An Exemplar-Informed Approach to Speech and Language Tasks

Rhiannon Mogridge

Supervisor: Dr. Anton Ragni

A thesis submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy in Computer Science

in the

Department of Computer Science

December 2024

Acknowledgements

Firstly, I would like to thank my supervisor, Anton Ragni, for his insight and expertise over the last four years.

I would also like to thank Rama Doddipatla, Mohan Li and Cong Thanh Do at Toshiba for their feedback and suggestions.

Thanks are also due to Stuart Wrigley and Lizzie Pass for managing to stay cheerful and helpful while I asked them far too many questions about paperwork.

Finally I would like to thank my family: my parents for their moral support; my brother, sister-in-law and nieces for distracting me when I needed it; and and my sister, for her endless patience (and occasional postal chocolate).

This work was supported by the Centre for Doctoral Training in Speech and Language Technologies and their Applications funded by the UK Research and Innovation grant EP/S023062/1. This work was also supported by Toshiba.

Abstract

The field of machine learning has drawn heavily from the fields of psychology and neuroscience, in particular in the development of artificial neural network architectures, which are based on simplified versions of structures in the brain. While effective for many tasks, neural networks do not, in general, incorporate any way of storing specific experiences, instead using training data to parameterise a model, and then discarding the training date prior to inference. We explore an alternative option: a simple, explainable model from the field of human psychology called Minerva 2, which uses previously seen examples to perform classification or regression. By comparing Minerva 2 with neural networks, we demonstrate that Minerva 2 is in fact a neural network itself, with parameters taken directly from the data, rather than being trained by backpropagation. We propose new architectures, which are based on Minerva 2 and incorporate both a memory of previous examples and parameterisation that allows model flexibility. We show that feature representation is crucial for this type of model, which might explain the lack of representation of this type of model in the literature. Speech and text representations have improved rapidly in recent years, however, and if this trend continues, simple, interpretable models such those proposed here will become more competitive. As evidence of this, we use high quality speech representations in conjunction with a Minerva-based model to demonstrate state-of-the-art performance on a speech intelligibility task.

Contents

1 Introduction **1**
- 1.1 Background . . . 1
- 1.2 Exemplar and prototype models . . . 2
- 1.3 Research questions . . . 4
- 1.4 Contributions . . . 5
- 1.5 Thesis outline . . . 6
- 1.6 Publications . . . 6
- 1.7 Notation . . . 7

2 Preliminaries **9**
- 2.1 Overview . . . 9
- 2.2 Minerva 2 . . . 9
  - 2.2.1 Echo-of-echoes . . . 11
  - 2.2.2 Learning and forgetting . . . 12
  - 2.2.3 Prior work in human psychology . . . 12
  - 2.2.4 Minerva 2 for speech and language tasks . . . 14
- 2.3 Alternative memory models . . . 15
- 2.4 Other exemplar approaches to speech and language tasks . . . 16
  - 2.4.1 k-nearest neighbour . . . 17
  - 2.4.2 Dynamic time warping . . . 17
  - 2.4.3 Concatenative speech synthesis . . . 18
- 2.5 Prototype approaches to speech and language tasks . . . 18
  - 2.5.1 Feed-forward neural network . . . 19
  - 2.5.2 Deep equilibrium models . . . 20
  - 2.5.3 Convolutional neural network . . . 20
  - 2.5.4 Recurrent neural network . . . 21
  - 2.5.5 Bi-directional RNN . . . 22
  - 2.5.6 Long Short-Term Memory . . . 23

2.5.7 Transformer . . . 24
2.5.8 Bidirectional encoder representations from transformers . . . 26
2.5.9 Training artificial neural networks . . . 27
2.5.10 Hyperparameter tuning . . . 28
2.6 Hybrid approaches to speech and language tasks . . . 29
2.6.1 Conditional neural processes . . . 29
2.6.2 Adding memory to parameterised models . . . 32
2.7 Feature representation . . . 32
2.7.1 Text representations . . . 33
2.7.2 Speech representations . . . 36
2.8 Evaluation metrics . . . 39
2.8.1 Accuracy, precision, recall and F_1 score . . . 39
2.9 Concluding remarks . . . 42

3 Theoretical framework 43
3.1 Introduction . . . 43
3.2 Minerva 2 as an artificial neural network . . . 44
3.3 Echo-of-echoes . . . 45
3.4 Exemplar selection . . . 48
3.4.1 Exemplar set size . . . 48
3.4.2 Exemplar selection . . . 48
3.5 Adaptation of Minerva 2 to speech and language tasks . . . 49
3.5.1 Minerva-R: Minerva with real input . . . 50
3.5.2 Minerva-RP: Learning feature transformations . . . 51
3.5.3 Minerva-RPE: Learning exemplar class representations . . . 53
3.5.4 Minerva-RPC: Learning ‘true’ class representations . . . 54
3.5.5 Minerva-RPES: Sequence Minerva . . . 55
3.5.6 Concluding remarks . . . 57

4 Experiments 59
4.1 Introduction . . . 59
4.2 Datasets . . . 60
4.2.1 TIMIT . . . 60
4.2.2 GoEmotions . . . 61
4.2.3 Clarity Prediction Challenge 2 . . . 64
4.3 Shared methodology . . . 66
4.4 Exemplar sets and the activation power . . . 66
4.4.1 Methodology . . . 67
4.4.2 Results and discussion . . . 67

4.5 Learned feature transformations 72
4.5.1 Methodology 72
4.5.2 Results and discussion 73
4.6 The impact of feature representation 78
4.7 Learned exemplar labels 79
4.7.1 Methodology 79
4.7.2 Results and discussion 79
4.8 Learned 'true' class representations 84
4.8.1 Methodology 84
4.8.2 Results and discussion 85
4.9 Unfixed exemplars 92
4.9.1 Methodology 93
4.9.2 Results and discussion 94
4.10 Adaptive exemplars 100
4.10.1 Methodology 100
4.10.2 Results and discussion 101
4.11 Sequence Minerva 103
4.11.1 Methodology 104
4.11.2 Results and discussion 104
4.12 Sequence Minerva and transformer 107
4.12.1 Methodology 107
4.12.2 Results and discussion 108
4.13 Concluding remarks 108

5 Applications: Clarity Prediction Challenge 2 111
5.1 Introduction 111
5.1.1 Overview 111
5.1.2 Chapter content 112
5.1.3 Background 112
5.2 Clarity Prediction Challenge 2 113
5.2.1 Challenge baseline 113
5.3 Prior approaches 113
5.4 Experimental setup 114
5.4.1 Feature representation 114
5.4.2 Model architecture 114
5.4.3 Prototype BLSTM SI prediction model 114
5.4.4 Exemplar-informed BLSTM SI prediction model 115
5.4.5 System combination 116
5.4.6 Simplified models 116

5.5 Results and discussion . . . 117
5.5.1 Prototype and exemplar-informed models . . . 118
5.5.2 Performance by intelligibility . . . 118
5.5.3 Model performance on unseen enhancement systems . . . 118
5.5.4 Whisper layer weights . . . 121
5.5.5 Performance comparison . . . 122
5.5.6 Concluding remarks . . . 122

6 Conclusions and further work **125**
6.1 Summary of contributions . . . 125
6.2 Further work . . . 127
6.2.1 Minerva initialisation . . . 127
6.2.2 Echo-of-echoes and deep equilibrium models . . . 127
6.2.3 Sequence Minerva . . . 128
6.2.4 Adaptative exemplars . . . 128

Appendices **143**

A Hyperparameters and tuning **145**

List of Figures

1.1 Prototype approach. 2
1.2 Exemplar approach. 3

2.1 Minerva 2. 10
2.2 Minerva 2 echo-of-echoes. 12
2.3 Example of a finite state grammar. 14
2.4 Convolutional Neural Network. 21
2.5 Two examples of CNNs with different strides and kernel sizes. 21
2.6 Elman RNN pseudo-dynamic Bayesian network. 22
2.7 Example bi-directional RNN pseudo-dynamic Bayesian network. 22
2.8 Basic transformer encoder architecture. 25
2.9 Conditional neural process. 30
2.10 Word2vec. 34
2.11 Example ROC curves. 41

3.1 Minerva-R. 50
3.2 Minerva-RP. 51
3.3 Minerva-RPE. 54
3.4 Minerva-RPCE. 54
3.5 Comparison of Minerva-RPES with 2-layer stacked self-attention. . . . 56

4.1 Distribution of phones within the TIMIT data set. 61
4.2 Proportion of different classes and class combinations in the GoEmotions dataset. There is imbalance in the classes, but the proportions are similar across the training, development and test sets. 62
4.3 The proportion of data for different correctness ranges in the CPC2 training data. Very high and very low correctness are common; middling values are rarer. 64
4.4 Performance and optimal β for different exemplar set sizes. 68

4.5 Comparison of classification accuracy on the TIMIT test set by phonetic class for the Minerva-R model using HuBERT features, with and without stratification of the exemplars by class. The dashed lines are the overall model accuracies. 70
4.6 Comparison of the confusion matrices of incorrectly classified frames using the non-stratified (top) and stratified (bottom) Minerva-R models trained on TIMIT. The colour represents the proportion of the true class that has been misclassified as the predicted class. 71
4.7 Performance of the Minerva-RP model with separate feature transforms, for different feature transform dimensions. 74
4.8 Comparison of classification accuracy on the TIMIT test set by phonetic class for the Minerva-RP models using HuBERT features, with and without stratification of the exemplars by class. The dashed lines are the overall model accuracies. 78
4.9 Comparison of classification accuracy on the TIMIT test set by phonetic class for the Minerva-RP and Minerva-RPE models using Mel spectrogram features. The dashed lines are the overall model accuracies. . . . 81
4.10 Heat map showing the annotated class and learned exemplar class of exemplars for which they are not the same, for the TIMIT Minerva-RPE model using Mel spectrogram features. 82
4.11 Minerva-RPE exemplar correctness vs learned exemplar labels. 83
4.12 Heat map representation of the covariance matrix for the learned ‘true’ class representations for the TIMIT Minerva-RPC model trained on Mel spectrogram features 87
4.13 Heat map representation of the covariance matrix for the learned ‘true’ class representations for the TIMIT Minerva-RPC model trained on Mel spectrogram features 88
4.14 Heat map representation of the covariance matrix for the learned ‘true’ class representations for the TIMIT Minerva-RPC model trained on HuBERT features. 89
4.15 Comparison of classification accuracy on the TIMIT test set by phonetic class for the Minerva-RP and Minerva-RPC models using Mel spectrogram features. The dashed lines are the overall model accuracies. . . . 90
4.16 Heat map representations of the similarities between the learned ‘true’ class representations for the Minerva-RPC model trained on GoEmotions with LSA, Word2vec and SBERT features showing a. all class similarities; and b. only cross-class similarities. 91

4.17 Performance of Minerva-RPC models on TIMIT test set with either: system combination of models with different exemplar sets; or increased exemplar set size. 95
4.18 Performance of Minerva-RPC models on the GoEmotions test set with either: system combination of models with different exemplar sets; or increased exemplar set size. 97
4.19 Performance of Minerva-RP models on the CPC2 test set with either: system combination of models with different exemplar sets; or increased exemplar set size. 99
4.20 Accuracy of Minerva-RPES on the TIMIT task with HuBERT features and increasing mid, backward and forward context. 105
4.21 Comparison of classification accuracy on the TIMIT test set by phonetic class for the Minerva-RPE and Minerva-RPES models using HuBERT features. The dashed lines are the overall model accuracies. 105
4.22 Confusion matrices for the Minerva-RPE and Minerva-RPES models, showing only frames where the models disagreed. 106

5.1 BLSTM system combination model architecture for Speech Intelligibility (SI) prediction. 115
5.2 Model performance by true correctness, for the BLSTM and FFNN/Minerva-RPE ensemble models. 119
5.3 Model performance by mean hearing aid system correctness. 120
5.4 Performance of proposed prediction system depending on enhancement system 120
5.5 Learned weights for the primary model Whisper decoder layers. 121
5.6 Comparison of performance of CPC2 entries. A * denotes an intrusive system. 122

List of Tables

1.1 Mathematical notation 7

2.1 Confusion matrix. 39

3.1 Minerva model descriptions. 50

4.1 Frame-based phone classification performance on the TIMIT development and tests sets using the Minerva-R and Minerva-RP models, with baseline Feed-Forward Neural Network (FFNN) for comparison, using different feature representations. Standard deviations are given in parentheses. The best result is in bold. 75

4.2 Performance on the GoEmotions task using different feature representations. Standard deviations are given in parentheses. The best result is in bold. 76

4.3 Performance on the CPC2 task using Minerva-R and Minerva-RP models, with baseline FFNN for comparison, using different feature representations. Standard deviations are given in parentheses. The best result is in bold. 77

4.4 Frame-based phone classification on TIMIT using Minerva-RP and Minerva-RPE models. Standard deviations are given in parentheses. The best result is in bold. 80

4.5 Emotion classification of text on GoEmotions using Minerva-RP and Minerva-RPE models. Standard deviations are given in parentheses. The best result is in bold. 80

4.6 Speech intelligibility prediction on CPC2 using Minerva-RP and Minerva-RPE models. Standard deviations are given in parentheses. The best result is in bold. 81

4.7 Frame-based phone classification on TIMIT with learned class representations. Standard deviations are given in parentheses. The best result is in bold. 85
4.8 Emotion classification of text on GoEmotions using Minerva models. Standard deviations are given in parentheses. The best result is in bold. 92
4.9 Frame-based phone classification on TIMIT using Minerva-RP and Minerva-RPC models, with fixed and unfixed exemplars. Standard deviations are given in parentheses. 96
4.10 Frame-based phone classification on GoEmotions using Minerva-RP and Minerva-RPC models, with fixed and unfixed exemplars. Standard deviations are given in parentheses. 98
4.11 Speech intelligibility prediction on CPC2 using Minerva-RP models, with fixed, unfixed and adaptive exemplars. Standard deviations are given in parentheses. 98
4.12 Frame-based phone classification on TIMIT using Minerva-RP models, with fixed, unfixed and adaptive exemplars. Standard deviations are given in parentheses. 101
4.13 Frame-based phone classification on TIMIT using Minerva-RPC models, with fixed, unfixed and adaptive exemplars. Standard deviations are given in parentheses. 102
4.14 Speech intelligibility prediction on CPC2 using Minerva-RP models, with fixed, unfixed and adaptive exemplars. Standard deviations are given in parentheses. 103
4.15 Frame-based phone classification on TIMIT using the sequence-based Minerva-RPES model. Standard deviations are given in parentheses. The result is in bold. 104
4.16 Performance of models on the Minerva-Transformer spectrum, using Mel spectrogram and Hubert features. Standard deviations are given in parentheses. 108

5.1 Model performance on the validation and evaluation splits. The best result is in bold. 117

A.1 Starting values for hyperparameter reduced grid search. 145
A.2 Tuned hyperparameter values for the TIMIT FFNN and Minerva-RP models (Table 4.1). 146
A.3 Tuned hyperparameter values for the GoEmotions FFNN and Minerva-RP models (Table 4.2). 146

A.4 Tuned hyperparameter values for the CPC2 FFNN and Minerva-RP models (Table 4.3.) 147
A.5 Tuned hyperparameter values for the TIMIT Minerva-RPE models (Table 4.4). 147
A.6 Tuned hyperparameter values for the GoEmotions Minerva-RPE models (Table 4.5). 147
A.7 Tuned hyperparameter values for the CPC2 Minerva-RPE models (Table 4.6). 148
A.8 Tuned hyperparameter values for the TIMIT Minerva-RPC and Minerva-RPCE models (Table 4.7). 148
A.9 Tuned hyperparameter values for the GoEmotions Minerva-RPC and Minerva-RPCE models (Table 4.8). 148
A.10 Tuned hyperparameter values for the TIMIT Minerva-RP and Minerva-RPC models with unfixed exemplars (Table 4.9). 149
A.11 Tuned hyperparameter values for the GoEmotions Minerva-RP and Minerva-RPC models with unfixed exemplars (Table 4.10). 149
A.12 Tuned hyperparameter values for the CPC2 Minerva-RP models with unfixed exemplars (Table 4.11). 149
A.13 Tuned hyperparameter values for the TIMIT Minerva-RP models with adaptive exemplars (Table 4.12). 150
A.14 Tuned hyperparameter values for the TIMIT Minerva-RPC models with adaptive exemplars (Table 4.13). 150
A.15 Tuned hyperparameter values for the CPC2 Minerva-RP models with adaptive exemplars (Table 4.14). 150
A.16 Tuned hyperparameter values for the TIMIT Minerva-RPES models (Table 4.15). 151
A.17 Tuned hyperparameter values for the TIMIT Minerva-RPES and transformer models with Mel spectrogram features (Table 4.16). 151
A.18 Tuned hyperparameter values for the TIMIT Minerva-RPES and transformer models with HuBERT features (Table 4.16). 151

Chapter 1

Introduction

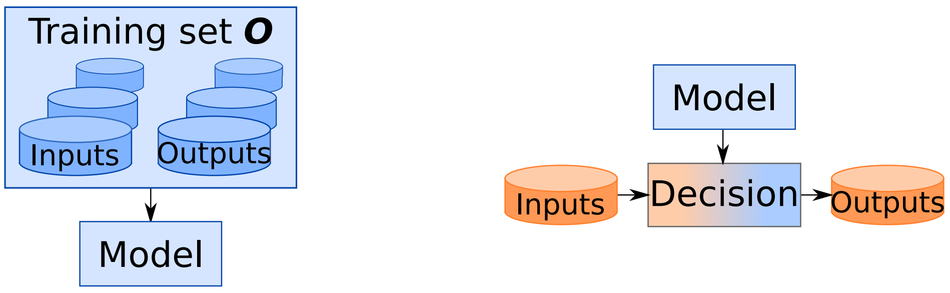
1.1 Background

The history of deep learning is littered with examples of cognitively-inspired architectures and frameworks. Most famously, Artificial Neural Networks (ANNs) are based on (and named for) the neural structures found in the brain. McCulloch & Pitts (1943) proposed simplified artificial neurons in an attempt to explain and understand the human brain, and the perceptron, upon which much of deep learning is based, was created to explore theories of brain function (Rosenblatt 1958). The Rectified Linear Unit (ReLU) activation, widely used in modern ANNs, is based on biological neuron function (Fukushima 1969). More sophisticated architectures are also cognitively inspired: the Neocognitron, an image processing network proposed by Fukushima (1980), and a forerunner of Convolutional Neural Networks (CNNs), is based on the eyes of cats (Hubel & Wiesel 1962, 1965). The attention mechanism that forms the core of the transformer architecture (Vaswani et al. 2017) is named for *attentional control*, which has been widely studied in humans (Cherry 1953, Schneider & Shiffrin 1977, Shiffrin & Schneider 1977). More general principals can also be traced back to theories of cognition: the Hebbian theory of learning (Do 1949), which has been summarised as "neurons that fire together wire together" has been used as biological motivation for unsupervised learning.

Since humans still outperform machines at many speech and language tasks, there is clearly more that can be learned from human cognition. One aspect of cognition that has not been widely explored in machine learning is that of memory, and in particular, memories of specific experiences. Modern deep learning models frequently use previous examples for training, but these examples are rarely retained for inference. Such examples are referred to as *exemplars*, and form the focus of this thesis.

1.2 Exemplar and prototype models

In the field of human psychology, the terms *prototype* and *exemplar* are used to describe two different approaches to concept categorisation (Medin & Coley 1998). In a prototype approach (Figure 1.1), examples of a category are used to construct a *prototype*, which can be thought of as a model or summary of the category's features. The features of new items to be categorised are assessed with respect to the features of the category, as summarised by the prototype. Only the prototype is used for decision-making; the examples used to construct it are discarded. As an example, a robin might be classified as a bird because the class "bird" includes features such a flight, singing, wings and a beak. The robin therefore has very similar features to the class.



(a) Training (b) Evaluation

Figure 1.1: *Prototype approach.*

In contrast, an *exemplar* approach makes direct use of category examples, or exemplars, so that a new item is categorised based on its similarity to known members of a category (Figure 1.2). No model or summary is constructed; all decision-making is based on the exemplars themselves. In this case, the robin might be classed as a bird because it is similar to a previously seen sparrow, raven and starling that are known to belong to the class "bird".

The same terms can be applied to models in machine learning. Models such as K-nearest neighbour directly use exemplars at decision time, and are exemplar models. ANNs, on the other hand, learn their parameters from data during training, but make no other use of the training data at inference; they are prototype models.

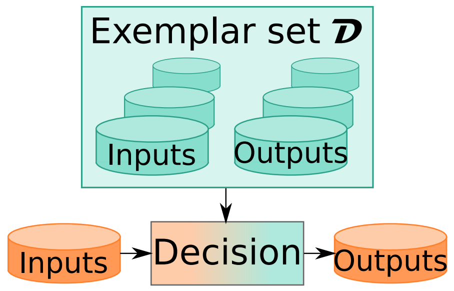


Figure 1.2: *Exemplar approach.*

Experiments suggest that humans may not exclusively use either prototype or exemplar approaches. Nosofsky et al. (1994) suggest that human categorisation is largely rule-based, but with specific exceptions (exemplars) also being referred to. Erickson & Kruschke (1998) propose a more balanced approach, with transitions between prototype and exemplar-based approaches. Rouder & Ratcliff (2006) concluded that humans use an exemplar-based approach when the objects to be classified are clearly distinct, but a prototype-based approach when they are easily confused, for example, classifying lines as “long” or “short” is more likely to be prototype-based, whereas classifying them as “red” or “yellow” is more likely to be exemplar-based. Natal et al. (2013) found that the choice of exemplar or prototype-based approach might vary depending on the person. More recently, researchers used techniques such as functional Magnetic Resonance Imaging (fMRI) to determine which approach is used. Mack et al. (2013) found evidence for a largely exemplar-based approach; Bowman & Zeithamova (2018) found results consistent with a prototype approach. The literature is therefore mixed with regard to *when* exemplar and prototype approaches are used, but there is evidence that humans use both prototype and exemplar-based approaches at least some of the time. This is not reflected in automated speech and language tasks, in which data-driven, parametric, prototype-based deep learning approaches are overwhelmingly more popular.

Exemplar approaches in machine learning do exist and have some benefits. They are interpretable, since it is generally simple to see which exemplars are used for each decision. They can also be used for limited data; theoretically, only a single exemplar for each class is required in order to perform classification, for example. The exemplar set can in some cases be changed at decision time with little computational cost, so that new data can be easily incorporated. Exemplar approaches are not typically scalable, however, so while they may be effective for small data sets, they are rarely used for large-scale data. This inability to scale leads to poor performance compared with modern, data-driven approaches.

As with humans, hybrid approaches, which make use of both the exemplar approach and the prototype approach, may be a better option that either alone. This has been explored to some extent: incorporating a memory into language models has shown promise (Wu et al. 2022), and Conditional Neural Processes (CNPs) are a form of hybrid models that has been demonstrated to work for both classification and regression (Garnelo et al. 2018).

The objective of this PhD is to explore the use of exemplars for speech and language tasks, drawing inspiration from theories of human cognition. Minerva 2 is an exemplar-based simulated memory model proposed by Hintzman (1984, 1986), and intended to test the exemplar theory of categorisation. It has been in use for decades, and has been compared against numerous human subject studies. It has also previously been adapted for speech and language tasks, having been used for vowel classification (Maier & Moore 2005) and restricted word recognition Moore & Maier (2007). It therefore serves as a useful starting point for for this work.

1.3 Research questions

When proposing the perceptron, on which modern ANNs are based, Rosenblatt (1958) identified three fundamental questions regarding cognition:

1. how information is sensed or detected;
2. how it is stored; and
3. how it affects recognition and behaviour.

The perceptron was intended to explore the second and third of these questions, which are still relevant to ANNs today, and which form the basis of the following research questions.

RQ1 What are the similarities and differences between the exemplar model Minerva 2 and ANNs?

Prototype and exemplar models present different approaches to storing information. What are the underlying similarities and differences? Are there any circumstances under which they are equivalent?

RQ2 What exemplars should be stored in memory, and how should they be represented?

Minerva 2's exemplars set is crucial to its function. How big should it be? What form should the exemplars take? Theories of human cognition typically assume good feature representation, which is not always the case in machine learning. What effect does feature representation have on the performance of exemplar and prototype models? How does feature representation affect how exemplar and prototype models compare with each other?

RQ3 How can an exemplar model be combined with parameters to form a hybrid exemplar-prototype model? What benefits, if any, does this bring?

Since humans appear to use both exemplar and prototype approaches, how can Minerva 2 be adapted to make used of both? What benefits does this have, with respect to performance and interpretability?

1.4 Contributions

This thesis provides an answer to Research Question 1 in Chapter 3, with the following contributions:

- It is shown that Minerva 2 with a fixed memory is a constrained form of feed-forward neural network.
- A process for inference described by Minerva 2's creator, and referred to here as *echo-of-echoes*, is shown to be a fixed point process and a form of Deep Equilibrium Model.

Research Question 2 is addressed in Chapter 4, with the following contributions:

- Feature representation is shown to be crucial to the function of Minerva 2, with improved feature representation not just improving performance, but closing the gap between the performance of the exemplar-based Minerva 2 models and prototype models.
- Several parameterised variants of Minerva 2, combining both exemplar and prototype approaches, are proposed and compared with prototype and exemplar models.

Research Question 3 is covered in Chapter 5, with the following contribution:

- A hybrid exemplar-prototype model based on Minerva 2 is shown to provide state-of-the-art performance on a the Clarity Prediction Challenge 2 speech intelligibility prediction task.

1.5 Thesis outline

This thesis is divided into six chapters. Following this introduction, Chapter 2 describes Minerva 2 in more detail, as well as describing how it has been used both in the field of human psychology and for speech and language tasks. It also gives relevant background information on exemplar, prototype and hybrid models that have been used for speech and language tasks, and provides a brief overview of aspects of machine learning that will be relevant to later chapters.

Chapter 3 covers the theoretical framework. It presents the equivalence between Minerva 2 and various prototype models, as well as discussing implications for the exemplar set size and describing proposed parameterised variants of Minerva 2 that combing exemplar and prototype approaches into hybrid models.

Chapter 4 describes the experimental work that has been carried out, testing the theoretical conclusions from Chapter 3 and testing the proposed hybrid models on several speech and language tasks.

Chapter 5 describes the use of a hybrid model to perform speech intelligibility prediction on the Clarity Prediction Challenge 2 task, on which achieves state-of-the-art performance.

In Chapter 6, the findings of previous chapters are summarised, and avenues for further research are proposed.

1.6 Publications

Mogridge, R., Close, G., Sutherland, R., Hain, T., Barker, J., Goetze, S., & Ragni, A., (2024), "Non-intrusive speech intelligibility prediction for hearing-impaired users using intermediate asr features and human memory models", in: *2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, pp. 306–310.

Mogridge, R., Ragni, A., 2024. "Learning from memory-based models", in: *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH 2024*.

1.7 Notation

Table 1.1 shows the non-standard mathematical notation that will be used in this thesis.

Table 1.1: *Mathematical notation*

Notation	Example	Meaning
non-bold	a, A	scalar
bold lower case	$\boldsymbol{a}$	vector
bold upper case	$\boldsymbol{A}$	matrix
circle power	$\boldsymbol{a}^{\circ b}$, $\boldsymbol{A}^{\circ b}$	raise elements to the power b
tilde on vector	$\tilde{\boldsymbol{a}}$	L2 normalise vector $\boldsymbol{a}$
tilde on matrix	$\tilde{\boldsymbol{A}}$	L2 normalise each column-vector in $\boldsymbol{A}$

Chapter 2

Preliminaries

2.1 Overview

This chapter provides an overview of the global memory model Minerva 2, as well as covering more general aspects of exemplar and prototype approaches to machine learning that will be relevant for future chapters. The chapter begins with a detailed description of the Minerva 2 cognition model, along with a summary of prior work that has made use of it in the fields of human psychology and machine learning. There follows an overview of other exemplar, prototype and hybrid approaches in machine learning, including an overview of how such models are trained. Several options for features representation are described, both for speech and text, since exemplar representation is a key research question for this thesis. Finally, there is a summary of the evaluation metrics that will be used in later chapters.

2.2 Minerva 2

Minerva 2 is a global memory model proposed by Hintzman (1984), created to test theories of human cognition. Global memory models in general are intended to mimic and explain a range of different experimental results, rather than focusing on a specific task. Minerva 2 is an exemplar-based model which uses previous experiences, or exemplars, to label new experiences. Each new input label is generated from the labels of exemplars, weighted by how similar the new experience is to each of the previous experiences. The mechanism by which the new label is produced, shown in Figure 2.1, is a form of attention, although Minerva 2 predates the term "attention".

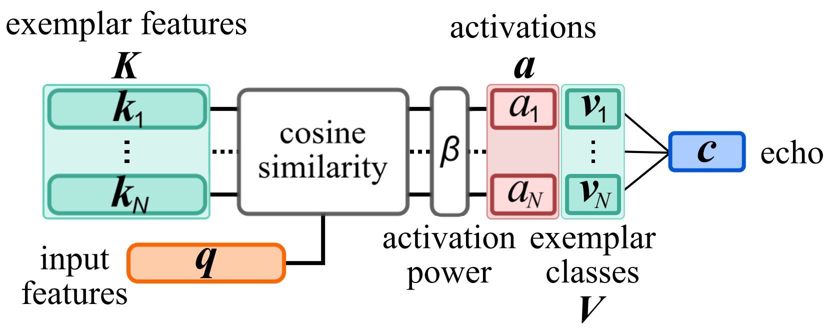


Figure 2.1: *Minerva 2.*

Let $\boldsymbol{q}$ be an input query, representing a new experience to be labelled. Each previous experience, or exemplar, is represented by a feature vector and a label vector. Let $\boldsymbol{K} = \begin{bmatrix}\boldsymbol{k}_1 & \dots & \boldsymbol{k}_N\end{bmatrix}$ be a matrix of column vectors, each of which represents an exemplar feature vector; in Figure 2.1, this is in green at the top left. Let $\boldsymbol{V} = \begin{bmatrix}\boldsymbol{v}_1 & \dots & \boldsymbol{v}_N\end{bmatrix}$ be matrix of column vectors representing the exemplar labels. In Figure 2.1, these are shown in green towards the right.

The elements of the query and exemplar vectors are restricted ± 1. The model also permits values of 0, meaning that the information is either irrelevant or not available. In Minerva 2's original form, the exemplar feature and label elements are in a single vector; for convenience, the feature and label information has been split into separate vectors. This is a notation change only, and does not affect the underlying model. The input query, exemplar features and exemplar labels are equivalent to the queries, keys and vectors in attention: to label a new query, $\boldsymbol{q}$, it is compared against each of the stored exemplar feature vectors, using a scaled form of dot product similarity,

$$\boldsymbol{s} = \frac{1}{F}\boldsymbol{K}^{\top}\boldsymbol{q}, \tag{2.1}$$

where F is the length of the query and exemplar feature vectors, ensuring that each element of $\boldsymbol{s}$ falls in the range $[-1, 1]$. The activation, $\boldsymbol{a}$, of the exemplars is a positively-accelerated function of the similarities, with Hintzman (1984) suggesting raising each element in the vector to an odd power β. The notation

$$\boldsymbol{a} = \boldsymbol{s}^{\circ\beta} \tag{2.2}$$

denotes raising each element of $\boldsymbol{s}$ to the power β. This form differs from conventional attention, in which a softmax function is used. The activation power helps prevent exemplars that are very similar to the probe from being drowned out by large numbers of exemplars with only limited similarity. In principle, any positively accelerated function that preserves the sign could be used. As with attention, the new label, $\boldsymbol{c}$, referred to as the echo, is generated as a weighted sum of the exemplar labels, with the activations as weights:

$$\boldsymbol{c} = \boldsymbol{V}\boldsymbol{a}. \tag{2.3}$$

Finally, the echo is normalised by dividing by its largest absolute element value, which is equivalent to the L_∞ norm,

$$\langle \boldsymbol{c} \rangle_\infty = \frac{\boldsymbol{c}}{||\boldsymbol{c}||_\infty}. \tag{2.4}$$

The *intensity* of the echo is a measure of how recognisable it is, and is associated with the human feeling of *familiarity*. It is given by,

$$I = \sum_{n=1}^{N} a_n. \tag{2.5}$$

While differing in some details, the overall mechanism of Minerva 2 is very similar to attention, except that in conventional self-attention, the queries, keys and values are all derived from the input. In Minerva 2, the exemplars, which make up the keys and values, are pre-defined and separate from the input.

2.2.1 Echo-of-echoes

Ideally, the normalised echo (Equation 2.4) will closely resemble a known class, but this is not guaranteed, leading to the problem of ambiguous recall. A class could be identified using a similarity or distance-based measure, comparing the normalised echo with some "true" class representation, but this is only possible if a true class representation is available. Instead, Hintzman (1988) suggests that normalising the echo and iterating the process of echo generation produces an 'echo-of-echoes' that closely resembles one of the classes. This process is shown in Figure 2.2.

Let $\langle \boldsymbol{c} \rangle_\infty^{(0)}$ be the initial normalised echo, as described in Equation 2.4. The ith echo-of-echo is compared to the exemplar class labels to give the similarity,

$$\boldsymbol{s}^{(i)} = \frac{1}{J}\boldsymbol{V}^\top \langle \boldsymbol{c} \rangle_\infty^{(i-1)} \tag{2.6}$$

$$\boldsymbol{a}^{(i)} = \boldsymbol{s}^{(i)^{\circ\beta}} \tag{2.7}$$

$$\boldsymbol{c}^{(i)} = \boldsymbol{V}\boldsymbol{a}^{(i)} \tag{2.8}$$

$$\langle \boldsymbol{c} \rangle_\infty^{(i)} = \frac{\boldsymbol{c}^{(i)}}{||\boldsymbol{c}^{(i)}||_\infty}. \tag{2.9}$$

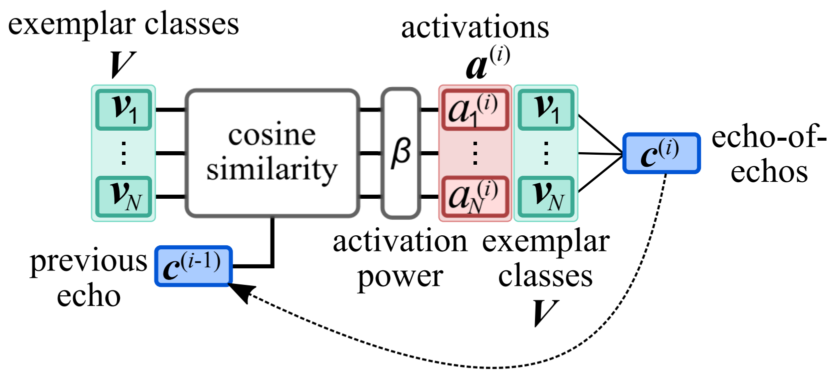


Figure 2.2: *Minerva 2 echo-of-echoes.*

Note that the ith echo is compared to the the exemplar *labels*, not the exemplar features in this process.

Hintzman (1988) found that iterating this process for any initial echo rapidly converges to echoes that closely resemble one of the exemplar classes, but Maier & Moore (2005) were unable to replicate this behaviour on a phone classification task. This process is examined more closely in Section 3.3.

2.2.2 Learning and forgetting

In Hintzman (1986), a learning and forgetting option for Minerva 2 was introduced, controlled by the learning parameter, p_L, and the forgetting parameter, p_F. During each learning cycle, there is a probability p_L that each element of the input is learned. Elements of the input that are not learned are set to zero. This means that, for $p_L < 1$, learning is imperfect. Similarly, during a forgetting cycle, there is a probability p_F that each element of any exemplars already in memory will be forgotten, and set to 0. For $p_F > 0$, memory degrades over time.

2.2.3 Prior work in human psychology

Minerva 2 has been used for decades in the field of human psychology, often in comparison with the results of human experiments. A brief overview of some of the work conducted is given here, with particular emphasis on speech and language related tasks.

2.2.3.1 Schema abstraction

Schema abstraction is the process of learning a class from examples (Elio & Anderson 1984). For example, a human may encounter a pigeon, a robin and a sparrow, and be told that each of them is a bird. From these examples, and others, they may learn about the category 'bird', and can use this knowledge to determine that a crow, which they have never seen before, is also a bird. This is contrasted with a rules-based approach to learning, in which they are explicitly told rules, such as that birds have wings and a beak. Hintzman (1986) showed that, like humans, Minerva 2 is able to perform schema abstraction, correctly classifying new inputs based on exemplars, without explicitly being told any rules.

2.2.3.2 True and false recognition

Humans may have a false sense of familiarity about concepts they have not previously encountered that are closely related to concepts that they *have* encountered. The Deese/Roediger—McDermott (DRM) associative memory illusion task (Deese 1959, Roediger & McDermott 1995) has been used to test this in human experiments. Participants are shown lists of words, with the following example list taken from Nick Reid & Jamieson (2023): BED, REST, AWAKE, TIRED, DREAM, WAKE, SNOOZE, BLANKET, DOZE, SLUMBER, SNORE, NAP, PEACE, YAWN, and DROWSY. Participants are then shown a separate list of words, some of which are on the previous list, and some which are not, including the "critical lure" SLEEP, which does *not* appear on the first list, but is highly associated with words that do. Typically, humans falsely recognise the critical lure as being on the original list at well above chance, despite it not appearing on the first list.

Minerva 2 has been shown to replicate this false sense of familiarity, with high intensity (Equation 2.5) for critical lures, as well as for items that actually appeared on the list. This was initially shown for a related task that made use of randomly simulated vectors to represent words (Arndt & Hirshman 1998, Gallo 2010). More recently, the true task was replicated by Nick Reid & Jamieson (2023), who made use of Latent Semantic Analysis (LSA) (see §2.7.1.1) to represent the words on each list as vectors.

2.2.3.3 Artificial grammar learning

In the artificial grammar learning task, proposed by Reber (1967), strings of characters are generated by a finite state grammar, such as that shown in Figure 2.3, which would allow the strings 'ABDA', 'CCB' and 'CDA', but not the string 'ACD'. Human participants are shown sets of strings. For the control group of participants, the strings are randomly constructed. For the test group of participants, the strings are constructed using the finite state grammar. The participants attempt to memorise these strings. Following this, participants are informed that the strings were constructed using rules, and are asked to sort new test strings according to whether they are grammatical or ungrammatical. Participants who were originally shown grammatical strings are better able to sort them than those who were shown random strings. Despite this, participants are typically unable to articulate the grammatical rules.

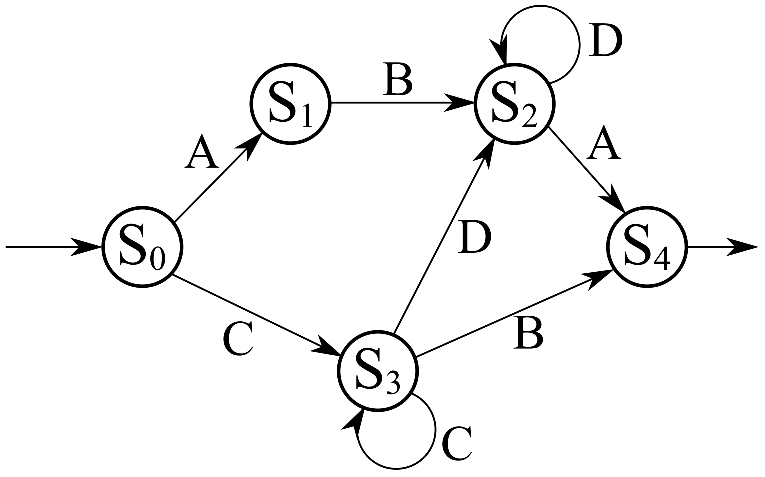


Figure 2.3: *Example of a finite state grammar.*

Jamieson & Mewhort (2009) was able to replicate this experiment using Minerva 2, replacing the strings with vectors, and using either random vectors for the example set (controls) or grammatical vectors (test group), with imperfect learning (see §2.2.2). They were able to closely replicate the results of the human experiment.

2.2.4 Minerva 2 for speech and language tasks

Although intended for psychology experiments, Minerva 2 has been used for speech and language tasks: Maier & Moore (2005) found that Minerva 2 showed promise on a vowel classification tasks using a restricted data set of 10 vowel sounds from 76 speakers (Peterson & Barney 1951). The model was later adapted to a restricted word corpus (Moore & Maier 2007), and the authors note that the exemplar-based model was able to retain fine phonetic detail, while prototype systems tend to generalise. This ties in with comments made by Sainath et al. (2012) in a review article, who noted that exemplar-based approaches are particularly useful for modelling rare events.

(Maier & Moore 2007) further adapted Minerva 2 into the Temporal Epsisodic Memory Model (TEMM) for speech recognition. This model uses sequential exemplars, allowing temporal information to be retained. The model was tested on a limited vocabulary speech recognition task, and while for matching training/test sets it performed less well than a comparison Hidden Markov Model (HMM), it was found to be more robust when the test set was less well matched.

2.3 Alternative memory models

Minerva 2 is one of several global memory models in the field of human psychology, all of which aim to mimic and explain results from multiple experiments. A selection of these models are described here.

The Generalised Context Model (GCM), proposed by Nosofsky (1986), is similar to Minerva 2, in that it compares an input to each of the exemplars held in memory. The comparison metric used is Minkowsky distance-based, rather than dot-product based. Given a set of classes $\mathcal{C}$, exemplars $\mathcal{D} = \{\boldsymbol{k}_n, v_n\}_{n=1}^{N}$ where $v_n \in \mathcal{C}$, and an input $\boldsymbol{q}$, the Minkowsky distance between $\boldsymbol{q}$ and each of the exemplars is given by,

$$d_n = \left(\sum_{f=1}^{F} |q_f - k_{fn}|^r \right)^{\frac{1}{r}}, \tag{2.10}$$

and the predicted probability that the class of $\boldsymbol{q}$ is $c \in \mathcal{C}$ is,

$$P(c|\boldsymbol{q}, \mathcal{D}) = \frac{b_c \sum_{v_n=c} e^{-d_n}}{\sum_{m\in\mathcal{C}} \left(b_m \sum_{v_n=m} e^{-d_n} \right)}, \tag{2.11}$$

where F is the dimension of $\boldsymbol{q}$ and the $\boldsymbol{k}_n$, r is the chosen power, and the b_c are class weights. The GCM class labels are discrete, rather than vector based as in Minerva 2, so it is fundamentally a classification model, rather than a regression model. The form of Equation 2.11 includes a softmax, so its output is bounded, unlike Minerva 2's.

There are several associative memory models, which focus on the associations between items, rather than on the items themselves. One example is Search of Associative Memory (SAM), proposed by Raaijmakers & Shiffrin (1981). It divides memory into *short term storage* (STS) and *long term storage* (LTS). STS contains specific items (exemplars) that can be recalled quickly and perfectly, but has a limited size and is overwritten with new input. LTS stores relationship information, rather than specific examples, with relationships based on the amount of time that items spend together in

STS. Let $\boldsymbol{K} = \{\boldsymbol{x}, \boldsymbol{y}\}_{n=1}^{N}$ be a set of paired data, where $\boldsymbol{x}$ will be referred to as a 'word' and $\boldsymbol{y}$ as its 'context'. STS can store the word-context pairs perfectly, until there are r in memory, at which point a new word-context pair will overwrite an existing pair at random. It is assumed that STS degrades over time; if there is a long wait, it will be empty. LTS is relationship-based. The strength of the relationship between two items is denoted $S(a, b)$. There are different types of relationship: $S(\boldsymbol{x}_n, \boldsymbol{y}_n)$ is the word-context relationship for the nth data pair, and is proportional to the time it spends in STS. $S(\{\boldsymbol{x}_n, \boldsymbol{y}_n\}, \{\boldsymbol{x}_m, \boldsymbol{y}_m\})$ is the word/context-word/context relationship for the nth and mth pairs, and is proportional to the time that the two pairs are *both* in STS. All word-context relationships and word/context pair relationships are held in LTS.

SAM has been used to explain differences and similarities between *recognition*, where an experience is perceived to be familiar, and *recall*, where a previous experience can be reproduced Clark (1992, 1995). For example, if shown a list of words which are then removed, recognition would be identifying that words were on the list if shown them; recall would be writing out the list.

Other associative memory models include: the Theory of Distributed Associative Memory (TODAM) model (Murdock 2014), which which uses convolutions of the paired vectors to create associations; and the matrix model (Pike 1984), which uses matrix multiplication, rather than convolutions, to create the associations.

Experimental and theoretical studies comparing some or all of these models have been conducted (Humphreys, Pike, Bain & Tehan 1989, Pike 1984, Humphreys, Bain & Pike 1989), with all models being imperfect and no clear evidence to select one model over the others. In the absence of a clear winner, Minerva 2 was selected as the primary model for study because it is simple and differentiable, it has been extensively used for the past four decades, it is still widely used today, and it has previously been used successfully for speech and language tasks.

2.4 Other exemplar approaches to speech and language tasks

While not as common at present, exemplar approaches have been widely used for speech and language tasks in the past. For classification tasks (such as phone recognition), exemplars can be used as templates for comparison, while for generative tasks (such as speech synthesis), they have been used as building blocks for constructing an output.

2.4.1 k-nearest neighbour

K-Nearest Neighbour (KNN) (Fix & Hodges 1951, Cover & Hart 1967) is one of the best-known exemplar approaches in machine learning. It is a purely exemplar-based, non-parametric algorithm. It has been used for a wide variety of speech and language tasks, including phone classification (Golipour & O'Shaughnessy 2009, Labiak & Livescu 2011), voice recognition Ranny (2016), speech emotion recognition (Venkata Subbarao et al. 2022) and voice conversion (Baas et al. 2023).

Let $\boldsymbol{q}$ be a feature vector, let the exemplar set be $\mathcal{D} = \{\boldsymbol{d}_n, \boldsymbol{v}_n\}_{n=1}^{N}$, where $\boldsymbol{d}_n$ is a feature vector and v_n is its corresponding label. Let $\alpha(\boldsymbol{q}, \boldsymbol{d}_n)$ be a distance metric, such as the Euclidean distance,

$$\alpha(\boldsymbol{q}, \boldsymbol{d}_n) = \sqrt{(\boldsymbol{q} - \boldsymbol{d}_n)^{\top}(\boldsymbol{q} - \boldsymbol{d}_n)}. \tag{2.12}$$

The k exemplars for which α is lowest are selected. The classification of $\boldsymbol{q}$ is then typically based on a majority vote of the k-nearest exemplars, for classification, or alternatively, the mean of their labels, for regression. Alternative distance or similarity measures can be used, such as cosine or dot-product similarity.

2.4.2 Dynamic time warping

Dynamic Time Warping (DTW) is a technique for comparing sequences of different lengths, which is able to account for signals being stretched or squashed. It has been used for exemplar-based Automatic Speech Recognition (ASR) (Vintsyuk 1968, Sakoe & Chiba 1978). The technique makes use of a library of labelled sound clips (the exemplars). Each new sound is classified by comparing it to the exemplars using the DTW similarity metric. While DTW was widely used in the 1970s and 80s, it was later largely replaced with data-driven prototype HMMs. Some work continued, however, with DTW used in conjunction with KNN to perform phone recognition (Golipour & O'Shaughnessy 2009, Wachter et al. 2007). The main drawback of DTW is its lack of scalability; hence, much of the work using DTW focused on techniques to reduce computational load. This typically fell into two categories: using less computationally expensive algorithms, such as approximate nearest neighbour (Arya & Mount 1993); and reducing the search space. Wachter et al. (2003) employed temporal information to reduce the search space. They recognised that, if the input sequence $\boldsymbol{Q}_{1:T}$ and an exemplar sequence $\boldsymbol{K}_{1:T'}$ are similar, then if exemplar $\boldsymbol{q}_{t'}$ is relevant to the input $\boldsymbol{y}_t$, then $\boldsymbol{k}_{t'+i}$ is likely to be relevant to $\boldsymbol{q}_{t+i}$ for reasonably small i. Their *time filter* algorithm used this assumption to reduce the search space to $< 1\%$ of its original size, while still maintaining a suitable selection of exemplars for consideration. Overall, they found that the baseline HMM system still outperformed the exemplar-based system, but that the two systems were complementary, achieving the best performance when used in combination. This work was conducted on a small vocabulary speech recognition task of approximately 1,000 words.

Additional reductions to the search space can be made using meta-information (Wachter et al. 2007, Demuynck et al. 2011), which may include: location of the exemplar within an utterance; phone duration; speaker ID; the word the phone/exemplar belongs to; and the environment type (Wachter et al. 2003, Demuynck et al. 2011).

2.4.3 Concatenative speech synthesis

In concatenative speech synthesis, short units of speech are joined together to create the desired speech. In diphone-based speech synthesis, units start in the middle of one phone, and transition to the middle of the next, so that they can be laid together like dominoes. Further processing is typically required to cover the 'seams', such as Pitch Synchronous Overlap and Add (PSOLA) (Charpentier & Stella 1986, Moulines & Charpentier 1990). An alternative, data-driven option is to select speech from a corpus using two objectives: the first is to match the speech unit to the desired speech; and the second is to match the speech unit to the previously selected unit, so that they concatenate smoothly. These, and other, options for concatenative speech synthesis have been widely used (Hunt & Black 1996, Rao et al. 2005, Sak et al. 2006, Kirchner et al. 2010, Tabet & Boughazi 2011, Khan & Chitode 2016).

2.5 Prototype approaches to speech and language tasks

While exemplar approaches have been widely used in the past for speech and language tasks, prototype methods are much more commonly used at present. In this section, several common prototype approaches are described, with particular emphasis on data driven, highly parametric ANN architectures, which show state-of-the-art performance on a wide variety of speech and language tasks.

The parameters of ANNs are typically initialised randomly, before being trained using an iterative process using a *loss function*. The loss function generally represents the task of interest, such that a low value for the loss function indicates good performance. Taking the gradient of the loss function with respect to each parameter allows the parameters to be updated. Each update tends to reduce the loss towards a local minimum.

Relevant architectures are described in §2.5.1 to §2.5.8, followed by a more detailed description of ANN training in §2.5.9.

2.5.1 Feed-forward neural network

The simplest form of ANN is a Feed-Forward Neural Network (FFNN), which consists of an input layer, one or more hidden layers, and an output layer. FFNNs can be used for both classification and regression. The ith layer of a FFNN is given by,

$$\boldsymbol{x}^{(i)} = \boldsymbol{\sigma}^{(i)}\left(\boldsymbol{W}^{(i)}\boldsymbol{x}^{(i-1)} + \boldsymbol{b}^{(i)}\right), \tag{2.13}$$

where the $\boldsymbol{\sigma}^{(i)}$ are non-linear activations, the $\boldsymbol{W}^{(i)}$ are the layer weights and the $\boldsymbol{b}^{(i)}$ are the layer biases. Activation functions are typically non-linear, bounded and differentiable. A wide range of activation functions are used, with common options being:

$$\sigma_{sig}(z) = \frac{1}{1+e^{-z}} \qquad \text{sigmoid} \tag{2.14}$$

$$\sigma_{ReLU}(z) = \begin{cases} z & z >= 0 \\ 0 & z < 0 \end{cases} \qquad \text{ReLU} \tag{2.15}$$

$$\sigma_{tanh}(z) = \frac{e^{x}-e^{-z}}{e^{z}+e^{-z}} \qquad \text{hyperbolic tangent.} \tag{2.16}$$

When performing classification, the final layer is typically W-dimensional, where W is the number of classes, and the final activation is typically a softmax,

$$\boldsymbol{\sigma}_{sm}(\boldsymbol{z})_w = \frac{e^{z_w}}{\sum_{n=1}^{W} e^{z_n}}. \tag{2.17}$$

The softmax output has the properties that each element falls in the range $(0, 1)$, and the outputs sum to 1. The output of a softmax function is generally assumed to represent a probability distribution over the classes, although this does not in general reflect the true confidence of the model. The predicted class is given by,

$$\hat{w} = \underset{w=1,\ldots,W}{argmax}\left(x_w^{(out)}\right). \tag{2.18}$$

For regression, the output layer has the same dimension as the desired prediction.

FFNNs are *function approximators*, and it has been shown that a FFNN with a single hidden layer and non-polynomial, bounded activations is capable of approximating a wide range of functions to an arbitrary degree of accuracy, given a sufficiently large hidden layer dimension (Hornik et al. 1989, Leshno et al. 1993). FFNNs, while simple, are used as building blocks for more sophisticated architectures.

2.5.2 Deep equilibrium models

Deep Equilibrium Models (DEMs), proposed by Bai et al. (2019), are infinite-depth ANNs which employ tied parameters. Referring to Equation 2.13, which gives the equation for a layer in a FFNN, a deep equilibrium model can be represented as,

$$\boldsymbol{x}^{(i)} = \boldsymbol{\sigma}\left(\boldsymbol{W}\boldsymbol{x}^{(i-1)} + \boldsymbol{b}\right) \qquad i = 0, 1, 2, ... \tag{2.19}$$

$$\boldsymbol{y} = \lim_{i\to\infty} \boldsymbol{x}^{(i)}, \tag{2.20}$$

where $\boldsymbol{x}^{(i)}$ is the output of the ith layer, and $\boldsymbol{y}$ is the final output of the DEM. DEMs make the empirically-supported assumption that the limit tends to converge (Bai et al. 2019). Practically, DEMs also use skip connections between layers, meaning that a better representation than Equation 2.19 is,

$$\boldsymbol{x}^{(i)} = \boldsymbol{\sigma}\left(\boldsymbol{W}_x\boldsymbol{x}^{(i-1)} + \boldsymbol{W}_0\boldsymbol{x}^{(0)} + \boldsymbol{b}\right). \tag{2.21}$$

While earlier work experimented with tied parameters for finite-depth ANNs (Dehghani et al. 2019), DEMs build on this by exploiting the fact that Equation 2.19 is a *fixed point equation.* Fixed point equations have the form,

$$\boldsymbol{x} = \boldsymbol{f}(\boldsymbol{x}), \tag{2.22}$$

and have been widely studied. Numerous iterative methods for solving fixed point equations are known (Hoffman & Frankel 2018). Crucially, Bai et al. (2019) demonstrate that the gradient required for backpropagation does not depend on the method used for finding the fixed point during the forward pass, nor does it depend on the outputs of each iteration. This makes backpropagation an efficient, single-step process, regardless of the number of iterations required to find the fixed point during the forward pass.

2.5.3 Convolutional neural network

CNNs were originally based on a simplified version of parts of the eye (Fukushima 1980) and have been extensively used for image recognition. A form of 1-dimensional CNN for speech recognition, called a Time Delay Neural Network (TDNN), was proposed by Waibel et al. (1989). An example of a CNN is shown in Figure 2.4. In the example shown, $\boldsymbol{h}_t$ depends directly on multiple observations $\boldsymbol{y}_{t-v}, ..., \boldsymbol{y}_t$.

Given an input $\boldsymbol{Y}_{1:T} = (\boldsymbol{y}_1, ..., \boldsymbol{y}_T)$, a TDNN with a single hidden layer and an output layer can be defined as follows:

$$\boldsymbol{h}_t = \boldsymbol{\sigma}\left(\sum_{i=0}^{I} \boldsymbol{W}^{(i)}\boldsymbol{y}_{t-i} + \boldsymbol{b}\right) \tag{2.23}$$

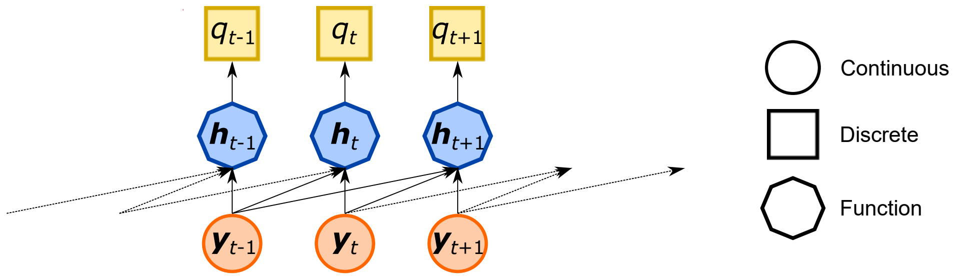
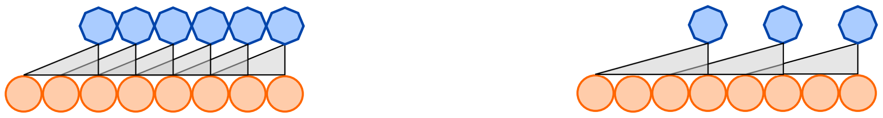


Figure 2.4: *Convolutional Neural Network.*

where the $\boldsymbol{W}_i$ are network weights and I is the size of the convolution window.



(a) stride 1, kernel size 3 (b) stride 2, kernel size 4

Figure 2.5: *Two examples of CNNs with different strides and kernel sizes.*

In a CNN, there need not be the same number of outputs as inputs. Figure 2.5 shows two examples of CNNs. Example (a) is similar to Figure 2.4, with each output of the convolutional layer depending on the previous 3 inputs, and one output for each input (excluding the first two inputs). Example (b) shows a different design, where each output depends on the previous 4 inputs, but there is only one output for every two inputs. The number of inputs between the outputs of the convolutional layer is known as the *stride*, and the range of the inputs that feed into each output of the convolutional layer is known as the *kernel size*. The example CNNs shown in Figures 2.4 and 2.5 are both causal, meaning that the output of the convolutional layer depends on previous observations, but not on future observations.

2.5.4 Recurrent neural network

Recurrent Neural Networks (RNNs) make use of recurrent units in order to handle variable-length sequences, such as speech. A simple form of RNN, first proposed by Elman (1990), uses a recurrent variable, $\boldsymbol{h}_t$, as shown in Figure 2.6, which depends on the history and the current observation.

The Elman architecture makes the assumption that the outputs are independent of each other, given previous and current observations. From Figure 2.6, it can be seen

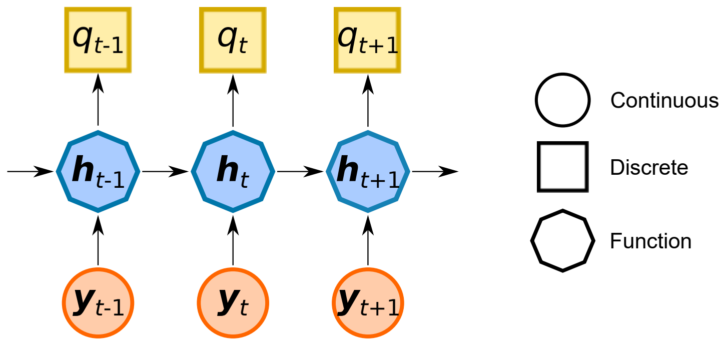


Figure 2.6: *Elman RNN pseudo-dynamic Bayesian network.*

that $\boldsymbol{q}_t$ depends on $\boldsymbol{Y}_{1:t}$ through the recurrent variable, $\boldsymbol{h}_t$, which is given by,

$$\boldsymbol{h}_t = \boldsymbol{\sigma}\left(\boldsymbol{A}\boldsymbol{h}_{t-1} + \boldsymbol{C}\boldsymbol{y}_t + \boldsymbol{b}\right) \tag{2.24}$$

where $\boldsymbol{A}, \boldsymbol{C}$ and $\boldsymbol{b}$ are parameters. The outputs can therefore depend on *all* previous observations through $\boldsymbol{h}_t$.

2.5.5 Bi-directional RNN

An alternative implementation of RNN is bi-directional, an example of which is shown in Figure 2.7. In this case, states depend on future observations, as well as past and current observations.

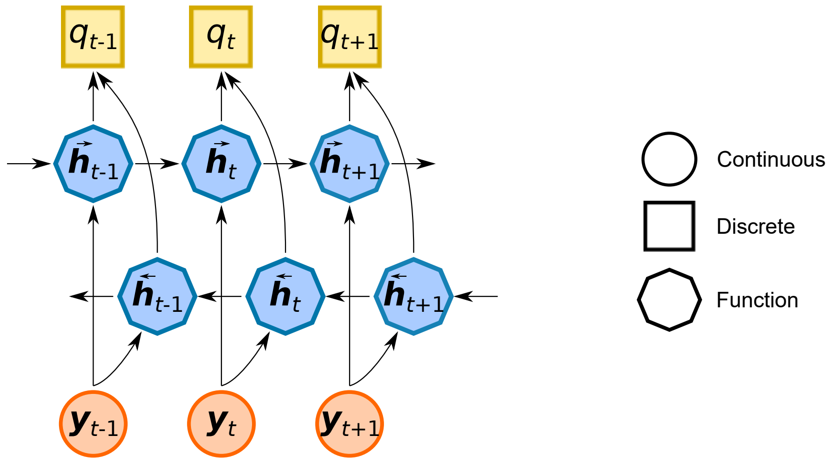


Figure 2.7: *Example bi-directional RNN pseudo-dynamic Bayesian network.*

The recurrent variable $\boldsymbol{h}_t$ incorporates both forward and backward recurrent elements,

$$\boldsymbol{h}_t = \begin{bmatrix} \overrightarrow{\boldsymbol{h}}_t \\ \overleftarrow{\boldsymbol{h}}_t \end{bmatrix} \tag{2.25}$$

$$\overrightarrow{\boldsymbol{h}}_t = \boldsymbol{\sigma}(\overrightarrow{\boldsymbol{A}}\overrightarrow{\boldsymbol{h}}_{t-1} + \overrightarrow{\boldsymbol{C}}\boldsymbol{y}_t + \overrightarrow{\boldsymbol{b}}) \tag{2.26}$$

$$\overleftarrow{\boldsymbol{h}}_t = \boldsymbol{\sigma}(\overleftarrow{\boldsymbol{A}}\overleftarrow{\boldsymbol{h}}_{t+1} + \overleftarrow{\boldsymbol{C}}\boldsymbol{y}_t + \overleftarrow{\boldsymbol{b}}) \tag{2.27}$$

This architecture requires access to future observations, which is not always available for streaming applications.

2.5.6 Long Short-Term Memory

RNNs suffer from vanishing gradient, in which back-propagating through long sequences results in the gradient diminishing to zero. The Long Short-Term Memory (LSTM) architecture was proposed by Hochreiter & Schmidhuber (1997) to solve this problem, which is done by means of *gates*. Each LSTM cell incorporates an *input gate*, an *output gate* and a *forget gate*. For an input $\boldsymbol{x}_t$,

$$\boldsymbol{f}_t = \boldsymbol{\sigma}_{sig}\left(\boldsymbol{W}_f\boldsymbol{x}_t + \boldsymbol{U}_f\boldsymbol{h}_{t-1} + \boldsymbol{b}_f\right) \quad \text{forget gate activation} \tag{2.28}$$

$$\boldsymbol{i}_t = \boldsymbol{\sigma}_{sig}\left(\boldsymbol{W}_i\boldsymbol{x}_t + \boldsymbol{U}_i\boldsymbol{h}_{t-1} + \boldsymbol{b}_i\right) \quad \text{input gate activation} \tag{2.29}$$

$$\boldsymbol{o}_t = \boldsymbol{\sigma}_{sig}\left(\boldsymbol{W}_o\boldsymbol{x}_t + \boldsymbol{U}_o\boldsymbol{h}_{t-1} + \boldsymbol{b}_o\right) \quad \text{output gate activation} \tag{2.30}$$

$$\tilde{\boldsymbol{c}}_t = \boldsymbol{\sigma}_c\left(\boldsymbol{W}_c\boldsymbol{x}_t + \boldsymbol{U}_c\boldsymbol{h}_{t-1} + \boldsymbol{b}_c\right) \quad \text{cell input activation} \tag{2.31}$$

$$\boldsymbol{c}_t = \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{t}_t \odot \tilde{\boldsymbol{c}}_t \quad \text{cell state vector} \tag{2.32}$$

$$\boldsymbol{h}_t = \boldsymbol{o}_t \odot \boldsymbol{\sigma}_h\left(\boldsymbol{c}_t\right) \quad \text{hidden state,} \tag{2.33}$$

where $\odot$ represents the Hadamard (elementwise) product. The function $\boldsymbol{\sigma}_{sig}$ is the sigmoid function (Equation 2.14), which acts elementwise, and which ensures that the elements of the activations of the input, output and forget gates fall in the range $(0, 1)$. The functions $\boldsymbol{\sigma}_c$ and $\boldsymbol{\sigma}_h$ are typically elementwise hyperbolic tangent functions (Equation 2.16)

LSTMs are highly parameterised, making them computationally intensive both during training and at inference. There are numerous variations of LSTM. Gated recurrent units (Cho et al. 2014) and minimal gated unit (Heck & Salem 2017) both seek to reduce the number of parameters, the first by removing the context vector and output gate, the second by unifying the input, output and forget gates. Peephole LSTM (Gers & Schmidhuber 2000, Gers et al. 2002) makes use of the cell state, rather than the hidden state. Bi-directional LSTM, similarly to bi-RNN, employs two LSTMs, to provide dependence on both backward and forward directions in a sequence (see Figure 2.7 for the RNN equivalent).

LSTMs have been used for a wide variety of sequence-based speech and language tasks, including ASR Graves & Schmidhuber (2005), Oruh et al. (2022), handwriting recognition (Graves & Schmidhuber 2008) and speech quality prediction (Tamm et al. 2022).

2.5.7 Transformer

Proposed by Vaswani et al. (2017) for machine translation, the basic transformer architecture employs layers of *attention*. Attention is a mechanism that produces weights by comparing an input, or *query*, to *keys*, and using their similarity to produce a weighted sum of associated *values*. There are variants of attention, but Vaswani et al. (2017) use scaled dot-produce attention. Letting $\boldsymbol{q}$ be the input query, and $\boldsymbol{K} = [\boldsymbol{k}_1, ..., \boldsymbol{k}_N]$ and $\boldsymbol{V} = [\boldsymbol{v}_1, ..., \boldsymbol{v}_N]$ be the keys and values respectively, scaled dot-product attention is given by,

$$attention(\boldsymbol{q}, \boldsymbol{K}, \boldsymbol{V}) = \boldsymbol{V}\boldsymbol{\sigma}_{sm}\left(\frac{\boldsymbol{K}^\top \boldsymbol{q}}{\sqrt{F}}\right) \tag{2.34}$$

where F is the dimension of $\boldsymbol{q}$ and each of the $\boldsymbol{k}_n$, and $\boldsymbol{\sigma}_{sm}$ is the softmax function (Equation 2.17).

The transformer architecture makes use of *self-attention*, in which the queries, keys and values are all generated from the input. Let $\boldsymbol{Q} = [\boldsymbol{q}_1, ..., \boldsymbol{q}_T]$ be a sequence input, and let $\boldsymbol{W}_q, \boldsymbol{W}_k$ and $\boldsymbol{W}_v$ be weight matrices of dimensions $D \times F$. Self attention is given by,

$$\boldsymbol{A}(\boldsymbol{Q}; \boldsymbol{W}_q, \boldsymbol{W}_k, \boldsymbol{W}_v) = attention(\boldsymbol{W}_q\boldsymbol{Q}, \boldsymbol{W}_k\boldsymbol{Q}, \boldsymbol{W}_v\boldsymbol{Q}) \tag{2.35}$$

where the softmax function $\boldsymbol{\sigma}_{sm}$ acts column-wise. In the transformer architecture, Vaswani et al. (2017) use a type of combined attention called *multi-head attention*, which uses several copies of scaled dot-product self-attention in parallel.

Multi-head attention is given by:

$$\boldsymbol{A}_{mh} = \boldsymbol{W}^{(o)} \begin{bmatrix} \boldsymbol{A}^{(1)}\left(\boldsymbol{Q}; \boldsymbol{W}_q^{(1)}, \boldsymbol{W}_k^{(1)}, \boldsymbol{W}_v^{(1)}\right) \\ \vdots \\ \boldsymbol{A}^{(H)}\left(\boldsymbol{Q}; \boldsymbol{W}_q^{(H)}, \boldsymbol{W}_k^{(H)}, \boldsymbol{W}_v^{(H)}\right) \end{bmatrix}, \tag{2.36}$$

where H is the total number of heads, $\boldsymbol{W}^{(o)}$ is a learned $HD \times HD$ weight matrix, and the $\boldsymbol{W}_q^{(h)}$, $\boldsymbol{W}_k^{(h)}$ and $\boldsymbol{W}_v^{(h)}$ are also learned parameters. Multi-head attention allows the model to attend to different aspects of the model in parallel. The computational complexity increases with the number of heads, so to keep the costs sufficiently low, the authors set $D = F/H$.

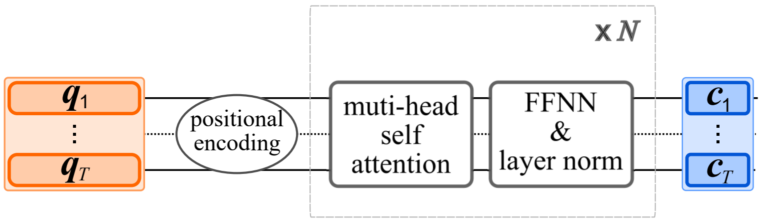


Figure 2.8: *Basic transformer encoder architecture.*

In practice, there are additional considerations. Figure 2.8 shows an example transformer encoder. Self-attention is invariant to permutations, meaning that it does not consider the order of the queries, keys or values. For this reason, a positional encoding is added to the input, so that positional information is included. The architecture also typically makes use of residual connections and layer normalisation (Lei Ba et al. 2016) following the multi-head attention layers, as well as intermediate FFNN layers. Vaswani et al. (2017) used an encoder-decoder architecture for sequence-to-sequence machine translation, and this architecture is common for other sequence-to-sequence tasks. The combination of multi-head attention and FFNN is typically repeated several times.

2.5.7.1 Positional encoding

Because self-attention ignores ordering, transformers typically make use of positional encoding, which is added to the input and provides information about where in a sequence each input is located. Positional encodings can be *absolute*, in which case the encoding is fixed with respect to the start of the sequence, or *relative*, in which case pair-wise relative distances between inputs in the sequence are preserved. Some positional encodings include parameters that are learned from data, such as that used by Gehring et al. (2017), but this is not required. Rotary Positional Encoding (RoPE) (Su et al. 2024) is an example of an unlearned relative positional encoding, as is the form of positional encoding used by Vaswani et al. (2017).

2.5.8 Bidirectional encoder representations from transformers

Bidirectional Encoder Representations from Transformers (BERT) is a language model proposed by Devlin et al. (2019), which has since been widely adapted and is an early type of Large Language Model (LLM). As the name suggests, BERT is a transformer-based encoder, which converts text tokens into fixed-length vectors, which can subsequently be used for downstream tasks. It uses self-supervised pre-training, and is typically trained using large amounts of unlabelled data. It can then be fine-tuned for specific tasks on labelled, task-specific data.

BERT uses a tokenizer to convert words (or sub-word units) to integers, followed by a projection layer that projects them into a dense embeddings space. This is followed by the transformer encoder, which produces contextualised embeddings. During pre-training, an additional task layer is included, but this not usually required for use. Instead, the contextualised embeddings produced by the transformer encoder are used for downstream tasks.

BERT takes as input one or two sentences (the term *sentence* here is used loosely; they need not be actual sentences), with a separation character between them. The output is a vector representing the first sentence as a whole, followed by a vector representing each word in the sentence, and then the same for the second sentence (if there is one). In this way, BERT can be used to generate representations of sequences of words as well as individual words.

Pretraining is performed with two tasks: masked language model; and next sentence prediction. In the masked language model task, input text sequences have a proportion of their words masked at random, so that the model cannot access them. The training objective is to predict them. In the next sentence task, sentences are given to the model in pairs, and the objective is to predict whether the second sentence follows on from the first in the corpus.

BERT word embeddings are contextual, which means that the embedding for *cat* in the sentence "the cat and the fiddle" will not be the same as the embedding for cat in the sentence "the cat on the roof", because they appear in different contexts.

Further work has improved on the original BERT. Liu (2019) produced Robustly Optimised BERT Pretraining Approach (RoBERTA) by changing training hyperparameters and experimenting with the best ways to select the paired sentences for training. Yang et al. (2019) introduced permuted language modelling, which allows the model to make use of dependencies for masked tokens preceding the current one in the sequence. Song et al. (2020) unified the masked language modelling of BERT and permuted language modelling to produce MPNet, which allows dependencies on both preceding and following masked tokens to be used.

2.5.9 Training artificial neural networks

Neural networks are typically trained iteratively on a dataset and task. For this overview, consideration is restricted to the supervised cases, where labelled training data is available.

2.5.9.1 Initialisation

The choice of initial values for ANN parameters can make a difference to how well and fast they train. Initialising from Gaussian or uniform distributions is common, but Glorot & Bengio (2010) propose choosing an Initialisation that preserves the input's variance through the layers during the first forward pass, while He et al. (2015) propose an Initialisation specifically for ReLU-type activations.

2.5.9.2 Training

Once initialised, an ANN is trained iteratively using training data. During the *forward pass*, the current parameters are used to make predictions based on the training data inputs. These predictions are compared with the training data labels using a *loss function*, also referred to as an *objective function*. Some examples of loss functions are given in §2.5.10.1. Model parameters are updated during a *backward pass*, using an optimiser such as Stochastic Gradient Descent (SGD), RProp (Riedmiller & Braun 1992) or Adam (Diederik 2014). Optimisers use the gradient of the loss function with respect to each parameter to update the parameters, and different loss functions do this in slightly different ways. Differentiation and backpropagation are typically handled by machine learning packages such as the Pytorch (Paszke et al. 2019) package for Python.

Training is typically performed by splitting the training data into *minibatches*, with parameter updates following each minibatch. Multiple passes through the training data may be used, with each pass referred to as an *epoch*. Training will ideally be terminated at a point where the model a) performs well on the training data and b) generalises well to unseen data. Training for too long can result in over-fitting, where the model performs extremely well on the training data, but is unable to generalise to unseen data. For this reason, a validation set is generally kept separate from the training data. Performance on the validation set can be tracked during training, and training can be terminated at an appropriate point. Training may also be terminated due to resource constraints, since training of larger models can be computationally expensive. Keeping both training data and validation data separate from the final test data is crucial.

2.5.9.3 Regularisation

Regularisation helps to prevent over-fitting. Two forms of regularisation are considered here: L2 regularisation, and dropout. L2 regularisation, also referred to as weight decay, adds a term to the training loss function that punishes large values for the model's parameters,

$$\mathcal{L}_{L2} = \lambda \sum_{\theta \in \boldsymbol{\Theta}} \theta^2, \tag{2.37}$$

where $\boldsymbol{\Theta}$ represents all the model's parameters, and λ is a hyperparameter (see §2.5.10). Reducing large values leads to a more uniform distribution of parameters, and therefore a less complex model.

Dropout acts during training, forcing the model to randomly ignore the outputs of some neurons. If $\boldsymbol{x}$ is the output of a neural network layer, for example, dropout works by setting each element of $\boldsymbol{x}$ to zero with probability p,

$$P(x_i = 0) = p. \tag{2.38}$$

Dropout acts only during training, preventing the model from relying too heavily on specific neurons. During inference, $p = 0$ and all information is retained.

2.5.10 Hyperparameter tuning

Hyperparameters are parameters that are used to configure an ANN, without being used at inference. They include the *learning rate*, which controls how fast model weights are changed; *minibatch size*; *weight decay*, which controls L2 regularisation; *dropout* and potentially others. Selecting hyperparameters is non-trivial, and a trial-and-error process is often required. The validation set can be used to assess the effects of different hyperparameters. For small models, gridsearch may be possible, where each hyperparameter has several options, and all possible combinations of hyperparameter options are tested. This is generally impractical for large models, however, due to the time and hardware required.

2.5.10.1 Loss functions

Loss functions must be suitable for the training task, with many options available. Three common loss functions for different tasks are described here.

For binary classification, given training data $\mathcal{T} = \{\boldsymbol{q}_i, w_i\}_{i=1}^{I}$, where $w_i \in \{0, 1\}$, and where the model output is assumed to be $y_i = P(w_i = 1|\boldsymbol{q}_i; \boldsymbol{\Theta})$, binary cross entropy loss can be used,

$$\mathcal{L}_{BCE}(\boldsymbol{\Theta}|\mathcal{T}) = -\frac{1}{I}\sum_{i=1}^{I} w_i \log P(w_i = 1|\boldsymbol{q}_i; \boldsymbol{\Theta}) - (w_i - 1)\log P(w_i = 0|\boldsymbol{q}_i; \boldsymbol{\Theta}) \tag{2.39}$$

For multi-class classification, where $\boldsymbol{w}_i \in \{1, ..., W\}$ and where W is the number of classes, categorical cross-entropy is commonly used as the objective function,

$$\mathcal{L}_{CE}(\boldsymbol{\Theta}|\mathcal{T}) = -\frac{1}{I}\sum_{i=1}^{I} \log P(w_i|\boldsymbol{q}_i; \boldsymbol{\Theta}). \tag{2.40}$$

For regression, given training data $\mathcal{T} = \{\boldsymbol{q}_i, \boldsymbol{w}_i\}_{i=1}^{I}$, Mean Squared Error (MSE) loss can be used,

$$\mathcal{L}_{MSE}(\boldsymbol{\Theta}|\mathcal{T}) = \frac{1}{IJ}\sum_{i=1}^{I} (\hat{\boldsymbol{w}}_i - \boldsymbol{w}_i)(\hat{\boldsymbol{w}}_i - \boldsymbol{w}_i)^{\top}. \tag{2.41}$$

where $\hat{\boldsymbol{w}}_i$ is the model's prediction of $\boldsymbol{w}_i$, and both have dimension J.

Loss functions are typically used in conjunction with L2 regularisation, as described in Equation 2.37.

2.6 Hybrid approaches to speech and language tasks

So far, the models described have been entirely exemplar-based or entirely prototype-based, but some models incorporate elements of both. Some examples are discussed here.

2.6.1 Conditional neural processes

First proposed by Garnelo et al. (2018), Conditional Neural Processs (CNPs) are related to stochastic processes in that they represent distributions over functions, conditioned on observations. In the case of CNPs, the distribution over functions represents a mapping from the inputs to the outputs. It is parameterised by an ANN, and conditioned on an exemplar set, $\mathcal{D}$. Figure 2.9 shows a graphical representation of a CNP.

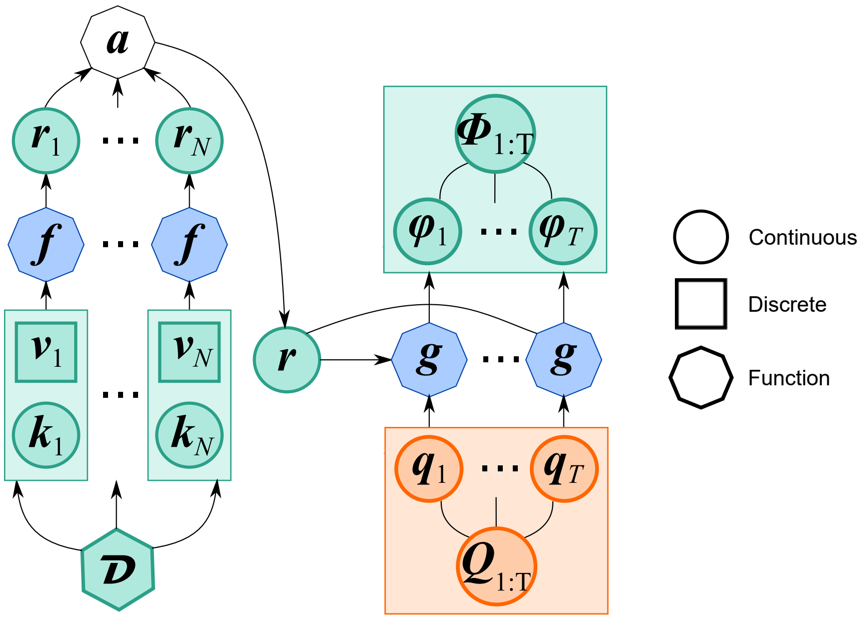


Figure 2.9: *Conditional neural process.*

Let $\mathcal{D} = \{\boldsymbol{k}_n, \boldsymbol{v}_n\}_{n=1}^{N}$ be the exemplar set, where the $\boldsymbol{k}_n$ are exemplar features and the $\boldsymbol{v}_n$ the corresponding labels. Let $\boldsymbol{Q}_{1:T} = [\boldsymbol{q}_1, ..., \boldsymbol{q}_T]$ be the inputs and $\boldsymbol{W}_{1:T} = [\boldsymbol{w}_1, ..., \boldsymbol{w}_T]$ be the corresponding target outputs. CNPs assume invariance to permutations in the inputs; changing the order of the inputs changes the order of the outputs, but has no other effect. CNPs are likewise invariant to permutations in the exemplars. Garnelo et al. (2018) ensure invariance by assuming a factored distribution,

$$P(\boldsymbol{W}_{1:T}|\boldsymbol{Q}_{1:I}, \mathcal{D}; \boldsymbol{\Theta}) = \prod_{i=1}^{I} P(\boldsymbol{w}_i|\boldsymbol{q}_i, \mathcal{D}; \boldsymbol{\Theta}), \tag{2.42}$$

although they note that this is merely an easy way, not the only way, of achieving the necessary invariance.

CNPs do not condition directly on the exemplars, but instead assume that $\mathcal{D}$ can be represented by a fixed-length embedding $\boldsymbol{r}$, as shown in Figure 2.9. Applying this assumption, Equation 2.42 can be rewritten as,

$$P(\boldsymbol{W}_{1:T}|\boldsymbol{Q}_{1:T}, \mathcal{D}; \boldsymbol{\Theta}) = \prod_{i=1}^{I} P(\boldsymbol{w}_i|\boldsymbol{q}_i, \boldsymbol{r}; \boldsymbol{\Theta}), \tag{2.43}$$

where

$$\boldsymbol{r}_n = \boldsymbol{f}(\boldsymbol{k}_n, \boldsymbol{v}_n; \boldsymbol{\Theta}_f) \qquad (\boldsymbol{k}_n, \boldsymbol{v}_n) \in \mathcal{D} \tag{2.44}$$

$$\boldsymbol{r} = \boldsymbol{r}_1 \oplus \boldsymbol{r}_2 \oplus \ldots \oplus \boldsymbol{r}_N. \tag{2.45}$$

The $\boldsymbol{r}_n$ can be considered embeddings for each of the individual exemplars $(\boldsymbol{q}_n, \boldsymbol{v}_n)$ and $\oplus$ represents an element-wise operation, such as the element-wise mean, to aggregate the individual embeddings into a combined exemplar set embedding for $\mathcal{D}$. This combined embedding is then used to obtain parameters, $\boldsymbol{\phi}_t$, for the distributions of the $\boldsymbol{q}_t$,

$$P(\boldsymbol{W}_{1:T}|\boldsymbol{Q}_{1:T}, \mathcal{D}; \boldsymbol{\Theta}) = \prod_{t=1}^{T} P(\boldsymbol{w}_t; \boldsymbol{\phi}_t) \tag{2.46}$$

where

$$\boldsymbol{\phi}_t = \boldsymbol{g}\left(\boldsymbol{q}_t, \boldsymbol{r}; \boldsymbol{\Theta}_g\right) \qquad t = 1, ..., T. \tag{2.47}$$

The functions $\boldsymbol{f}$ and $\boldsymbol{g}$ are ANNs, and their parameters, $\boldsymbol{\Theta}_f$ and $\boldsymbol{\Theta}_g$, are learned during training, forming the prototype part of the hybrid model.

The choice of distribution to parameterise with the $\boldsymbol{\phi}_t$ will depend on the task. For a regression task, for example, $\boldsymbol{\phi}_t = (\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$ could parameterise a Gaussian, $\mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$, for each target output. For a classification task, $\boldsymbol{\phi}_t$ could parameterise the logits of the class probabilities.

2.6.1.1 Exemplar selection

The exemplars have been represented as being distinct from the input, but in practice, CNPs typically condition on a subset of the input, that is $\mathcal{D} \subset \mathcal{Q}$ where $\mathcal{Q} = \{(\boldsymbol{q}_t, \boldsymbol{w}_t)\}_{t=1}^{T}$. This makes CNPs effective for generative tasks, such as image reconstruction, where a subset of the image's pixels are available for inference, as demonstrated by Garnelo et al. (2018).

2.6.1.2 Training

Let $\mathcal{T} = \{\boldsymbol{q}_n, \boldsymbol{v}_n\}_{n=1}^{N}$ be the training data. At each step in the training process, a random subset of the input is selected to be the exemplar set. The loss function is the negative log probability of the outputs, given the inputs and the exemplars,

$$\mathcal{L}(\boldsymbol{\Theta}_f, \boldsymbol{\Theta}_g|\mathcal{T}, \mathcal{D}) = -\sum_{n=1}^{N} \log P\left(\boldsymbol{w}_n|\boldsymbol{q}_n, \boldsymbol{r}; \boldsymbol{\Theta}_f, \boldsymbol{\Theta}_g\right). \tag{2.48}$$

Note that this loss function scores the model on its predictions for *all* the elements of $\mathcal{T}$, whether they fall into the exemplar set or not.

2.6.2 Adding memory to parameterised models

Most ANN architectures distil the information in training data into the parameters of the model, rather than using exemplars, and as such are purely prototype models. Recently, adding memory to ANNs, particularly for language modelling, has been explored with some success. Wu et al. (2022) experimented with including non-parametric memory (effectively an exemplar set) into transformer-based language models. The exemplar set in this case is made up of previous word embeddings generated by the model (which incorporate local context). As new embeddings are generated, they are added to the exemplar set. The model uses approximate KNN to identify relevant exemplars. Zhong et al. (2022) and Wang et al. (2024) incorporate the use of memory into training. The addition of the non-parametric memory improves the performance, which the authors believe is due to the model's ability to include long-term context.

2.7 Feature representation

Evidence from the field of human psychology suggests that humans use sophisticated representations for speech and language that include acoustic, phonetic and semantic information, as well as being highly contextualised (Repp 1982, Stanovich & West 1983, Pisoni et al. 1985). Recent studies utilising fMRI to perform scans of the human brain support this conclusion, with evidence that representations are detailed and contextualised (Viganò et al. 2021, Jamali et al. 2024). In contrast, feature representations in machine learning may not be as effective. Representing the information contained in speech signals or in text is non-trivial, and is a major avenue for current research, with great strides being made in recent years. Self-supervised speech representations such as Wav2vec (Baevski et al. 2020), XLSR (Conneau et al. 2020) and HuBERT (Hsu et al. 2021) have shown excellent performance in a wide variety of speech tasks, and

BERT (Devlin et al. 2019) and variants of it have been used to extract representations of both words and sentences (Reimers & Gurevych 2019).

In this section, a selection of speech and text representations are described. They cover a range of techniques to produce feature representations of differing qualities.

2.7.1 Text representations

Text representations take a word or sequence of words, and convert it into a vector or sequence of vectors. Three options for text representations are covered here: latent semantic analysis; Word2vec; and BERT-based sentence embeddings. The first two are word-based, meaning that vector representations of individual words are produced. The third is sentence-based, meaning that it can take account of how words work together.

2.7.1.1 Latent semantic analysis

LSA is an unsupervised technique for learning word representations, which makes use of the distributional hypothesis in linguistics, summarised by Firth (1957) as “you shall know a word by the company it keeps”. LSA exploits the assumption that similar words in a set of documents will have similar distributions. Consider documents in a collection, each of which is composed of a list of words. Let $\mathcal{W} = \{w_1, ..., w_N\}$ be the set of unique words across all the documents. Each document can be represented by an N-dimensional vector giving the count of each word that occurs in that document. Note that this is a ‘bag of words’ technique, meaning that the *order* of words is not considered. Given M documents, the document collection can be represented as a $N \times M$ occurrence matrix, $\boldsymbol{O}$, where each column represents a document, and each row a word. The occurrence matrix is typically sparse, with many words not appearing in most documents. LSA uses Singular Value Decomposition (SVD) to reduce the rank of the occurrence matrix. Any real matrix $\boldsymbol{O}$ can be decomposed into a product,

$$\boldsymbol{O} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\top} \tag{2.49}$$

where $\boldsymbol{U}$ is a $N \times N$ orthogonal matrix, $\boldsymbol{V}$ is a $M \times M$ orthogonal matrix, and $\boldsymbol{\Sigma}$ is a $M \times N$ rectangular diagonal matrix. The rank of $\boldsymbol{O}$ is $r \leq \min(M, N)$, and there will be r non-zero values along the diagonal of $\boldsymbol{\Sigma}$, which can be chosen to be ordered such that $\Sigma_{11} \geq \Sigma_{22} \geq ... \geq \Sigma_{rr}$. By selecting the largest singular values, and their corresponding singular vectors from $\boldsymbol{U}$ and $\boldsymbol{V}$, a reduced rank approximation of $\boldsymbol{O}$ can be found. If $r < M, N$, then the rank can be reduced to r without loss of information. Reducing the rank further than this can lose information, but this may come predominantly from any 'noise' present, since this is more likely to be uncorrelated. Further, the sparsity of $\boldsymbol{O}$ means that rank can in some cases be reduced substantially with only limited loss of information. Selecting the first k columns of $\boldsymbol{U}$, corresponding to the largest k singular values in $\boldsymbol{\Sigma}$, and the first k rows of $\boldsymbol{U}$ produces a k-dimensional space into which all of the documents and words are projected. Reduced-rank word vectors of dimension k can be extracted by,

$$\boldsymbol{T}_k = \boldsymbol{U}_k \boldsymbol{\Sigma}_k, \tag{2.50}$$

where $\boldsymbol{T}_k$ is a $N \times k$ matrix containing k-dimensional row vectors for each of the N unique words. Thus, SVD can be used to obtain vectors of any desired dimension to represent words. Due to the distributional hypothesis in linguistics, similar words typically have similar representations, which makes LSA features useful for determining semantic similarity. Once learned, LSA representations contain no information about word order, and include no context; each word is represented in isolation, without reference to other words in the same sentence. LSA can only produce embeddings for words seen in the training corpus.

2.7.1.2 Word2vec

Word2vec is a family of models for obtaining vector representations of words (Mikolov, Chen, Corrado & Dean 2013, Mikolov, Sutskever, Chen, Corrado & Dean 2013). The Word2vec architectures are relatively simple compared to its contemporaries, with the authors trading model power for ease of training on large quantities of data. There are two main forms of model: Continuous Bag-Of-Words (CBOW); and continuous skip-gram.

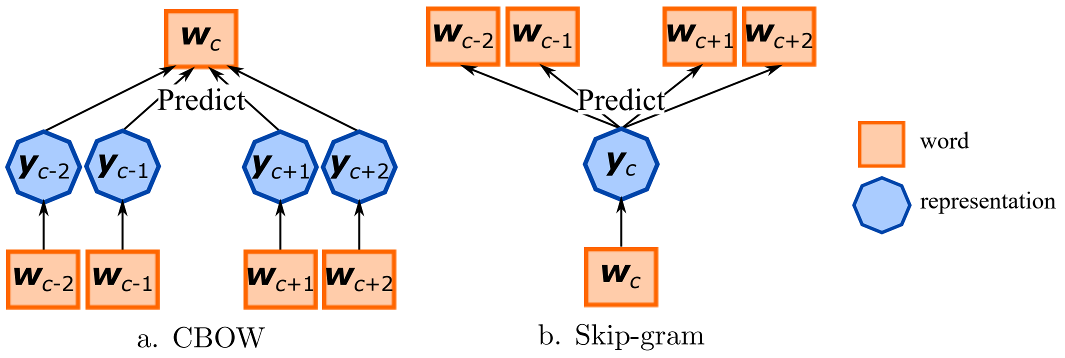


Figure 2.10: *Word2vec.*

The underlying assumption of the CBOW model is that knowing a good representation of a word should give information about its near-neighbours, as shown in Figure 2.10a.

The underlying assumption of the skip-gram model is that knowing good representations of a word's neighbours should give you information about the word, as shown in Figure 2.10b. In both cases, word representations $\boldsymbol{y}_c$ (show in blue in Figure 2.10), are found by maximising one of the following for a corpus $\mathcal{C}$:

$$\prod_{c\in\mathcal{C}} P(w_c|w_{c-M},...,w_{c-1},w_{c+1},...,w_{c+M}), \qquad \text{CBOW} \qquad (2.51)$$

$$\prod_{c\in\mathcal{C},m=\pm1,...,\pm M} P(w_{c+m}|w_c), \qquad \text{skip-gram,} \qquad (2.52)$$

where M is some integer giving the context over which to train. In both models, a vector embedding, $\boldsymbol{y}_c$, is found for each unique word in the training corpus, and is in general similar to the vectors of the words that are often found near it.

The skip-gram model is more computationally expensive, but typically outperforms the CBOW model. Finding the probabilities $P(w_{c+m}|w_c)$ is non-trivial, since the distribution is over the entire vocabulary, and a softmax over such a large dimension could be prohibitively expensive. Mikolov, Sutskever, Chen, Corrado & Dean (2013) propose two options: hierarchical softmax, in which a binary decision tree is used to reduce the dimension of the output while still approximating the full softmax; or negative sampling, in which the objective is to select the correct w_{c+m} from a list of k distractor samples.

Like LSA embeddings, Word2vec embeddings include no context or word order information, and have largely been replaced by more sophisticated, contextual, sentence-level representations, such as the BERT-based embeddings described next.

2.7.1.3 Sentence BERT

Sentence BERT (SBERT) (Reimers & Gurevych 2019) uses a pretrained BERT model (Devlin et al. 2019), or a similar model such as MPNet (Song et al. 2020), and performs further pre-training to optimise the sentence embedding. Three training objectives are used, all of which require sentence pairs as input. The first objective also requires labeling of the pairs: *contradiction*, *entailment* or *neutral*; the training consists of predicting the class of sentence pairs, and uses cross entropy loss. The second objective uses mean squared error loss between the vectors representing each sentence in the pair. The third objective uses triplet loss (Chechik et al. 2010) to compare one of the sentences in each pair to *positive* and *negative* samples: the positive sample is the other sentence in the pair; the negative sample is selected randomly from the corpus. The authors found that models tuned in this way produced better sentence representations that the original BERT models when evaluated on semantic similarity tasks.

2.7.2 Speech representations

Raw audio signals for speech tasks are typically converted into frame-based features prior to performing ASR. These are often spectrogram-based features, but more recently, much more sophisticated feature extractors have been developed using self-supervised learning. These Self-Supervised Speech Representations (SSSRs) use large quantities of unlabelled speech data to learn highly-informative representations that can be used for a wide variety of tasks.

2.7.2.1 Spectrogram

Spectrogram-based features convert speech waveforms from the amplitude-time domain to the frequency-amplitude-time domain. This is typically done frame-wise, using Short Time Fourier Transforms (STFTs) (Allen 1977). A common additional step is to convert the frequencies to the mel scale (Stevens et al. 1937). Humans can hear differences between low frequencies more easily than differences between high frequencies, and this difference in perception is not linear. The mel scale is a non-linear transform of the frequencies to make them match human perception, rather than physical quantities. It can provide better results when dealing with speech.

Spectrogram features may be supplemented by additional features, such as delta and delta-delta features (Furui 1986), which give a measure of the rate of change of the signal and the acceleration of the signal respectively. The simplest method for calculating the delta and delta-delta features is as follows:

$$\boldsymbol{\delta}_t^{(1)} = \boldsymbol{x}_{t+1} - \boldsymbol{x}_{t-1} \qquad \text{delta} \tag{2.53}$$

$$\boldsymbol{\delta}_t^{(2)} = \boldsymbol{\delta}_{t+1}^{(1)} - \boldsymbol{\delta}_{t-1}^{(1)} \qquad \text{delta-delta} \tag{2.54}$$

where $\boldsymbol{x}_t$ is the spectrogram (or alternative features) for the tth frame. Other methods for calculating the delta and delta-delta features, for example, using regression, can also be used. The delta and delta-delta features can then be concatenated with the original features to make a new feature vector,

$$\boldsymbol{y}_t = \begin{bmatrix} \boldsymbol{x}_t \\ \boldsymbol{\delta}_t^{(1)} \\ \boldsymbol{\delta}_t^{(2)} \end{bmatrix} . \tag{2.55}$$

The spectrogram-based features described here can be performed as pre-processing, and do not require training data. Despite their relative simplicity, they are still used as input to state-of-the-art models such as Whisper (Radford et al. 2023), an ASR model which is discussed in more detail later.

2.7.2.2 Wav2vec2.0

Wav2vec2.0 is a SSSR framework proposed by Baevski et al. (2020), which builds on the earlier Wav2vec framework (Schneider et al. 2019). Both Wav2vec and Wav2vec2.0 use self-supervised pre-training on unlabelled data, and take the audio waveform as input. The Wav2vec2.0 model uses a CNN, quantization, and a transformer to extract embeddings from the raw audio signal. The CNN, referred to by the authors as the *encoder* network, creates frame-based feature vectors $\boldsymbol{z}_t$ from the raw audio data. The feature vectors are quantized using product quantization (Jegou et al. 2010), producing discrete, rather than continuous, representations, which are used by the transformer, referred to by the authors as the *context* network, to combine them into contextual embeddings, $\boldsymbol{c}_t$. The loss function is contrastive loss,

$$\mathcal{L}_{CL} = -\log \frac{\exp(\tilde{\boldsymbol{c}}_t^\top \tilde{\boldsymbol{q}}_t/\kappa)}{\sum_{\boldsymbol{r}\in\mathcal{Q}_t} \exp(\tilde{\boldsymbol{c}}_t^\top \tilde{\boldsymbol{r}}/\kappa)} \tag{2.56}$$

where $\boldsymbol{q}_t$ is the quantized representation at time t and $\mathcal{Q}_t = \{\boldsymbol{q}_{i_1}, ..., \boldsymbol{q}_{i_K}, \boldsymbol{q}_t\}$ is the set of K distractor samples and $\boldsymbol{q}_t$. The distractor samples are quantized representations randomly selected from the same utterance as $\boldsymbol{q}_t$, such that $i_k \neq t$. The contextualised embedding $\boldsymbol{c}_t$ is the output of the context network, which takes the utterance that $\boldsymbol{q}_t$ belongs to as input, but with $\boldsymbol{q}_t$ and the distractor samples masked. The objective is to pick $\boldsymbol{q}_t$ out of the set of distractors, conditioned on $\boldsymbol{c}_t$. The hyperparameter κ is the softmax temperature, which controls the sharpness of the contrastive distribution. An additional loss, $\mathcal{L}_d$, which encourages diversity amongst the quantized representations, is added, to prevent all representations from from converging,

$$\mathcal{L} = \mathcal{L}_{CL} + \alpha\mathcal{L}_d. \tag{2.57}$$

Wav2vec2.0 was developed for ASR, but has been used successfully for a range of downstream tasks, including speech emotion recognition, (Chen & Rudnicky 2023, Sharma 2022, Wang et al. 2021), speaker verification (Fan et al. 2021, Tak et al. 2022, Wang et al. 2021) and language identification (Fan et al. 2021, Tjandra et al. 2022, Liu et al. 2022). Cross-Lingual Representation Learning for Speech Recognition (XLSR) (Conneau et al. 2020) extends Wav2vec2.0 to produce multilingual embeddings by training on multilingual data.

2.7.2.3 HuBERT

Hidden Unit BERT (HuBERT) (Hsu et al. 2021) is a SSSR that uses a masked prediction task similar to BERT, but using pseudo-phonetic labels. The pseudo-phonetic labels are derived from a non-supervised clustering model such as k-means, using a frame-based spectrogram or Mel Frequency Cepstral Coefficient (MFCC) representation of the data (Mermelstein 1976). Although the derivation of the labels uses frame-based features, the input to HuBERT is the waveform. HuBERT's architecture is similar to Wav2vec2.0, with a CNN-based encoder and a BERT-based context network.

HuBERT is trained with masking. Given an input sequence $\boldsymbol{Q}_{1:T}$ with pseudo-phonetic labels $\boldsymbol{z}_{1:T}$, a random subset of time steps $\mathcal{M} \subset \{1, ..., T\}$ is chosen to be masked. Each vector $\boldsymbol{q}_m$, for $m \in \mathcal{M}$, is replaced with a learned mask embedding $\hat{\boldsymbol{q}}$. This masked sequence will be referred to as $\hat{\boldsymbol{Q}}_{1:T}$. The objective is to predict $\boldsymbol{z}_{1:T}$. The loss is weighted depending on whether the input being predicted is masked or not,

$$\mathcal{L}_m(\boldsymbol{\Theta}; \boldsymbol{Q}_{1:T}, \mathcal{M}, \boldsymbol{z}_{1:T}) = \sum_{t \in \mathcal{M}} \log p_{\Theta}(z_t | \hat{\boldsymbol{Q}}_{1:T}, t) \tag{2.58}$$

$$\mathcal{L}_u(\boldsymbol{\Theta}; \boldsymbol{Q}_{1:T}, \mathcal{M}, \boldsymbol{z}_{1:T}) = \sum_{t \notin \mathcal{M}} \log p_{\Theta}(z_t | \hat{\boldsymbol{Q}}_{1:T}, t) \tag{2.59}$$

$$\mathcal{L}(\boldsymbol{\Theta}; \boldsymbol{Q}_{1:T}, \mathcal{M}, \boldsymbol{z}_{1:T}) = \alpha \mathcal{L}_m(\boldsymbol{\Theta}; \boldsymbol{Q}_{1:T}, \mathcal{M}, \boldsymbol{z}_{1:T}) + (1 - \alpha) \mathcal{L}_u(\boldsymbol{\Theta}; \boldsymbol{Q}_{1:T}, \mathcal{M}, \boldsymbol{z}_{1:T}). \tag{2.60}$$

The pseudo-phonetic labelling initialised by k-means clustering is updated periodically during training. In practice, performing clustering with different values of k to predict multiple sets of labels, and treating each label set as parallel training tasks, is found to give better results.

HuBERT has shown promise for downstream tasks such as ASR (Hsu et al. 2021, Chang et al. 2021), speaker verification (Wang et al. 2021, Chen, Chen, Wu, Qian, Wang, Liu, Qian & Zeng 2022) and speech emotion recognition (Wang et al. 2021, Morais et al. 2022).

2.7.2.4 Whisper

Whisper (Radford et al. 2023) is a transformer-based encoder-decoder ASR model. It takes as input 30 s sequences of audio represented by log Mel spectrograms. Audio that is longer or shorter than 30 s is padded or truncated. Whisper is trained on extremely large quantities of data obtained from the web. The resulting labelled data is noisy, but Radford et al. (2023) demonstrate that large quantities of lower quality data can give better performance than smaller quantities of high quality data, especially if sophisticated automatic filtering mechanisms are used to remove very poor quality examples. Whisper is pre-trained on $680,000$ hours of multi-lingual data to do multiple tasks, including: ASR, speech translation, language identification and voice activity detection.

Whisper gives state-of-the-art results for ASR (Radford et al. 2023). Using intermediate representations from trained Whisper models, taken either from the encoder or decoder layers, has been shown to be effective for tasks such as speaker diarisation (Papala et al. 2023), sentiment analysis (Papala et al. 2023) and speaker verification (Zhang et al. 2024).

Whisper's training data is not clearly specified. Care must therefore be taken when using Whisper representations to ensure that the task's test or evaluation data has not been part of Whisper's training data.

2.8 Evaluation metrics

In order to compare the performance of different models, an evaluation metric is required. The metric should, ideally, give an understanding of how well the relevant task was performed. Different metrics are used, depending on the task, and some example evaluation metrics that are relevant for speech and language tasks are described here.

2.8.1 Accuracy, precision, recall and F_1 score

The results of a binary classification task can be summarised in a *confusion matrix*, as shown in 2.1. *False positives* are also referred to a Type I errors, and *false negatives* are referred to as Type II errors.

Table 2.1: *Confusion matrix.*

		Actual condition	
		True	False
Predicted	True	TP (true positive)	FP (false positive)
Condition	False	FN (false negative)	TN (true negative)

Accuracy is given by,

$$\text{accuracy} = \frac{\text{correctly predicted items}}{\text{total number of items}} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{2.61}$$

Accuracy can be misleading for imbalanced classes. For example, if 95% of the data is labelled *negative*, then 95% accuracy can be achieved by predicting that all data points are *negative*, which is clearly unhelpful. In this case, *precision* and *recall* may be more useful, along with the F_1 score,

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2.62}$$

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.63}$$

$$F_1 = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} = \frac{2 \times \text{TP}}{2 \times \text{TP} + \text{FP} + \text{FN}}. \tag{2.64}$$

2.8.1.1 Area under receiver operating characteristic curve

Given a binary classifier that returns a result $r \in (0, 1)$, where higher values mean a greater probability that the label is *positive*, where should the threshold be drawn? The choice of threshold affects the confusion matrix, and therefore also the accuracy, precision, recall and F_1 score. An alternative, threshold-independent measure is discussed here.

The Receiver Operating Characteristic (ROC) curve plots the *true positive rate* against the *false positive rate* for different thresholds.

$$\text{true positive rate} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.65}$$

$$\text{false positive rate} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{2.66}$$

Examples of ROC curves are shown in Figure 2.11. When the threshold is 0, all examples will be classified negative, so both true positive rate and false positive rate will be 0. When the threshold is 1, all inputs will be classified positive, and the true positive rate and false positive rate will both be 1. At intermediate thresholds, the higher the true positive rate is compared to the false positive rate, the better. If they are always equal, then the classifier is equivalent to random chance (for a balanced dataset), shown by the dashed black line in Figure 2.11. The blue curve is for a classifier that performs better than chance, and the red curve is for a classifier that behaves better still.

The better the true positive rate is compared to the false positive rate, the greater the area under the ROC curve will be. The Area Under ROC Curve (AUC) is therefore a useful measure of how effective a classifier is, without requiring a defined threshold.

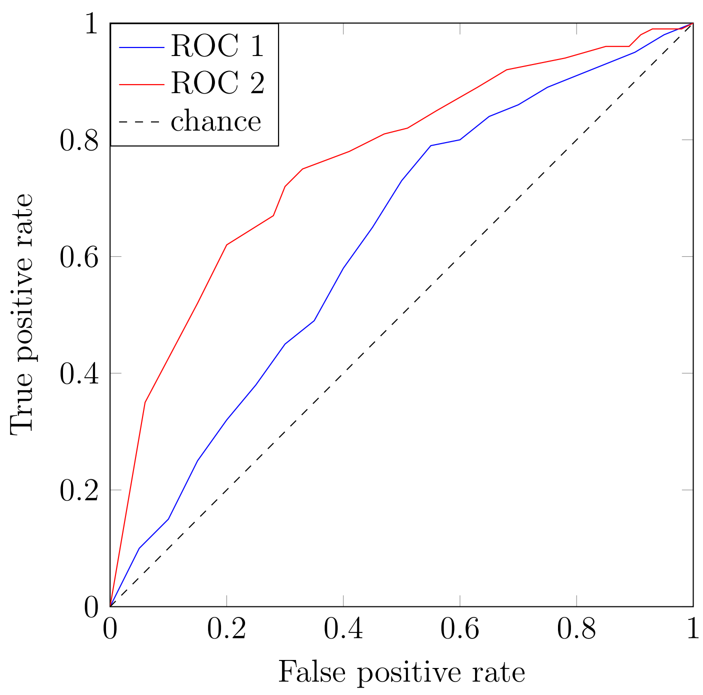


Figure 2.11: *Example ROC curves.*

2.8.1.2 Root mean squared error

Given a prediction $\hat{\boldsymbol{x}}$ and true value $\boldsymbol{x}$, the MSE is given by,

$$\mathrm{MSE}(\hat{\boldsymbol{x}}, \boldsymbol{x}) = \frac{1}{J}(\boldsymbol{x} - \hat{\boldsymbol{x}})^\top(\boldsymbol{x} - \hat{\boldsymbol{x}}), \tag{2.67}$$

where J is the dimension of $\boldsymbol{x}$ and $\hat{\boldsymbol{x}}$. The Root Mean Square Error (RMSE) is $\sqrt{\mathrm{MSE}}$, which is equivalent to the Euclidean distance or L2 distance between $\boldsymbol{x}$ and $\hat{\boldsymbol{x}}$. RMSE is generally used for evaluating the performance of a regression.

2.8.1.3 Word error rate and phone error rate

Using the ground truth sentence *the cat and the fiddle*, three types of errors can be defined:

1. *Substitutions*, S, in which one word is incorrectly swapped with another, e.g. *the bat and the fiddle*
2. *Insertions*, I, in which a word is added that shouldn't be there, e.g. *the black cat and the fiddle*

3. *Deletions*, D, in which a word is removed that should be present, e.g. *the cat and fiddle*

The Word Error Rate (WER) is given by,

$$\text{WER} = \frac{S + D + I}{N} \tag{2.68}$$

where N is the total number of words in the ground truth sequence.

Closely related to the WER is the Phone Error Rate (PER), which is the same process by applies to phones rather than to words.

2.9 Concluding remarks

This chapter has provided a description of Minerva 2, the human cognition model that is the focus of this thesis, as well as covering previous work that has made use of it. It has covered other exemplar, prototype and hybrid models that have been used for speech and language tasks. Since representation of exemplars is a key research question, several options for feature representation have been described.

Minerva 2 bears a strong resemblance to the attention mechanism found in the transformer architecture. What further similarities are there between Minerva 2 and other architectures? This forms our focus in the next chapter, where the mathematics behind Minerva 2 are explored in more detail, drawing connections between Minerva 2 and other models.

Chapter 3

Theoretical framework

3.1 Introduction

In the previous chapter, clear similarities were noted between Minerva 2 and attention. This leads to the first research question:

RQ1 What are the similarities and differences between the exemplar model Minerva 2 and ANNs?

In addition to its relationship with attention, it will be shown in §3.2 that Minerva 2 with a fixed exemplar set is a constrained form of a feed-forward neural network. It will also be shown in §3.3 that Minerva 2's echo-of-echoes process, described in §2.2.1, is a fixed point problem and a form of deep equilibrium model.

In §3.5, several parameterised models based on Minerva 2 are proposed. These are hybrid prototype-exemplar models, which can be trained by backpropagation, and relate to Research Questions 2 and 3:

RQ2 What exemplars should be stored in memory, and how should they be represented?

RQ3 How can an exemplar model be combined with parameters to form a hybrid exemplar-prototype model? What benefits, if any, does this bring?

Several of the models incorporate a learned linear transformation of the input features, relating to Research Question 2. All of the them incorporate learned parameters, potentially increasing the model's power and flexibility. In this chapter, a theoretical framework for these models is provided, setting the stage for experimental work in later chapters.

3.2 Minerva 2 as an artificial neural network

Memory-based models such as Minerva 2 are often contrasted with ANNs, but it can be shown that, when using a fixed exemplar set, Minerva 2 is itself a special case of FFNN. The ith layer in a FFNN can be represented as,

$$\boldsymbol{h}^{(i)} = \boldsymbol{\sigma}^{(i)}(\boldsymbol{W}^{(i)}\boldsymbol{h}^{(i-1)} + \boldsymbol{b}^{(i)}) \tag{3.1}$$

where the $\boldsymbol{W}^{(i)}$ are the layer weights, the $\boldsymbol{b}^{(i)}$ are the layer biases, and the $\boldsymbol{\sigma}^{(i)}$ are the activation functions. Reproducing the Minerva 2 equations from §2.2 (Equations 2.1 to 2.4),

$$\begin{aligned}
\boldsymbol{s} &= \frac{1}{F}\boldsymbol{K}^\top\boldsymbol{q} && (3.2)\\
\boldsymbol{a} &= \boldsymbol{s}^{\circ\beta} && (3.3)\\
\boldsymbol{c} &= \boldsymbol{V}\boldsymbol{a} && (3.4)\\
\langle\boldsymbol{c}\rangle_\infty &= \frac{\boldsymbol{c}}{||\boldsymbol{c}||_\infty}, && (3.5)
\end{aligned}$$

it can be seen that, if the exemplar set is fixed, Minerva 2 can be expressed as a FFNN:

$$\begin{aligned}
\boldsymbol{W}^{(1)} &= \frac{1}{F}\boldsymbol{K}^\top & \boldsymbol{b}^{(1)} &= \boldsymbol{0} & \boldsymbol{\sigma}^{(1)}(\boldsymbol{x}) &= \boldsymbol{x}^{\circ\beta} && (3.6)\\
\boldsymbol{W}^{(2)} &= \boldsymbol{V} & \boldsymbol{b}^{(2)} &= \boldsymbol{0} & \boldsymbol{\sigma}^{(2)}(\boldsymbol{x}) &= \langle\boldsymbol{x}\rangle_\infty. && (3.7)
\end{aligned}$$

Thus, Minerva 2 is an FFNN, where the exemplars form the parameters. Minerva 2 is initialised from data, rather than trained on it. This has implications for Minerva 2's performance in comparison with FFNNs. Exemplar and prototype models are often contrasted, but in this case, they are structurally equivalent; the difference is in how parameters are selected. Despite being a FFNN, Minerva 2 does not benefit from the Universal Approximation Theory that states that FFNNs are function approximators (see §2.5.1) because its activation function on the first layer is polynomial.

3.3 Echo-of-echoes

The echo-of-echoes process can be re-framed as a FFNN in the same way. Reproducing Equations 2.6 to 2.9,

$$\boldsymbol{s}^{(i)} = \frac{1}{J}\boldsymbol{V}^\top \langle \boldsymbol{c}\rangle_\infty^{(i-1)} \tag{3.8}$$

$$\boldsymbol{a}^{(i)} = \boldsymbol{s}^{(i)^{\circ\beta}} \tag{3.9}$$

$$\boldsymbol{c}^{(i)} = \boldsymbol{V}\boldsymbol{a}^{(i)} \tag{3.10}$$

$$\langle \boldsymbol{c}\rangle_\infty^{(i)} = \frac{\boldsymbol{c}^{(i)}}{||\boldsymbol{c}^{(i)}||_\infty} \tag{3.11}$$

it can be seen that,

$$\boldsymbol{a}^{(i)} = \boldsymbol{\sigma}^{(3)}\left(\boldsymbol{W}^{(3)}\langle \boldsymbol{c}\rangle_\infty^{(i-1)} + \boldsymbol{b}^{(3)}\right) \tag{3.12}$$

$$\langle \boldsymbol{c}\rangle_\infty^{(i)} = \boldsymbol{\sigma}^{(4)}\left(\boldsymbol{W}^{(4)}\boldsymbol{a}^{(i)} + \boldsymbol{b}^{(4)}\right) \tag{3.13}$$

where

$$\boldsymbol{W}^{(3)} = \frac{1}{J}\boldsymbol{V}^\top \qquad \boldsymbol{b}^{(3)} = \boldsymbol{0} \qquad \boldsymbol{\sigma}^{(3)}(\boldsymbol{x}) = \boldsymbol{x}^{\circ\beta} \tag{3.14}$$

$$\boldsymbol{W}^{(4)} = \boldsymbol{V} \qquad \boldsymbol{b}^{(4)} = \boldsymbol{0} \qquad \boldsymbol{\sigma}^{(4)}(\boldsymbol{x}) = \langle \boldsymbol{x}\rangle_\infty. \tag{3.15}$$

Each iteration of the echo-of-echoes process is a 2-layer FFNN, with parameters shared with previous iterations. As an infinite-depth FFNN in which the layers share parameters, the echo-of-echoes process is also a form of DEM (Bai et al. (2019), and see §2.5.2).

Further examination of Minerva's echo-of-echoes process shows that it is not in general effective. In order to solve the problem of ambiguous recall, the echo-of-echoes process must converge to something that can be linked to a single class with little or no ambiguity. The ith echo-of-echoes, $\boldsymbol{c}^{(i)}$, can be written in terms of $\boldsymbol{c}^{(i-1)}$ by combining Equations 3.8 to 3.11,

$$\boldsymbol{c}^{(i)} = \frac{1}{J^\beta}\boldsymbol{V}\left(\boldsymbol{V}^\top \langle \boldsymbol{c}^{(i-1)}\rangle_\infty\right)^{\circ\beta}. \tag{3.16}$$

In order to be useful, the $\boldsymbol{c}^{(i)}$ must converge, that is,

$$\lim_{i\to\infty} \boldsymbol{c}^{(i)} = \boldsymbol{c}^*. \tag{3.17}$$

If the sequence converges, then $\boldsymbol{c}^*$ must satisfy,

$$\boldsymbol{c}^* = \frac{1}{J^\beta}\boldsymbol{V}\left(\boldsymbol{V}^\top\langle\boldsymbol{c}^*\rangle_\infty\right)^{\circ\beta}. \tag{3.18}$$

Noting that $\langle\boldsymbol{c}^*\rangle_\infty$, the $L\infty$ normalised version of $\boldsymbol{c}^*$, is simply a scaled version of $\boldsymbol{c}^*$, this can be rewritten,

$$\boldsymbol{c}^* = \alpha\boldsymbol{V}\left(\boldsymbol{V}^\top\boldsymbol{c}^*\right)^{\circ\beta}, \tag{3.19}$$

where α can be any positive real number. This type of equation, in which the objective is to find a value for a function where the input is equal to the output, is a fixed point problem. Fixed point problems have been widely studied, and there are efficient iterative algorithms to solve them (Hoffman & Frankel 2018).

The echo-of-echoes process differs from a typical DEM in two ways. Firstly, DEMs usually have a single layer repeated, rather than two. More importantly, for the echo-of-echoes process, fixed points are expected to be the exemplar class representations: $\boldsymbol{c}^* \in \{\boldsymbol{v}_1, ..., \boldsymbol{v}_N\}$. This is not assumed for DEMs, and warrants further investigation.

First considering the case where there are W unique, linearly independent class representations, each of which is represented in the exemplar set once, and denoting this restricted set of exemplars $\boldsymbol{U} = [\boldsymbol{u}_1, ..., \boldsymbol{u}_W]$. According to the echo-of-echoes process, each of these class representations must be a fixed point,

$$\boldsymbol{u}_1 = \alpha_1\boldsymbol{U}\left(\boldsymbol{U}^\top\boldsymbol{u}_1\right)^{\circ\beta} \tag{3.20}$$

$$\vdots$$

$$\boldsymbol{u}_W = \alpha_W\boldsymbol{U}\left(\boldsymbol{U}^\top\boldsymbol{u}_W\right)^{\circ\beta}. \tag{3.21}$$

These can be written in matrix form as,

$$\boldsymbol{U} = \boldsymbol{U}\left(\boldsymbol{U}^\top\boldsymbol{U}\right)^{\circ\beta}\boldsymbol{\Lambda} \tag{3.22}$$

where

$$\boldsymbol{\Lambda} = \begin{bmatrix}\alpha_1 & \dots & 0\\ \vdots & \ddots & \vdots\\ 0 & \dots & \alpha_W\end{bmatrix}. \tag{3.23}$$

Since $\boldsymbol{U}$ has been defined to have full rank, it must have an inverse, so,

$$\boldsymbol{U}^{-1}\boldsymbol{U} = \boldsymbol{U}^{-1}\boldsymbol{U}\left(\boldsymbol{U}^\top\boldsymbol{U}\right)^{\circ\beta}\boldsymbol{\Lambda} \tag{3.24}$$

$$\boldsymbol{I}_W = \left(\boldsymbol{U}^\top\boldsymbol{U}\right)^{\circ\beta}\boldsymbol{\Lambda} \tag{3.25}$$

where $\boldsymbol{I}_W$ is the W-dimensional identity matrix, and therefore,

$$\boldsymbol{\Lambda}^{-1} = \left(\boldsymbol{U}^\top \boldsymbol{U}\right)^{\circ\beta}. \tag{3.26}$$

Since $\boldsymbol{\Lambda}$ is diagonal, and the inverse of a diagonal matrix is also diagonal, the matrix $\left(\boldsymbol{U}^\top \boldsymbol{U}\right)^{\circ\beta}$ must be a diagonal matrix, and since $\boldsymbol{\beta}$ acts element-wise, so too is $\boldsymbol{U}^\top \boldsymbol{U}$. This is only the case if the column vectors $\boldsymbol{u}_1, ..., \boldsymbol{u}_W$ are orthogonal. Thus, if the class representations are linearly independent, they must also be orthogonal.

In the case that Hintzman (1986) explored, which used 3 classes and 10-dimensional class representations, the probability of high correlation between classes is relatively low, which may explain why the echo-of-echoes process was found to work. Consider, however, an example in which the classes are highly correlated with each other,

$$\begin{aligned}
\boldsymbol{v}_1 &= \begin{bmatrix}1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1\end{bmatrix}^\top \\
\boldsymbol{v}_2 &= \begin{bmatrix}1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & -1\end{bmatrix}^\top \\
\boldsymbol{v}_3 &= \begin{bmatrix}-1 & -1 & -1 & 1 & 1 & 1 & 1 & 1 & 1 & 1\end{bmatrix}^\top.
\end{aligned}$$

Feeding $\boldsymbol{v}_1$, $\boldsymbol{v}_2$ and $\boldsymbol{v}_3$ into Equation 3.16 and iterating until convergence gives the following fixed points:

$$\begin{aligned}
\langle \boldsymbol{c}_1^* \rangle_\infty &= \begin{bmatrix}0.98 & 0.98 & 0.98 & 1 & 1 & 1 & 1 & 1 & 1 & 0.01\end{bmatrix}^\top \\
\langle \boldsymbol{c}_2^* \rangle_\infty &= \begin{bmatrix}0.98 & 0.98 & 0.98 & 1 & 1 & 1 & 1 & 1 & 1 & 0.01\end{bmatrix}^\top \\
\langle \boldsymbol{c}_3^* \rangle_\infty &= \begin{bmatrix}-0.88 & -0.88 & -0.88 & 1 & 1 & 1 & 1 & 1 & 1 & 0.99\end{bmatrix}^\top
\end{aligned}$$

None of the chosen exemplar class representations converges to itself; the exemplar classes are not fixed points. Further, both $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ converge to the same fixed point, which is a vector that falls between the two classes. The results of the echo-of-echoes process are, in this case, completely ambiguous with regard to classes 1 and 2.

The results above were derived by assuming linear independence between the exemplars, but what if this is not the case? Equation 3.22 is then replace with,

$$\boldsymbol{V} = \boldsymbol{V}(\boldsymbol{V}^\top \boldsymbol{V})^{\circ\beta} \boldsymbol{\Lambda}_V. \tag{3.27}$$

In this case, $\boldsymbol{V}$ is a $W \times N$ matrix, where N is the number of exemplars. Since the exemplars have some linear dependence, it has rank $r < N$, and as such has no left inverse. This means that there may be multiple solutions. Equation 3.27 is not true for every $\boldsymbol{V}$, however; the exemplar labels would need to be chosen to be suitable.

Rather than constraining the exemplar labels to be fixed point solutions, replacing Equation 3.27 with,

$$\boldsymbol{U} = \boldsymbol{V}(\boldsymbol{V}^\top \boldsymbol{U})^{\circ\beta}\boldsymbol{\Lambda}, \tag{3.28}$$

allows the exemplar labels and fixed points to be separate. In this case, the exemplar labels $\boldsymbol{V}$ and the prediction labels $\boldsymbol{U}$ no longer share the same label space. This is how conventional DEMs work: since their parameters are learned through backpropagation, there is no expectation that they will resemble the fixed point class labels.

The similarity with DEMs means that the echo-of-echoes process could in principal be trained, leveraging known fixed-point algorithms, and making use of the efficient backpropagation of DEMs.

3.4 Exemplar selection

3.4.1 Exemplar set size

Minerva 2 works on the assumption that similar feature vectors are likely to share the same label, and identical feature vectors are certain to share the same label. More mathematically, these assumptions can be expressed as:

$$\lim_{\tilde{\boldsymbol{q}} \to \tilde{\boldsymbol{k}}_n} P(\boldsymbol{w} = \boldsymbol{v}_n) = 1. \tag{3.29}$$

If these assumptions are true, then as the number of exemplars increases, so does the probability of there being an exemplar that is a) very similar to $\boldsymbol{q}$; and b) shares its class. A large exemplar set also includes more exemplars that are *slightly* similar to $\boldsymbol{q}$, but do not share its class. A large exemplar set may therefore require a higher value for the activation power β, which leaves similarities close to 1 (and -1) mostly unchanged, but reduces the magnitude of all other similarities. Provided that the assumptions hold, tuning β while increasing the exemplar set size will improve performance, although with diminishing returns.

3.4.2 Exemplar selection

If Minerva 2's exemplar set is fixed, then the exemplar features and labels can be considered parameters of the model. Representing Minerva 2 as a function $\boldsymbol{f}$,

$$\boldsymbol{c} = \boldsymbol{f}(\boldsymbol{q}; \boldsymbol{K}, \boldsymbol{V}). \tag{3.30}$$

Options exist for how the exemplar set should be selected: curated or random, or stratified by class or other attribute.

Rather than a fixed exemplar set, the exemplars can in principal be changed, either during training, during inference, or both, in which case the output is conditioned on the exemplars,

$$\boldsymbol{c} = \boldsymbol{f}(\boldsymbol{q}|\boldsymbol{K}, \boldsymbol{V}). \tag{3.31}$$

This gives the option of selecting exemplars in some way that is meaningful: choosing recent examples to give context; or conditioning on an exemplar set that matches the input in some way. For example, exemplars could be chosen from the current speaker, for speech-based task. Exemplars used in this way might allow for adaptation at inference.

3.5 Adaptation of Minerva 2 to speech and language tasks

Minerva 2 was designed to replicate human experiments as simply as possible, and as such has serious limitations for real tasks:

1. The feature and class elements in the input and exemplars are restricted to the values $\{-1, 1\}$.
2. Ambiguous recall: the class elements in the returned echo will usually not precisely match a class.
3. With no learned parameters, the model is dependent on the feature and class vectors being truly representative.
4. Many speech and language tasks are sequence tasks, but Minerva 2 is not a sequence model.

There follow descriptions of proposed adaptations of Minerva 2 intended to address some of these limitations. Most the adapted models use a hybrid prototype-exemplar approach, incorporating learned parameters into the Minerva model. A summary of the proposed models is given in Table 3.1.

Table 3.1: *Minerva model descriptions.*

Model	Input type	Learned feature transform	Learned exemplar labels	Inference type	Sequence based
Minerva 2	± 1	no	no	one-hot	no
Minerva-R	$\mathbb{R}$	no	no	one-hot	no
Minerva-RP	$\mathbb{R}$	yes	no	one-hot	no
Minerva-RPC	$\mathbb{R}$	yes	no	dot-product	no
Minerva-RPE	$\mathbb{R}$	yes	yes	one-hot	no
Minerva-RPCE	$\mathbb{R}$	yes	yes	dot-product	no
Minerva-RPES	$\mathbb{R}$	yes	yes	one-hot	yes

3.5.1 Minerva-R: Minerva with real input

Limitation 1 can be addressed by replacing Equation 2.1 with the cosine similarity,

$$\boldsymbol{s} = \tilde{\boldsymbol{K}}^\top \tilde{\boldsymbol{q}} \tag{3.32}$$

where $\tilde{\boldsymbol{q}} = \dfrac{\boldsymbol{q}}{||\boldsymbol{q}||_2}$, $\tilde{\boldsymbol{k}}_n = \dfrac{\boldsymbol{k}_n}{||\boldsymbol{k}_n||_2}$ and $\tilde{\boldsymbol{K}} = \begin{bmatrix}\tilde{\boldsymbol{k}}_1 & \dots & \tilde{\boldsymbol{k}}_N\end{bmatrix}$.

This version of the similarity has previously been used by Nick Reid & Jamieson (2023). Alternative options exist: Maier & Moore (2005) used a distance-based measure instead, but, in the case where the elements of $\boldsymbol{q}$ and $\boldsymbol{K}$ are ± 1, the cosine similarity gives an identical result to Equation 2.1, making it a more natural successor.

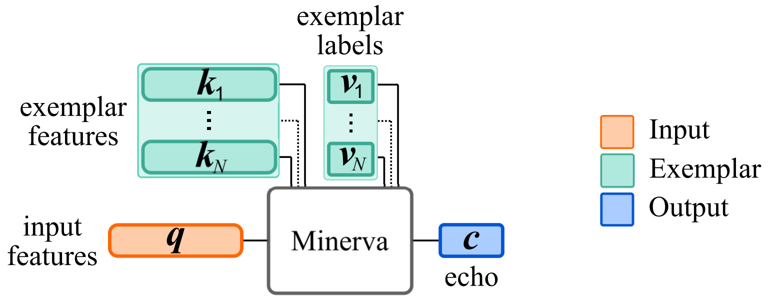


Figure 3.1: *Minerva-R.*

To address Limitation 2, one-hot representation is used for the exemplar classes. This is a common choice in machine learning. Under these conditions, the predicted class label is given by,

$$\hat{w} = \underset{w=1,\ldots,W}{argmax} \left(c_w \right) . \tag{3.33}$$

This version of Minerva has no learned parameters, and a single tuned parameter β. It is shown in Figure 3.1 and will be referred to as Minerva-R, since it makes use of real input.

For Minerva 2, the $L\infty$ normalisation applied in Equation 3.5 ensures that the highest-magnitude element of the echo has a value ± 1, so that it matches the scaling of the inputs, which are restricted to ± 1. In Minerva-R, the inputs and exemplars are no longer restricted to ± 1, so this is no longer appropriate. However, without any form of normalisation, Minerva-R has no innate scaling at all. Large exemplar sets could lead to extremely high magnitude echoes (see Equation 3.4). For classification tasks, this has no effect, since the *argmax* in Equation 3.33 will still function. For regression tasks, however, this lack of scaling is likely to be a problem, and some form of scaling or calibration will be required on the output echo.

3.5.2 Minerva-RP: Learning feature transformations

To partially address Limitation 3, which notes that Minerva 2 is entirely dependent on good feature representation, learned feature transforms can be added. These allow the model to emphasise relevant information in the features, and discard irrelevant information, making the similarity measure between the input and the exemplars more meaningful. This model will be referred to as Minerva-RP, since it uses real input and has learned parameters. The model is shown in Figure 3.2.

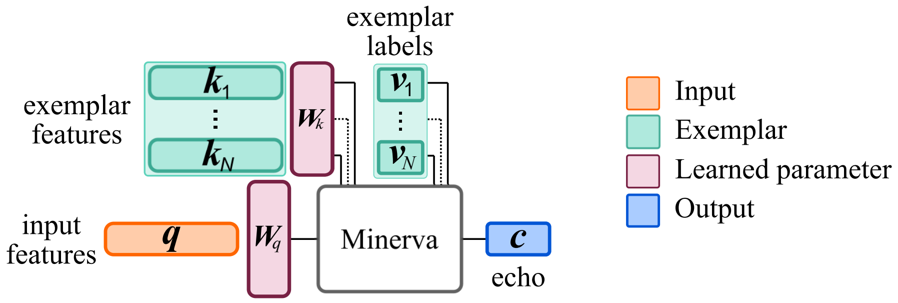


Figure 3.2: *Minerva-RP.*

Simple linear transforms (shown in red in Figure 3.2) are applied to the input and exemplar features,

$$\boldsymbol{q}_w = \boldsymbol{W}_q \boldsymbol{q} \tag{3.34}$$

$$\boldsymbol{K}_w = \boldsymbol{W}_k \boldsymbol{K} \tag{3.35}$$

$$\boldsymbol{s} = \tilde{\boldsymbol{K}}_w^\top \tilde{\boldsymbol{q}}_w. \tag{3.36}$$

The transforms $\boldsymbol{W}_q$ and $\boldsymbol{W}_k$ can either be identical or separate. There are potential advantages to each: if identical, the Minerva similarity mechanism will still have meaning, even prior to training, since correlated input and exemplar features $\boldsymbol{q}$ and $\boldsymbol{k}_n$ will lead to correlated transformed input and exemplar features. In the case of separate transforms, the Minerva attention mechanism may learn to attend to different aspects of the model, for example, tuning into or out of common confusions or adjacent phones. This model can be used either for classification or regression.

3.5.2.1 Regression

For regression, as with Minerva-R, scaling or calibration is likely to be necessary, which will depend on the nature of the task. The calibration can be learned with the other parameters. Given the output echo, $\boldsymbol{c}$, the scaled output is given by,

$$\boldsymbol{y} = a\boldsymbol{c} + b\mathbf{1}, \tag{3.37}$$

where a and b are learned parameters and $\mathbf{1}$ is a vector of 1s of the same dimension as $\boldsymbol{c}$.

MSE loss is used for training. Given training data $\mathcal{T} = \{\boldsymbol{q}_i, \boldsymbol{u}_i\}_{i=1}^{I}$, where $\boldsymbol{q}_i$ is the feature representation for the ith input, and $\boldsymbol{u}_i$ is its corresponding label,

$$\mathcal{L}_{MSE}(\boldsymbol{\Theta}_L|\mathcal{T}, \boldsymbol{\Theta}_U) = \frac{1}{I}\sum_{i=1}^{I}(\boldsymbol{y}_i - \boldsymbol{u}_i)(\boldsymbol{y}_i - \boldsymbol{u}_i)^\top, \tag{3.38}$$

where $\boldsymbol{y}$ is the scaled output, $\boldsymbol{\Theta}_L = \{\boldsymbol{W}_q, \boldsymbol{W}_k\}$ are the model's learned parameters, and $\boldsymbol{\Theta}_U = \{\boldsymbol{K}, \boldsymbol{V}\}$ are the model's exemplars, which can be thought of as pre-determined, or unlearned, parameters.

3.5.2.2 Classification

For classification, the lack of scaling on the echo has no effect at inference (the *argmax* in Equation 3.33 still functions), but may have implications for training. Using a softmax (Equation 2.17) as the final activation yields a distribution over the classes, but the distribution may be extremely sharp for large exemplar sets, due to the high magnitude of the resulting echo. For this reason, the model may benefit from some form of normalisation or scaling on the echo prior to the softmax, although the nature of this scaling is likely to depend on factors such as task, exemplar set size and activation power. Cross-entropy loss can be used for training. Given training data $\mathcal{T} = \{\boldsymbol{q}_i, u_i\}_{i=1}^{I}$, where $\boldsymbol{q}_i$ is an input feature vector and u_i its corresponding target class label, let $\boldsymbol{y}_i$ be the normalised or scaled model output for the ith input. The cross entropy loss is given by,

$$\mathcal{L}_{CE}(\boldsymbol{\Theta}_L|\mathcal{T}, \boldsymbol{\Theta}_U) = -\frac{1}{I}\sum_{i=1}^{I} \log y_{u_i,i} \tag{3.39}$$

3.5.2.3 Equivalence to artificial neural networks

Like Minerva-R, Minerva-RP is equivalent to a FFNN. Equations 3.6 and 3.7, are replaced with,

$$\boldsymbol{W}^{(1)} = \tilde{\boldsymbol{K}}_w^\top \boldsymbol{W}_q \qquad \boldsymbol{b}^{(1)} = \boldsymbol{0} \qquad \boldsymbol{\sigma}^{(1)}(\boldsymbol{x}) = \boldsymbol{x}^{\circ\beta}. \tag{3.40}$$

$$\boldsymbol{W}^{(2)} = \boldsymbol{V} \qquad \boldsymbol{b}^{(2)} = \boldsymbol{0}, \tag{3.41}$$

and the final activation $\boldsymbol{\sigma}^{(2)}$ depends on the nature of the task being performed, as discussed above, such as a softmax for classification.

Once trained, the multiplication to produce the first layer's parameters, $\tilde{\boldsymbol{K}}_w^\top \boldsymbol{W}_q$, need only be calculated once. The computational complexity of Minerva-RP at inference depends on the dimension of the feature transformations; if the dimension is reduced, it will have lower computational complexity than a Minerva-R model using the same exemplars.

3.5.3 Minerva-RPE: Learning exemplar class representations

Minerva-RP adds a feature transformation, and now a model a model in which exemplar class representations are also learned is described. This model is referred to as Minerva-RPE, since it uses real input, a parameterised transform for the features, and also learned exemplar labels. It is shown in Figure 3.3. In the case of regression, learning the exemplar labels can reduce the 'noise' associated with each exemplar. In the case of classification, it allows an exemplar to fall on a spectrum between classes. In both cases, it also allows for correction of mislabelled data in the exemplars, and increases modelling power. The model equations remain unchanged from Minerva-RP, but the learned parameters in the loss functions (Equations 3.38 and 3.39) are

$\boldsymbol{\Theta}_L = \{\boldsymbol{W}_q, \boldsymbol{W}_k, \boldsymbol{V}\}$ and the unlearned parameters are $\boldsymbol{\Theta}_U = \{\boldsymbol{K}\}$. Since it learns representations for specific exemplars, this version of Minerva can only be used with a fixed exemplar set.

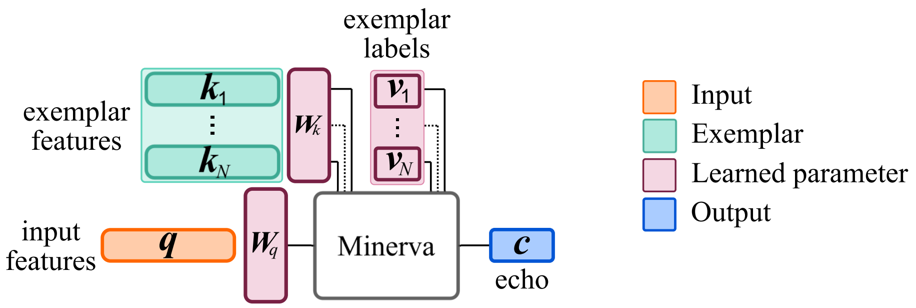


Figure 3.3: *Minerva-RPE.*

3.5.4 Minerva-RPC: Learning 'true' class representations

When learning exemplar class labels, as described in the previous section, there is still an assumption when performing classification that the 'true' class representations are one-hot. This is a common assumption in machine learning, but the enforced orthogonality of one-hot representations may not always be appropriate. Consider emotion classification, for example: is it reasonable to assume that positive and negative emotions are orthogonal? Or is it more reasonable to assume that there is a negative correlation between them?

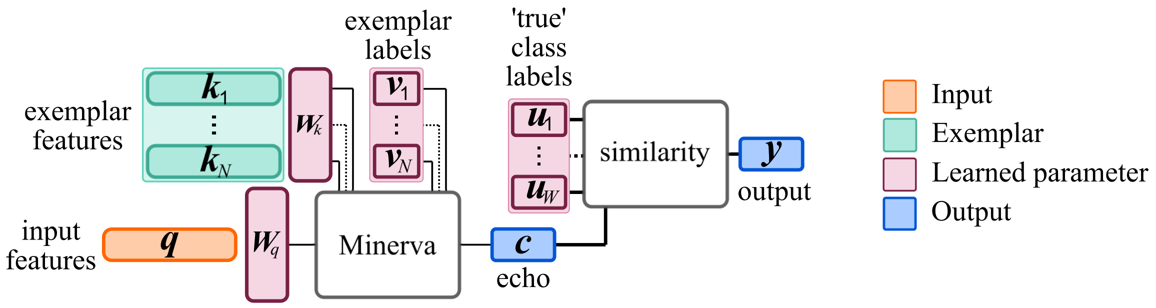


Figure 3.4: *Minerva-RPCE.*

It may be possible to learn more meaningful class representations. Let $\boldsymbol{U} = \begin{bmatrix}\boldsymbol{u}_1 & \dots & \boldsymbol{u}_W\end{bmatrix}$, where $\boldsymbol{u}_w$ is the J-dimensional vector representation of the wth class. The class vectors could be defined by experts, but could also be learned from data. This formulation allows classes to be correlated with each other. The one-hot vector representations used previously are directly interpretable, but using learned class representations brings back to the problem of ambiguous recall. Some kind of test is needed to determine what class a returned echo belongs to, such as using either a similarity or a distance measure, and selecting the closest or most similar class. Let $d(\boldsymbol{c}, \boldsymbol{u}_w)$ be some function that returns the similarity of $\boldsymbol{c}$ with class vector $\boldsymbol{u}_w$. A wide variety of similarity and distance measures could be used, but for this work, focus will be restricted to the cosine and dot-product similarities,

$$d_{cosine}(\boldsymbol{c}, \boldsymbol{u}_w) = \tilde{\boldsymbol{u}}_w^\top \tilde{\boldsymbol{c}} \qquad \text{cosine similarity} \tag{3.42}$$
$$d_{dp}(\boldsymbol{c}, \boldsymbol{u}_w) = \boldsymbol{u}_w^\top \boldsymbol{c} \qquad \text{dot-product similarity.} \tag{3.43}$$

The predicted class is given by,

$$\hat{w} = \underset{w=1,...,W}{argmax} \left(d(\boldsymbol{c}, \boldsymbol{u}_w) \right). \tag{3.44}$$

The different similarity measures may be useful for different situations. The cosine distance compares the direction, rather than the magnitude, of the vectors. This may be useful, given that Minerva has no innate scaling. In contrast, if the parameterised Minerva model *does* learn some meaningful magnitude for the echo, the dot-product similarity may be more effective. Both cosine and dot-product similarity allow for negative values. Either of these options can be combined with any of the models previously described.

3.5.5 Minerva-RPES: Sequence Minerva

Given the parallels between Minerva and attention, the transformer architecture can be used as inspiration for a sequence version of Minerva, which will be referred to as Minerva-RPES, and which is shown in Figure 3.5a. It is composed of two Minerva modules: the first is Minerva-RPE (see §3.5.3), which assigns initial labels to the input sequence, with no contextual information. The second Minerva module is a kind of 'self-Minerva', in which the input sequence is also used for the exemplars, making use of the labels produced by the first Minerva module.

Let $\boldsymbol{Q} = [\boldsymbol{q}_1, ..., \boldsymbol{q}_T]$ be a sequence input of length T. The first Minerva module, shown to the left in Figure 3.5a., is Minerva-RPE, which can be represented in matrix form for the entire utterance by,

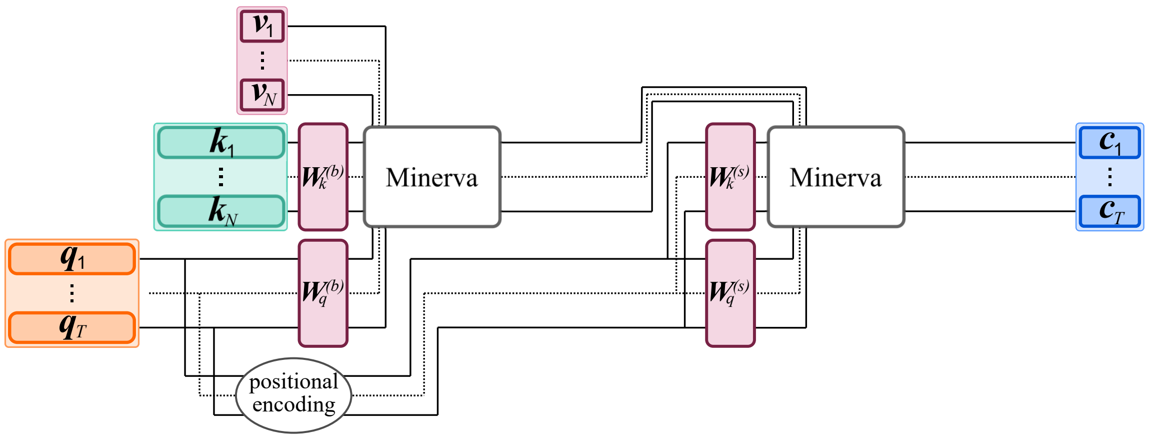
$$\boldsymbol{Q}^{(b)} = \boldsymbol{W}_q^{(b)}\boldsymbol{Q} \tag{3.45}$$

$$\boldsymbol{K}^{(b)} = \boldsymbol{W}_k^{(b)}\boldsymbol{K} \tag{3.46}$$

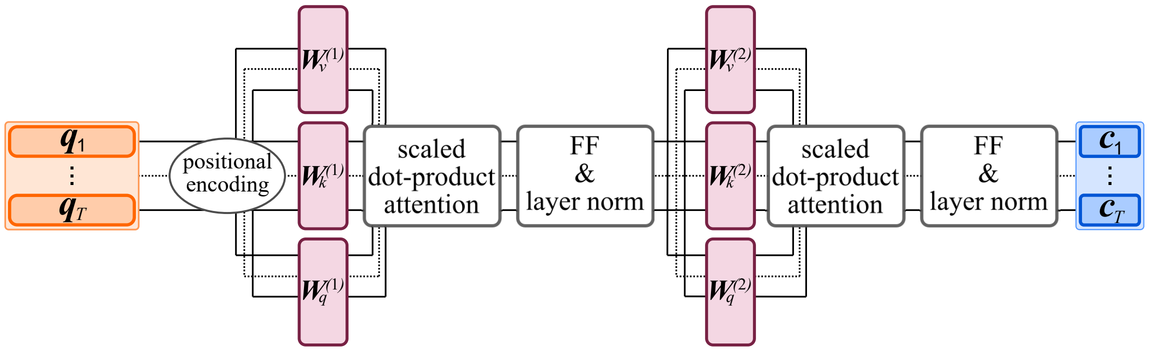
$$\boldsymbol{S}^{(b)} = \tilde{\boldsymbol{K}}^{(b)\top}\tilde{\boldsymbol{Q}}^{(b)} \tag{3.47}$$

$$\boldsymbol{A}^{(b)} = \left(\boldsymbol{S}^{(b)}\right)^{\circ\beta} \tag{3.48}$$

$$\boldsymbol{C}^{(b)} = \boldsymbol{V}\boldsymbol{A}. \tag{3.49}$$



a. Minerva-RPES.



b. Stacked self-attention.

Figure 3.5: *Comparison of Minerva-RPES with 2-layer stacked self-attention.*

$\boldsymbol{C}^{(b)} = [\boldsymbol{c}_1^{(b)}, ..., \boldsymbol{c}_T^{(b)}]$ gives the base (non-sequence) predicted labels of the sequence. Minerva takes no account of the order of an input, so drawing inspiration from transformers, a positional embedding such as RoPE (Su et al. 2024) is used. Let $pos(\boldsymbol{Q})$ be the input sequence with positional embeddings. Combined with the estimated labels, $\boldsymbol{C}^{(b)}$, Minerva can be used with the now-labelled sequence as both input and exemplars:

$$\boldsymbol{Q}^{(s)} = \boldsymbol{W}_q^{(s)} pos(\boldsymbol{Q}) \tag{3.50}$$

$$\boldsymbol{K}^{(s)} = \boldsymbol{W}_k^{(s)} pos(\boldsymbol{Q}) \tag{3.51}$$

$$\boldsymbol{S}^{(s)} = \tilde{\boldsymbol{K}}^{(s)\top} \tilde{\boldsymbol{Q}}^{(s)} \tag{3.52}$$

$$\boldsymbol{A}^{(s)} = \left(\boldsymbol{S}^{(s)}\right)^{\circ\beta} \tag{3.53}$$

$$\boldsymbol{C} = \boldsymbol{C}^{(b)} \boldsymbol{A}^{(s)} \tag{3.54}$$

The matrices $\boldsymbol{W}_q^{(b)}, \boldsymbol{W}_k^{(b)}, \boldsymbol{W}_q^{(s)}$ and $\boldsymbol{W}_k^{(s)}$ are all learned transformations. The class labels $\boldsymbol{V}$ are also trained. The model can be trained for classification or regression using Equations 3.38 or 3.39, where the unlearned parameters are $\boldsymbol{\Theta}_U = \{\boldsymbol{K}\}$, and the learned parameters are $\boldsymbol{\Theta}_L = \{\boldsymbol{W}_q^{(b)}, \boldsymbol{W}_k^{(b)}, \boldsymbol{W}_q^{(s)}, \boldsymbol{W}_k^{(s)}, \boldsymbol{V}\}$.

Since frames located close to each other are likely to be correlated, Minerva-RPES may allow similar, nearby frames to 'vote together', and reach a majority conclusion about what class they are. This might help with common confusions, but would be less useful for transition boundaries. Alternatively or additionally, the model might learn information about what classes follow each other under what conditions.

Minerva 2 has previously been adapted for sequence modelling by Maier & Moore (2007). Their Temporal Episodic Memory Model (TEMM) for speech recognition makes use of sequential exemplars, in which multiple consecutive frames of a speech signal are used as exemplars. The approach used in the Minerva-RPES model differs from this in a crucial way: rather than using a set of distinct sequential exemplars, as in TEMM, the Minerva-RPE model uses the sequential input signal itself as the exemplars.

3.5.6 Concluding remarks

In this chapter, it has been shown that Minerva 2 is a constrained FFNN, with parameters taken directly from the data. This relates directly to Research Question 1:

RQ1 What are the similarities and differences between the exemplar model Minerva 2 and ANNs?

This finding has implications for Minerva 2's performance. Given the same architecture, the modelling power of Minerva 2 and an equivalent conventionally-trained FFNN is theoretically the same; in practice they are likely to differ. The practical implications are explored experimentally in the next chapter.

Also relating to Research Question 1, it has been shown §3.3 that the iterative echo-of-echoes process is a form of DEM. This means that the echo-of-echoes process can, in principal, benefit from efficient algorithms proposed by Bai et al. (2019).

To address Research Questions 2 and 3, in §3.5 a range of hybrid prototype-exemplar models base on Minerva 2 have been proposed. These proposed models form one of the contributions of this work, and provide a framework for experiments to test the effect of feature representation, label representation and parameterisation. These experiments are reported in the next chapter.

Chapter 4

Experiments

4.1 Introduction

This chapter gives details of the experimental work carried out. Having concluded the work on Research Question 1 in Chapter 3, Research Questions 2 and 3 form the focus of this chapter:

RQ2 What exemplars should be stored in memory, and how should they be represented?

RQ3 How can an exemplar model be combined with parameters to form a hybrid exemplar-prototype model? What benefits, if any, does this bring?

This chapter begins with a description of the three speech and language tasks used in this chapter: frame-based phone recognition on the TIMIT dataset; emotion classification of text on the GoEmotions dataset; and speech intelligibility prediction on the CPC2 dataset.

The experimental work begins in §4.4 with exploration of the Minerva-R model described in §3.5.1, covering the relationship between exemplar set size and activation power β. In §4.6, the effects of feature representation are examined, not only the performance of the models, but on the relative differences in performance between models.

In §4.5 to §4.8, each of the models described in the previous chapter (§3.5) is tested and analysed. Several options for exemplar sets are explored: random and stratified by class (§4.4.2.2); fixed and changing (§4.9); and exemplar adaptation, in which exemplars are matched to the input in some way (§4.10).

4.2 Datasets

Three tasks with associated datasets were selected, covering both classification and regression, and both speech and language tasks. All of these tasks have some degree of human judgement used in their annotation, which makes them a good fit for trialling a model based on human cognition. The datasets are:

- TIMIT (Garofolo 1993), which has human-annotated phonetic labelling, as well as transcription;
- GoEmotions (Demszky et al. 2020), a collection of Reddit posts that have multi-class human-annotated emotion labels; and
- Clarity Prediction Challenge 2 (CPC2) (Barker et al. 2024), a speech intelligibility prediction task, where the intelligibility score is based on human perception of the words.

4.2.1 TIMIT

TIMIT is a dataset composed of short, single-sentence sentence utterances labelled with phonetic information as well as a text transcription (Garofolo 1993). The *training set* has 3696 utterances from 462 speakers (326 male, 136 female), the *development set* has 311 utterances from 50 speakers (32 male, 18 female), and the *test set* has 192 utterances from 24 speakers (16 male, 8 female). There is no speaker overlap between training, development and test sets. This work makes use of TIMIT's phonetic labelling, using a reduced set of 39 labels (Lopes & Perdigão 2011), rather than the 61 labels used in the original data. Figure 4.1 shows the distribution of the 39 labels for the training, test and development sets. While similar to each other, the sets are imbalanced, with the 'silence' label representing almost a quarter of the data. The least represented label is /g/, at 0.3%. All utterances are labelled with speaker ID, speaker gender and the speaker's self-reported regional accent. All speakers are native American English speakers.

4.2.1.1 Feature representation

In order to test the effects of feature representation, three representations of differing qualities were chosen for comparison. In order of quality, they are:

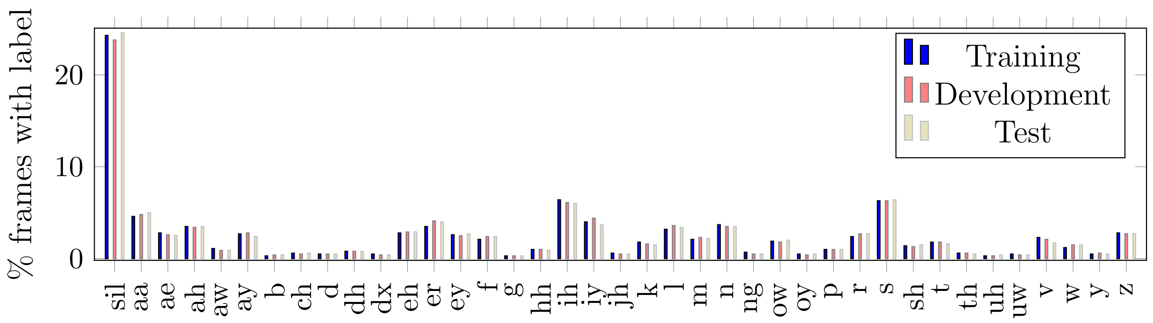


Figure 4.1: *Distribution of phones within the TIMIT data set.*

- **Log Mel spectrogram**: 96-dimensional features were obtained from the 32 channel log Mel spectrogram. The stride was 20 ms, chosen to match the Wav2vec and HuBERT features (see next), and the window was 32 ms. Delta (differential) and delta-delta (acceleration) features were also included.
- **Wav2vec2.0**: 768-dimensional features were obtained from the final layer of a pre-trained[1] Wav2vec2.0 (Baevski et al. 2020) self-supervised speech representation model. The model was pre-trained on Librispeech, using 960 hours of unlabelled data.
- **HuBERT**: 768-dimensional features were obtained from the final layer of a pre-trained[2] HuBERT (Hsu et al. 2021) self-supervised speech representation model. Like the Wav2vec features, the HuBERT model is pre-trained on Librispeech.

Mel spectrogram-based features have been widely used for speech tasks, making them an good basic option. The SSSRs Wav2vec2.0 and HuBERT have both shown promise on a range of speech tasks (see §2.7.2.2 and §2.7.2.3), but HuBERT typically outperforms Wav2vec2.0. All of these representations have been used to good effect for ASR, which is a close relative of the frame-based phone classification task being performed.

4.2.2 GoEmotions

GoEmotions (Demszky et al. 2020) is dataset of Reddit posts paired with annotated human emotion labels: *positive*, *negative*, *neutral* and *ambiguous*. This is a multi-classification task, with each utterance potentially belonging to more than one class;

[1]https://pytorch.org/audio/stable/generated/torchaudio.pipelines.WAV2VEC2_ASR_BASE_960H.html

[2]https://pytorch.org/audio/main/generated/torchaudio.pipelines.HUBERT_BASE.html

for example, one post might be considered both *neutral* and *positive* by the same or different annotators. There are 58,009 annotated posts in total, divided into defined training/development/test splits of size 43,410 / 5,426 / 5427 (80% / 10% / 10%) respectively.

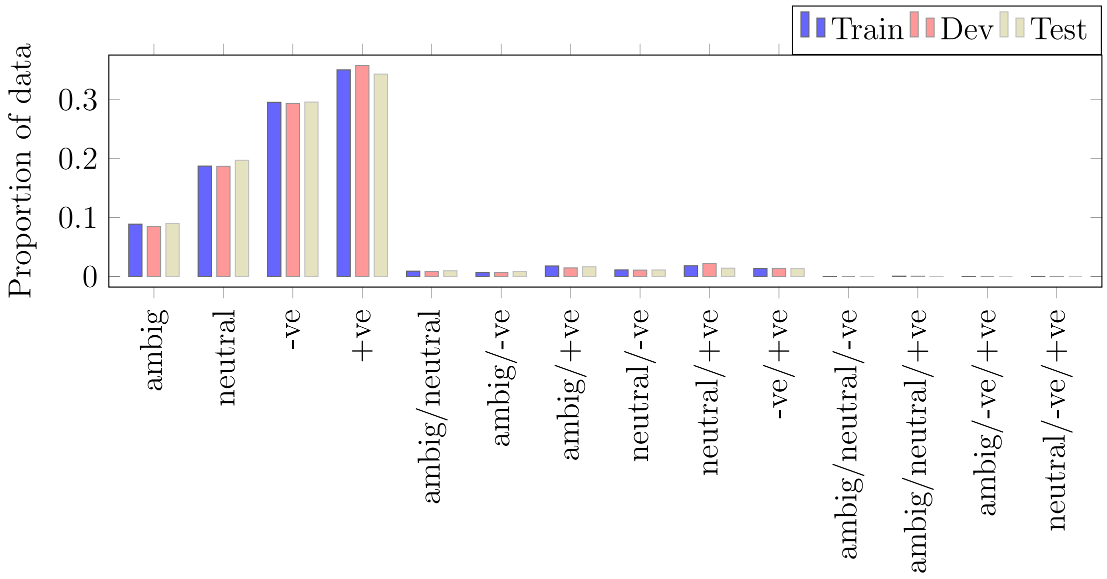


Figure 4.2: *Proportion of different classes and class combinations in the GoEmotions dataset. There is imbalance in the classes, but the proportions are similar across the training, development and test sets.*

This dataset was selected in order to test the effectiveness of the Minerva models on a multi-classification task, and also to include a text-based task. The dataset has more detailed annotation than that used here, subdividing the *negative* and *positive* classes into more specific emotions. This work made use of the labels described here, since fewer classes make it easier for data visualisation and to assess the differences between different models.

Figure 4.2 shows the prevalence of different class combinations in the dataset. For the majority of posts, annotators agreed on a single emotion, with positive emotions being the most common. Combinations of classes represent around 8% of the training data, with the most common combination being *ambiguous/positive* with 1.7% of the training data. The distribution of classes is similar for the training, development and test sets.

The human annotation of the posts is subjective, with disagreement between different annotators. The creators note that 94% of the posts have agreement between at least two annotators on at least one label; given that every post was annotated by at least three people, this leaves room for plenty of disagreement. The subjective nature of the labels makes GoEmotions a fitting task for testing a model based on human cognition.

4.2.2.1 Feature representation

Three feature representations of varying quality were selected to test the effectiveness of the models with different features. In order of quality, they are:

- **LSA**: The LSA features[3] are described in Günther et al. (2015), and are pretrained on the Touchstone Applied Science Associates (TASA) corpus. They are word-based 300-dimensional vectors, which were averaged over the words to produce a single vector representation of each sentence.
- **Word2vec**: The Word2vec features[4] (Mikolov, Chen, Corrado & Dean 2013) were obtained from a model pretrained on a part of the Google News dataset (around 100 billion words), resulting in 1024-dimensional word vectors, which were averaged over words/tokens to produce a single vector representation of each sentence.
- **SBERT**: The SBERT features[5] were obtained using the sentence-transformers python package (Reimers & Gurevych 2019), producing 768-dimensional vectors. The model is based on MPNet (Song et al. 2020), then further trained on a variety of datasets. The model output is a single vector to represent the entire sentence.

The LSA features were selected primarily because they have previously been used with Minerva 2 for comparison with human experiments (Nick Reid & Jamieson 2023). They have also been used in machine learning. They include no context information, and they are taken from a linear transformation of the term-document matrix (see §2.7.1.1). Word2vec, as a self-supervised representation, is still not context-based, but it is no longer linear, and can therefore, in principle, discover more complex patterns in the data. The SBERT features are contextual, unlike the previous options, enabling them to retain more information from the sentences. The varying quality of representation allows us to test what effect feature representation has on the performance of the models. More detail on LSA, Word2vec and SBERT is given in §2.7.1.1, §2.7.1.2 and §2.7.1.3 respectively.

[3]https://sites.google.com/site/fritzgntr/software-resources/semantic_spaces?authuser=0
[4]https://huggingface.co/fse/word2vec-google-news-300
[5]https://huggingface.co/sentence-transformers/paraphrase-mpnet-base-v2

4.2.3 Clarity Prediction Challenge 2

The CPC2 dataset (Barker et al. 2024) consists of utterances that have artificial noise added, before being enhanced by an enhancement system (a simulated hearing aid). The enhancement system is matched to a specific hearing-impaired listener, who listens to the enhanced noisy utterance, and repeats it back. The utterance is labelled with the 'correctness': the percentage of words the listener was able to repeat back correctly. The correctness is used as a measure of intelligibility. The objective of this task is to predict the correctness from the enhanced noisy speech waveform. Additional information is available, such as the clean audio and basic information about the listener's degree of hearing loss, but this work made use of only the enhanced noisy speech waveform and the correctness label.

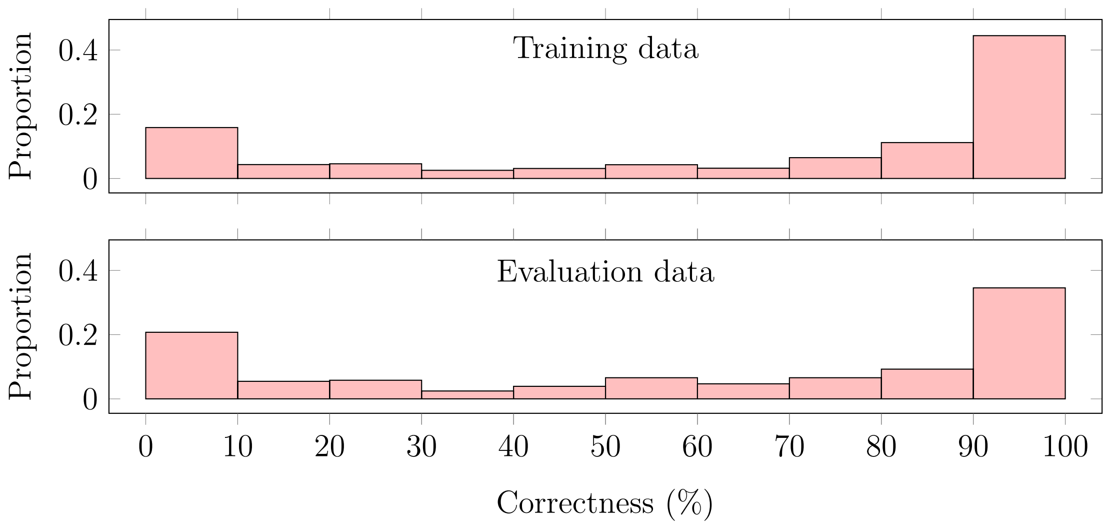


Figure 4.3: *The proportion of data for different correctness ranges in the CPC2 training data. Very high and very low correctness are common; middling values are rarer.*

The CPC2 data is divided into three training/evaluation pairs. The listeners and enhancement systems present in each evaluation set are not present in the corresponding training set, requiring models to generalise to unseen listeners and enhancement systems. There is overlap between the three training sets, but the evaluations sets are distinct. There are no defined development sets, so for each training/evaluation pair, two listeners and two enhancement systems were selected at random to form a disjoint development set. All data using these listeners and enhancement systems were removed from the training data. Of the remaining training data, 10% was separated into a non-disjoint development set. The original Split 1 has 8599/305 data pairs for training/evaluation; Split 2 has 8135/294; and Split 3 has 7896/298. Following the creation of disjoint and non-disjoint development sets, the training/non-disjoint/disjoint sizes were: 5190/577/170 for Split 1; 5087/566/169 for Split 2; and 5213/580/166 for Split 3. The disjoint development sets were used for hyperparameter tuning and model selection. The non-disjoint validation set was used to assess the difference in performance of different models on previously-seen listeners and enhancement systems, which in turn gives information on how well the model generalises to unseen listeners and enhancement systems.

There is not an even balance of correctness values within the data. Correctness values of 0 (unintelligible) and 100 (clear) are very common, while middling values are rarer. This is likely due to the nature of the task; speech is typically intelligible or not, with few examples in the middle ground. It has even been successfully treated as a binary classification task (Andersen et al. 2016).

4.2.3.1 Feature representation

Three different feature representations of varying quality were chosen. In order of quality, they are:

- **Log Spectrogram**: Fast Fourier transforms were used to compute 257-dimensional log magnitude spectrogram features, with a window of 32 ms and a stride of 16 ms.
- **XLS-R**: 1024-dimensional XLS-R features were obtained from a model pre-trained on 436k hours of multilingual data[6].
- **Whisper**: 768-dimensional features were taken from the 8th decoder layer of a pretrained[7] Whisper ASR model (Radford et al. 2023), which has been found to be effective for speech intelligibility prediction (see §5). This *small* Whisper model uses 12 encoder layers and 12 decoder layers.

The 32 kHz CPC2 waveforms were downsampled to 16 kHz prior to feature extraction, since the XLS-R and Whisper models were pre-trained on 16 kHz waveforms.

The log spectrogram, HuBERT and XLS-R features have all been shown to work for speech intelligibility prediction, with XLS-R performing best on unseen listeners and systems in the first Clarity Prediction Challenge (Close et al. 2023), and log spectrogram performing worst. The low-quality log-spectrogram features were retained, but the HuBERT features were replaced with Whisper-based features, since initial trials

[6]https://huggingface.co/facebook/wav2vec2-xls-r-300m
[7]https://huggingface.co/openai/whisper-small

with Whisper features showed very good performance, especially on unseen listeners and systems. The three feature representations therefore provide a range of feature quality. The pretrained Whisper model was pretrained prior to the CPC2 evaluation sets being released, so they cannot have been used to train it.

All features, regardless of how they were obtained, were averaged over the time-domain to provide a single vector features representation for each utterance. More sophisticated methods exist, but this simple method has been found to perform competitively, as will be shown in Chapter 5.

4.3 Shared methodology

All code was written in Python 3 and used the Pytorch package (Paszke et al. 2019). All trained models were trained using the Adam optimiser (Diederik 2014).

Unless otherwise stated, all trained models had the following hyperparameters tuned: learning rate; weight decay; dropout; and, for the Minerva models, the activation power β. Tuning was carried out on the established developments sets for the TIMIT and GoEmotions tasks, and using the constructed disjoint validation sets for the CPC2 task (see §4.2.3). Unless otherwise stated, minibatch sizes were 8, 512 and 128 for TIMIT, GoEmotions and CPC2 models respectively. Hyperparameter values for all experiments are reported in Appendix A.

All results reported in tables are the mean of five models trained with different random initialisations and, when using fixed exemplars, with different randomly selected exemplar sets. Unless otherwise stated, statistical tests between models are t-tests on the five differently initialised models.

4.4 Exemplar sets and the activation power

The exemplars are a key component of the Minerva models, and the objective of this experiment is to explore how performance changes with exemplar set size and activation power, and how the exemplar set size and activation power interact with each other. The Minerva-R model is used (see §3.5.1), which is purely exemplar-based, and has no learned parameters.

4.4.1 Methodology

Minerva-R models with varying numbers of exemplars were assessed on the TIMIT, GoEmotions and CPC2 tasks. For TIMIT, the number of exemplars varied from 39 to 79872, with exemplars selected randomly from the training data. For smaller numbers of exemplars, the exemplar set did not always include examples of all classes. The class with the highest value in the resulting echo was taken as the predicted class (see Equation 3.33).

For the GoEmotions task, exemplars were selected randomly from the training data. Exemplar set sizes ranged from 4 to 65536. As with TIMIT, small exemplar sets did not always include exemplars from every class. Since GoEmotions is a multi-classification task, meaning that each input can belong to more than one class, taking the *argmax* of the output is not appropriate. Instead, a sigmoid activation (Equation 2.14) was applied to the output, and AUC was used as the assessment metric, which does not require classification thresholds to be set (see §2.8.1.1 for more details).

For the CPC2 task, exemplars were again selected randomly from the training data, with the exemplar set size ranging from 4 to 8192. Since the Minerva-R model has no innate scaling (as explained in § 3.5.1), a calibration was performed on the model output: the exemplar features were used as input to the model, and the resulting model output was matched to the true exemplar labels using least-squares logistic regression. The parameters for this regression were then used to scale model output at inference.

The optimal activation power β for each exemplar set size was found by tuning on the development set.

4.4.2 Results and discussion

4.4.2.1 Exemplar set size and activation power

Figure 4.4 shows the performance and optimal activation power β of the Minerva-R models for each task and feature representation. The spectrogram features for the CPC2 task are excluded, since they yielded performance no better than chance on the evaluation set, and therefore the results are not meaningful. It is clear that better feature representation leads to better performance, and this is discussed in much more detail in §4.6. For now, discussion is restricted to the exemplar set.

For all tasks, as the exemplar set size increases, performance improves, although with diminishing returns. For the TIMIT and GoEmotions tasks, as the exemplar set size increases, the optimal value of the activation power β increases. For the CPC2 task, however, the optimal value for β appears to peak and then fall. The activation power

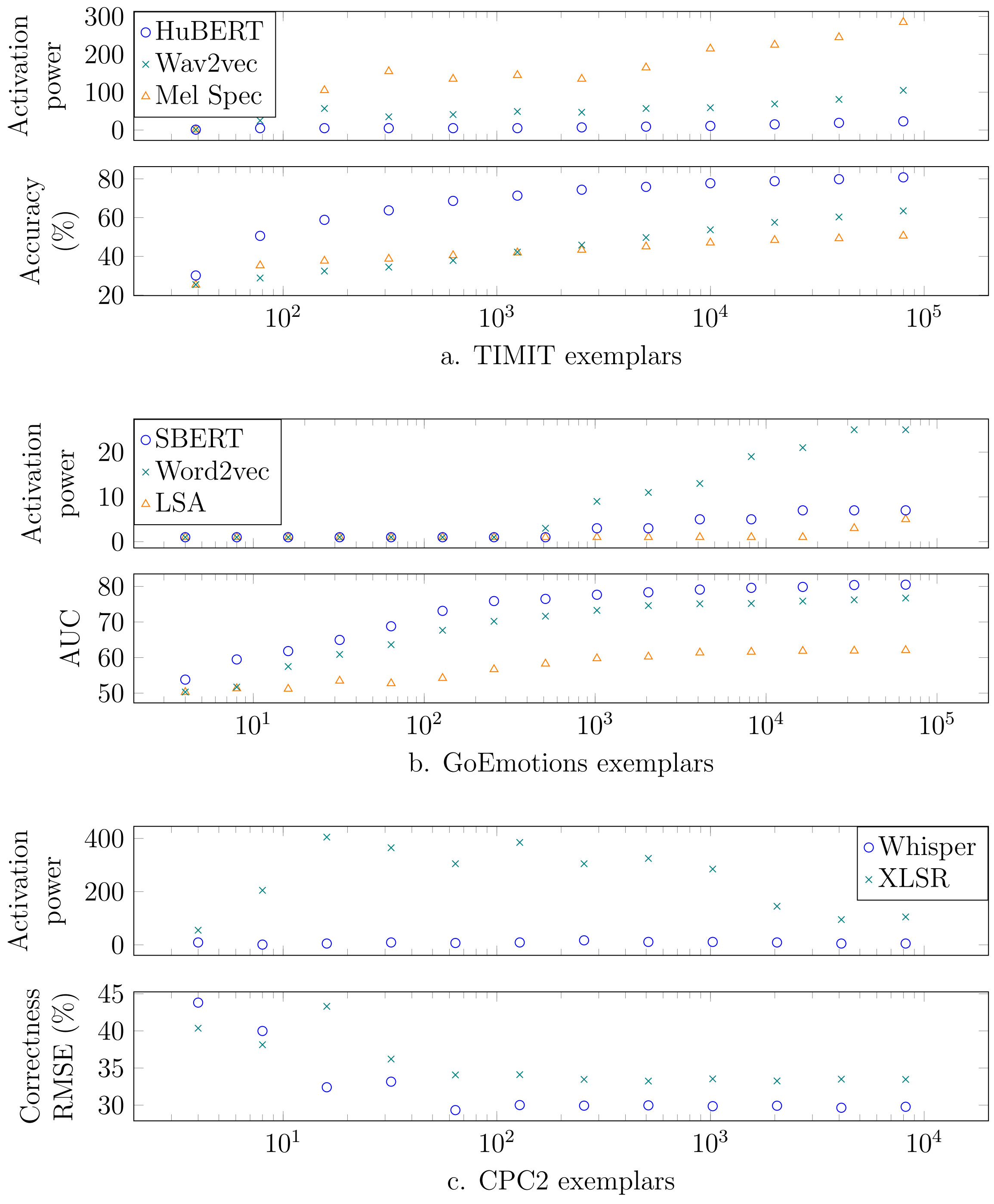


Figure 4.4: *Performance and optimal β for different exemplar set sizes.*

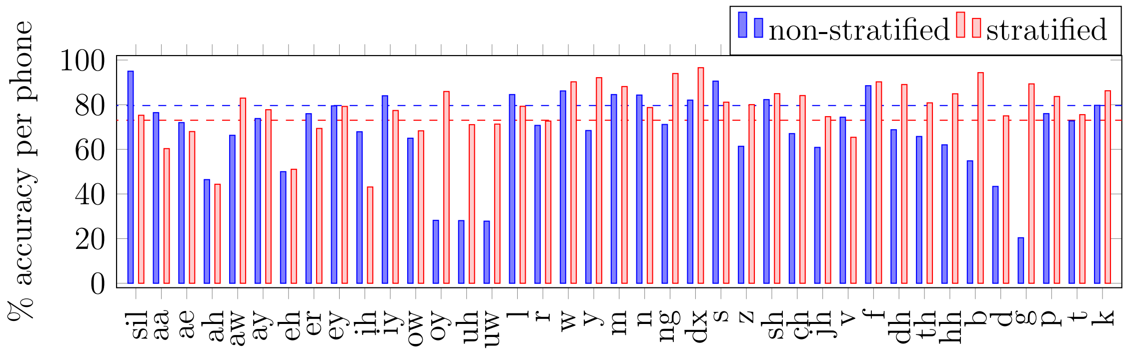
fulfills several purposes, any or all of which may affect the optimal value. It controls how sharp the activations are in the Minerva attention mechanism (Equation 2.2). When considering Minerva as a FFNN, it acts as a non-linearity, and it also affects the scale of the output: each activation weight falls between -1 and 1, and so increasing the activation power reduces the magnitude of the weights, and therefore also the magnitude of the resulting echo (see Equations 2.1 and 2.2). In contrast, increasing the exemplar set size increases the magnitude of the echo. These two effects may serve to counteract each other to some extent.

The CPC2 task differs from the other two tasks in that it is regression, not classification. The scaling of the output is therefore more critical, and may explain why the optimal value for β is 1 for the Whisper features; the linear activation function may make the calibration easier. In contrast, the optimal activation power for the XLS-R features is high for exemplar set sizes ranging from 8 to 1892, with $\beta > 100$. This results in negligible activation for any exemplar that is not extremely similar to the input. Reducing the number of highly activated exemplars may be another way of controlling the scaling of the output. Overall, predicting the effect of the activation power on the resultant echo is not always easy, and the simplest course is to treat it as a parameter to be tuned.

The activation power bears a similarity to the inverse of the temperature of a softmax function. Both fulfill a similar purpose: controlling the sharpness of the resulting output. The temperature of a softmax is known to affect confidence estimates during inference (Guo et al. 2017), but also influences training by changing the scale of the logits (Agarwala et al. 2020); it may be possible to learn similar lessons about the activation power. Furthermore, since different activation functions behave differently, there may be situations in which different activation functions are more suitable. In some classification tasks, classes may be considered ‘opposites’ of each other - for example, the positive and negative classes in the GoEmotions tasks - and so the use of an odd activation power, which preserves the sign of the weights, may be particularly useful. In other cases, a softmax function, which assumes orthogonality between classes, may be a more meaningful choice.

4.4.2.2 TIMIT exemplar set stratification

Given the imbalance in the phonetic classes for the TIMIT data set (see Figure 4.1), a randomly selected exemplar set is likely to contain substantially more exemplars from common classes, such as silence, and far fewer exemplars from rare classes, such as /g/. Smaller exemplar sets may have *no* exemplars from rare classes, which would render the model incapable of classifying that class. To determine what effect this has on the performance of the models, two different exemplar selection strategies were compared: random sampling; and random sampling stratified by class. In both cases, 14976 exemplars were selected from the training data. For the stratified exemplar set, this resulted in 384 exemplars from each of the 39 classes. Figures 4.5 shows the accuracy for each selection strategy by class for the HuBERT feature representations.



a. Minerva-R accuracy by phone

Figure 4.5: *Comparison of classification accuracy on the TIMIT test set by phonetic class for the Minerva-R model using HuBERT features, with and without stratification of the exemplars by class. The dashed lines are the overall model accuracies.*

Although the non-stratified Minerva-R model performs substantially better on overall accuracy (78.55% compared with 73.04% for the stratified model, $p < 0.01$), it has extremely poor performance on four classes in particular: /g/, /oy/, /uh/ and /uw/, which are rare classes, making up 0.3%, 0.5%, 0.3% and 0.5% of the training set respectively. The non-stratified model is better at predicting silence, which, given that it makes up around 24% of the data (both training and test), gives it a substantial advantage in overall accuracy, despite it performing worse on 28 out of 39 classes. Discounting silence, the accuracies of the models are 73.16% and 72.31% for the non-stratified and stratified exemplar sets respectively, meaning that the non-stratified exemplar set still provides better accuracy, although the difference between the models is substantially reduced. The non-stratified model's superior accuracy comes largely from its better performance on the classes /s/ and /ih/, which are the second and third most common class in the test set after silence.

Figure 4.6 shows the confusion matrices for misclassified frames for the non-stratified and stratified models. The non-stratified model misclassifies as silence a great deal, especially for consonants, which can be seen as a bar of colour to the bottom right. It misclassifies over 40% of the frames labelled /uw/ as /v/, and a similar proportion of

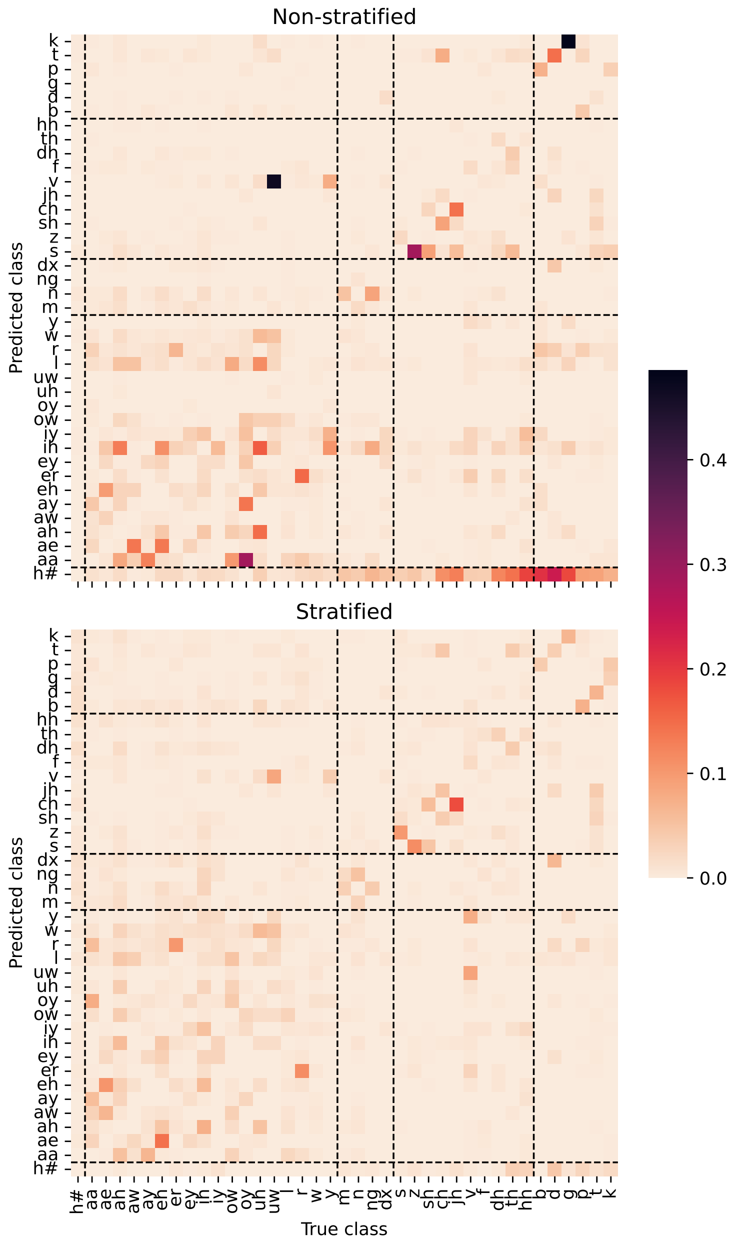


Figure 4.6: *Comparison of the confusion matrices of incorrectly classified frames using the non-stratified (top) and stratified (bottom) Minerva-R models trained on TIMIT. The colour represents the proportion of the true class that has been misclassified as the predicted class.*

the frames labelled /g/ are misclassified as /k/. The exemplar sets are large enough that all classes are represented, even in the non-stratified model, but the imbalance in the exemplars clearly affects the model's ability to classify the less common phonetic classes. Neither of these two confusions is particularly common for the stratified model, although shared confusions certainly exist: /jh/ as /ch/; /z/ as /s/; and /eh/ as /ae/ are common for both models.

Imbalanced datasets therefore have implications for exemplar models and exemplar selection. Is it better to sacrifice performance on rare classes in order to perform better overall? Or is rare class performance as important as common class performance? The answers to these questions are likely to depend on the task being performed.

4.5 Learned feature transformations

Adding learned feature transformations, as described in §3.5.2, is one option for parameterising Minerva. The objective of this experiment is to explore the effects of adding learned feature transformations, as described in Equations 3.34 to 3.36, and to answer the following questions:

1. What should the feature transformation dimension be?
2. Should the linear transformations of the input and exemplar features be the same ($\boldsymbol{W}_q = \boldsymbol{W}_k$) or different ($\boldsymbol{W}_q \neq \boldsymbol{W}_k$)?

4.5.1 Methodology

Minerva-RP models, as described in §3.5.2, were trained on TIMIT, GoEmotions and CPC2 with both separate and shared feature transformations, and with a range of different transformation dimensions. For comparison purposes, Minerva-R models with the same number of exemplars and baseline FFNNs were also assessed.

For the TIMIT task, the Minerva-R and Minerva-RP models used 14976 exemplars. In initial tests, the model was found to train poorly when there was no normalisation on the output, so L2 normalisation was added prior to the softmax as discussed in 3.5.2.2. The model is trained with cross entropy loss (Equation 3.39).

For the GoEmotions task, the Minerva-R and Minerva-RP models used 1024 exemplars. The exemplar labels, and therefore the echo, are 4-dimensional, with elements representing each of the 4 classes. Since any given input can belong to more than one class, using L2 normalisation, as described for the TIMIT task, is not appropriate. Instead, learned linear scaling is applied to each output element individually, allowing each class to be scaled separately. This is followed by a sigmoid activation (Equations 2.14), to produce separate probabilities for each class. Binary cross entropy loss summed across all classes is used for training.

For the CPC2 task, the Minerva-R and Minerva-RP models used 128 exemplars. The correctness values for each speech clip is a scalar, and so is the output of the model. A final learned transformation is therefore added, as described in §3.5.2.1, prior to the final sigmoid function. MSE loss is used for training (Equation 3.38).

The 3-layer FFNN models trained for comparison had a hidden layer with dimension 1024 for the TIMIT and CPC2 tasks, and 512 for the GoEmotions task, since larger FFNNs were found to over-train. The model parameters were initialised with Kaiming initialisation (He et al. 2015).

Hyperparameters for all models were tuned as described in 4.3, and final hyperparmeter values are given in Appendix A in Tables A.2, A.3 and A.4 for the TIMIT, GoEmotions and CPC2 models respectively.

4.5.2 Results and discussion

4.5.2.1 Feature transform dimension

Figure 4.7 shows the performance of the Minerva-RP models with separate transforms as the feature transform dimension increases, for each of the tasks, and with each of the feature representations (log spectrogram feature for CPC2 are not shows, as they yielded performance no better than chance on the CPC2 evaluation set). For the TIMIT task, performance increases up to a transform dimension of 64, with transform dimension larger than this offering no statistically significant improvement ($p > 0.1$). A similar pattern is seen with the GoEmotions data, but with performance plateauing at a feature transform dimension of 32. For the CPC2 task, performance *decreases* at higher transform dimensions, likely due to over-fitting, and despite extensive experimentation with L2-regularisation and dropout. In all cases, the ideal feature transform dimension is smaller than the input features, often by a large margin: the highest quality features for each task (HuBERT for TIMIT, SBERT for GoEmotions and Whisper for CPC2) all have a dimension of 768. This reduction in the feature dimension likely reduces the 'noise' in the features, retaining only information that is relevant to the task at hand. The dimension reduction is linear, since there is no activation function, and in this respect is a relative of Principal Component Analysis (PCA). It differs from PCA in that the dimension reduction is supervised and performed with a specific task in mind, rather than unsupervised through explaining the variance.

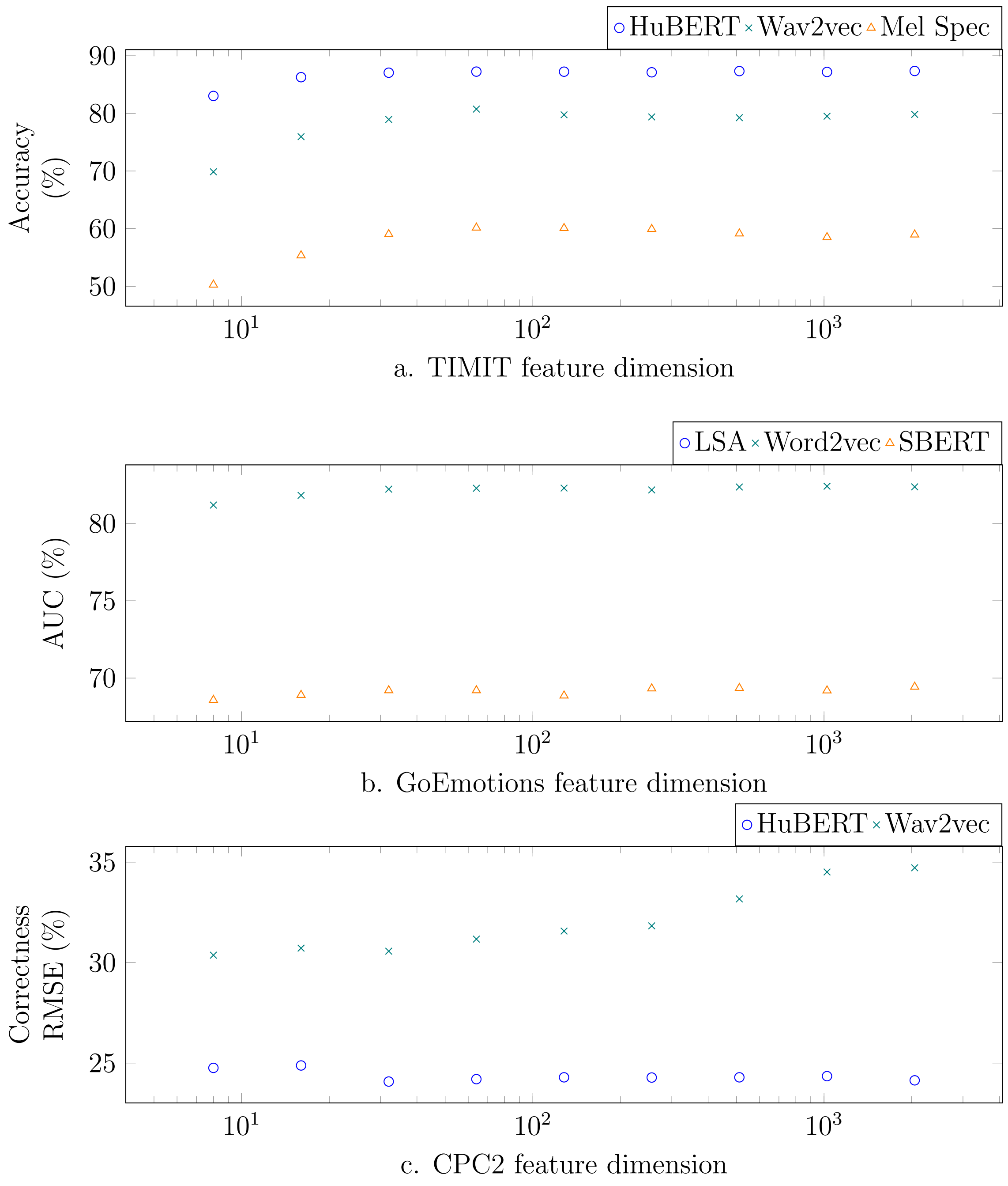


Figure 4.7: *Performance of the Minerva-RP model with separate feature transforms, for different feature transform dimensions.*

Table 4.1: *Frame-based phone classification performance on the TIMIT development and tests sets using the Minerva-R and Minerva-RP models, with baseline FFNN for comparison, using different feature representations. Standard deviations are given in parentheses. The best result is in bold.*

Features	**Classifier**	**Feature transform**	**β**	**Learned params**	**Accuracy (%)**	
					Dev	**Test**
Mel Spec	Minerva-R	-	135	0	41.32 (0.28)	40.64 (0.31)
	Minerva-RP	Shared	9	0.01 M	57.75 (0.06)	56.92 (0.23)
		Separate	9	0.01 M	60.91 (0.17)	59.93 (0.24)
	FFNN	-	-	1.19 M	69.71 (0.05)	68.20 (0.08)
Wav2vec	Minerva-R	-	55	0	50.67 (0.17)	50.69 (0.19)
	Minerva-RP	Shared	9	0.05 M	79.44 (0.08)	78.33 (0.07)
		Separate	9	0.10 M	80.78 (0.21)	79.62 (0.20)
	FFNN	-	-	1.88 M	82.29 (0.05)	81.07 (0.12)
HuBERT	Minerva-R	-	15	0	72.94 (0.24)	72.59 (0.41)
	Minerva-RP	Shared	7	0.05 M	87.94 (0.04)	87.24 (0.07)
		Separate	7	0.10 M	88.20 (0.06)	87.35 (0.07)
	FFNN	-	-	1.88 M	88.39 (0.04)	**87.66 (0.02)**

All of the high-quality feature representations have been shown to work well for a variety of tasks: HuBERT has been used for speech emotion recognition, speaker verification and spoken language understanding (Wang et al. 2021), as well as for ASR (Hsu et al. 2021); the SBERT-based features have been used for sentiment prediction and question-type classification amongst other tasks (Reimers & Gurevych 2019); and Whisper-based features have been shown to work for speech sentiment analysis and speaker diarisation (Papala et al. 2023). Some of the information required for these tasks is likely to be irrelevant to the specific tasks being carried out here. Reducing the dimension to exclude irrelevant information (or noise) is therefore likely to make the similarity measure with Minerva more meaningful. This also explains the substantial reduction seen in the optimal value of the activation power β: with less noise, the correlation between feature vectors of the same class is likely to be higher, making them more distinguishable.

4.5.2.2 Minerva-RP model performance

Tables 4.1, 4.2 and 4.3 show the Minerva-RP results for the ideal feature transform dimension (64 for TIMIT, 32 for GoEmotions and CPC2), along with the equivalent Minerva-R results and results from the baseline FFNNs, for the TIMIT, GoEmotions and CPC2 tasks respectively. It can be seen that the Minerva-RP models give a substantial improvement in performance over the equivalent Minerva-R models for all feature representations and tasks (t-tests: $p < 0.01$ in all cases), but do not perform as well as a similarly-complex FFNN. This result is hardly surprising, since it has already been established that Minerva-RP is a special case of 2-layer FFNN with restricted parameters. The baseline FFNN has an additional layer, and access to a larger parameter space, and is therefore expected to perform better.

Table 4.2: *Performance on the GoEmotions task using different feature representations. Standard deviations are given in parentheses. The best result is in bold.*

Features	**Classifier**	**Feature transform**	$\boldsymbol{\beta}$	**Learned params**	**AUC** (%)	
					Dev	**Test**
LSA	Minerva-R	-	1	0	61.47 (0.26)	61.36 (0.34)
	Minerva-RP	Shared	1	0.01 M	67.91 (0.08)	68.71 (0.06)
		Separate	3	0.02 M	69.16 (0.06)	69.67 (0.12)
	FFNN	-	-	1.36 M	69.98 (0.07)	70.95 (0.05)
Word2vec	Minerva-R	-	19	0	75.12 (0.17)	75.69 (0.17)
	Minerva-RP	Shared	3	0.03 M	80.49 (0.09)	80.76 (0.07)
		Separate	3	0.07 M	81.98 (0.07)	82.09 (0.05)
	FFNN	-	-	1.36 M	83.23 (0.05)	83.30 (0.09)
SBERT	Minerva-R	-	5	0	79.59 (0.19)	79.45 (0.22)
	Minerva-RP	Shared	3	0.02	84.59 (0.09)	84.77 (0.09)
		Separate	3	0.05 M	85.00 (0.04))	85.27 (0.07)
	FFNN	-	-	1.84 M	85.50 (0.04)	**86.02 (0.04)**

Using separate feature transforms rather than shared feature transforms for Minerva-RP improves performance in all cases, although the difference is smaller for higher quality feature representations. Allowing the feature transforms to differ gives more modelling power, and also lets the Minerva attention mechanism attend to different aspects of the features.

The version of Minerva-RP with shared feature transforms has a benefit though: very high inductive bias. The model performs substantially better than chance prior to training, achieving 68% accuracy on TIMIT with HuBERT features. The same benefit is not seen with the others tasks, in this instance, since they use learned scaling in addition to the Minerva mechanism (see §3.5.2 for details), but it would be possible to initialise them using the same method for scaling used in the Minerva-R models. This suggests an intriguing possibility: use Minerva as an initialisation for an FFNN.

Table 4.3: *Performance on the CPC2 task using Minerva-R and Minerva-RP models, with baseline FFNN for comparison, using different feature representations. Standard deviations are given in parentheses. The best result is in bold.*

Features	**Classifier**	**Feature transform**	β	**Learned params**	**RMSE**	
					Dev	**Test**
Log spec	Minerva-R	-	195	0	42.30 (0.08)	40.74 (0.05)
	Minerva-RP	Shared	1	0.01 M	30.07 (0.10)	42.12 (0.42)
		Separate	1	0.02 M	29.95 (0.09)	40.13 (0.81)
	FFNN	-	-	0.53 M	28.78 (0.07)	40.17 (0.54)
XLSR	Minerva-R	-	115	0	33.28 (0.16)	36.71 (0.42)
	Minerva-RP	Shared	1	0.03 M	27.50 (0.25)	33.13 (0.79)
		Separate	1	0.07 M	24.19 (0.08)	27.82 (0.28)
	FFNN	-	-	1.31 M	23.28 (0.02)	27.70 (0.27)
Whisper	Minerva-R	-	1	0	28.03 (0.30)	29.24 (0.46)
	Minerva-RP	Shared	1	0.02 M	23.11 (0.17)	25.64 (0.72)
		Separate	1	0.05 M	22.72 (0.04)	25.05 (0.56)
	FFNN	-	-	1.05 M	22.68 (0.06)	**24.47 (0.24)**

Many existing initialisations select initial values to control the variance of the layer output, rather than to improve performance (Glorot & Bengio 2010, He et al. 2015), but data-driven initialisations have also been derived, often by using some form of clustering (Gan et al. 2015, Krähenbühl et al. 2015, Alberti et al. 2017). Scaling is an issue, however. As previously noted, the Minerva attention mechanism doesn't include scaling, resulting in outputs that can be substantially larger or smaller than the inputs, so some method for controlling the variance of each layer's outputs would be required, and this presents an interesting option for future work.

4.5.2.3 Non-stratified and stratified exemplars for TIMIT

As seen in §4.4.2.2, Minerva-R performs better with non-stratified exemplars, but as Figure 4.8 shows, this is not the case for Minerva-RP, for which the results for the two options are extremely similar. It appears that learning a feature transformation allows the model to correct for imbalances in the exemplar set, while still retaining high performance on common classes. There is no statistically significant difference between the Minerva-RP models with non-stratified and stratified exemplars ($p > 0.1$ in all cases). Further models trained on TIMIT and discussed in this work use stratified exemplars, since this produces more repeatable results when using smaller exemplar sets, and gives equivalent performance.

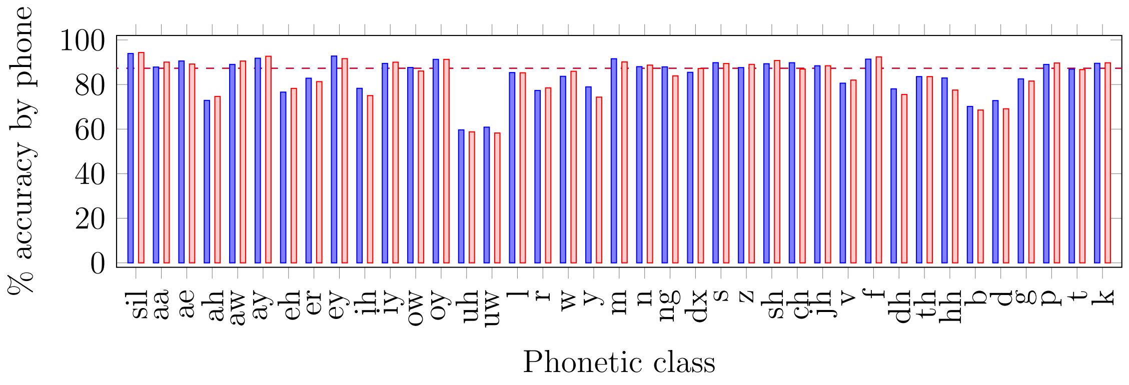


Figure 4.8: *Comparison of classification accuracy on the TIMIT test set by phonetic class for the Minerva-RP models using HuBERT features, with and without stratification of the exemplars by class. The dashed lines are the overall model accuracies.*

4.6 The impact of feature representation

Improving feature representation improves performance for all models across all tasks, as can be seen from Tables 4.1, 4.2 and 4.3. More interesting are the relative differences between the models: the difference between models reduces with improved feature representation. This trend is seen for both the TIMIT and GoEmotions tasks. On TIMIT, the difference in performance between the Minerva-R and FFNN models drops from 27.6% for Mel spectrogram features to 15.1% with HuBERT features. For the Minerva-RP models, the differences are smaller but follow the same pattern, dropping from 8.3% to 0.3%. On GoEmotions, the difference in performance between the Minerva-R and FFNN models drops from 9.6% for the LSA features to 6.5% with SBERT features. For Minerva-RP on GoEmotions, the difference drops from 2.21% with LSA features to 1.5% with SBERT features. For CPC2, neither the Minerva-R nor Minerva-RP models are effective on the evaluation set using Mel spectrogram features, and since the difference between the FFNN and Minerva-RP models is not statistically, significant, the order is less meaningful. The Minerva-R model performance difference from the FFNN follows the same pattern, however, dropping from 9.0% RMSE for the XLSR features to 4.8% RMSE for the Whisper features.

Likely reasons for the reduced difference in performance between models include: reduction in the amount of noise in the high–quality features, making Minerva's attention mechanism more effective; and improved linearity, in which case the reduced parameter space and lack of conformity to the universal approximation theorem of Minerva-R and Minerva-RP becomes less significant.

Memory-based models, such as dynamic time warping for speech recognition, or concatenative speech synthesis, have been competitive in the past, but they have fallen out of use. In contrast, these results suggest that, as feature representations continue to improve, simple, interpretable memory-based models such as Minerva-R become more competitive.

4.7 Learned exemplar labels

The objective of these experiments is to determine:

1. whether learned exemplar labels as described in §3.5.3 improve performance; and
2. what form the labels take.

4.7.1 Methodology

Minerva-RPE models using separate feature transforms were trained on the TIMIT, GoEmotions and CPC2 tasks and compared with equivalent Minerva-RP models, which do not have learned exemplar labels. The TIMIT models used 384 exemplars per class (14976 total), and a feature transformation dimension of 64. The GoEmotions models used 1024 exemplars, and a feature transformation dimension of 32. The CPC2 models used 128 exemplars, and a feature transformation dimension of 32. All models used the same scaling/normalisation as the equivalent Minerva-RP models. The exemplar labels were initialised one-hot according to the class of the exemplar.

All hyperparameters were tuned as described in §4.3, and final hyperparameter values are given in Appendix A in Tables A.5, A.6 and A.7 for the TIMIT, GoEmotions and CPC2 models respectively.

4.7.2 Results and discussion

Tables 4.4, 4.5 and 4.6 show the results for the TIMIT, GoEmotions and CPC2 tasks respectively. For TIMIT with Mel spectrogram features, learning exemplar labels with Minerva-RP gives a large improvement in performance over the Minerva-RP models of around 7.0% ($p < 0.001$). For the Wav2vec features, a smaller improvement of around 1.1% ($p < 0.001$) is obtained. A very small but still statistically significant improvement is found with the HuBERT features of 0.12% ($p = 0.01$).

Table 4.4: *Frame-based phone classification on TIMIT using Minerva-RP and Minerva-RPE models. Standard deviations are given in parentheses. The best result is in bold.*

Features	**Classifier**	β	**Learned params**	**Accuracy (%)**	
				Dev	**Test**
Mel Spec	Minerva-RP*	9	0.01 M	60.91 (0.17)	59.93 (0.24)
	Minerva-RPE	7	0.59 M	68.31 (0.06)	66.98 (0.22)
Wav2vec	Minerva-RP*	9	0.10 M	80.78 (0.21)	79.62 (0.20)
	Minerva-RPE	5	0.68 M	82.11 (0.07)	80.74 (0.09)
HuBERT	Minerva-RP*	7	0.10 M	88.20 (0.06)	87.35 (0.07)
	Minerva-RPE	5	0.68 M	88.32 (0.03)	**87.47 (0.04)**

*Reproduced from Table 4.1.

For the GoEmotions task, improvement in performance between Minerva-RP and Minerva-RPE is not significant in the case of the LSA and Word2vec features ($p = 0.7$ and $p = 0.15$ respectively) but it is weakly significant for the SBERT feature representations ($p = 0.04$), although the improvement in the AUC is very small.

Table 4.5: *Emotion classification of text on GoEmotions using Minerva-RP and Minerva-RPE models. Standard deviations are given in parentheses. The best result is in bold.*

Features	**Classifier**	β	**Learned params**	**AUC** (%)	
				Dev	**Test**
LSA	Minerva-RP*	3	0.02 M	69.16 (0.06)	69.67 (0.12)
	Minerva-RPE	3	0.02 M	69.05 (0.10)	69.64 (0.13)
Word2vec	Minerva-RP*	3	0.07 M	81.98 (0.07)	82.09 (0.05)
	Minerva-RPE	3	0.07 M	82.25 (0.10)	82.19 (0.14)
SBERT	Minerva-RP*	3	0.05 M	85.00 (0.04))	85.27 (0.07)
	Minerva-RPE	3	0.05 M	85.06 (0.04)	**85.36 (0.04)**

*Reproduced from Table 4.2.

For the CPC2 task, there is no improvement for the log spectrogram features, but this is unsurprising, since neither the Minerva-R nor the Minerva-RP models perform better than chance on the evaluation set. For both the XLSR and Whisper features, there is a statistically significant improvement of 0.5% correctness ($p = 0.002$) for the XLSR features, and 0.7% correctness ($p = 0.03$) for the Whisper features.

Table 4.6: *Speech intelligibility prediction on CPC2 using Minerva-RP and Minerva-RPE models. Standard deviations are given in parentheses. The best result is in bold.*

Features	**Classifier**	β	**Learned params**	**RMSE**	
				Dev	**Test**
Log spec	Minerva-RP*	1	0.02 M	29.95 (0.09)	40.13 (0.81)
	Minerva-RPE	1	0.02 M	29.79 (0.17)	40.88 (0.71)
XLSR	Minerva-RP*	1	0.07 M	24.19 (0.08)	27.82 (0.28)
	Minerva-RPE	1	0.07 M	23.54 (0.05)	27.34 (0.22)
Whisper	Minerva-RP*	1	0.05 M	22.72 (0.04)	25.05 (0.56)
	Minerva-RPE	1	0.05 M	22.76 (0.01)	**24.36 (0.19)**

*Reproduced from Table 4.3.

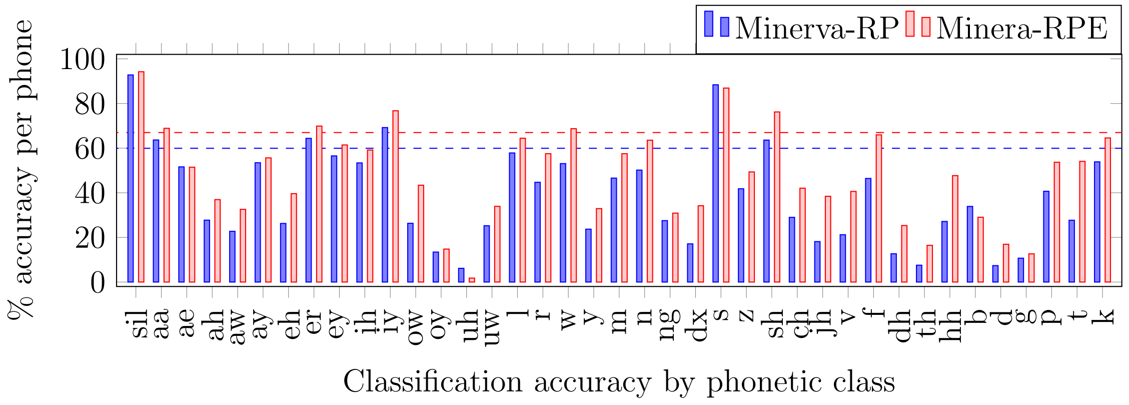


Figure 4.9: *Comparison of classification accuracy on the TIMIT test set by phonetic class for the Minerva-RP and Minerva-RPE models using Mel spectrogram features. The dashed lines are the overall model accuracies.*

Where is the improvement coming from on the TIMIT and CPC2 tasks? Starting with the TIMIT model using Mel spectrogram features, a combination which gave the largest improvement in performance from learning exemplar class labels, Figure 4.9 shows the performance by class of the Minerva-RP and Minerva-RPE models on the TIMIT test set. Minerva-RPE performs better than Minerva-RP on all classes but three: /uh/, /s/ and /b/. Of these classes, /uh/ and /b/ are rare, each representing

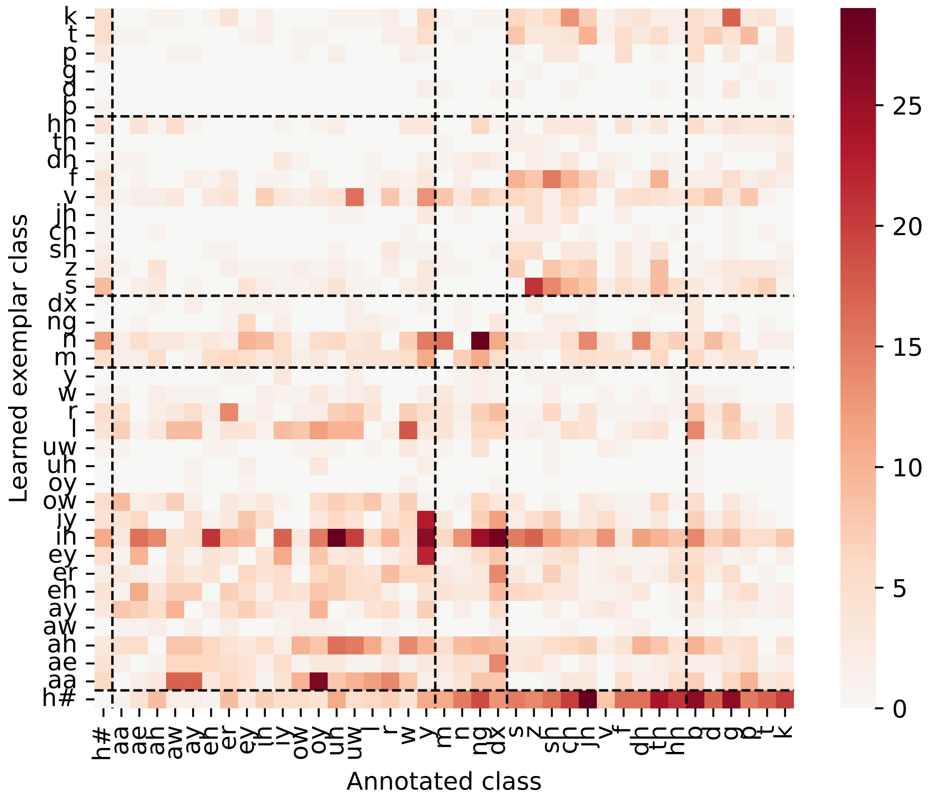


Figure 4.10: *Heat map showing the annotated class and learned exemplar class of exemplars for which they are not the same, for the TIMIT Minerva-RPE model using Mel spectrogram features.*

0.4% of the TIMIT test set. The class /s/ is the second most common, at 6.4%, so the model is not learning common classes at the expense of rare ones.

There are three label types for each exemplar: firstly, its annotated label from the data set; secondly, its learned exemplar label; and thirdly, the predicted label when it is used as input to the model. For the TIMIT model using Mel spectrogram features, the learned exemplar labels do not match the annotated labels around 28% of the time. Despite this, over 50% of these exemplars with ‘incorrect’ learned labels have correctly predicted classes when classified using the model. A similar pattern is seen with the HuBERT Minerva-RPE model: around 30% of exemplar labels do not match the annotated labels, but nevertheless, around 73% of these exemplars with ‘incorrect’ learned labels are still predicted correctly. It is clear that the learned exemplar label

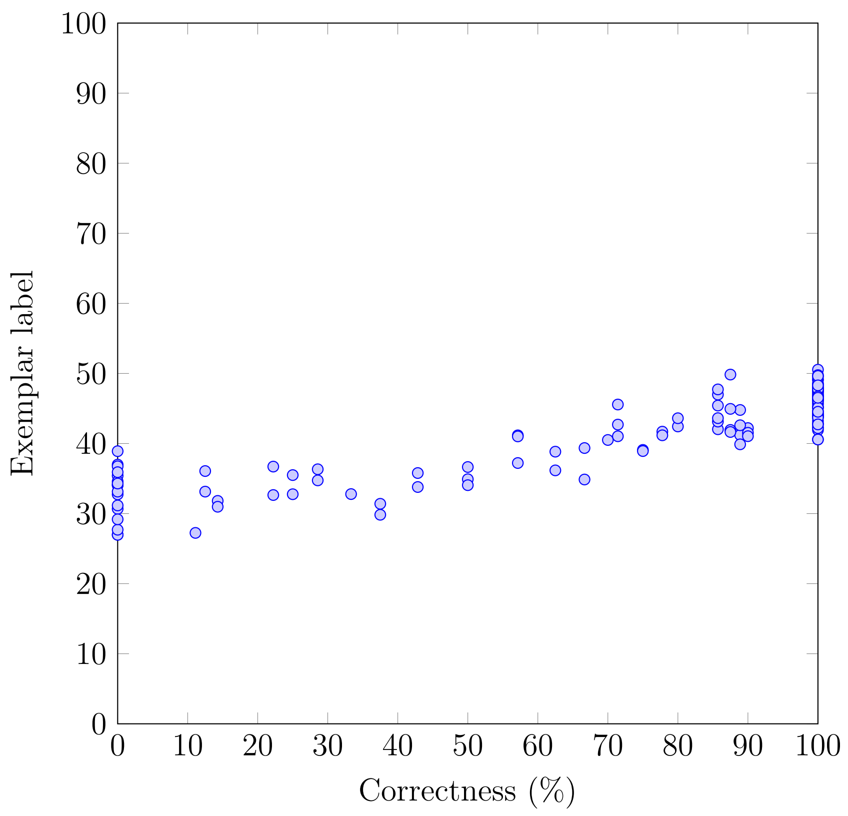


Figure 4.11: *Minerva-RPE exemplar correctness vs learned exemplar labels.*

space does not perfectly match the prediction label space. This makes some sense, since the labelling of each input is affected by every exemplar label to some extent, and therefore will not perfectly match the closest exemplar. The exception to this is when the activation power $\beta \to \infty$, in which case Minerva approximates 1-nearest neighbour (see §2.2). In this case, the exemplar label space and the prediction label space will ideally match each other perfectly. The smaller β becomes, the more effect distant exemplars have, and the more different the labels spaces have to become to accommodate this.

The CPC2 exemplar labels are easier to visualise, since they are scalars. Figure 4.11 shows them plotted against the correctness labels from the data set. There is a clear correlation, but while the correctness labels are clustered around 0% correctness and 100% correctness, the exemplar labels are bunched much more closely together: the total range is from 27 to 50. Given that each predicted label is a weighted sum of learned exemplar labels, it is not at all surprising that the scaling of the two label sets is different, but the correlation is imperfect, meaning that the prediction and exemplar label spaces are once again different. In this case, some of this may be due to the listener variability. The correctness scores are derived from a single listener attempting to repeat back the words they have heard. Given an identical speech clip, different listeners can potentially give quite different correctness scores. Learning the exemplar labels allows the model to adjust for this.

4.8 Learned 'true' class representations

Previous models have all used one-hot encoding for the 'true' class representations. As discussed in §3.5.4, this makes the class representations orthogonal to each other, which may not reflect reality. For example, two of the emotions in the GoEmotions emotion classification task are *positive* and *negative*. It seems reasonable to assume that these will be negatively correlated, rather than orthogonal. Likewise, on the TIMIT phone classification task, this makes the assumption that all classification errors are equally important, which may not be true: misclassifying /ih/ as /eh/ seems more reasonable than misclassifying /ih/ as /g/, for example.

The objective of this experiment is to determine what benefit there may be from using the learned 'true' class representations described in §3.5.4, and compare them with previous models to see if performance is improved.

4.8.1 Methodology

Minerva-RPC and Minerva-RPCE models (see §3.5.4) were trained for the TIMIT and GoEmotions tasks. The Minerva-RPC model replaces one-hot representation for the classes with learned class representations, allowing classes to be correlated with each other. The Minerva-RPCE model does the same, but additionally learns a label for each individual exemplar.

The TIMIT models used 384 exemplars per class (14976 total), and a feature transformation dimension of 64. The exemplars were stratified by class and randomly selected from the training data. The GoEmotions models used 1024 exemplars, randomly selected from the training data, and a feature transformation dimension of 32. All models used the same scaling/normalisation as the equivalent Minerva-RP models, described in §4.5.1. In all cases, the class representations were initialised as one-hot. Since the CPC2 task is not a classification task, it was excluded from this experiment.

All hyperparameters were tuned as described in §4.3, and details are given in Appendix A in Tables A.8 and A.9 for the TIMIT and GoEmotions models respectively.

4.8.2 Results and discussion

Table 4.7 shows the results for the Minerva-RPC and Minerva-RPCE models trained on TIMIT, along with comparison models from earlier experiments. In all cases, the Minerva-RPC models performed better than the Minerva-RP models. The difference was not statistically significant for the Mel spectrogram features ($p = 0.43$), but was weakly significant for the Wav2vec ($p = 0.01$) and HuBERT features ($p = 0.04$).

Table 4.7: *Frame-based phone classification on TIMIT with learned class representations. Standard deviations are given in parentheses. The best result is in bold.*

Features	Classifier	β	Learned params	Accuracy (%) Dev	Accuracy (%) Test
Mel Spec	Minerva-RP*	9	0.01 M	60.91 (0.17)	59.93 (0.24)
	Minerva-RPC	7	0.01 M	62.95 (0.01)	61.94 (0.34)
	Minerva-RPE†	7	0.59 M	68.31 (0.06)	66.98 (0.22)
	Minerva-RPCE	5	0.59 M	68.29 (0.06)	66.95 (0.12)
	FFNN*	-	1.19 M	69.71 (0.05)	68.20 (0.08)
Wav2vec	Minerva-RP*	9	0.10 M	80.78 (0.21)	79.62 (0.20)
	Minerva-RPC	7	0.10 M	81.19 (0.05)	79.96 (0.12)
	Minerva-RPE†	5	0.68 M	82.11 (0.07)	80.74 (0.09)
	Minerva-RPCE	5	0.68 M	82.18 (0.03)	80.84 (0.13)
	FFNN*	-	1.88 M	82.29 (0.05)	81.07 (0.12)
HuBERT	Minerva-RP*	7	0.10 M	88.20 (0.06)	87.35 (0.07)
	Minerva-RPC	9	0.10 M	88.26 (0.01)	87.46 (0.07)
	Minerva-RPE†	5	0.68 M	88.32 (0.03)	87.47 (0.04)
	Minerva-RPCE	7	0.68 M	88.33 (0.03)	87.50 (0.04)
	FFNN*	-	1.88 M	88.39 (0.04)	**87.66 (0.02)**

*Reproduced from Table 4.1.
†Reproduced from Table 4.4.

Adding learned exemplar class representations is more beneficial than adding learned 'true' class representations, however, as shown by the Minerva-RPE models outperforming the Minerva-RPC models in all cases. This is not surprising, since the learned exemplar class representations can provide more power to the model than learning 'true' class representations, as can be seen from the number of learned parameters in Table 4.7.

The Minerva-RPCE models use both learned exemplar representations *and* learned 'true' class representations, but this appears to provide no benefit over the Minerva-RPE models, which learn only the exemplar class representations ($p = 0.83$ for Mel spectrogram features, $p = 0.18$ for Wav2vec features and $p = 0.25$ for HuBERT features).

Figure 4.12 shows a heat map representation of the similarity matrix for the learned 'true' class representations from the Minerva-RPC model using Mel spectrogram features. The similarities are obtained by taking the dot-product of the learned class representations with each other. For the one-hot representation used in previous experiments, this would result in 1 along the diagonal, and 0 elsewhere. Figure 4.12 differs from this in two ways:

1. The scale of each class differs, as can be seen by some points on the diagonal being darker than others; and
2. Some of the classes have slight negative correlation with each other, which shows as pale blue in Figure 4.12.

To make the cross-class similarities easier to see, Figure 4.13 shows the same similarity matrix for the Mel spectrogram features with the diagonal set to 0. Interestingly, the most dissimilar pairs are, phonetically, similar to each other: /s/ and /z/, /aw/ and /ey/, and /g/ and /k/. Referring to Figure 4.15, which shows the model performance by phonetic class for the Minerva-RP and Minerva-RPC models, it can be seen that performance is reduced on class /s/, but improved by a larger amount on class /z/. Likewise, performance is slightly reduced on class /ay/, but improved by on /aw/. Both /g/ and /k/ see improved performance. Rather than learning similar representations for similar classes, in many cases the model instead appears to learn *dissimilar* representations for common confusions. There is an exception: /m/ and /n/ are phonetically similar and commonly confused, but their learned 'true' class representations are similar, as can be seen from the dark orange colour in Figure 4.13. Interestingly, both of these classes see slightly improved performance using the Minerva-RPC model.

Figure 4.14 shows the learned 'true' class similarity matrix for the Minerva-RPC model trained on TIMIT using HuBERT features. This is very close to being a diagonal matrix, with any cross-similarity between classes being very small, meaning that the class representations are orthogonal to each other, and hence differ from the previously-used one-hot class representations only in the magnitudes of the class representations (shown by lighter and darker squares on the diagonal). In this case, scaling the class

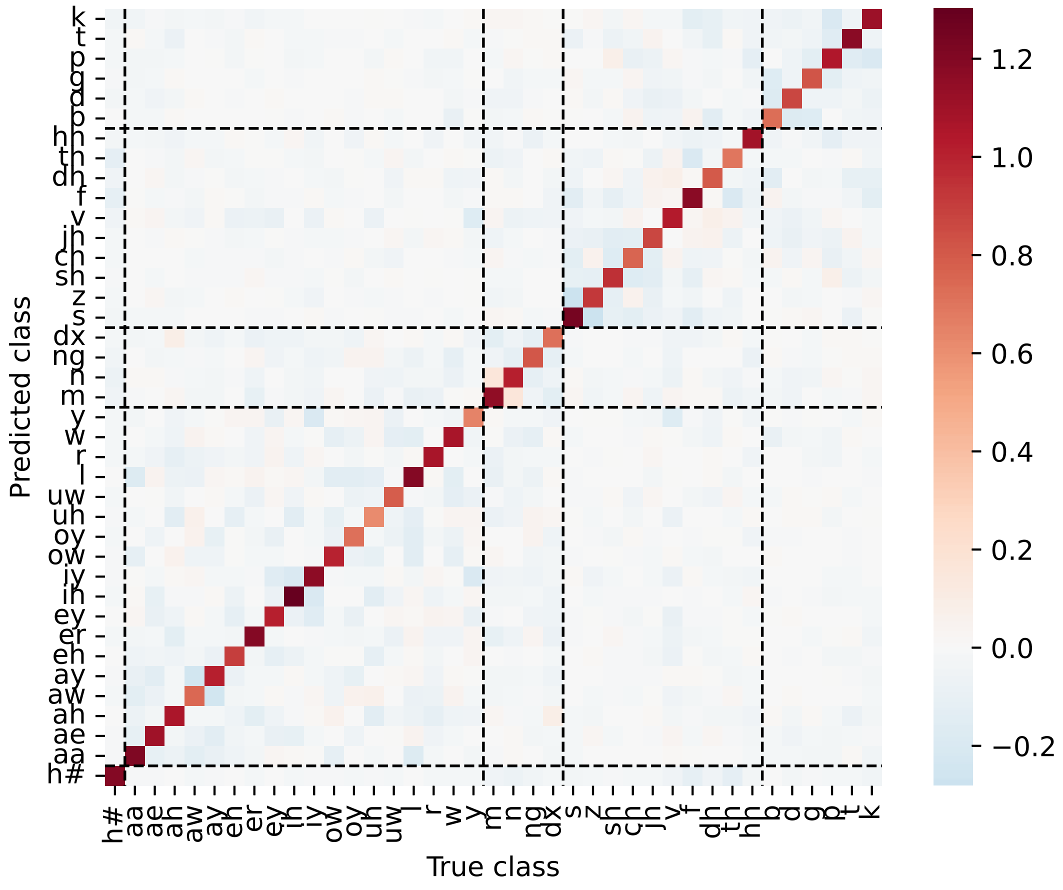


Figure 4.12: *Heat map representation of the covariance matrix for the learned 'true' class representations for the TIMIT Minerva-RPC model trained on Mel spectrogram features*

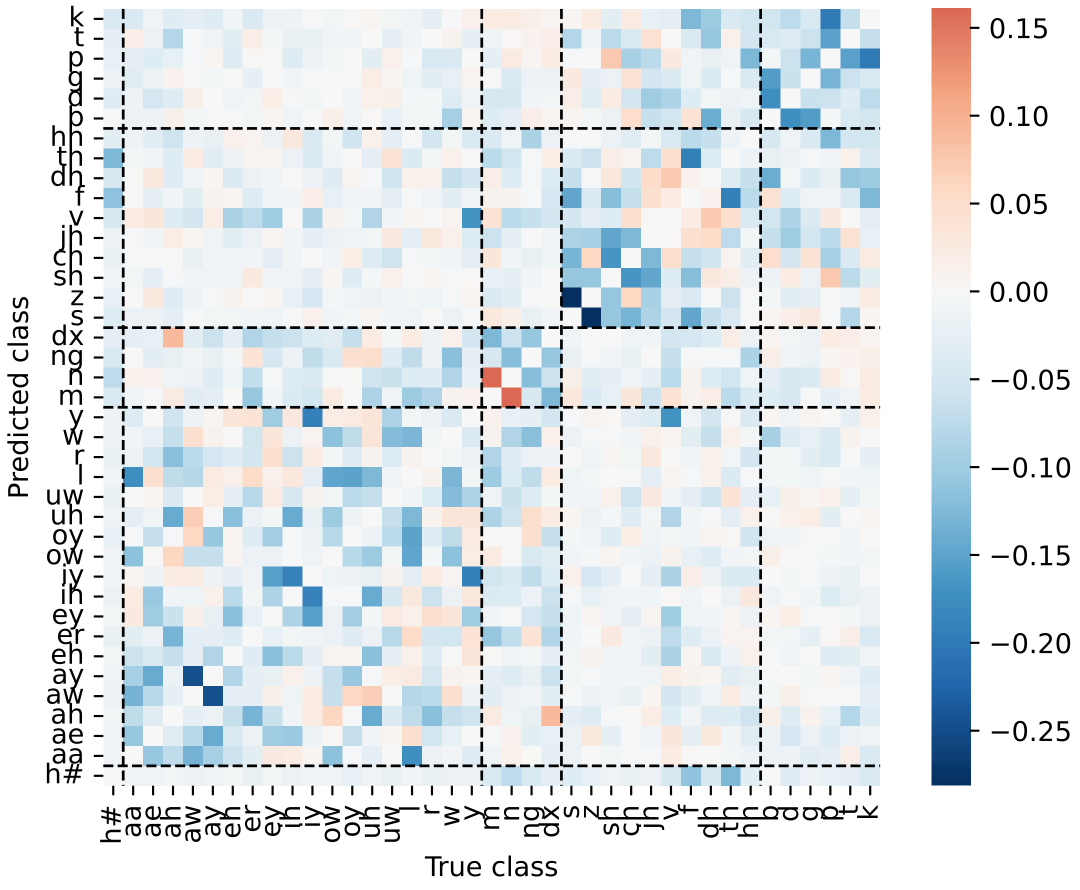


Figure 4.13: *Heat map representation of the covariance matrix for the learned 'true' class representations for the TIMIT Minerva-RPC model trained on Mel spectrogram features*

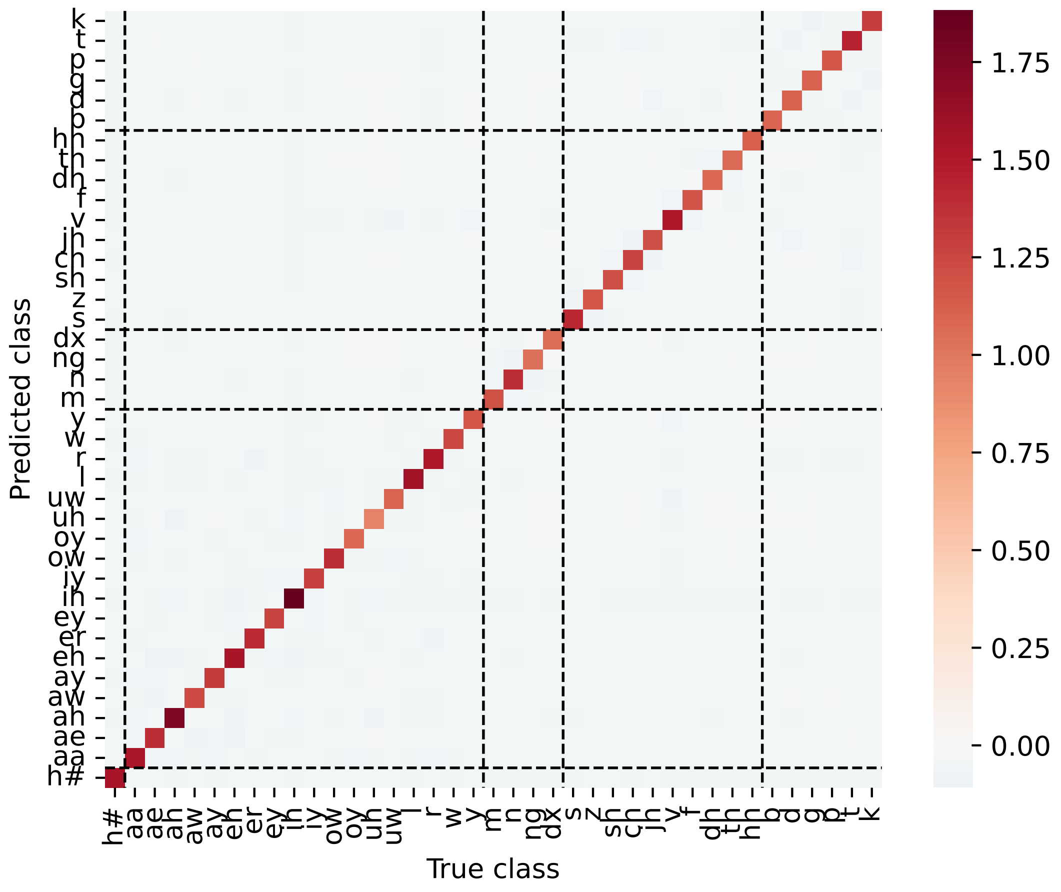


Figure 4.14: *Heat map representation of the covariance matrix for the learned 'true' class representations for the TIMIT Minerva-RPC model trained on HuBERT features.*

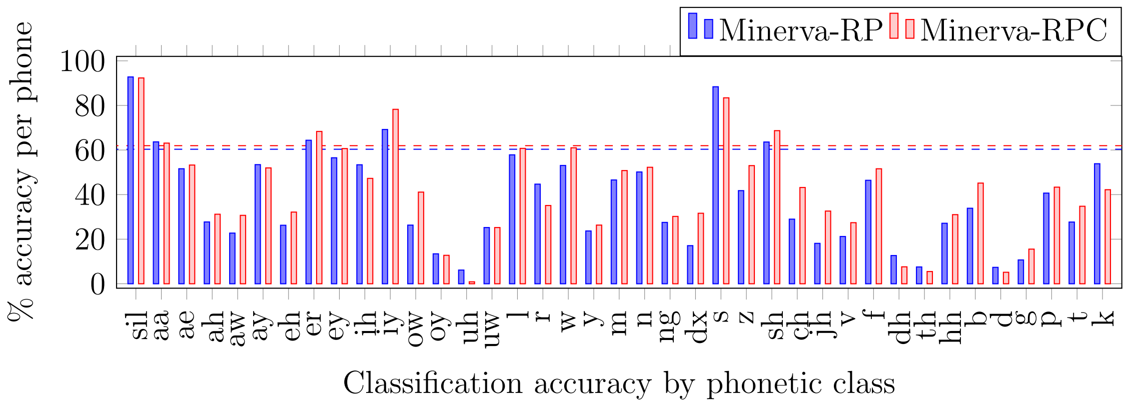


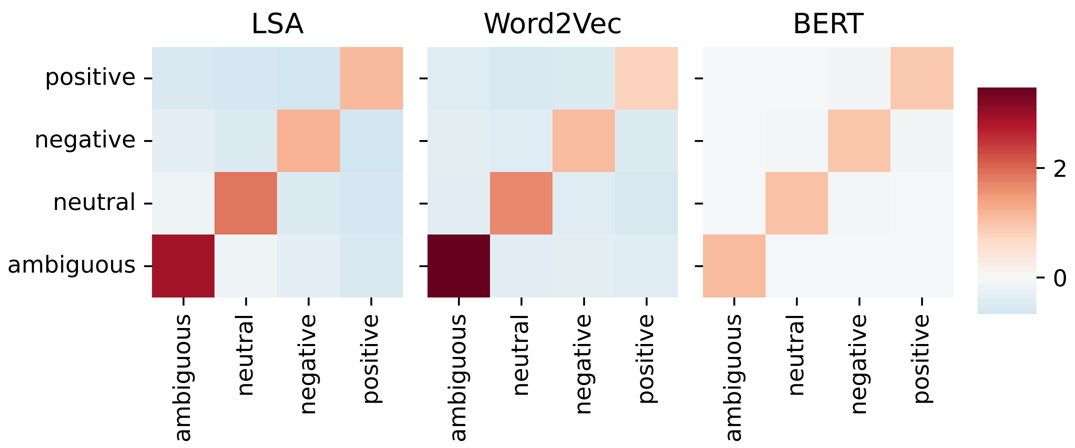
Figure 4.15: *Comparison of classification accuracy on the TIMIT test set by phonetic class for the Minerva-RP and Minerva-RPC models using Mel spectrogram features. The dashed lines are the overall model accuracies.*

representations to have different magnitudes appears to make little difference to the overall performance.

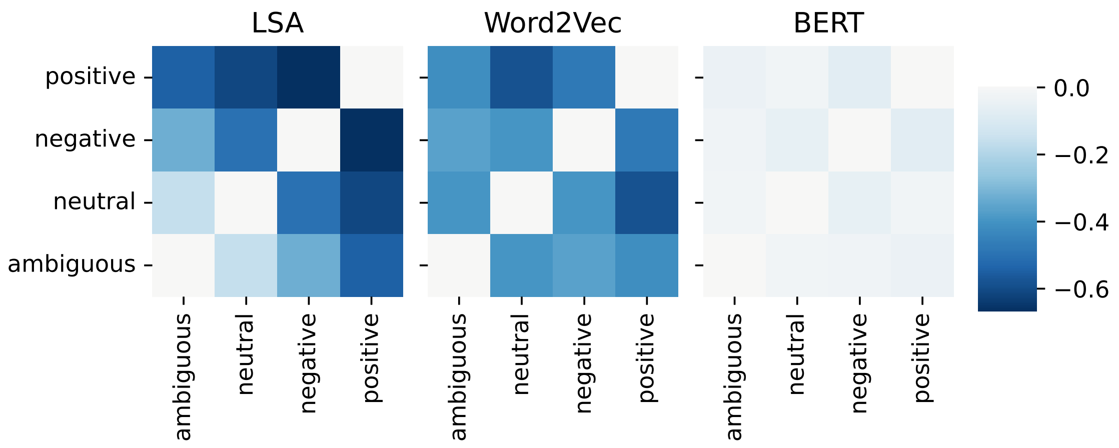
Table 4.8 shows the performance of the Minerva-RPC and Minerva-RPCE models on the GoEmotions task. The Minerva-RPC model shows no improved performance over the Minerva-RP model for the LSA features ($p = 0.15$) and SBERT features ($p = 0.06$), but shows a small but statistically significant improvement for the Word2vec features ($p < 0.01$). As with the TIMIT task, there is no benefit to having both learned exemplar representations and learned 'true' class representations: the differences between the Minerva-RPE and Minerva-RPCE models are not statistically significant ($p > 0.05$ in all cases).

Figure 4.16a. shows the similarity matrices for the 'true' class representations for each of the feature representations. Figure 4.16b. shows the same matrix, but with the diagonal elements set to zero to make the cross-class similarities easier to see. There are strong negative cross-class correlations for LSA and Word2vec features, as well as differences in the magnitudes of the class representations. The SBERT class representations, on the other hand, are balanced in terms of magnitude, and have very little cross-class correlation; they are effectively one-hot. This explains why there is no improvement for this feature set.

For the LSA and Word2vec features, the *positive* and *negative* classes are anti-correlated, as expected, but interestingly, the *neutral* class is strongly negatively correlated with the *positive* class as well. Negative correlation between two classes means that, if an input is similar to exemplars from the first class, it makes it less likely to belong to



a. All class similarities.



b. Only cross-class similarities.

Figure 4.16: *Heat map representations of the similarities between the learned 'true' class representations for the Minerva-RPC model trained on GoEmotions with LSA, Word2vec and SBERT features showing a. all class similarities; and b. only cross-class similarities.*

Table 4.8: *Emotion classification of text on GoEmotions using Minerva models. Standard deviations are given in parentheses. The best result is in bold.*

Features	**Classifier**	$\boldsymbol{\beta}$	**Learned params**	**AUC** (%)	
				Dev	**Test**
LSA	Minerva-RP*	3	0.02 M	69.16 (0.06)	69.67 (0.12)
	Minerva-RPC	3	0.02 M	69.22 (0.10)	69.85 (0.21)
	Minerva-RPE†	3	0.02 M	69.05 (0.10)	69.64 (0.13)
	Minerva-RPCE	3	0.02 M	69.08 (0.15)	69.61 (0.17)
	FFNN*	-	1.36 M	69.98 (0.07)	70.95 (0.05)
Word2vec	Minerva-RP*	3	0.07 M	81.98 (0.07)	82.09 (0.05)
	Minerva-RPC	3	0.07 M	82.30 (0.11)	82.27 (0.10)
	Minerva-RPE†	3	0.07 M	82.25 (0.10)	82.19 (0.14)
	Minerva-RPCE	3	0.07 M	82.38 (0.08)	82.37 (0.07)
	FFNN*	-	1.36 M	83.23 (0.05)	83.30 (0.09)
SBERT	Minerva-RP*	3	0.05 M	85.00 (0.04)	85.27 (0.07)
	Minerva-RPC	3	0.05 M	85.01 (0.13)	85.14 (0.11)
	Minerva-RPE†	3	0.05 M	85.06 (0.04)	85.36 (0.04)
	Minerva-RPCE	3	0.05 M	85.01 (0.12)	85.25 (0.11)
	FFNN*	-	1.84 M	85.50 (0.04)	**86.02 (0.04)**

*Reproduced from Table 4.2.
†Reproduced from Table 4.5.

the second class. This relationship is unsurprising for the *positive* and *negative* classes, since a single sentence is unlikely to belong to both. For the Word2vec features, however, the negative correlation is stronger between the *positive* and *neutral* classes than between the *positive* and *negative* classes, which is not intuitive.

4.9 Unfixed exemplars

All previous experiments have been conducted with a fixed exemplar set, but the exemplar set can in principle be changed, either during inference or during training, or both. Previous experiments have used a fixed exemplar set, in which case the output can be represented as a function of the input, $\boldsymbol{q}$,

$$\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{q}; \boldsymbol{K}, \boldsymbol{V}, \boldsymbol{\Theta}). \tag{4.1}$$

In this case, the exemplar features $\boldsymbol{K}$ and exemplar labels $\boldsymbol{V}$ are treated as parameters. They remain fixed throughout inference, along with any other parameters of the model, $\boldsymbol{\Theta}$, which might include the learned linear feature transform in Minerva-RP, and/or the 'true' class representations in the Minerva-RPC model. Using unfixed exemplars means that they are no longer treated as parameters. Now the output of the model depends on both the input and the exemplars,

$$\boldsymbol{y}_j = \boldsymbol{f}(\boldsymbol{q}, \boldsymbol{K}_j, \boldsymbol{V}_j; \boldsymbol{\Theta}). \tag{4.2}$$

One of the benefits of a changing exemplar set is that multiple variants of the same model can be used at inference, each using the same parameters, but a different exemplar set. Since the model need only be trained once, this is an efficient way of creating an ensemble model,

$$\boldsymbol{y}_e = \frac{1}{J} \sum_{j=1}^{J} \boldsymbol{f}(\boldsymbol{q}, \boldsymbol{K}_j, \boldsymbol{V}_j; \boldsymbol{\Theta}) \tag{4.3}$$

where $\boldsymbol{y}_e$ is the output of the ensemble.

Alternatively, inference can be performed with a larger exemplar set than the model is trained with. Since larger exemplar sets give better performance (see §4.4), this may provide better performance at inference without additional computational requirements for training.

The objective of this experiment is to determine what effect a changing exemplar set has on the performance of the Minerva-RP and Minerva-RPC models.

4.9.1 Methodology

Minerva-RP and Minerva-RPC models using randomly selected exemplars were trained on the TIMIT, GoEmotions and CPC2 tasks. All models used separate transforms $\boldsymbol{W}_q \neq \boldsymbol{W}_k$ for the input and exemplars (see Equations 3.34 to 3.36). A new exemplar set was randomly selected for each new minibatch, during both training and inference.

The TIMIT models used 384 exemplars per class (14976 total), and a feature transformation dimension of 32. The GoEmotions models used 1024 exemplars and a feature transform dimension of 32. The CPC2 models used 128 exemplars. The feature transformation dimension was 32.

All models used the same scaling/normalisation as the equivalent Minerva-RP models (see §4.5.1). All hyperparameters were tuned as described in §4.3, and final hyperparameters are given in Appendix A in Tables A.10, A.11 and A.12 for the TIMIT, GoEmotions and CPC2 models respectively.

Once models were trained, they were evaluated in two ways:

1. With k different exemplar sets of the same size they were trained with, forming an ensemble, as described in Equation 4.3; and
2. With an exemplar set k times larger than the one they were trained with.

The value k will be referred to as the *exemplar set multiple*, since it gives the factor of increase in the number of exemplars used at inference compared to training. For a given value of k, both options use exactly the same exemplars.

When using option 2, the GoEmotions and CPC2 models require an adaptation to the scaling to account for the larger number of exemplars. Both models use a sigmoid activation as the final activation to produce an output that falls between 0 and 1. In both cases, the logits scaled by a factor of $1/k$ prior to the sigmoid. The value of k ranged from 1 to 10.

4.9.2 Results and discussion

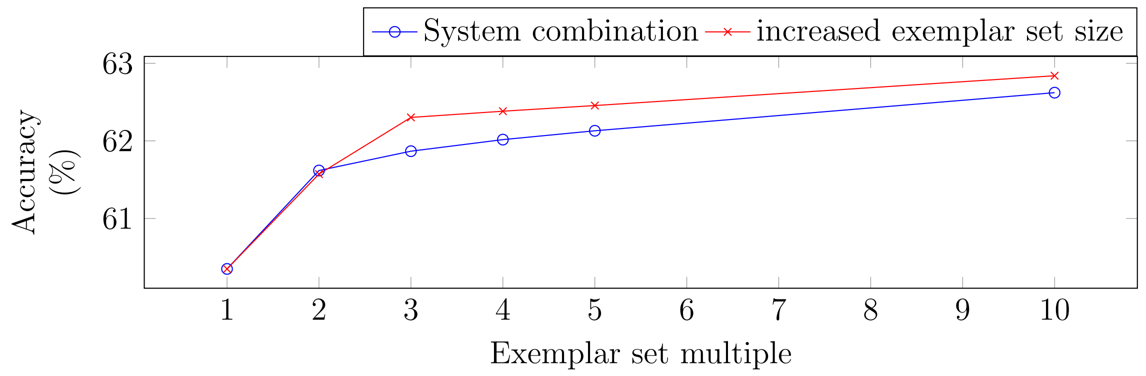
Table 4.9 shows performance of the Minerva-RP and Minerva-RPC models with fixed and changing exemplars trained on the TIMIT task.

In all cases, the models using changing exemplars performed worse than the models with fixed exemplars. This is not surprising, since randomly changing exemplar will inevitably add 'noise' into the model. This might, in some situations, provide regularisation, in much the same way that dropout does. It differs from dropout, however, in that the noise will still be present during inference.

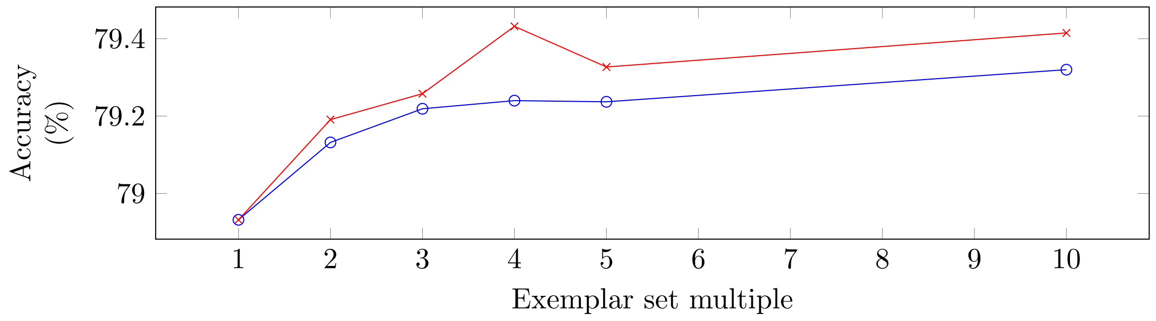
Figure 4.17 shows the performance on the TIMIT test set of ensembles with increasing numbers of systems, compared with models using an exemplar set increasing size. The x-axis shows the exemplar set size multiplier: in the case of the ensembles, this is the number of models forming the ensemble; in the case of the larger exemplar sets, this is the factor by which the exemplar set size has been increased from the training set size.

Performance generally increases as the exemplar multiplier increases, with similar improvements for the two methods. The Mel spectrogram models benefit most from increasing the number of exemplars, with an improvement in accuracy of around 2%. The improvement using HuBERT features is less that 0.2%. For both the Wav2vec and HuBERT features, the fixed exemplar set models still perform better than the unfixed exemplar models, despite the unfixed models using 10 times as many exemplars.

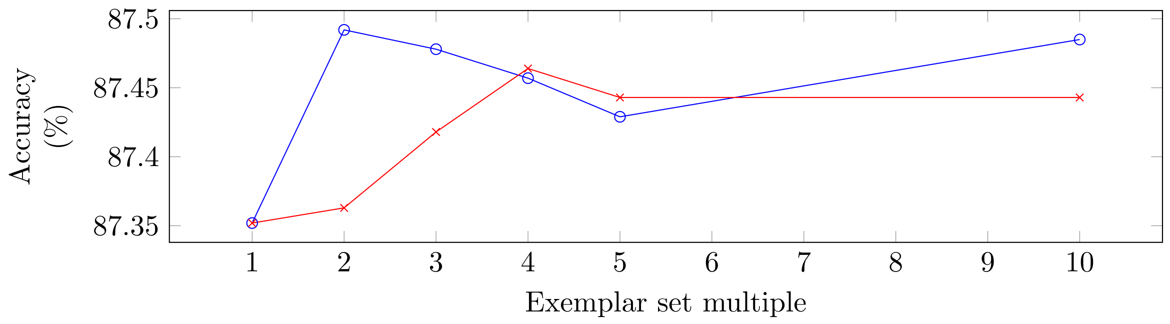
The performance of the fixed and unfixed Minerva-RP and Minerva-RPC models on the GoEmotions task is given in Table 4.10. As with the TIMIT results, the unfixed



a. Mel spectrogram features



b. Wav2vec features



C. HuBERT features

Figure 4.17: *Performance of Minerva-RPC models on TIMIT test set with either: system combination of models with different exemplar sets; or increased exemplar set size.*

Table 4.9: *Frame-based phone classification on TIMIT using Minerva-RP and Minerva-RPC models, with fixed and unfixed exemplars. Standard deviations are given in parentheses.*

Features	**Model**	**Exemplars**	β	**Learned params**	**Accuracy (%) Dev**	**Accuracy (%) Test**
Mel Spec	Minerva-RP	Fixed*	9	0.01 M	60.91 (0.17)	59.93 (0.24)
		Unfixed	9	0.01 M	58.76 (0.22)	58.10 (0.23)
	Minerva-RPC	Fixed†	7	0.01 M	62.95 (0.01)	61.94 (0.34)
		Unfixed	9	0.01 M	61.27 (0.11)	60.29 (0.18)
Wav2vec	Minerva-RP	Fixed*	9	0.10 M	80.78 (0.21)	79.62 (0.20)
		Unfixed	9	0.10 M	79.57 (0.06)	78.57 (0.13)
	Minerva-RPC	Fixed†	7	0.10 M	81.19 (0.05)	79.96 (0.12)
		Unfixed	7	0.10 M	80.06 (0.12)	78.97 (0.09)
HuBERT	Minerva-RP	Fixed*	7	0.10 M	88.20 (0.06)	87.35 (0.07)
		Unfixed	7	0.10 M	88.01 (0.04)	87.27 (0.06)
	Minerva-RPC	Fixed†	9	0.10 M	88.26 (0.01)	87.46 (0.07)
		Unfixed	9	0.10 M	88.10 (0.06)	87.33 (0.12)

*Reproduced from Table 4.1.

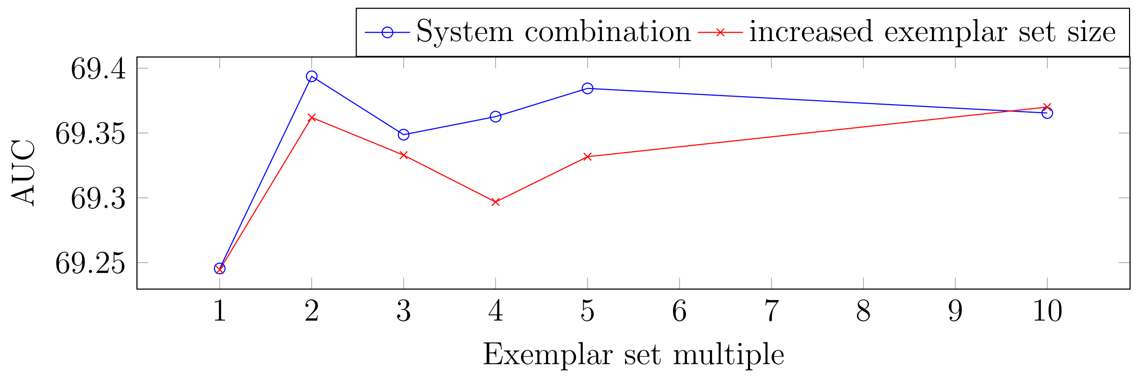
†Reproduced from Table 4.7.

exemplar set models give worse performance than the equivalent fixed exemplar set models ($p < 0.01$ in all cases).

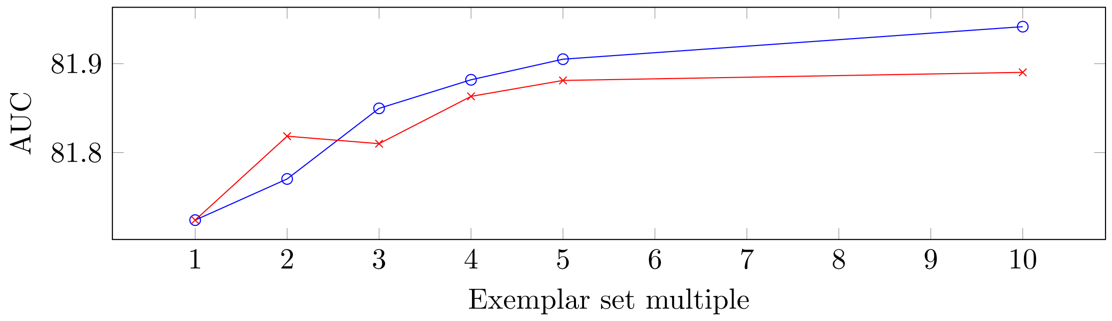
Figure 4.18 shows the performance of the GoEmotions ensemble models and models with increased exemplar set sizes. Although the performance tends to increase with an increased exemplar set multiple, none of the models match the performance of the equivalent fixed exemplar set model.

The results for the Minerva-RP models trained using unfixed exemplars on the CPC2 dataset are given in Table 4.11.

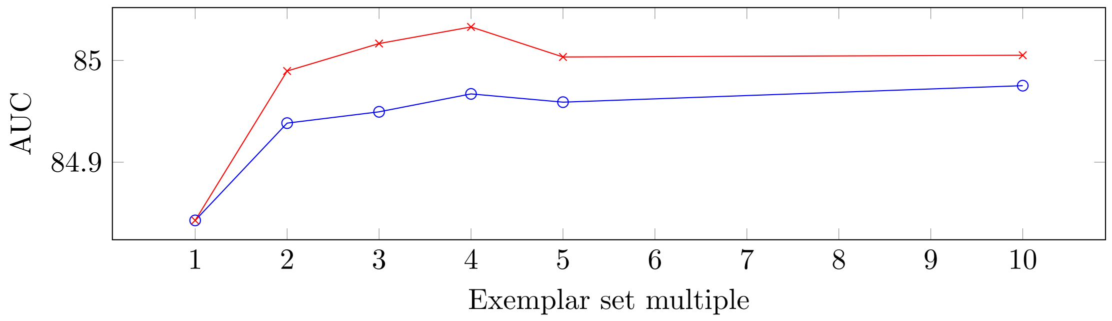
Unlike the TIMIT and GoEmotions tasks, there is no statistically significant difference in performance between fixed and unfixed exemplar sets ($p = 0.28$, $p = 0.85$ and $p = 0.86$ for the log spectrogram, XLSR and Whisper features respectively). This may tie in to the innate variability of the correctness labels: two different listeners can easily have different correctness scores for the same utterance. This 'noise' on the labels may mask the differences between the fixed and unfixed exemplar sets. Another possible reason is that 128 exemplars is sufficient to meaningfully represent the data; in §4.4, the performance of the model by exemplar set size began to plateau at 128



a. LSA features



b. Word2vec features



C. SBERT features

Figure 4.18: *Performance of Minerva-RPC models on the GoEmotions test set with either: system combination of models with different exemplar sets; or increased exemplar set size.*

Table 4.10: *Frame-based phone classification on GoEmotions using Minerva-RP and Minerva-RPC models, with fixed and unfixed exemplars. Standard deviations are given in parentheses.*

Features	**Model**	**Exemplars**	β	**Learned params**	**Accuracy (%) Dev**	**Test**
LSA	Minerva-RP	Fixed*	3	0.02 M	69.16 (0.06)	69.67 (0.12)
		Unfixed	3	0.02 M	68.30 (0.06)	68.73 (0.24)
	Minerva-RPC	Fixed†	3	0.02 M	69.22 (0.10)	69.85 (0.21)
		Unfixed	3	0.02 M	68.63 (0.07)	69.20 (0.14)
Word2vec	Minerva-RP	Fixed*	3	0.07 M	81.98 (0.07)	82.09 (0.05)
		Unfixed	3	0.07 M	81.22 (0.05)	81.36 (0.13)
	Minerva-RPC	Fixed†	3	0.07 M	82.30 (0.11)	82.27 (0.10)
		Unfixed	3	0.07 M	81.41 (0.03)	81.67 (0.06)
SBERT	Minerva-RP	Fixed*	3	0.05 M	85.00 (0.04))	85.27 (0.07)
		Unfixed	3	0.05 M	84.71 (0.02)	84.87 (0.10)
	Minerva-RPC	Fixed†	3	0.05 M	85.01 (0.13)	85.14 (0.11)
		Unfixed	3	0.05 M	84.71 (0.04)	84.86 (0.07)

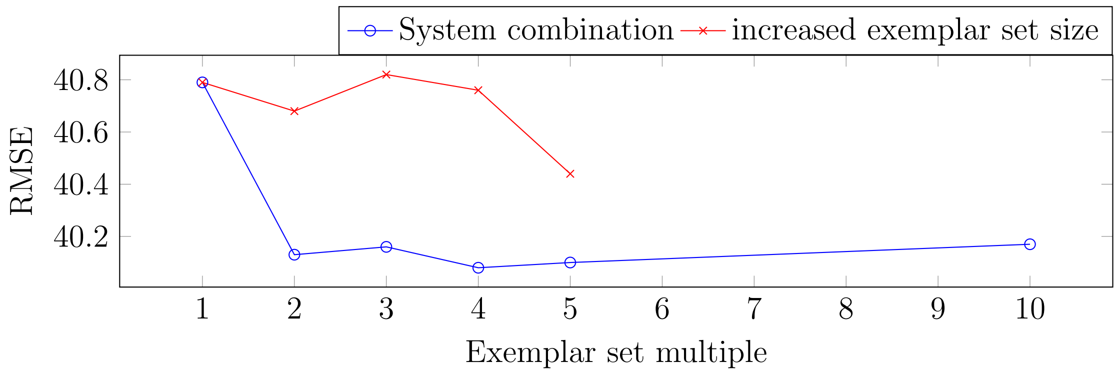
*Reproduced from Table 4.2.
†Reproduced from Table 4.8.

Table 4.11: *Speech intelligibility prediction on CPC2 using Minerva-RP models, with fixed, unfixed and adaptive exemplars. Standard deviations are given in parentheses.*

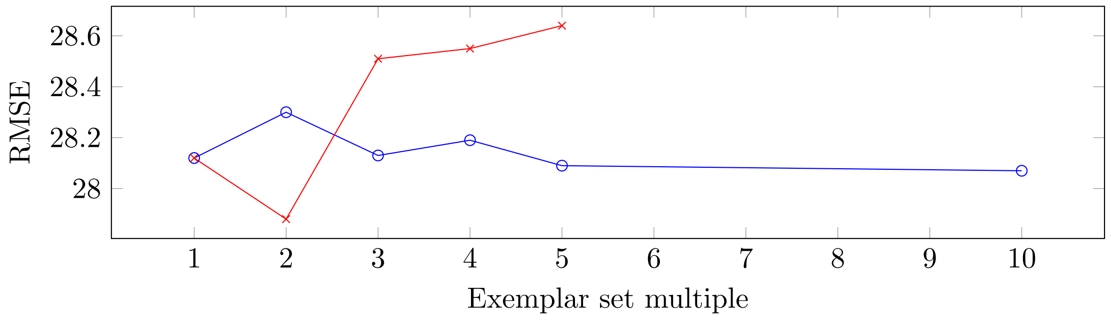
Features	**Exemplars**	β	**Learned params**	**RMSE Dev**	**Test**
Log spec	Fixed*	1	0.02 M	29.95 (0.09)	40.13 (0.81)
	Unfixed	5	0.02 M	29.99 (0.15)	40.59 (0.35)
XLSR	Fixed*	1	0.07 M	24.19 (0.08)	27.82 (0.28)
	Unfixed	1	0.07 M	23.43 (0.26)	27.87 (0.46)
Whisper	Fixed*	1	0.05 M	22.72 (0.04)	25.05 (0.56)
	Unfixed	3	0.05 M	23.04 (0.11)	25.10 (0.23)

*Reproduced from Table 4.3.

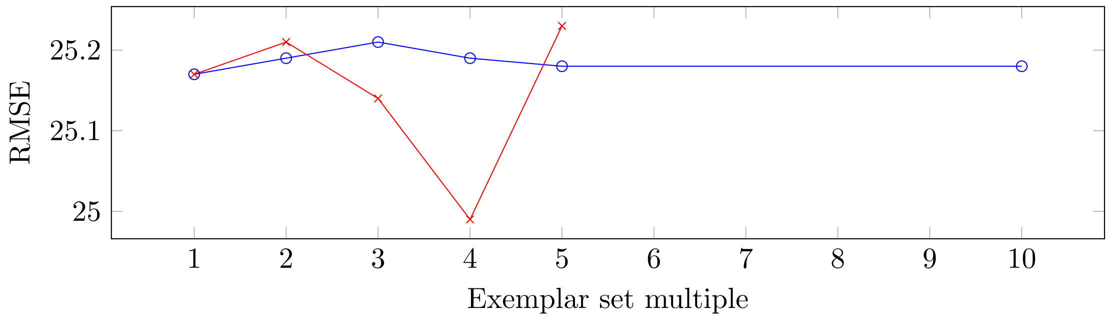
exemplars, with little further improvement in performance. This may also explain why the ensemble systems and larger exemplar sets, shown in Figure 4.19, give little benefit.



a. Log spectrogram features



b. XLSR features



C. Whisper decoder features

Figure 4.19: *Performance of Minerva-RP models on the CPC2 test set with either: system combination of models with different exemplar sets; or increased exemplar set size.*

4.10 Adaptive exemplars

Rather than selecting unfixed exemplar sets at random, more information may be made available to the model by selecting exemplars that are in some way relevant to the input under test. This will be referred to as an *adaptive exemplar set*, described in §3.4.2. In this case, the output is *conditioned* on the exemplar set; that is, the exemplar set is expected to tell us something useful about the input. In this respect, the adaptive exemplar models are similar to CNPs (see §2.6.1). The objective of this experiment is to determine whether performance can be improved by conditioning on exemplars based on metadata.

4.10.1 Methodology

Minerva-R, Minerva-RP and Minerva-RPC models using adaptive exemplars were trained on the TIMIT and CPC2 tasks. All models used separate transforms $\boldsymbol{W}_q \neq \boldsymbol{W}_k$ for the input and exemplars (see Equations 3.34 to 3.36). The GoEmotions task was excluded from this experiment because it has no metadata to use for adapting exemplars.

The TIMIT models used 384 exemplars stratified by class (14976 total), and a feature transformation dimension of 64. The exemplar adaptation was by gender: two separate exemplar sets were randomly selected for each minibatch, one consisting of frames from female speakers, and the other consisting of frames from male speakers. Exemplar sets were changed for every minibatch, whether during training or inference. For each minibatch, inputs from female speakers were classified using the female exemplar set, and inputs from male speakers were classified using the male exemplar set. All training minibatches contained both male and female speakers. All exemplars were randomly selected from the training data.

Models were trained on the CPC2 task using both *listener-adaptive* exemplars and *system-adaptive* exemplars. Each exemplar set had 16 exemplars; this is a reduced number of exemplars from previous experiments, which used 128 exemplars, because insufficient exemplars for each listener or system were available. During training, exemplar sets from each listener or system in the minibatch were constructed and used to predict the intelligibility of the relevant inputs. All minibatches used data from more than one listener or system. Since the CPC2 evaluation sets have no speaker or system overlap with their training sets, all exemplars at evaluation came from the evaluation sets. For this reason, the results reported for this experiment are not directly comparable with other work, since the model cheats by ‘peeking’ at the evaluation data during inference. The evaluation exemplars were matched to the relevant listener or speaker for each input. In all cases, a new exemplar set was randomly selected for each new minibatch. The feature transformation dimension was 32. For comparison, models with equivalent architecture were trained with both fixed and random (non-adaptive) exemplar sets with 16 exemplars, since models from previous experiments used more exemplars than this, and are therefore not directly comparable.

All models used the same scaling/normalisation as the equivalent Minerva-RP models (see §4.5.1). All hyperparameters were tuned as described in §4.3, and final hyperparameter values are given in Appendix A in Tables 4.12, 4.13 for the TIMIT models and A.15 for the CPC2 models.

4.10.2 Results and discussion

Table 4.12 gives the results for the Minerva-RP models trained on TIMIT. For the Mel spectrogram features, using gender-adaptive exemplars improves performance over the non-adaptive unfixed exemplar model ($p < 0.01$), but still gives lower performance than the non-adaptive fixed exemplar set. It appears that, in this case, the benefit of adaptation is not sufficient to make up for the 'noise' added by randomly changing the exemplar set.

Table 4.12: *Frame-based phone classification on TIMIT using Minerva-RP models, with fixed, unfixed and adaptive exemplars. Standard deviations are given in parentheses.*

Features	**Exemplars**	$\boldsymbol{\beta}$	**Learned params**	**Accuracy (%)** Dev	Test
Mel Spec	Fixed*	9	0.01 M	60.91 (0.17)	59.93 (0.24)
	Unfixed†	9	0.01 M	58.76 (0.22)	58.10 (0.23)
	Adaptive	9	0.01 M	60.11 (0.10)	59.29 (0.12)
Wav2vec	Fixed*	9	0.10 M	80.78 (0.21)	79.62 (0.20)
	Unfixed†	9	0.10 M	79.57 (0.06)	78.57 (0.13)
	Adaptive	9	0.10 M	79.29 (0.13)	78.27 (0.13)
HuBERT	Fixed*	7	0.10 M	88.20 (0.06)	87.35 (0.07)
	Unfixed†	7	0.10 M	88.01 (0.04)	87.27 (0.06)
	Adaptive	9	0.10 M	88.00 (0.07)	87.21 (0.08)

*Reproduced from Table 4.1.
†Reproduced from Table 4.9.

For the Wav2vec and HuBERT features, there is no statistically significant difference between the performance of models with unfixed and adaptive exemplar sets. It is possible that these sophisticated speech representations already include information on gender, making the adaptive exemplar sets superfluous. Both Wav2vec and HuBERT features have previously been used to good effect for gender classification, further supporting this hypothesis.

Table 4.13 shows the results for the Minerva-RPC models trained on TIMIT. There is no benefit to using adaptive exemplars for the Wav2vec and HuBERT features, supporting the hypothesis that these representations already meaningfully convey gender information. For the Mel spectrogram features, however, the adaptive exemplar set gives an improvement over both the unfixed exemplar set ($p < 0.01$), but still fails to match the performance of the fixed exemplar set model.

Table 4.13: *Frame-based phone classification on TIMIT using Minerva-RPC models, with fixed, unfixed and adaptive exemplars. Standard deviations are given in parentheses.*

Features	**Exemplars**	$\boldsymbol{\beta}$	**Learned params**	**Accuracy (%)**	
				Dev	**Test**
Mel Spec	Fixed*	7	0.01 M	62.95 (0.01)	61.94 (0.34)
	Unfixed†	9	0.01 M	61.27 (0.11)	60.29 (0.18)
	Adaptive	11	0.01 M	62.65 (0.08)	61.54 (0.05)
Wav2vec	Fixed*	7	0.10 M	81.19 (0.05)	79.96 (0.12)
	Unfixed†	7	0.10 M	80.06 (0.12)	78.97 (0.09)
	Adaptive	7	0.10 M	79.92 (0.10)	78.79 (0.15)
HuBERT	Fixed*	9	0.10 M	88.26 (0.01)	87.45 (0.07)
	Unfixed†	9	0.10 M	88.10 (0.06)	87.33 (0.12)
	Adaptive	7	0.10 M	88.11 (0.05)	87.27 (0.10)

*Reproduced from Table 4.7.
†Reproduced from Table 4.9.

Table 4.14 shows the results for the CPC2 task. Interestingly, both the listener- and system-adaptive log spectrogram models perform better than chance on the evaluation set; these are the only models that do so out of any experiment. Closer examination of the data reveals that the system-adaptive model is acting as a system-predictor, rather than an intelligibility predictor, however. These results are therefore less promising than they appear.

For the XLSR features, the adaptive models have worse performance on the evaluation set than the non-adaptive models. It is possible that this is due to poor scaling: if all the exemplars change their range of values, as may happen with different listeners providing the correctness scores, the learned scaling of the output may be inappropriate.

Using Whisper features, the listener adaptive system performs best out of all those reported. The difference is not statistically significant, however: $p = 0.42$ for the unfixed exemplar set, and $p = 0.62$ for the fixed exemplar set.

Table 4.14: *Speech intelligibility prediction on CPC2 using Minerva-RP models, with fixed, unfixed and adaptive exemplars. Standard deviations are given in parentheses.*

Features	**Exemplars**	β	**Learned params**	**RMSE Dev**	**RMSE Test**
Log spec	Fixed	1	0.02 M	30.44 (0.52)	40.85 (0.73)
	Unfixed	5	0.02 M	30.38 (0.08)	40.55 (0.66)
	Listener adapted	5	0.02 M	33.13 (0.44)	39.67 (0.80)
	System adapted	7	0.02 M	28.47 (0.05)	33.47 (0.99)
XLSR	Fixed	1	0.07 M	23.44 (0.20)	27.81 (0.47)
	Unfixed	1	0.07 M	24.48 (0.67)	29.31 (0.93)
	Listener adapted	1	0.07 M	23.15 (0.16)	30.82 (0.51)
	System adapted	3	0.07 M	25.81 (0.59)	33.67 (1.86)
Whisper	Fixed	1	0.05 M	22.82 (0.14)	24.89 (0.28)
	Unfixed	3	0.05 M	22.75 (0.14)	25.03 (0.20)
	Listener adapted	1	0.05 M	22.55 (0.18)	24.66 (0.93)
	System adapted	3	0.05 M	23.73 (0.16)	26.14 (0.39)

Overall, it appears that meaningful adaptation using exemplar sets is not as simple as training randomly changing models. There is information to be gained, however, especially for the CPC2 listener adaptation, where the listener information is poorly represented by the provided audiogram, and numerous utterances labelled by each listener are available. A more intelligent method of training the model might yield better results, perhaps by incorporating meta-learning, or by learning a non-linear scaling mechanism, rather than scaling with a single neuron. This is certainly an avenue for further research.

4.11 Sequence Minerva

The objective of this experiment is to perform preliminary tests on the Minerva-RPES model for sequence-to-sequence tasks by comparing it to equivalent non-sequence Minerva models and a baseline FFNN. The quantity of context that is useful is also explored.

4.11.1 Methodology

Minerva-RPES models, as described in §3.5.5, were trained on TIMIT with Mel spectrogram, Wav2vec and HuBERT features. The models used a fixed exemplar set with 384 exemplars per class, for 14976 total, and a feature transform dimension of 64 for both the initial Minerva modules and the self-Minerva module. Since GoEmotions and CPC2 are not sequence-to-sequence tasks, only the TIMIT task was used for this experiment. Hyperparameter tuning was carried out as described in §4.3, and final hyperparameter values are given in Appendix A in Table A.16. The model was trained with cross entropy loss.

4.11.2 Results and discussion

Table 4.15 includes results for the sequence Minerva-RPES model, which performs equivalently to the Minerva-RPE models with Mel spectrogram and Wav2vec features, but performs better than either the Minerva-RPE model or the FFNN with the HuBERT features. In both cases, the improvement is statistically significant ($p < 0.01$ for both).

Table 4.15: *Frame-based phone classification on TIMIT using the sequence-based Minerva-RPES model. Standard deviations are given in parentheses. The result is in bold.*

Features	**Classifier**	β	**Learned params**	**Accuracy (%)**	
				Dev	**Test**
Mel Spec	Minerva-RPE†	7	0.59 M	68.31 (0.06)	66.98 (0.22)
	Minerva-RPES	5	0.61 M	68.29 (0.06)	66.95 (0.12)
	FFNN*	-	1.19 M	69.71 (0.05)	68.20 (0.08)
Wav2vec	Minerva-RPE†	5	0.68 M	82.11 (0.07)	80.74 (0.09)
	Minerva-RPES	5	0.78 M	82.18 (0.03)	80.84 (0.13)
	FFNN*	-	1.88 M	82.29 (0.05)	81.07 (0.12)
HuBERT	Minerva-RPE†	5	0.68 M	88.32 (0.03)	87.47 (0.04)
	Minerva-RPES	5	0.78 M	88.73 (0.11)	**87.88 (0.12)**
	FFNN*	-	1.88 M	88.39 (0.04)	87.66 (0.02)

*Reproduced from Table 4.1.
†Reproduced from Table 4.4.

Minerva-RPES uses the entire utterance in the sequence-based Minerva modules by default, but further experiments were performed to determine how much context is useful, shown in Figure 4.20.

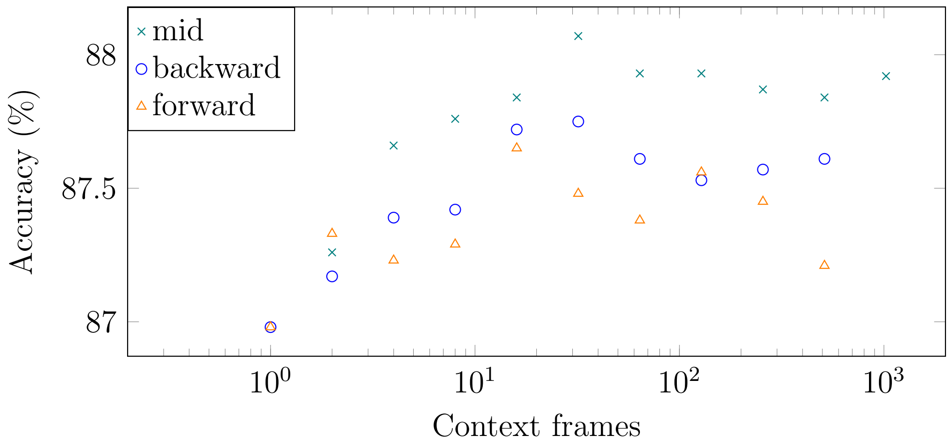


Figure 4.20: *Accuracy of Minerva-RPES on the TIMIT task with HuBERT features and increasing mid, backward and forward context.*

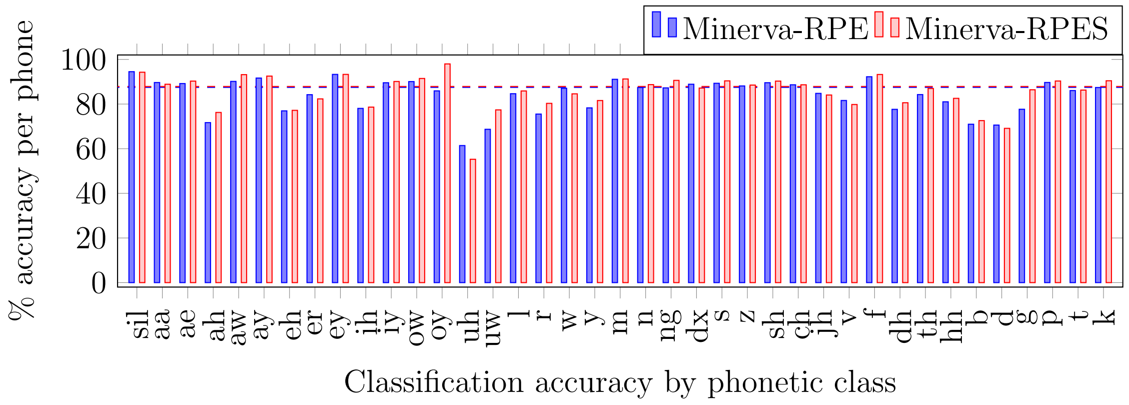


Figure 4.21: *Comparison of classification accuracy on the TIMIT test set by phonetic class for the Minerva-RPE and Minerva-RPES models using HuBERT features. The dashed lines are the overall model accuracies.*

Context is measured in terms of frames, which have a stride of 10 ms, and ranged from zero (effectively the same as Minerva-RPE) to 1024, which, since the TIMIT utterances are short, is sufficient to provide full context for all utterances. Results for mid, backward and forward context are given. All three types of context are useful, with mid-context being the most effective. Performance peaks at around 32 frames of context, following which it plateaus.

Figure 4.21 shows the performance by class of the Minerva-RPE and Minerva-RPES models using HuBERT features. There are substantial gains for the classes /oy/ and

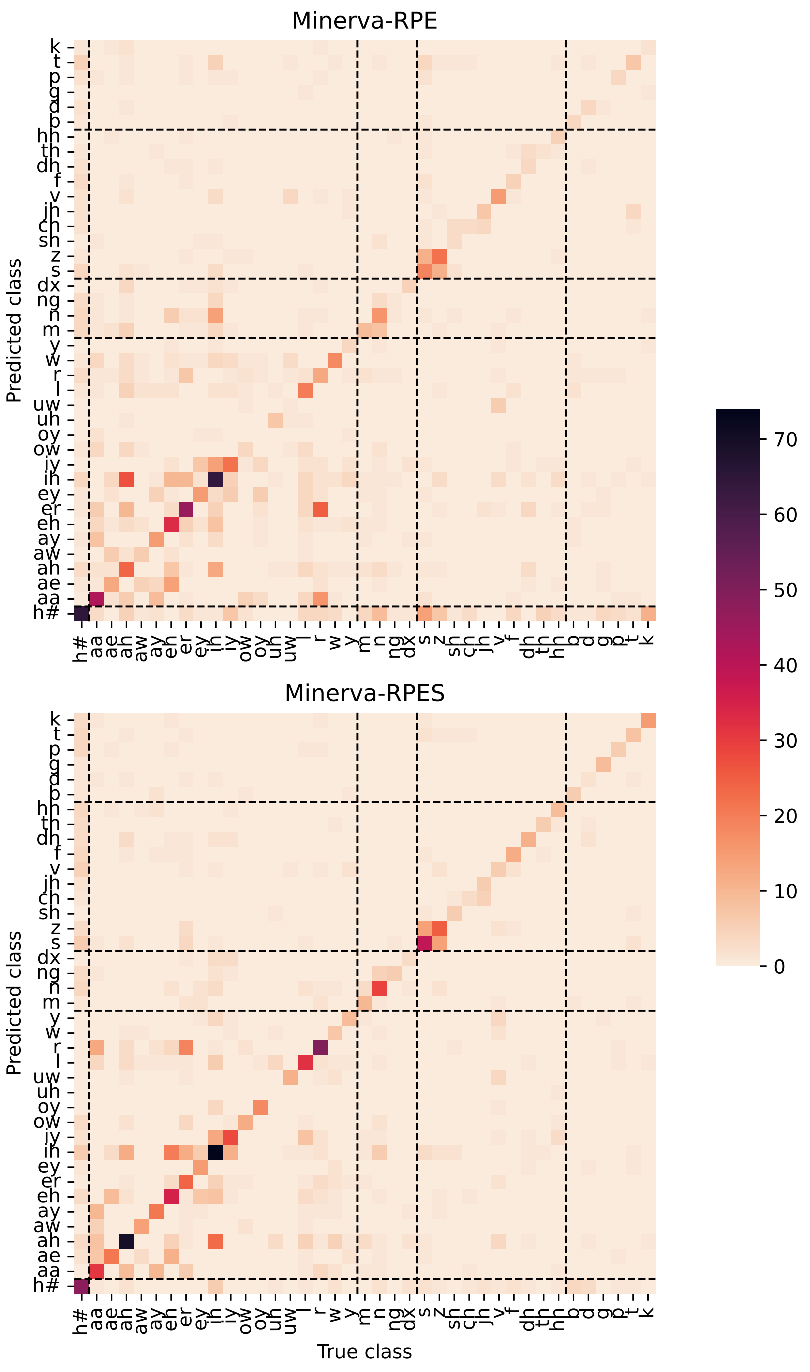


Figure 4.22: *Confusion matrices for the Minerva-RPE and Minerva-RPES models, showing only frames where the models disagreed.*

/uw/. Figure 4.22 shows the confusion matrices for the two models, but only for those frames where they disagreed. Improvements are not restricted to any particular type of sound: there are improvements to vowels, fricatives and plosives. Minerva-RPES is also less likely to incorrectly classify frames as silence.

4.12 Sequence Minerva and transformer

As was previously discussed in §3.5.5, Minerva-RPES is architecturally similar to transformers, with three main differences:

1. Transformers derive keys and values from the input, while Minerva-RPES uses exemplars as the keys and values;
2. Transformers use a softmax on the similarities between the keys and values to produce the activations, while Minerva-RPES uses the activation power; and
3. Transformers typically incorporate intermediate feed-forward layers and layer normalisation (Lei Ba et al. 2016) between self-attention modules.

The objective of this experiment was to compare models that mix-and-match each of these properties in order to explore the relationship between the architecture.

4.12.1 Methodology

Minerva-RPES and equivalent transformer encoders with two layers of self-attention were trained for the TIMIT frame-based phone classification task. The feature transformation dimension for the Minerva-RPES models was 64, as was the model dimension for the transformers. The transformers incorporated additional feed-forward layers and layer normalisation following each self-attention module.

The transformer encoder model takes a complete utterance as input. In order for the two models to be equivalent, the number of exemplars must therefore match the number of frames in the input. The utterance length in TIMIT is on average 152 frames. Using 4 exemplars per class results in an exemplar set of size 156, which is a close match.

In addition to the Minerva-RPES and transformer encoder models, models that mix-and-match the characteristics of both were trained, leading to a total of eight model architectures. Each model architecture was trained using Mel spectrogram and HuBERT features, using binary cross entropy loss. All models had hyperparameters tuned as described in §2.5.10, and final hyperparameter values are given in Table A.18 and A.17 for the Mel spectrogram and HuBERT models respectively.

4.12.2 Results and discussion

Table 4.16 shows the performance on the TIMIT test set of all eight model architectures, using both Mel spectrogram and HuBERT features. For both feature types, the Minerva-RPES model performs best, although the Mel spectrogram model benefits greatly from extra feed-forward layers and layer normalisation, whereas the HuBERT model does not.

It should be noted that the HuBERT features already incorporate context, so the modest gains using these features are to be expected. In contrast, the gains for the sequence models for the Mel spectrogram features are unsurprisingly much more impressive, since the spectrogram features offer less in the way of context.

These are preliminary results, and more work is required to put them into context, but the model performance is promising.

Table 4.16: *Performance of models on the Minerva-Transformer spectrum, using Mel spectrogram and Hubert features. Standard deviations are given in parentheses.*

Exemplars	**Activations**	**FFNN + layer norm**	**Learned params MelSpec/Hubert**	**Accuracy (%) MelSpec**	**Accuracy (%) Hubert**
Yes	Minerva	None	31 k / 203 k	65.66	**87.77**
		Both	34 k / 206 k	**69.58**	87.52
	scaled dot-product	None	31 k / 203 k	64.65	87.72
		Both	34 k / 206 k	68.10	87.38
No	Minerva	None	29 k / 158 k	53.50	86.03
		Both	35 k / 164 k	69.24	87.10
	scaled dot-product	None	29 k / 158 k	48.78	86.45
		Both	35 k / 164 k	67.29	86.35
FFNN*	-	No	13 k / 56 k	63.54	86.70

*The FFNN has the same model dimension as the sequence-based models, but is much less complex.

4.13 Concluding remarks

It has been shown that good feature representation is crucial for Minerva 2 and the models derived from it. As such, performance of the untrained Minerva-R model can be substantially improved by:

1. Using high-quality input features, such as HuBERT (Hsu et al. 2021) or Whisper-derived features (Radford et al. 2023) for speech; or BERT-based features for text
2. Incorporating a learned linear feature transformation on the input features

This combination results in the best performance, and also decreases the difference in performance between models. Given that feature representations for speech and text are likely to continue improving, simple models such as Minerva are likely to become more competitive.

Three options for the exemplar set have been trialled: fixed exemplars; randomly changing exemplars; and adaptive exemplars. Of these, the fixed exemplars appear to perform best, especially in conjunction with learned exemplar labels in the Minerva-RPE model. The adaptive exemplar option trialled here was found to be ineffective, but this option should not be ruled out. CNPs (Garnelo et al. 2018) condition on a exemplar set effectively, and might provide a template for using adaptive exemplars in a Minerva-based model.

Of the models that were proposed in Chapter 3 and tested here, the Minerva-RPES sequence model and the Minerva-RPE model show the most promise. For the sequence model the results are preliminary, and further experimentation is required, ideally including comparison with other architectures on a range of tasks. The Minerva-RPE model shows particular promise on the CPC2 task, and this result is explored further in the next chapter.

Chapter 5

Applications: Clarity Prediction Challenge 2

5.1 Introduction

5.1.1 Overview

The work reported in previous chapters has focused on exploring how Minerva is related to prototype models, and how it can be parameterised to improve its performance. In Chapter 4, it was found that the Minerva-RPE model, which uses a learned feature transform and learned exemplar labels, performed best in general. In this chapter, this finding is built on by using a Minerva-RPE model to demonstrate state-of-the-art performance on the CPC2 speech intelligibility prediction task. Part of this chapter covers a more sophisticated model, which was submitted to the Clarity Prediction Challenge 2, and which placed 2nd. This submitted model is compared with the much simpler models described in Chapter 4, which proved to be more effective, as well as simpler, smaller and faster. The performance of all these models is put into context with other submissions to the challenge.

The submission to CPC2 is reported in Mogridge et al. (2024). I was lead author, and the paper was co-authored by fellow PhD students George Close and Robert Sutherland, along with their supervisors.

Co-author contributions:

- George contributed initial code which had been used for the closely-related Clarity Prediction Challenge 1 (CPC1), implementing a LSTM model with various different feature representations, including log spectrogram and XLSR. He also contributed substantially to the introduction, conducted some of the analysis, and provided review of the finished paper.

- Robbie conducted some of the hyperparameter tuning, contributed to the introduction of the paper, and reviewed the completed paper.
- I adapted George's code from CPC1 to CPC2, added the use of Whisper encoder and decoder, implemented the Minerva models and conducted some of the hyperparameter tuning. I also conducted some of the analysis and wrote the majority of the paper.

All work on the simpler models reported in earlier chapters, and used for comparison here, was conducted by me.

5.1.2 Chapter content

This chapter begins with some background on the Clarity Prediction Challenge 2, followed by a summary of work conducted on the previous iteration of the challenge. Several models are described, including prototype and hybrid variants, followed by reporting and analysis of the results.

5.1.3 Background

Hearing loss is a growing problem throughout the world, and since it is often an age-related problem, and given that the world population is aging, incidence of hearing loss is expected to rise from around 466 million people in 2018, to around 630 million in 2030, and 900 million in 2050[1].

Hearing Aids (HAs) can improve quality of life for those with hearing loss, and machine learning algorithms show promise for improving the effectiveness of HAs (Doclo et al. 2015, Goetze et al. 2010). While assessment of HA algorithms by human listeners is essential, it is also time consuming and expensive. Automated speech intelligibility predictors therefore have a place in developing new and improved HA algorithms.

In this work, several models of varying complexity are proposed and tested to perform non-intrusive speech intelligibility prediction. All of the models make use of Whisper decoder layer features (Radford et al. 2023), previously described in §2.7.2.4. Both pure

[1]World Health Organisation 2018: https://apps.who.int/iris/bitstream/handle/10665/260336/9789241550260-eng.pdf

prototype and Minerva-based models are included, as well as system combinations of the two, which are found to be particularly effective.

5.2 Clarity Prediction Challenge 2

A description of the CPC2 dataset is given in §4.2.3, but more background and motivation is given here. The Clarity Project runs two challenges in sequence with the aim of improving HA technology: the *Clarity Enhancement Challenge* and the *Clarity Prediction Challenge*. The Clarity Enhancement Challenge (CEC) objective is to design HA systems to enhance noisy signals for hearing-impaired listeners. The Clarity Prediction Challenge (CPC) objective is to predict the intelligibility of the CEC systems.

The challenge has both a *non-intrusive* track, where only the enhanced noisy signal processed by the hearing aid can be used, and an *intrusive* track, where a clean version of the input audio can also be used. All of the models described here are non-intrusive, and although the CPC2 data provides additional information about the listener, all models described use as input only the enhanced noisy audio signal. This choice was made because the listener audiograms, which describe each listener's level of hearing loss at different frequency ranges, were found to be unhelpful in previous studies (Barker et al. 2022).

5.2.1 Challenge baseline

The baseline system provided by the challenge organisers (Barker et al. 2024) makes use of the Hearing-Aid Speech Perception Index (HASPI), version 2 (Kates & Arehart 2021). This is an intrusive system that makes use of both the enhanced noisy signal and the clean speech signal. A HASPI score is computed for both the left and right ear signals and logistic regression is used to predict the correctness scores from the higher HASPI score.

5.3 Prior approaches

Approaches used for the first version of the challenge, CPC1, varied widely. The winning non-intrusive submission used an ensemble of ASR models, and used disagreements between the ensemble's transcript predictions to model uncertainty about transcripts, with high uncertainty mapping to low intelligibility (Tu et al. 2022). Other approaches used pre-trained SSSRs features Zezario et al. (2022), Close et al. (2023, 2022), for direct prediction using neural networks.

CPC2 differs from CPC1 in that its evaluation sets are disjoint in terms of listener and hearing aid system relative to its training sets. This means that some of the better-performing approaches to CPC1, which operated effectively as predictors of the hearing aid system, were not at all useful in CPC2, which was discovered in early experiments for this work with the CPC2 data.

5.4 Experimental setup

5.4.1 Feature representation

The Whisper model used for feature extraction consists of both an encoder and a decoder. Whisper encoder features are frame-based, encoding 30 s of audio at a time. Audio with a shorter duration is padded, and audio with a longer duration is truncated. In the case of the CPC2 data, all audio clips are shorter than 30 s and are padded. The output of the Whisper encoder layers is $1500 \times 768 \times 12$, where 1500 is the 30 s time duration (Whisper encoder frames have a stride of 20 ms), the model dimension is 768 and 12 is the number of layers. The decoder features are label-synchronous, rather than frame synchronous, and have dimension $W \times 768 \times 12$, where W is the number of predicted tokens (typically around 10 to 20 for the CPC2 data), 768 is the model dimension, and 12 is the number of layers. Preliminary experiments suggested that the decoder features were more effective.

5.4.2 Model architecture

Bidirectional Long Short-Term Memory (BLSTM)-based architectures incorporating attention pooling, were used for the challenge submission, since they have been shown to be effective for speech quality prediction by Tamm et al. (2022), and further used for speech intelligibility prediction by Close et al. (2023). Both prototype and exemplar-informed versions of the BLSTM model are proposed, as well as a system combinations of the two. Figure 5.1 shows the architecture of the system combination of the BLSTM models.

5.4.3 Prototype BLSTM SI prediction model

The prototype BLSTM model is shown to the right in Figure 5.1. The model uses a learnable weighted sum of the Whisper decoder representations, implemented as a learnable linear layer with 12 parameters, all initialised to 0, followed by a softmax to ensure that the layer weights sum to 1. This representation of dimension $W \times 768$ is then

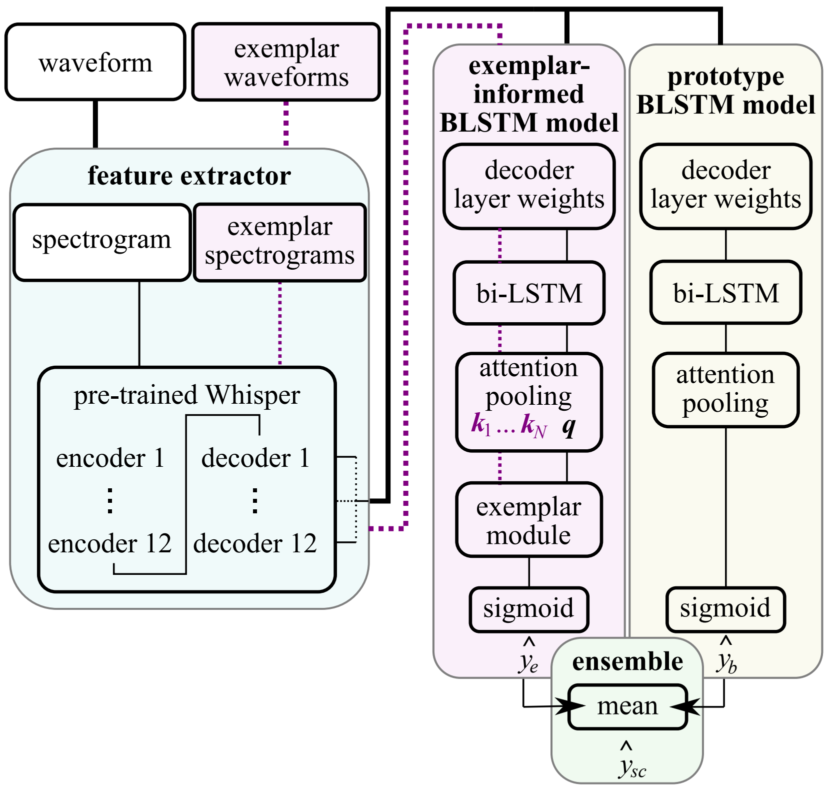


Figure 5.1: *BLSTM system combination model architecture for SI prediction.*

processed by 2 BLSTM layers with an input size of 768 and a hidden layer size of 384. Finally, an attention pooling feed-forward layer with sigmoid activation outputs to a single neuron which represents the primary predicted correctness value $\hat{y}_p$ normalized between 0 and 1. The prototype model has approximately 8.3 M parameters. The prototype model is trained for 25 epochs with batch size 8, learning rate 10^{-5} and weight decay 10^{-4}.

5.4.4 Exemplar-informed BLSTM SI prediction model

The exemplar-informed BLSTM model differs from the prototype BLSTM model in that the attention-pooling output feeds into an exemplar-informed Minerva-RP module, which feeds to a single neuron used for scaling the model output (see §3.5.2). The exemplar-informed module incorporates an exemplar set drawn from the training data. A new set of 16 random exemplars is selected for each training or inference minibatch.

Exemplars are processed in the same way as the input, feeding through the BLSTM and attention pooling, before being used in the exemplar module.

Let $\boldsymbol{q}$ be the output of the attention pooling for a given input (see Figure 5.1). The exemplars, $\boldsymbol{k}_1, ..., \boldsymbol{k}_N$, are processed in the same way as the input, with layer weights, BLSTM and attention pooling producing them from the exemplar inputs. The output, c, of the Minerva-RP module passes through a single linear neuron, and then through a sigmoid activation, which yields the exemplar-informed model's prediction, $\hat{y}_e$, normalised to fall between 0 and 1. The exemplar model has approximately 10 M parameters. The exemplar-informed model is trained for 50 epochs with learning rate 2×10^{-6} and weight decay 10^{-4}.

5.4.5 System combination

The prototype and exemplar-informed BLSTM models each output a prediction for the correctness of each input, and are trained separately with MSE loss. The output of the BLSTM system combination, $\hat{y}_{sc}$, for a given input signal is the mean of the outputs of the prototype $\hat{y}_p$ and exemplar-informed systems $\hat{y}_e$,

$$\hat{y}_{sc} = \frac{\hat{y}_e + \hat{y}_p}{2}. \tag{5.1}$$

5.4.6 Simplified models

The sophisticated BLSTM model described above is compared with simpler architectures from the previous chapter, which are reused here:

1. The FFNN described in §4.5.1, with results reported in Table 4.3, which is the best performing prototype model from the previous chapter;

2. The Minerva-RP model with unfixed exemplars, described in §3.5.2, with results reported in §4.14, which uses same number of exemplars and the same exemplar selection strategy as the exemplar-informed BLSTM model, but uses them directly, rather than incorporating a BLSTM and attention pooling; and

3. The Minerva-RPE model, described in §3.5.3 and §4.7.1, with results reported in Table 4.6, which is the best-performing exemplar-informed model from the previous chapter.

5.5 Results and discussion

Table 5.1 shows the results for the prototype BLSTM model and the exemplar-informed BLSTM model, as well as replicating the results for the simplified models from Chapter 4. The CPC2 challenge baseline model's overall performance is also shown (detailed results for this model are not available).

The validation sets contain enhancement systems and listeners seen in the corresponding training training data. The evaluation sets are disjoint, containing only unseen enhancement systems and listeners. All of the models reported here beat the baseline system by a substantial margin, despite being non-intrusive systems, while the baseline is intrusive.

Table 5.1: *Model performance on the validation and evaluation splits. The best result is in bold.*

Model	**RMSE**							
	validation split				**evaluation split**			
	1	2	3	**all**	1	2	3	**all**
CPC2 baseline								28.70
BLSTM prototype	21.6	23.5	22.8	22.66	28.5	23.9	23.3	25.26
BLSTM exemplar	21.7	23.5	22.7	22.65	29.1	24.5	23.4	25.75
BLSTM ensemble	21.6	23.4	22.7	22.54	28.6	23.9	23.2	25.27
FFNN*	22.2	23.1	22.8	22.68	26.6	24.7	22.1	24.47
Minerva-RPE fixed†	22.2	23.0	23.1	22.76	26.7	24.5	21.9	24.36
Minerva-RP unfixed**	22.0	23.3	22.9	22.75	27.6	25.3	22.2	25.03
FFNN/Minerva-RPE ensemble	22.1	22.9	22.8	22.57	26.5	24.4	21.8	**24.21**

*Reproduced from Table 4.3
†Reproduced from Table 4.6
**Reproduced from Table 4.14

All of the simplified models (FFNN, Minerva-RPE, Minerva-RP unfixed) outperform the BLSTM models by a substantial margin ($p < 0.001$ in all cases) on the evaluation sets, despite the BLSTM models showing good performance on previous speech quality and intelligibility studies. The differences between the BLSTM models and simplified models on the non-disjoint validation sets is not statistically significant ($p > 0.1$ in all cases). It appears that the simpler models generalise better to unseen listeners and systems.

5.5.1 Prototype and exemplar-informed models

The prototype and exemplar-informed BLSTM models show similar performance on the validation sets, with the ensemble of the two outperforming either, and in fact giving the best performance out of any of the models on the validation sets. This is not replicated on the evaluation set, where the primary model outperforms the secondary model, and performs equivalently to the ensemble.

As previously seen, the FFNN and Minerva-RPE models show similar performance to each other on both validation and evaluation sets. An ensemble of the two performs better than either individual model for all splits on both the validation and evaluation sets.

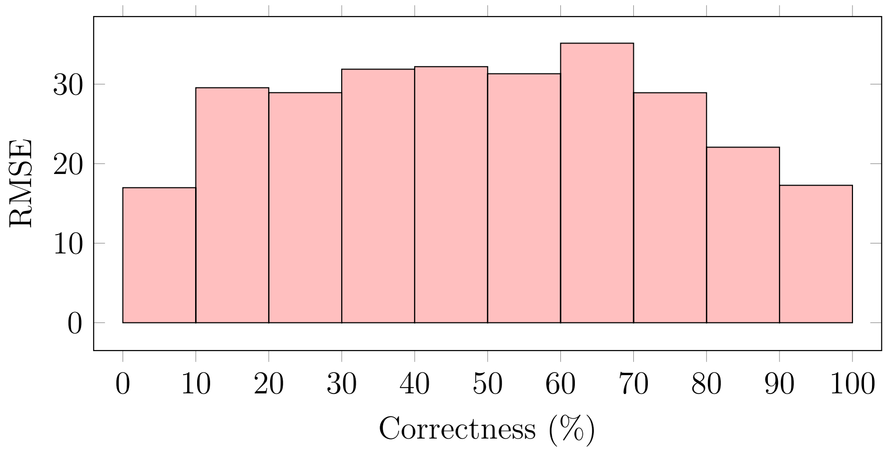
5.5.2 Performance by intelligibility

Figure 5.2a. shows the performance of the BLSTM ensemble model for different correctness value ranges, and Figure 5.2b. shows the performance of the FFNN/Minerva-RPE ensemble model for different correctness value ranges. The BLSTM ensemble model performs well for very low intelligibility (0 correctness) and for very high intelligibility (100 correctness), but performs poorly between the two extremes. This corresponds with the distribution of true correctness scores in the training data (see Figure 4.3 in §4.2.3), in which 0 and 100 correctness are over-represented. In contrast, the FFNN/Minerva-RPE ensemble model performs fairly similarly for different correctness ranges, but trending towards lower prediction error for higher intelligibility.

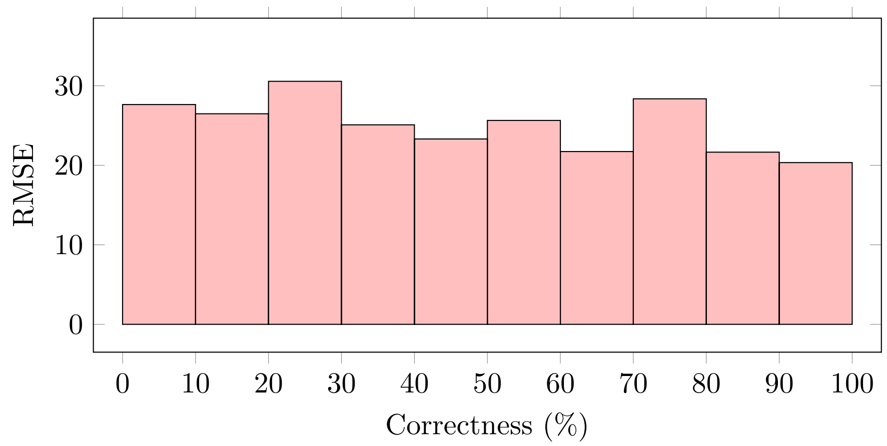
The differing performance between the two ensemble models suggest that combining all models in a 4-way ensemble, incorporating the prototype BLSTM, exemplar BLSTM, FFNN and Minerva-RPE models, might be beneficial. This combination gives a slight improvement on the validation sets over the individual and 2-way ensemble models, with a validation RMSE of 22.3% correctness. There is no improvement over the FFNN/Mineva-RPE ensemble on the evaluation sets, however, with evaluation RMSE of 24.3% correctness. Nevertheless, this may be a useful avenue for further research.

5.5.3 Model performance on unseen enhancement systems

All the models show lower performance on Evaluation Split 1. This appears to stem from the presence of audio enhanced by enhancement system E001 (the baseline in the Clarity Enhancement Challenge 2), which is present in this evaluation set. Audio enhanced by this system has an average correctness value of 28.7%, which is significantly lower than the average of the other two enhancement systems in Evaluation Set 1, E022



a. BLSTM ensemble model



b. FFNN/Minerva-RPE ensemble model.

Figure 5.2: *Model performance by true correctness, for the BLSTM and FFNN/Minerva-RPE ensemble models.*

and E031, which have average correctness values of 73.0% and 84.3%, respectively.

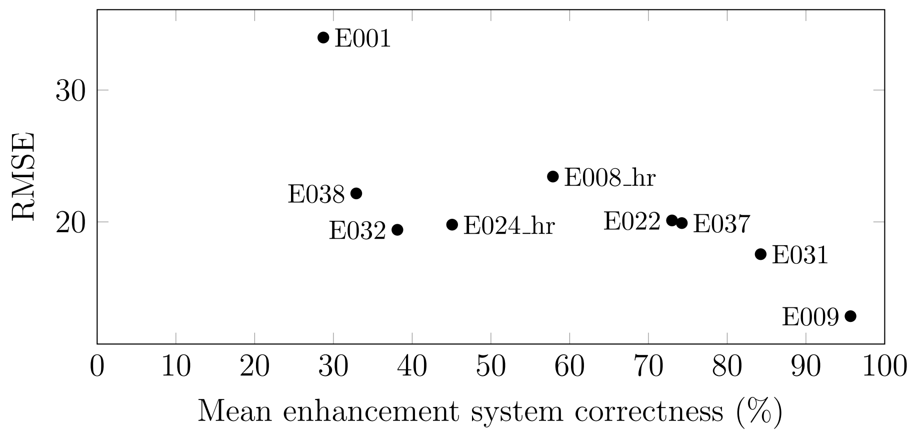


Figure 5.3: *Model performance by mean hearing aid system correctness.*

Although the models can generalise to unseen enhancement systems to some extent, they appear to predict correctness less accurately on some system than others.

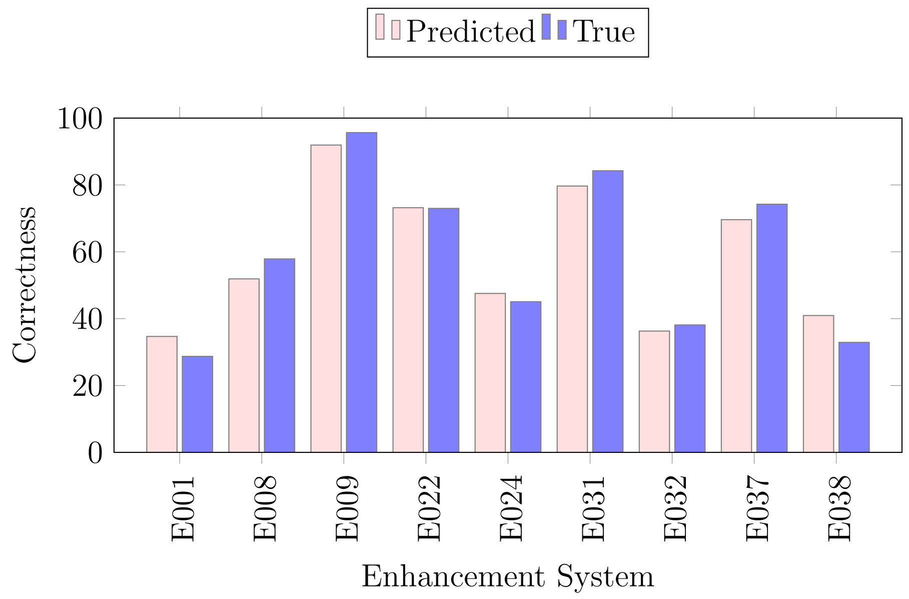


Figure 5.4: *Performance of proposed prediction system depending on enhancement system*

Figure 5.3 shows the FFNN/Minerva-RPE ensemble model's performance by enhancement system correctness across all enhancement systems. The model is very good at predicting the correctness for enhancement system E009, but poor at predicting the correctness for enhancement system E001. It is interesting that E001 is the worst-performing enhancement system and E009 is the best-performing enhancement system. This suggests that better-performing enhancement systems, which produce outputs with high correctness scores, may be easier to predict. Figure 5.4 shows the predicted (left) and true (right) correctness across all enhancement systems, showing that the FFNN/Minerva-RPE ensemble model overestimates the correctness of the two worst-performing enhancement systems, E001 and E038, while generally slightly underestimating the correctness of the better-performing enhancement systems.

5.5.4 Whisper layer weights

The learned weights for the Whisper decoder layers from the primary BLSTM model are shown in Figure 5.5. These show how the model uses the information to weight each decoder layer feature, and therefore the higher the value the more useful the model finds the layer to be. The weights are shown for each of the three models trained on the different training splits.

The general pattern for the different training splits is similar, with layers 7 and 8 having the highest weights across all splits. This suggests that layers 7 and 8 contain the most relevant information for intelligibility. Interestingly, the model trained on Split 3 learns weights that emphasise layers 7 and 8 more strongly. This information was used to select layer 8 as the input features to the simplified models used in the previous chapter, including the FFNN and Minerva-RPE models examined here.

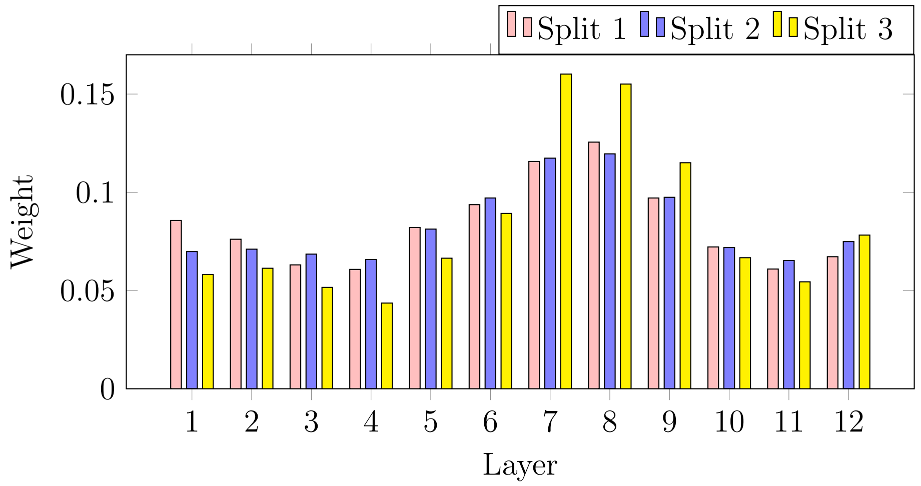


Figure 5.5: *Learned weights for the primary model Whisper decoder layers.*

5.5.5 Performance comparison

Figure 5.6 shows the performance of the FFNN/Minerva-RPE model, which is the best performing model reported in this thesis, compared to all other challenge entries, as well as the challenge HASPI baseline. System P002 is the BLSTM ensemble model, which was submitted to the challenge and placed second. *Prior* is a system which always predicts the average of the intelligibility over the training set, regardless of the input. P011 was the winning submission to the challenge (Cuervo & Marxer 2024). The FFNN/Minerva-RPE ensemble model outperforms all other models submitted to the challenge by a substantial margin; the winner of challenge, system P011, achieved a RMSE of 25.1%.

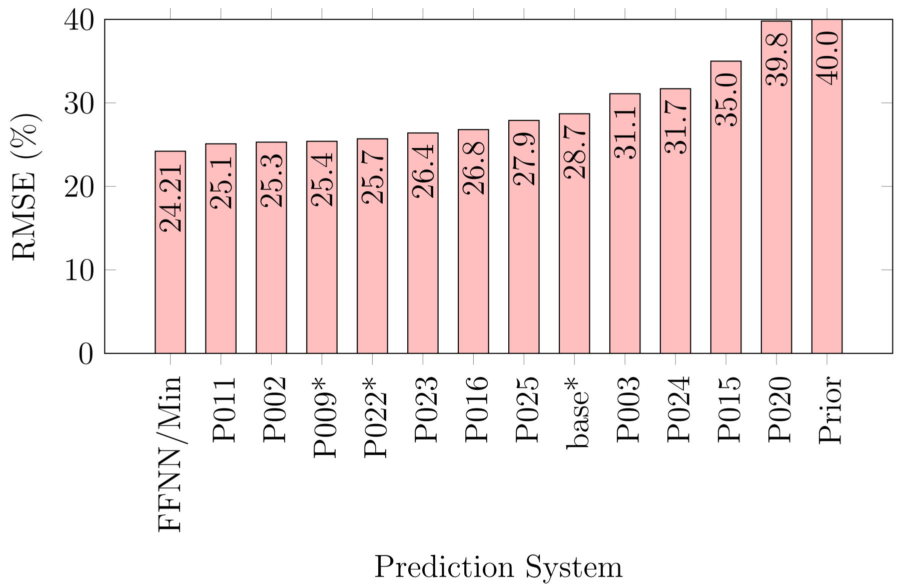


Figure 5.6: *Comparison of performance of CPC2 entries. A * denotes an intrusive system.*

5.5.6 Concluding remarks

The Whisper decoder layers are a useful feature representation for speech intelligibility prediction, with layers 7 and 8 appearing to be the most relevant. Several other challenge submissions also used Whisper features, including the challenge winner (Cuervo & Marxer 2024), demonstrating that Whisper's intermediate layers, both encoder and decoder, are effective feature representations for speech intelligibility prediction.

A non-intrusive ensemble combining a simple FFNN model with a Minerva-RPE model achieves state-of-the-art performance on the CPC2 task, outperforming the HASPI regression baseline and all challenge submissions by a substantial margin, despite several of these systems being intrusive systems that have access to the reference signal. Simple models with good feature representation appear to work best, providing better generalisation to unseen listeners and enhancement systems compared to models that use lower quality features or more sophisticated architecture.

Chapter 6

Conclusions and further work

6.1 Summary of contributions

In this thesis, the use of exemplars has been investigated using the Minerva 2 exemplar-based model from the field of human psychology. In Chapter 1, three research questions were introduced, which are now revisited.

RQ1 What are the similarities and differences between the exemplar model Minerva 2 and artificial neural networks?

In §3.2, it was shown that, rather than being separate from artificial neural networks, Minerva 2 is in fact itself a 2-layer feed-forward neural network, but with parameters taken directly from data, rather than being learned through backpropagation.

Further, the iterative inference process, proposed by Minerva 2's creator and referred to here as the echo-of-echoes process, has been shown to be a fixed point problem. It is also an infinite-depth artificial neural network with tied parameters, and as such, is a form of Deep Equilibrium Model (DEM) (Bai et al. 2019). This means that previous findings relating to DEMs can be applied to the echo-of-echoes process, including the use of well-understood and efficient algorithms to find the fixed point solutions (Hoffman & Frankel 2018). There is also potential to train the echo-of-echoes process in the same way as a DEM, offering up a further avenue of research.

RQ2 What exemplars should be stored in memory, and how should they be represented?

It was shown empirically in §4.4 that as exemplar set size increases, so does performance, albeit with diminishing returns. The exemplar set size is analogous to the model dimension of an artificial neural network, since the exemplar features form the parameters of the first layer.

In §4.6, it was shown that good feature representation is crucial for Minerva 2 and the models derived from it. Two options for improving the feature representation were demonstrated: firstly, by using a learned linear transformation to perform dimension reduction of the input features; and secondly, by using sophisticated pre-trained features for the input. The first of these options both improves model performance, and also reduces the computational complexity. The two options work particularly well in tandem. Using high quality feature representation improves the performance of all models, but also reduced the *difference* in performance between the simple, untrained Minerva-R model and a trained feed-forward neural network. Feature representations for both speech and text are constantly improving, with rapid gains having been made in recent years. Numerous self-supervised speech representations such as Wav2vec2.0 (Baevski et al. 2020), XLSR (Conneau et al. 2020), HuBERT (Hsu et al. 2021) and (Chen, Wang, Chen, Wu, Liu, Chen, Li, Kanda, Yoshioka, Xiao, Wu, Zhou, Ren, Qian, Qian, Wu, Zeng, Yu & Wei 2022) have been proposed in the past few years. Text representations have also improved substantially, with linear representations based on word-document frequency, such as LSA, replaced first with more sophisticated but non-context based representations derived from deep learning, such as Word2vec Mikolov, Chen, Corrado & Dean (2013), and then with contextual representations from BERT-based models (Devlin et al. 2019, Song et al. 2020, Reimers & Gurevych 2019). Continuing progress on both speech and text representations seems likely. If so, the use of simple, interpretable models such as Minerva and its derivatives may become increasingly competitive.

RQ3 Can an exemplar model be combined with parameters to form a hybrid exemplar-prototype model? What benefits, if any, does this bring?

In §3.5 a family of Minerva-based models has been proposed with an increasing degree of parameterisation, and in Chapter 4 these models have been compared against each other other and against a baseline feed-forward neural network on three separate speech and language tasks. Preliminary results on a sequence-based version of Minerva show particular promise. The sequence-Minerva model is closely related to the transformer architecture (Vaswani et al. 2017), a connection that could potentially be exploited for further performance gains. Numerous different options are available for transformers, including options for positional encoding (Vaswani et al. 2017, Su et al. 2024), the addition of convolutional layers (Gulati et al. 2020), to name just two. Given the similarities between the sequence-Minerva model and transformers, many options that apply to transformers might be viable for Minerva-derived models as well.

For non-sequence tasks, the Minerva-RPE model, which incorporates a learned linear transformation and learned exemplar classes into the Minerva model, showed the most promise. In Chapter 5, a system that includes Minerva-RPE as a key component was described, which provides state-of-the-art performance on the Clarity Prediction Challenge 2 task Barker et al. (2024). The system consists of a parameterised Minerva-RPE model used in combination with a simple feed-forward classifier. The resulting system provides the best performance seen on the CPC2 dataset, outperforming the challenge winner by a substantial margin (Barker et al. 2024, Cuervo & Marxer 2024), despite being a much smaller model. This is further evidence that simpler models are competitive when used in conjunction with high-quality features.

6.2 Further work

6.2.1 Minerva initialisation

The untrained Minerva-R model is a feed-forward neural network with parameters set directly from data, rather than being trained. Furthermore, its parameterised variant, Minerva-RP, which incorporates a learned linear transform of the input features, also shows high inductive bias, performing substantially better than chance at two classification tasks prior to training. It might be possible to benefit from both Minerva's high inductive bias and the flexibility and power of training an artificial neural network by backpropagation by treating Minerva as an initialisation of a feed-forward neural network. This may prove an useful avenue for further research, particularly in situations where there is limited data.

Since Minerva does not include innate scaling, this may be an area to focus on. The output of the first layer can, depending on feature representation and exemplar set size, result in an output orders of magnitude larger than its input. It may be possible to account for this by a linear scaling of the layers, either mathematically, or more practically, by using an initial minibatch of data to determine the likely gain and account for it.

6.2.2 Echo-of-echoes and deep equilibrium models

Since the echo-of-echoes process is a form of deep equilibrium model, as shown in §3.3, it would be interesting to to test its effectiveness on real tasks. In particular, testing under what circumstances the exemplar classes are also fixed point outputs of the model, either theoretically or experimentally, would answer the question of whether the echo-of-echoes process in it original form is fit for purpose.

6.2.3 Sequence Minerva

The sequence Minerva-RPES model shows promise on the preliminary experiments on a TIMIT frame-based phone classification task, as shown in §4.11. These initial results are insufficient to demonstrate effectiveness, however, and further work comparing this model with transformers and other architectures on additional sequence tasks would be a logical next step.

6.2.4 Adaptative exemplars

In §4.10, an option for using adaptive exemplars was tested, which was not successful. It appears that meaningful adaptation using exemplar sets is not as simple as matching exemplars to the inputs during training. There may be more effective ways of handling the training, however, so that the model learns a tool for using the exemplars, rather than using them directly as parameters. Conditional Neural Processes (Garnelo et al. 2018), for example, have been shown to condition effectively on an exemplar set, and might provide a guide for more effective adaptive exemplars. A more intelligent method of training the model might also yield better results, perhaps by incorporating meta-learning, or developing loss functions specifically to target exemplar effectiveness.

Bibliography

Agarwala, A., Pennington, J., Dauphin, Y. & Schoenholz, S. (2020), 'Temperature check: theory and practice for training models with softmax-cross-entropy losses', *arXiv preprint arXiv:2010.07344* .

Alberti, M., Seuret, M., Pondenkandath, V., Ingold, R. & Liwicki, M. (2017), Historical document image segmentation with lda-initialized deep neural networks, *in* 'Proceedings of the 4th international workshop on historical document imaging and processing', pp. 95–100.

Allen, J. (1977), 'Short term spectral analysis, synthesis, and modification by discrete fourier transform', *IEEE Transactions on Acoustics, Speech, and Signal Processing* **25**(3), 235–238.

Andersen, A. H., Schoenmaker, E. & van de Par, S. (2016), Speech intelligibility prediction as a classification problem, *in* '2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)', pp. 1–6.

Arndt, J. & Hirshman, E. (1998), 'True and false recognition in minerva2: Explanations from a global matching perspective', *Journal of Memory and Language* **39**(3), 371–391.

Arya, S. & Mount, D. M. (1993), Approximate nearest neighbor queries in fixed dimensions, *in* 'SODA 1993: Proceedings of the fourth annual ACM-SIAM symposium on discrete algorithms', pp. 271–280.

Baas, M., van Niekerk, B. & Kamper, H. (2023), Voice conversion with just nearest neighbors, *in* 'Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH 2023', pp. 2053–2057.

Baevski, A., Zhou, Y., Mohamed, A. & Auli, M. (2020), wav2vec 2.0: A framework for self-supervised learning of speech representations, *in* 'Advances in Neural Information Processing Systems', Vol. 33, pp. 12449–12460.

Bai, S., Kolter, J. Z. & Koltun, V. (2019), 'Deep equilibrium models', *Advances in neural information processing systems* **32**.

Barker, J., Akeroyd, M., Bailey, W., Cox, T. J., Culling, J. F., Firth, J., Graetzer, S., & Naylor, G. (2024), The 2nd clarity prediction challenge: A machine learning challenge for hearing aid intelligibility prediction, *in* '2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)', pp. 11551–11555.

Barker, J., Akeroyd, M., Cox, T. J., Culling, J. F., Firth, J., Graetzer, S., Griffiths, H., Harris, L., Naylor, G., Podwinska, Z., Porter, E. & Munoz, R. V. (2022), The 1st Clarity Prediction Challenge: A machine learning challenge for hearing aid intelligibility prediction, *in* 'Proc. Interspeech', pp. 3508–3512.

Bowman, C. R. & Zeithamova, D. (2018), 'Abstract memory representations in the ventromedial prefrontal cortex and hippocampus support concept generalization', *Journal of Neural Science* **38**(10), 2605–2614.

Chang, X., Maekaku, T., Guo, P., Shi, J., Lu, Y.-J., Subramanian, A. S., Wang, T., Yang, S.-w., Tsao, Y., Lee, H.-y. & Watanabe, S. (2021), An exploration of self-supervised pretrained representations for end-to-end speech recognition, *in* '2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)', pp. 228–235.

Charpentier, F. & Stella, M. (1986), Diphone synthesis using an overlap-add technique for speech waveforms concatenation, *in* '1986 IEEE International Conference on Acoustics, Speech, and Signal Processing conference proceedings', Vol. 11, IEEE, pp. 2015–2018.

Chechik, G., Sharma, V., Shalit, U. & Bengio, S. (2010), 'Large scale online learning of image similarity through ranking.', *Journal of Machine Learning Research* **11**(3).

Chen, L.-W. & Rudnicky, A. (2023), Exploring wav2vec 2.0 fine tuning for improved speech emotion recognition, *in* 'ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)', IEEE, pp. 1–5.

Chen, S., Wang, C., Chen, Z., Wu, Y., Liu, S., Chen, Z., Li, J., Kanda, N., Yoshioka, T., Xiao, X., Wu, J., Zhou, L., Ren, S., Qian, Y., Qian, Y., Wu, J., Zeng, M., Yu, X. & Wei, F. (2022), 'Wavlm: Large-scale self-supervised pre-training for full stack speech processing', *IEEE Journal of Selected Topics in Signal Processing* **16**(6), 1505–1518.

Chen, Z., Chen, S., Wu, Y., Qian, Y., Wang, C., Liu, S., Qian, Y. & Zeng, M. (2022), Large-scale self-supervised speech representation learning for automatic speaker verification, *in* 'ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)', pp. 6147–6151.

Cherry, E. C. (1953), 'Some experiments on the recognition of speech, with one and with two ears', *The Journal of the acoustical society of America* **25**(5), 975–979.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H. & Bengio, Y. (2014), Learning phrase representations using RNN encoder–decoder for statistical machine translation, *in* 'Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)', pp. 1724–1734.

Clark, S. E. (1992), 'Word frequency effects in associative and item recognition', *Memory & Cognition* **20**, 231–243.

Clark, S. E. (1995), 'The generation effect and the modeling of associations in memory', *Memory & Cognition* **23**(4), 442–455.

Close, G., Hain, T. & Goetze, S. (2023), Non Intrusive Intelligibility Predictor for Hearing Impaired Individuals using Self Supervised Speech Representations, *in* 'Proc. Workshop on Speech Foundation Models and their Performance Benchmarks (SPARKS), ASRU sattelite workshop', Taipei, Taiwan.

Close, G., Hollands, S., Hain, T. & Goetze, S. (2022), Non-intrusive Speech Intelligibility Metric Prediction for Hearing Impaired Individuals, *in* 'Proc. Interspeech', pp. 3483–3487.

Conneau, A., Baevski, A., Collobert, R., Mohamed, A. & Auli, M. (2020), 'Unsupervised cross-lingual representation learning for speech recognition', *arXiv preprint arXiv:2006.13979* .

Cover, T. & Hart, P. (1967), 'Nearest neighbor pattern classification', *IEEE Transactions on Information Theory* **13**(1), 21–27.

Cuervo, S. & Marxer, R. (2024), Speech foundation models on intelligibility prediction for hearing-impaired listeners, *in* 'ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)', pp. 1421–1425.

Deese, J. (1959), 'On the prediction of occurrence of particular verbal intrusions in immediate recall.', *Journal of experimental psychology* **58**(1), 17.

Dehghani, M., Gouws, S., Vinyals, O., Uszkoreit, J. & Kaiser, Ł. (2019), Universal transformers, *in* '2019 International Conference on Learning Respresentations (ICLR)'.

Demszky, D., Movshovitz-Attias, D., Ko, J., Cowen, A., Nemade, G. & Ravi, S. (2020), 'GoEmotions: A dataset of fine-grained emotions', *arXiv preprint arXiv:2005.00547* .

Demuynck, K., Seppi, D., Van Compernolle, D., Nguyen, P. & Zweig, G. (2011), Integrating meta-information into exemplar-based speech recognition with segmental conditional random fields, *in* '2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)', pp. 5048–5051.

Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. (2019), Bert: Pre-training of deep bidirectional transformers for language understanding, *in* 'North American Chapter of the Association for Computational Linguistics'.
URL: *https://api.semanticscholar.org/CorpusID:52967399*

Diederik, P. K. (2014), 'Adam: A method for stochastic optimization', *(No Title)* .

Do, H. (1949), 'The organization of behavior', *New York* .

Doclo, S., Kellermann, W., Makino, S. & Nordholm, S. E. (2015), 'Multichannel Signal Enhancement Algorithms for Assisted Listening Devices: Exploiting spatial diversity using multiple microphones', *IEEE Signal Processing Magazine* **32**(2), 18–30.

Elio, R. & Anderson, J. R. (1984), 'The effects of information order and learning mode on schema abstraction', *Memory & cognition* **12**(1), 20–30.

Elman, J. L. (1990), 'Finding structure in time', *Cognitive Science* **14**, 179–211.

Erickson, M. A. & Kruschke, J. K. (1998), 'Rules and exemplars in category learning.', *Journal of Experimental Psychology: General* **127**.

Fan, Z., Li1, M., Zhou, S. & Xu, B. (2021), Exploring wav2vec 2.0 on speaker verification and language identification, *in* 'Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH 2021', pp. 1509–1513.

Firth, J. (1957), 'A synopsis of linguistic theory, 1930-1955', *Studies in linguistic analysis* pp. 10–32.

Fix, E. & Hodges, J. L. (1951), 'Discriminatory analysis, non-parametric discrimination', *USAF School of Aviation Medicine, Randolph Field, Tex., Project 21-49-004, Rept. 4, Contract AF41(128)-31* .

Fukushima, K. (1969), 'Visual feature extraction by a multilayered network of analog threshold elements', *IEEE Transactions on Systems Science and Cybernetics* **5**(4), 322–333.

Fukushima, K. (1980), 'Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position', *Biological cybernetics* **36**(4), 193–202.

Furui, S. (1986), Speaker-independent isolated word recognition based on emphasized spectral dynamics, *in* 'ICASSP '86. IEEE International Conference on Acoustics, Speech, and Signal Processing', Vol. 11, pp. 1991–1994.

Gallo, D. A. (2010), 'False memories and fantastic beliefs: 15 years of the drm illusion', *Memory & cognition* **38**, 833–848.

Gan, Y., Liu, J., Dong, J. & Zhong, G. (2015), 'A pca-based convolutional network', *arXiv preprint arXiv:1505.03703* .

Garnelo, M., Rosenbaum, D., Maddison, C. J., Ramalho, T., Saxton, D., Shanahan, M., Teh, Y. W., Rezende, D. J. & Eslami, S. M. A. (2018), 'Conditional neural processes', *arXiv* **1807.01613**.

Garofolo, J. S. (1993), 'Timit acoustic phonetic continuous speech corpus', *Linguistic Data Consortium, 1993* .

Gehring, J., Auli, M., Grangier, D., Yarats, D. & Dauphin, Y. N. (2017), Convolutional sequence to sequence learning, *in* 'International conference on machine learning', PMLR, pp. 1243–1252.

Gers, F. A. & Schmidhuber, J. (2000), Recurrent nets that time and count, *in* 'Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium', Vol. 3, IEEE, pp. 189–194.

Gers, F. A., Schraudolph, N. N. & Schmidhuber, J. (2002), 'Learning precise timing with lstm recurrent networks', *Journal of machine learning research* **3**(Aug), 115–143.

Glorot, X. & Bengio, Y. (2010), Understanding the difficulty of training deep feedforward neural networks, *in* 'Proceedings of the thirteenth international conference on artificial intelligence and statistics', JMLR Workshop and Conference Proceedings, pp. 249–256.

Goetze, S., Xiong, F., Rennies, J., Rohdenburg, T. & Appell, J. (2010), Hands-Free Telecommunication for Elderly Persons Suffering from Hearing Deficiencies, *in* 'IEEE Int. Conf. on E-Health Networking, Application and Services (Healthcom'10)'.

Golipour, L. & O'Shaughnessy, D. (2009), Context-independent phoneme recognition using a k-nearest neighbour classification approach, *in* '2009 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)', pp. 1341–1344.

Graves, A. & Schmidhuber, J. (2005), 'Framewise phoneme classification with bidirectional lstm and other neural network architectures', *Neural networks* **18**(5-6), 602–610.

Graves, A. & Schmidhuber, J. (2008), 'Offline handwriting recognition with multidimensional recurrent neural networks', *Advances in neural information processing systems* **21**, 545–552.

Gulati, A., Qin, J., Chiu, C.-C., Parmar, N., Zhang, Y., Yu, J., Han, W., Wang, S., Zhang, Z., Wu, Y. et al. (2020), 'Conformer: Convolution-augmented transformer for speech recognition', *arXiv preprint arXiv:2005.08100* .

Guo, C., Pleiss, G., Sun, Y. & Weinberger, K. Q. (2017), On calibration of modern neural networks, *in* 'International conference on machine learning', PMLR, pp. 1321–1330.

Günther, F., Dudschig, C. & Kaup, B. (2015), 'Lsafun - an r package for computations based on latent semantic analysis', *Behavior Research Methods* **47**, 930–944.

He, K., Zhang, X., Ren, S. & Sun, J. (2015), Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, *in* 'Proceedings of the IEEE international conference on computer vision', pp. 1026–1034.

Heck, J. C. & Salem, F. M. (2017), Simplified minimal gated unit variations for recurrent neural networks, *in* '2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)', IEEE, pp. 1593–1596.

Hintzman, D. (1984), 'Minerva 2: a simulation model of human memory', *Behav. Res. Methods Instrum. Comput.* **16**, 96–101.

Hintzman, D. L. (1986), '"schema abstraction" in a multiple-trace memory model.', *Psychological review* **93**(4), 411.

Hintzman, D. L. (1988), 'Judgments of frequency and recognition memory in a multiple-trace memory model', *Psychological review* **95**(4), 528.

Hochberg, J., ed. (1998), *Handbook of perception and cognition (2nd ed.). Perception and cognition at century's end*, second edn, Academic Press.

Hochreiter, S. & Schmidhuber, J. (1997), 'Long short-term memory', *Neural computation* **9**, 1735–1780.

Hoffman, J. D. & Frankel, S. (2018), *Numerical methods for engineers and scientists*, 2nd edn, CRC press.

Hornik, K., Stinchcombe, M. & White, H. (1989), 'Multilayer feedforward networks are universal approximators', *Neural Networks* **2**(5), 359–366.

Hsu, W.-N., Bolte, B., Tsai, Y.-H. H., Lakhotia, K., Salakhutdinov, R. & Mohamed, A. (2021), 'Hubert: Self-supervised speech representation learning by masked prediction of hidden units', *IEEE/ACM transactions on audio, speech, and language processing* **29**, 3451–3460.

Hubel, D. H. & Wiesel, T. N. (1962), 'Receptive fields, binocular interaction and functional architecture in the cat's visual cortex', *The Journal of physiology* **160**(1), 106.

Hubel, D. H. & Wiesel, T. N. (1965), 'Receptive fields and functional architecture in two nonstriate visual areas (18 and 19) of the cat', *Journal of neurophysiology* .

Humphreys, M. S., Bain, J. D. & Pike, R. (1989), 'Different ways to cue a coherent memory system: A theory for episodic, semantic, and procedural tasks.', *Psychological Review* **96**(2), 208.

Humphreys, M. S., Pike, R., Bain, J. D. & Tehan, G. (1989), 'Global matching: A comparison of the sam, minerva ii, matrix, and todam models', *Journal of Mathematical Psychology* **33**(1), 36–67.

Hunt, A. J. & Black, A. W. (1996), Unit selection in a concatenative speech synthesis system using a large speech database, *in* '1996 IEEE international conference on acoustics, speech, and signal processing conference proceedings', Vol. 1, IEEE, pp. 373–376.

Jamali, M., Grannan, B., Cai, J., Khanna, A. R., Muñoz, W., Caprara, I., Paulk, A. C., Cash, S. S., Fedorenko, E. & Williams, Z. M. (2024), 'Semantic encoding during language comprehension at single-cell resolution', *Nature* **631**(8021), 610–616.

Jamieson, R. K. & Mewhort, D. (2009), 'Applying an exemplar model to the artificial-grammar task: Inferring grammaticality from similarity', *Quarterly Journal of Experimental Psychology* **62**(3), 550–575.

Jegou, H., Douze, M. & Schmid, C. (2010), 'Product quantization for nearest neighbor search', *IEEE transactions on pattern analysis and machine intelligence* **33**(1), 117–128.

Kates, J. M. & Arehart, K. H. (2021), 'The Hearing-aid Speech Perception Index (HASPI) Version 2', *Speech Communication* **131**, 35–46.

Khan, R. A. & Chitode, J. S. (2016), 'Concatenative speech synthesis: A review', *International Journal of Computer Applications* **136**(3), 1–6.

Kirchner, R., Moore, R. K. & Chen, T.-Y. (2010), 'Computing phonological generalization over real speech exemplars', *Journal of Phonetics* **38**(4), 540–547.

Krähenbühl, P., Doersch, C., Donahue, J. & Darrell, T. (2015), 'Data-dependent initializations of convolutional neural networks', *arXiv preprint arXiv:1511.06856* .

Labiak, J. & Livescu, K. (2011), Nearest neighbors with learned distances for phonetic frame classification, *in* 'Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH 2011', pp. 2337–2340.

Lei Ba, J., Kiros, J. R. & Hinton, G. E. (2016), 'Layer normalization', *ArXiv e-prints* pp. arXiv–1607.

Leshno, M., Lin, V. Y., Pinkus, A. & Schocken, S. (1993), 'Multilayer feedforward networks with a nonpolynomial activation function can approximate any function', *Neural Networks* **6**(6), 861–867.

Liu, H., Perera, L. P. G., Khong, A. W., Chng, E. S., Styles, S. J. & Khudanpur, S. (2022), 'Efficient self-supervised learning representations for spoken language identification', *IEEE Journal of Selected Topics in Signal Processing* **16**(6), 1296–1307.

Liu, Y. (2019), 'Roberta: A robustly optimized bert pretraining approach', *arXiv preprint arXiv:1907.11692* .

Lopes, C. & Perdigão, F. (2011), 'Phone recognition on timit database', *Speech Technologies* **1**, 285–382.

Mack, M. L., Preston, A. R. & Love, B. C. (2013), 'Decoding the brain's algorithm for categorization from its neural implementation', *Current Biology* **23**(20), 2023–2027.

Maier, V. & Moore, R. K. (2005), An investigation into a simulation of episodic memory for automatic speech recognition, *in* 'Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH 2005', pp. 1245–1248.

Maier, V. & Moore, R. K. (2007), Temporal episodic memory model: an evolution of Minerva2, *in* 'Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH 2007', pp. 866–869.

McCulloch, W. S. & Pitts, W. (1943), 'A logical calculus of the ideas immanent in nervous activity', *The bulletin of mathematical biophysics* **5**, 115–133.

Medin, D. L. & Coley, J. D. (1998), *Concepts and Categorization*, Vol. 1 of Hochberg (1998), second edn, chapter 3, pp. 403–439.

Mermelstein, P. (1976), 'Distance measures for speech recognition, psychological and instrumental', *Pattern recognition and artificial intelligence* **116**, 374–388.

Mikolov, T., Chen, K., Corrado, G. & Dean, J. (2013), 'Efficient estimation of word representations in vector space', *arXiv preprint arXiv:1301.3781* .

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S. & Dean, J. (2013), 'Distributed representations of words and phrases and their compositionality', *Advances in neural information processing systems* **26**.

Mogridge, R., Close, G., Sutherland, R., Hain, T., Barker, J., Goetze, S. & Ragni, A. (2024), Non-intrusive speech intelligibility prediction for hearing-impaired users using intermediate asr features and human memory models, *in* 'ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)', IEEE, pp. 306–310.

Moore, R. & Maier, V. (2007), Preserving fine phonetic detail using episodic memory: Automatic speech recognition using minerva2, *in* 'International Congress of Phonetic Sciences, 2007', pp. 197–203.

Morais, E., Hoory, R., Zhu, W., Gat, I., Damasceno, M. & Aronowitz, H. (2022), Speech emotion recognition using self-supervised features, *in* 'ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)', pp. 6922–6926.

Moulines, E. & Charpentier, F. (1990), 'Pitch-synchronous waveform processing techniques for text-to-speech synthesis using diphones', *Speech communication* **9**(5-6), 453–467.

Murdock, B. B. (2014), Learning in a distributed memory model, *in* 'Current issues in cognitive processes', Psychology Press, pp. 69–106.

Natal, S., McLaren, I. & Livesey, E. (2013), 'Generalization of feature- and rule-based learning in the categorization of dimensional stimuli: evidence for dual processes under cognitive control', *J Exp Psychol Anim Behav Process* **39**(2), 140–51.

Nick Reid, J. & Jamieson, R. K. (2023), 'True and false recognition in minerva 2: Extension to sentences and metaphors', *Journal of Memory and Language* **129**, 104397.

Nosofsky, R. M. (1986), 'Attention, similarity, and the identification–categorization relationship.', *Journal of experimental psychology: General* **115**(1), 39.

Nosofsky, R. M., Palmeri, T. J. & Stephen C, M. (1994), 'Rule-plus-exception model of classification learning', *Psychological Review* **101**(1), 53–79.

Oruh, J., Viriri, S. & Adegun, A. (2022), 'Long short-term memory recurrent neural network for automatic speech recognition', *IEEE Access* **10**, 30069–30079.

Papala, G., Ransing, A. & Jain, P. (2023), 'Sentiment analysis and speaker diarization in hindi and marathi using using finetuned whisper: Sentiment analysis in hindi and marathi', *Scalable Computing: Practice and Experience* **24**(4), 835–846.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J. & Chintala, S. (2019), Pytorch: An imperative style, high-performance deep learning library, *in* H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox & R. Garnett, eds, 'Advances in Neural Information Processing Systems 32', Curran Associates, Inc., pp. 8024–8035.

Peterson, G. E. & Barney, H. L. (1951), 'Control methods used in a study of the vowels', *Journal of the Acoustical Society of America* **24**, 175–184.

Pike, R. (1984), 'Comparison of convolution and matrix distributed memory systems for associative recall and recognition.', *Psychological Review* **91**(3), 281.

Pisoni, D. B., Nusbaum, H. C., Luce, P. A. & Slowiaczek, L. M. (1985), 'Speech perception, word recognition and the structure of the lexicon', *Speech Communication* **4**(1), 75–95.

Raaijmakers, J. G. & Shiffrin, R. M. (1981), 'Search of associative memory.', *Psychological review* **88**(2), 93.

Radford, A., Kim, J. W., Xu, T., Brockman, G., McLeavey, C. & Sutskever, I. (2023), 'Robust speech recognition via large-scale weak supervision'.

Ranny (2016), Voice recognition using k nearest neighbor and double distance method, *in* '2016 International Conference on Industrial Engineering, Management Science and Application (ICIMSA)', pp. 1–5.

Rao, M. N., Thomas, S., Nagarajan, T. & Murthy, H. A. (2005), Text-to-speech synthesis using syllable-like units, *in* 'National conference on communication', pp. 227–280.

Reber, A. S. (1967), 'Implicit learning of artificial grammars', *Journal of verbal learning and verbal behavior* **6**(6), 855–863.

Reimers, N. & Gurevych, I. (2019), Sentence-bert: Sentence embeddings using siamese bert-networks, *in* 'Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing', Association for Computational Linguistics.

Repp, B. H. (1982), 'Phonetic trading relations and context effects: New experimental evidence for a speech mode of perception.', *Psychological bulletin* **92**(1), 81.

Riedmiller, M. & Braun, H. (1992), Rprop: a fast adaptive learning algorithm, *in* 'Proc. of the Int. Symposium on Computer and Information Science VII'.

Roediger, H. L. & McDermott, K. B. (1995), 'Creating false memories: Remembering words not presented in lists.', *Journal of experimental psychology: Learning, Memory, and Cognition* **21**(4), 803.

Rosenblatt, F. (1958), 'The perceptron: a probabilistic model for information storage and organization in the brain.', *Psychological review* **65**(6), 386.

Rouder, J. N. & Ratcliff, R. (2006), 'Comparing exemplar- and rule-based theories of categorization', *Current Directions in Psychological Science* **15**.

Sainath, T. N., Ramabhadran, B., Nahamoo, D., Kanevsky, D., Compernolle, D. V., Demuynck, K., Gemmeke, J. F., Bellegarda, J. R. & Sundaram, S. (2012), 'Exemplar-based processing for speech recognition: An overview', *IEEE Signal Processing Magazine* **29**(6), 98–113.

Sak, H., Güngör, T. & Safkan, Y. (2006), 'A corpus-based concatenative speech synthesis system for turkish', *Turkish Journal of Electrical Engineering and Computer Sciences* **14**(2), 209–223.

Sakoe, H. & Chiba, S. (1978), 'Dynamic programming algorithm optimization for spoken word recognition', *IEEE Transactions on Acoustics, Speech, and Signal Processing* **26**(1), 43–49.

Schneider, S., Baevski, A., Collobert, R. & Auli, M. (2019), wav2vec: Unsupervised pre-training for speech recognition, *in* 'Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH 2019', pp. 3465–3469.

Schneider, W. & Shiffrin, R. M. (1977), 'Controlled and automatic human information processing: I. detection, search, and attention.', *Psychological review* **84**(1), 1.

Sharma, M. (2022), Multi-lingual multi-task speech emotion recognition using wav2vec 2.0, *in* 'ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)', IEEE, pp. 6907–6911.

Shiffrin, R. M. & Schneider, W. (1977), 'Controlled and automatic human information processing: Ii. perceptual learning, automatic attending and a general theory.', *Psychological review* **84**(2), 127.

Song, K., Tan, X., Qin, T., Lu, J. & Liu, T.-Y. (2020), Mpnet: Masked and permuted pre-training for language understanding, *in* 'NeurIPS 2020', ACM.

Stanovich, K. E. & West, R. F. (1983), 'On priming by a sentence context.', *Journal of Experimental Psychology: General* **112**(1), 1.

Stevens, S. S., Volkmann, J. & Newman, E. B. (1937), 'A scale for the measurement of the psychological magnitude pitch', *The journal of the acoustical society of america* **8**(3), 185–190.

Su, J., Ahmed, M., Lu, Y., Pan, S., Bo, W. & Liu, Y. (2024), 'Roformer: Enhanced transformer with rotary position embedding', *Neurocomputing* **568**, 127063.

Tabet, Y. & Boughazi, M. (2011), Speech synthesis techniques. a survey, *in* 'International Workshop on Systems, Signal Processing and their Applications, WOSSPA', IEEE, pp. 67–70.

Tak, H., Todisco, M., Wang, X., weon Jung, J., Yamagishi, J. & Evans, N. (2022), Automatic Speaker Verification Spoofing and Deepfake Detection Using Wav2vec 2.0 and Data Augmentation, *in* 'Proc. The Speaker and Language Recognition Workshop (Odyssey 2022)', pp. 112–119.

Tamm, B., Balabin, H., Vandenberghe, R. & hamme, H. V. (2022), Pre-trained Speech Representations as Feature Extractors for Speech Quality Assessment in Online Conferencing Applications, *in* 'Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH 2022', ISCA, pp. 4083–4087.

Tjandra, A., Choudhury, D. G., Zhang, F., Singh, K., Conneau, A., Baevski, A., Sela, A., Saraf, Y. & Auli, M. (2022), Improved language identification through cross-lingual self-supervised learning, *in* 'ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)', IEEE, pp. 6877–6881.

Tu, Z., Ma, N. & Barker, J. (2022), Unsupervised Uncertainty Measures of Automatic Speech Recognition for Non-intrusive Speech Intelligibility Prediction, *in* 'Proc. Interspeech', pp. 3493–3497.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. & Polosukhin, I. (2017), 'Attention is all you need', *Advances in neural information processing systems* **30**.

Venkata Subbarao, M., Terlapu, S. K., Geethika, N. & Harika, K. D. (2022), Speech emotion recognition using k-nearest neighbor classifiers, *in* P. Shetty D. & S. Shetty, eds, 'Recent Advances in Artificial Intelligence and Data Engineering', Springer Singapore, Singapore, pp. 123–131.

Viganò, S., Rubino, V., Soccio, A. D., Buiatti, M. & Piazza, M. (2021), 'Grid-like and distance codes for representing word meaning in the human brain', *NeuroImage* **232**, 117876.

Vintsyuk, T. (1968), 'Speech discrimination by dynamic programming', *Cybernetics and System Analysis* **4**(1), 52–57.

Wachter, M. D., Demuynck, K., Compernolle, D. V. & Wambacq, P. (2003), Data driven example based continuous speech recognition, *in* 'Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH 2003', pp. 1133–1136.

Wachter, M., Matton, M., Demuynck, K., Wambacq, P., Cools, R. & Van Compernolle, D. (2007), ‘Template-based continuous speech recognition’, *IEEE Transactions on audio, Speech, and Language Processing* **15**, 1377 – 1390.

Waibel, A., Hanazawa, T., Hinton, G., Shikano, K. & Lang, K. (1989), ‘Phoneme recognition using time-delay neural networks’, *IEEE Transactions on Acoustics, Speech, and Signal Processing* **37**.

Wang, W., Dong, L., Cheng, H., Liu, X., Yan, X., Gao, J. & Wei, F. (2024), ‘Augmenting language models with long-term memory’, *Advances in Neural Information Processing Systems* **36**.

Wang, Y., Boumadane, A. & Heba, A. (2021), ‘A fine-tuned wav2vec 2.0/hubert benchmark for speech emotion recognition, speaker verification and spoken language understanding’, *arXiv preprint arXiv:2111.02735* .

Wu, Y., Rabe, M. N., Hutchins, D. & Szegedy, C. (2022), Memorizing transformers, *in* ‘2022 International Conference on Learning Respresentations (ICLR)’.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R. & Le, Q. V. (2019), Xlnet: Generalized autoregressive pretraining for language understanding, *in* H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox & R. Garnett, eds, ‘Advances in Neural Information Processing Systems’, Vol. 32, Curran Associates, Inc.

Zezario, R. E., Chen, F., Fuh, C.-S., Wang, H.-M. & Tsao, Y. (2022), ‘MBI-Net: A Non-Intrusive Multi-Branched Speech Intelligibility Prediction Model for Hearing Aids’.

Zhang, L., Jiang, N., Wang, Q., Li, Y., Lu, Q. & Xie, L. (2024), ‘Whisper-sv: Adapting whisper for low-data-resource speaker verification’, *Speech Communication* **163**, 103103.

Zhong, Z., Lei, T. & Chen, D. (2022), ‘Training language models with memory augmentation’, *arXiv preprint arXiv:2205.12674* .

Appendices

Appendix A

Hyperparameters and tuning

Models were initially trained with the hyperparameters given in Table A.1. Each hyperparameter was then adjusted both up and down in the following way:

- the learning rate was incremented/decremented by a factor of 10
- the weight decay was incremented/decremented by a factor of 10
- dropout was incremented/decremented by a step of 0.1
- activation power was incremented/decremented by a step of 2

This resulted in a search space of 9 initial models (or 7 for models that do not use β). If the best performing model fell at the edge of the search space (i.e. it did not use the hyperparameters in Table A.1), then the 'base' value for that parameter was adjusted to the better value, and the search was repeated. This process was repeated until the best performing model did not fall at the edge of the search space.

Table A.1: *Starting values for hyperparameter reduced grid search.*

Learning rate	**Weight decay**	**Dropout**	$\boldsymbol{\beta}$
10^{-3}	10^{-3}	0.1	3

Tables A.2 to A.18 give the final hyperparameter values for all experiments reported in Chapter 4.

Table A.2: *Tuned hyperparameter values for the TIMIT FFNN and Minerva-RP models (Table 4.1).*

Features	**Model**	**Feature transform**	**Learning rate**	**Weight decay**	**Dropout**
Mel spec	Minerva-RP	Shared	10^{-3}	10^{5}	0.1
		Separate	10^{-3}	10^{-6}	0.0
	FFNN	-	10^{-5}	10^{-4}	0.3
Wav2vec	Minerva-RP	Shared	10^{-3}	10^{-7}	0.0
		Separate	10^{-3}	10^{-7}	0.0
	FFNN	-	10^{-5}	10^{-5}	0.5
HuBERT	Minerva-RP	Shared	10^{-4}	10^{-5}	0.1
		Separate	10^{-3}	10^{-7}	0.1
	FFNN	-	10^{-4}	10^{-6}	0.7

Table A.3: *Tuned hyperparameter values for the GoEmotions FFNN and Minerva-RP models (Table 4.2).*

Features	**Classifier**	**Feature transform**	**Learning rate**	**Weight decay**	**Dropout**
LSA	Minerva-RP	Shared	10^{-2}	10^{-3}	0.2
		Separate	10^{-4}	10^{-3}	0.0
	FFNN	-	10^{-4}	10^{-1}	0.6
Word2vec	Minerva-RP	Shared	10^{-2}	10^{-3}	0.1
		Separate	10^{-1}	10^{-3}	0.1
	FFNN	-	10^{-4}	1	0.7
SBERT	Minerva-RP	Shared	10^{-2}	10^{-2}	0.3
		Separate	10^{-2}	10^{-3}	0.4
	FFNN	-	10^{-4}	1	0.9

Table A.4: *Tuned hyperparameter values for the CPC2 FFNN and Minerva-RP models (Table 4.3.)*

Features	**Classifier**	**Feature transform**	**Learning rate**	**Weight decay**	**Dropout**
Log spec	Minerva-RP	Shared	10^{-2}	10^{-4}	0.0
		Separate	10^{-2}	10^{-4}	0.0
	FFNN	-	10^{-2}	10^{-4}	0.1
XLSR	Minerva-RP	Shared	10^{-2}	10^{-4}	0.0
		Separate	10^{-2}	10^{-4}	0.0
	FFNN	-	10^{-4}	10^{-5}	0.2
Whisper$^{\text{decoder}}$	Minerva-RP	Shared	10^{-2}	10^{-4}	0.0
		Separate	10^{-2}	10^{-4}	0.0
	FFNN	-	10^{-3}	10^{-3}	0.0

Table A.5: *Tuned hyperparameter values for the TIMIT Minerva-RPE models (Table 4.4).*

Features	**Classifier**	**Learning rate**	**Weight decay**	**Dropout**
Mel Spec	Minerva-RPE	10^{-3}	10^{-7}	0.0
Wav2vec	Minerva-RPE	10^{-3}	10^{-7}	0.0
HuBERT	Minerva-RPE	10^{-3}	10^{-7}	0.2

Table A.6: *Tuned hyperparameter values for the GoEmotions Minerva-RPE models (Table 4.5).*

Features	**Classifier**	**Learning rate**	**Weight decay**	**Dropout**
LSA	Minerva-RPE	10^{-2}	10^{-2}	0.2
Word2vec	Minerva-RPE	10^{-1}	10^{-3}	0.1
SBERT	Minerva-RPE	10^{-2}	10^{-2}	0.3

Table A.7: *Tuned hyperparameter values for the CPC2 Minerva-RPE models (Table 4.6).*

Features	**Classifier**	**Learning rate**	**Weight decay**	**Dropout**
Log spec	Minerva-RPE	10^{-3}	10^{-3}	0.0
XLSR	Minerva-RPE	10^{-3}	10^{-3}	0.0
Whisper	Minerva-RPE	10^{-2}	10^{-3}	0.0

Table A.8: *Tuned hyperparameter values for the TIMIT Minerva-RPC and Minerva-RPCE models (Table 4.7).*

Features	**Classifier**	**Learning rate**	**Weight decay**	**Dropout**
Mel Spec	Minerva-RPC	10^{-3}	10^{-7}	0.1
	Minerva-RPCE	10^{-3}	10^{-6}	0.0
Wav2vec	Minerva-RPC	10^{-2}	10^{-7}	0.0
	Minerva-RPCE	10^{-3}	10^{-7}	0.0
HuBERT	Minerva-RPC	10^{-3}	10^{-7}	0.2
	Minerva-RPCE	10^{-3}	10^{-7}	0.2

Table A.9: *Tuned hyperparameter values for the GoEmotions Minerva-RPC and Minerva-RPCE models (Table 4.8).*

Features	**Classifier**	**Learning rate**	**Weight decay**	**Dropout**
LSA	Minerva-RPC	10^{-2}	10^{-2}	0.2
	Minerva-RPCE	10^{-2}	10^{-3}	0.3
Word2vec	Minerva-RPC	10^{-2}	10^{-4}	0.2
	Minerva-RPCE	10^{-1}	10^{-4}	0.2
SBERT	Minerva-RPC	10^{-4}	10^{-3}	0.3
	Minerva-RPCE	10^{-4}	10^{-3}	0.3

Table A.10: *Tuned hyperparameter values for the TIMIT Minerva-RP and Minerva-RPC models with unfixed exemplars (Table 4.9).*

Features	**Model**	**Learning rate**	**Weight decay**	**Dropout**
Mel Spec	Minerva-RP	10^{-3}	10^{-7}	0.1
	Minerva-RPC	10^{-3}	10^{-7}	0.0
Wav2vec	Minerva-RP	10^{-3}	10^{-7}	0.0
	Minerva-RPC	10^{-3}	10^{-7}	0.0
HuBERT	Minerva-RP	10^{-3}	10^{-6}	0.1
	Minerva-RPC	10^{-3}	10^{-7}	0.1

Table A.11: *Tuned hyperparameter values for the GoEmotions Minerva-RP and Minerva-RPC models with unfixed exemplars (Table 4.10).*

Features	**Model**	**Learning rate**	**Weight decay**	**Dropout**
LSA	Minerva-RP	10^{-2}	10^{-4}	0.0
	Minerva-RPC	10^{-2}	10^{-5}	0.3
Word2vec	Minerva-RP	10^{-2}	10^{-4}	0.3
	Minerva-RPC	10^{-2}	10^{-4}	0.3
SBERT	Minerva-RP	10^{-3}	10^{-3}	0.4
	Minerva-RPC	10^{-3}	10^{-1}	0.3

Table A.12: *Tuned hyperparameter values for the CPC2 Minerva-RP models with unfixed exemplars (Table 4.11).*

Features	**Model**	**Learning rate**	**Weight decay**	**Dropout**
Log spec	Minerva-RP	10^{-2}	10^{-4}	0.0
XLSR	Minerva-RP	10^{-5}	10^{-2}	0.0
Whisper	Minerva-RP	10^{-3}	10^{-3}	0.2

Table A.13: *Tuned hyperparameter values for the TIMIT Minerva-RP models with adaptive exemplars (Table 4.12).*

Features	**Model**	**Learning rate**	**Weight decay**	**Dropout**
Mel Spec	Minerva-RP	10^{-3}	10^{-6}	0.1
Wav2vec	Minerva-RP	10^{-2}	10^{-7}	0.0
HuBERT	Minerva-RP	10^{-3}	10^{-6}	0.1

Table A.14: *Tuned hyperparameter values for the TIMIT Minerva-RPC models with adaptive exemplars (Table 4.13).*

Features	**Model**	**Learning rate**	**Weight decay**	**Dropout**
Mel Spec	Minerva-RP	10^{-3}	10^{-6}	0.0
Wav2vec	Minerva-RP	10^{-2}	10^{-7}	0.0
HuBERT	Minerva-RP	10^{-3}	10^{-6}	0.1

Table A.15: *Tuned hyperparameter values for the CPC2 Minerva-RP models with adaptive exemplars (Table 4.14).*

Features	**Exemplars**	**Learning rate**	**Weight decay**	**Dropout**
Log spec	Fixed	10^{-2}	10^{-4}	0.0
	Unfixed	10^{-2}	10^{-4}	0.0
	Listener adapted	10^{-2}	10^{-2}	0.0
	System adapted	10^{-2}	10^{-4}	0.0
XLSR	Fixed	10^{-4}	10^{-3}	0.0
	Unfixed	10^{-5}	10^{-2}	0.0
	Listener adapted	10^{-5}	10^{-3}	0.0
	System adapted	10^{-5}	10^{-2}	0.0
Whisper	Fixed	10^{-3}	10^{-3}	0.0
	Unfixed	10^{-3}	10^{-3}	0.1
	Listener adapted	10^{-3}	10^{-3}	0.4
	System adapted	10^{-3}	10^{-4}	0.2

Table A.16: *Tuned hyperparameter values for the TIMIT Minerva-RPES models (Table 4.15).*

Features	**Classifier**	**Learning rate**	**Weight decay**	**Dropout**
Mel Spec	Minerva-RPES	10^{-3}	10^{-7}	0.0
Wav2vec	Minerva-RPES	10^{-3}	10^{-7}	0.0
HuBERT	Minerva-RPES	10^{-3}	10^{-7}	0.4

Table A.17: *Tuned hyperparameter values for the TIMIT Minerva-RPES and transformer models with Mel spectrogram features (Table 4.16).*

Exemplars	**Activations**	**FFNN +**	**Learning**	**Weight**	**Dropout**
		layer norm rate	**decay**		
Yes	Minerva	None	10^{-3}	10^{-3}	0.0
		Both	10^{-3}	10^{-7}	0.0
	scaled dot-	None	10^{-3}	10^{-7}	0.0
	product	Both	10^{-3}	10^{-6}	0.0
No	Minerva	None	10^{-2}	10^{-7}	0.0
		Both	10^{-2}	10^{-7}	0.0
	scaled dot-	None	10^{-4}	10^{-7}	0.1
	product	Both	10^{-3}	10^{-5}	0.0
FFNN	-	No	10^{-3}	10^{-5}	0.0

Table A.18: *Tuned hyperparameter values for the TIMIT Minerva-RPES and transformer models with HuBERT features (Table 4.16).*

Exemplars	**Activations**	**FFNN +**	**Learning**	**Weight**	**Dropout**
		layer norm rate	**decay**		
Yes	Minerva	None	10^{-4}	10^{-5}	0.2
		Both	10^{-3}	10^{-6}	0.3
	scaled dot-	None	10^{-3}	10^{-6}	0.2
	product	Both	10^{-3}	10^{-7}	0.4
No	Minerva	None	10^{-3}	10^{-7}	0.2
		Both	10^{-3}	10^{-7}	0.2
	scaled dot-	None	10^{-4}	10^{-6}	0.2
	product	Both	10^{-3}	10^{-3}	0.1
FFNN	-	No	10^{-4}	10^{-7}	0.1