UNIVERSITY OF CAMBRIDGE

Department of Engineering

Neural Network Based Diarisation

Author Name: Danyi (Oli) Liu

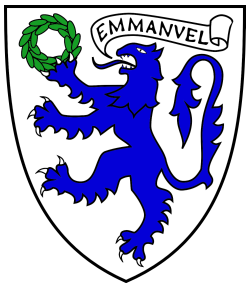
Supervisor: Prof. Phil Woodland and Dr. Chao Zhang

Date: 27/05/2020

I hereby declare that, except where specifically indicated, the work submitted herin is my own original work.

Signed _____ *date* 27/05/2020

Neural Network Based Diarisation



Danyi (Oli) Liu

Supervisor: Prof. Phil Woodland
Dr. Chao Zhang

Department of Engineering
University of Cambridge

Final Report for Fourth-year Project
Submitted in Partial Fulfilment of the Requirements for the Degree of
Master of Engineering in Information and Computer Engineering

Emmauel College June 2020

Neural Network Based Diarisation
Technical Abstract

Danyi Liu, Emmanuel College

Speaker diarisation is the process of determining "who spoke when" given a stream of audio data. Central to the accuracy of the task is the quality of *speaker embeddings*, which are high-dimensional vectors extracted to represent speaker-specif characteristics. With the recent development of deep leaning, extraction methods based on deep neural networks (DNN) have exhibited superiority over traditional methods such as factor analysis. Although most state-of-the-art embedding extractors only utilise the acoustic features of the input audio stream, this thesis investigates using content cues to inform speaker identity by incorporating phoneme and grapheme transcriptions into the DNN extractor.

We first proposed using an early fusion model to combine content features and acoustic features, allowing complementary information from the two modalities to be combined by the extractor. Having evaluated fusion models using 10 different pairings of training evaluation transcriptions, each pair under 4 experiment settings, we discovered that the fusion model outperformed the baseline in average speaker error rate by as much as 21.61 % in one particular setting and was only outperformed by the baseline once by 1.64% among total 40 results. The best performances were obtained when the model was trained on reference grapheme transcriptions, and the same good performance was maintained regardless of whether reference or hypothesis transcription was used during evaluation.

Another approach that we explored was performing multi-task adversarial training in order to obtain speaker embeddings that are discriminative in speaker characteristics and at the same time invariant to the phone being pronounced. We added a phone classifier to the embedding extractor, and introduced a gradient reversal layer to optimise for minimum speaker classification loss and maximum phone classification loss at the same time, with the weighting of the two loss terms controlled by the reversal coefficient of the gradient reversal layer. However, the performance of the extractor trained this way could hardly match that of the baseline and could only converge when the phone loss was weighted to be more than

three magnitudes smaller than the speaker loss. However, when we performed multi-task learning that minimised the phone loss and the speaker loss simultaneously, there were minor improvements in the classification accuracy, potentially because of the regularisation effect of the phone classifier.

In conclusion, multimodal fusion of acoustic feature and grapheme transcriptions can significantly improve diarisation accuracy and is worth further research to examine its robustness against transcription errors as well as generalisability to other languages.

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 12,000 words including figures and appendices, but not counting the title page, bibliography and technical abstract.

Danyi (Oli) Liu
June 2020

Acknowledgements

I would like to thank my supervisors Professor Phil Woodland and Dr. Chao Zhang for their constant support throughout the project, which has been unwavering throughout the year and ever stronger towards the end, despite the very special circumstances. I am grateful to Prof. Woodland for his many inductive and inspiring suggestions regarding both technical details of experiments and high-level research methodologies. I am also in debt to Dr. Zhang, who have helped me with and advised me on almost every aspect of the experiments in this project, from whom I have learnt countless lessons about doing proper research, designing and implementing experiments.

I would also like to than Brian and Yassir for their patient help with putting up the baseline system, as well as Florian and Eric for their kind advice. I am also thankful for my friends here at Cambridge and also at home, whose encouragement has helped me to combat all the stress in the past year.

Finally, I am forever grateful to my parents and other faimilies for their constant love and support, which made all of this possible.

Table of contents

1 Introduction **1**

2 Background **3**

2.1 Diarisation System 3

2.1.1 Initial Segmentation 3

2.1.2 Extraction of Speaker Embeddings 4

2.1.3 Clustering 4

2.1.4 Optional Re-segmentation 4

2.2 Motivations 5

2.2.1 Capturing Idiosyncrasies in Pronunciation 5

2.2.2 Suppressing Phone Information 5

2.2.3 Exploiting Idiosyncrasies in Word Choice 5

3 Diarisation with D-vectors **6**

3.1 Neural Network Architecture 6

3.2 Aggregation Method 8

3.3 Training Objective 9

3.4 Clustering 11

4 Incorporation of Content Information **14**

4.1 Multimodal Fusion 14

4.2 Phone-invariant Training Via Adversarial Learning 14

4.2.1 Multi-task and Adversarial Learning 14

4.2.2 Gradient Reversal Layer 16

4.2.3 Phone-invariant training 17

4.2.4 Probing for Phone Information 18

5 Experiments **19**

5.1 Experiment Setup 19

5.2 Datasets 19

5.2.1 Data Preparation 19

5.2.2 Evaluation Metric 20

5.2.3 Voice Activity Detector 20

5.2.4 Transcriptions Used 21
5.2.5 Baseline Model 22
5.2.6 Clustering 22
5.3 Multimodal Fusion 25
5.3.1 Results and Analysis 25
5.4 Phone-invariant Training 30
5.4.1 Results and Analysis 30

6 Conclusions **33**

References **34**

Appendix A Covid-19 Disruption **37**

Appendix B Risk Assessment Retrospection **38**

Chapter 1

Introduction

As of the time of this writing, most of the companies, schools and institutes across the world are operating online due to the impact of the worldwide pandemic, *Covid-19*. With all the meetings, classes and events going virtual, there has been an increasing opportunities and demands for the use of transcription software. In fact, the trend of moving communications online has appeared before the quarantine started. Online meetings have the advantage of being more flexible and often reduce financial and environmental costs. Massive open online courses (MOOCs) have made educational resources accessible to a wider range of population. The intensive use of computers for communications, combined with decreasing cost of processing power and storage capacity, has created a great amount of audio documents of meetings, lectures, podcasts *etc*.

Automatically generated transcriptions allow access to these audio documents through the additional modality of texts and thus save manual efforts in note-taking, allowing people to fully focus on the content while listening. Moreover, text transcripts make possible efficient indexing, searching and accessing, which is crucial for exploiting individual recording as well as navigating an entire archive of all audio documents.

Central to all transcription software is the automatic speech recognition (ASR) system that outputs the sequence of words being spoken. However, in multi-speaker scenarios such as meetings and classes, raw transcripts are often hard to read on their own. It helps a lot if they were complemented with additional meta-data such as speaker turns and sentence boundaries. Apart from making the transcripts more readable, these annotations would also be useful for information extraction tasks like summarisation or machine translation.

An audio document is defined as a recording that captures audio from multiple sources including different speakers, music as well as background noises, and diarisation is the task of marking and categorising the different audio sources within a audio document. Speaker diarisation, in particular, deals with separating out different speakers. Performing speaker diarisation facilitates indexing and searching by speaker as well as providing information that could be used by ASR systems to perform speaker adaptation.

Speaker diarisation systems typically consist of three stages: (1) speech segmentation, where non-speech segments are filtered out and the remaining speech activity are further divided to shorter segments(2) embedding extraction, where speaker distinctive features, referred to as

speaker embeddings, are extracted from acoustic features of segmented input audio (3) clustering, where the number of speakers is determined if not already known, and speaker embeddings extracted in the previous stage are clustered and assigned labels accordingly.

With recent development in deep learning, there has been a lot of research on using deep neural networks to extract speaker embeddings [1–3], often called *d-vectors* in this case, that have outperformed traditional techniques such as *i-vectors* [4–6]. While the performance of *i-vectors* asymptotes rather quickly with the increase of training data size, *d-vectors* can exploit large scale datasets much more effectively. Moreover, deep neural networks allows easier incorporation of different modalities, which is the focus of this project. Specifically, we are interested in how the content of speech could be exploited to better identify the speaker.

In this report, we will explore the use of transcriptions in addition to original audio input in extraction of speaker embeddings. We trained neural networks using graphemic and phonemic transcriptions as *(a)* training labels to perform *phone-invariant training* and *(b)* input features to create *multimodal fusion* model.

The rest of the report is organised as follows: chapter 2 provides an overview of the diarisation pipeline and motivates the use of transcriptions in diarisation, after which chapter 3 will describe in detail how neural networks are used to create discriminative speaker embeddings, accompanied with introductions to various options for each component in the model. Chapter 4 will compare different architectures for *multimodal fusion* and explain the mechanism of *phone-invariant training*, with relevant experiment setups and results recorded in chapter 5. Finally, the report is wrapped up with a summary of methods and findings in chapter 6.

Chapter 2

Background

2.1 Diarisation System

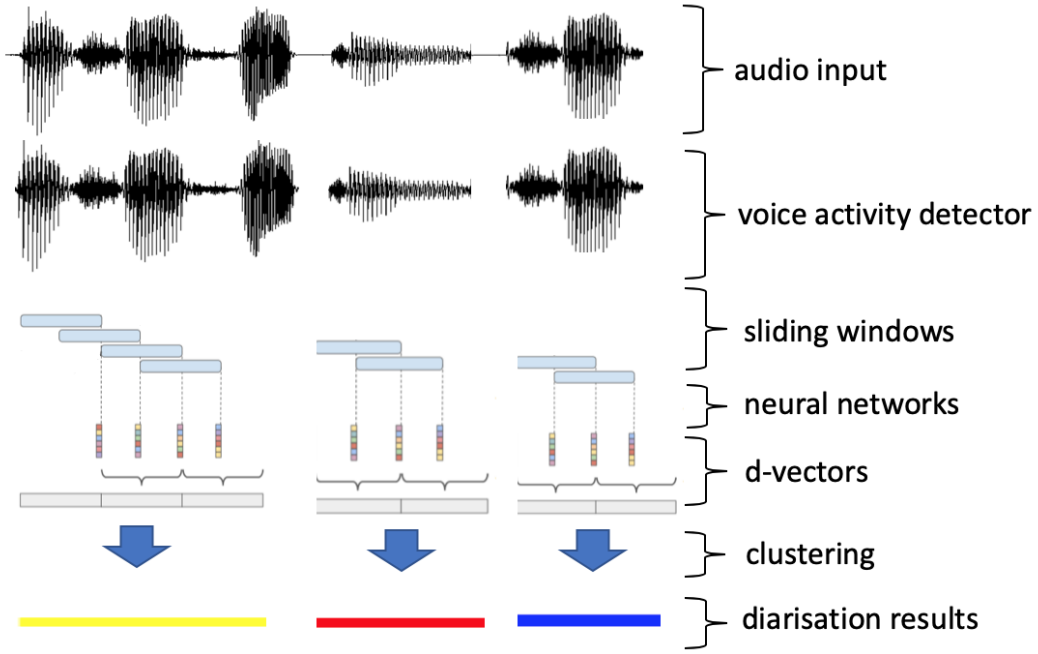


Fig. 2.1 Illustration of the speaker diarisation pipeline using DNN

2.1.1 Initial Segmentation

The pipeline starts with a voice activity detector (VAD) that removes silence or non-speech background noise from the input audio stream. With the diarisation error rate (DER) being the sum of missed speech (MS), false alarm (FA) and speaker error rate (SER), VAD accounts for the first two of these three components. Many research on diarisation has focused on the development of downstream stages by using manually segmented audio and only reporting the speaker error rate, which is also known as the oracle VAD error rate [4, 5, 7]. Nevertheless, VAD in itself has attracted a lot of research regarding which features [8] and model architectures [9] to use.

Traditional models apply Gaussian mixture models (GMMs) combined with Viterbi decoding to separate speech from non-speech while later methods explore using neural architectures combining the advantages of convolutional neural networks (CNN), long short-term memory (LSTM) recurrent neural networks and Deep Neural Networks (DNN) [9]. Note that in VAD there is often the trade-off between missed speech and false alarm since the former means losing information while the latter could create noise for clustering. Having obtained pure speech signals, a Speaker Change Detector (SCD) is sometimes used to split speech into speaker-homogeneous segments by computing a distance measure between two adjacent sliding windows. An alternative popular practice is to further divide speech into shorter fixed-length short partitions, which is the approach taken in this project.

2.1.2 Extraction of Speaker Embeddings

As in speaker verification, low-dimensional representations rich in speaker information, *speaker embeddings*, are extracted from speech signals for downstream processing. Traditionally, i-vectors obtained from factor analysis are used [10]. Motivated by the i-vector concept of encoding speaker characteristics of an utterance into a fixed-length vector, d-vectors trained with neural networks have gained popularity in more recent research, with further endeavor in utilizing attention mechanism to combine i-vectors and d-vectors [11]. In order to create more discriminative d-vectors, researchers have investigated various network architectures including LSTM [1] and TDNN [3], as well as different training criteria such as probabilistic linear discriminant analysis [12], triplet losses [13] and large margin angular softmax losses [14]. Compared to i-vectors, d-vectors have the advantages of being directly trained to discriminate speakers and doing better in exploiting the availability of large-scale training data [15].

2.1.3 Clustering

Clustering is arguably the most challenging stage in the pipeline due to it being inherently an unsupervised task. It is non-trivial to determine the number of speakers involved in the audio stream as well as to correctly identify overlapping speech, let alone correct assignment of each segment. Among common clustering algorithms used for diarisation, agglomerative Hierarchical Clustering (AHC) works by greedily merging small clusters until reaching a threshold [16]. Spectral clustering is an alternative method that utilises an affinity matrix to classify speaker identity. More recently, research efforts have been directed to develop supervised clustering algorithms [17, 18].

2.1.4 Optional Re-segmentation

Some researchers has taken an iterative approach that make use of the clustering results to refine the boundaries drawn in the initial segmentation before generating new speaker embeddings from re-segmented audio input [4, 7].

2.2 Motivations

Most traditional approaches for the extraction of speaker embeddings only make use of acoustic features while we think content information encapsulated in transcriptions could potentially provide additional cues of speaker identity in at least three different ways.

2.2.1 Capturing Idiosyncrasies in Pronunciation

Apart from vocal timbre, speakers mainly differ in pronunciation habits of certain phones. Therefore, we use multimodal fusion model in the hope to distill this correlation between acoustic and phone features. The details of the model and corresponding results are detailed in chapter 5.

2.2.2 Suppressing Phone Information

Partly inspired by the work done in [19] where speaker feature was suppressed and phone discriminability was maximised to improve ASR accuracy, we suggest doing the exact opposite in order to enhance diarisation performance. In other words, the variance in a pair of speaker embeddings could potentially be due to two different speakers pronouncing the same phone, or it could be resulted from one speaker pronouncing two different phones. We want to encourage the former type of variance while reducing the latter. This line of thought is supported by the theory of multi-task adversarial learning detailed in chapter 4 and with results presented in chapter 5.

2.2.3 Exploiting Idiosyncrasies in Word Choice

Apart from speaker-specific pronunciation habit, there is also idiosyncrasies in word choice and word frequency. Previously, there has been research that incorporates lexical and acoustic cues in the diarisation system using sequence-to-sequence neural network to emulate how humans employ information [20]. However, the system outputs speaker turns instead of speaker embeddings. As a result, the method applied was limited to two-speaker scenarios and could not generalise to audio documents involving more speakers. Moreover, the DER reported was substantially higher than that of state-of-the-art traditional systems such as in [17].

Alternative approaches to implement this idea include perform fusion for word embeddings and speaker embedding extracted from acoustic features. One challenge in pursuing this line of thought is that we want to exploit frequency of words to identify but also need to control the extent to which we rely on the content to determine the speaker. We did not go into this direction of research in our project.

Chapter 3

Diarisation with D-vectors

3.1 Neural Network Architecture

Creating a neural network for extracting d-vectors involves designing frame- and segment-level sub-networks, which are ofetn connected by either a statistics pooling layer or a self-attentive layer.

Say that a segment with T frames corresponds to the acoustic feature matrix $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_{T-1}, \mathbf{x}_T]$ of shape $T \times n_x$, where n_x is the input feature size, the frame-level transformation can be written as:

$$\mathbf{f}_t = G_{\mathrm{f}}(\mathbf{x}_t, \theta_{\mathrm{f}}) \tag{3.1}$$

where $\mathbf{f}_t$ denotes the frame-level embedding, $G_{\mathrm{f}}(\cdot)$ the feed-forward function and θ_{f} the parameters of the frame-level network.

Having obtained the segment-level representation through aggregating frame level features, $\mathbf{r} = \text{aggreagation}(\mathbf{f}_1, \mathbf{f}_2, ..., \mathbf{f}_T)$, it is then passed through additional hidden layers to give the speaker embedding s.

$$\mathbf{s} = G_{\mathrm{s}}(\mathbf{r}, \theta_s) \tag{3.2}$$

During training, a fully-connected layer is placed at the output to compute the posterior on speakers and optimise for the loss function of choice. The posterior of speaker i would be given as:

$$P(i|\mathbf{s}) = \text{softmax}\big(G_{\mathrm{o}}(\mathbf{s}, \theta_o)\big) \tag{3.3}$$

In the rest of this chapter, we will talk about the design and choices for $G_{\mathrm{f}}, G_{\mathrm{s}}$and aggregation$(\cdot)$.

Time delay neural network (TDNN) is among the popular choices for the frame-level model G_{f}. Sometimes seen as a precursor to the convolutional neural networks (CNN), TDNNs operate in a modular and incremental manner [22]. Whereas the initial layer in a DNN learns an affine transform for the entire input context, TDNN has an initial transform learnt from a narrower window. By combining and process features that are more specific to the local scale and are extracted by the lower layer, each layer in the TDNN gives a lower temporal resolution than the previous with each node covering a wider context. The transforms in the TDNN are tied across

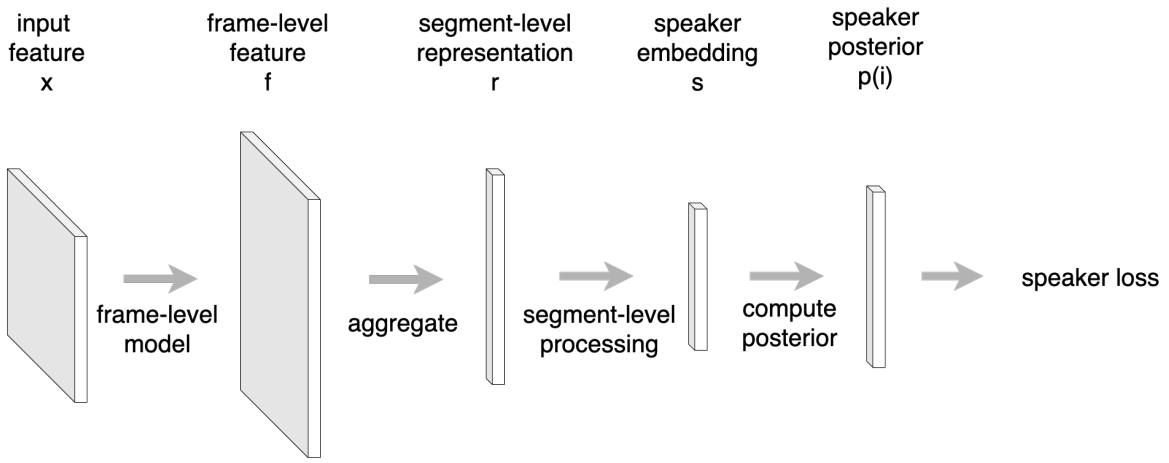


Fig. 3.1 Model flowchart of speaker embedding extractor

time steps within the same layer, analogous to the convolutional filter used in CNN. The tying operation allows translation invariance features to be learnt.

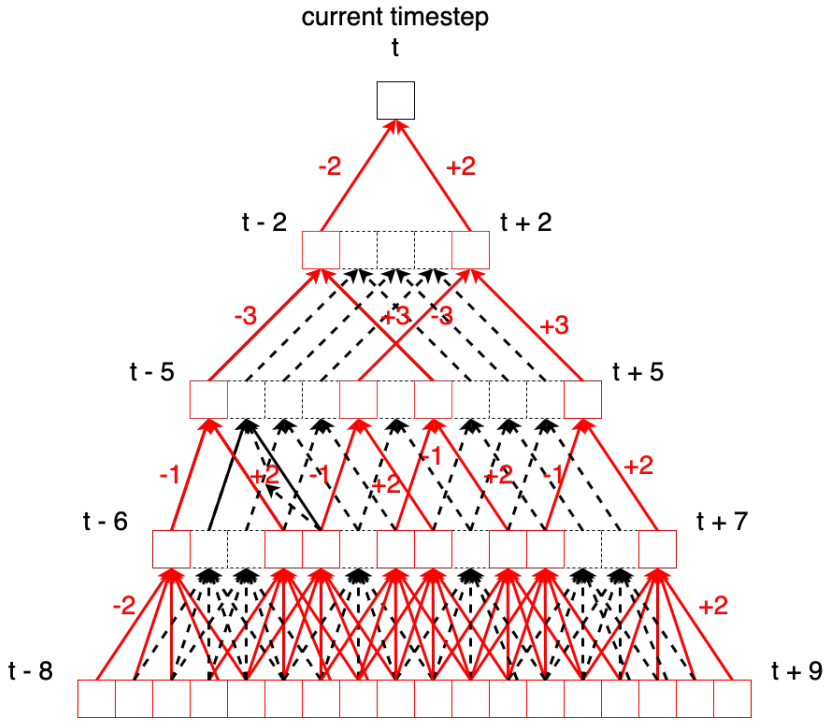


Fig. 3.2 Computation diagram of TDNN (red = TDNN with sub-sampling, red+black = TDNN without sub-sampling)

Recurrent neural network (RNN) is another common choice for modeling frame-level acoustic features. In RNNs, dependency between hidden vectors h across time steps is introduced to allow information to be preserved through time. At time step t, h_t is computed from the hidden vector at the previous time step h_{t-1} and the input x.

$$h_t = f(\mathbf{W}\mathbf{x}_t + \mathbf{U}\mathbf{h}_{t-1} + \mathbf{b}) \tag{3.4}$$

where $\mathbf{W}, \mathbf{U}$ are the weights and $f(\cdot)$ is the activation function.

When the activation $f(\cdot)$ is chosen to be *sigmoid function* $\sigma(x) = \frac{1}{1+e^{-x}}$, RNNs suffer from the problem of vanishing and exploding gradient. This issue was addressed by the long short-term memory (LSTM) model through the introduction of a *memory cell*. The operations performed could be defined as follows: The state of the memory cell at the current time-step, c_t, is interpolated from that of the previous time-step c_{t-1} and the new candiate state $\tilde{\mathbf{c}}_t$, controlled by the forget gate $\mathbf{f}_t$ and the input gate $\mathbf{i}_t$. Activated with the tanh function and then scaled by the output gate $\mathbf{o}_t$, $\mathbf{c}_t$ would give the hidden state $\mathbf{h}_t$.

A modified version of RNN, the *high order RNN* (HORNN) [23], has also being applied to extract frame-level features for speaker embeddings [11]. In HORNN, the gradient vanishing problem is avoided by relaxing the first order Markov conditional dependence constraint and creating high order connections that link earlier hidden vector, $\mathbf{h}_{t-n}(n > 1)$, to the current input, $\mathbf{x}_t$:

$$\mathbf{h}_t = f(\mathbf{W}\mathbf{x}_t + \mathbf{U}_1\mathbf{h}_{t-1} + \mathbf{U}_n\mathbf{h}_{t-n} + \mathbf{b}) \tag{3.5}$$

When using ReLU activation function, HORNN matches the performance of LSTM with similar performance in speech recognition while only using less than half of the layer parameters.

Training TDNNs generally requries less computing power than RNNs. Among aforementioned traditional RNN, LSTM and HORNN, only HORNN can be comparable to TDNN in terms of parameter size while maintaining similar capacity. Moreover, parellelisation can be more easily exploited to TDNN since due to the dependencies between time-frames within RNNs. Thus, we have chosen TDNN for our frame-level model G_{f}.

3.2 Aggregation Method

If the segment-level representation $\mathbf{r}$ is produced by a statistics pooling layer, then it would be the concatenation of the mean $\mathbf{m}$ and the standard deviation σ of the frame-level output $\mathbf{f}$:

$$\mathbf{r}^T = [\mathbf{m}^T, \sigma^T]b \tag{3.6}$$

$$\mathbf{m} = \frac{1}{T}\sum_{t=1}^{T}\mathbf{f}_t \tag{3.7}$$

$$\sigma = \Big(\frac{1}{T}\sum_{t=1}^{T}(\mathbf{f}_t - \mathbf{m})^2\Big)^{1/2} \tag{3.8}$$

Alternatively, if a self-attentive layer is used, an h-head annotation matrix $\mathbf{A}(T \times h)$ would be computed and used to project frame-level features $\mathbf{F} = [\mathbf{f}_1, \mathbf{f}_2, ..., \mathbf{f}_{T-1}, \mathbf{f}_T](T \times n_f)$ to segment-

level representation $\mathbf{R}(h \times n_f)$:

$$\mathbf{A} = \text{Softmax}\big(G_{\mathbf{A}}(\mathbf{F})\big) \tag{3.9}$$

$$\mathbf{R} = \mathbf{A}^T\mathbf{P} \tag{3.10}$$

The softmax operation is applied to ensure each column of the annotation matrix, referred to as annotation vectors, sums to one.

A penalty term J is often added to the loss function to encourage each head to extract different features:

$$J = \mu||\mathbf{A}^T\mathbf{A} - \mathbf{I}||_F^2 = \mu\big(\sum_{i=1}^{h}(\mathbf{a}_i^T\mathbf{a}_i - 1)^2 + \sum_{i,j,i\neq j}^{h}(\mathbf{a}_i^T\mathbf{a}_i)^2\big)$$

Here $\mathbf{a}_i$ denotes the $i-th$ annotation vector and $||\cdot||_F$ the Frobenious norm.

In [11], a modification to the penalty term was proposed to control the smoothness of the annotation vectors. Heads with 'spiky' annotation vectors tend to focus on a small number of frames within the segment window while heads with 'smooth' annotation vectors are aggregated from a wider range of frame-level features. The modified penalty term is given as:

$$J^{'} = \mu||\mathbf{A}^T\mathbf{A} - \Lambda||_F^2 \tag{3.11}$$

where Λ is a diagonal matrix with its diagonal elements, $\lambda_i = \Lambda_{ii}$, controlling the smoothness of the annotation vector of the i-th head. The annotation vector goes from smooth to spiky as the corresponding λ goes from zero to one.

Comparing the two aggregation methods, only statistics pooling allows encoding of variable-length segments into fixed-length d-vectors. Another major difference is that statistics pooling assigns equal weight to each frame-level feature while a self-attentive layer make possible more complicated manipulation of the frame-level features within the segment window, which is in turn informed by the features themselves. We prioritise the latter advantage over the former and hence have used the self-attentive layer in our systems.

3.3 Training Objective

In order to obtain embedding extractor that captures discriminative features and generalise well to unseen speakers, we want to maximise inter-class distance and intra-class compactness during training. Cross-entropy loss is most commonly used as the training objective.

Traditional cross-entropy loss is defined as:

$$L_{\text{CE}} = -\sum_{i=1}^{N}\sum_{k=1}^{M} y_k^i log\hat{y_k}^i \tag{3.12}$$

where N is the number of samples, M is the number of classes, y and $\hat{y}$ denote the label and the prediction respectively. In the context of speaker classification, it can be expressed as:

$$L_{\text{CE}} = -\sum_{i=1}^{N} \log \left(\frac{e^{\mathbf{s}_i^T \mathbf{w}_t}}{\sum_{k=1}^{M} e^{\mathbf{s}_i^T \mathbf{w}_k}} \right) \tag{3.13}$$

where $\mathbf{s}_i$ is the i-th speaker embedding, $\mathbf{w}_k$ is the weight vector associated with class k and t being the target class. Since the marginalisation term $\sum_{k=1}^{M} e^{\mathbf{s}_i^T \mathbf{w}_k}$ is constant for all w_j, we have

$$\hat{y}_j^i \propto e^{\mathbf{s}_i^T \mathbf{w}_j} \tag{3.14}$$

$$\hat{\mathbf{y}}^{\mathbf{i}} \propto e^{\mathbf{W}\mathbf{s}_i} \tag{3.15}$$

Each column within the weight matrix $\mathbf{W}$ of the last hidden layer corresponds to one speaker in the training set and the prediction value of a speaker embedding s_i belonging to speaker j is proportion to the dot product between s_i and w_j, which is also proportion to the cosine similarity between the two vectors.

Large-margin softmax (L-softmax) loss [24], which can be seen as a generalisation of the cross-entropy loss , was developed to further encourage the separation of different classes.

$$L_{\text{L-softmax}} = -\sum_{i=1}^{N} \log \left(\frac{e^{||\mathbf{s}_i|| \cdot ||\mathbf{w}_t|| \psi(\theta_t)}}{e^{||\mathbf{s}|| \cdot ||\mathbf{w}_t|| \psi(\theta_t) + \sum_{k \neq t}^{M} e^{\mathbf{s}_i \mathbf{w}_k}}} \right) \tag{3.16}$$

$$\psi(\theta_t) = (-1)^k \cos(m\theta_t) - 2k, \ \theta_t \in [\frac{k\pi}{m}, \frac{(k+1)\pi}{m}] \tag{3.17}$$

$\psi(\theta_t)$ is a monotically non-increasing function that satisfies $\cos(\theta_t) \geq \psi(\theta_t)$ for $\theta_t \in [0, \pi]$ and is often approximated with $\psi(\theta_t) \approx cos(m\theta_t)$ where $m > 1$. Being strictly smaller than the cosine distance, this function shrinks the similarity between the speaker embedding and the weight vector of the correct target class, making L-softmax harder to optimise for than regular cross-entropy, hence encouraging class separateness.

Outside the family of cross-entropy loss, triplet loss is another popular objective function used in speaker verification and diarisation [13]. Triplet loss works by first sampling an utterance triplets consisting of $\mathbf{x}_i^a$, an *anchor* utterance of a specific person, $\mathbf{x}_i^p$, a *positive* utterance of the same person, and $\mathbf{x}_i^n$, a *negative* utterance correponding to any other person. The aim of training is for all the d-vector triplets corresponding to these utterance triplets, $(\mathbf{s}_i^a, \mathbf{s}_i^p, \mathbf{s}_i^n)$, to behave as follows:

$$||\mathbf{s}_i^a - \mathbf{s}_i^p||_2^2 + \alpha < ||\mathbf{s}_i^a - \mathbf{s}_i^n||_2^2 \tag{3.18}$$

where α is a empirically chosen hyper-parameter that defines the margin between positive and negative pairs.

The triplet loss is thus formalised as:

$$L_{\text{triplet}} = \sum_i \left[||\mathbf{s}_i^a - \mathbf{s}_i^p||_2^2 - ||\mathbf{s}_i^a - \mathbf{s}_i^n||_2^2 + \alpha \right] \tag{3.19}$$

While triplet loss allows end-to-end training of speaker verification systems, this is not the case for diarisation. Also, the performance of triplet loss depends heavily on the sampling scheme and requires experimenting with different data mining strategy. In our experiments, we used the generic cross-entropy loss, which is equivalent to the angular softmax loss with m set to 1.

3.4 Clustering

The task of finding good clusters has been one of the major problems in the field of unsupervised machine learning. Here we introduce three clustering algorithms, placing special emphasis on the spectral clustering algorithm, which was applied in this project.

K-means is one of the simplest unsupervised generative clustering algorithms. Ofen coupled with the expectation-maximisation (EM) algorithm, k-means seeks to minimise the average squared distance within clusters. Using careful initialisation technique such as *k-means++* often significantly improves the speed and accuracy of k-means. As for determining the cluster number, it is common to apply the "elbow" method which uses the number that maximises the mean squared cosine distancs between each embedding to its cluster centroid.

Spectral clustering is known to handle complex and unknown cluster shapes where k-means and mixture modeling methods struggle. Instead of creating an explicit model to estimate the distribution of the raw data, spectral clustering works by analysing the eigen-structure of an affinity matrix.

When constructing the affinity matrix $\mathbf{A} \in n \times n$, where n is the number of segments within the segmented audio in our context of diarisation, each element A_{ij} is computed as the cosine similarity between the embeddings, $\mathbf{s}_i$ and $\mathbf{s}_j$, corresponding to the i-th and j-th segments. Choosing cosine similarity as the measure for similarity nicely aligns with the training objective of angular softmax which seeks to maximise cosine similarity within classes. Having obtained the affinity matrix, a sequence of refinement operations can be applied to denoise affinity matrix [1]:

1. Apply a Gaussian blur with standard deviation σ to smooth the input data.

2. Perform row-wise thresholding by setting to zero elements smaller than its row's p-percentile. This helps to reduce the effect of noise coming from embedding pairs that are unlikely to be from the same speaker.

3. Symmetrise rows and columns of $\mathbf{A}$ ($Y_{ij} = \max(X_{ij}, X_{ji})$) to restore the inherent symmetry in affinity values.

4. Use outer product diffusion ($\mathbf{Y} = \mathbf{X}\mathbf{X}^T$) to produce clearer section boundaries.

5. Perform row-wise normalisation ($Y_{ij} = X_{ij}/\max_k X_{ik}$) to rescale the spectrum of the matrix.

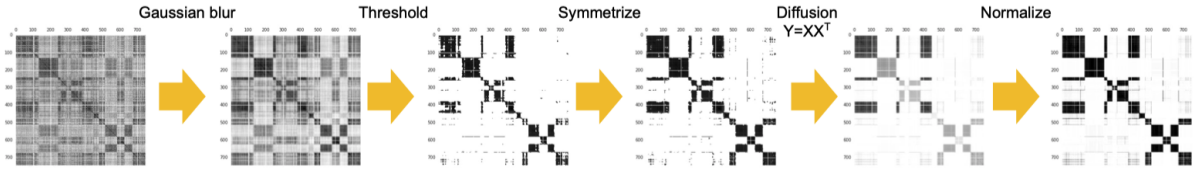


Fig. 3.3 Refinement operations of the affinity matrix, illustration taken from [1]

Having refined the affinity matrix, we perform eigen-decomposition and use the maximal eigen-gap to determine the speaker number $\tilde{k}$:

$$\tilde{k} = \arg\max_k \frac{\lambda_k}{\lambda_{k+1}} \tag{3.20}$$

where $\Lambda = [\lambda_1, \lambda_2, ..., \lambda_n]$ is arranged in a descending order. An alternative way of estimating $\tilde{k}$, which was not adopted in our project, is to fit a smooth exponential $exp(-\alpha k)$ to Λ and take $\tilde{k}$ to be the smallest value to make the derivative $-\alpha exp(-\alpha k)$ exceeds an empirically estimated threshold θ. This approach is based on the observation that elements in Λ exhibits a trend of exponential decay and that the number of speakers in each audio stream often correponds to when the gradient of the affinity eigenvalues exceeds a certain threshold. [25].

Having determined the number of clusters, we then take the the largest $\tilde{k}$ eigenvectors to form the matrix $\mathbf{V} = [\mathbf{v}_1, ..., \mathbf{v}_{\tilde{k}}] \in \mathbb{R}^{n \times \tilde{k}}$. Treating each row of $\mathbf{V}$ as a vector in $\mathbb{R}^{\tilde{k}}$, we cluster them into $\tilde{k}$ clusters via k-means. Finally, we assign the original embedding $\mathbf{s}_i$ to cluster m if and only if row i in $\mathbf{V}$ was assigned to cluster m.

To obtain an intuition of the mechanism of spectral clustering, let us consider the ideal case of K clusters where embedding pairs from the same cluster has cosine similarity of 1, and those from different classes 0. This would result in a block-diagonal affinity matrix $\tilde{\mathbf{A}}$ and its eigenvalues and eigenvectors would be the union of the eigenvalues and eigenvectors of its blocks, with the latter padded appropriately with zeros. Also, K of the eigenvalues would be equal to 1 with their correponding eigenvector spanning the correponding block in $\tilde{A}$. As a result, if the i-th embedding belongs to the cluster m, the m-th element of the i-th row of $\tilde{V}$ would be equal to one and all other elements in the row would be zero.

Apart from the two unsupervised clustering algorithms mentioned above, supervised clustering techniques applied specifically to diarisation were also developed in the hope of creating end-to-end diarisation systems. Among them, the discriminative neural clustering (DNC) method [18] proposed using the encode-decoder transformer architecture to produce a sequence of speaker labels. Compared to spectral clustering, DNC is able to identify the number of speakers more accurately and proves to be able to capture more complex class boundaries. However,

the system is limited to sessions with less than or equal to four speakers and also requires data augmentation techniques to mitigate data sparsity.

Chapter 4

Incorporation of Content Information

4.1 Multimodal Fusion

In recent years, multimodal research has drawn considerable attention in the field of machine learning. As one of the main topics in multimodal research, multimodal fusion is aimed at combining multiple modalities to exploit the complementarity of heterogeneous data and provide more robust predictions.

The models used for multimodal fusion can be split into three main categories. Early fusion, which was the approach we have taken in this project, concatenates input-level features to generate a joint representation for feeding into a joint model. However, this method is sensitive to the scale and dimension of the input features and would also potentially leave out context and temporal dependencies within each modality [26]. Late fusion, on the other hand, uses separate models for each modality and combines the output using majority voting [27] or weighted averaging [28]. Consequently, this would often allow less inter-modal dynamics to be captured. To address these issues, intermediate approaches using tensors were proposed to balance the trade-off between modeling intra-modal and inter-modal interactions. One method uses bilinear pooling to exploit the interactions between visual and textual cues in the task of visual question answering [29]. Bilinear pooling computes the outer product of two input feature vectors, allowing a multiplicative interaction between all elements of both vectors. Despite its superior expressiveness, bilinear pooling suffers from significant computational complexity due to its high dimensionality, which could be alleviated by applying low-rank tensor techniques [26].

4.2 Phone-invariant Training Via Adversarial Learning

4.2.1 Multi-task and Adversarial Learning

In training neural networks, we typically focus on optimising for one single objective. While this generally achieves acceptable performance, there are often training signals that can help us do better on the metric we originally care about. Our model can be trained to generalise better on our original task through sharing representations between related auxiliary tasks. There are

several hypotheses of why multi-task learning (MTL) works [30]. One explanation is that MTL leads to implicit data augmentation. Since there are different noise patterns within different tasks, learning multiple tasks simultaneously allows the model to obtain a better representation that does not overfit to one particular type of noise pattern. Another hypothesis is that the model is allowed to "eavesdrop" through MTL, with some features easier to learn for certain tasks.

In the context of deep learning, multi-task learning is generally implemented with either hard of soft parameter sharing of the hidden layers. In hard parameter sharing, classifiers for selected tasks share the same lower layers. In other words, a low-level representation is learnt with supervision from all selected tasks and task-specific classifiers are applied on this representation at the output. Soft parameter sharing, implied by its name, encourages similar parameter between models for selected tasks through regularising the distance between the weights of the hidden layers.

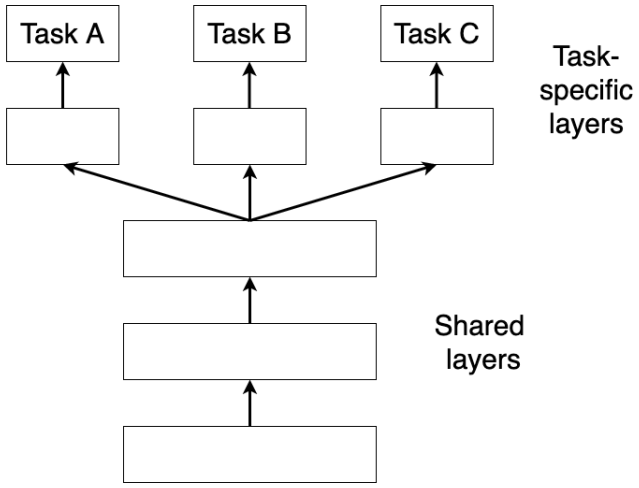


Fig. 4.1 Hard parameter sharing for DNN

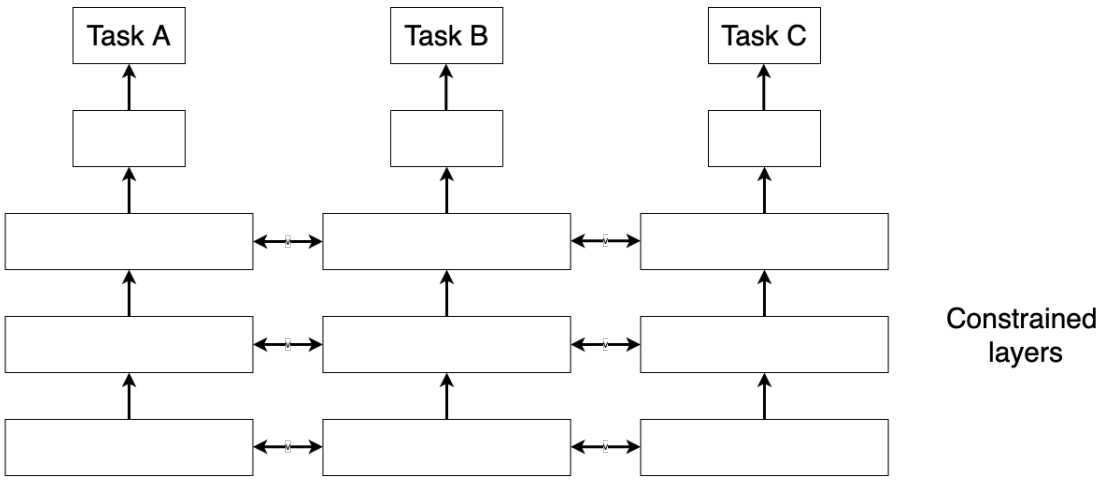


Fig. 4.2 Soft parameter sharing for DNN

Adversarial learning is a special case of MTL that seeks to maximise the confusion in the auxiliary task and has been applied to tackle domain adaptation. In the context of deep learning,

the main goal of adversarial learning is to train the neural network to make final classification decisions on features that are discriminative for the main task and at the same time invariant to the change of task-irrelevant properties. This could be embedded into the traditional process of learning representation through the use of a *gradient reversal layer*[31], which proved to work for domain adaptation.

4.2.2 Gradient Reversal Layer

To formalise the DNN architecture that utilises a gradient reversal layer, we could divide it into three sub-networks. With each input $\mathbf{x}$ corresponding to a main task label $y \in Y$ and an auxiliary task label $d \in D$, a *feature extractor* G_f is first used to transform the input to a feature vector $\mathbf{f}$, i.e. $\mathbf{f} = G_f(\mathbf{x};\theta_f)$. Then, the feature vector $\mathbf{f}$ is mapped to $\hat{y} \in Y$ and $\hat{d} \in D$ by classifier G_y with parameters θ_y and classifier G_d with parameters G_d respectively. While an input feature x_i is determined by both of the property of interest, y_i, and the auxiliary property, d_i, we are interested in extracting features $\mathbf{f}$ that are discriminative in the former but invariant to the latter. In other words, we want $\mathbf{G}_f(\mathbf{x};\theta_f|y = Y_i, d = D_j)$ and $\mathbf{G}_f(\mathbf{x};\theta_f|y = Y_i, d = D_k)$ to be similar if for all (i, j, k). However, it is not trivial to measure the similarity between the two, which is why we try to use the loss of the classifier G_d, L_d, as a proxy for their dissimilarity, provided that θ_d has been trained to discriminate the auxiliary property in an optimal way. In other words, we need to ensure that when L_d is maximised, information that is discriminative for d was not lost at all in G_d, but indeed in G_f, thereby creating features $\mathbf{f}$ invariant to d.

Therefore, at training time, we seek parameters θ_f that maximise L_d and at the same time seeking parameters θ_d that minimise L_d while simultaneously minimising the loss of the main task classifier, L_y. This objective can be formalised as follows.

$$E(\theta_f, \theta_y, \theta_d) = \sum_{i=1}^{n} L_y(G_y(G_f(\mathbf{x}_i;\theta_f);\theta_y)) - \lambda \sum_{i=1}^{n} L_d(G_d(G_f(\mathbf{x}_i;\theta_f);\theta_d)) \tag{4.1}$$

$$= \sum_{i=1}^{n} L_y^i(\theta_f, \theta_y) - L_d^i(\theta_f, \theta_d) \tag{4.2}$$

$$(\hat{\theta}_f, \hat{\theta}_y) = arg \min_{\theta_f, \theta_y} E(\theta_f, \theta_y, \hat{\theta}_d) \tag{4.3}$$

$$\hat{\theta}_d = arg \max_{\theta_d} E(\hat{\theta}_f, \hat{\theta}_y, \theta_d) \tag{4.4}$$

where L_y^i and L_d^i are the corresponding loss function evaluated for the i-th training sample and λ is the hyperparameter controlling the trade-off of the two objectives, referred to as the *reversal coefficient*. In order to find the saddle point that satisfies the above conditions, the following

update rules were proposed:

$$\theta_f \longleftarrow \theta_f - \mu\left(\frac{\partial L_y^i}{\partial \theta_f} - \lambda \frac{\partial L_d^i}{\partial \theta_f}\right) \tag{4.5}$$

$$\theta_y \longleftarrow \theta_y - \mu \frac{\partial L_y^i}{\partial \theta_y} \tag{4.6}$$

$$\theta_d \longleftarrow \theta_d - \mu \frac{\partial L_d^i}{\partial \theta_d} \tag{4.7}$$

where μ is the learning rate.

These updates can be made compatible with SGD with a *gradient reversal layer* (GRL), which has no weights associated with it as normal hidden layers do. In fact, it act as an identity transform in forward propagation. However, during backward propagation, it takes the gradient back-propagated from the immediate higher layer and multiplies it by the gradient reversal parameter λ. The GRL can be defined mathematically as a "pseudo-function" $R_\lambda(\mathbf{x})$ with forward propagation behaviour:

$$R_\lambda(\mathbf{x}) = \mathbf{x} \tag{4.8}$$

and backward propagation behaviour:

$$\frac{dR_\lambda(\mathbf{x})}{d\mathbf{x}} = -\lambda \mathbf{I} \tag{4.9}$$

This GRL is placed between the auxiliary classifier G_d and the feature extractor G_f. When the back-propagated gradient passes through the GRL, it is multiplied by $-\lambda$ and thus $\frac{\partial L_d^i}{\partial \theta_f}$ is replaced with $-\frac{\partial L_d^i}{\partial \theta_f}$ and therefore the training objective being optimised by SGD could be written as:

$$E^{'}(\theta_f, \theta_y, \theta_d) = \sum_{i=1}^{n}\left[L_y(G_y(G_f(\mathbf{x}_i;\theta_f);\theta_y)) + L_d(G_d(R_\lambda(G_f(\mathbf{x}_i;\theta_f);\theta_d)))\right] \tag{4.10}$$

4.2.3 Phone-invariant training

Among adversarial learning applications, phone-invariant training poses the unique challenge of level mismatch between the extraction of speaker and phone information [32]. Whereas speaker embeddings are extracted on a segment level after aggregating hundreds of frame-level sub-features, each phone label generally covers a much smaller context of several frames and is therefore classified on the frame level. Expressed in our previous formulation, G_y takes an entire segment as input and thus L_y is computed for each segment:

$$L_y^{[i-w/2,i+w/2]} = L_y(G_y([G_f(\mathbf{x}_{i-w/2};\theta_f),...,G_f(\mathbf{x}_i;\theta_f),...,G_f(\mathbf{x}_{i+w/2};\theta_f)]), y_i) \tag{4.11}$$

where w is the number of frames in each segment. and L_d is evaluated on a frame-by-frame basis:

$$L_d^i = L_d(G_y(G_f(\mathbf{x}_i;\theta_f)), d_i) \tag{4.12}$$

One way of addressing this issue would be to introduce a hybrid training method. For instance, [32] proposed a hybrid multi-task learning framework that back-propagates gradients from the speaker classification loss and the phone classification loss alternately. Another potential solution would be to introduce a hybrid loss function which requires designing a measure of similarity between two phone sequences.

In our experiments, we have chosen to perform phone-invariant pre-training at the frame level before removing the phone classifier and fine-tuning the segment-level model, which could be implemented by the readily available framework.

4.2.4 Probing for Phone Information

To evaluate the phone discriminability of speaker embeddings after phone-invariant training, we have tried probing the phone content in the speaker embedding using a phone classifier [33]. The classifier is a multi-layer perceptron with only one hidden layer activated with ReLu. This classifier takes the speaker embedding to be probed as input and is trained to perform the task of phone prediction. We would expect embeddings with a greater amount of phone information would give high classification accuracy and vice versa.

Chapter 5

Experiments

5.1 Experiment Setup

5.2 Datasets

There are various datasets available for training and evaluating diarisation systems, covering multiple domains (telephone conversation, recorded meetings, broadcast) and languages. To name a few, CALLHOME[34], the AMI meeting corpus[35], 2000 NIST SRE and 2003 NIST RTED are all common datasets used by researchers in the field. Apart from domain and languages, these data sets also differ in the number of channels and the number of speakers. The diversity in the choices for datasets makes it hard to compare the performance of different diarisation systems trained on different dataset. Moreover, there is also concern that individual system could overfit to particular characteristics of the data set on which it has been trained and evaluated on, and thus fail to generalise. In this work, we have focused on the AMI Meeting dataset for both training and evaluation.

5.2.1 Data Preparation

The AMI meeting corpus is dedicated to research aiming at improving group meeting effectiveness by creating better access to meeting history. The corpus consists of 100 hours of recorded meetings with each meeting involving 3 to 5 participants. In partitioning the training, test and evaluation sets, we have adopted the Full-corpus partition detailed on the AMI website. Apart from two speakers that occurr in both training and development sets, all speakers only occurr in one of the sets. In our experiments, 10% of each meeting in the training set was put together to form the cross-validation set for hyper-parameter tuning while the rest 90% was solely for model training.

40-d log-mel filter bank features (25ms-long frame, 10ms-long step) were extracted from the audio stream of the Multiple Distance Microphone (MDM) data before being passed through beam-forming [36] and then normalised to give the acoustic features that were fed into the speaker embedding extractor.

set	hours of speech	# of meetings	# of speakers
Training	70.66	135	155
Development	7.65	18	21 (2 in training)
Evaluation	7.33	16	16 (0 in training)

Table 5.1 Meeting and speaker statistics of each set

5.2.2 Evaluation Metric

Speaker diarisation systems are evaluated on its diarisation error rate (DER), which is the sum of false alarm (FA), missed speech(MS) and speaker error rate (SER). FA and MS are the proportion of false alarm and missed speech in total meeting duration. Most diarisation research has treated voice activity detector (VAD) as a separate component and only focuses on the extraction and clustering of speaker embeddings by assuming oracle VAD. In that case, DER would be equal to SER, the fraction of speaker time that is not correctly attributed to that specific speaker, which is the primary metric for diarisation. In fact, the computation of SER is much more complicated than those of MS and FA since it involves a mapping between speaker labels in the diarisation output and the speaker identities in the reference label file.

In our experiments, we evaluated DER obtained with both oracle VAD and real non-ideal VAD. Also, we followed the common practice of ignoring overlapping speech and allowing a forgiveness collar of 250 milliseconds at each speaker transition during evaluation.

The diarisation task is also defined by the amount of prior knowledge allowed. For instance, sometimes there would be example speech from the speakers present in the audio. In this project, we were concerned with the knowledge about the number of speakers involved. One reason for our consideration is that, since the clustering algorithm is usually very noisy, using oracle speaker number would allow us to better isolate the effect of the changes we have made to the d-vector extractor. We would be presenting results obtained with and without oracle speaker number in later experiments.

5.2.3 Voice Activity Detector

The acoustic model of our VAD is a feed forward neural network that utilises an input context of 55 frames to compute the emission probability for speech (S) and non-speech (NS) of the central frame. We originally used the state network shown in Figure 5.1 for decoding, yielding a MS rate of around 10%. To tackle the short silences in the VAD output, we created a new state network with additional non-speech sub-states. Moreover, as a post-processing step, we toggled the state of non-speech segments shorter than 100 frames which further reduces MS with a small sacrifice in FA. The last adaption we made was to penalise the state changes shown in red. These operations reduced the MS rate to below 5%.

	development		evaluation	
	MS	FA	MA	FA
original state network	9.80	1.40	11.70	2.20
adapted state network + threshold + penalty	3.60	1.70	4.70	2.10

Table 5.2 Missed speech (MS) and false alarm (FA) rate of the VAD

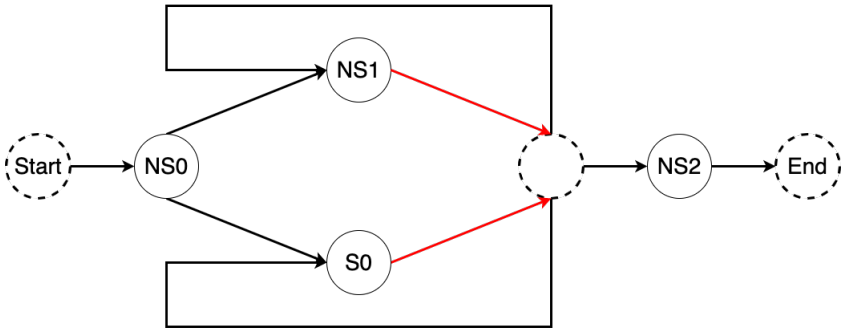


Fig. 5.1 Original state network (the dashed circles are *null nodes* that are useful for reducing the number of arcs required)

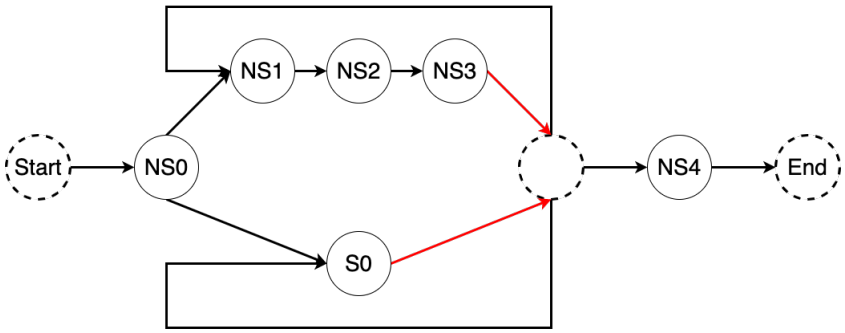


Fig. 5.2 Adapted state network

Note that the speech segments filtered by VAD was partitioned into 2s-long segments with 1s overlap before extracting d-vectors from each segment.

5.2.4 Transcriptions Used

We experimented with both reference and hypothesis transcriptions of phoneme and grapheme when exploring multimodal fusion, but have only used reference phonemic transcriptions in phone-invariant training. For both grapheme and phoneme, two sets of hypothesis transcriptions were tested in our experiments. The systems used to obtain them differ in the choice of language model, which were used to differiate them when presenting the results below.

The hypothesis transcriptions of each type were provided by Dr. Chao Zhang, whose description of the system utilised to perform ASR and alignment was as follows:

"The hybrid hidden Markov models (HMMs) used for speech recognition and alignment were with TDNN with residual connections (ResTDNN) [37] constructed using HTK [38] based on the 80 hour AMI MDM dataset. The phonetic and graphemic acoustic models have 5,951 and 6,848 tied triphone states respectively generated by the decision-tree-based state tying approach [39], which are used as the output units of the ResTDNN models. All hidden layers are 1,000-dimensional and with sigmoid activation function. The ResTDNN-HMMs were first trained with cross-entropy training and then refined by minimum phone error training [40]."

set	# of utterances
training	107311
development	13098
evaluation	12643

Table 5.3 Number of utterances in each set

transcript type		# of utterances failed to be recognised or aligned		
		training	development	evaluation
phoneme	reference	2	0	1
	tri-gram	1725	0	104
	four-gram	1725	0	90
grapheme	reference	1	100	80
	tri-gram	3000	7	3
	four-gram	3000	7	3

Table 5.4 Number of utterances failed to be recognised or aligned in each set

5.2.5 Baseline Model

The frame-level time delay neural network (TDNN) [41], took in a context of 15 frames and had dimensions shown in Figure 5.6. Segment representations were aggregated with a multi-head self-attentive layer from fame-level features yielded by TDNN, covering a context of 200 frames (2 seconds). The diagonal elements of the matrix Λ in the modified penalty term described in Chapter 3 were set to $[1, 1, 0.2, 0.2, 0.01]$. The self-attentive layer was followed by a bottleneck layer before giving the speaker embedding, as illustrated in Figure 5.4. The objective function being optimised for during training was the conventional cross-entropy loss. During training, we first performed pretraining on the frame-level model before finetuning the segment-level model.

5.2.6 Clustering

We applied spectral clustering, including the refinement steps, as detailed in section 3.4. The standard deviation of the Gaussian blur was fixed to 0.2 while the percentile used in row-wise

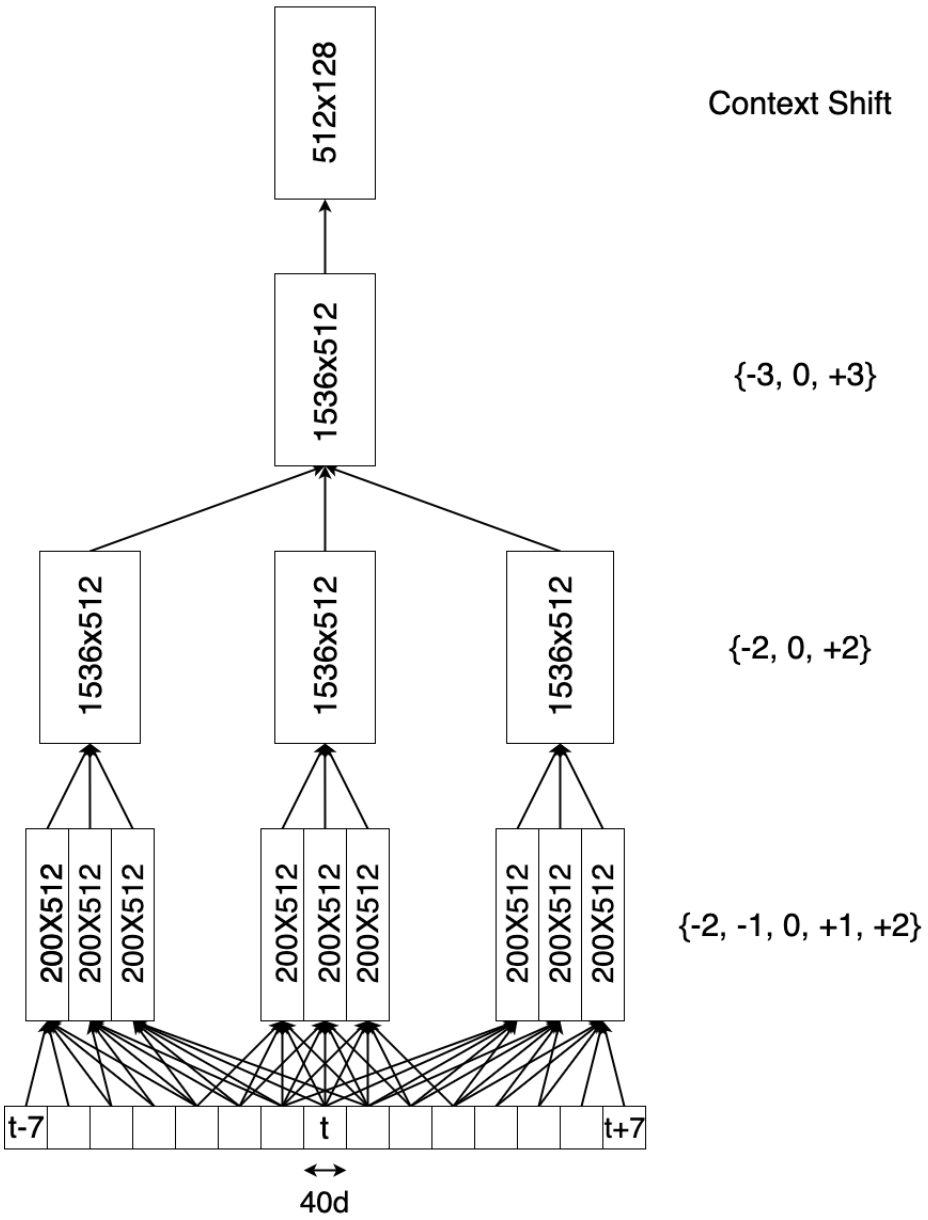


Fig. 5.3 Baseline frame-level TDNN model with input x output dimension written in each block

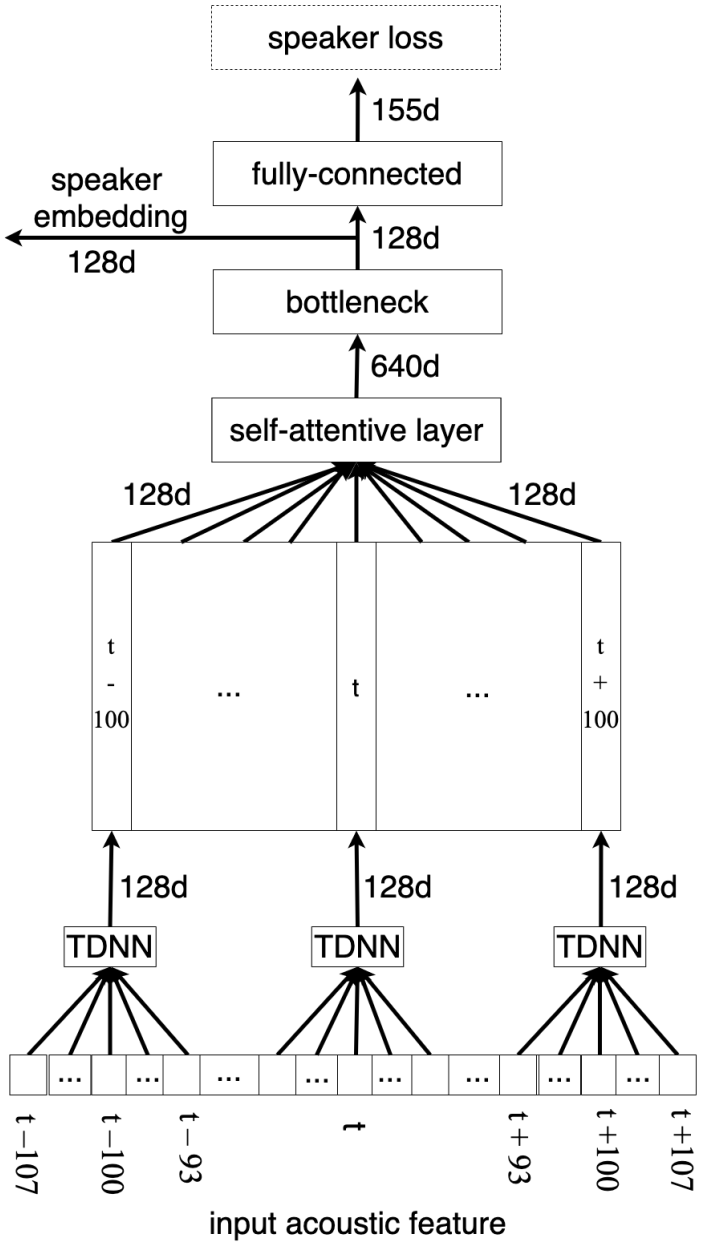


Fig. 5.4 Segment-level model with dimension of output vector written next to outgoing arrow

thresholding was tuned on the development set before being applied to the evaluation set. The number of clusters in each meeting is chosen between the range of 2 to 9.

When working with oracle VAD segmentation, we assumed each initial long segment, which were partitioned into fixed-length segments each correponding to one speaker embedding, only included one speaker. Therefore we post-processed the class assignment from spectral clustering by computing the mean vector of speaker embeddings each long segment and assigning new labels according to the distances of the mean embedding with cluster centroids obtained in spectral clustering. In the case of real VAD segmentation, we discarded this assumption and preserved the original labels.

5.3 Multimodal Fusion

Phoneme and grapheme features were obtained from the respective transcripts via one-hot encoding. There are 47 types of phonemes and 26 types of graphemes. Assigning an additional one-hot vector to *silence* results in the feature dimension of 48 for phoneme and 27 for grapheme. As for the utterances where the phoneme or grapheme sequences are either not recognised by the ASR system or could not be aligned with timeframes, we tried two ways of filling these 'blanks', one was to use an all-zero encoding and the other to use the encoding corresponding to silence.

In an *early fusion* architecture, the content features were passed through a fully-connected layer to give the content embeddings, which were then concatenated to the acoutic features for extracting frame-level features. The transform from content encoding to embedding was learnt simultaneously with the rest of the embedding extractor. The only difference between the baseline model and the early fusion model lied in the frame-level extractor.

An alternative strcuture for incorporating content uses *late fusion* by concatenating the content embedding to the frame-level embedding distilled from the acoustic features before the aggregation operation. The *late fusion* model shared the same frame-level extractor with the baseline and introduced content features at the input to the self-attentive layer.

5.3.1 Results and Analysis

In a preliminary experiment, the early and late fusion models gave the classification accruacies during training as shown in Table 5.5. Early fusion achieved higher classification accuracies during training, presumably because late fusion failed to exploit the dynamics between acoustic and phoneme features to a greater extent than the early fusion model's insufficiency in capturing intra-modal correlations.

In later experiments, we discovered that overfitting could be alleviated and DER reduced when we reduced the dimension of the hidden layers in the frame-level TDNN from 512 to 256 and imposed stronger regularisation by increasing weight decay from 0.001 to 0.005. Under this new setting, we compared the effect of using all-zero vector and silence encoding for utterances failed to be recognised or aligned. Note that in this experiment, we used oracle

model	training accuracy	cross-validation accuracy
baseline	86.09	81.58
early fusion	86.59	82.60
intermediate fusion	82.95	79.19

Table 5.5 Classification accuracies of fusion models using reference phoneme transcription

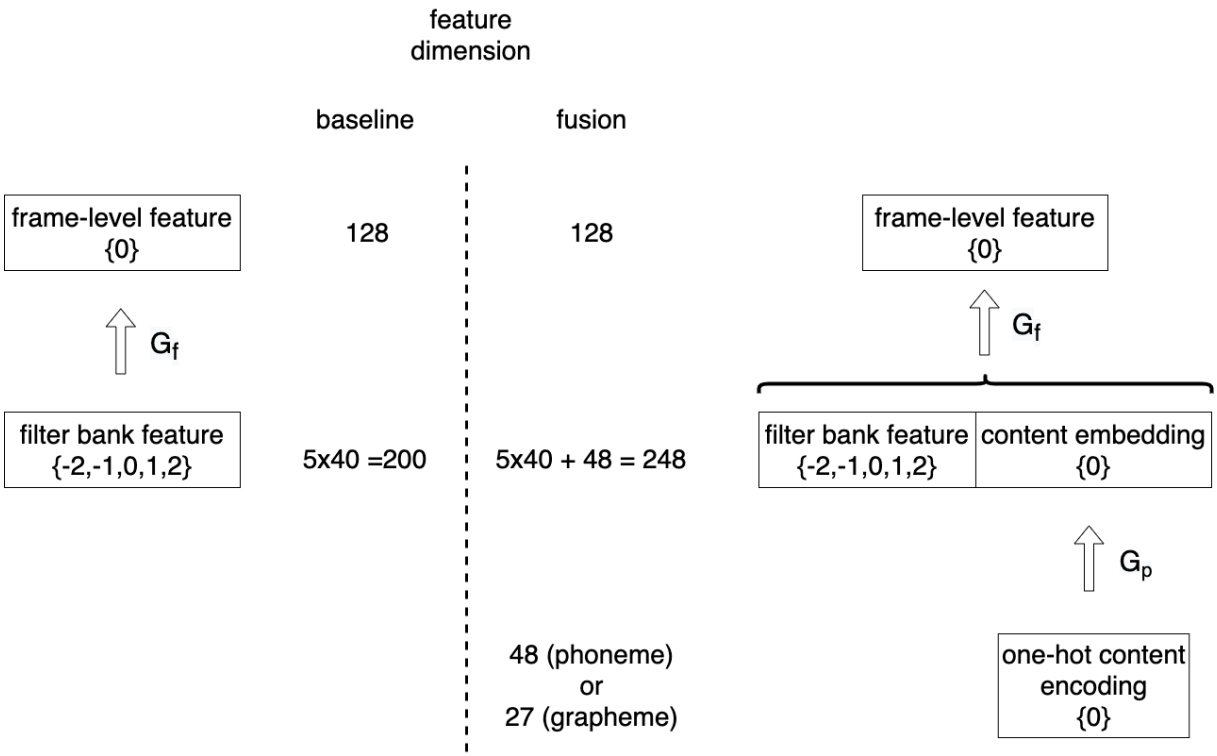


Fig. 5.5 Comparison of baseline and early fusion model on the frame level

VAD segmentation, oracle speaker number, and reference transcriptions for both training and evaluation.

filler used	grapheme		phoneme	
	dev	eval	dev	eval
all-zero vector	11.16	11.33	13.71	10.98
silence encoding	13.11	12.72	15.30	12.00

Table 5.6 Comparison of SER of systems with different fillers

Using silence encoding instead of all-zero vectors to fill utterances where alignment was unavailable would lead to an increase of more than one point in the DER across different sets of grapheme and phoneme systems. Hence we have sticked to early fusion systems using all-zero vector as filler in all later experiments.

As shown in Table 5.7, fusion models increase the classification accuracy by as much as 0.5% for the training set and 0.75% for the cross-validation set. Systems using grapheme transcripts

has consistently given better classification accracies than those using phoneme. Transcripts obtained with four-gram language model yielded the lowest accuracies for both phoneme and grapheme despite that four-gram language model was stronger tri-gram.

transcript		training accuracy	cross-validation accuracy
none (baseline)		79.20	76.67
phoneme	reference	79.34	76.72
	tri-gram	79.47	77.27
	four-gram	78.88	76.61
grapheme	reference	**79.70**	77.20
	tri-gram	79.53	**77.42**
	four-gram	79.29	76.90

Table 5.7 Classification accuracies of the early fusion system

transcription type	transcription used during training	transcription used during evaluation	oracle VAD		real VAD	
			oracle speaker #	unknown speaker #	oracle speaker #	unknown speaker #
phoneme	reference	reference	2.07	14.50	10.96	8.54
		tri-gram	3.38	17.63	10.98	9.70
		four-gram	3.20	17.47	7.48	10.84
	tri-gram	tri-gram	-1.04	7.65	10.58	2.50
	four-gram	four-gram	0.61	7.94	5.76	1.29
grapheme	reference	reference	9.14	8.57	20.23	9.32
		tri-gram	7.80	12.10	21.61	11.88
		four-gram	8.56	8.92	20.27	12.93
	tri-gram	tri-gram	3.36	3.99	20.98	8.47
	four-gram	four-gram	1.06	7.62	16.73	5.41

Table 5.8 Percentage in reduction of the SER using baseline model as the reference

While the raw speaker error rate results were presented in Tables 5.9 and 5.10, Table 5.8 below showed the relative reduction in SER. From these results, we have made comparison for three pairs of settings of the transcriptions used.

Grapheme versus Phoneme

We could see that the fusion system trained on reference grapheme transcriptions performed the best which consistently reduced the SER by more than 7% of the baseline and the edge was more pronounced when the real VAD was used. Apart from its superiority in performance, grapheme

transcription type	transcription used during training	transcription used during evaluation	dev		eval	
			oracle speaker #	unknown speaker #	oracle speaker #	unknown speaker #
baseline	none	none	14.66	14.61	10.73	19.88
phoneme	reference	reference	13.71	**12.70**	10.98	16.70
		tri-gram	13.78	13.08	10.65	**14.99**
		four-gram	13.75	13.07	10.71	15.03
	tri-gram	tri-gram	16.51	14.28	11.60	15.66
	four-gram	four-gram	13.21	13.82	11.66	17.80
grapheme	reference	reference	**11.16**	13.93	11.33	17.40
		tri-gram	11.32	13.78	11.50	16.20
		four-gram	11.31	15.37	11.34	15.30
	tri-gram	tri-gram	12.20	13.49	11.81	19.82
	four-gram	four-gram	11.82	14.67	**10.54**	16.77

Table 5.9 Speaker error rates of the early fusion system trained with various types of transcriptions and **oracle VAD segmentation**

transcription type	transcription used during training	transcription used during evaluation	dev		eval	
			oracle speaker #	unknown speaker #	oracle speaker #	unknown speaker #
baseline	none	none	23.29	20.00	21.32	20.47
phoneme	reference	reference	21.52	17.06	18.27	21.73
		tri-gram	21.53	16.40	20.31	22.17
		four-gram	21.24	16.61	20.01	19.50
	tri-gram	tri-gram	22.01	17.85	17.98	21.64
	four-gram	four-gram	22.97	19.04	19.16	21.98
grapheme	reference	reference	17.21	16.81	18.26	19.92
		tri-gram	16.47	16.79	18.35	**18.89**
		four-gram	**16.24**	15.98	19.13	19.29
	tri-gram	tri-gram	18.02	**15.81**	**17.20**	21.29
	four-gram	four-gram	17.89	18.99	19.13	19.29

Table 5.10 Speaker error rates of the early fusion system trained with various types of transcriptions and **real VAD segmentation**

transcriptions can also be more easily obtained when textual reference transcripts are available, whereas mapping text to phoneme would require using a dictionary.

Reference versus Hypothesis

If originally trained on the reference transcriptions, using hypothesis transcriptions during evaluation would cause reasonably small variation in the SER. This is more obvious in Tables 5.9 and 5.10. On the other hand, switching to use hypothesis transcriptions during training would cause a much greater harm to SER, although the edge was better maintained when using grapheme transcriptions.

Tri-gram versus Four-gram

The performance difference between using tri-gram and four-gram transcriptions is not as consistent as in the comparisons made above. Reports of the word error rate of these transcriptions could potentially provide some insights by quantifing the real differences between these transcriptions.

Notes on Using Oracle Speaker Number

We noticed that in some cases, setting the number of clusters to the correct value gave higher SER than having the clustering algorithm to determine the number of clusters. This was counter-intuitive at first, but having looked at the clustering results for individual meetings, we realised that when using oracle speaker number, it was more common for one dominant speaker to be split into different clusters, whereas the spectral clustering algorithm would often underestimate the number of clusters, causing different speakers to be merged into one cluster. The former type of mistake would cause greater harm in SER because of the mapping operation that maps labels in the diarisation output to speakers in the reference file.

Future Experiments

Given more time, the project could be extended with the following experiments in order to examine the robustness and generalisability of the model.

1. Create mismatched transcriptions to use during evaluation, gradually increase the proportion of mismatched transcripts as a stress-testing experiment.
2. Systematically replace certain graphemes to observe the importance of certain graphemes in discriminating speaker identity as well as the effect of certain confusions made by the ASR system.
3. Try the same experiments on another language.
4. Apply intermediate fusion with tensor operation.

5.4 Phone-invariant Training

We introduced the gradient reversal layer described in section 4.2.2 to perform phone-invariant pretraining. The baseline experiment used 3 epochs for pre-training and 20 for finetuning. For

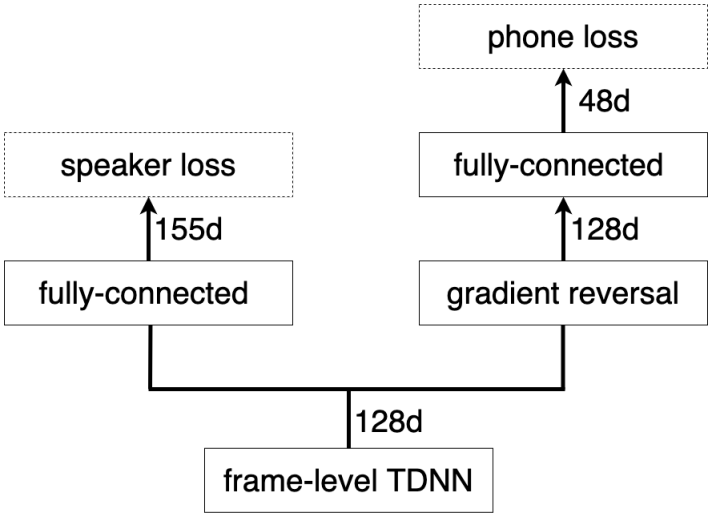


Fig. 5.6 Phone-invariant pre-training

phone-invariant pretraining, we increased the number of epochs to 10 while using 20 epochs for finetuning. As for finetuning, the only difference between phone-invariant training and the baseline was that the frame-level model would be freezed for the former, i.e. the parameters of the TDNN will not be updated during finetuning.

5.4.1 Results and Analysis

reversal coefficient λ	frame level training accuracy	frame level cv accuracy	segment level training accuracy	segment level cv accuracy
-2	57.91	54.68	84.52	81.36
-1.5	59.15	55.38	84.64	81.62
-1	61.11	56.75	85.33	**82.04**
-0.1	68.19	59.43	85.08	79.61
0	68.18	59.00	**87.58**	81.31
0.0001	56.03	52.58	77.41	72.94

Table 5.11 classification accuracy of phone-invariant training

Recall that the gradient reversal layer would multiply the incoming back-propagated gradient by $-\lambda$. Thus the frame-level feature extractor would be trained to maximise phone classification loss when $\lambda > 0$ and to minimise it when $\lambda < 0$. $\lambda = 0$ was equivalent the baseline model trained with extra epochs during pre-training.

To determine the appropriate range for tuning the reversal coefficient, we checked the gradient back-propagated from the phone classifier and the speaker classifier to the frame-level feature extractor and confirmed they are of the same scale. However, as shown in the table above, the performance deteriorated very quickly when we applied phone-invariant training with a very small λ, i.e. the gradient back-propagated from the phone classifier was four orders-of-magnitude smaller than that from the speaker classifier. When we increased λ to 0.001, the model could not even converge. Interestingly, when we set λ to negative values, the model withstood λ of much greater magnitude and its performance saw much subtler variations. We have also tried finetuning without freezing the frame-level model, which gave similar results.

Liu et al. [32] suggested that speaker and phonetic components share common information such as formant, pitch trajectory and spectral energy distribution, thus phone classification can actually be used as a auxiliary task for regular MTL. Note that in our experiments, although using phone labels for MTL caused a drop in training accuracy, the cross-validation remained stable and even outperformed the baseline. Therefore, it was also possible that the phone classifier provided regularisation for the speaker embedding extractor.

Since the phone-invariant pre-training was done on the frame level, to evaluate its effect, we used frame-level embeddings for probing. For comparison, we used randomly generated vectors as a baseline to observe how much phone information would be extracted from completely random "embeddings". We made the following observations from Figure 5.7. First, applying MTL by setting negative λ did enhance the phone information in the embeddings, which can be further enhanced by making λ more negative. Moreover, since the training curves converged to the same value for the random baseline, embeddings obtained with and without using phone-invariant training ($\lambda = -0.001$ and 0 respectively), we can conclude that using phone-invariant training did not prevent the extraction of phone information from the embeddings trained as long as enough epochs were trained. The final observation was the drastic drop of cv accuracy for frame-level embeddings obtained with phone-invariant training. This was possibly because the embeddings obtained through phone-invariant training lied in an unstable region in the high-dimensional space, causing the classification result to be very sensitive to small variations in the projection matrix, *i.e.* the hidden layer weight, leading to the greater fluctuation in the accuracy across epochs.

There are at least three potential reasons why our phone-variant training method did not work:

1. Recall from 4.2.2, the architecture that we used to perform adversarial learning was based on the assumption that the phone classification loss was a proxy for the dissimilarity between feature vectors corresponding to segment pairs with the same speaker and different phones. This might not be a good enough approximation, which would contradict the fundamental assumption of our approach.

2. Another major assumption of this approach was the optimal discriminability of the phone classifier. It is likely that a single layer does not have enough capacity to perform well in

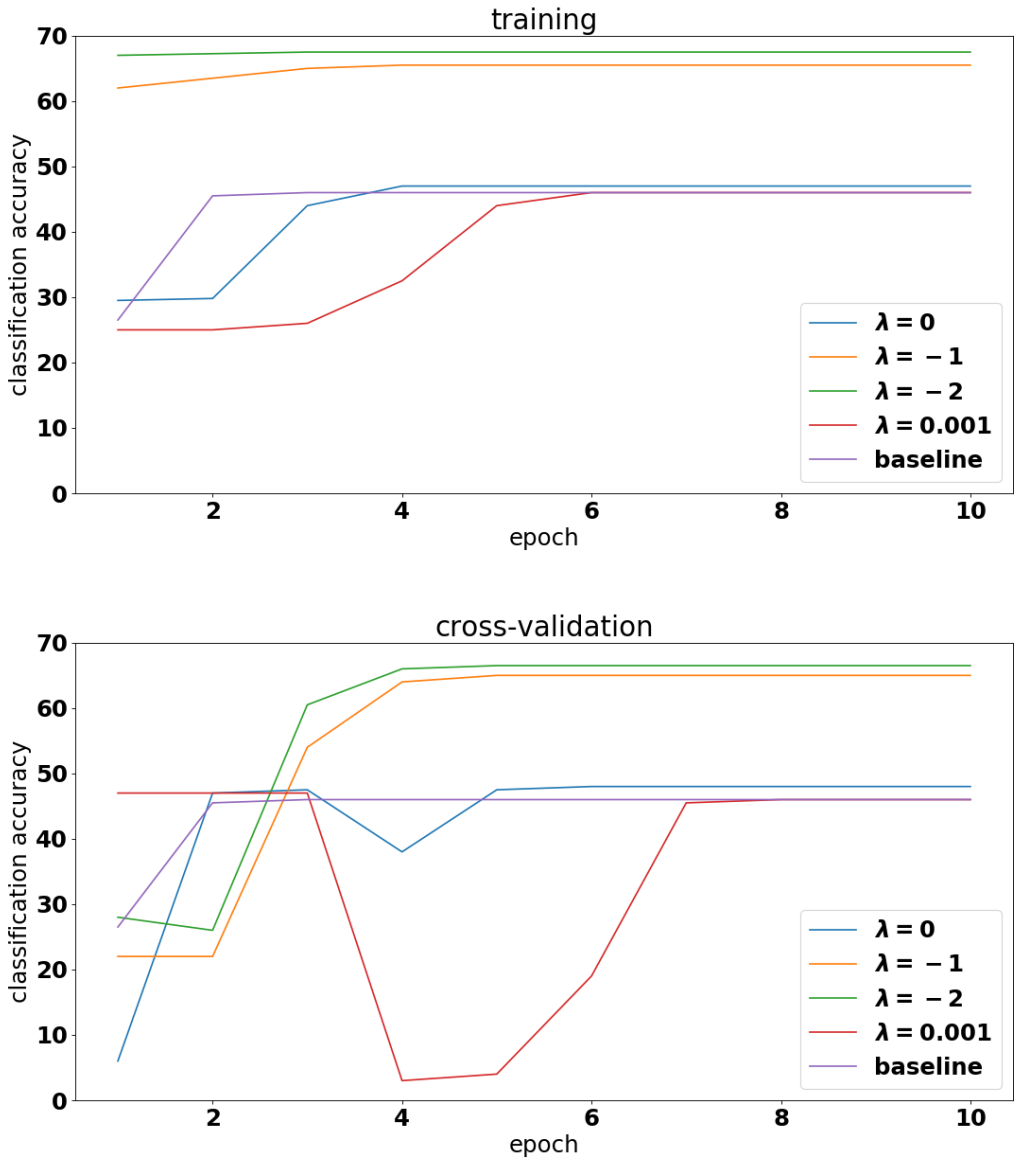


Fig. 5.7 Classification accuracy during trainng of the probe classifier

phone classification. Thus, additional experiments should be done using more complex phone classifiers.

3. It might be working on the frame-level limits the effect of phone-invariant training. Freezing the TDNN during fine-tuning prevents the frame-level model from updating to extract feature that is most discriminative on the segment level. On the other hand, if we do not freeze the frame-level model, it is likely that the effect of phone-invariant training will be overwritten by the updates during finetuning.

4. Adapting the training schedule, e.g. reducing the learning weight, could potentially help with the convergence of phone-invariant training.

Chapter 6

Conclusions

This project proposed and investigated two ways of utilising transcriptions in the extraction of speaker embeddings for diarisation. In the *multimodal fusion* architecture, the transcriptions act as input features providing complementary information for the original acoustic features. On the other hand, in phone-invariant training, the phonemes are used as labels to supervise the training of the embedding extractor, aiming at filtering out phone information and only leaving information regarding phone-independent speaker characteristics in the speaker embeddings.

To the best of our knowledge, our work is the first to apply multimodal fusion to diarisation by incorporating acoustic features and grapheme/phoneme transcriptions for the extraction of speaker embeddings. Our multimodal extractor constantly outperformed the baseline system under settings with and without prior knowledge of oracle VAD segmentation or number of speakers, with grapheme transcriptions giving better results than its phoneme counterpart. In general, the results were encouraging and beg for more experiments on examining the robustness and generalisability of the model, as well as attempts to utilise more complex fusion models that use tensor operations for intermediate fusion.

Our proposed method for phone-invariant training, on the other hand, has not given much positive results. Yet more experiments using more complex phone classifier might need to be done before we can confirm that the method is not feasible. However, we could also argue that the success of the multimodal fusion architecture suggest that phone information, when distilled to the appropriate form, is actually beneficial for creating speaker discriminative embeddings, which agrees with the deterioration observed in classification accuracy when adversarial training was applied to remove phone information in speaker embeddings.

References

[1] Q. Wang, C. Downey, L. Wan, P. A. Mansfield, and I. L. Moreno, "Speaker diarization with LSTM,"

[2] M.-Z. Li and X.-L. Zhang, "Learning deep representations by multilayer bootstrap networks for speaker diarization,"

[3] D. Garcia-Romero, D. Snyder, G. Sell, D. Povey, and A. McCree, "Speaker diarization using deep neural network embeddings," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4930–4934, IEEE.

[4] S. H. Shum, N. Dehak, R. Dehak, and J. R. Glass, "Unsupervised methods for speaker diarization: An integrated and iterative approach," vol. 21, no. 10, pp. 2015–2028.

[5] M. Senoussaoui, P. Kenny, T. Stafylakis, and P. Dumouchel, "A study of the cosine distance-based mean shift for telephone speech diarization," vol. 22, no. 1, pp. 217–227. Conference Name: IEEE/ACM Transactions on Audio, Speech, and Language Processing.

[6] G. Sell and D. Garcia-Romero, "Speaker diarization with plda i-vector scoring and unsupervised calibration," in *2014 IEEE Spoken Language Technology Workshop (SLT)*, pp. 413–417.

[7] G. Sell and D. Garcia-Romero, "Diarization resegmentation in the factor analysis subspace," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4794–4798. ISSN: 2379-190X.

[8] T. Drugman, Y. Stylianou, Y. Kida, and M. Akamine, "Voice activity detection: Merging source and filter-based information," vol. 23, no. 2, pp. 252–256.

[9] R. Zazo-Candil, T. N. Sainath, G. Simko, and C. Parada, "Feature learning with raw-waveform CLDNNs for voice activity detection," in *INTERSPEECH*.

[10] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," vol. 19, no. 4, pp. 788–798.

[11] G. Sun, C. Zhang, and P. Woodland, "Speaker diarisation using 2d self-attentive combination of embeddings,"

[12] S. J. D. Prince, "Probabilistic linear discriminant analysis for," in *Inferences About Identity ," ICCV*.

[13] C. Zhang and K. Koishida, "End-to-end text-independent speaker verification with triplet loss on short utterances," in *Interspeech 2017*, pp. 1487–1491, ISCA.

[14] Y. Fathullah, C. Zhang, and P. C. Woodland, "IMPROVED LARGE-MARGIN SOFTMAX LOSS FOR SPEAKER DIARISATION," p. 5.

[15] D. Snyder, D. Garcia-Romero, D. Povey, and S. Khudanpur, "Deep neural network embeddings for text-independent speaker verification," in *Interspeech 2017*, pp. 999–1003, ISCA.

[16] G. Sell, D. Snyder, A. McCree, D. Garcia-Romero, J. Villalba, M. Maciejewski, V. Manohar, N. Dehak, D. Povey, S. Watanabe, and S. Khudanpur, "Diarization is hard: Some experiences and lessons learned for the JHU team in the inaugural DIHARD challenge," in *Interspeech 2018*, pp. 2808–2812, ISCA.

[17] A. Zhang, Q. Wang, Z. Zhu, J. Paisley, and C. Wang, "Fully supervised speaker diarization,"

[18] Q. Li, F. L. Kreyssig, C. Zhang, and P. C. Woodland, "Discriminative neural clustering for speaker diarisation,"

[19] Z. Meng, J. Li, Z. Chen, Y. Zhao, V. Mazalov, Y. Gong, Biing-Hwang, and Juang, "Speaker-invariant training via adversarial learning," pp. 5969–5973.

[20] T. J. Park and P. Georgiou, "Multimodal speaker segmentation and diarization using lexical and acoustic cues via sequence to sequence neural networks,"

[21] C. Zhang and P. Woodland, "High order recurrent neural networks for acoustic modelling,"

[22] Y. Liu, L. He, and J. Liu, "Large margin softmax loss for speaker verification,"

[23] S. Shum, N. Dehak, and J. Glass, "On the use of spectral and iterative methods for speaker diarization," p. 4.

[24] Z. Liu, Y. Shen, V. B. Lakshminarasimhan, P. P. Liang, A. Bagher Zadeh, and L.-P. Morency, "Efficient low-rank multimodal fusion with modality-specific factors," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2247–2256, Association for Computational Linguistics.

[25] T. Wörtwein and S. Scherer, "What really matters — an information gain analysis of questions and reactions in automated PTSD screenings," in *2017 Seventh International Conference on Affective Computing and Intelligent Interaction (ACII)*, pp. 15–20. ISSN: 2156-8111.

[26] B. Nojavanasghari, D. Gopinath, J. Koushik, T. Baltrušaitis, and L.-P. Morency, "Deep multimodal fusion for persuasiveness prediction," in *Proceedings of the 18th ACM International Conference on Multimodal Interaction - ICMI 2016*, pp. 284–288, ACM Press.

[27] A. Fukui, D. H. Park, D. Yang, A. Rohrbach, T. Darrell, and M. Rohrbach, "Multimodal compact bilinear pooling for visual question answering and visual grounding,"

[28] S. Ruder, "An overview of multi-task learning in deep neural networks,"

[29] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," p. 10.

[30] Y. Liu, L. He, J. Liu, and M. T. Johnson, "Introducing phonetic information to speaker embedding for speaker verification," vol. 2019, no. 1, p. 19.

[31] S. Wang, Y. Qian, and K. Yu, "What does the speaker embedding encode?," in *Interspeech 2017*, pp. 1497–1501, ISCA.

[32] A. Canavan, D. Graff, and G. Zipperlen, "Callhome american english speech ldc97s42,"

[33] J. Carletta, S. Ashby, S. Bourban, M. Flynn, M. Guillemot, T. Hain, J. Kadlec, V. Karaiskos, W. Kraaij, M. Kronenthal, G. Lathoud, M. Lincoln, A. Lisowska, I. McCowan, W. Post, D. Reidsma, and P. Wellner, *The AMI Corpus*.

[34] X. Anguera, C. Wooters, and J. Hern, "Acoustic beamforming for speaker diarization of meetings," pp. 2011–2023.

[35] F. L. Kreyssig, C. Zhang, and P. C. Woodland, "Improved tdnns using deep kernels and frequency dependent grid-RNNS," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4864–4868. ISSN: 2379-190X.

[36] S. Young, G. Evermann, M. Gales, T. Hain, D. Kershaw, X. A. Liu, G. Moore, J. Odell, D. Ollason, Z. Povey, A. Ragni, V. Valtchev, P. Woodland, and C. Zhang, *The HTK book*. 2015.

[37] S. J. Young, J. J. Odell, and P. C. Woodland, "Tree-based state tying for high accuracy acoustic modelling," in *Proceedings of the workshop on Human Language Technology - HLT '94*, p. 307, Association for Computational Linguistics.

[38] D. Povey and P. C. Woodland, "MINIMUM PHONE ERROR AND i-SMOOTHING FOR IMPROVED DISCRIMINATIVE TRAINING," p. 4.

[39] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust DNN embeddings for speaker recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5329–5333, IEEE.

Appendix A

Covid-19 Disruption

The Covid-19 did not cause disruption to the experiments in this project since they are purely computer-based.

Thea main disruption caused is in the communications between my supervisors and me. Although we have been holding online meetings on the same weekly basis, the past two months has seen a considerable reduction in the amount of time that I visit the lab to speak to my supervisors about the experiments.

Appendix B

Risk Assessment Retrospection

The project is purely computer-based. The hazards identified in the risk assessment form were not encountered and no additional hazards were identified in the course of the project.