FL convergence. FedFT outperforms all baselines, with FedFT-EDS slightly better than FedFT-RDS.

random local data selection. The rest baselines are their standard implementations with the adoption of the same pretrained model on Small ImageNet, without applying any active data selection. In addition, the centralised learning performance is reported to indicate the upper bound of the global model performance.

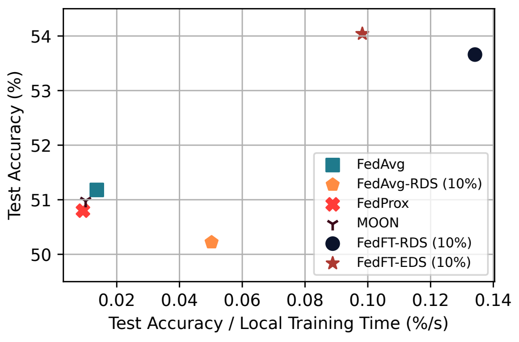
The experiment results are highly revealing in several ways. First, it is apparent that the introduction of the pretrained model can significantly improve the global performance on both datasets. The most substantial boost is witnessed in the scenario of a strong data heterogeneity, $Diri(0.1)$, consistent with the observation in Section 6.4.1. Second, both FedFT-RDS and FedFT-EDS outperform baselines by obvious margins, up to 5% on CIFAR-10 and up to 3% on CIFAR-100. They can significantly close the performance gap between FL and the centralised learning by 30% to 75%. The superior performance achieved by FedFT-RDS over FedAVG-RDS



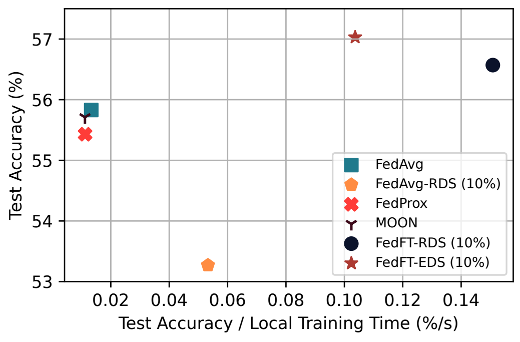
(a) CIFAR-10, $Diri(0.1)$



(b) CIFAR-10, $Diri(0.5)$



(c) CIFAR-100, $Diri(0.1)$



(d) CIFAR-100, $Diri(0.5)$

Figure 6.12: Comparison of the learning efficiency, calculated by dividing top test accuracy by total local training time. FedFT-EDS at least triples the learning efficiency of baselines and achieves the best global model performance in all cases, demonstrating significant improvement.

in the condition of both selecting 10% of local data demonstrates that fine-tuning the upper part of the pretrained model is more beneficial than updating it entirely. Finally, the entropy-based data selection displays its superiority over the naive random data selection. FedFT-EDS increases the performance of FedFT-RDS by 0.4% to 2.7%.

Further, Figure 6.11 illustrates the learning curves of FedFT-RDS, FedFT-EDS, and baselines over FL iterations. FedFT-RDS and FedFT-EDS clearly outperform all baselines, with faster convergence and better generalisation. Conversely, FedFT-EDS has a noticeable edge over FedFT-RDS in terms of both convergence and generalisation of the global model by actively sampling useful training instances with their entropy information. Additionally, leveraging the pretrained global model leads to significantly improved FL performance, as seen by comparing FedAvg without pretraining to other baselines using the pretrained model.

FedFT-RDS and FedFT-EDS are also observed with better learning efficiency. Learning efficiency is calculated by dividing the top test accuracy of the global model by the total local training time on all participating clients over FL iterations. It is a metric that measures how much global model performance is achieved per unit of local training time. Better learning efficiency implies that learning on clients is more energy efficient, requiring less training effort while contributing more to the global model performance. Figure 6.12 compares the chosen baselines with FedFT-RDS and FedFT-EDS. With only 10% of client data selected for learning, FedFT-EDS not only achieves the best absolute global model performance but also at least triples the learning efficiency of baselines, including FedAvg, FedProx and MOON, on CIFAR-10. The learning efficiency is further increased by 5 times on CIFAR-100. Although FedAvg-RDS has a learning efficiency close to that of FedFT-EDS, its global performance is substantially penalised by up to 10% on CIFAR-10 and 4% on CIFAR-100, disqualifying it as an practical FL solution. Regarding FedFT-RDS, it has a better learning efficiency than FedFT-EDS because the calculation of entropy introduces minimal computation overhead to clients in FedFT-EDS. However, FedFT-EDS achieves the best global model performance compared to all other methods, demonstrating the efficacy of entropy-based data selection.

The superior performance achieved by FedFT-EDS is even more pronounced in a realistic scenario involving 100 clients with straggler dropout simulation. Specifically, Table 6.9 compares the global model performance of FedFT-RDS, FedFT-EDS and FedAvg with varying client participation rates. The notation of f_n defined in Chapter 4 is reused to indicate the proportion of non-stragglers in the client pool. The remaining clients are assumed to be stragglers and are discarded by the server. f_n is set to 100%, 20%, 10% to simulate scenarios ranging from full client participation to low client participation.

Meanwhile, local updates in FedFT-RDS and FedFT-EDS are computationally efficient, as demonstrated in Figure 6.12. Hence, all clients are assumed to be non-stragglers in their setups. Besides selecting 10% local data for local updates, we additionally study the impact of increasing the number of selected instances by

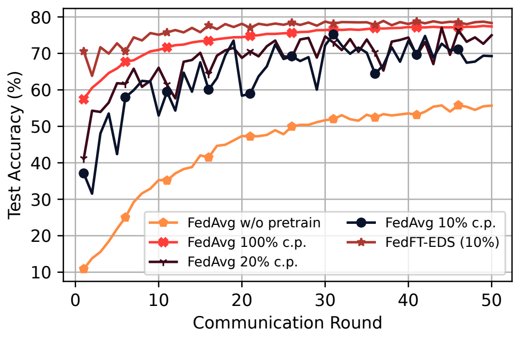
Method	f_n	P_{ds}	CIFAR-10		CIFAR-100	
			$\alpha = 0.1$	$\alpha = 0.5$	$\alpha = 0.1$	$\alpha = 0.5$
FedAvg w/o pret.	100%	100%	55.79	72.00	25.97	30.66
FedAvg	100%	100%	77.54	80.00	46.60	49.78
FedAvg	20%	100%	77.03	80.77	45.94	49.80
FedAvg	10%	100%	75.20	80.49	44.17	49.20
FedFT-RDS	100%	10%	78.20	80.25	47.64	50.23
FedFT-EDS	100%	10%	78.92 ↑	81.74 ↑	48.22 ↑	50.74 ↑
FedFT-ALL	100%	100%	78.96	81.26	50.39	53.23
FedFT-RDS	100%	50%	79.02	81.57	50.51	53.33
FedFT-EDS	100%	50%	79.80 ↑	82.46 ↑	51.54 ↑	54.22 ↑

Table 6.9: 100 clients scenario with straggler simulations. All variants of FedFT assume full client participation as their partial model fine-tuning are highly efficient. FedFT-EDS remains the best method in this scenario and its advantageous performance is further enhanced when FedAvg is penalised by the straggler issue. A critical finding is not all local data is beneficial for federated learning as FedFT-EDS with 50% outperforms FedFT-ALL.

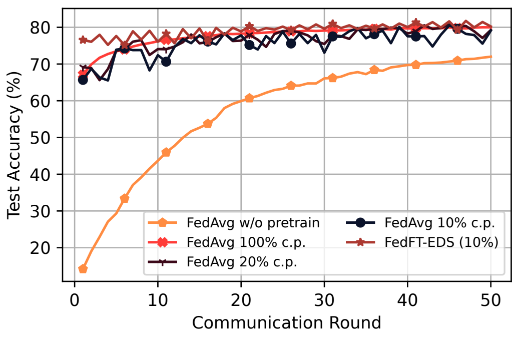
setting P_{ds} to 50% and 100%. In the special case where $P_{ds} = 100\%$, data selection is trivial as all local instances are used for local updates. Therefore, this setup is denoted as FedFT-ALL in Table 6.9. The performance of FedAvg without using the pretrained model is also included in the Table to demonstrate the advantage of pretraining.

Table 6.9 demonstrates that FedFT-RDS and FedFT-EDS consistently outperform FedAvg with full client participation in the scenario involving a larger client pool. When practical straggler dropout is considered in FedAvg, their performance gap is further enlarged to over 4% to 7%, particularly under a strong data heterogeneity with $Diri(0.1)$. The significant performance boost achieved by FedFT-RDS and FedFT-EDS underscores their advantages in allowing contributions from stragglers. Once again, FedFT-EDS outperforms FedFT-RDS in all scenarios, highlighting the efficacy of entropy-based data selection in improving global model performance. However, the improvement from FedFT-RDS to FedFT-EDS is marginal. Our insight on this observation is that the benefit from learning on hard client samples diminishes as the global model has repeatedly learned from these hard samples. Consequently, Fed-RDS can catch up the performance of Fed-EDS at later FL rounds. This hypothesis is supported by the learning curves shown in Figure 6.14, as we observe that the performance of Fed-RDS starts closing to that of Fed-EDS. Although the absolute improvement made by Fed-EDS is limited, Fed-EDS demonstrates its effectiveness in accelerating learning. At early FL rounds, Fed-EDS is evident with faster performance growth compared to Fed-RDS, making FL less demanding on resource-constrained devices for reaching a desired performance level.

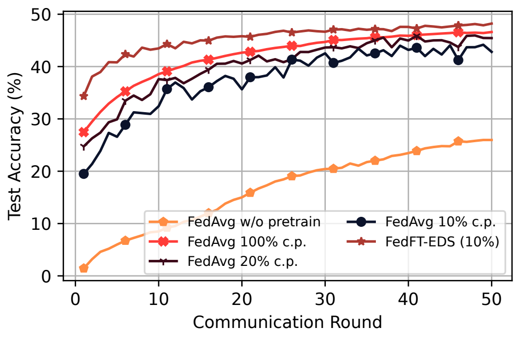
Compared to selecting 10% of local instances, selecting 50% of local instances increases the performance of FedFT-RDS and FedFT-EDS by around 1% on CIFAR-10 and 3.5% on CIFAR-100. Finally, by comparing FedFT-ALL and FedFT-EDS selecting 50% of local data, the most insightful conclusion from Table 6.9 is not all client data is beneficial for the global model performance. Training the client model on half of the local data actively selected based on entropy, FedFT-EDS outperforms FedFT-ALL by around 1% in all cases. The result suggests that including the other



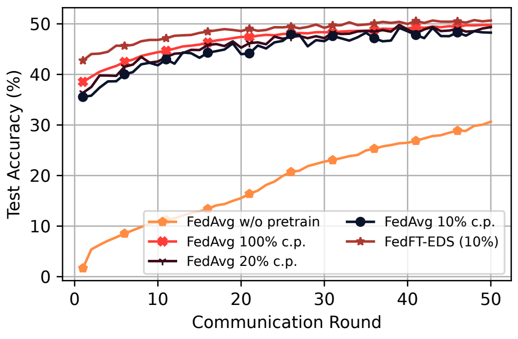
(a) CIFAR-10, $Diri(0.1)$



(b) CIFAR-10, $Diri(0.5)$



(c) CIFAR-100, $Diri(0.1)$

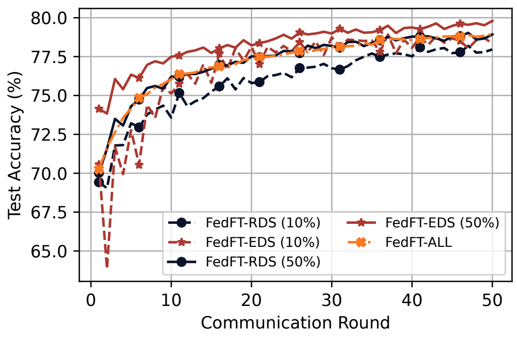


(d) CIFAR-100, $Diri(0.5)$

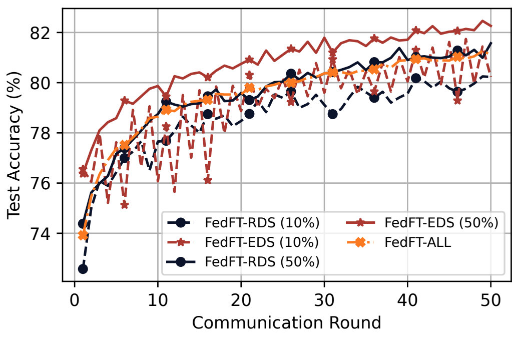
Figure 6.13: Learning curves of FedFT-EDS and baselines. FedFT-EDS consistently outperforms all baselines throughout FL iterations, with a particularly notable advantage over FedFT-RDS in the 100-client setup, demonstrating the scalability of entropy-based data selection in FL. Also, leveraging the pretrained global model greatly accelerates FL convergence.

half of local data, which is pruned by entropy, degrades global model performance. As discussed in Section 6.4.2, training client models on selected local data with high entropy can mitigate the model drift on heterogeneous data, whereas using all local data indiscriminately to update client models aggravates the model shift. This telling observation underscores the significance of employing an effective data selection method to actively identify useful local training instances in heterogeneous FL. However, our observation that half of the local data is non-beneficial for FL is limited to our chosen experimental scenarios, such as datasets, heterogeneity simulation, and etc. We anticipate the proportion of non-beneficial data could vary in different FL setups.

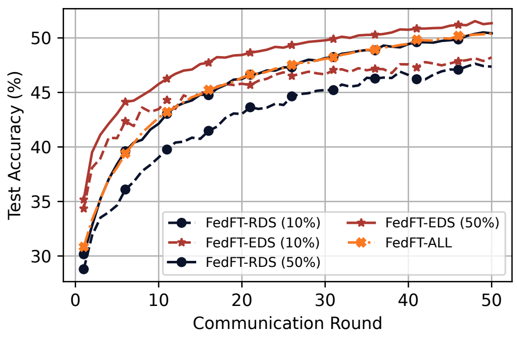
Figure 6.13 illustrates the learning curves of FedFT-EDS and FedAvg baselines



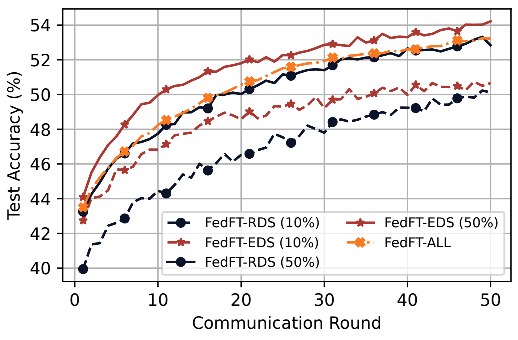
(a) CIFAR-10, $Diri(0.1)$



(b) CIFAR-10, $Diri(0.5)$



(c) CIFAR-100, $Diri(0.1)$

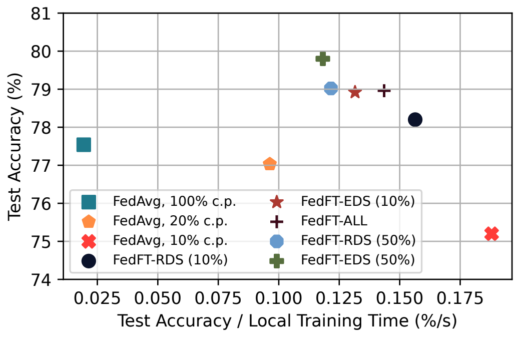


(d) CIFAR-100, $Diri(0.5)$

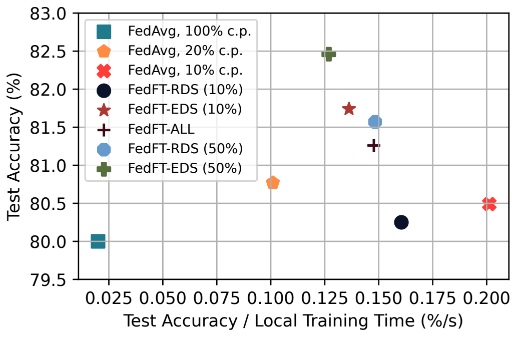
Figure 6.14: Comparison of FedFT-RDS and FedFT-EDS with 10% or 50% local data selection for local updates. In both cases, FedFT-EDS outperforms FedFT-RDS significantly in convergence and top test accuracy. The performance of FedFT-ALL falls behind FedFT-EDS (50%) by a large margin, indicating not all local instances are beneficial for FL.

in the scenario with 100 clients. Compared to the 10-client scenario, FedFT-EDS achieves a more substantial improvement in both convergence and top test accuracy of the global model, demonstrating the adaptability of entropy-based data selection in heterogeneous FL at scale. Figure 6.14 compares the learning curves of FedFT-RDS and FedFT-EDS with different volumes of selected data. It is apparent that FedFT-EDS greatly accelerates FL convergence and improves global model performance compared with FedFT-RDS.

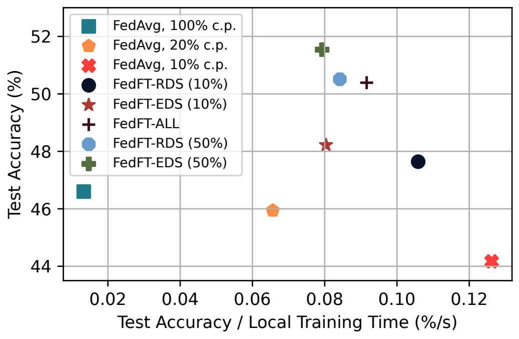
Finally, Figure 6.15 compares the learning efficiency achieved by FedFT-EDS and the baselines in the scenario with 100 clients. FedFT-EDS consistently exhibits superior learning efficiency while achieving the best global model performance, demonstrating entropy-based data selection effectively addresses the straggler issue without incurring performance penalisation. Figure 6.15 further shows that



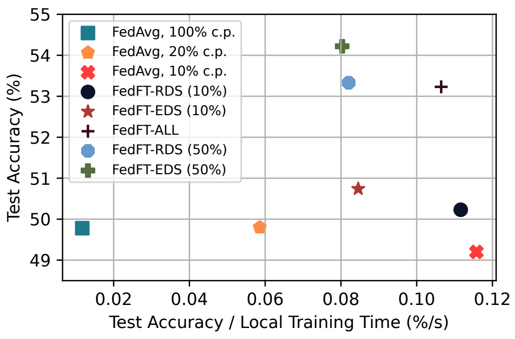
(a) CIFAR-10, $Diri(0.1)$



(b) CIFAR-10, $Diri(0.5)$



(c) CIFAR-100, $Diri(0.1)$



(d) CIFAR-100, $Diri(0.5)$

Figure 6.15: The learning efficiency of FedFT-RDS, FedFT-EDS, FedFT-ALL, and the baselines in the 100-client scenario. FedFT-EDS (50%) trades a small amount of learning efficiency to achieve the best global model performance. While FedAvg achieves the best learning efficiency with 10% client participation, its global model performance is significantly compromised by the straggler issue.

FedFT-EDS (50%) slightly sacrifices efficiency to boost the test accuracy of the global model by up to 3.5%. This observation indicates that FedFT-EDS is only marginally penalised in learning efficiency even when up to 50% of local data is selected for local updates. This is because the main reduction in local training time in FedFT-EDS comes from partial model fine-tuning, with the reduction in training data playing a lesser role. However, including more training data leads to improved global model performance.

In contrast, FedAvg, despite showing the best learning efficiency with only 10% of clients as non-stragglers, suffers significantly in global model performance due to the straggler issue.

Method	P_{ds}	ImageNet	GSC valid
Centralised w/o pret.	100%	91.22	
Centralised w/ pret.	100%	89.81	91.18
FedAvg w/o pret.	100%	41.33	
FedAvg w/ pret.	100%	55.40	86.31
FedFT-RDS	10%	54.96	83.67
FedFT-EDS	10%	55.60 ↑	84.39 ↑
FedFT-RDS	50%	55.56	86.08
FedFT-EDS	50%	59.32 ↑	86.47 ↑

Table 6.10: The test performance of FedFT on GSC using Small ImageNet and GSC validation set for the pretraining phase. The pretrained model from Small ImageNet and GSC validation set is applied to the GSC learning task separately. The former case creates a strong domain shift for FedFT, whereas the latter case minimises the domain shift. FedFT is still effective even in the strong domain shift case. However, it is preferred to employ a model pretrained on a similar domain to the downstream FL task. The entropy data selection is more effective for improving the model performance than random data selection regardless of the domain shift.

6.5.2 Evaluation on GSC

With respect to the adopted pretraining method, the evaluation of FedFT-EDS on CIFAR-10 and CIFAR-100 focuses on scenarios involving closely related source and target domains. This section extends the evaluation to GSC, a significantly different domain from the source. The experiment sets the total client number to 100 and partitions client data with strong data heterogeneity with $Diri(0.1)$. All clients are assumed to be non-stragglers in the simulation. For comparison, a pretraining phase is additionally conducted on the GSC validation set, simulating minimal domain shift. This setup complies with the privacy-preserving principle of FL, as clients only hold data from the GSC training set, ensuring the global model remains unexposed to client data prior to FL.

Table 6.10 reports the test performance of centralised training, FedAvg without pretraining, FedAvg with pretraining, FedFT-RDS, and FedFT-EDS. In centralised learning, using a model pretrained on Small ImageNet results in a slight decrease in performance, indicating limited transferability of ImageNet knowledge to GSC in this context. However, FedAvg benefits significantly from Small ImageNet pretraining, improving the text accuracy of the global model by over 14%. However, pretraining on the GSC validation set boosts the performance by over 45%. This observation suggests: 1) Pretraining the global model on a different source domain can still improve FL performance. 2) A reduced domain shift between pretraining and FL can lead to a substantial performance improvement.

Similar to FedAvg, although FedFT-RDS and FedFT-RDS demonstrate their performance advantages in this case with a significant domain shift, the results suggest the importance of employing a source domain closer to the FL domain whenever possible. Finally, FedFT-EDS is evident to consistently outperform FedFT-RDS regardless of the source domain utilised, showing the effectiveness of the entropy-based data selection independent of domain shift. Specifically, when the GSC validation set is used for pretraining, FedFT-EDS achieves the best global performance with 50% of local data selected, outperforming the FedAvg baseline using all local data for training. Again, this observation implies that not all local instances are beneficial for FL.

6.6 Summary

This chapter advances active data selection methods for addressing the straggler issue. Two primary goals are approached: 1) The introduced active data selection methods should be computationally cheap for clients to perform. 2) FL can preserve or even improve its performance when less data is used for local updates. To achieve these goals, two efficient data selection methods, clustering-based data selection and entropy-based data selection, are introduced and explored.

The first part of this chapter introduces a split FL paradigm for clustering-based data selection. This paradigm splits the global model into a lower part that is trained with standard FedAvg, and an upper part of the model that is trained on the server side using selected feature maps that are representative for local data. Representative feature maps are selected using K-means clustering from the latent space of client models. The selected feature maps are then uploaded by clients to the server for training the upper part of a composed global model. The experiments show that with just 1.6% of the feature maps, the composed global model preserves most of its generalisation capability, decreasing by just 4.35% on CIFAR-10.

FedFT-EDS is proposed in the second part of this chapter to completely eliminate performance penalties resulting from reduced training data. FedFT-EDS adopts two key ingredients to achieve this objective, a unique pretraining phase to initialise the global model and an entropy-based client data selection method. CKA analysis shows that the pretraining can initialise a global model that resists model shift in downstream FL tasks. On the other hand, the entropy-based data selection designs an importance ranking method for local data with Shannon entropy and harden softmax activation to effectively identify useful instances for the learning task at minimal computational cost.

On CIFAR-10 and CIFAR-100, FedFT-EDS is observed to consistently improve FL performance over chosen baselines in various scenarios, tripling the learning efficiency and boosting test accuracy of the global model by up to 6%. FedFT-EDS effectively enhances FL performance rather than penalises it while fewer training instances are selected for learning. Most importantly, FedFT-EDS demonstrates that over half of local instances are detrimental to FL performance. In a drastic domain shift scenario where GSC is the target domain, FedFT-EDS retains its efficacy in improving FL performance.

In summary, the development of FedFT-EDS successfully addresses the research question derived for the approach of active data selection. Active data selection leveraging pretraining and entropy information to select the most beneficial learning instances significantly: 1) improves the learning efficiency of FL, thereby addressing the straggler issue; 2) enhances the global model performance rather than compromising it, even with substantially reduced training data.

By bridging the research gap in the approach of data selection in FL, Chapter 6 is the final chapter of the three technical chapters that aims to address the straggler issue. In the next chapter, we will summarise our work and propose future research directions for encouraging resource-constrained devices to participate in FL.

Chapter 7

Conclusions

7.1 Summary of Thesis

This thesis conducts substantial research to improve the participation of resource-constrained devices in Federated Learning (FL) through advancing efficient machine learning approaches. Through comprehensive literature review, this thesis identifies critical research gaps existing in partial model training, custom-size client models, and active data selection–three efficient machine learning approaches that are commonly employed in FL to address the straggler issue. These research gaps are unmitigated model shift, weak knowledge transfer with a single knowledge source, and understudied active data selection. To bridge these research gaps, this thesis proposes and evaluates novel methods: few-shot fine-tuning, attention transfer and metadata training, clustering-based data selection and entropy-based data selection. This research contributes valuable novelties and insights that can inspire future FL studies aiming to encourage more resource-constrained devices to contribute their knowledge to FL. They are summarised as follows.

FedFSC, the proposed FL algorithm adopting few-shot fine-tuning, bridges the research gap of unmitigated model shift in the partial model training approach. FedFSC allows stragglers to perform few-shot updates, assisted by a feature extractor learned on non-stragglers. It demonstrates that few-shot updates can reduce the training time on stragglers by 90% and result in a less biased classifier on heterogeneous data compared to traditional local updates. With these profound insights, FedFSC alleviates training efforts on stragglers and mitigates model shift simultaneously, by constructing the global model with the few-shot updated classifier contributed by stragglers. Extensive experiments show that FedFSC outperforms popular FL baselines by significant margins. FedFSC advances the state-of-the-art research by allowing partial model training to not only improve the participation of stragglers but also mitigate model shift, accelerating FL convergence and reducing straggler energy consumption.

FedAT and FedKAD, the two proposed FL algorithms that adopt attention transfer and metadata training, address the research gap in the approach of custom-size client models, which is the weak knowledge transfer with a single knowledge source. FedAT leverages attention maps and metadata as knowledge sources to transfer locally learned knowledge from custom-size client models to the global model. We demonstrate that AT and metadata training successfully enable the global model to generalise on unseen classes and prevent the global model from overfitting to generic dataset during knowledge transfer. Further, FedKAD addresses the negative knowledge transfer problem induced by model shift by regulating knowledge distillation with prototype feature maps–metadata in a compressed form. In strong data heterogeneity environments, FedKAD outperforms FedMD, another FL baseline allowing custom-size client models but using a single knowledge source for knowledge transfer. The results show that prototype feature maps convey essential intra-class knowledge to improve the quality of knowledge transfer. FedAT and FedKAD contribute to the approach of custom-size client models by proposing novel knowledge sources to enhance knowledge transfer and proving their efficacy with improved global model performance. The application of custom-size client models for addressing the straggler issue is significantly improved with contributions from FedAT and FedKAD.

Finally, the proposed FedSplit and FedFT-EDS, algorithms adopting clustering-based data selection and entropy-based data selection respectively, enrich the research of leveraging active data selection to address the straggler issue. FedSplit uses K-means clustering method to formulate a small subset of training data consisting of most representative feature maps on clients to refine the global model on the server. It is observed that the global model performance is slightly penalised when the size of training data is significantly reduced. FedFT-EDS is proposed to address the performance penalisation for active data selection exposed by FedSplit. FedFT-EDS first introduces a unique pretraining phase that can initialise a global model resistant to model shift. Shannon entropy calculated with the hardened softmax is proposed to identify the most beneficial client instances for learning. FedFT-EDS shows that it can not only reduce workloads for stragglers but also improve FL performance in terms of both convergence and global model performance. FedSplit and FedFT-EDS broaden the research of active data selection in FL, an area that has been understudied for its potential to address the straggler issue. This work successfully demonstrates that active data selection can effectively address the straggler issue.

The three efficient machine learning approaches studied in this thesis can be adopted simultaneously to further enhance their utilities for reducing the workloads for resource-constrained devices. Take our proposed FedFT-EDS for instance, it is a such method that employs the partial model training and active data selection, where participating devices only need to update a subset of model parameters on reduced training data. FedFT-EDS is observed to improve the federated learning efficiency by at least three times comparing to the baselines in our experiments, demonstrating the advantages of leveraging the approaches of both training parameter reduction and training data reduction.

7.2 Limitations and Future Research

7.2.1 Limitations

- A major limitation of FedFSC is that it cannot mitigate the model shift in the feature extractor. Therefore, the global model performance is limited by the quality of feature extractor trained on non-stragglers. This limitation is indicated by the observation that as fewer non-stragglers contribute to learning the feature extractor, performance of FedFSC gets worse.

- A critical problem of FedAT and FedKAD is that metadata adds communication overhead to FL while it is introduced as an additional knowledge source for improving knowledge transfer. This increased overhead raises the expenses related to storage, connectivity, bandwidth, etc., for stragglers.
- The performance of FedFT-EDS is highly sensitive to the tuning of the temperature ρ in the hardened softmax. When this hyperparameter is not optimally tuned, it is observed that the performance of FedFT-EDS could even fall behind that of the chosen baselines.

7.2.2 Future Research

This research can inspire future works that aims to encourage resource-constrained devices to participate in FL. The following are a few directions that are worthy to explore starting from this thesis:

- **One-round FL.** This thesis has demonstrates the benefits of accelerating FL convergence to improve learning efficiency. An extreme case is FL convergences within a single communication round. This research direction has been explored by a handful of previous works (Guha et al., 2019; Zhou et al., 2020b; Gong et al., 2021; Zhang et al., 2022a; Su et al., 2023), dubbed one-round FL or one-shot FL. In one-round FL, the server collects updated client models from participants once for all. With this strategy, there is no need for the server to set a deadline for local updates in order to start the next communication round. As such, one-round FL offers resource-constrained devices with maximal computing tolerance. However, the challenge of one-round FL is how to effectively optimise the global model to acquire all of the learned knowledge from clients, particularly in the presence of model shift induced by data heterogeneity. Although substantial research endeavours have been made, such as DOSFL (Zhou et al., 2020b) that leverages KD, FedPCL (Tan et al., 2022b) that employs pretrained models, FedDISC (Yang et al., 2024) that generates synthetic dataset, one-round FL remains largely underperforms the traditional multi-round FL.

- **Model quantisation.** Besides knowledge distillation discussed in this thesis, another popular approach for compressing machine learning models is quantisation. Model quantisation not only saves the memory for storing models but also reduces the computational overhead of model training by reducing the number of bits of model parameters (Liu et al., 2021a). Recent research (Liu et al., 2021a; Thakur et al., 2024) demonstrates that quantised models save a significant among of energy and time while being trained on the edge. However, quantising client models often leads to a tradeoff in the global model performance. Hence, there is a need to mitigate this tradeoff for the model quantisation approach in FL. We can further explore the combination of model quantisation and knowledge distillation for compressing client models while achieving competitive model performance.

- **Pretraining and fine-tuning large models.** Recently, large models, particularly Foundation Models (FM), including stable diffusion models and LLMs, which are commonly built with stacks of Transforms, are the driving forces of some of the most groundbreaking AI applications, such as ChatGPT and SORA. FL has a great potential to provide sufficient data for training these large models (Babakniya et al., 2023; Sani et al., 2024). However, a large number of the edge devices will be deemed resource-constrained and become stragglers when dealing with large models. Hench, there is a great need to mitigate this bottleneck so that large models can leverage a wealth of user-generated data on various devices. An increasing number of FL works are proposed to approach this challenge. They utilise various efficient model tuning techniques, such as prompt tuning (Che et al., 2023), parameter-efficient fine-tuning (PEFT) (Jiang et al., 2023), instruction tuning (Zhang et al., 2024), etc., to alleviate training efforts on FL participants. With the rapidly growing data need for learning large models, we can envision that adapting large model training to FL through efficient on-device learning will be a hot research topic in near future.

- **Addressing client data challenges.** FL faces significant data challenges when training on a large number of diverse clients. Notably, client data could be unlabeled, non-stationary, and maliciously poisoned, potentially hindering the capability and scalability of FL (Ji et al., 2024). As such, it is becoming increasingly important to advance FL to tackle these data challenges, ensuing the success of FL in the wild. On one hand, unsupervised FL (Zhuang et al., 2021; Han et al., 2022; Zhuang et al., 2022; Rehman et al., 2023) and lifelong FL (Yu et al., 2022; Zhang et al., 2023b; Jin et al., 2023; Wang et al., 2024b) are in the frontiers of FL research for dealing with unlabeled data and non-stationary data in various machine learning fields, such as computer vision, robots, IoTs, etc. On the other hand, techniques of protecting FL from adversarial attacks (Roy Chowdhury et al., 2022; Shejwalkar et al., 2022; Kumar et al., 2023) has gained a wider research attention recently for their utilities of preserving the integrity of client data in practice. In summary, data heterogeneity only represents one aspect of data challenges that FL is facing. In the further, we can project that the FL research community will shift its concentration to tackle practical data challenges that could be more significant for production FL systems.

- **Novel FL paradigms with confidential cloud computation.** FL is evolving fast. Recently, novel FL paradigms that allow clients to upload their data to powerful and secured computing nodes on the cloud have been explored (Daly et al., 2024). Such FL paradigms could eliminate the straggler issue by equipping the computing nodes with powerful hardware, further enabling the potential of FL for training large models. However, novel security measures are needed in place to ensure that user privacy is not rendered. Particularly, researchers in Google recently propose the Trusted Execution Environments (TEEs) (Eichner et al., 2024) for FL to establish the client trust of uploading their encrypted data to the cloud. TEEs leverage public key infrastructure and ledger service to allow clients to verify and approve privacy-preserving workloads requested from the server, providing clients with full control of how their data is used during FL. With these works, we are witnessing the transformation of FL paradigms, offering FL with more flexible ways of leveraging decentralised data for training machine learning models.

This thesis paves on three avenues of FL optimisations, yet other optimisations could be equally impactful. No matter which routes or methods will become popular in future, there is a stringent need for FL optimisation to eliminate stragglers. With a growing scope of FL to extract knowledge from data silos on the edge, enabling the participation of all clients, irrespective of their computational resources, becomes paramount.

Bibliography

Ahmed M Abdelmoniem, Chen-Yu Ho, Pantelis Papageorgiou, and Marco Canini. A comprehensive empirical study of heterogeneity in federated learning. *IEEE Internet of Things Journal*, 2023.

Durmus Alp Emre Acar, Yue Zhao, Ramon Matas Navarro, Matthew Mattina, Paul N Whatmough, and Venkatesh Saligrama. Federated learning based on dynamic regularization. *arXiv preprint arXiv:2111.04263*, 2021.

Charu C Aggarwal, Xiangnan Kong, Quanquan Gu, Jiawei Han, and S Yu Philip. Active learning: A survey. In *Data classification*, pages 599–634. Chapman and Hall/CRC, 2014.

Jin-Hyun Ahn, Yeeun Ma, Seoyun Park, and Cheolwoo You. Federated active learning (f-al): an efficient annotation strategy for federated learning. *IEEE Access*, 2024.

Samiul Alam, Luyang Liu, Ming Yan, and Mi Zhang. Fedrolex: Model-heterogeneous federated learning with rolling sub-model extraction. *Advances in Neural Information Processing Systems*, 35:29677–29690, 2022.

Laith Alzubaidi, Muthana Al-Amidie, Ahmed Al-Asadi, Amjad J Humaidi, Omran Al-Shamma, Mohammed A Fadhel, Jinglan Zhang, Jesus Santamaría, and Ye Duan. Novel transfer learning approach for medical imaging with limited labeled data. *Cancers*, 13(7):1590, 2021.

Apple. Designing for privacy. URL `https://developer.apple.com/videos/play/wwdc2019/708/`.

Sara Babakniya, Ahmed Roushdy Elkordy, Yahya H Ezzeldin, Qingfeng Liu, Kee-Bong Song, Mostafa El-Khamy, and Salman Avestimehr. Slora: Federated parameter efficient fine-tuning of language models. *arXiv preprint arXiv:2308.06522*, 2023.

William H Beluch, Tim Genewein, Andreas Nürnberger, and Jan M Köhler. The power of ensembles for active learning in image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9368–9377, 2018.

Daniel J Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Javier Fernandez-Marques, Yan Gao, Lorenzo Sani, Kwing Hei Li, Titouan Parcollet, Pedro Porto Buarque de Gusmão, et al. Flower: A friendly federated learning research framework. *arXiv preprint arXiv:2007.14390*, 2020.

Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin Calo. Analyzing federated learning through an adversarial lens. In *International Conference on Machine Learning*, pages 634–643. PMLR, 2019.

Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for federated learning on user-held data. *arXiv preprint arXiv:1611.04482*, 2016.

Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečnỳ, Stefano Mazzocchi, H Brendan McMahan, et al. Towards federated learning at scale: System design. *arXiv preprint arXiv:1902.01046*, 2019.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Cristian Buciluă, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541, 2006.

Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. *arXiv preprint arXiv:1812.00332*, 2018.

Capital Economics and DCMS. AI Activity in UK Businesses Report, 2022. URL `https://www.gov.uk/government/publications/ai-activity-in-uk-businesses`. Capital Economics and DCMS, January 2022.

Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European conference on computer vision (ECCV)*, pages 132–149, 2018.

Hongyan Chang, Virat Shejwalkar, Reza Shokri, and Amir Houmansadr. Cronus: Robust and heterogeneous collaborative learning with black-box knowledge transfer. *arXiv preprint arXiv:1912.11279*, 2019.

Tianshi Che, Ji Liu, Yang Zhou, Jiaxiang Ren, Jiwen Zhou, Victor S Sheng, Huaiyu Dai, and Dejing Dou. Federated learning of large language models with parameter-efficient prompt tuning and adaptive optimization. *arXiv preprint arXiv:2310.15080*, 2023.

Dengsheng Chen, Jie Hu, Vince Junkai Tan, Xiaoming Wei, and Enhua Wu. Elastic aggregation for federated optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12187–12197, 2023.

Hong-You Chen and Wei-Lun Chao. Fedbe: Making bayesian model ensemble applicable to federated learning. *arXiv preprint arXiv:2009.01974*, 2020.

Mingqing Chen, Rajiv Mathews, Tom Ouyang, and Françoise Beaufays. Federated learning of out-of-vocabulary words. *arXiv preprint arXiv:1903.10635*, 2019a.

Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A closer look at few-shot classification. *arXiv preprint arXiv:1904.04232*, 2019b.

Yae Jee Cho, Andre Manoel, Gauri Joshi, Robert Sim, and Dimitrios Dimitriadis. Heterogeneous ensemble knowledge transfer for training large models in federated learning. *arXiv preprint arXiv:2204.12703*, 2022.

Smita Chormunge and Sudarson Jena. Correlation based feature selection with clustering for high dimensional data. *Journal of Electrical Systems and Information Technology*, 5(3):542–549, 2018.

Imran Arshad Choudhry, Saeed Iqbal, Musaed Alhussein, Khursheed Aurangzeb, Adnan N Qureshi, Muhammad Shahid Anwar, and Faheem Khan. Privacy-preserving ai for early diagnosis of thoracic diseases using iots: A federated learning approach with multi-headed self-attention for facilitating cross-institutional study. *Internet of Things*, 27:101296, 2024.

Patryk Chrabaszcz, Ilya Loshchilov, and Frank Hutter. A downsampled variant of imagenet as an alternative to the cifar datasets. *arXiv preprint arXiv:1707.08819*, 2017.

Samuele Cornell, Matthew Wiesner, Shinji Watanabe, Desh Raj, Xuankai Chang, Paola Garcia, Matthew Maciejewski, Yoshiki Masuyama, Zhong-Qiu Wang, Stefano Squartini, et al. The chime-7 dasr challenge: Distant meeting transcription with multiple devices in diverse scenarios. *arXiv preprint arXiv:2306.13734*, 2023.

Google Advanced Machine Learning Course. k-means advantages and disadvantages. URL `https://developers.google.com/machine-learning/clustering/algorithm/advantages-disadvantages`.

Pierre Courtiol, Charles Maussion, Matahi Moarii, Elodie Pronier, Samuel Pilcer, Meriem Sefta, Pierre Manceron, Sylvain Toldo, Mikhail Zaslavskiy, Nolwenn Le Stang, et al. Deep learning-based classification of mesothelioma improves prediction of patient outcome. *Nature medicine*, 25(10):1519–1525, 2019.

Xiaodong Cui, Songtao Lu, and Brian Kingsbury. Federated acoustic modeling for automatic speech recognition. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6748–6752. IEEE, 2021.

Yutong Dai, Zeyuan Chen, Junnan Li, Shelby Heinecke, Lichao Sun, and Ran Xu. Tackling data heterogeneity in federated learning with class prototypes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 7314–7322, 2023.

Katharine Daly, Hubert Eichner, Peter Kairouz, H Brendan McMahan, Daniel Ramage, and Zheng Xu. Federated learning in practice: Reflections and projections. *arXiv preprint arXiv:2410.08892*, 2024.

Enmao Diao, Jie Ding, and Vahid Tarokh. Heterofl: Computation and communication efficient federated learning for heterogeneous clients. *arXiv preprint arXiv:2010.01264*, 2020.

Daniel Madrigal Diaz, Andre Manoel, Jialei Chen, Nalin Singal, and Robert Sim. Project florida: Federated learning made easy. *arXiv preprint arXiv:2307.11899*, 2023.

Jie Ding, Eric Tramel, Anit Kumar Sahu, Shuang Wu, Salman Avestimehr, and Tao Zhang. Federated learning challenges and opportunities: An outlook. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8752–8756. IEEE, 2022.

Christophe Dupuy, Tanya G Roosta, Leo Long, Clement Chung, Rahul Gupta, and Salman Avestimehr. Learnings from federated learning in the real world. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8767–8771. IEEE, 2022.

Hubert Eichner, Daniel Ramage, Kallista Bonawitz, Dzmitry Huba, Tiziano Santoro, Brett McLarnon, Timon Van Overveldt, Nova Fallen, Peter Kairouz, Albert Cheu, et al. Confidential federated computations. *arXiv preprint arXiv:2404.10764*, 2024.

Ahmed El Ouadrhiri and Ahmed Abdelhadi. Differential privacy for deep and federated learning: A survey. *IEEE access*, 10:22359–22380, 2022.

European Parliament and Council of the European Union. Regulation (EU) 2016/679 of the European Parliament and of the Council, 2016. URL `https://data.europa.eu/eli/reg/2016/679/oj`.

Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach. *Advances in Neural Information Processing Systems*, 33:3557–3568, 2020.

Hao Fang, Yixiang Qiu, Hongyao Yu, Wenbo Yu, Jiawei Kong, Baoli Chong, Bin Chen, Xuan Wang, and Shu-Tao Xia. Privacy leakage on dnns: A survey of model inversion attacks and defenses. *arXiv preprint arXiv:2402.04013*, 2024.

Hossein Fereidooni, Samuel Marchal, Markus Miettinen, Azalia Mirhoseini, Helen Möllering, Thien Duc Nguyen, Phillip Rieger, Ahmad-Reza Sadeghi, Thomas Schneider, Hossein Yalame, et al. Safelearn: Secure aggregation for private federated learning. In *2021 IEEE Security and Privacy Workshops (SPW)*, pages 56–62. IEEE, 2021.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

Yifan Fu, Xingquan Zhu, and Bin Li. A survey on instance selection for active learning. *Knowledge and information systems*, 35:249–283, 2013.

Fei Gao, Hyunsoo Yoon, Teresa Wu, and Xianghua Chu. A feature transfer enabled multi-task deep learning model on medical imaging. *Expert Systems with Applications*, 143:112957, 2020.

Liang Gao, Huazhu Fu, Li Li, Yingwen Chen, Ming Xu, and Cheng-Zhong Xu. Feddc: Federated learning with non-iid data via local drift decoupling and correction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10112–10121, 2022.

Xuan Gong, Abhishek Sharma, Srikrishna Karanam, Ziyan Wu, Terrence Chen, David Doermann, and Arun Innanje. Ensemble attention distillation for privacy-preserving federated learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15076–15086, 2021.

Xuan Gong, Liangchen Song, Rishi Vedula, Abhishek Sharma, Meng Zheng, Benjamin Planche, Arun Innanje, Terrence Chen, Junsong Yuan, David Doermann, et al. Federated learning with privacy-preserving ensemble attention distillation. *IEEE transactions on medical imaging*, 2022.

Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129:1789–1819, 2021.

Filip Granqvist, Matt Seigel, Rogier Van Dalen, Aine Cahill, Stephen Shum, and Matthias Paulik. Improving on-device speaker verification using federated learning with privacy. *arXiv preprint arXiv:2008.02651*, 2020.

Neel Guha, Ameet Talwalkar, and Virginia Smith. One-shot federated learning. *arXiv preprint arXiv:1902.11175*, 2019.

Antonio Gulli. Ag's corpus of news articles, 2005. URL `http://groups.di.unipi.it/`.

Niv Haim, Gal Vardi, Gilad Yehudai, Ohad Shamir, and Michal Irani. Reconstructing training data from trained neural networks. *Advances in Neural Information Processing Systems*, 35:22911–22924, 2022.

Chunjing Han, Tong Li, Qiuyi Chen, Yulei Wu, and Jifeng Qin. Distributed and collaborative lightweight edge federated learning for iot zombie devices detection. *ACM Transactions on Sensor Networks*, 2024a.

Sungwon Han, Sungwon Park, Fangzhao Wu, Sundong Kim, Chuhan Wu, Xing Xie, and Meeyoung Cha. Fedx: Unsupervised federated learning with cross knowledge distillation. In *European Conference on Computer Vision*, pages 691–707. Springer, 2022.

Zeyu Han, Chao Gao, Jinyang Liu, Sai Qian Zhang, et al. Parameter-efficient fine-tuning for large models: A comprehensive survey. *arXiv preprint arXiv:2403.14608*, 2024b.

Emrah Hancer, Bing Xue, and Mengjie Zhang. A survey on feature selection approaches for clustering. *Artificial Intelligence Review*, 53:4519–4545, 2020.

Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604*, 2018.

Andrew Hard, Kurt Partridge, Cameron Nguyen, Niranjan Subrahmanya, Aishanee Shah, Pai Zhu, Ignacio Lopez Moreno, and Rajiv Mathews. Training keyword spotting models on non-iid data with federated learning. *arXiv preprint arXiv:2005.10406*, 2020.

Manuel Haussmann, Fred A Hamprecht, and Melih Kandemir. Deep active learning with adaptive acquisition. *arXiv preprint arXiv:1906.11471*, 2019.

Chaoyang He, Murali Annavaram, and Salman Avestimehr. Group knowledge transfer: Federated learning of large cnns at the edge. *Advances in Neural Information Processing Systems*, 33:14068–14080, 2020a.

Chaoyang He, Songze Li, Jinhyun So, Xiao Zeng, Mi Zhang, Hongyi Wang, Xiaoyang Wang, Praneeth Vepakomma, Abhishek Singh, Hang Qiu, et al. Fedml: A research library and benchmark for federated machine learning. *arXiv preprint arXiv:2007.13518*, 2020b.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *Journal of Machine Learning Research*, 22(241):1–124, 2021.

Samuel Horvath, Stefanos Laskaridis, Mario Almeida, Ilias Leontiadis, Stylianos Venieris, and Nicholas Lane. Fjord: Fair and accurate federated learning under heterogeneous targets with ordered dropout. *Advances in Neural Information Processing Systems*, 34:12876–12889, 2021.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR, 2019.

Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1314–1324, 2019.

Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335*, 2019.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pages 646–661. Springer, 2016.

Wenke Huang, Mang Ye, and Bo Du. Learn from others and be yourself in heterogeneous federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10143–10153, 2022.

IBM. Building privacy-preserving federated learning to help fight financial crime. URL `https://research.ibm.com/blog/privacy-preserving-federated-learning-finance`.

Sohei Itahara, Takayuki Nishio, Yusuke Koda, Masahiro Morikura, and Koji Yamamoto. Distillation-based semi-supervised federated learning for communication-efficient collaborative training with non-iid private data. *IEEE Transactions on Mobile Computing*, 22(1):191–205, 2021.

Anil K Jain and Richard C Dubes. *Algorithms for clustering data.* Prentice-Hall, Inc., 1988.

Johan Ludwig William Valdemar Jensen. Sur les fonctions convexes et les inégalités entre les valeurs moyennes. *Acta mathematica*, 30(1):175–193, 1906.

Eunjeong Jeong, Seungeun Oh, Hyesung Kim, Jihong Park, Mehdi Bennis, and Seong-Lyun Kim. Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data. *arXiv preprint arXiv:1811.11479*, 2018.

Wonyong Jeong, Jaehong Yoon, Eunho Yang, and Sung Ju Hwang. Federated semi-supervised learning with inter-client consistency & disjoint learning. *arXiv preprint arXiv:2006.12097*, 2020.

Shaoxiong Ji, Yue Tan, Teemu Saravirta, Zhiqin Yang, Yixin Liu, Lauri Vasankari, Shirui Pan, Guodong Long, and Anwar Walid. Emerging trends in federated learning: From model fusion to federated x learning. *International Journal of Machine Learning and Cybernetics*, pages 1–22, 2024.

Jingang Jiang, Xiangyang Liu, and Chenyou Fan. Low-parameter federated learning with large language models. *arXiv preprint arXiv:2307.13896*, 2023.

Yihan Jiang, Jakub Konečnỳ, Keith Rush, and Sreeram Kannan. Improving federated learning personalization via model agnostic meta learning. *arXiv preprint arXiv:1909.12488*, 2019.

Dong Jin, Shuangwu Chen, Huasen He, Xiaofeng Jiang, Siyu Cheng, and Jian Yang. Federated incremental learning based evolvable intrusion detection system for zero-day attacks. *Ieee Network*, 37(1):125–132, 2023.

Ian T Jolliffe. *Principal component analysis for special types of data.* Springer, 2002.

Ajay J Joshi, Fatih Porikli, and Nikolaos Papanikolopoulos. Multi-class active learning for image classification. In *2009 ieee conference on computer vision and pattern recognition*, pages 2372–2379. IEEE, 2009.

Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021.

Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, pages 5132–5143. PMLR, 2020.

Angelos Katharopoulos and François Fleuret. Not all samples are created equal: Deep learning with importance sampling. In *International conference on machine learning*, pages 2525–2534. PMLR, 2018.

Latif U Khan, Walid Saad, Zhu Han, Ekram Hossain, and Choong Seon Hong. Federated learning for internet of things: Recent advances, taxonomy, and open challenges. *IEEE Communications Surveys & Tutorials*, 23(3):1759–1799, 2021.

Naimul Mefraz Khan, Nabila Abraham, and Marcia Hon. Transfer learning with intelligent training data selection for prediction of alzheimer's disease. *IEEE Access*, 7:72726–72735, 2019.

Mikhail Khodak, Maria-Florina Balcan, and Ameet Talwalkar. Adaptive gradient-based meta-learning methods. *arXiv preprint arXiv:1906.02717*, 2019.

Sunder Ali Khowaja, Ik Hyun Lee, Kapal Dev, Muhammad Aslam Jarwar, and Nawab Muhammad Faseeh Qureshi. Get your foes fooled: Proximal gradient split learning for defense against model inversion attacks on iomt data. *IEEE Transactions on Network Science and Engineering*, 10(5):2607–2616, 2022.

Hansol Kim, Youngjun Kwak, Minyoung Jung, Jinho Shin, Youngsung Kim, and Changick Kim. Protofl: Unsupervised federated learning via prototypical distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6470–6479, 2023a.

SangMook Kim, Sangmin Bae, Hwanjun Song, and Se-Young Yun. Re-thinking federated active learning based on inter-class diversity. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3944–3953, 2023b.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

Trupti M Kodinariya, Prashant R Makwana, et al. Review on determining number of cluster in k-means clustering. *International Journal*, 1(6):90–95, 2013.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International conference on machine learning*, pages 3519–3529. PMLR, 2019.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

Kummari Naveen Kumar, Chalavadi Krishna Mohan, and Linga Reddy Cenkeramaddi. The impact of adversarial attacks on federated learning: A survey. *IEEE*

Transactions on Pattern Analysis and Machine Intelligence, 46(5):2672–2691, 2023.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521 (7553):436–444, 2015.

David Leroy, Alice Coucke, Thibaut Lavril, Thibault Gisselbrecht, and Joseph Dureau. Federated learning for keyword spotting. In *ICASSP 2019-2019 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 6341–6345. IEEE, 2019.

Anran Li, Lan Zhang, Juntao Tan, Yaxuan Qin, Junhao Wang, and Xiang-Yang Li. Sample-level data selection for federated learning. In *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*, pages 1–10. IEEE, 2021a.

Daliang Li and Junpu Wang. Fedmd: Heterogenous federated learning via model distillation. *arXiv preprint arXiv:1910.03581*, 2019.

Lei Li, Linda Yu-Ling Lan, Lei Huang, Congting Ye, Jorge Andrade, and Patrick C Wilson. Selecting representative samples from complex biological datasets using k-medoids clustering. *Frontiers in Genetics*, 13:954024, 2022.

Lusi Li, Haibo He, and Jie Li. Entropy-based sampling approaches for multi-class imbalanced problems. *IEEE Transactions on Knowledge and Data Engineering*, 32(11):2159–2170, 2019a.

Qinbin Li, Bingsheng He, and Dawn Song. Model-contrastive federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10713–10722, June 2021b.

Qinbin Li, Zeyi Wen, Zhaomin Wu, Sixu Hu, Naibo Wang, Yuan Li, Xu Liu, and Bingsheng He. A survey on federated learning systems: Vision, hype and reality for data privacy and protection. *IEEE Transactions on Knowledge and Data Engineering*, 2021c.

Tian Li, Maziar Sanjabi, Ahmad Beirami, and Virginia Smith. Fair resource allocation in federated learning. *arXiv preprint arXiv:1905.10497*, 2019b.

Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated Learning: Challenges, Methods, and Future Directions. *IEEE Signal Processing Magazine*, 37(3):50–60, may 2020a. ISSN 15580792. doi: 10.1109/MSP.2020.2975749.

Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE signal processing magazine*, 37(3):50–60, 2020b.

Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2:429–450, 2020c.

Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of fedavg on non-iid data. *arXiv preprint arXiv:1907.02189*, 2019c.

Xiaoxiao Li, Meirui Jiang, Xiaofei Zhang, Michael Kamp, and Qi Dou. Fedbn: Federated learning on non-iid features via local batch normalization. *arXiv preprint arXiv:2102.07623*, 2021d.

Xingjian Li, Haoyi Xiong, Hanchao Wang, Yuxuan Rao, Liping Liu, Zeyu Chen, and Jun Huan. Delta: Deep learning transfer using feature map with attention for convolutional networks. *arXiv preprint arXiv:1901.09229*, 2019d.

Yangfan Li and Chunjiang Bian. Few-shot fine-grained ship classification with a foreground-aware feature map reconstruction network. *IEEE Transactions on Geoscience and Remote Sensing*, 60:1–12, 2022.

Zexi Li, Xinyi Shang, Rui He, Tao Lin, and Chao Wu. No fear of classifier biases: Neural collapse inspired federated learning with synthetic and fixed classifier. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5319–5329, 2023.

Paul Pu Liang, Terrance Liu, Liu Ziyin, Nicholas B Allen, Randy P Auerbach, David Brent, Ruslan Salakhutdinov, and Louis-Philippe Morency. Think locally, act globally: Federated learning with local and global representations. *arXiv preprint arXiv:2001.01523*, 2020.

Tao Lin, Lingjing Kong, Sebastian U Stich, and Martin Jaggi. Ensemble distillation for robust model fusion in federated learning. *Advances in Neural Information Processing Systems*, 33:2351–2363, 2020.

Boyi Liu, Lujia Wang, and Ming Liu. Lifelong federated reinforcement learning: a learning architecture for navigation in cloud robotic systems. *IEEE Robotics and Automation Letters*, 4(4):4555–4562, 2019a.

Fangxin Liu, Wenbo Zhao, Zhezhi He, Yanzhi Wang, Zongwu Wang, Changzhi Dai, Xiaoyao Liang, and Li Jiang. Improving neural network efficiency via post-training quantization with adaptive floating-point. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5281–5290, 2021a.

Ruixuan Liu, Fangzhao Wu, Chuhan Wu, Yanlin Wang, Lingjuan Lyu, Hong Chen, and Xing Xie. No one left behind: Inclusive federated learning over heterogeneous devices. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 3398–3406, 2022.

Shikun Liu, Edward Johns, and Andrew J Davison. End-to-end multi-task learning with attention. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1871–1880, 2019b.

Tao Liu, Zhi Wang, Hui He, Wei Shi, Liangliang Lin, Ran An, and Chenhao Li. Efficient and secure federated learning for financial applications. *Applied Sciences*, 13(10):5877, 2023.

Xiaoyuan Liu, Hongwei Li, Guowen Xu, Zongqi Chen, Xiaoming Huang, and Rongxing Lu. Privacy-enhanced federated learning against poisoning adversaries. *IEEE Transactions on Information Forensics and Security*, 16:4574–4588, 2021b.

Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE international conference on computer vision*, pages 2736–2744, 2017.

Zhuoming Liu, Hao Ding, Huaping Zhong, Weijia Li, Jifeng Dai, and Conghui He. Influence selection for active learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9274–9283, 2021c.

Mi Luo, Fei Chen, Dapeng Hu, Yifan Zhang, Jian Liang, and Jiashi Feng. No fear of heterogeneity: Classifier calibration for federated learning with non-iid data. *Advances in Neural Information Processing Systems*, 34:5972–5984, 2021.

Wenjie Luo, Alex Schwing, and Raquel Urtasun. Latent structured active learning. *Advances in Neural Information Processing Systems*, 26, 2013.

James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

Mohamad Mansouri, Melek Önen, Wafa Ben Jaballah, and Mauro Conti. Sok: Secure aggregation based on cryptographic schemes for federated learning. *Proceedings on Privacy Enhancing Technologies*, 2023.

Othmane Marfoq, Chuan Xu, Giovanni Neglia, and Richard Vidal. Throughput-optimal topology design for cross-silo federated learning. *Advances in Neural Information Processing Systems*, 33:19478–19487, 2020.

Brendan McMahan and Daniel Ramage. Federated learning: Collaborative machine learning without centralized training data. URL `https://blog.research.google/2017/04/federated-learning-collaborative.html`.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *AIStat*. PMLR, 2017.

Yiqun Mei, Pengfei Guo, Mo Zhou, and Vishal Patel. Resource-adaptive federated learning with all-in-one neural composition. *Advances in Neural Information Processing Systems*, 35:4270–4284, 2022.

Matias Mendieta, Taojiannan Yang, Pu Wang, Minwoo Lee, Zhengming Ding, and Chen Chen. Local learning matters: Rethinking data heterogeneity in federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8397–8406, 2022.

Sören Mindermann, Jan M Brauner, Muhammed T Razzak, Mrinank Sharma, Andreas Kirsch, Winnie Xu, Benedikt Höltgen, Aidan N Gomez, Adrien Morisot, Sebastian Farquhar, et al. Prioritized training on points that are learnable, worth learning, and not yet learnt. In *International Conference on Machine Learning*, pages 15630–15649. PMLR, 2022.

Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3994–4003, 2016.

Jose G Moreno-Torres, Troy Raeder, Rocío Alaiz-Rodríguez, Nitesh V Chawla, and Francisco Herrera. A unifying view on dataset shift in classification. *Pattern recognition*, 45(1):521–530, 2012.

L Muñoz-González, KT Co, and EC Lupu. Byzantine-robust federated machine learning through adaptive model averaging. *arXiv preprint arXiv:1909.05125*, 2019.

Lokesh Nagalapatti, Ruhi Sharma Mittal, and Ramasuri Narayanam. Is your data relevant?: Dynamic selection of relevant data for federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 7859–7867, 2022.

Ali Bou Nassif, Ismail Shahin, Imtinan Attili, Mohammad Azzeh, and Khaled Shaalan. Speech recognition using deep neural networks: A systematic review. *IEEE access*, 7:19143–19165, 2019.

Megha Natarajan. Understanding cumulative explained variance in pca with python. URL `https://medium.com/@megha.natarajan/understanding-cumulative-explained-variance-in-pca-with-python-653e3592a77c`.

Behnam Neyshabur, Hanie Sedghi, and Chiyuan Zhang. What is being transferred in transfer learning? *Advances in neural information processing systems*, 33: 512–523, 2020.

Duy Phuong Nguyen, Sixing Yu, J Pablo Muñoz, and Ali Jannesari. Enhancing heterogeneous federated learning with knowledge extraction and multi-model fusion. In *Proceedings of the SC'23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis*, pages 36–43, 2023.

John Nguyen, Kshitiz Malik, Hongyuan Zhan, Ashkan Yousefpour, Mike Rabbat, Mani Malek, and Dzmitry Huba. Federated learning with buffered asynchronous aggregation. In *International Conference on Artificial Intelligence and Statistics*, pages 3581–3607. PMLR, 2022.

Thao Nguyen, Maithra Raghu, and Simon Kornblith. Do wide and deep networks learn the same things? uncovering how neural network representations vary with width and depth. *arXiv preprint arXiv:2010.15327*, 2020.

Alex Nichol and John Schulman. Reptile: a scalable metalearning algorithm. *arXiv preprint arXiv:1803.02999*, 2(3):4, 2018.

Takayuki Nishio and Ryo Yonetani. Client selection for federated learning with heterogeneous resources in mobile edge. In *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, pages 1–7, 2019. doi: 10.1109/ICC.2019.8761315.

Jaehoon Oh, Sangmook Kim, and Se-Young Yun. Fedbabu: Towards enhanced representation for federated image classification. *arXiv preprint arXiv:2106.06042*, 2021.

Martijn Oldenhof, Gergely Ács, Balázs Pejó, Ansgar Schuffenhauer, Nicholas Holway, Noé Sturm, Arne Dieckmann, Oliver Fortmeier, Eric Boniface, Clément Mayer, et al. Industry-scale orchestrated federated learning for drug discovery. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 15576–15584, 2023.

Fernanda C Orlandi, Julio CS Dos Anjos, Juan F de P Santana, Valderi RQ Leithardt, and Claudio FR Geyer. Entropy to mitigate non-iid data problem on federated learning for the edge intelligence environment. *IEEE Access*, 2023.

Archit Parnami and Minwoo Lee. Learning from few examples: A summary of approaches to few-shot learning. *arXiv preprint arXiv:2203.04291*, 2022.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

Duc Truong Pham, Stefan S Dimov, and Chi D Nguyen. Selection of k in k-means clustering. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 219(1):103–119, 2005.

Renjie Pi, Weizhong Zhang, Yueqi Xie, Jiahui Gao, Xiaoyu Wang, Sunghun Kim, and Qifeng Chen. Dynafed: Tackling client data heterogeneity with global dynamics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12177–12186, 2023.

Antonio Polino, Razvan Pascanu, and Dan Alistarh. Model compression via distillation and quantization. *arXiv preprint arXiv:1802.05668*, 2018.

Joaquin Quinonero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D Lawrence. *Dataset shift in machine learning.* Mit Press, 2008.

Sashank Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečný, Sanjiv Kumar, and H Brendan McMahan. Adaptive federated optimization. *arXiv preprint arXiv:2003.00295*, 2020.

Yasar Abbas Ur Rehman, Yan Gao, Pedro Porto Buarque De Gusmão, Mina Alibeigi, Jiajun Shen, and Nicholas D Lane. L-dawa: Layer-wise divergence aware weight aggregation in federated self-supervised visual representation learning. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 16464–16473, 2023.

Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Brij B Gupta, Xiaojiang Chen, and Xin Wang. A survey of deep active learning. *ACM computing surveys (CSUR)*, 54(9):1–40, 2021.

Paul D Rosero-Montalvo, Pınar Tözün, and Wilmar Hernandez. Optimized cnn architectures benchmarking in hardware-constrained edge devices in iot environments. *IEEE Internet of Things Journal*, 2024.

Amrita Roy Chowdhury, Chuan Guo, Somesh Jha, and Laurens van der Maaten. Eiffel: Ensuring integrity for federated learning. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 2535–2549, 2022.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115:211–252, 2015.

Lorenzo Sani, Alex Iacob, Zeyu Cao, Bill Marino, Yan Gao, Tomas Paulik, Wanru Zhao, William F Shen, Preslav Aleksandrov, Xinchi Qiu, et al. The future of large language model pre-training is federated. *arXiv preprint arXiv:2405.10853*, 2024.

Felix Sattler, Tim Korjakow, Roman Rischke, and Wojciech Samek. Fedaux: Leveraging unlabeled auxiliary data in federated learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

George Seif. The 5 clustering algorithms data scientists need to know — by george seif — towards data science, 2018. URL `https://towardsdatascience.com/the-5-clustering-algorithms-data-scientists-need-to-know-a36d136ef68`.

Burr Settles. Active learning literature survey. 2009.

Osama Shahid, Seyedamin Pouriyeh, Reza M Parizi, Quan Z Sheng, Gautam Srivastava, and Liang Zhao. Communication efficiency in federated learning: Achievements and challenges. *arXiv preprint arXiv:2107.10996*, 2021.

Claude Elwood Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

Virat Shejwalkar, Amir Houmansadr, Peter Kairouz, and Daniel Ramage. Back to the drawing board: A critical evaluation of poisoning attacks on production federated learning. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1354–1371. IEEE, 2022.

Hongrui Shi and Valentin Radu. Towards federated learning with attention transfer to mitigate system and data heterogeneity of clients. In *Proceedings of the 4th international workshop on edge systems, analytics and networking*, pages 61–66, 2021.

Hongrui Shi and Valentin Radu. Data selection for efficient model update in federated learning. In *Proceedings of the 2nd European Workshop on Machine Learning and Systems*, pages 72–78, 2022.

Hongrui Shi, Valentin Radu, and Po Yang. Distributed training for speech recognition using local knowledge aggregation and knowledge distillation in heterogeneous systems. In *Proceedings of the 3rd Workshop on Machine Learning and Systems*, pages 64–70, 2023a.

Hongrui Shi, Valentin Radu, and Po Yang. Lightweight workloads in heterogeneous federated learning via few-shot learning. In *Proceedings of the 4th International Workshop on Distributed Machine Learning*, pages 21–26, 2023b.

Neta Shoham, Tomer Avidor, Aviv Keren, Nadav Israel, Daniel Benditkis, Liron Mor-Yosef, and Itai Zeitak. Overcoming forgetting in federated learning on non-iid data. *arXiv preprint arXiv:1910.07796*, 2019.

Xian Shuai, Yulin Shen, Siyang Jiang, Zhihe Zhao, Zhenyu Yan, and Guoliang Xing. Balancefl: Addressing class imbalance in long-tail federated learning. In *2022 21st ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 271–284. IEEE, 2022.

Samarth Sinha, Sayna Ebrahimi, and Trevor Darrell. Variational adversarial active learning. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5972–5981, 2019.

sklearn. sklearn-decomposition-pca. URL `https://sciki-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html`.

Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet Talwalkar. Federated multi-task learning. *arXiv preprint arXiv:1705.10467*, 2017.

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30, 2017.

Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. Training very deep networks. *Advances in neural information processing systems*, 28, 2015.

Stephen V Stehman. Selecting and interpreting measures of thematic classification accuracy. *Remote sensing of Environment*, 62(1):77–89, 1997.

Shangchao Su, Bin Li, and Xiangyang Xue. One-shot federated learning without server-side training. *Neural Networks*, 164:203–215, 2023.

Jelal Sultanov. What is few-shot learning?, 2020. URL `https://medium.com/ai-theory-practice-business/what-is-few-shot-learning-4b2842646b47`.

Lichao Sun and Lingjuan Lyu. Federated model distillation with noise-free differential privacy. *arXiv preprint arXiv:2009.05537*, 2020.

Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1199–1208, 2018.

Alysa Ziying Tan, Han Yu, Lizhen Cui, and Qiang Yang. Towards personalized federated learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2022a.

Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *Proc. CVPR*, 2019.

Yue Tan, Guodong Long, Jie Ma, Lu Liu, Tianyi Zhou, and Jing Jiang. Federated learning from pre-trained models: A contrastive learning approach. *Advances in neural information processing systems*, 35:19332–19344, 2022b.

Dipanwita Thakur, Antonella Guzzo, and Giancarlo Fortino. Hardware-algorithm co-design of energy efficient federated learning in quantized neural network. *Internet of Things*, page 101223, 2024.

Sebastian Thrun and Lorien Pratt. *Learning to learn.* Springer Science & Business Media, 2012.

Yonglong Tian, Yue Wang, Dilip Krishnan, Joshua B Tenenbaum, and Phillip Isola. Rethinking few-shot image classification: a good embedding is all you need? In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIV 16*, pages 266–282. Springer, 2020.

Jack Turner, Elliot J Crowley, Michael O'Boyle, Amos Storkey, and Gavin Gray. Blockswap: Fisher-guided block substitution for network compression on a budget. *arXiv preprint arXiv:1906.04113*, 2019.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

Jake VanderPlas. *Python data science handbook: Essential tools for working with data.* " O'Reilly Media, Inc.", 2016.

A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. *Advances in neural information processing systems*, 29, 2016.

Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H Vincent Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization. *Advances in neural information processing systems*, 33:7611–7623, 2020a.

Jianyu Wang, Zachary Charles, Zheng Xu, Gauri Joshi, H Brendan McMahan, Maruan Al-Shedivat, Galen Andrew, Salman Avestimehr, Katharine Daly, Deepesh Data, et al. A field guide to federated optimization. *arXiv preprint arXiv:2107.06917*, 2021.

Jiaqi Wang, Xingyi Yang, Suhan Cui, Liwei Che, Lingjuan Lyu, Dongkuan DK Xu, and Fenglong Ma. Towards personalized federated learning via heterogeneous model reassembly. *Advances in Neural Information Processing Systems*, 36, 2024a.

Lin Wang and Kuk-Jin Yoon. Knowledge distillation and student-teacher learning for visual intelligence: A review and new outlooks. *IEEE transactions on pattern analysis and machine intelligence*, 44(6):3048–3068, 2021.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *Advances in Neural Information Processing Systems*, 33: 5776–5788, 2020b.

Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. Generalizing from a few examples: A survey on few-shot learning. *ACM computing surveys (csur)*, 53(3):1–34, 2020c.

Zi Wang, Fei Wu, Feng Yu, Yurui Zhou, Jia Hu, and Geyong Min. Federated continual learning for edge-ai: A comprehensive survey. *arXiv preprint arXiv:2411.13740*, 2024b.

Pete Warden. Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv preprint arXiv:1804.03209*, 2018.

Shinji Watanabe, Michael Mandel, Jon Barker, Emmanuel Vincent, Ashish Arora, Xuankai Chang, Sanjeev Khudanpur, Vimal Manohar, Daniel Povey, Desh Raj, et al. Chime-6 challenge: Tackling multispeaker speech recognition for unsegmented recordings. *arXiv preprint arXiv:2004.09249*, 2020.

Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H Yang, Farhad Farokhi, Shi Jin, Tony QS Quek, and H Vincent Poor. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE transactions on information forensics and security*, 15:3454–3469, 2020.

Yuan Wen, Andrew Anderson, Valentin Radu, Michael FP O'Boyle, and David Gregg. Taso: Time and space optimization for memory-constrained dnn inference. In *Proc. SBAC-PAD)*. IEEE, 2020.

Davis Wertheimer, Luming Tang, and Bharath Hariharan. Few-shot classification with feature map reconstruction networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8012–8021, 2021.

WIKIPEDIA. Entropy (information theory). URL `https://en.wikipedia.org/wiki/Entropy-(information-theory)`.

Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *Proc. CVPR*, 2019.

Michael Xie, Neal Jean, Marshall Burke, David Lobell, and Stefano Ermon. Transfer learning from deep features for remote sensing and poverty mapping. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

Jian Xu, Xinyi Tong, and Shao-Lun Huang. Personalized federated learning with feature alignment and classifier collaboration. *arXiv preprint arXiv:2306.11867*, 2023.

Mingzhao Yang, Shangchao Su, Bin Li, and Xiangyang Xue. Exploring one-shot semi-supervised federated learning with pre-trained diffusion models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 16325–16333, 2024.

Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–19, 2019.

Taojiannan Yang, Sijie Zhu, and Chen Chen. Gradaug: A new regularization method for deep neural networks. *Advances in Neural Information Processing Systems*, 33:14207–14218, 2020.

Yu Yang, Hao Kang, and Baharan Mirzasoleiman. Towards sustainable learning: Coresets for data-efficient deep learning. In *International Conference on Machine Learning*, pages 39314–39330. PMLR, 2023.

Mang Ye, Xiuwen Fang, Bo Du, Pong C Yuen, and Dacheng Tao. Heterogeneous federated learning: State-of-the-art and research challenges. *ACM Computing Surveys*, 56(3):1–44, 2023.

Junho Yim, Donggyu Joo, Jihoon Bae, and Junmo Kim. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4133–4141, 2017.

Yupeng Yin, Xianglong Zhang, Huanle Zhang, Feng Li, Yue Yu, Xiuzhen Cheng, and Pengfei Hu. Ginver: Generative model inversion attacks against collaborative inference. In *Proceedings of the ACM Web Conference 2023*, pages 2122–2131, 2023.

Xianjia Yu, Jorge Pena Queralta, and Tomi Westerlund. Towards lifelong federated learning in autonomous mobile robots with continuous sim-to-real transfer. *Procedia Computer Science*, 210:86–93, 2022.

Chunhui Yuan and Haitao Yang. Research on k-value selection method of k-means clustering algorithm. *J*, 2(2):226–235, 2019.

Tianning Yuan, Fang Wan, Mengying Fu, Jianzhuang Liu, Songcen Xu, Xiangyang Ji, and Qixiang Ye. Multiple instance active learning for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5330–5339, 2021.

Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. *arXiv preprint arXiv:1612.03928*, 2016a.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016b.

Xueying Zhan, Huan Liu, Qing Li, and Antoni B Chan. A comparative survey: Benchmarking for pool-based active learning. In *IJCAI*, pages 4679–4686, 2021.

Fengda Zhang, Kun Kuang, Long Chen, Zhaoyang You, Tao Shen, Jun Xiao, Yin Zhang, Chao Wu, Fei Wu, Yueting Zhuang, et al. Federated unsupervised representation learning. *Frontiers of Information Technology & Electronic Engineering*, 24(8):1181–1193, 2023a.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

Jiale Zhang, Junjun Chen, Di Wu, Bing Chen, and Shui Yu. Poisoning attack in federated learning using generative adversarial nets. In *2019 18th IEEE international conference on trust, security and privacy in computing and communi-*

cations/13th IEEE international conference on big data science and engineering (TrustCom/BigDataSE), pages 374–380. IEEE, 2019.

Jianyi Zhang, Saeed Vahidian, Martin Kuo, Chunyuan Li, Ruiyi Zhang, Tong Yu, Guoyin Wang, and Yiran Chen. Towards building the federatedgpt: Federated instruction tuning. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6915–6919. IEEE, 2024.

Jie Zhang, Song Guo, Xiaosong Ma, Haozhao Wang, Wenchao Xu, and Feijie Wu. Parameterized knowledge transfer for personalized federated learning. *Advances in Neural Information Processing Systems*, 34:10092–10104, 2021.

Jie Zhang, Chen Chen, Bo Li, Lingjuan Lyu, Shuang Wu, Shouhong Ding, Chunhua Shen, and Chao Wu. Dense: Data-free one-shot federated learning. *Advances in Neural Information Processing Systems*, 35:21414–21428, 2022a.

Jie Zhang, Chen Chen, Weiming Zhuang, and Lingjuan Lyu. Target: Federated class-continual learning via exemplar-free distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4782–4793, 2023b.

Lin Zhang, Li Shen, Liang Ding, Dacheng Tao, and Ling-Yu Duan. Fine-tuning global model via data-free knowledge distillation for non-iid federated learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10174–10183, 2022b.

Tuo Zhang, Lei Gao, Chaoyang He, Mi Zhang, Bhaskar Krishnamachari, and A Salman Avestimehr. Federated learning for the internet of things: Applications, challenges, and opportunities. *IEEE Internet of Things Magazine*, 5(1): 24–29, 2022c.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28, 2015.

Yu Zhang and Qiang Yang. A survey on multi-task learning. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.

Yu Zheng, Wei Song, Minxin Du, Sherman SM Chow, Qian Lou, Yongjun Zhao, and Xiuhua Wang. Cryptography-inspired federated learning for generative adversarial networks and meta learning. In *International Conference on Advanced Data Mining and Applications*, pages 393–407. Springer, 2023.

Dongzhan Zhou, Xinchi Zhou, Wenwei Zhang, Chen Change Loy, Shuai Yi, Xuesen Zhang, and Wanli Ouyang. Econas: Finding proxies for economical neural architecture search. In *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, pages 11396–11404, 2020a.

Yanlin Zhou, George Pu, Xiyao Ma, Xiaolin Li, and Dapeng Wu. Distilled one-shot federated learning. *arXiv preprint arXiv:2009.07999*, 2020b.

Hongbin Zhu, Yong Zhou, Hua Qian, Yuanming Shi, Xu Chen, and Yang Yang. Online client selection for asynchronous federated learning with fairness consideration. *IEEE Transactions on Wireless Communications*, 22(4):2493–2506, 2022.

Zhuangdi Zhu, Junyuan Hong, and Jiayu Zhou. Data-free knowledge distillation for heterogeneous federated learning. In *International conference on machine learning*, pages 12878–12889. PMLR, 2021.

Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2020.

Weiming Zhuang, Xin Gan, Yonggang Wen, Xuesen Zhang, Shuai Zhang, and Shuai Yi. Towards unsupervised domain adaptation for deep face recognition under privacy constraints via federated learning. *arXiv preprint arXiv:2105.07606*, 2021.

Weiming Zhuang, Yonggang Wen, and Shuai Zhang. Divergence-aware federated self-supervised learning. *arXiv preprint arXiv:2204.04385*, 2022.

Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710, 2018.