Discriminative Neural Clustering

James Thompson

Department of Engineering
University of Cambridge

This dissertation is submitted for the degree of
Master of Engineering

Emmanuel College June 2022

To Grandad, friend of technology.

Acknowledgements

There are several people who have been invaluable in helping this project come to fruition. First, I would like to thank Prof. Phil Woodland for giving me the opportunity to undertake this project. His guidance and advice have been instrumental in shaping the project's direction. I would also like to give thanks to Florian Kreyssig, Qiujia Li and Dr Chao Zhang who were present at our weekly meetings and gave indispensable help. They were patient answering my questions (of which there were many), both on Microsoft Teams and in the Machine Intelligence Laboratory. Furthermore, Qiujia, Florian, Chao and Phil were responsible for writing the original paper on Discriminative Neural Clustering, without which this thesis could not exist. Thanks also go to Evonne Lee who kindly provided me with her Wav2Vec2 embeddings, which are the result of her own MEng project. Finally, I am thankful to all the other staff in the Department of Engineering who allow MEng projects to function, from the administrators to the IT experts.

Abstract

Discriminative Neural Clustering (DNC) is a proposed method of supervised clustering for speaker diarisation. The task of diarisation is to label who is speaking in a multi-speaker audio stream at each point in time. This is a key component in enabling the automatic transcription of meetings and conversations, and can be applied to a wide range of scenarios. Research into solving the diarisation problem has been ongoing for about 30 years. Since the advent of deep learning, the focus has shifted from Gaussian Mixture Models and Hidden Markov Models towards neural networks. DNC is a step in this direction and could be extended to make an end-to-end system.

In this project, it is assumed that earlier pre-processing stages in the pipeline have been completed so that the input is a sequence of vectors from which the aim is to produce a one-to-one mapping to a sequence of speaker labels. The absolute speaker identities are not important, only the relative speaker positions (the first speaker is labelled '1', the second speaker '2' and so on). A variety of inputs are experimented with in this project. There are two kinds of segmentation: 'segment-level' refers to splitting the input stream into speaker-homogeneous segments, and 'window-level' refers to splitting each utterance into fixed length windows of about 1 s. Both of these assume that Voice Activity Detection (VAD) has been performed, but only at the segment-level is Change Point Detection (CPD) assumed, meaning it is already known where the speaker changes. The input vectors themselves consist of some combination of 32-dimensional speaker embeddings, TDOA values and GCC-PHAT values. The embeddings are either 'd-vectors' or 'Wav2Vec2-based embeddings', both of which have been produced with the intention that utterances from the same speaker will be converted to embeddings that are close in the embedding space. TDOA and GCC-PHAT values give directional information which is available when multiple microphones are used to record the meeting. In real-world applications, this will not always be viable, whereas embeddings can always be produced.

Our task can hence be framed as a clustering problem. This is the last stage of the diarisation pipeline and aims to group the input vectors so that each cluster corresponds to a different speaker label. Typically, the clustering algorithms used for this are unsupervised, such as Spectral Clustering (SC) which is a more sophisticated form of k-means clustering and is used as a baseline in this project. DNC was recently introduced by members of the Speech Research Group in the Cambridge University Engineering Department and is based on the Transformer model, a neural network architecture used for sequence tasks. The model must learn a distance measure rather than assuming one, unlike spectral clustering which uses cosine similarity. The availability of training data is limited, hence several forms of data augmentation are investigated. Sub-meeting randomisation and input vectors randomisation are used to augment input sequences by sub-sampling and re-arranging real meetings, while L_2-normalised speaker embeddings can be augmented by rotating them to a different part of the unit hypersphere with Diaconis augmentation. Two forms of augmentation are proposed for TDOA/GCC-PHAT vectors which involve attempting to add noise to the values and permuting the values within vectors consistently across a meeting. A significant part of this project has been coding a new, more flexible implementation of DNC, which includes augmenting data 'on-the-fly' rather than as a pre-processing stage with large storage cost. DNC is trained using Adam optimisation and the cross-entropy loss function. Dropout is used for regularisation, and curriculum learning, which involves beginning with simple training examples and progressing to harder ones, is key for handling longer meetings. One restriction of DNC is that the maximum possible number of speakers (in thus case 4) must be specified beforehand.

The model was trained and evaluated on the AMI dataset. This is a more challenging dataset than others such as CALLHOME since meetings are long (averaging over 30 minutes) and there are up to 4 speakers with no single speaker dominating. The experimental strategy was to first test different input settings for short sub-meetings (50 speaker-homogeneous segments or the equivalent length in windows), and then try full meetings with the best settings found. This meant first experimenting with the different d-vector augmentation methods at the segment-level. The best combination was found to be sub-meeting randomisation with Diaconis augmentation which achieved a DER of **12.71%** on the eval set, compared to a spectral clustering baseline of 15.66%. This augmentation method was used for all future experiments where relevant. Moving onto window-level clustering, different window lengths were investigated, with the best being 1.5s. This achieved an eval DER of **19.54%** on meetings of 101 windows, which is marginally better than the SC baseline of 19.73%. This was expected to be worse than segment-level clustering since the lack of CPD means that when there are two speakers in a window, the model can only give one label to the whole window and so there is guaranteed to be some error. TDOA vectors performed exceptionally well. Used on their own, they achieved **6.79%** eval DER compared to an SC baseline of 16.15%. The best result was for d-vectors, TDOA values and GCC-PHAT values together which achieved **5.85%** compared to a 12.43% SC baseline. TDOA vectors are the dominant feature, but d-vectors still provide useful information. In isolation, Wav2Vec2 performed better at the window-level than d-vectors, with a DER of **10.91%**, but once TDOA/GCC-PHAT values were added, their performance was marginally worse than the d-vector equivalent. On full meetings, which is a much harder task, the performance of DNC deteriorates but is still better than SC, with an eval DER of **17.25%** achieved with an input of d-vectors, TDOA and GCC-PHAT at the window-level compared to 17.86% for SC.

This project shows that supervised clustering is a viable alternative to unsupervised clustering for speaker diarisation. The most novel and interesting outcome has been the success of including TDOA as an additional input. Future work could involve enabling overlap detection and creating an end-to-end diarisation system.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview of Speaker Diarisation

Speaker diarisation is the task of determining ‘who spoke when’ in a multi-speaker audio stream [1], which is a key requirement for many speech applications. It is required for automatic transcriptions of conversations, and would be essential for systems to produce automatic minutes of meetings, notes from interviews or records of court proceedings. Speaker diarisation can also be applied to automatic subtitle generation for videos, though in the case of video calls, it is already known who is speaking (assuming one speaker per device). The ability for a machine to determine who is speaking will become increasingly important as virtual assistants like Siri and Alexa continue to be integrated into existing technologies to the point of ubiquity.

The typical pipeline for speaker diarisation is given in Figure 1.1. The input is a continuous stream of audio which first undergoes Voice Activity Detection (VAD), where regions containing speech are labelled. Next, Change Point Detection (CPD) identifies the points where the speaker changes, giving a sequence of speaker-homogeneous segments. Each segment is converted to a series of speaker embeddings, where vectors corresponding to the same speaker should be close in the embedding space. The embeddings encapsulate information about the speaker’s voice over a predefined period of time. This project deals with embeddings that take information from 15 or 215 surrounding frames, where each frame represents 10 ms of audio. The embeddings themselves are at shifts of 10 ms and so each embedding shares some input information with its neighbours since the number of input frames per embedding is longer than the shifts between embeddings. It is common to take the mean of all the embeddings in a speaker-homogeneous segment to give one embedding per segment. There is then a sequence of embeddings that has a one-to-one mapping to a sequence of speaker labels. Finally, clustering is performed on the embeddings to group segments close in the embedding space. Each cluster corresponds to a speaker label and so the final output is a sequence of speaker labels for the input audio stream. The main focus of this project is the clustering phase.

This project begins with the baseline model developed by Li et al. [2] called Discriminative Neural Clustering (DNC). This uses d-vectors taken from a Time Delay Neural Network (TDNN) [3] as input and assumes that the first three phases of the typical diarisation pipeline have

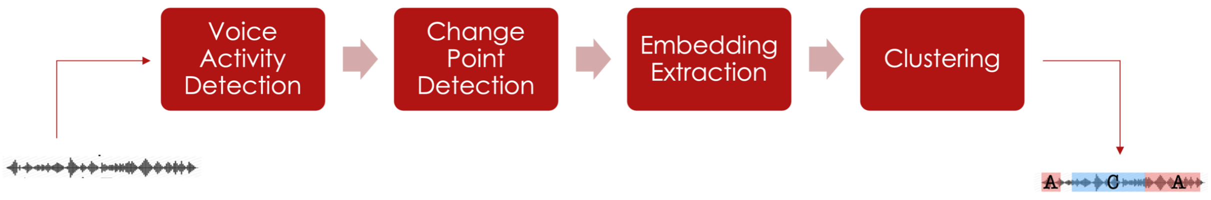


Figure 1.1: Speaker diarisation pipeline

been completed. The clustering phase is then achieved using a Transformer model [4]. This supervised method is a departure from unsupervised techniques such as k-means clustering and spectral clustering [5]. The aim of this project is to expand on DNC by removing some of the original assumptions, improve the implementation and achieve better results by including additional input types. Specifically, this project investigates window-level clustering, where CPD is not assumed to have been done and inputs are fixed-length windows rather than speaker-homogeneous segments. Data augmentation is implemented 'on-the-fly' during training rather than having to produce a large batch of augmented data beforehand which has large storage cost. Different inputs experimented with are TDOA vectors, which give directional information when multiple microphones are available and can be used in isolation or in conjunction with speaker embeddings, and Wav2Vec2 embeddings which are used as alternative to d-vectors.

1.2 Outline of the Report

The remainder of the report is structured as follows. First, some more background is given regarding the task of speaker diarisation and related work in the area. Relevant theory is discussed to provide an understanding of the techniques used in the project. It is presented both in generic terms, and as applied to the task at hand, beginning with neural networks in general before discussing the Transformer model, DNC and speaker embedding generation as well as Beamforming and other clustering techniques. Next, details of the implementation are given since a significant part of this project has been to develop a flexible codebase from which experiments can be performed. The most important aspect of the implementation is the different forms of data augmentation. Finally, the most relevant results from the project are presented and discussed, beginning with short sub-meetings and progressing to full meetings. Analysis is given throughout the results chapter as well as in the conclusion, where ideas for future work are also considered.

Chapter 2

Background

Speaker diarisation technology has progressed significantly over the past 30 years and is well documented, not just by original papers, but also in surveys charting the development of different diarisation techniques [6] [7]. Attempts to solve the problem of speaker diarisation began in the 1990s with the aim of aiding Automatic Speech Recognition (ASR) by enabling speaker adaptive processing [1]. In this pre-deep learning landscape, an early method for Change Point Detection and Clustering was based on the Bayesian Information Criterion (BIC) [8] which uses a penalised Maximum Likelihood approach and models the input audio data using a Gaussian distribution. By the early 2000s, several research consortia had formed, including the Augmented Multiparty Interaction (AMI) Consortium, whose corpus of data is used in this project [9]. With the onset of deep learning in the 2010s, traditional methods such as BIC and Factor Analysis [10] were eschewed in favour of neural networks. These have quickly replaced each component of the diarisation pipeline, with clustering one of the last to be tackled. Discriminative Neural Clustering is a deep learning approach to clustering and acts as a step towards enabling end-to-end speaker diarisation, where the entire pipeline is modelled by a single end-to-end trainable neural network.

2.1 Categorisation of Diarisation Techniques

Park et al. [1] suggest dividing diarisation techniques into 4 categories. These separate methods are based on whether they are trained on a diarisation objective or not, and whether they jointly optimise multiple steps in the diarisation pipeline, or just a single task. Using this taxonomy, segment-level DNC would be classified as 'single-module optimisation trained on a diarisation objective'. Window-level DNC, which does not assume CPD, may be classified as joint optimisation.

Techniques classified as 'trained on non-diarisation objective' generally refer to traditional methods that do not use deep learning. There continues to be research in this area, such as the work of Diez et al. which uses Bayesian Hidden Markov Models to address the entire speaker diarisation pipeline [11] or just to cluster x-vectors [12]. Their end-to-end system achieves

9.0% DER on the CALLHOME dataset (see Section 4.5 for DER definition). Deep-learning technologies in this category include the generation of speaker embeddings that have come from neural networks with a non-diarisation objective, such as the d-vectors in the work of Variani et al. [13] where the task is speaker verification (determining if a speaker matches a known identity), not diarisation. Further details on types of embeddings, as well as unsupervised clustering methods are given in Chapter 3.

2.2 Fully Supervised Speaker Diarisation

Our main interest in previous work is other attempts at supervised clustering, which may be part of a fully supervised speaker diarisation system. One of the motivations behind DNC is to enable an end-to-end supervised system, which is at the cutting-edge of speaker diarisation technology.

The Unbounded Interleaved-State Recurrent Neural Network (UIS-RNN) model was introduced by Zhang et al. in 2019 [14] to provide supervised clustering of speaker embeddings. Each speaker is modelled by an instance of an RNN, where each RNN shares the same parameters but with different state sequences. UIS-RNN performs well on the CALLHOME dataset, achieving 7.6% DER. Although this model can deal with an unknown number of speakers, while DNC requires the maximum number of speakers to be specified, it imposes some assumptions on the distribution of speaker embeddings (such as an identity covariance matrix) not required by DNC. DNC also tackles the more challenging AMI dataset which includes meetings of four speakers, averaging over 30 minutes in length, compared to roughly 2 minute conversations for CALLHOME, with two speakers dominating for 90% of the time.

UIS-RNN is used a baseline model and extended by Wan et al. [15] to allow online diarisation (which assumes only the audio up to the current moment is available, not the whole input sequence). Clustering is performed on their 'in-house' d-vectors. With each utterance split into fixed 2 s windows, this model achieves 7.83% DER on a subset of the AMI dataset that does not include meetings from the TNO room.

The most promising framework for end-to-end speaker diarisation is EEND (End-to-End Neural Diarisation), first proposed by Fujita et al. [16]. This has since been extended to allow an unlimited number of speakers [17] and online diarisation [18]. EEND achieves 15.29% DER on the CALLHOME dataset for an unknown number of speakers while handling overlap. Due to the different dataset and assumptions, it is not directly comparable to the DNC in this project.

Ongoing developments in speaker diarisation include handling video data to improve diarisation accuracy [19] and jointly optimising, not just all stages of speaker diarisation, but ASR as well. It is an unsolved problem whether the interdependence between the two tasks means that a joint optimisation can outperform separately tuned systems [20]. A joint system may learn to use linguistic features as well as acoustic ones for speaker diarisation.

Chapter 3

Theory

3.1 Neural Networks

A neural network is a deep learning model consisting of layers of neurons which perform non-linear activation functions, $\phi()$, on a linear transformation of their inputs [21]. These are parameterised by weight matrices, $\boldsymbol{W}^{(j)}$, and bias vectors, $\boldsymbol{b}^{(j)}$, which transform the inputs. The aim is to learn a function that maps some input, $\boldsymbol{X}$, to an output, $\boldsymbol{Y}$, by choosing the network parameters that minimise a loss function. Different model architectures are used for different tasks, the simplest of which is the fully connected network, or 'multi-layer perceptron'. A fully connected network is defined by the number of layers and the number of nodes in each layer. Each neuron in the hidden layers takes the output of each node in the previous layer as input. For layer j:

$$\boldsymbol{y}^{(j)} = \phi(\boldsymbol{z}^{(j)}) = \phi(\boldsymbol{W}^{(j)}\boldsymbol{x}^{(j)} + \boldsymbol{b}^{(j)})$$

Typical activation functions include ReLU, tanh and sigmoid:

$$\textsc{ReLU}(z_i^{(j)}) = \max(0, z_i^{(j)}) \qquad \tanh(z_i^{(j)}) = \frac{e^{z_i^{(j)}} - e^{-z_i^{(j)}}}{e^{z_i^{(j)}} + e^{-z_i^{(j)}}} \qquad \textsc{Sigmoid}(z_i^{(j)}) = \frac{1}{1 + e^{-z_i^{(j)}}}$$

The final layer for classification tasks is usually a softmax function whose outputs are between 0 and 1 and sum to 1, meaning they can be interpreted as class probabilities. In a speaker classification application, the classes would correspond to speaker labels and the assigned speaker label would be the class with the greatest probability. The softmax function is given by:

$$\textsc{Softmax}(z_i^{(j)}) = \frac{e^{z_i^{(j)}}}{\sum_{k=1}^{K} e^{z_k^{(j)}}}$$

where K is the number of classes. Figure 3.1 shows a fully connected network.

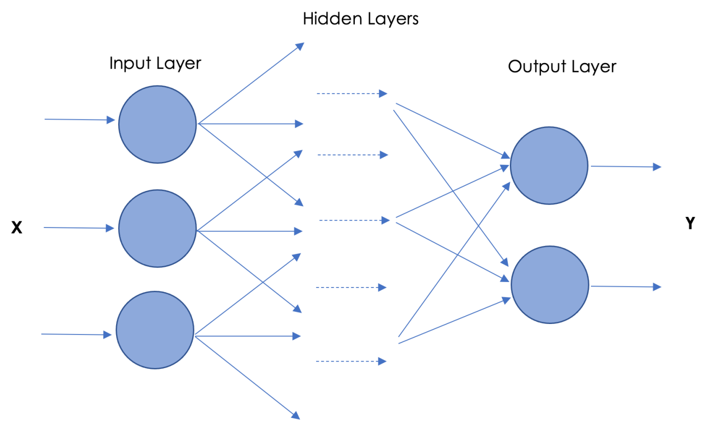


Figure 3.1: Fully connected neural network

3.2 Sequence-to-Sequence Models

When dealing with sequence data, as in this project, there are often better model architectures than the fully connected network, although these models may include fully connected layers (such as in DNC). One class of models used for sequence data is Recurrent Neural Networks (RNNs), as used by Zhang et al. for fully-supervised speaker diarisation in their UIS-RNN model [14]. Since its introduction in 2017 [4], the Transformer model has become the pre-eminent neural network architecture for handling sequence data and has been applied to a wide variety of speech tasks. It forms the underlying structure of Discriminative Neural Clustering.

3.2.1 Recurrent Neural Networks

The simplest form of Recurrent Neural Network is the Elman Network [22] which maps each item of the input sequence to the corresponding item in the output sequence. This is the goal in the clustering phase of speaker diarisation, where each item in the input sequence is a speaker embedding which maps to a speaker label. The Elman network consists of hidden vectors, $\boldsymbol{h}_t$, which store information about the history of the sequence. The following equations and Figure 3.2 give details of a recurrent unit, which is the fundamental building block of RNNs (not just the Elman network):

$$\boldsymbol{h}_t = \boldsymbol{f}^h(\boldsymbol{W}_h^f \boldsymbol{x}_t + \boldsymbol{W}_h^r \boldsymbol{h}_{t-1} + \boldsymbol{b}_h)$$

$$\boldsymbol{y}_t = \boldsymbol{f}^f(\boldsymbol{W}_y \boldsymbol{h}_t + \boldsymbol{b}_y)$$

RNNs can be further developed by introducing Long Short Term Memory (LSTM) [23] or Gated Recurrent Units (GRUs) [24]. These are more complicated architectures which use the concept of gates (sigmoid functions) and memory cells, and use more parameters to control the flow of information through the network. These are used extensively in many RNN models but will not be considered in detail here as they are not present in DNC.

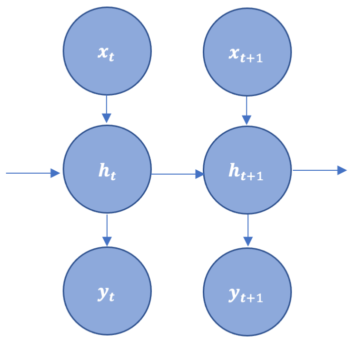


Figure 3.2: The Elman Network. Each item of the input sequence, $\boldsymbol{x}_t$, is mapped to an item of the output sequence, $\boldsymbol{y}_t$ via the history vector, $\boldsymbol{h}_t$, which stores information from previous parts of the input sequence.

3.2.2 Transformer Model

The Transformer model uses an encoder-decoder architecture, relying only on the attention mechanism rather than recurrent units. Other work using encoder-decoder models for speaker diarisation includes that of Horiguchi et al. [17] - their end-to-end speaker diarisation (EEND) can handle overlapping speakers, which DNC cannot, but is tested on the less challenging CALLHOME dataset.

First, the generic Transformer model is considered before seeing how it is applied to DNC in Section 3.6.3. In an encoder-decoder model, the input sequence is first 'encoded' into a hidden representation before being 'decoded' to get the output sequence. In this project's case, the input and output sequences are the same length, though one of the benefits of encoder-decoders is the ability to handle tasks where this is not the case. Figure 3.3 shows the architecture for the Transformer, taken from the original paper, 'Attention Is All You Need' [4]. Residual connections around the sub-layers (which help speed up training) are not shown [25]. The feed-forward layers are fully-connected layers with the ReLU activation function as specified at the start of this chapter. The multi-head attention layers are defined as follows, where $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}$ are called queries, keys and values, the parameters are the projection matrices, $\boldsymbol{W}$, and D is the dimension of the keys:

$$\text{MHA}(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V}) = \text{concatenate}(\textsc{Head}_1(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V}),\ldots,\textsc{Head}_H(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V}))\boldsymbol{W}_o$$

$$\begin{aligned}\textsc{Head}_h(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V}) &= \textsc{Attention}(\boldsymbol{Q}\boldsymbol{W}_h^Q, \boldsymbol{K}\boldsymbol{W}_h^K, \boldsymbol{V}\boldsymbol{W}_h^V) \\ &= \textsc{Softmax}\left(\frac{(\boldsymbol{Q}\boldsymbol{W}_h^Q)(\boldsymbol{K}\boldsymbol{W}_h^K)^\top}{\sqrt{D}}\right)(\boldsymbol{V}\boldsymbol{W}_h^V)\end{aligned}$$

There are two self-attention layers (where $Q = V = K$), one each in the encoder (left block) and decoder (right block), and one source-attention layer (where only $V = K$) in the decoder. Scaled dot-product attention is used as a faster alternative to additive attention. The scaling

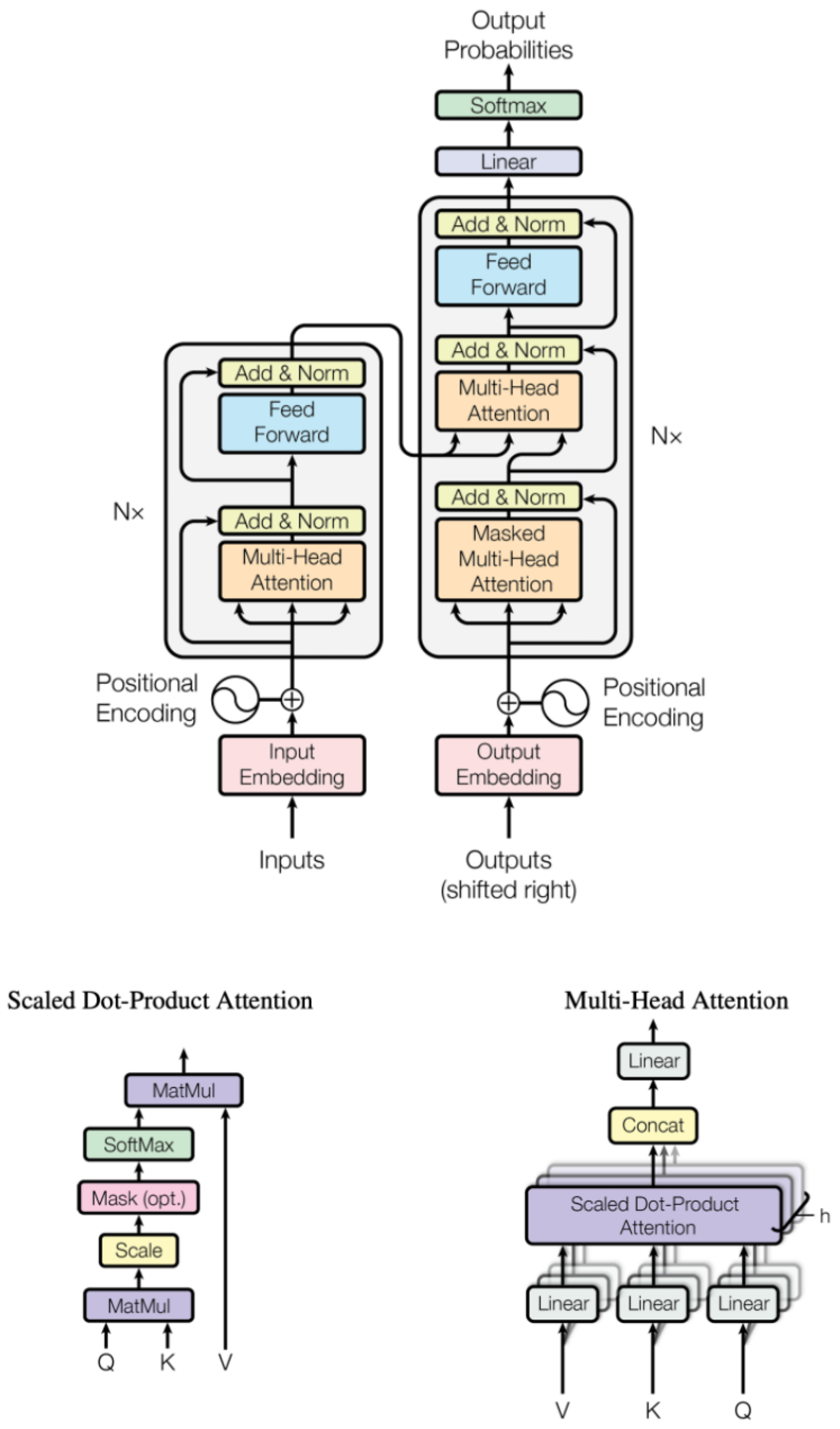


Figure 3.3: The Transformer architecture (from [4])

allows good performance to be maintained at higher dimensions. The positional encoder adds information about the position of items in the sequence to the input. Each dimension, i, of the input embedding has $\sin(\text{pos}/10000^{\frac{2i}{d}})$ added to it, where pos is the position in the sequence.

3.3 Time Delay Neural Networks

Time Delay Neural Networks (TDNNs) are used to model patterns with shift-invariance and long temporal contexts, and so have been shown to be useful for acoustic modelling [3]. It is required to understand them for this project since they are used to produce the speaker embeddings taken as input for clustering.

A TDNN consists of fully connected (FC) layers, shifted at different time steps [26]. It can be thought of as a 1-dimensional precursor to Convolutional Neural Networks (CNNs). The first layer takes inputs which are of a given temporal context (eg. 4 frames) and with a given shift (eg. 8 frames) across a given window (eg. [-10, 10]) as shown in Figure 3.4. Later layers are smaller and take as input the previous layer at different time steps. In this way, earlier layers model shorter temporal contexts while later layers deal with more of the window. Early TDNNs used a shift of 1 however this is computationally expensive and requires a large number of parameters and so recent networks have used larger and sometimes non-uniform shifts.

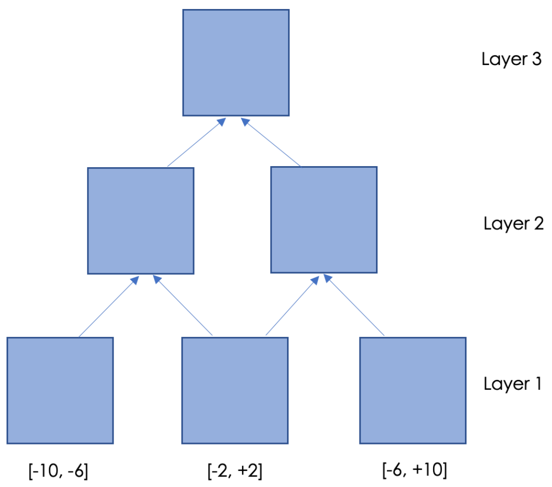


Figure 3.4: A simple example of a TDNN with an input context of 4 and input shifts of 8 across a [-10, 10] window.

3.4 Training

Training a neural network requires us to specify a loss function and optimisation method. The parameters can then be iteratively updated by inputting training data, using backpropagation to calculate the gradient of the loss with respect to the parameters, and changing the parameters to reduce the loss. Regularisation is usually included in order to prevent the model overfitting to the train set.

3.4.1 Loss Function

The loss function used in this project is the cross-entropy loss function which can be generically expressed as:

$$L(\boldsymbol{\theta}) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk}\log(y_k(\boldsymbol{x}_n))$$

where $\boldsymbol{\theta}$ are the parameters, t_{nk} is 1 if k is the true class label for the nth data point and 0 otherwise, and $y_k(\boldsymbol{x}_n)$ is the output probability for class k given the input $\boldsymbol{x}_n$. DNC deals with sequential data and so the sum is over the sequences and each item in each sequence. Again, the classes correspond to speaker labels for speaker classification tasks.

3.4.2 Optimisation

The simplest optimisation technique is Stochastic Gradient Descent (SGD) where the parameters are updated via gradient descent after each random mini-batch of data, where a mini-batch is a subset of the whole batch and updates for different mini-batches can be computed by a GPU (Graphics Processing Unit) in parallel. This is faster than updating over the whole batch and gets a better estimate of the gradient than updating after every data point. The gradient descent algorithm can be described as:

$$\boldsymbol{\theta}^{(\tau+1)} = \boldsymbol{\theta}^{(\tau)} - \eta\nabla L(\boldsymbol{\theta})|_{\boldsymbol{\theta}^{(\tau)}}$$

where η is the learning rate. One optimisation technique which has been increasingly adopted for its computational efficiency (and is used in this project) is Adam [27]:

$$\theta_i^{(\tau+1)} = \theta_i^{(\tau)} - \eta\frac{\mu_{\tau i}}{\sigma_{\tau i}+\varepsilon}$$

$$\mu_{\tau i} = (\beta_1\mu_{(\tau-1)i} + (1-\beta_1)\nabla_i L(\theta)|_{\theta^{(\tau)}})/(1-\beta_1^{\tau})$$
$$\sigma_{\tau i}^2 = (\beta_2\sigma_{(\tau-1)i}^2 + (1-\beta_2)\nabla_i L(\theta)|_{\theta^{(\tau)}}^2)/(1-\beta_2^{\tau})$$

where $\beta_1 = 0.9, \beta_2 = 0.98$, and $\varepsilon = 10^{-9}$ are fixed but the learning rate is adaptive. η ramps up linearly from 0 to 8 in the first 20,000 warm up steps and then decreases in proportion to $\frac{1}{\sqrt{\tau}}$.

3.4.3 Regularisation

The method of regularisation used in this project is dropout [28]. In general, this involves deactivating a random subset of the nodes in a network on each iteration to prevent individual nodes from learning to specialise to a particular task. In the Transformer model, dropout is applied to output of each sub-layer. This project uses a dropout rate of 10%.

3.4.4 Normalisation

Layer normalisation is applied in DNC to reduce training time [29]. For each training example, normalisation is applied to the output of each neuron in a layer using the mean and variance from the neurons in that layer (ie. the mean is subtracted before dividing by the standard deviation). This differs from batch normalisation where the statistics are calculated across a mini-batch for each neuron.

3.4.5 Teacher Forcing

Teacher forcing is a technique specific to sequence models which is commonly used during training [30]. It means that ground truth samples for the previous values in the sequence are fed back into the network to be used when predicting the next item. In other words, predictions are conditioned on the true previous values of the sequence and is a consequence of using maximum likelihood estimation (see Section 3.6.3). This forces the network to stay close to the true sequence rather than diverging during training which would hinder learning. Teacher forcing is not used during evaluation of the model which means it is possible for prediction errors to compound over time. Attempts at remedying this mismatch include Professor Forcing [31], though this is not used in this project.

3.4.6 Curriculum Learning

When training a network to model a difficult task, it is sometimes too challenging to begin with the most complex training examples. It is therefore often best to begin by training on ‘easy’ examples before moving onto harder examples in later stages of training [32] - a strategy known as curriculum learning. In the context of speaker diarisation, this means training on short meetings to begin with before increasing the meeting length until the model can handle full-length meetings. As detailed in Section 5.4, this is the approach used by DNC.

3.5 Decoding

The final model at the end of training may not be the best version if the network was trained for too long and began to overfit to the train set. Overfitting is where the model becomes too specialised to the train set, learning things which are specific to that set and do not generalise to unseen data. To spot this phenomenon, a validation set can be used during training as well as the train set. Periodically, the network will be tested against this set to measure how well it performs on unseen data. The values of the loss function and the accuracy are recorded and the values of the parameters for the current best model are saved. In this project, accuracy is the metric used to determine the best model from training.

$$\text{accuracy} = \frac{\#\text{ correct predictions}}{\#\text{ total predictions}}$$

In some applications, particularly binary classification, precision and recall (defined below) may also be of importance. These are less relevant in this project as they would have to be calculated for each of many classes and so they are unlikely to provide a clear indication of the model's clustering performance. Details of the metrics are given since they would be relevant for measuring performance in overlap detection (see Section 6.1).

$$\text{precision} = \frac{\text{true positive}}{\text{true positive} + \text{false positive}} \quad \text{recall} = \frac{\text{true positive}}{\text{true positive} + \text{false negative}}$$

Once the network has been trained, and the best model has been saved, its performance can be measured against a test set. This is a set of data which has not been used for training and so is new to the network. This data is input and predictions are computed using the fixed best values of the parameters. At this stage, teacher forcing cannot be used since the ground truth cannot be given to the network. Dropout is also no longer employed since the best prediction possible is desired and the network is not being trained anymore. Having decoded the model, the accuracy can be calculated to measure its performance. However, for speaker diarisation, the Diarisation Error Rate (DER) is the most relevant performance metric (see Section 4.5).

3.6 Clustering

Clustering is the process of separating data points into groups (clusters) based on their similarity [33]. In speaker diarisation, the final stage of the pipeline is a clustering problem since it is desired to group similar embeddings together in the hope that they have come from the same speaker. It is not the absolute identity of the speaker that matters, only the relative identities in the context of a meeting or conversation. For this reason, the clustering phase has traditionally been tackled using unsupervised methods and so Discriminative Neural Clustering, a supervised method, is a novel approach. Previous work has used k-means clustering [34], agglomerative clustering [35] or spectral clustering [36]. In this project, and in the original DNC paper [2], spectral clustering is used to get baseline results.

3.6.1 K-Means Clustering

To begin with, k-means clustering is discussed since it is a commonly used clustering method for speaker diarisation and is a key component of spectral clustering, which is used in this project. The algorithm partitions the space into k clusters, each with a different mean vector $\boldsymbol{\mu}_i$. Each data point is assigned to the cluster with the closest mean (shortest Euclidean distance). The set S_i contains the data points in cluster i. The objective is to choose the cluster means such that the total 'within cluster sum-of-squares' is minimised:

$$\min \sum_{i=1}^{k} \sum_{\boldsymbol{x}_n \in S_i} ||\boldsymbol{x}_n - \boldsymbol{\mu}_i||^2$$

The naive k-means algorithm alternates two steps until convergence (when assignments no longer change):

1. Assignment step: assign each data point to the cluster with the nearest mean.

$$S_i = \{\boldsymbol{x}_n : ||\boldsymbol{x}_n - \boldsymbol{\mu}_i||^2 \leq ||\boldsymbol{x}_n - \boldsymbol{\mu}_j||^2 \forall j, 1 \leq j \leq k\}$$

2. Update step: recompute means given new cluster assignments:

$$\boldsymbol{\mu}_i = \frac{1}{|S_i|} \sum_{\boldsymbol{x}_n \in S_i} \boldsymbol{x}_n$$

3.6.2 Spectral Clustering

Spectral clustering is a more sophisticated form of clustering, which performs dimensionality reduction via eigen-decomposition before clustering. The steps are as follows [5]: First produce an affinity matrix, A, where element A_{ij} is the cosine similarity between embeddings i and j. The diagonal elements, A_{ii}, take the value of the maximum in each row. Note that the cosine similarity between two vectors, $\boldsymbol{a}$ and $\boldsymbol{b}$, is the cosine of the angle between them, θ, given by:

$$\text{cosine similarity} = \cos(\theta) = \frac{\boldsymbol{a} \cdot \boldsymbol{b}}{||\boldsymbol{a}||||\boldsymbol{b}||}$$

The matrix is then refined with the following operations: Gaussian blur smooths the data and removes the effect of outliers. Row-wise thresholding gives a value of zero to any element less than the p-percentile in its row, where p is a hyper-parameter that can be tuned (see Section 5.2.1). The matrix is forced to be symmetric by setting $A_{ij} \leftarrow \max(A_{ij}, A_{ji})$ so that eigen-decomposition can be performed. Finally, each row is normalised: $A_{ij} \leftarrow A_{ij} / \max_k A_{ik}$.

Next, eigen-decomposition is performed on the matrix and the number of clusters, $\tilde{k}$, is set to be:

$$\tilde{k} = \arg \max_{1 \leq k \leq n} \frac{\lambda_k}{\lambda_{k+1}}$$

where $\lambda_1 > ... > \lambda_n$ are the eigenvalues. The corresponding eigenvectors are $v_1, ..., v_n$. Finally, k-means clustering is performed on new embeddings, e_i, where the kth element of the ith embedding is given by the ith dimension of the kth eigenvector: $e_i = [v_{1i}, ..., v_{\tilde{k}i}]^\top$

3.6.3 Discriminative Neural Clustering

Discriminative Neural Clustering (DNC) was introduced by Li et al.[2] in 2020 and uses a Transformer model to perform supervised clustering. A requirement of DNC is that the maximum possible number of speakers is specified (throughout this project, that number is 4). For a given meeting of length N, each speaker embedding, $\boldsymbol{x}_n$, in the input sequence $\boldsymbol{X} = [\boldsymbol{x}_1, ..., \boldsymbol{x}_N]^\top$ has an associated speaker identity z_n. However, the diarisation task is to determine a sequence of speaker labels, $y_{1:N}$, rather than the absolute identities. Figure 3.5 shows how a sequence of speaker identities (which could be thought of as the names of the speakers) would correspond to speaker labels. The first speaker to speak is labelled 1 and subsequent new speakers are labelled 2, 3, etc. While any permutation of speaker labels would be correct, this convention is enforced to simplify training.

identity sequence $z_{1:N}$	cluster label sequence $y_{1:N}$
E A C A E E C	1 2 3 2 1 1 3
A C A B B C D B D	1 2 1 3 3 2 4 3 4

Figure 3.5: Speaker identities and labels (from [2])

When predicting a speaker label, y_i, the decision is conditioned on the entire input sequence, $\boldsymbol{X}$, and the previously assigned labels $y_{0:i-1}$:

$$P(y_{1:N}|\boldsymbol{X}) = \prod_{i=1}^{N} P(y_i|y_{0:i-1}, \boldsymbol{X})$$

where y_0 is the symbol indicating the start of the sequence. This conditioning enables the model to choose the number of clusters and clustering boundaries.

The equations for DNC are given below, where $\boldsymbol{H} = [\boldsymbol{h}_1, ..., \boldsymbol{h}_N]^\top$ is the encoded hidden representation of $\boldsymbol{X}$. Comparing to the general definition of a Transformer in Section 3.2, $\boldsymbol{X}$ is the input to the encoder (left block in Figure 3.3 with $\boldsymbol{Q} = \boldsymbol{V} = \boldsymbol{K} = \boldsymbol{X}$), $\boldsymbol{H}$ is the output of the encoder and one of the inputs to the decoder (right block with $\boldsymbol{V} = \boldsymbol{K} = \boldsymbol{H}$), and the other decoder input is the embedding, $\tilde{\boldsymbol{X}}$, for the previous speaker labels $y_{1:i-1}$. Positional encoding is removed in the encoder, since during encoding, the position of the embedding in the sequence should not make a difference to the clustering result.

$$\boldsymbol{H} = \textsc{Encoder}(\boldsymbol{X})$$

$$y_i = \textsc{Decoder}(y_{0:i-1}, \boldsymbol{H})$$

Note that the parameters are different for each head and layer. The encoder learns to produce $\boldsymbol{H}$ to capture the similarity of embeddings from the same speaker. The model is effectively learning a distance measure for clustering rather than having one predefined such as with cosine similarity in spectral clustering. DNC also learns information about temporal relationships between speakers. The decoder outputs the speaker label by either looking up the label for an existing cluster or using a new label if it is determined to be a new speaker.

3.6.4 t-SNE

t-Distributed Stochastic Neighbour Embedding (t-SNE) is a method of visualising high dimensional data in 2 or 3 dimensions [37]. It is used in this project to visualise input vectors (see Section 5.2.2) in 2-D, where data points can be colour-coded to represent the different clusters (ie. speaker labels). This allows us to spot where clustering methods are making mistakes, and qualitatively observe how different clustering methods compare against each other. The algorithm to map high-dimensional $\boldsymbol{x}_1,...,\boldsymbol{x}_N$ to low-dimensional $\boldsymbol{y}_1,...,\boldsymbol{y}_N$ that preserve the similarity between embeddings is described as follows:

First, compute $p_{j|i}\ \forall i,j : \ i \neq j$, which are conditional probabilities proportional to the similarity between vectors $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, given by:

$$p_{j|i} = \frac{\exp(-||\boldsymbol{x}_i - \boldsymbol{x}_j||^2/2\sigma_i^2)}{\sum_{k\neq i}\exp(-||\boldsymbol{x}_i - \boldsymbol{x}_k||^2/2\sigma_i^2)}$$

where σ_i^2 are predefined Gaussian kernels. Then define:

$$p_{ij} = \frac{p_{i|j} + p_{j|i}}{2N} \quad q_{ij} = \frac{(1+||\boldsymbol{y}_i - \boldsymbol{y}_j||^2)^{-1}}{\sum_k \sum_{l\neq k}(1+||\boldsymbol{y}_k - \boldsymbol{y}_l||^2)^{-1}}$$

Finally, choose $\boldsymbol{y}_1,...,\boldsymbol{y}_N$ that minimise the Kullback–Leibler (K-L) divergence between distributions P and Q by using gradient descent, where the K-L divergence is given by:

$$\mathrm{KL}(P||Q) = \sum_{i\neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

3.7 Speaker Embeddings

In this project, it is assumed that the embedding extraction phase of the speaker diarisation pipeline has already been completed. Two kinds of speaker embeddings are investigated: d-vectors [38] and Wav2Vec2-based embeddings[1] [39]. Both types are 32-dimensional and L_2-normalised, meaning they lie on the 32-dimensional unit hypersphere.

3.7.1 d-vectors

Traditionally, the most widely adopted choice of speaker embeddings for speaker diarisation was i-vectors, which are generated from Gaussian Mixture Models (GMMs) [40]. As deep learning gained prominence, d-vectors began to surpass i-vectors as the superior embedding [41]. These embeddings are produced as the output of a hidden layer of a neural network. It is possible to combine the outputs of hidden layers from different neural networks to yield what are sometimes called c-vectors [42] (though this distinction in nomenclature will not be made in this paper). Combining d-vectors can reduce the effect of weaknesses in different neural network structures. When d-vectors are produced at the window-level, they are sometimes known as x-vectors [43], though once again these will continue to be referred to as d-vectors.

The d-vectors used in this project are produced from a TDNN trained for speaker classification using the angular softmax (ASoftmax) loss function [44] on the AMI dataset (see Section 5.1).

[1]The Wav2Vec2-based embeddings are the outcome of another MEng student's project this year (Evonne Lee).

There are 4 layers in the TDNN, the first 3 have size 512 and the fourth has size 128 and a linear activation function. The output vectors, $\boldsymbol{H} = [\boldsymbol{h}_1, ..., \boldsymbol{h}_T]^\top$, are then combined using a self-attentive layer with weights $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$ to get the final output, $\boldsymbol{E}$, as follows [45]:

$$\boldsymbol{A} = \textsc{Softmax}(\tanh(\boldsymbol{H}\boldsymbol{W}_1)\boldsymbol{W}_2)$$

$$\boldsymbol{E} = \boldsymbol{A}^\top \boldsymbol{H}$$

All d-vectors used in this project have an input context of [-7, +7] frames (where each frame is 10 ms apart), though different input windows are used for segment-level and window-level clustering: [-107, +107] and [-7, +7] respectively. These will be referred to as having '215 frames per d-vector' and '15 frames per d-vector'. Since the window size is equal to the context size in the latter case, clearly there is no shift for the TDNN input layers for each d-vector. The d-vectors themselves (ie. the centres of the d-vector windows) are at shifts of 1 frame. Figure 3.6 illustrates what information each d-vector represents. Because they are L_2-normalised, it is assumed that d-vectors are uniformly distributed with a mean of 0 and a variance of $1/32$. This was empirically verified in the original DNC paper [2]. Therefore, the variance is normalised by multiplying by $\sqrt{32}$.

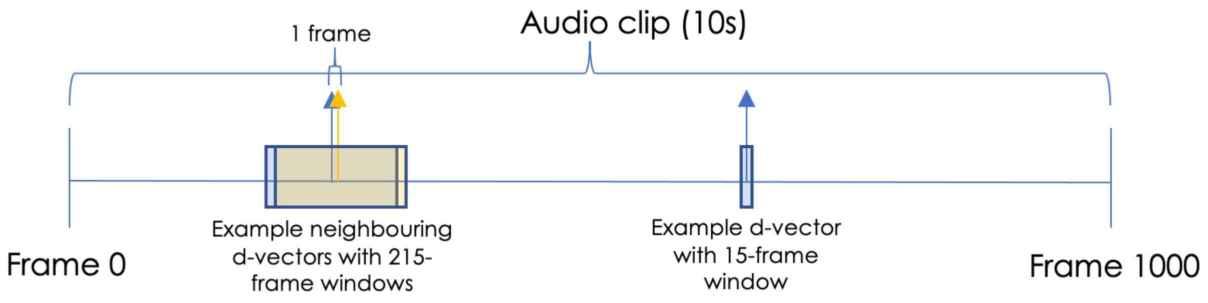

1 frame
Audio clip (10s)
Frame 0
Example neighbouring d-vectors with 215-frame windows
Example d-vector with 15-frame window
Frame 1000

Figure 3.6: Example d-vectors in an audio clip of 10 s. There are 1000 d-vectors, each separated by 1 frame (10 ms). The windows represent the information input for each d-vector: examples for 215 frames per d-vector and 15 frames per d-vector are shown.

3.7.2 Wav2Vec2-based Embeddings

Wav2Vec2-based embeddings are also produced using a neural network, but in a different way from d-vectors. Whereas d-vectors were produced per frame, the Wav2Vec2-based embeddings were produced to provide one per segment (or per window, see Section 4.1). They were also trained on VoxCeleb1 and VoxCeleb2 [46] which again differs from the d-vectors. The embedding generator uses contrastive self-supervised learning [47] and angular prototypical (AP) loss. During training, N segments are selected to form a mini-batch. A 2 s input window called the 'anchor', $\boldsymbol{x}_{i,1}$, is sampled from each segment and paired with a 'positive' sample, $\boldsymbol{x}_{i,2}$, from a segment in a different utterance with the same speaker label. The model is trained so that the corresponding embeddings, $\boldsymbol{e}_{i,1}$ and $\boldsymbol{e}_{i,2}$, which are the outputs of the network, attract while embeddings taken from segments of other speakers repel. This leaves a set of embeddings that should be able to be clustered with clear boundaries between embeddings of different speakers. The AP loss function used to achieve this is:

$$L_{ap} = -\frac{1}{N}\sum_{i=1}^{N} \log \frac{\exp(w\boldsymbol{e}_{i,1} \cdot \boldsymbol{e}_{i,2} + b)}{\sum_{j=1}^{N} \exp(w\boldsymbol{e}_{i,1} \cdot \boldsymbol{e}_{j,2} + b)}$$

3.8 Beamforming

A key feature of this project is the ability of DNC to use directional information, enabled by the presence of multiple microphones when recording each meeting (as illustrated in Figure 3.7). While the original DNC implementation used beamformed audio from all available microphones, it did not make full use of the available directional data as in this project. Details of the implementation are given in Section 4.1 but first beamforming will be explored in general terms.

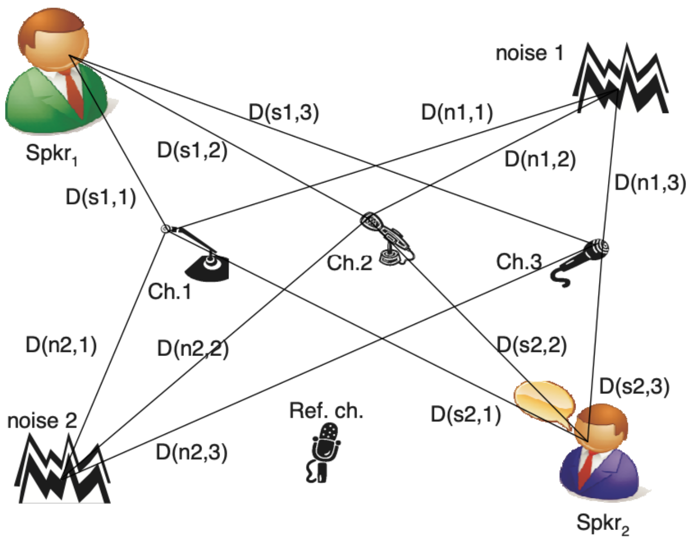


Figure 3.7: An example speaker and microphone layout in a meeting room with additional undesired noises (from [48]). The different distances, D(), between audio sources and microphones allow time delays to be calculated, from which directional information can be inferred.

The BeamformIt algorithm was introduced by Anguera et al. in 2007 [48]. The aim is to convert M (8 in AMI) audio channels, x_m, each from a different microphone, into 1 beamformed channel, y, which contains some directional information. In this process, TDOA (Time Delay of Arrival) and GCC-PHAT (Generalised Cross-Correlation with Phase Transform) values are produced. It is these vectors which are produced per frame and used as additional inputs to DNC, either on their own or by appending to speaker embeddings.

The first stage of the algorithm is to choose a reference channel, which will be the channel against which time delays are measured. In qualitative terms, the reference channel should have a high signal-to-noise ratio (SNR) and be located centrally in the meeting room. The algorithm chooses the channel, m, with the highest time-averaged cross-correlation, $\overline{\text{XCORR}}_m$, with the other channels, n:

$$\overline{\text{XCORR}}_m = \frac{1}{K(M-1)} \sum_{k=1}^{K} \sum_{n \neq m}^{M} \text{XCORR}[m,n;k]$$

where K is the number of 1s blocks, k, used in the average. $\text{XCORR}[m,n;k]$ is the cross-correlation between channels m and n for block k:

$$\text{XCORR}[m,n;k] = \sum_{i=0}^{K-k-1} x_m[i+k]x_n[i]$$

The next step is to compute GCC-PHAT values, which are a more robust version of XCORR, for each channel relative to the reference. The GCC-PHAT value, $\hat{R}^{m,ref}_{\text{PHAT}}(d)$, for time step, d, within a 10 ms segment (equivalent to one frame), between channel m and the reference ref is given by:

$$\hat{R}^{m,ref}_{\text{PHAT}}(d) = \mathscr{F}^{-1}\left(\frac{X_m(f)X_{ref}(f)^*}{|X_m(f)X_{ref}(f)^*|}\right)$$

where $\mathscr{F}^{-1}$ denotes the inverse Fourier transform, * denotes the complex conjugate, and $X(f)$ is the Fourier transformed signal. The segment length is chosen to be the same length as a frame so embeddings and TDOA/GCC-PHAT values are aligned in time. The best TDOA value for each channel in a segment is then derived as:

$$\text{TDOA}^m = \arg\max_d \hat{R}^{m,ref}_{\text{PHAT}}(d)$$

where it is the *best* TDOA value in the sense that it has been selected as the one with the most appropriate delay after pre-processing the 4 greatest GCC-PHAT values. A ‘TDOA vector’ is the vector of best TDOA values for each channel, with one vector per segment. The ‘GCC-PHAT’ vector contains the corresponding GCC-PHAT values for each channel in a segment. The reference channel always has a TDOA of 0 and a GCC-PHAT of 1 since it is being compared to itself in the calculations. For this reason, it does not need to be included in the input to DNC as it provides no additional information, hence vectors of length $M-1$ are used.

Finally, the beamformed audio stream, y, which is used to as input to the d-vector generator, is produced as:

$$y[n] = \sum_{m=1}^{M} W_m[n]x_m[n - \text{TDOA}^{(m,ref)}[n]]$$

where each n is a frame and W_m is the relative weight for microphone m so that $\sum_{m=1}^{M} W_m[n] = 1$. The weights are used to normalise the input audio streams so they have comparable amplitudes and make full use of the available dynamic range. They are calculated using windowed averages of the absolute maximum value.

Chapter 4

Implementation

A significant portion of this project has been devoted to re-implementing many of the functions in the DNC codebase in a way that would allow further developments to be made. This section discusses the changes made and new functionality that has been added. The code can be found on GitHub[1] and Figure 4.1 shows the overall pipeline of the code.

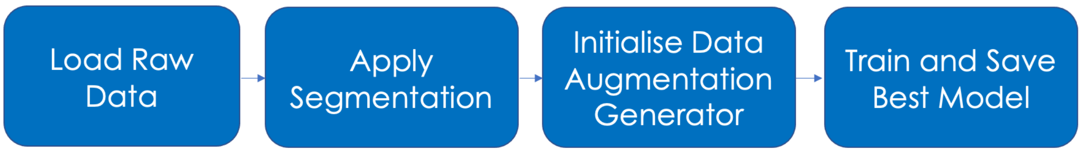

Load Raw Data
Apply Segmentation
Initialise Data Augmentation Generator
Train and Save Best Model

Figure 4.1: Overall DNC pipeline. First data is loaded from the relevant files and segmented either into speaker-homogeneous segments or fixed-length windows. A generator is initialised which produces an augmented mini-batch of data on each iteration of training. The network trains for a set number of epochs (see epoch definition in Section 4.3.1) and the model with the best accuracy on the dev set is saved. Decoding and scoring are then conducted using separate scripts.

4.1 Data Loading

The first step in the DNC pipeline is to load the relevant data. This involves some combination of speaker embeddings and TDOA/GCC-PHAT data. D-vectors are loaded using Kaldi [49] from *.ark* and *.scp* files, Wav2Vec2-based embeddings are loaded directly as Numpy arrays [50] from *.npy* files, and TDOA/GCC-PHAT vectors are taken from *.del* files produced by the BeamformIt software [48]. The vectors of the desired input types are concatenated to get the overall input vectors to DNC.

Two kinds of segmentation are explored: ‘segment-level’ and ‘window-level’. Segmenting at the segment level refers to inputting a sequence of embeddings where each embedding corresponds to a speaker-homogeneous segment. Effectively, it is assumed that Change Point

[1]https://github.com/jamest08/newDNC/

Detection has already been completed. The segmentation data is saved in *.rttm* (Rich Transcription Time Marked [51]) files where each line specifies the: meeting ID, start time, end time, start index, end index (referring to the indexing of d-vectors in *.ark* files) and a speaker label. These files are provided with the AMI dataset. It should be noted that overlap is ignored in this project and it is assumed that each segment only has one speaker.

Window-level segmentation refers to splitting each meeting into fixed-length windows. The speaker label used in training for a given window is that of the dominant speaker during the window, ie. the speaker who speaks for the most time. In the case of an exact tie, the speaker label is randomly chosen from the tied speakers. This is a more difficult task than clustering at the segment level and it will not be possible to perform perfect speaker diarisation. Even if each label is predicted correctly, there will be many windows which actually straddle multiple speakers. However, the benefit of window-level clustering is that CPD does not have to be assumed, only VAD does. With small enough windows, the issue of having multiple speakers per window should not be prohibitive.

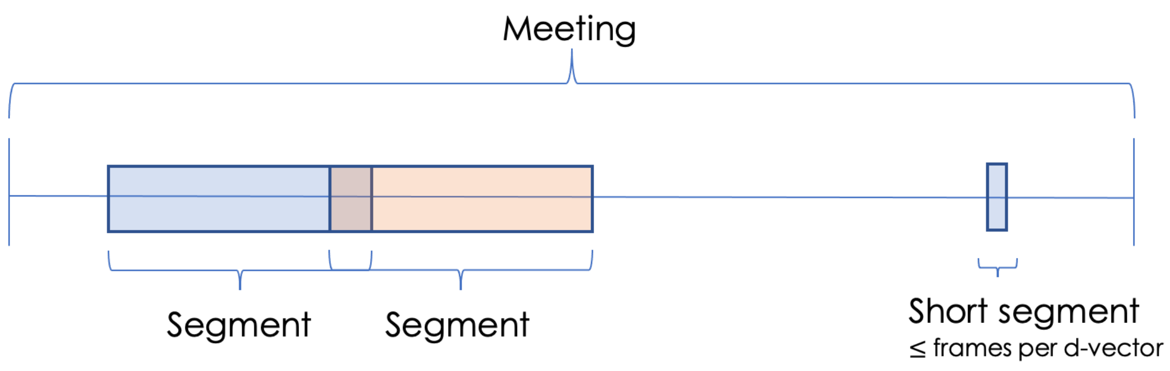
Figure 4.2 illustrates how meetings are split into segments and windows. For window-level segmentation, a new *.rttm* file is derived from the corresponding original segment-level *.rttm*. Windows are defined such that a period of voice activity (which will be called a ‘combined segment’) has its start aligned with the start of a window and its end aligned with the end of a window, with no windows specified for regions of silence. The start and end indices for a window align with its start and end times, apart from at the edges of a combined segment. In this case, the start and end times align with the start and end of the combined segment, but the indices are padded slightly to account for the window of information input to each d-vector (ie. the frames per d-vector). This avoids unwanted edge-effects at voice activity boundaries. This rule means that if windows of fixed length Δt^{des} seconds are desired, the closest that can be achieved in a given combined segment of duration T seconds, is windows of length Δt^{act} seconds. This is chosen by first considering how many full windows of the desired length could be fit into the combined segment:

$$N = \left\lceil \frac{T - fs}{\Delta t^{des}} \right\rceil$$

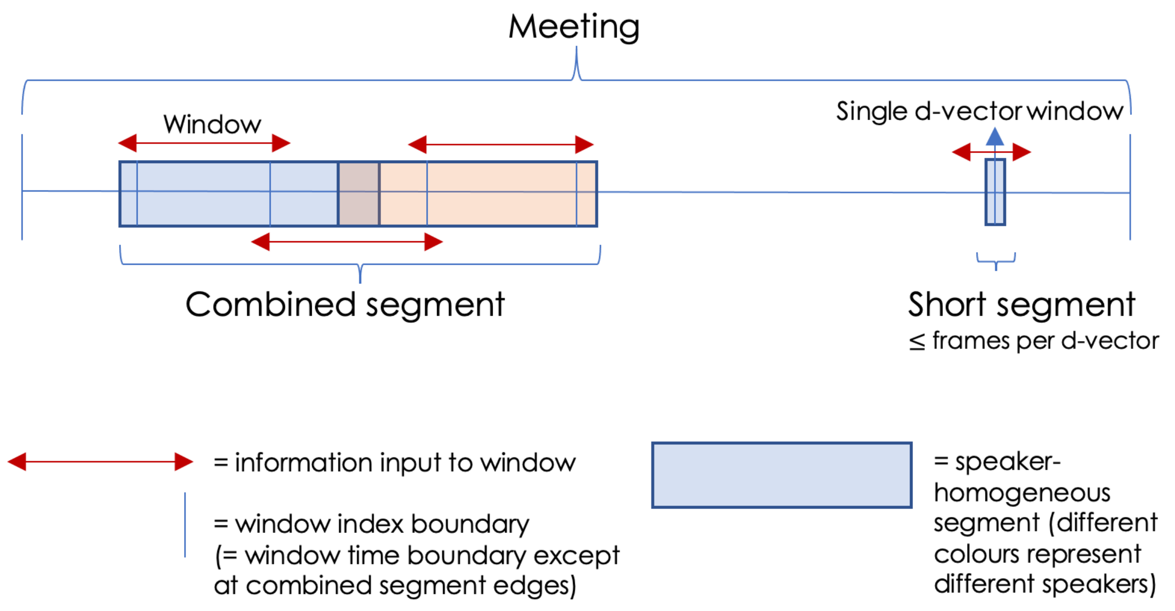
where N is the number of windows, f is the number of frames per embedding (15 or 215) and $s = 10$ ms is the time per frame. This number of windows is then used but the duration of each window is increased slightly so that the actual length of each window in seconds is given by:

$$\Delta t^{act} = \frac{T - fs}{N}$$

If the length of a combined segment is shorter than the length in time represented by one embedding ($T < fs$), only one embedding is used for the window, with the embedding at the centre of the combined segment. This occurs rarely for 15 frame d-vectors ($f = 15$) and windows of order 1 s - only 0.1% of windows are single d-vectors in the train set for a desired window size of $\Delta t^{des} = 1.5$ s. It should be noted that for both types of segmentation, speaker-homogeneous segments that are fully encompassed by another segment are filtered out.



(a) Segment-level segmentation


Meeting
Window
Single d-vector window
Combined segment
Short segment
≤ frames per d-vector
= information input to window
= window index boundary
(= window time boundary except at combined segment edges)
= speaker-homogeneous segment (different colours represent different speakers)

(b) Window-level segmentation

Figure 4.2: An example meeting divided into speaker-homogeneous segments and windows. Two of the segments overlap in time, but these are treated as distinct segments when input to DNC. The model does not attempt to label two speakers - only one is chosen. Likewise at the window-level, the model only attempts to predict the dominant speaker in each window.

4.2 Data Augmentation

A very large amount of data is required to train a Transformer model robustly, more than the available amount of real diarisation data. Therefore, several forms of data augmentation are investigated, with the procedures and assumptions for each given below. Data augmentation refers to the task of editing real data in some way to produce new data which can be added to the full set of available training examples. Augmented data must come from the same (or a similar) distribution to the real data in order for the model to learn something useful from it. This additional data is only used in training and not in validation or evaluation. Data augmentation is a common process when training neural networks, though the specific methods used will depend on the type of data in use.

Sub-meeting randomisation, input vectors randomisation and Diaconis augmentation were introduced in the original DNC paper [2]. The first two of these can be applied to the input sequence for any type and combination of input data (but not in combination with each other), while the final method applies only to speaker embeddings (and can be used in conjunction with one of the input sequence augmentation methods). Diaconis augmentation was originally intended for d-vectors but its use has been extended to Wav2Vec2-based embeddings as well. Meanwhile, average augmentation and permute augmentation are introduced in this project to augment TDOA/GCC-PHAT data (and can be used in conjunction with each other).

4.2.1 Sub-meeting Randomisation

The simplest form of augmentation used, which is effectively the bare minimum processing that can be applied to input data, is sub-meeting randomisation. This involves sampling sub-meetings of a desired length from real meetings. Hence, if the original meeting of length n was $(\boldsymbol{X}, y_{1:n})$ where $X = [\boldsymbol{x}_1, ..., \boldsymbol{x}_n]^\top$ is the input sequence and $y_{1:n}$ are the corresponding speaker labels, then a randomly chosen sub-meeting of length e would be $(\boldsymbol{X}_{s:e}, y_{s:e})$, where the start index, s, is randomly chosen. This can be applied multiple times for each available real meeting to greatly increase the size of the train set, though clearly there is a finite number of possible sub-meetings (there are 83,121 possible 50 segment sub-meetings in the train set, compared to a total of 735,000 not-necessarily-unique training examples typically required). One of the benefits of this type of augmentation is that the same speaker identity, z_i, may be assigned a different speaker label, y_i, in different sub-meetings due to the requirement that the first speaker label in a meeting (or sub-meeting) is 1. This prevents the model from associating a given input vector, x_i, with a particular speaker label.

4.2.2 Input Vectors Randomisation

Developing the idea of sub-meeting randomisation further yields input vectors randomisation. To begin with, a speaker label sub-sequence, $y_{s:e}$, is sampled but rather than keeping the corresponding input sequence, a new one is assigned. Each speaker label, y_i, in the sequence is mapped to a new speaker identity, z_i. Then for each speaker identity in the sequence, an input vector is chosen from the set of all input vectors associated with that speaker (see Figure 4.3). Speaker identities are sampled randomly without replacement (so the same speaker identity does not appear twice with different speaker labels in one sub-meeting), while input vectors are sampled with replacement (to ensure there are enough input vectors to match the length of the sampled sub-meeting). Input vectors randomisation can be conducted at the 'global-level', where any speaker identity can be chosen from across the whole training set, or the 'meeting-level', where it is restricted to speaker identities and input vectors from the same meeting that $y_{1:n}$ was sampled from. Both variants can be performed on embeddings, but when TDOA/GCC-PHAT values are involved, only meeting-level input vectors randomisation can be used since the position of microphones in the constructed meeting must be consistent.

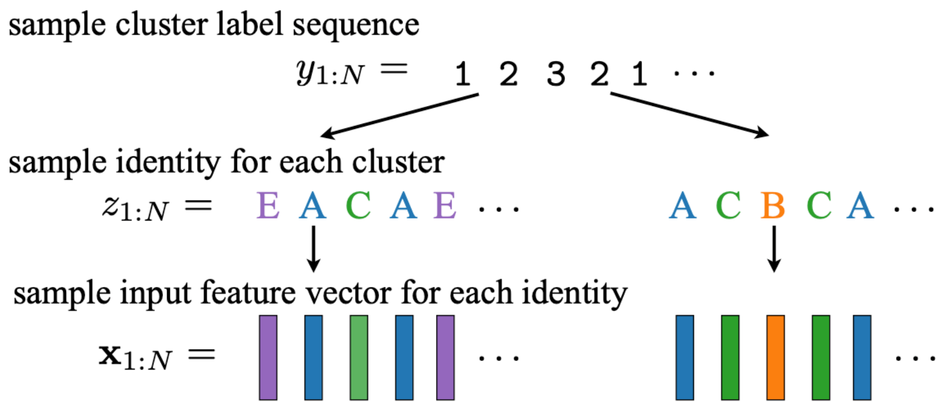

sample cluster label sequence
$y_{1:N} =$ 1 2 3 2 1 ⋯
sample identity for each cluster
$z_{1:N} =$ E A C A E ⋯
A C B C A ⋯
sample input feature vector for each identity
$\mathbf{x}_{1:N} =$

Figure 4.3: Input vectors randomisation (from [2]). In this example, two different input sequences are produced from one speaker label sequence.

4.2.3 Diaconis Augmentation

Diaconis augmentation can be applied to the speaker embedding portion of an input vector. It involves producing an entirely new input sequence, $\boldsymbol{X}'$, from a real one by post-multiplying it by a randomly sampled rotation matrix, $\boldsymbol{R}$:

$$\boldsymbol{X}' = \boldsymbol{X}\boldsymbol{R}$$

This requires the speaker embeddings to be L_2-normalised so they lie on the unit hypersphere (as is the case for d-vectors and Wav2Vec2-based embeddings, which are 32-dimensional). A meeting is rotated to a new part of the hypersphere, preserving the distance between input vectors and hence the relative position of clusters (see Figure 4.4). The name of this form of clustering is taken from Diaconis et al. who developed the algorithm for randomly sampling high-dimensional rotation matrices [52].

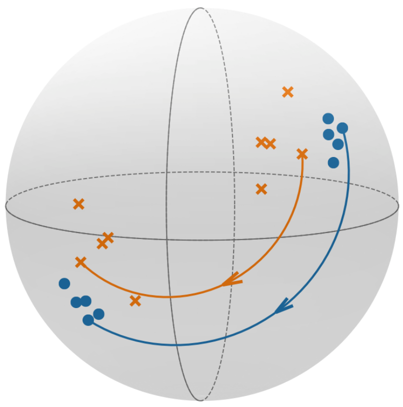


Figure 4.4: Diaconis augmentation (from [2]). An input sequence of speaker embeddings is rotated across the unit hypersphere. Different colours correspond to different speaker labels.

4.2.4 Speaker Embedding Normalisation

Speaker embeddings are L_2-normalised before applying Diaconis augmentation. It is assumed that speaker embeddings are roughly uniformly distributed on the unit hypersphere so that elements of the vector have mean 0 and variance $1/D$, where $D = 32$ is the dimension of the embeddings. Therefore, variance normalisation is applied (after Diaconis augmentation, if it occurred) by multiplying each embedding by $\sqrt{32}$. The same variance normalisation is applied to the test set during evaluation in the same way as for the train set.

4.2.5 Average Augmentation

The idea behind average augmentation is to add noise to TDOA or GCC-PHAT values in order to generate new data. For both types of segmentation, input vectors (which are available at every 10 ms frame) are averaged across the segment so there is one vector per segment. The average used is the mean: $\frac{1}{N}\sum_{i=1}^{N}\boldsymbol{x}_i$ for N vectors in the segment. One way of adding noise is to use only a subset of the available TDOA/GCC-PHAT vectors in the average. The proportion of vectors used in the average, p, can be chosen to determine the amount of noise. If $p = 1$, all available vectors are used and effectively no augmentation is performed. As p tends towards 0.5, the number of possible averages increases, with the maximum at $p = 0.5$, since the number of combinations from n available input vectors for a choice of r vectors, (C_r^n), is maximised by $r = \lfloor n/2 \rfloor$. For this form of augmentation to work, it is desired that ‘noisy’ averages are similar enough to the true average to be representative of the input data for the whole segment and so map to the correct speaker label, while being different enough to provide a useful new training example.

4.2.6 Permute Augmentation

Permute augmentation is the process of permuting the values in TDOA or GCC-PHAT vectors consistently across a meeting. The permutation is chosen at random, uniformly across all possible permutations, then the values in each vector are swapped with each other accordingly. This effectively corresponds to changing the labelling of the microphones in a meeting, so that if a microphone previously corresponded to the ith channel (and ith element of each TDOA/GCC-PHAT vector), it might now correspond the the jth channel (and jth input vector). The assumption is that this would form a valid set of directional data associated with a meeting, providing a non-trivial new example for DNC to learn from.

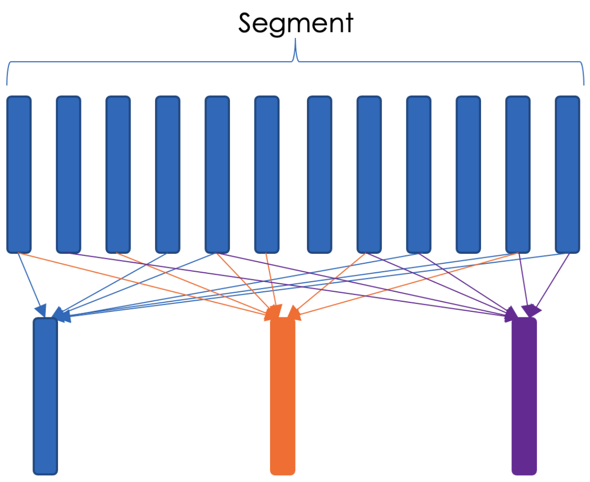


Figure 4.5: Average augmentation. This example shows three different TDOA/GCC-PHAT vectors produced for a segment by randomly averaging $p = 50\%$ of the available vectors for the segment.

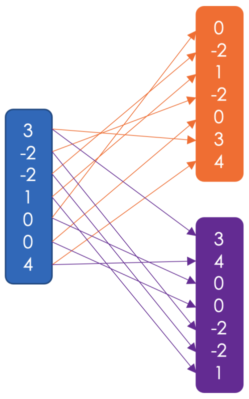


Figure 4.6: Permute augmentation. This example shows two different TDOA vectors produced from an original TDOA vector by permuting the values. The same principle can be applied to GCC-PHAT vectors. The permutation must be consistent for all TDOA and GCC-PHAT vectors in the meeting.

4.2.7 TDOA/GCC-PHAT Normalisation

As well as augmentation, it is also proposed to normalise TDOA and GCC-PHAT vectors to aid with training. This involves first empirically calculating the mean, μ_j, and variance,

$\sigma_j^2 = \frac{1}{N}\sum_{i=1}^{N}(x_{i,j} - \mu_j)^2$, of each element, j, of TDOA and GCC-PHAT vectors from the N examples in the train set. Then each element of the vector is normalised by subtracting the mean and dividing by the standard deviation:

$$x_{i,j} \leftarrow \frac{x_{i,j} - \mu_j}{\sigma_j}$$

Normalisation is performed before augmentation. The same statistics are used for normalising vectors from the evaluation set during testing as for the train set during training.

4.3 ESPnet

Discriminative Neural Clustering is implemented using ESPnet, a speech processing platform developed by Watanabe et al. [53]. ESPNet itself makes use of the PyTorch toolkit [54]. The platform provides a Transformer class which can be edited to suit different tasks: originally ESPnet focused on Automatic Speech Recognition (ASR). The main interaction with ESPnet in this project has been editing the iterator to allow data augmentation to take place during training, rather than beforehand. It would also be possible to design a curriculum learning scheduler which would allow the meeting length to be gradually increased during training, though this was not deemed necessary in this project as discussed in Section 6.1.

4.3.1 On-The-Fly Augmentation

In the original implementation of DNC, data had to be augmented as a pre-processing stage and stored as a large batch which would be iterated through in training. This is both inconvenient and has a large storage cost (over 1 Gb when each real training example is augmented 5000 times for 50 segment sub-meetings; this number quickly becomes very large as the sub-meeting length increases). This storage cost was the reason the original authors chose d-vectors of 32 dimensions since higher dimensions would incur a prohibitively high cost. In this project, data augmentation is implemented 'on-the-fly'. DNC reads in the real data and then augments it dynamically for each new mini-batch by using a generator function. This has some implications for training regarding the definitions of epochs and batches.

In the original DNC, one 'epoch' referred to one pass through the whole batch (ie. all of the already augmented data). Since the whole batch is not saved in the new implementation, and each mini-batch is produced on-the-fly, there is no 'batch' to pass through. Therefore, the new definition of epoch is one pass through a set number of mini-batches. The mini-batches per epoch can be specified as a parameter and is chosen so that the number of training examples per epoch is the same in the new implementation as the old. The main difference is then that, while each epoch in the old version passed through the same batch of data, in the new implementation, each epoch is different, since each mini-batch is produced randomly and independently from previous mini-batches. This should not make a difference for sub-meeting randomisation (where the finite number of possible sub-meetings is less than the total number of training examples) but will affect the other forms of augmentation (that give a far greater number of possible augmented meetings). The new implementation will hence enable a wider range of training examples to be seen and hopefully lead to better results. 32-dimensional embeddings continue to be used in this project to enable direct comparisons with results from the original paper, but it would now be easy to use higher-dimensional embeddings, whereas before this was not possible.

4.4 Spectral Clustering

The baseline results in this project are taken from spectral clustering. This uses the implementation developed by Wang et al. [5] and used in the original DNC paper. Data is loaded using the new functions built for this project. No augmented data needs to be used since spectral clustering does not require training.

4.5 Scoring

An essential step in the development of speaker diarisation is the ability to test and score a model, allowing comparisons to be made between different techniques. Any permutation of speaker labels is valid, and so the one which gives the highest accuracy is used for scoring. The key metric is the percentage Diarisation Error Rate (DER), which is the total duration of time attributed to the wrong speaker [55]. Regions of overlapped speech are not included. The DER is given by [56]:

$$\text{DER} = \frac{\phi_{fa} + \phi_{miss} + \phi_{err}}{\phi_{total}}$$

where ϕ_{total} is the total duration of the reference, ϕ_{fa} is the duration of 'false alarm' time (where a region of silence is given a speaker label), ϕ_{miss} is the duration of 'missed speech' time (where no speaker label is given for a region of speech), and ϕ_{err} is the duration of 'confusion' time, where an incorrect speaker label is given. Throughout the project, manual VAD is assumed, and so it is always true that $\phi_{fa} = \phi_{miss} = 0$ (if this were not the case it would indicate a bug in the code). Therefore, in this project, DER = SER, where the Speaker Error Rate, SER, is simply given by:

$$\text{SER} = \frac{\phi_{err}}{\phi_{total}}$$

The NIST scoring script is used for evaluation as in the original paper. The collar size is 250 ms - this is the amount of leeway permitted when checking predicted speaker labels against the reference. Several additional scripts had to be written to enable decoding and scoring to work at the window-level, and for different combinations of inputs.

Accuracy, as defined in Section 3.5, is sometimes used to supplement analysis. Unlike DER, this is not time weighted and simply counts the number of correctly labelled input vectors (one per segment or window), and reports it as a percentage of the total number of segments/windows.

Chapter 5

Results and Discussion

Experiments were conducted on the AMI dataset. It is the DER on the evaluation set that is of most interest - train and validation accuracies as well as validation DERs were also computed, though not always reported. Comparisons are made with spectral clustering as a baseline, as well as with results from the original DNC paper where possible.

The experimental strategy was as follows. Experiments began by investigating segment-level clustering with d-vectors and short sub-meetings. The six possible combinations of augmentation were tested and this is directly comparable with Table 2 in the original DNC paper [2]. Next, the best augmentation settings were retained and used to test four different window lengths for window-level clustering. The six augmentation methods were also investigated for two of the window lengths. Following this, different combinations of TDOA/GCC-PHAT inputs and corresponding augmentation were experimented with. Experiments were also performed using Wav2Vec2-based embeddings, though in less depth than the d-vectors. Finally, curriculum learning was used to train DNC to handle longer and eventually full meetings for different settings of inputs and augmentation.

The same settings of hyperparameters from the experiments in the original paper were used (8 for the learning rate and 20,000 for the number of warm-up steps). Once progressing to window-level clustering, the learning rate and number of warm-up steps were varied to try to tune the hyperparameters, though the original values were the best and were used throughout most experiments in this project. It is worth noting that the performance of DNC is quite sensitive to changes in the learning rate parameters. If they deviated too much from the optimum settings used, DNC would effectively learn nothing and make the trivial and useless prediction that the same speaker was speaking all the time.

5.1 AMI Dataset

The multi-modal AMI Meeting Corpus was developed by Carletta et al. [9] in 2005, 'dedicated to the research and development of technology that will help groups interact better'. One of these technologies is speaker diarisation, which is a step towards enabling automatic meeting transcription, though other uses of the dataset include training ASR and video-processing. Although video data from meetings is available, this project is only concerned with audio data. It is more challenging to perform speaker diarisation on the AMI dataset than some others such as CALLHOME as explained in Section 2.2.

The dataset contains 169 original meetings which are split into three datasets called ‘train’, ‘dev’ and ‘eval’ as detailed in Table 5.1. Hence, the term ‘dev set’ is used interchangeably with ‘validation set’. In the original DNC paper and this project, DNC is restricted to handling a maximum of four speakers per meeting. There are some meetings in the train set which have five speakers per meeting and so each of these was split into five meetings of four speakers by removing the inputs associated with one of the speakers each time - this expanded the train set to 147 meetings as quoted in the table.

	#meetings	mean duration (min)	#speakers
train	147	37.9	155
dev	18	32.3	21
eval	16	34.0	16

Table 5.1: Details of meetings in the AMI Corpus

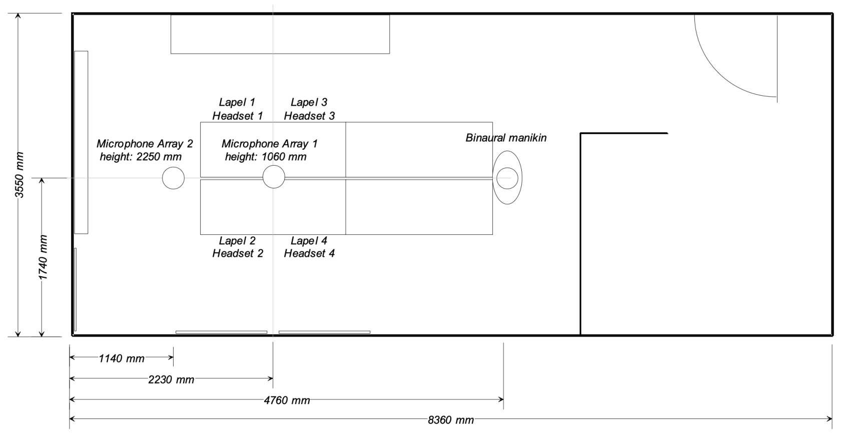


Figure 5.1: The ‘IDIAP Room’, one of three rooms used to record meetings in the AMI Corpus (from [9]). The positions of speakers are shown around Microphone Array 1 which records the audio used as input to the d-vector generator.

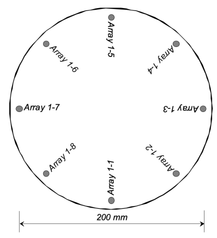


Figure 5.2: An array of 8 microphones (from [9]). Each provides a different channel used in the input to Beamforming for d-vectors and for the computation of TDOA and GCC-PHAT vectors.

5.2 Segment-Level Clustering

The first experiments performed were for speaker-homogeneous segments in order to allow direct comparison with the original DNC results, and later with window-level clustering.

5.2.1 Baseline

Performing spectral clustering at the segment-level for sub-meetings of 50 segments achieved a DER of **15.66%** and an accuracy of 68.48% on the eval set. The number of clusters was allowed to be between 2 and 4, and the hyperparameters were tuned on the dev set. A Gaussian blur of 0.1 and a p-percentile of 0.94 were used for all spectral clustering experiments throughout the project. Some augmented data were also input to spectral clustering, to check if a similar accuracy was achieved. This was the case, and implies that augmented data come from a similar distribution to the real data, which is vital for training DNC.

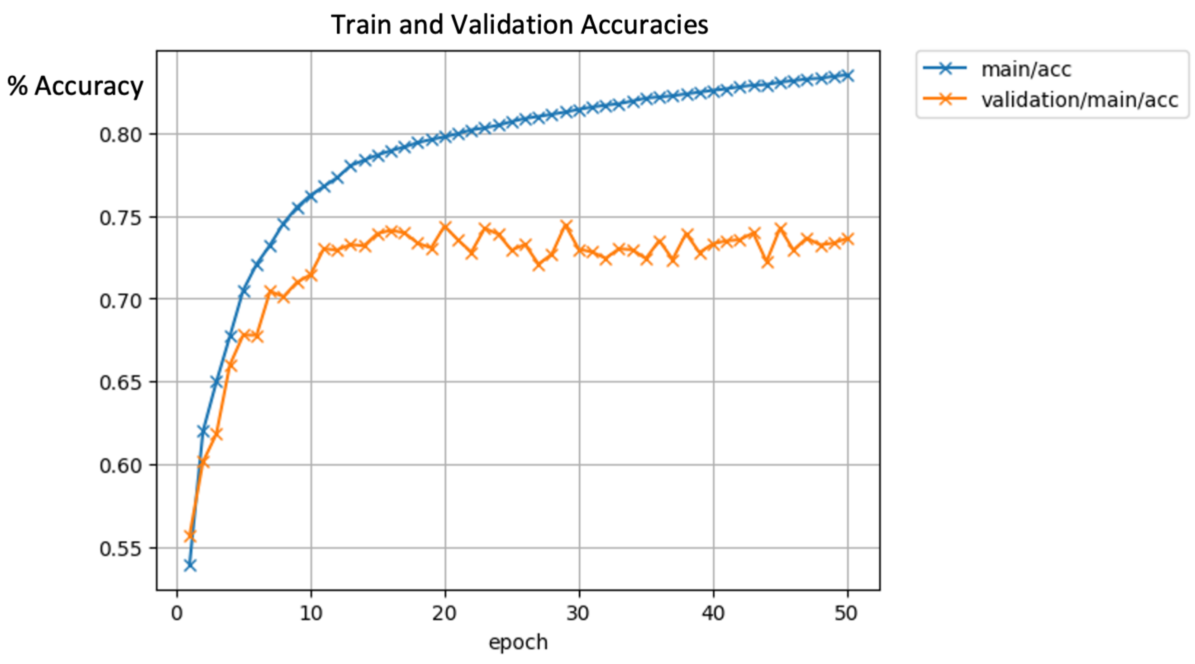
5.2.2 d-vector Augmentation (Segment-Level)

The results for segment-level clustering of 50 segment sub-meetings for different combinations of augmentation methods using the new implementation of DNC are shown in Table 5.2. Training was run for 50 epochs with 2940 mini-batches per epoch and a mini-batch size of 50, giving the same number of iterations as the original DNC results (but more distinct training examples due to ‘on-the-fly’ augmentation). Table 5.3 shows the corresponding results from the original DNC paper. ‘None’ refers to sub-meeting randomisation, ‘meeting’ refers to meeting-level input vectors randomisation and ‘global’ refers to global-level input vectors randomisation.

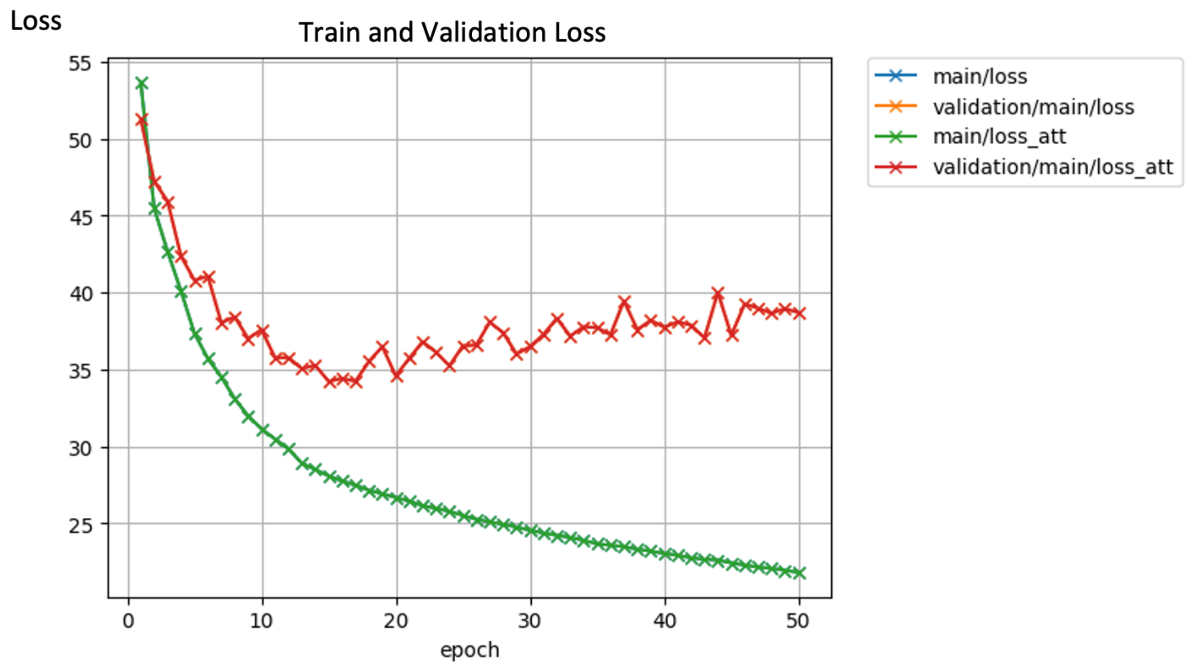
Figure 5.3 shows the training curves for the new implementation experiment of sub-meeting randomisation and Diaconis. The curves have mostly converged after about 20 epochs. Beyond this point, the train accuracy continues to increase (with the train loss decreasing) while the dev set values stay constant or worsen. It is for this reason that the model with the highest validation accuracy in training is used for evaluation, rather than the final model at the end of training.

Figure 5.4 visualises clustering for an example sub-meeting of 50 segments from the eval set. The different colours corresponds to different speaker labels and the 32-D d-vectors are reduced to 2-D embeddings using t-SNE [37]. Spectral clustering only assigns two clusters, whereas DNC assigns four which is the correct number of speaker labels. The plots give an example of how DNC can draw more sophisticated and subtle cluster boundaries than spectral clustering.

Comparing the two tables, they have slightly different DERs but a similar pattern between the augmentation methods. The main reason for the slight discrepancies is that DER is noisy, and can be quite different even for the same accuracy score since it is time weighted and so depends on the duration of correctly classified segments. The Diaconis results from the new implementation were often better than from the old implementation. A possible reason for this is that, in the old implementation, an entire mini-batch is rotated using the same rotation matrix, however in the new implementation, a new rotation matrix is randomly sampled for each meeting

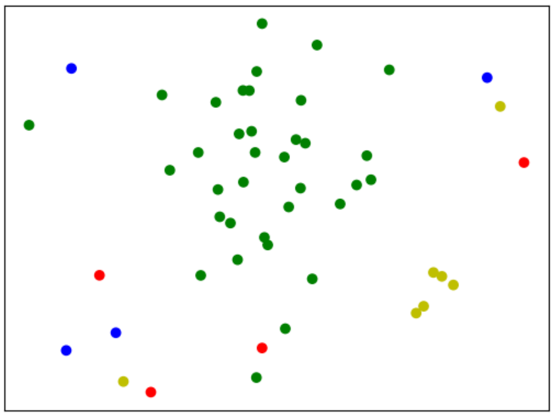


(a) Train Set (blue) and Validation Set (orange) Accuracies

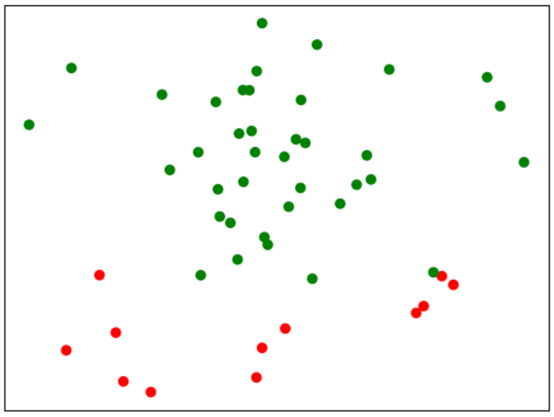


(b) Train Set (green) and Validation Set (red) Loss

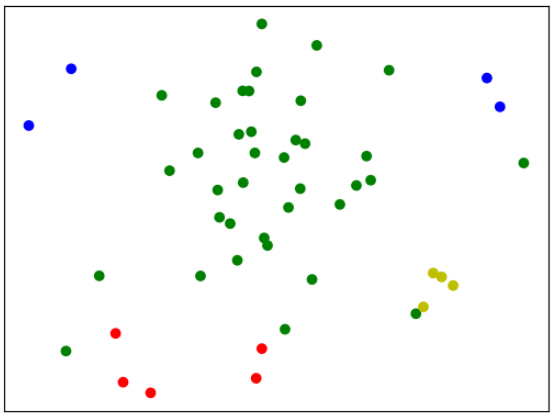
Figure 5.3: Training curves for DNC at the segment-level for sub-meetings of 50 segments using sub-meeting randomisation and Diaconis augmentation. The loss function is the cross-entropy loss.



(a) Ground Truth



(b) Spectral Clustering



(c) Discriminative Neural Clustering

Figure 5.4: Visualising the clustering of an example sub-meeting of 50 segments from the eval set. Different colours correspond to different speaker labels. The 32-D d-vectors are reduced to 2-D embeddings using t-SNE (see Section 3.6.4).

Randomisation	w/o Diaconis	w/ Diaconis
None	24.53	**12.71**
Global	20.41	18.74
Meeting	24.72	14.16

Table 5.2: % DERs from the **new** DNC implementation for different augmentation techniques with meeting length of 50 segments on the eval set. Using 215 frame d-vectors. The SC baseline is **15.66%**.

Randomisation	w/o Diaconis	w/ Diaconis
None	20.19	15.25
Global	14.47	19.80
Meeting	23.03	13.57

Table 5.3: % DERs from the **old** DNC implementation for different augmentation techniques with meeting length of 50 segments on the eval set. Using 215 frame d-vectors. The SC baseline is **15.66%**.

in the mini-batch. Therefore, in the new implementation, DNC can be exposed to a greater diversity of rotated meetings, covering more of the unit hypersphere, leading to better training. The results without Diaconis were generally slightly worse for the new implementation. It was initially thought that this may be due to a difference in how the d-vectors are processed - in the old implementation they were L_2-normalised before and after averaging across a segment but in the new implementation they are only normalised after averaging - however further experiments showed that this makes no difference. Since later experiments always used Diaconis, this discrepancy does not matter very much and so was not investigated in great detail. The fact that some Diaconis results were better than previously was sufficient to move on to novel experiments.

Comparing the randomisation types for the new implementation (Table 5.2), sub-meeting randomisation outperforms input vectors randomisation. This suggests that input vectors randomisation produces atypical, but not useless data. Furthermore, global-level randomisation performs worse than meeting-level, since this is a further step away from the original data. Sampling embeddings from different meetings may make clustering more difficult as conditions are different in different meeting rooms and with different microphones. For short sub-meetings, there are enough possible training examples from sub-meeting randomisation (which preserves the order of embeddings in meetings) that input vectors randomisation is not necessary. The most important consideration from this set of experiments is that Diaconis is a very useful form of augmentation and can provide many new useful training examples, leading to far better results than without. The best result from the new DNC implementation (**12.71%**) outperformed spectral clustering by a relative 18.8%. Hence, clearly DNC is a useful improvement on unsupervised clustering methods for meetings of 50 segments.

5.3 Window-level Clustering

Having established sub-meeting randomisation with Diaconis as the best augmentation method, experiments proceeded to use window-level clustering, which is novel and was not attempted in the original DNC paper. It is a much harder task than segment-level clustering since CPD is not assumed and meetings are longer in terms of number of windows (since windows are defined to be shorter than segments).

5.3.1 Window Lengths (d-vectors)

Table 5.4 shows the results for window-level clustering on different fixed window lengths for the Spectral Clustering (SC) baseline and Discriminative Neural Clustering (DNC). Again, short sub-meetings were used. The sub-meeting length in each case was chosen to be the number of windows such that, on average, the length was equivalent to 50 speaker-homogeneous segments. This allows comparisons to be made between the window lengths and with segment-level clustering on 50 segment sub-meetings. The average duration of a speaker-homogeneous segment in the train set is 3.0187s and so the window-level meeting lengths were calculated as:

$$\text{meeting length in windows} = \frac{50 \times 3.0187}{\text{window length in seconds}}$$

The speaker embeddings were 15 frame d-vectors. The transition from 215 frame d-vectors at the segment-level to 15 frame d-vectors at the window-level was made to reduce the overlap of information between windows, minimising unwanted edge-effects.

Window length (s)	Meeting length (#windows)	DNC	SC
0.5	302	25.44	25.57
1.0	151	20.20	21.24
1.5	101	**19.54**	19.73
2.0	75	19.64	18.42

Table 5.4: % DERs from Spectral Clustering (SC) and Discriminative Neural Clustering (DNC) on the eval set for different window lengths with meeting lengths equivalent to 50 speaker-homogeneous segments on average. Using 15 frame d-vectors with sub-meeting randomisation and Diaconis.

First, comparing DNC and the spectral clustering baseline, DNC performed slightly better than SC for every window length except 2.0s. A reason for this could be that, for longer window lengths, it is more likely that a window will actually contain multiple speakers, with only the dominant speaker labelled. This means that when the embeddings for the window are averaged, it is actually averaging embeddings from two different speakers, which by definition should be quite different. The more of these there are in the train set, the harder it will be for DNC to train, since the averaged embedding will not be closely associated with the dominant labelled speaker. This could cause DNC to learn spurious mappings, which cause a worse result in evaluation for windows with only one speaker, not just those where there are actually multiple speakers. Meanwhile, for spectral clustering, there is no training phase and so, even if there are examples of this type of window in the eval set which it fails to cluster accurately, there will be no effect on windows with only one speaker.

Comparing the DNC results to each other, the best DER was for a window of 1.5s. At first, the DER gets better with increasing window length and then worsens. This is because there are two competing factors affecting clustering. When the window length is small, more windows are required to cover the same amount of time. Therefore there are more data points to cluster, which is harder. If it were assumed that windows always contain just one speaker and the window length were made very large, the required meeting length in windows would be very small and it would be very easy to perform clustering. However, this is clearly not the case. The competing factor is that as the window length increases, there will be a greater proportion of windows which actually cover multiple speakers and are hence difficult to cluster (as used to explain SC outperforming DNC for a window length of 2.0s). If experiments were performed with longer window lengths, it would expected that the DER for DNC would continue to worsen. For as long as the 'multiple speaker per window' problem is more of a problem in training than evaluation, SC will continue to perform well, but eventually the data points will not be representative of the underlying speaker labels and the DER for SC will quickly deteriorate.

For DNC, there is a trade-off for which the optimum window length is 1.5s. It is this window length which continued to be used for subsequent experiments at the window-level.

5.3.2 d-vector Augmentation (Window-Level)

Before moving on and focusing on just one window-length, the set of experiments from Section 5.2.2 were repeated at the window-level for two different window lengths, 1.0s (Table 5.5) and 1.5s (Table 5.6).

Randomisation	w/o Diaconis	w/ Diaconis
None	30.55	**20.20**
Global	25.92	30.67
Meeting	27.32	21.10

Table 5.5: % DERs from **1.0s** window-level DNC for different d-vector augmentation techniques with meeting length of 151 windows on the eval set. Using 32-D 15 frame d-vectors. The SC baseline is **21.24%**.

These results confirm the patterns found in Sections 5.2.2 and 5.3.1. Comparing the two tables, clustering 1.5s windows outperformed clustering 1.0s windows for every data point. This provides confidence that clustering using windows of 1.5s is the best decision.

Furthermore, better results were always achieved using Diaconis, apart from for global-level input vectors randomisation. This is the same observation the original DNC paper found, although it was not the case for this project's implementation at the segment level (perhaps

Randomisation	w/o Diaconis	w/ Diaconis
None	22.55	19.54
Global	22.48	29.85
Meeting	21.06	**16.92**

Table 5.6: % DERs from **1.5s** window-level DNC for different d-vector augmentation techniques with meeting length of 101 windows on the eval set. Using 32-D 15 frame d-vectors. The SC baseline is **19.73%**.

only due to the noisiness of the DER metric). A reason for this is that there are already very many training examples available from global-level input vectors randomisation and so Diaconis augmentation (which cannot perfectly match the original embedding distribution) is not necessary and so actually worsens performance.

The only other difference in the pattern of augmentation results is that, for 1.5s windows, meeting-level input vectors randomisation outperformed sub-meeting randomisation (both with Diaconis), which was not observed at the segment-level or 1.0s window-level. Despite this, if the validation accuracies for 1.5s windows are compared, it is found that the accuracy for sub-meeting randomisation with Diaconis (84.17%) was higher than for meeting-level input vectors randomisation with Diaconis (82.99%). This was also true at the segment-level and 1.0s window-level, justifying the continued use of sub-meeting randomisation as the superior method of input sequence augmentation.

While the improvement on spectral clustering is not as great for window-level clustering as previously, DNC continued to be investigated due to its ability to handle multiple input types, which can far outperform SC, as the following sections demonstrate.

5.3.3 TDOA/GCC-PHAT Vectors Augmentation

At this stage, TDOA vectors and GCC-PHAT vectors were introduced as novel inputs to DNC. The first step was to experiment with the proposed forms of TDOA/GCC-PHAT augmentation without including speaker embeddings yet. The results are given in Table 5.7.

Input Vector	w/o Aug	w/ Average Aug	w/ Permute Aug	SC
TDOA	**6.79**	7.11	7.12	16.15

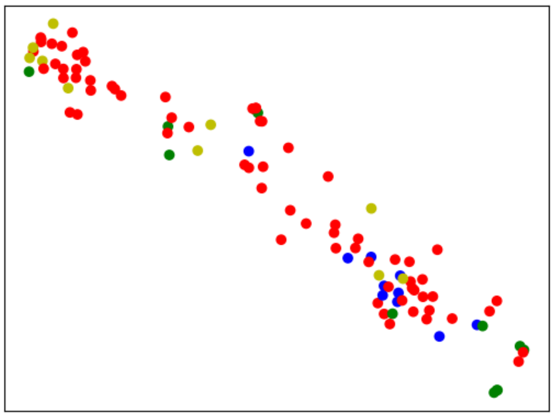
Table 5.7: % DERs from DNC for different TDOA augmentation techniques with meeting length of 101 1.5s windows on the eval set. Using 7-D un-normalised TDOA vectors and $p = 0.7$ for average augmentation. Sub-meeting randomisation is used. SC result given for comparison.

The first observation to be made is that, with just one 7-D TDOA vector per speaker label in the input sequence (and no augmentation) DNC performed exceptionally well. DNC got a DER of **6.79%** compared to only 16.15% for SC. This is a very large difference and is far more significant than the difference between DNC and SC with d-vectors as input. The improvement in DNC going from d-vectors to TDOA is also much larger than the improvement in SC making the same change. It is now that the full power of DNC as a supervised clustering method begins to be revealed. Whereas d-vectors are designed to be close in the embedding space (according to cosine similarity) for the same speaker label, enabling traditional clustering methods to work, TDOA vectors do not have this quality. They are closer to raw data, having not been previously processed by a neural network, and require a different similarity measure. TDOA vectors can vary over time for a given speaker label as speakers are allowed to move position throughout a meeting. DNC is able to learn the meaning of the directional data provided by TDOA vectors, indicating the location of a speaker over a period of time, whereas SC cannot. This is illustrated by Figure 5.5 which visualises clustering TDOA vectors for a particular sub-meeting. The fact that t-SNE itself struggles to separate clusters into different regions of space suggests that ‘similar’ TDOA vectors do not necessarily lie close to each other. The struggle of spectral clustering, which performed much worse than DNC, can be seen in Figure 5.5.

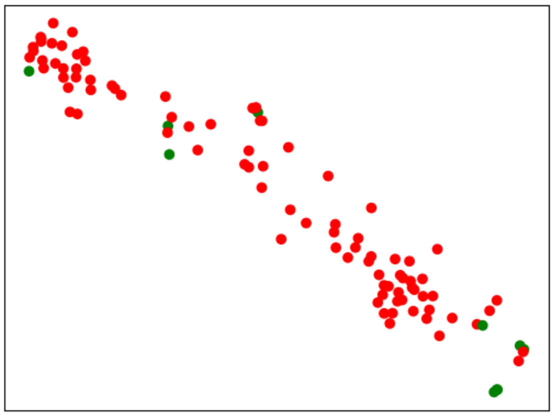
Inspecting the results for different augmentation methods, it appears that augmenting the data gives very similar, though slightly worse DERs. The reason for this is likely that the new training examples produced by augmentation are too similar to the existing training examples, and so nothing new is learnt. In the case of average augmentation, TDOA vectors change very little across a window and so taking averages of subsets of the vectors in a window is likely to give the same, or a very similar, result. This means that the new training examples produced are not of much extra use compared to the un-augmented versions. The result for permute augmentation is more surprising - it suggests that it is not sufficient to change the order of TDOA values in a vector to give a useful new training example. It implies that for an augmented vector to be useful, it would have to contain different values, not just the same values in a different order, however accurately manufacturing TDOA vectors with new values would be challenging. Since it is clear that these types of augmentation have no useful effect, it was decided not to experiment with augmentation on GCC-PHAT vectors (which contain similar information to TDOA) or combinations of TDOA and GCC-PHAT. Therefore, TDOA/GCC-PHAT augmentation was not used and it is sufficient that, even without augmentation, TDOA vectors perform very well on DNC.

It is worth noting that some experiments were also done using TDOA normalisation as described in Section 4.2.5 (whereas the results in Table 5.7 are for un-normalised TDOA values). It was quickly found that using normalisation was not necessary and led to worse results, and so it was not pursued any further. It is likely that TDOA values were already in a ‘useful’ range for DNC to be able to deal with.

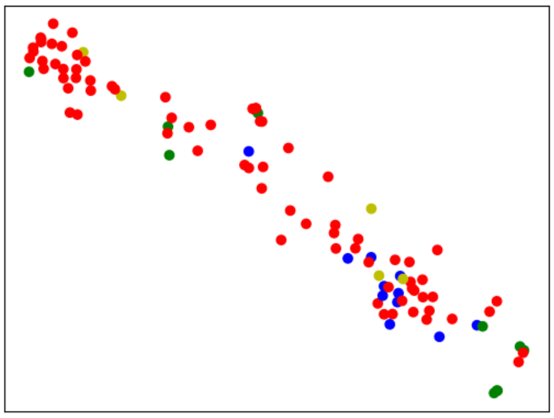
The main take-away from this set of experiments is that TDOA values are a highly useful input which allow far more accurate diarisation than using speaker embeddings for AMI meetings. The caveat is that these values require multiple microphones to be used (8 in this case), whereas speaker embeddings can be produced from a single channel. In real-world meeting rooms, this may feasible to supply, but for diarisation elsewhere it is unlikely to be available.



(a) Ground Truth



(b) Spectral Clustering



(c) Discriminative Neural Clustering

Figure 5.5: Visualising the clustering of TDOA vectors for an example sub-meeting of 101 windows from the eval set. Different colours correspond to different speaker labels. The 7-D TDOA vectors are reduced to 2-D embeddings using t-SNE (see Section 3.6.4).

5.3.4 TDOA/GCC-PHAT Vectors and d-vectors

Having determined that no TDOA/GCC-PHAT augmentation is necessary, experiments were done with different combinations of inputs. The input vector consisted of the chosen input types concatenated together, always in the order: embedding, TDOA, then GCC-PHAT. Again, windows of 1.5s were used with sub-meeting randomisation, and Diaconis is applied where relevant, as shown in Table 5.8.

Input Vector	Augmentation Techniques	DNC	SC
TDOA	None	6.79	16.15
GCC-PHAT	None	28.23	31.91
TDOA + GCC-PHAT	None	6.87	16.16
d-vector + TDOA	Diaconis	6.16	12.34
d-vector + GCC-PHAT	Diaconis	16.01	19.41
d-vector + TDOA + GCC-PHAT	Diaconis	**5.85**	12.43

Table 5.8: % DERs from DNC for different combinations of inputs using best augmentation techniques with meeting length of 101 1.5s windows on the eval set. Diaconis, when applied, is only on the d-vector part of the input vector. Sub-meeting randomisation is applied in all cases. SC is given for comparison (for which augmentation techniques do not apply).

The first row in the table is repeated from Table 5.7. Given the success of using TDOA, the GCC-PHAT result is surprising - it achieved a very poor DER of 28.23%. The SC result for just GCC-PHAT was also very high, which suggests that the high DER for DNC is not a consequence of poor training, but that GCC-PHAT vectors are inherently hard to cluster. Given that TDOA values are derived from GCC-PHAT values (see Section 3.8), it is surprising that DNC can infer little useful information from them. Meanwhile, the experiment with TDOA and GCC-PHAT values concatenated performed about the same as TDOA on its own for both SC and DNC. In the spectral clustering case, during the dimensionality reduction phase it is likely removing the GCC-PHAT data, and for discriminative neural clustering it is simply learning to focus on the TDOA portion. Although the DER was slightly higher for TDOA+GCC-PHAT compared to just GCC-PHAT (6.87% compared to 6.79%), the validation accuracy was also higher (85.46% compared to 84.79%). This corroborates the findings in other experiments that the addition of GCC-PHAT can yield better results.

Experiments were then conducted on d-vectors in combination with TDOA/GCC-PHAT. For d-vector + TDOA, the DER was slightly improved compared to just TDOA (from 6.79% to 6.16%), implying that most of the useful information comes from TDOA but that the d-vector part still provides some benefit. The d-vector + GCC-PHAT result (16.01%) was greatly improved compared to just GCC-PHAT (28.23%) since, as discussed, DNC does not infer much of use from GCC-PHAT, so the addition of d-vectors is significant. The result for just d-vectors (see Table 5.6) was 19.54%, which means that GCC-PHAT must help somewhat. The best result was using all of the available inputs: d-vector + TDOA + GCC-PHAT achieved **5.85%** DER. As seen previously when TDOA vectors are involved, this is a large improvement on spectral clustering, which only achieves 12.43%.

5.3.5 Wav2Vec2-based Embeddings

The final input type to be experimented with was Wav2Vec2-based embeddings. This is another type of speaker embedding which replaces d-vectors. Preliminary experiments were conducted at the segment level for 50 segment sub-meetings to check that the same functions that had been used on d-vectors also worked on the Wav2Vec2-based embeddings (see Table 5.9). The best segment-level result was for sub-meeting randomisation with Diaconis augmentation which got **9.26%** DER on the eval set. This was an improvement on the equivalent d-vector result and, as expected, was better than spectral clustering. The fact that the spectral clustering result for Wav2Vec2-based embeddings was better than the corresponding d-vector result suggests that Wav2Vec2-based embeddings are inherently better, with clearer cluster boundaries.

Input Vector	Augmentation	DNC	SC
d-vector	Sub-meeting w/o Diaconis	24.53	15.66
d-vector	Sub-meeting w/ Diaconis	**12.71**	15.66
Wav2Vec2	Sub-meeting w/o Diaconis	30.83	13.05
Wav2Vec2	Sub-meeting w/ Diaconis	**9.26**	13.05

Table 5.9: % DERs from DNC at the **segment level** for Wav2Vec2-based embeddings with a meeting length of 50 segments on the eval set.

This project is most interested in window-level clustering and so most of the Wav2Vec2 experiments were done at the window level. These experiments combined Wav2Vec2-based embeddings with TDOA/GCC-PHAT vectors for 1.5s windows and meeting lengths of 101 windows so that Table 5.10 for Wav2Vec2-based embeddings is directly comparable to Table 5.8 for d-vectors. It should be noted that the learning rate had to be reduced from 8 to 2 for DNC to train successfully on Wav2Vec2-based embeddings at the window level.

Input Vector	DNC	SC
Wav2Vec2	10.91	17.63
Wav2Vec2 + TDOA	**7.10**	16.02
Wav2Vec2 + GCC-PHAT	8.11	11.60
Wav2Vec2 + TDOA + GCC-PHAT	7.22	16.09

Table 5.10: % DERs from DNC at the **window level** for Wav2Vec2-based embeddings and different combinations of TDOA/GCC-PHAT vectors using sub-meeting randomisation and Diaconis augmentation (only on the Wav2Vec2 part) with a meeting length of 101 1.5s windows on the eval set. SC shown for comparison.

One striking observation is that clustering Wav2Vec2-based embeddings at the window level did not increase the DER by much compared to the segment level (going from 9.26% to 10.91%). Meanwhile, for d-vectors it had been found that the transition to the window level caused the DER to go from 12.71% to 19.54%. At first, this suggests that Wav2Vec2-based embeddings may be the best choice for clustering at the window level.

Comparing the DNC results to the SC results, DNC performed much better when TDOA/GCC-PHAT vectors were involved, in line with previous results. However, there was one anomaly that SC performed uncharacteristically well on Wav2Vec2 + GCC-PHAT - much better than just on Wav2Vec2-based embeddings. This spurious result is not focused on since all other evidence suggests that GCC-PHAT does not generally help clustering.

Comparing the DNC results in Table 5.10 with each other, there is a similar pattern to what was seen with d-vectors + TDOA/GCC-PHAT. Though interestingly, the best result (**7.10%** for Wav2Vec2 + TDOA) was not actually any better than just TDOA (6.79%). This enforces the observation that TDOA inputs dominate in DNC, but it is a shame that Wav2Vec2-based embeddings cannot provide more information to yield an even better clustering result. In fact, the best result from this set of experiments was slightly worse than the overall best window-level result for short meetings, 5.85% for d-vector + TDOA + GCC-PHAT. It is this superior combination of inputs with which further experiments are performed.

5.4 Full Meetings

So far, experiments had been conducted for short meeting lengths (equivalent to 50 speaker-homogeneous segments), but next it was attempted to handle full length meetings. This is a much harder task since the average number of segments per full meeting in the eval set is 702 and the longest meeting in the train set has 1682 segments. Hence, there are far more data points to cluster and the potential for errors to propagate through the output sequence is high.

Two input settings were experimented with, one each for segment-level and window-level clustering. The segment-level experiment used sub-meeting randomisation and Diaconis augmentation on d-vectors and so extended the best result from Table 5.2, which is repeated as the first item in Table 5.11. Likewise, the window-level experiment extended the best result from Table 5.8 (reprinted in Table 5.12), which used d-vector, TDOA and GCC-PHAT inputs with sub-meeting randomisation, Diaconis augmentation on the d-vector part and no TDOA/GCC-PHAT augmentation. The window length is 1.5s.

A curriculum learning approach was used for training. Each experiment began by training on sub-meetings of 50 segments (or 101 windows) as before. Then, this model was used to initialise training on 200 segment (or 404 window) sub-meetings, and so on until full meeting length was reached. For all but the first stage of training, the actual sub-meeting length was randomly sampled to be 50-100% of the desired meeting length in order to allow a greater number of training examples. Each stage also contained two sub-stages: pre-training (PT) and fine-tuning (FT). For PT, Diaconis was used, then for FT it was not so that fine tuning occurs only on sub-sampled real meetings. The learning rate was 8 for PT in the first stage, then all subsequent stages used a learning rate of 2. As meetings got longer, the mini-batch size had to be reduced from 50 to avoid overloading the GPU memory. To compensate for this, gradient accumulation was used which means the gradient is only updated after a certain number of mini-batches.

The most relevant result from each table (in bold) was after fine tuning on full meetings. In

Meeting Length (# Segments)	PT	FT	SC
50	12.71	11.23	15.66
200	18.96	16.22	22.18
500	26.58	24.68	23.44
Full	24.99	**23.23**	24.71

Table 5.11: % DERs from SC and DNC for **segment-level** clustering. The input vectors are **d-vectors**. DNC uses sub-meeting randomisation with Diaconis in pre-training (PT) and just sub-meeting randomisation in fine tuning (FT).

Meeting Length (# Windows)	PT	FT	SC
101	5.85	5.63	19.73
404	9.88	10.43	17.37
1010	15.05	13.14	18.54
Full	18.42	**17.25**	17.86

Table 5.12: % DERs from SC and DNC for 1.5s **window-level** clustering. The input vectors are concatenated **d-vector + TDOA + GCC-PHAT**. DNC uses sub-meeting randomisation with Diaconis in pre-training (PT) and just sub-meeting randomisation in fine tuning (FT).

both cases, this DER was only just better than spectral clustering and significantly higher than the DER for the shortest meeting length. First, analysing Table 5.11, it is clear that at each stage, fine tuning manages to lower the DER by about an absolute 1-2%. However, each time the meeting length is increased there is a large increase in DER (apart from between 500 and full, since the eval set meetings are not much longer than 500 segments). A key reason for this is likely to be the choice of augmentation. As previously discussed, when using sub-meeting randomisation there is a finite number of possible sub-meetings, and as the sub-meeting length increases, the number of possible sub-meetings decreases. Hence, there are fewer training examples available and so DNC may perform less well. In the original DNC paper [2], the same experiment was conducted but with meeting-level input vectors randomisation. Unlike sub-meeting randomisation, the number of possible augmented meetings was not reduced much by increasing the sub-meeting length. The full meeting FT result in this case was only 16.92% which supports the theory that sub-meeting randomisation may cause higher DERs for long meeting lengths. It may also be the case that training was less well tuned in this experiment than in the original one.

The reason segment-level sub-meeting randomisation was investigated for long meeting lengths and continued to be used at the window level is that input vectors randomisation may be inappropriate when TDOA/GCC-PHAT values are involved. Global-level input vectors randomisation certainly cannot be used since it is likely to be misleading in training to employ directional information when inputs are taken from speakers in different meetings. Since speakers are allowed to move in meetings, even meeting-level input vectors randomisation is likely to cause problems for long meeting lengths. The best chance of minimising these effects is to use sub-meeting randomisation, even if it is known that this restricts the number of augmented meetings. In Table 5.12, there is again a large increases in DER at each stage of curriculum learning. The relative DER increases are greater than for Table 5.11. This could occur if TDOA features become less useful for longer meeting lengths. When restricted to short meetings, speakers are unlikely to be able to move much, but for long meetings, they may move which will change their TDOA/GCC-PHAT values and DNC will struggle to cope. Despite this, it is impressive that DNC can still outperform SC, and clearly TDOA/GCC-PHAT values provide some use since the final DER, **17.25%**, was much lower than the segment-level equivalent without TDOA (**23.23%**), bearing in mind that window-level clustering is a harder task. It is important to note that if the four meetings from the TNO meeting room are removed from the eval set (as has been done in some other work [15]), the full meeting window-level DER is reduced to 8.31%. This is likely a result of TDOA inconsistency in TNO meetings which may be due to speakers moving more than in other meetings. It was attempted to manually analyse the TDOA values in TNO meetings compared to other meetings but no useful insights were found.

Curriculum learning is a long process with lots of scope for sub-optimal training and so several attempts were made to improve results by tuning hyperparameters. This had moderate success and further attempts to fine tune the long training process may lead to even better DNC results.

5.5 Results Summary

With window-level clustering having been the main focus of this project, Figure 5.6 presents the key window-level result from each set of experiments, together with the relevant baseline. Each result is for 1.5s windows, sub-meeting randomisation and Diaconis, with only the input types and sub-meeting lengths varying. The first four results are for sub-meetings of 101 windows, while the final result is for full meetings. This acts as a good illustration that DNC outperforms SC and is able to handle TDOA information well for shorter meetings. The results for full meetings show smaller improvements.

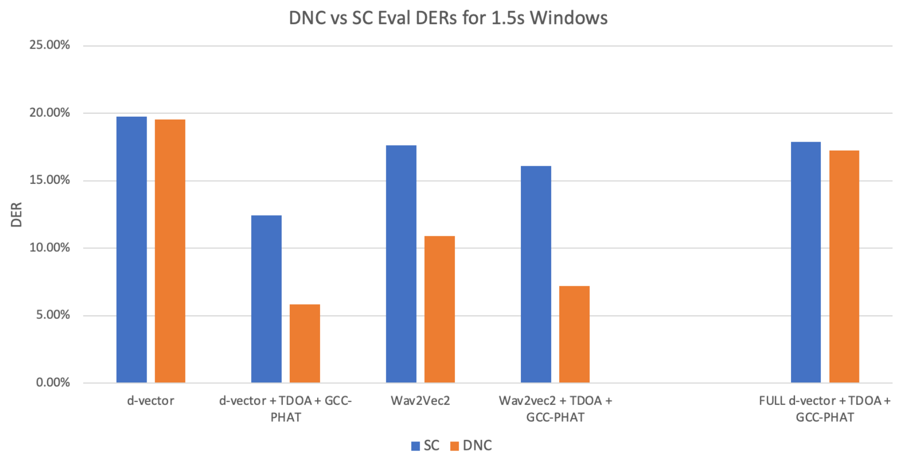


Figure 5.6: Summary of key results from each set of 1.5s window-level experiments. The first four results are for sub-meetings of 101 windows and the final one is for full meetings. Sub-meeting randomisation with Diaconis is used in each case.

Chapter 6

Conclusions

This project has extended the Discriminative Neural Clustering framework to remove the assumption of Change Point Detection and allow additional inputs. The following is a summary of the key conclusions:

- ‘On-the-fly’ augmentation allows data to be augmented dynamically during training in order to reduce storage cost and provide a more flexible codebase.
- Diaconis augmentation is the most important form of data augmentation and typically reduces the DER significantly (from 24.53% to **12.71%** using d-vectors with sub-meeting randomisation at the segment level for short meetings).
- Dividing utterances into fixed length windows requires Voice Activity Detection to have been performed but does not assume Change Point Detection. It is a harder task than segment-level clustering and the spectral clustering baseline has a higher DER (19.73% for 1.5s windows and d-vector input compared to 15.66% at the segment level). The optimum window length for DNC is 1.5s which achieves an eval DER of **19.54%** on short meetings using d-vectors with sub-meeting randomisation and Diaconis augmentation.
- Using TDOA vectors as an input to DNC is very powerful and gives much lower DERs for short meetings (**6.79%** on its own at the 1.5s window-level) than the equivalent SC baseline (16.15%) and d-vectors without TDOA (19.54%).
- The proposed forms of TDOA/GCC-PHAT augmentation are unnecessary as they do not create novel enough training examples. This is not a problem as TDOA performs well without augmentation.
- An input of just Wav2Vec2 performs better than an input of just d-vectors for short meetings (**10.91%** compared to **19.54%** for 1.5s windows and 101 window sub-meetings), but this improvement becomes negligible or non-existent once TDOA vectors are included, which are the dominant feature.

- The best result for window-level clustering is using 1.5s windows and an input of d-vectors, TDOA and GCC-PHAT. The DER on the eval set for short meetings is **5.85%**.
- The performance of DNC deteriorates for longer meetings as training becomes more difficult. However, results are still better than SC (**17.25%** compared to 17.86% at the 1.5s window-level for full meetings and an input of d-vectors, TDOA and GCC-PHAT). It may be that sub-meeting randomisation and TDOA information become less useful as the meeting length increases, though it is difficult to determine how much of a problem this is and how much results could be improved by refining curriculum learning.

6.1 Possible Extensions

There are several ways DNC could be extended by future work. While several of the original limitations have been removed, there is still progress to be made. The following are suggested future developments:

- Curriculum learning could be integrated into ‘on-the-fly’ augmentation so that it could be performed in one round of training, rather than several models being initialised on each other. This may be more convenient if properly fine-tuned but could prove a hindrance if hyper-parameters need to be manually changed during training. This is why it was not implemented in this project.
- Since it may be the case that TDOA values are inconsistent in the eval TNO meetings, ways of reducing the reliance of DNC on TDOA could be investigated. Alternatively, it could be attempted to perform some form of ‘speaker movement detection’ so if a speaker changes seat, the resulting change in TDOA does not cause DNC problems.
- If VAD and embedding generation could be included in the DNC pipeline, it would lead to an end-to-end diarisation system. To begin with, the modules could be separate but it would be possible to jointly optimise them.
- Currently, DNC cannot handle overlapping speakers. An ideal model would label all speakers speaking at one time, but a useful first step would be to merely label when overlap occurs. DNC could be adapted to allow this by treating ‘overlap’ as a possible speaker label. At first, overlap detection could be scored using precision and recall and treated separately from the speaker error rate. Eventually, a further extension would be that if overlap is detected, the two most probable speaker labels are outputted rather than just the first one.
- Video input is sometimes available when performing speaker diarisation. This would be a non-trivial addition to DNC but could lead to better results and could also help deal with overlapped speech.

Appendix A

Risk Assessment Retrospective

At the start of the project, I submitted the following risk assessment:

> Computer use hazards include risk to eyesight/eyestrain, finger joint problems and back pain. These can be avoided by using a suitably sized display (I usually use a monitor rather than laptop screen), correctly positioned mouse and keyboard and properly adjusted seat. I will also take regular breaks and walks.

Thanks to these precautions, I did not suffer any problems related to extended computer use. There are no changes I would make to the risk assessment if repeating the project.

References

[1] T. J. Park, N. Kanda, D. Dimitriadis, K. J. Han, S. Watanabe, and S. Narayanan, *A review of speaker diarization: Recent advances with deep learning*, 2021. arXiv: 2101.09624 [eess.AS].

[2] Q. Li, F. L. Kreyssig, C. Zhang, and P. C. Woodland, *Discriminative neural clustering for speaker diarisation*, 2020. arXiv: 1910.09703 [eess.AS].

[3] F. Kreyssig, C. Zhang, and P. Woodland, *Improved tdnns using deep kernels and frequency dependent grid-rnns*, 2018. [Online]. Available: https://arxiv.org/abs/1802.06412.

[4] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, *Attention is all you need*, 2017. arXiv: 1706.03762 [cs.CL].

[5] Q. Wang, C. Downey, L. Wan, P. A. Mansfield, and I. L. Moreno, *Speaker diarization with lstm*, 2018. arXiv: 1710.10468 [eess.AS].

[6] H. Gish, M.-H. Siu, and R. Rohlicek, "Segregation of speakers for speech recognition and speaker identification," in *[Proceedings] ICASSP 91: 1991 International Conference on Acoustics, Speech, and Signal Processing*, 1991, 873–876 vol.2. DOI: 10.1109/ICASSP.1991.150477.

[7] S. Tranter and D. Reynolds, "An overview of automatic speaker diarization systems," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 5, pp. 1557–1565, 2006. DOI: 10.1109/TASL.2006.878256.

[8] S. Shaobing and C. Gopalakrishnan, "Speaker, environment and channel change detection and clustering via the bayesian information criterion," Aug. 2000.

[9] J. Carletta, S. Ashby, S. Bourban, *et al.*, "The ami meeting corpus: A pre-announcement," Jul. 2005, ISBN: 978-3-540-32549-9. DOI: 10.1007/11677482_3.

[10] P. Kenny, D. Reynolds, and F. Castaldo, "Diarization of telephone conversations using factor analysis," *Selected Topics in Signal Processing, IEEE Journal of*, vol. 4, pp. 1059–1070, Jan. 2011. DOI: 10.1109/JSTSP.2010.2081790.

[11] M. Diez, L. Burget, and P. Matejka, "Speaker diarization based on bayesian hmm with eigenvoice priors," Jun. 2018, pp. 147–154. DOI: 10.21437/Odyssey.2018-21.

[12] F. Landini, J. Profant, M. Diez, and L. Burget, *Bayesian hmm clustering of x-vector sequences (vbx) in speaker diarization: Theory, implementation and analysis on standard tasks*, 2020. [Online]. Available: https://arxiv.org/abs/2012.14952.

[13] E. Variani, X. Lei, E. McDermott, I. L. Moreno, and J. Gonzalez-Dominguez, "Deep neural networks for small footprint text-dependent speaker verification," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 4052–4056. DOI: 10.1109/ICASSP.2014.6854363.

[14] A. Zhang, Q. Wang, Z. Zhu, J. Paisley, and C. Wang, *Fully supervised speaker diarization*, 2018. [Online]. Available: `https://arxiv.org/abs/1810.04719`.

[15] X. Wan, K. Liu, and H. Zhou, "Online Speaker Diarization Equipped with Discriminative Modeling and Guided Inference," in *Proc. Interspeech 2021*, 2021, pp. 3091–3095. DOI: `10.21437/Interspeech.2021-261`.

[16] Y. Fujita, N. Kanda, S. Horiguchi, K. Nagamatsu, and S. Watanabe, *End-to-end neural speaker diarization with permutation-free objectives*, 2019. [Online]. Available: `https://arxiv.org/abs/1909.05952`.

[17] S. Horiguchi, Y. Fujita, S. Watanabe, Y. Xue, and K. Nagamatsu, *End-to-end speaker diarization for an unknown number of speakers with encoder-decoder based attractors*, 2020. [Online]. Available: `https://arxiv.org/abs/2005.09921`.

[18] E. Han, C. Lee, and A. Stolcke, "BW-EDA-EEND: Streaming END-TO-END neural speaker diarization for a variable number of speakers," in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, Jun. 2021. [Online]. Available: `https://doi.org/10.1109%5C%2Ficassp39728.2021.9414371`.

[19] T. Yoshioka, I. Abramovski, C. Aksoylar, *et al.*, *Advances in online audio-visual meeting transcription*, 2019. [Online]. Available: `https://arxiv.org/abs/1912.04979`.

[20] L. E. Shafey, H. Soltau, and I. Shafran, "Joint speech recognition and speaker diarization via sequence transduction," 2019. [Online]. Available: `https://arxiv.org/abs/1907.05337`.

[21] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, `http://www.deeplearningbook.org`.

[22] J. L. Elman, "Finding structure in time," *COGNITIVE SCIENCE*, vol. 14, no. 2, pp. 179–211, 1990.

[23] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, Dec. 1997. DOI: `10.1162/neco.1997.9.8.1735`.

[24] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, *On the properties of neural machine translation: Encoder-decoder approaches*, 2014. [Online]. Available: `https://arxiv.org/abs/1409.1259`.

[25] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, 2015. [Online]. Available: `https://arxiv.org/abs/1512.03385`.

[26] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. Lang, "Phoneme recognition using time-delay neural networks," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, no. 3, pp. 328–339, 1989. DOI: `10.1109/29.21701`.

[27] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2014. [Online]. Available: `https://arxiv.org/abs/1412.6980`.

[28] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014. [Online]. Available: `http://jmlr.org/papers/v15/srivastava14a.html`.

[29] J. L. Ba, J. R. Kiros, and G. E. Hinton, *Layer normalization*, 2016. [Online]. Available: `https://arxiv.org/abs/1607.06450`.

[30] R. J. Williams and D. Zipser, “A learning algorithm for continually running fully recurrent neural networks,” *Neural Computation*, vol. 1, no. 2, pp. 270–280, 1989. DOI: 10.1162/neco.1989.1.2.270.

[31] A. Lamb, A. Goyal, Y. Zhang, S. Zhang, A. Courville, and Y. Bengio, *Professor forcing: A new algorithm for training recurrent networks*, 2016. [Online]. Available: https://arxiv.org/abs/1610.09038.

[32] J. Elman, “Learning and development in neural networks: The importance of starting small,” *Cognition*, vol. 48, pp. 71–99, Aug. 1993. DOI: 10.1016/0010-0277(93)90058-4.

[33] D. Richard O., H. Peter E., and S. David G., *Pattern Classification.* Wiley-Interscience, 2001, vol. 2nd ed, ISBN: 9780471056690. [Online]. Available: https://ezp.lib.cam.ac.uk/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=531488&site=ehost-live&scope=site.

[34] Z. Huang, S. Watanabe, Y. Fujita, *et al.*, *Speaker diarization with region proposal network*, 2020. [Online]. Available: https://arxiv.org/abs/2002.06220.

[35] 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)*, 2017, pp. 4930–4934. DOI: 10.1109/ICASSP.2017.7953094.

[36] S. H. Shum, N. Dehak, R. Dehak, and J. R. Glass, “Unsupervised methods for speaker diarization: An integrated and iterative approach,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 10, pp. 2015–2028, 2013. DOI: 10.1109/TASL.2013.2264673.

[37] L. van der Maaten and G. Hinton, “Visualizing data using t-sne,” *Journal of Machine Learning Research*, vol. 9, no. 86, pp. 2579–2605, 2008. [Online]. Available: http://jmlr.org/papers/v9/vandermaaten08a.html.

[38] F. L. Kreyssig and P. C. Woodland, *Cosine-distance virtual adversarial training for semi-supervised speaker-discriminative acoustic embeddings*, 2020. [Online]. Available: https://arxiv.org/abs/2008.03756.

[39] A. Baevski, H. Zhou, A. Mohamed, and M. Auli, *Wav2vec 2.0: A framework for self-supervised learning of speech representations*, 2020. [Online]. Available: https://arxiv.org/abs/2006.11477.

[40] A. Senior and I. Lopez-Moreno, “Improving dnn speaker independence with i-vector inputs,” in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 225–229. DOI: 10.1109/ICASSP.2014.6853591.

[41] R. Milner and T. Hain, *Dnn-based speaker clustering for speaker diarisation*, © 2016 ISCA. This is an author produced version of a paper subsequently published in Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH. Uploaded in accordance with the publisher’s self-archiving policy., Sep. 2016. [Online]. Available: https://eprints.whiterose.ac.uk/109281/.

[42] G. Sun, C. Zhang, and P. C. Woodland, “Combination of deep speaker embeddings for diarisation,” *Neural Networks*, vol. 141, pp. 372–384, Sep. 2021, ISSN: 0893-6080. [Online]. Available: http://dx.doi.org/10.1016/j.neunet.2021.04.020.

[43] 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)*, 2018, pp. 5329–5333. DOI: 10.1109/ICASSP.2018.8461375.

[44] Y. Li, F. Gao, Z. Ou, and J. Sun, *Angular softmax loss for end-to-end speaker verification*, 2018. [Online]. Available: https://arxiv.org/abs/1806.03464.

[45] G. Sun, C. Zhang, and P. Woodland, *Speaker diarisation using 2d self-attentive combination of embeddings*, 2019. [Online]. Available: https://arxiv.org/abs/1902.03190.

[46] A. Nagrani, J. S. Chung, and A. Zisserman, “VoxCeleb: A large-scale speaker identification dataset,” in *Interspeech 2017*, ISCA, Aug. 2017. [Online]. Available: https://doi.org/10.21437%5C%2Finterspeech.2017-950.

[47] P. Khosla, P. Teterwak, C. Wang, *et al.*, *Supervised contrastive learning*, 2020. [Online]. Available: https://arxiv.org/abs/2004.11362.

[48] X. Anguera, C. Wooters, and J. Hernando, “Acoustic beamforming for speaker diarization of meetings,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 7, pp. 2011–2022, 2007. DOI: 10.1109/TASL.2007.902460.

[49] D. Povey, A. Ghoshal, G. Boulianne, *et al.*, “The kaldi speech recognition toolkit,” in *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*, IEEE Catalog No.: CFP11SRW-USB, Hilton Waikoloa Village, Big Island, Hawaii, US: IEEE Signal Processing Society, Dec. 2011.

[50] C. R. Harris, K. J. Millman, S. J. van der Walt, *et al.*, “Array programming with NumPy,” *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2.

[51] NIST, “The 2009 (rt-09) rich transcription meeting recognition evaluation plan,” 2009. [Online]. Available: https://web.archive.org/web/20100606041157if_/http://www.itl.nist.gov/iad/mig/tests/rt/2009/docs/rt09-meeting-eval-plan-v2.pdf.

[52] P. Diaconis and M. Shahshahani, “The subgroup algorithm for generating uniform random variables,” *Probability in the Engineering and Informational Sciences*, vol. 1, no. 1, pp. 15–32, 1987. DOI: 10.1017/S0269964800000255.

[53] S. Watanabe, T. Hori, S. Karita, *et al.*, *Espnet: End-to-end speech processing toolkit*, 2018. [Online]. Available: https://arxiv.org/abs/1804.00015.

[54] A. Paszke, S. Gross, F. Massa, *et al.*, *Pytorch: An imperative style, high-performance deep learning library*, 2019. [Online]. Available: https://arxiv.org/abs/1912.01703.

[55] O. Galibert, “Methodologies for the evaluation of speaker diarization and automatic speech recognition in the presence of overlapping speech,” Aug. 2013.

[56] H. Dubey, A. Sangwan, and J. H. L. Hansen, *Robust speaker clustering using mixtures of von mises-fisher distributions for naturalistic audio streams*, 2018. [Online]. Available: https://arxiv.org/abs/1808.06045.