University of Sheffield

Improving the Robustness of Speaker Recognition in Noise and Multi-Speaker Conditions Using Deep Neural Networks

Yanpei Shi

Supervisor: Prof. Thomas Hain

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

in the

Department of Computer Science

October 19, 2021

Declaration

Acknowledgements

When I started my PhD, I could not imagine what it would be like on the day when I was able to graduate: at the time, that day appeared to be almost beyond my reach. In the three and a half years, I have been studying, however, I have grown a lot. While at first I often felt lonely and afraid during that period, studying for a PhD in a foreign country, alone, I have nevertheless survived and thrived, overcoming all of the difficulties I faced, and emerging in the dawn of victory.

I would like to thank my supervisor, Professor Thomas Hain: because of him, I have had a wonderful time during the last three years of my PhD studies. When I was in the first year of my PhD, I struggled to find a direction, producing an excess of random ideas. He, fortunately, has had a great deal of patience with me, and he took the time to teach me how to think about problems and find a direction for my focus. Because of his patience, I am now more enthusiastic than ever about doing research. I would also like to thank him for his work to secure funding support for me from the HIRP project; this not only offered me valuable funding, but as the only scholarship I have received during my 20 years of schooling, it has also enabled me to feel affirmed and valued, and allowed me to find my way during a most difficult time.

Looking back on my PhD period, and indeed the past 26 years of my life, I would like to acknowledge that I have been very lucky. I would also like to thank my family, especially my parents, as no matter what difficulties I have encountered, they have always provided me with strong backing and support. They are the most important people in my world, and it is thanks to them that I am the first person to gain a PhD degree in my family. I am thus very happy that my PhD will make my family proud of me. My grandma and grandpa, who accompanied me as I grew up, have also been my spiritual pillars. Although I have now been living away for several years, I remain concerned about their well being, and I am very excited that as I am about to end my PhD career I can return to my hometown and spend some quality time with them.

My girlfriend, Yuchen Zhu, always supports me, even when I am in bad mood. I am a very anxious person, yet with her help and in her company, I have gradually become better, becoming more gentle and stable. I have begun to learn how to enjoy life and how to relieve my stress in a positive manner. Her greatest support has been her continuing presence: in 2019, as I experienced the hardest section of my studies, she chose to give up the opportunity to return to China and stayed with me in the UK, yet when I wanted to return to China because of the situation, she supported me, ensuring that I could return to China smoothly and safely.

I would also like to acknowledge my best friends, Yao Xiao and Shi Tang: we grew up together, and even though we have lived far from each other for most of the last few years, I have felt like they have retained by my side. No matter how late, and no matter what the time difference, when I called them with me difficulties, they always encouraged me and filled me with hope. Even now, when we have good news, we share it with each other, and when we have difficulties, we encourage each other. I believe that when we are all old, we will still remember growing up together, and it will be our best memory and our life's wealth.

My appreciation also goes to Dr Chang Liu, my best friend; we met each other when I was studying for a master's degree, and my decision to study for a PhD was greatly influenced by her. This decision changed my life, and I am very grateful for it. I also admire her greatly, and would like to remind her that a friend is a chosen relative. My friends are my relatives in this way, and they form a vital part of my world. Dr Meng Fan, my room mate, is also a great friend of mine, and he too has had a huge influence on me. I admire his integrity and his political persistence across the years; I would like to thank him from the bottom of my heart, as I have far less courage and wisdom, and I appreciate his example.

I would also like to thank Dr Qiang Huang, who gave me a lot of help with various projects and who, since 2019, has helped me with my papers, and worked alongside me. Without his help, it would have been much more difficult for me to publish papers, and the help he offered without any expectation of return has made me feel particularly warm and welcome, especially with regard to my residence in a foreign country. I have learned a lot from him, and we have become firm friends.

I am looking forward to starting the next stage of my life, but the unique experience of PhD study along with the people I have met, the difficulties I experienced, and the knowledge I have learned, have all contributed great to the richness of my life, and I will remember them forever.

Abstract

In speaker recognition, deep neural networks deliver state-of-the-art performance due to their large capacities and powerful feature extraction abilities. However, this performance can be highly affected by interference from background noise and other speakers. This thesis focuses on new neural network architectures that are designed to overcome such interference and thereby improve the robustness of the speaker recognition system.

In order to improve the noise robustness of the speaker recognition model, two novel network architectures are proposed. The first is the hierarchical attention network, which is able to capture both local and global features in order to improve the robustness of the network. The experimental results show it can deliver results that are comparable to the published state-of-the-art methods, reaching 4.28% equal error rate using the Voxceleb1 training and test sets. The second approach is the speech enhancement and speaker recognition joint system that consists of two networks; the first integrates speech enhancement and speaker recognition into one framework to better filter out noise, while the other makes further use of speaker embeddings input to a speech enhancement network. This provides prior knowledge for the speech enhancement network which improves its performance. The results show that a joint system with a speaker dependent speech enhancement model can deliver results that are comparable to the published state-of-the-art methods, reaching 4.15% equal error rate using the Voxceleb1 training and test sets.

In order to overcome interfering speaker, two novel approaches are proposed. The first is referred to as the embedding de-mixing approach that separates the speaker and content properties from a two-speaker signal in an embedding space, rather than in a signal space. The results show that the de-mixed embeddings are close to the clean embeddings in terms of quality, and the back-end speaker recognition model can make use of the de-mixed embeddings to reach 96.9% speaker identification accuracy, compared to those achieved using clean embeddings (98.5%) on TIMIT dataset. The second approach is the first end-to-end weakly supervised speaker identification approach based on a novel hierarchical transformer network architecture. The results show that the proposed model can capture speaker properties from two speakers in one input utterance. The hierarchical transformer network can reach more than 3% relative improvement compared to the baselines in all of the test conditions.

Contents

List of Figures

List of Tables

Acronyms / Abbreviations

GMM	Gaussian Mixture Model
UBM	Universal Background Model
SVM	Support Vector Machine
JFA	Joint Factor Analysis
I-vector	The Identity Vector
DNN	Feed Forward Neural Network
MLP	Multi Layer Perceptron
Sigmoid	Sigmoid Activation Function
ReLU	Rectified Linear Unit
Tanh	Hyperbolic Tangent
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short Term Memory
GRU	Gated Recurrent Unit
GPU	Graphics Processing Unit
STFT	Short Term Fourier Transform
DCT	Discrete Cosine Transform
MFCC	Mel-Frequency Ceptral Coefficients
A-softmax	Angular Softmax
AM-softmax	Additive Margin Softmax
AAM-softmax	Additive Angular Margin Softmax
EER	Equal Error Rate
DET	Detection Error Tradeoff
DCF	Detection Cost Function
Voxceleb1	The Voxceleb1 Dataset
SWBC	The Switchboard Cellular Part1 Dataset
MUSAN	The MUSAN Dataset
SRE08	The NIST SRE 2008 Part1 Dataset
CHE	The CallHome American English Speech Dataset
TIMIT	The TIMIT Dataset
MC-WSJ	The Multi-Channel Wall Street Journal Audio Visual Corpus

H-vector	The Hierarchical Attention Network
SE-Net	The Speech Enhancement Model
SR-Net	The Speaker Recognition Model
MS	The Multi-Stage Attention Mechanism
PESQ	The Perceptual Evaluation of Speech Quality Measurement
STOI	The Short-Time Objective Intelligibility Measurement
PER	Phone Error Rate
T-vector	Hierarchical Transformer Network

List of Symbols

$\boldsymbol{\alpha}$	The attention weight vector as defined in Section 2.3.2
α_t	The attention weight value at tth time step
$\boldsymbol{\alpha}_m$	The normalized attention score vector for $\boldsymbol{H}_m$ in H-vector of Chapter 3
$\boldsymbol{\alpha}_S$	The attention weight vector of the segment-level encoder in H-vector of Chapter 3
$\boldsymbol{\alpha}_{C,k}$	The channel attention weight vector of the multi-stage attention model
$\boldsymbol{\alpha}_{F,k}$	The frequency attention weight vector of the multi-stage attention model
$\boldsymbol{\alpha}_{T,k}$	The time attention weight vector of the multi-stage attention model
$\boldsymbol{A}_m$	The weighted output of the mth frame-level encoder in H-vector
$\mathrm{avg}(.)$	Average pooling operation
$\mathrm{avg}^{T\times F\times C}$	The average pooling operation with the kernel size of $T \times F \times C$
$[\boldsymbol{a}; \boldsymbol{b}]$	The concatenation of the vectors $\boldsymbol{a}$ and $\boldsymbol{b}$
$\boldsymbol{b}$	A bias vector
B	The batch size
$\mathrm{Broad}(.)^{T\times F}$	The broadcasting function with target dimension of $T \times F$
C	The dimensionality of the channel dimension
$\mathrm{Conv}^{2\times 7\times 2}$	The convolutional operation with the kernel size of $2 \times 7 \times 2$
d	The dimensionality of the embeddings
$\boldsymbol{e}_{S_m}$	The segment vector of the mth segment in H-vector
$\boldsymbol{E}_S$	The sequence of the segment vectors in H-vector
E	The dimension of the feature vector of $\boldsymbol{H}$ as defined in Section 2.2.4
$\boldsymbol{e}_u$	The utterance vector in H-vector
$\boldsymbol{e}_{X1}$	The embedding obtained in step1 in Section 4.4
$\boldsymbol{e}_{X2}$	The embedding obtained in step2 in Section 4.4
$\boldsymbol{e}_{mix}$	The mixed embedding

$\boldsymbol{e}_{s1}$	The clean embedding of the speaker1
$\boldsymbol{e}_{s1}^{'}$	The predicted embedding of the speaker1
$\boldsymbol{e}_{s2}$	The clean embedding of the speaker2
$\boldsymbol{e}_{s2}^{'}$	The predicted embedding of the speaker2
$\boldsymbol{e}_{c1}$	The clean embedding of the spoken content by the speaker1
$\boldsymbol{e}_{c1}^{'}$	The predicted embedding of the spoken content by the speaker1
$\boldsymbol{e}_{c2}$	The clean embedding of the spoken content by the speaker2
$\boldsymbol{e}_{c2}^{'}$	The predicted embedding of the spoken content by the speaker2
$f(.)$	A mapping function
F	The dimensionality of the frequency dimension
$f_{demix}(.)$	The de-mixing function
G	The number of speakers in one input utterance $\boldsymbol{X}_{mix}$ in Chapter 6
$\boldsymbol{H}$	The hidden states, hidden layer output or refined feature map
$\boldsymbol{h}_t$	The hidden state at the tth time step
$\boldsymbol{H}_m$	The output of the mth frame-level encoder
$\boldsymbol{H}_k$	The input feature map of the kth CONV-MS block or RES-MS block
$\boldsymbol{H}_k^{'}$	The output feature map of the channel attention
$\boldsymbol{H}_k^{''}$	The output feature map of the frequency attention
$\boldsymbol{H}_k^{'''}$	The output feature map of the time attention
$\boldsymbol{h}_i$	The ith attention head
h	The number of the attention heads
J	The number of inputs of a perceptron
$\boldsymbol{K}$	The Key matrix
L	The length of the feature sequence $\boldsymbol{H}$ in attentive X-vector
$\mathcal{L}$	Loss function
L_{win}	The window size in H-vector
L_{step}	The step size in H-vector
M	Total number of segments in H-vector model
$\max(.)$	Max pooling operation
$\max^{T\times F\times C}$	The max pooling operation with the kernel size of $T \times F \times C$

MultiHead($\boldsymbol{Q}$, $\boldsymbol{K}$, $\boldsymbol{V}$)	The multi-head attention mechanism of $\boldsymbol{Q}$, $\boldsymbol{K}$ and $\boldsymbol{V}$
N	Total number of speakers or total number of classes
PE(.)	The positional encoding operation
$\boldsymbol{Q}$	The Query matrix
$\boldsymbol{q}$	The one's vector or matrix
ReLU(.)	ReLU activation function
Sigmoid(.)	Sigmoid activation function
$\boldsymbol{S}_m$	The mth segment
T	The dimensionality of the time dimension
Tanh(.)	Tanh activation function
$\boldsymbol{V}$	The Value matrix
$\boldsymbol{W}$	A weight matrix
$\boldsymbol{X}$	The input sequence
$\boldsymbol{x}_t$	A feature vector at time t in $\boldsymbol{X}$
$\boldsymbol{X}_N \in \mathcal{R}^{T\times F\times C}$	Noisy input spectrogram of the dimensionality of $T \times F \times C$
$\boldsymbol{X}^{'}_{C1}$	The enhanced spectrogram for step1 in Section 4.4
$\boldsymbol{X}^{'}_{C2}$	The enhanced spectrogram for step2 in Section 4.4
$\boldsymbol{X}_{mix}$	The mixed signal
y	Ground truth label
$\boldsymbol{y}$	One hot vector of the ground truth label
$\boldsymbol{y}_x$	The label vector of $\boldsymbol{X}_{mix}$
$\boldsymbol{y}^{'}_x$	The predicted score vector for $\boldsymbol{X}_{mix}$
$\boldsymbol{\theta}$	The model parameters
η	The learning rate
τ	The threshold
$\nabla(w)$	The gradient of parameter w
$\sigma(.)$	Activation function

Chapter 1

Introduction

Speaker recognition aims to recognise the identities of speakers from the corresponding characteristics of their voices (Campbell 1997, Markel et al. 1977). The fact that a persons' voice contains unique traits makes automatic speaker recognition by computers possible (Kinnunen & Li 2010). Speaker recognition is useful and has a wide range of applications (Bai & Zhang 2021). For example, in the security domain, speaker recognition systems provide a step towards identity authorisation, thereby making an online payment system or a personal electronic device more secure (Peacocke & Graf 1995). Moreover, the speaker related information that is learned from a speaker recognition system (referred to as the speaker embeddings) can also benefit many downstream tasks in speech technology. For example, the information in the speaker embeddings, captured by a speaker recognition system, can be fed into a speaker dependent automatic speech recognition (ASR) system in order to better recognise specific target speakers (Huang & Lee 1993, Fontaine & Bourlard 1997). Furthermore, in speaker dependent text to speech synthesis (TTS) systems, the learned speaker embeddings can help the model to better synthesis the voices of the target speakers (Sreenu et al. 2004, Hojo et al. 2016).

The most common approach in speaker recognition is to represent a variable length signal into a fixed length speaker embedding that can then be used to identify the speakers. Conventionally, the GMM based i-vector approach achieved the state-of-the-art performance in speaker recognition (Dehak et al. 2010). In recent years, due to the strong ability of extracting and representing features, deep learning techniques have been widely adopted (Poddar et al. 2017). Deep neural networks can be built both larger and deeper in order to be trained with large amounts of data, which improves their feature representation and speaker recognition abilities (Zhang 2017, Lei et al. 2014), and this has led to deep neural networks to become the state-of-the-art method now (Variani et al. 2014, Snyder et al. 2018).

However, deep neural network and other machine learning approaches still suffer from the effects of background noise. According to Zhao et al. (2014) and Ming et al. (2007), speaker recognition systems based on deep neural networks perform well in clean conditions, but performance deteriorates under noisy conditions when the background noise potentially covers up (or masks) the information of the speaker's voice. This can affect some key features that make it difficult for deep neural networks to recognise speaker identities.

Another challenge comes from interfering speakers which, according to Le Prell & Clavier (2017), can compete for the same frequency band as the target speaker (i.e. the speaker that the system needs to recognise), making it difficult for the system to distinguish between the elements of the target speaker and those from the interfering speakers.

In order to overcome these challenges, this thesis focuses on developing new deep neural network architectures to improve the robustness of a speaker recognition system against background noise and interfering speakers.

1.1 Proposed Methods

1.1.1 Hierarchical Attention Network

One way to improve the robustness of the speaker recognition models in noisy conditions is to use an attention mechanism that is built into the network architecture. The attention mechanism can allocate different weight values to different parts of the signal, thereby allowing the model to focus more on what is important, and less on what is corrupted (Anwar & Barnes 2019, Zhang et al. 2020). The most widely used attention mechanism in speaker recognition is global self-attention, which uses attention weights for the whole input sequence (Wang, Okabe, Lee, Yamamoto & Koshinaka 2018, Okabe et al. 2018). The limitation here, however, is with longer sequences, where the global self-attention is unable to focus on all the important features in a input signal and, in particular, it is unable to pay sufficient attention to the local features. This is explored in more detail in Section 2.3.2.

In order to overcome the problems mentioned above, Chapter 3 proposes a hierarchical attention network to generate robust, utterance-level embeddings (H-vectors) for speaker recognition. The proposed method splits the global self-attention into two levels: frame-level and segment-level. Such a hierarchical structure aims to learn speaker related information both locally and globally. The experimental results show that the proposed method can obtain better noise robustness than the global self-attention mechanism, and can deliver results comparable to the published state-of-the-art methods.

The contributions of Chapter 3 are:

- Introduces a hierarchical attention network that can capture both local and global features to improve the noise robustness of the speaker recognition system.
- The proposed hierarchical attention network contains novel model architectures that can deliver results comparable to the published state-of-the-art model.

1.1.2 Speech Enhancement For Speaker Recognition

Speech enhancement is widely used to overcome noise interference. Unlike the attention mechanism, speech enhancement is usually used independently, ahead of the speaker recognition model, working as a pre-processing stage that filters out the noise information. The denoised signal can then be used as input to the speaker recognition model. The disadvantage of such a set-up is that deploying speech enhancement and speaker recognition separately may lead to a mismatch problem, discussed in more depth in Section 2.3.3. The mismatch problem, caused by the fact that the two models are trained separately, means the speech enhancement model may corrupt some of the important speaker information when filtering out the noise interference. The speech enhancement model does not have the constraint to retain useful features for the back-end speaker recognition model (Shon et al. 2019, Wang & Wang 2016, Sadjadi & Hansen 2010).

In order to overcome the mismatch problem and improve the performance of the speaker recognition models in noisy conditions, Chapter 4 proposes two architectures that jointly train the two models.

In the first joint training system, the speech enhancement and the speaker recognition models are trained together using one objective function. The goal of the training is to not only filter out the noise information, but also retain the useful features for the back-end speaker recognition model. For a greater level of robustness, a novel, multi-stage attention mechanism is proposed, which applies the attention mechanism in different dimensions of the input data to filter out noise interference in those different dimensions. The experimental results show the joint training system can ease the mismatch problem and can reach better performance than the separate training strategy.

In order to further improve performance, in the second joint training system, a speaker dependent speech enhancement approach is proposed, based on the joint training framework but making use of the pre-trained speaker embeddings. The speaker embeddings contain information for the specific speakers which helps the speech enhancement model retain more speaker related information. The experimental results show that the speaker dependent speech enhancement model can reach better performance than the speaker independent speech enhancement model, and can deliver results comparable to the state-of-the-art methods.

The contributions of Chapter 4 are:

- Introduces a novel joint training system for the speech enhancement and speaker recognition models to improve the performance of the speaker recognition model under noisy conditions.
- Introduces a novel, multi-stage attention mechanism that is applied across the time, frequency and channel dimensions in order to better filter out the noisy information in the input signal.
- Introduces a novel, speaker dependent speech enhancement model to filter out the noise information in the input signal, thereby improving the performance of the speaker recognition model under various noise interference conditions.
- Introduces a residual auto-encoder model architecture, combining the pre-trained speaker embeddings to achieve a speaker dependent speech enhancement approach.

1.1.3 Embedding De-mixing Networks

Interference from speech by other speakers is particularly challenging as they often compete for the same frequency band as the target speaker, making it difficult for speaker recognition system to pick out the elements of the target speaker. The most common approach to overcome this problem is the target speaker extraction method which isolates the voice of the target speaker from a multi-speaker signal (see Section 2.4.2). The ideal output contains the target voices only. However, when the goal is to improve the performance of the back-end speaker recognition model in a multi-speaker environment, it may not be necessary to separate the target speaker's voice and construct it in signal space. Instead, separating the target speaker's information in an embedding space may be more efficient. One advantage of separating the target speaker's information in an embedding space is that it does not require a decoding process, which reduces the complexity of the model and makes it easier to train, as do the fixed length, low dimensional embeddings. The back-end model does not need to learn the mapping from the signal to the speaker identities, but rather from the embeddings to the speaker identities.

Judging by a review of the literature, the approach of separating speaker properties in an embedding space, as laid out in Section 2.4.2, has never before been proposed. Chapter 5 introduces the first approach that can separate the overlapped information in a two-speaker signal into embeddings space, which is further referred to as the embedding de-mixing approach. The proposed approach not only filters out the information of one speaker using the corresponding embeddings, but also can extract the speaker information or content information based on the corresponding content or speaker embeddings. The contributions of Chapter 5 are:

- Introduces the first approach that can separate the speaker and content properties in the embedding space rather than in the signal space. The proposed approach covers three scenarios.

- Introduces a speaker embedding de-mixing approach that can filter out the influence of the interfering speaker in a two-speaker signal using pre-trained embeddings. The results from the de-mixed speaker embedding approach come close in the speaker identification task to what the system can achieve with a clean signal.

- Introduces a content embedding de-mixing approach that can filter out the spoken content information from the interfering speaker using the corresponding pre- trained embeddings. The de-mixed content embedding delivers comparable results in the content classification task to those obtained using clean signals.

- Introduces a speaker and content embedding de-mixing approach that is used to extract the speaker or content properties of the target speaker by the embeddings. Both of the de-mixed content embeddings and the the de-mixed speaker embeddings can deliver results that are comparable to those obtained using clean signals.

1.1.4 Weakly Supervised Speaker Identification

In a conversation scenario that multiple speakers speaking at the same time, it is useful if a system can recognise the identities of all the speakers present. In this case, there is no target speaker, so the model needs to recognise all of the speakers that occur in the input utterance. A natural way to achieve this is to manually annotate the speaker identities and positions of the multi-speaker signal in the training set, but this is expensive and time consuming. Using a pre-trained speaker diarization system is also an option, but that still requires manually annotated data for training purposes. A better option is to apply weakly supervised learning, end-to-end, using coarse grained labels or utterance-level labels. This can make direct use of the training data that only contains a set of speaker identity labels, but no time information. This can reduce the cost, and make use of a large amount of training data without the need for manually annotated time labels for each speaker.

Chapter 6 proposes the first end-to-end weakly supervised speaker identification approach. Firstly, the developed hierarchical attention network (proposed in Chapter 3) is adapted to the weakly supervised speaker identification task since it was shown to be more robust in noisy conditions. The experimental results show that, although the hierarchical attention network can deliver better performance than the widely used X-vector and attentive X-vector baselines, it cannot capture multiple speaker properties from one input utterance.

In order to overcome this problem and obtain better performance, a novel hierarchical transformer network is proposed that captures the information from multiple speakers in a single input utterance through the use of the multi-head attention mechanism. The experimental results show the multi-head attention mechanism can capture the information of two speakers from a two-speaker signal. Another advantage of the hierarchical transformer network is the novel memory mechanism that allows the frame-level encoders to share information, which can help the model to capture multiple speaker information and improve the performance.

The contributions of Chapter 6 are:

- Introduces the first end-to-end weakly supervised speaker identification approach that can directly learn from the coarse grained labels of the multi-speaker input signals.
- Introduces a weakly supervised training strategy of the hierarchical attention network.
- Introduces a novel hierarchical transformer network that makes use of the multi-head attention mechanism to capture the features of multiple speakers in a single input utterance.
- Introduces a memory mechanism in the hierarchical transformer network that can store and share information between each frame-level encoder to deliver a better performance.

1.1.5 List of Publications

The publications refer to the works presented in this thesis are:

- Shi, Y., Huang, Q. & Hain, T. "H-VECTORS: Utterance-Level Speaker Embedding Using A Hierarchical Attention Model" in "ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)", IEEE, pp. 7579–7583.

- Shi, Y., Huang, Q. & Hain, T. Robust speaker recognition using speech enhancement and attention model, in "Odyssey 2020 The Speaker and Language Recognition Workshop" pp. 451–458.

- Shi, Y., Huang, Q. & Hain, T. "Speaker re-identification with speaker dependent speech enhancement", in Interspeech 2020 pp. 1530–1534.

- Shi, Y., Huang, Q. & Hain, T. "Weakly supervised training of hierarchical attention networks for speaker identification", in Interspeech 2020 pp. 2992–2996.

- Shi, Y. & Hain, T. "Supervised speaker embedding de-mixing in two-speaker environment", in "IEEE Spoken Language Technology Workshop 2021 (SLT2021)", pp 758-765.

- Shi, Y. & Hain, T. "Contextual joint factor acoustic embeddings", in "IEEE Spoken Language Technology Workshop 2021 (SLT2021)", pp 750-757.

- Chen, M., Shi, Y. Huang, Q. & Hain, T. "Towards Low-Resource StarGan Voice Conversion Using Weight Adaptive instance Normalization" in "ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)"

- Shi, Y., Huang, Q. & Hain, T. "H-VECTORS: Improving the Robustness in Utterance-level Speaker Embedding Using A Hierarchical Attention Model" in "Neural Networks – Special Issue on Advances in Deep Learning Based Speech Processing".

- Shi, Y., Huang, Q. & Hain, T. "Weakly supervised speaker desertification in multi-speaker scenarios with hierarchical speech segment encoding" in "the special issue of Computer Speech and Language on Separation, Recognition, and Diarization of Conversational Speech" (submitted on review).

1.2 Thesis Outline

The remainder of this thesis is organised as follow:

- Chapter 2 discusses the literature on deep learning based speaker recognition methods, including the input features, loss functions, evaluation metrics and deep neural network architectures. There then follows a discussion on the impact of background noise and interfering speakers, and the ways in which the developed methods, including the attention mechanism, speech enhancement, target speaker extraction and weakly supervised speaker identification, are designed to overcome these two issues. The datasets that were used for the experiments are introduced. Finally, a baseline system is built to show how background noise can influence the performance of deep neural network models.

- Chapter 3 proposes the hierarchical attention network. The model generalisation ability and the noise robustness of the hierarchical attention network are evaluated.

- Chapter 4 proposes two novel joint speech enhancement and speaker recognition systems. The first contains a joint training framework and a novel multi-stage attention mechanism; the second is the speaker dependent speech enhancement model, which makes use of the pre-trained embedding to deliver better performance.

- Chapter 5 proposes the first embedding de-mixing approach, which includes the speaker embedding de-mixing, content embedding de-mixing, and speaker and content embedding de-mixing approaches.

- Chapter 6 proposes the first end-to-end weakly supervised speaker identification method. The developed hierarchical attention network is firstly adapted to this task, and then a novel hierarchical transformer network is proposed to further improve the performance.

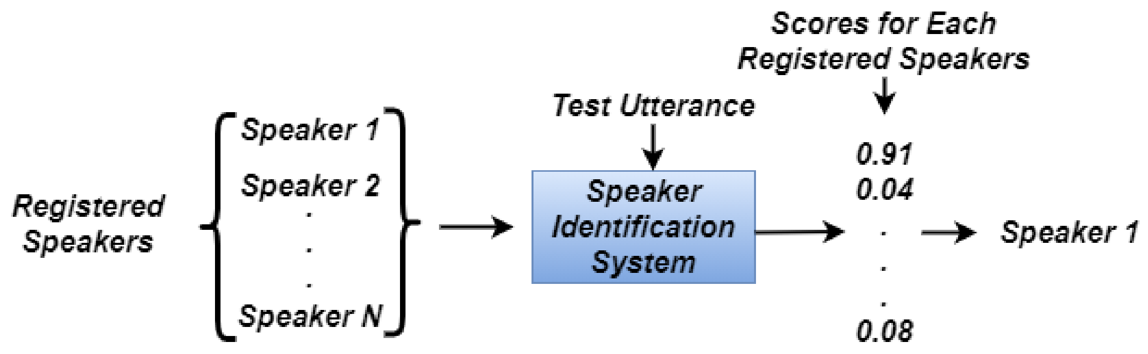
- Chapter 7 provides a conclusion for the whole thesis.

Chapter 2

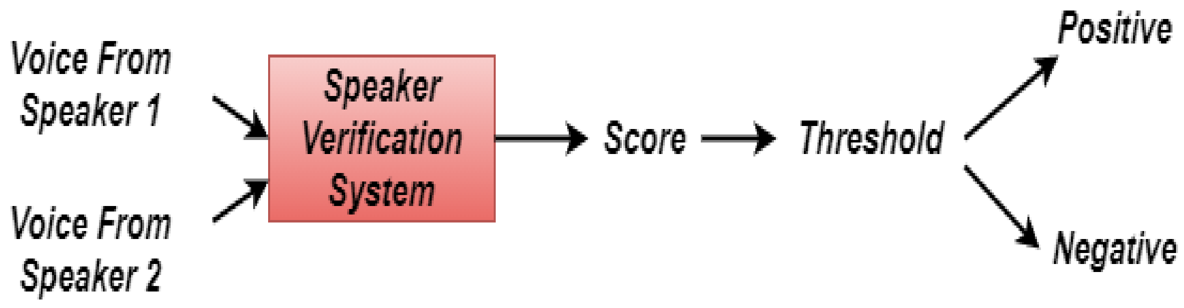
Automatic Speaker Recognition

The goal of speaker recognition is to automatically recognise the identity of a speaker from his/her voice, using computers. According to Campbell (1997) and Kinnunen & Li (2010), the field broadly divides into two branches, namely speaker identification and speaker verification. Speaker identification aims to match a speaker from the registered speakers and assumes that data from the test speakers are available when training the speaker identification model. The relevant diagram for speaker identification is shown in Figure 2.1(a), where N speakers are registered by the system in advance. The speaker identification system takes the input of the test utterance and outputs a similarity score that measures the similarity between the test utterance and the voices of all the registered speakers. The identity of the speaker of the test utterance is classified into the same class as the registered speaker who obtains the highest score. This is a closed-set task, where the speakers in the enrollment database are registered beforehand (Reynolds 2002, Hansen & Hasan 2015). There is also an open set version in which, when none of the registered speakers matches the test speaker, the test speaker can be classified as "unknown" speaker. In this thesis, only the closed-set scenario will be discussed.

Speaker verification is an open-set task (see Figure 2.1(b)), it aims to authentication of a claimed identity from measurements on the voice signal. One of the most commonly



(a): The diagram of a speaker identification system



(b): The diagram of a speaker verification system

Figure 2.1: *Diagrams of (a): a speaker identification system and (b): a speaker verification system.*

used application of speaker verification is that the model has no prior information about the speakers and is simply attempting to ascertain whether or not two input utterances belong to the same speaker. This system computes a similarity score of the two input utterances, and a threshold determines whether the two input utterances are spoken by the same speaker (Jin & Yoo 2010, Campbell 1997).

Speaker recognition can be further divided into text-dependent or text-independent modes (Jin & Yoo 2010). In the text-dependent mode, the words that are spoken by the target speakers are pre-defined, while in the text-independent mode, the speakers are allowed to talk freely (Kinnunen & Li 2010). In this thesis, only the text-independent mode will be discussed.

According to Reynolds & Rose (1995), one of the earliest speaker identification systems was based on Gaussian mixture model (GMM). GMM is the combination of different Gaussian density functions (PDFs). This method is commonly used to model multivariate data and, when applied to speaker recognition, it outputs a speaker-dependent GMM that can then be evaluated, at various data points, to calculate the similarity between a speaker-dependent GMM and the data of an unknown speaker. In one of the simplest versions of the GMM-based method, a GMM is generated for each known speaker. Then, a test utterance is compared to each GMM for known speakers, and the GMM providing the highest similarity score is then assumed to be the GMM for the speaker of the test utterance .

In practice, it is difficult to collect enough data to train the GMM for each target speaker. To address this problem, the GMM-UBM was proposed by Reynolds et al. (2000). The universal background model (UBM) is used as the "universal" model that models the background data and the target speaker's GMM is then adapted using the maximum a posterior (MAP) method (Greig et al. 1989) by adjusting the parameters of the trained UBM instead of directly training the GMM for each speaker.

One of the biggest challenges for early speaker recognition systems was to compare two utterances of different durations. As Markel et al. (1977) pointed out, recognition effectiveness can be improved by generating and evaluating fixed-dimensional representations of each utterance. Achieving this opens up the possibility of using any one of a variety of classifiers developed in various machine learning studies, one of which is to form a GMM supervector by concatenating the parameters of a GMM model in order to generate a fixed-dimensional vector from a variable-duration utterance (Kuhn et al. 1998, Kenny et al. 2003). Campbell et al. (2006) found that GMM supervectors can be successfully deployed for the purposes of speaker identification and verification using support vector machines (SVMs) (Cortes & Vapnik 1995), whereby the positive examples were the supervectors generated from the training utterances, while the negative examples were a set of imposter utterances.

However, according to Campbell et al. (2006), the GMM supervector tends to be so large that reduces the performance of the back-end classification module, making them difficult to train effectively. Furthermore, there is a second problem identified by Dehak et al. (2010) that conventional GMM-UBM systems are highly sensitive to variations in the utterances, whether from the channel or from the speaker themselves.

An attempt was made to address these issues by Dehak et al. (2010) who used joint factor analysis (JFA) (Kenny 2005) to extract speaker features that were used as input into the SVM classifier (Dehak et al. 2009). Then, speaker and channel (the signal path) factors can be combined into a single space called the total variability space, having realised that channel factors can contain speaker-dependent information. According to Dehak et al. (2010), a speaker and session dependent GMM supervector can be generated from the total variability space and from the hidden variables. The hidden variables initially known as the total factors which, while not observable, can be estimated by their posterior expectation and used as features for the next stage classifiers. These later became known as identity vectors in speaker identification systems, or sometimes intermediate vectors, as they are in the intermediate space between a supervector and an acoustic feature vector. Both cases were then shortened to i-vectors and this approach, unlike the use of JFA, does not distinguish between speaker and channel, but is rather a method to reduce the dimensions of the GMM supervector.

The success of the i-vector eventually led to the extraction of speaker embeddings in both identification and verification systems (the workflow of which is shown in Figure 2.2) becoming the most widely used approach (Dehak et al. 2009) since fixed length speaker embeddings can represent speaker features from signals of variable length, making the back-end classifier easier to train. The input waveform is first pre-processed (e.g. using short-term Fourier transform (STFT), see Section 2.2.1) and the acoustic features are generated using one of the various acoustic feature extraction techniques (see Section 2.2). After pre-processing, the acoustic features are input into a machine learning model (or speaker feature extraction method), which extracts the speaker features and projects them into the embedding space. The output are fixed length

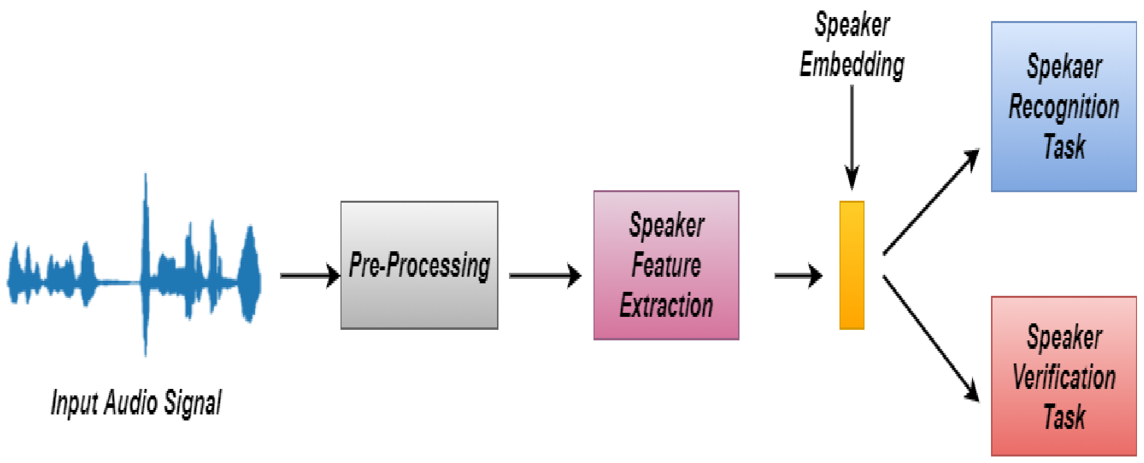


Figure 2.2: *The workflow of a speaker embedding extraction method for speaker identification and speaker verification tasks.*

embeddings, known as speaker embeddings. In this workflow, it is the speaker feature extractor that extracts the i-vector, and the i-vector is equivalent to a speaker embedding. The registered speakers in the speaker identification system are represented by the speaker embeddings, and the speaker verification system computes the score between two embeddings, rather than the two utterances.

More recently, deep neural networks, with their large capacities and powerful feature extraction capabilities allowed for considerable leaps in performance in areas such as computer vision and natural language processing (Wu & Chen 2015, Singh et al. 2017). Several studies, including Snyder et al. (2018) and Variani et al. (2014) have demonstrated that deep neural networks out-perform the GMM models with regard to feature extraction. Thus, in the workflow in Figure 2.2, the speaker feature extractor can be replaced with a deep neural network, and the speaker embeddings can be extracted from a bottleneck layer of the deep neural network, which are called deep speaker embeddings or deep embeddings (Bai & Zhang 2021).

The rest of this chapter is organised as follows:

- Section 2.1 briefly discusses typical deep neural network architectures, including feed forward neural networks (DNN), convolutional neural networks (CNN) and recurrent neural networks (RNN).

- Section 2.2 focuses on the deep learning techniques in speaker recognition, including the input features, loss functions, evaluation metrics and the typical neural network architectures in speaker recognition that have been proposed in recent years.
- Section 2.3 discusses the issue of the influence of background noise and noise robust speaker recognition models, using deep learning, that have been developed in recent years, including attention mechanism and speech enhancement.
- Section 2.4 discusses the techniques for tackling interfering speaker, including target speaker extraction techniques and weakly supervised speaker identification.
- Section 2.5 introduces widely used datasets that have been recorded under various conditions.
- In Section 2.6, a baseline system is designed to show how noise can influence deep learning based speaker recognition systems, and the results are provided, together with a discussion.
- Section 2.7 provides a summary.

2.1 Deep Neural Networks

2.1.1 Feed-Forward Neural Network

Deep neural networks have become the state-of-the-art method in the field of speaker recognition due to their significant capacity, strong feature extraction ability and flexible network architectures (Snyder et al. 2018, Variani et al. 2014, Bai & Zhang 2021).

Feed-forward neural networks (FFN), or multi-layer perceptrons (MLP) is one of the most basic neural network architectures and, for the purposes of this thesis, the two terms will be used interchangeably as DNN. It is a bio-inspired model and the concept dates back to the 1940s (McCulloch & Pitts 1943).

A DNN defines a non-linear mapping $y^{'} = f(\boldsymbol{x}; \boldsymbol{\theta})$ from the input data sample $\boldsymbol{x}$ to the score of the class label $y^{'}$, parameterized by the model parameters $\boldsymbol{\theta}$. The mapping function f, or the network contains multiple layers of perceptrons. One perceptron (or neuron) takes J inputs $x_1, x_2, ..., x_J$, and computes the weighted sum of the inputs $z = \sum_{i=1}^{J} w_i x_i + b_i$, where w_i and b_i are the weight and bias to the input x_i (Waibel et al. 1989, Goodfellow et al. 2016). The output is then input to an activation function that defines the threshold and maps the input to output in a non-linear fashion.

The multilayer perceptron organises the perceptrons into multiple layers. Figure 2.3 shows a diagram of a DNN. With the exception of the input and the last layer, the inner layers are called hidden layers that non-linearly maps the output of the previous layer to the next layer: $\boldsymbol{h}_l = \sigma(\boldsymbol{W}_l \boldsymbol{h}_{l-1} + \boldsymbol{b}_l)$, where $\boldsymbol{h}_l$ and $\boldsymbol{h}_{l-1}$ denotes the output of the current and previous layer, $\boldsymbol{W}_l$ and $\boldsymbol{b}_l$ denotes the weights and bias of the lth layer, and σ denotes the activation function. A typical sigmoid activation function is shown in Equation 2.1 (from Ito (1991)).

$$\text{Sigmoid}(z) = \frac{1}{1 + e^{-z}} \tag{2.1}$$

The sigmoid function can map its input to the numbers between zero and one. Another commonly used activation function is the Hyperbolic Tangent (Tanh), which

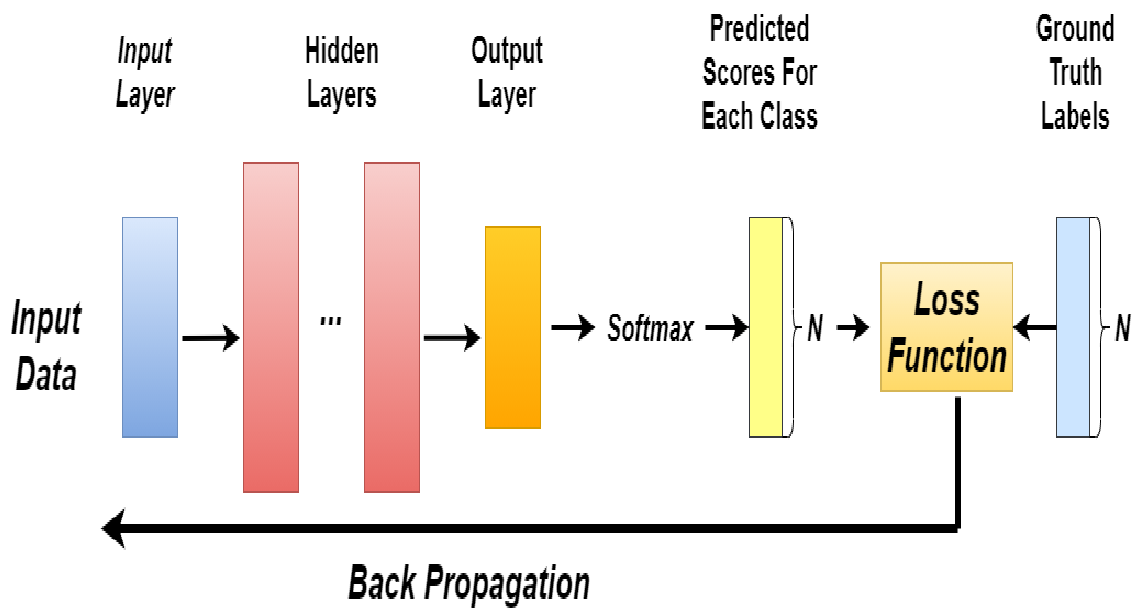


Figure 2.3: *The diagram of the DNN with softmax loss function and back propagation.*

maps its input to -1 and 1 (Karlik & Olgac 2011).

However, according to Nair & Hinton (2010), both Sigmoid and Tanh activations can cause a gradient vanishing problem. When the input is large or small, the sigmoid function will output zero or one, and the Tanh function will output -1 or 1, with a derivative extremely close to 0 which can cause the DNN to cease learning during the training stage. Another problem is the computationally expensive problem, where both sigmoid and Tanh have an exponential operation. To solve these problems, a rectified linear unit (ReLU) was proposed by Nair & Hinton (2010) (shown in Equation 2.2).

$$\text{ReLU}(x) = \begin{cases} 0 \quad if \quad x \leq 0 \\ x \quad if \quad x > 0 \end{cases} \tag{2.2}$$

Typically in a multi-class classification task, in the last layer of the DNN, the model will output the predicted scores for each of the classes which is modelled by the softmax function (Bridle 1989). Equation 2.3 shows the softmax function, where z_i denotes the score for the ith class and N denotes the total number of classes. The softmax function can normalize the output of a DNN and make all of the probabilities sum to one, which provides a probability distribution over classes.

$$\text{Softmax}(z_i) = \frac{\exp(z_i)}{\sum_{n=1}^{N} \exp(z_n)} \tag{2.3}$$

After the output of the softmax function, the cost function or loss function measures the difference between the predicted scores and the ground truth labels for each class. The commonly used loss function for multi-class classification tasks is the categorical cross-entropy loss (shown in Equation 2.4).

$$\mathcal{L} = -\sum_{i=1}^{N} y_i \text{log} y_i^{'} \tag{2.4}$$

The softmax function combined with the cross entropy loss is commonly referred to as the "softmax loss" (Goodfellow et al. 2016) in which $y_i^{'}$ and y_i denotes the predicted score and the ground truth score for the ith class, $i \in \{1, 2, ..., N\}$, N is the total number of classes. The ground truth label is often formulated by the one hot encoding (Waibel et al. 1989, Hinton & Salakhutdinov 2006, Brownlee 2017) which encodes the speaker identities into a vector representation that can be directly used by the neural networks. Suppose $\boldsymbol{y}$ denotes the ground truth label in one hot encoding for an input data. $\boldsymbol{y}$ has the dimensionality of N, where each dimension contains the ground truth score for each class, only the positive class has the value of 1, the negative classes have the values of 0. The predicted scores can also be organised in a similar way, $\boldsymbol{y}^{'}$ denotes the vector of the model prediction for each class, which has the dimensionality of N and each dimension contains the predicted score of the classes using the DNN (LeCun et al. 2015).

Having built the neural network architecture, the next step is the optimisation or training of the neural networks that is implemented by the gradient descent method (Hinton & Salakhutdinov 2006). The gradient descent is used to find the values of the neural network parameters that minimises the cost function, thereby reducing the error between the predicted scores and the ground truth labels to make the DNN perform better. In practice, gradient descent updates each of the parameters in a

DNN iteratively (LeCun et al. 2015). Equation 2.5 shows an example of the gradient descent of a parameter w.

$$w^{'} \leftarrow w - \eta \nabla(w) \tag{2.5}$$

For each iteration, w subtracts its gradient $\nabla(w)$ that scaled by a scalar number η which is called the learning rate. The gradient $\nabla(w)$ is the partial derivative of the loss function to the parameter w (Bottou 2012).

According to (Li et al. 2014), the three gradient descent methods – batch (also known as vanilla), stochastic and mini-batch – differ primarily in the amount of data they use. Batch gradient descent calculates the error for each data within the training set and then updates the model once all examples have been evaluated; this cycle is known as a training epoch. Although this method produces a stable error gradient and convergence, these are not always the best possible from the model. Furthermore, as (Ruder 2016) points out, the batch gradient descent model does require the entire training set to be in the memory and available to the algorithm. The stochastic gradient descent model (SGD), on the other hand, updates the parameters after each example, and such frequent updates can make it faster with certain problems (Bottou 2012). The cost of this is an increase in computational cost, and the increased update frequency can lead to noisy gradients, with an erratic error rate as opposed to a gradual decrease. Finally, the mini-batch gradient descent method is a combination of the former two, splitting the dataset into small batches and updating each in turn, which delivers a balance between the efficiency of the batch gradient descent and the robustness of the SGD method. Due to the mention parallel processing in GPUs (Graphic Processing Unit), mini-batch gradient decent is the most widely used (Li et al. 2014).

In the SGD method, the batch size is one, while in the vanilla gradient descent, the batch size is the total number of samples in the training set. In this thesis, all of the proposed models are trained using the mini-batch gradient descent method, and the batch size is denoted as B.

2.1.2 Convolutional Neural Network

Convolutional neural network (CNN) is another type of deep neural network architecture in which the basic components are the convolutional kernels and the convolutional operation. The term 'convolution' refers to the mathematical combination of two functions, effectively merging two sets of information, to produce a third function. Equation 2.6 shows the convolutional process of two functions $f[x]$ and $g[x]$ to produce a third function $y[x]$, where x is defined as a discrete variable, with k being a discrete time, $*$ is the convolution operator and $\times$ is ordinary multiplication.

$$y[x] = f[x] * g[x] = \sum_{k=-\infty}^{+\infty} f[k] \times g[x-k] \tag{2.6}$$

In the case of a CNN, the convolution is performed on the input data with the use of a filter or kernel to then produce a feature map. Specifically, in a commonly used two-dimensional CNN, a convolutional kernel can be viewed as a small two-dimensional matrix, and the input can be viewed as a large two-dimensional matrix (LeCun et al. 1989, Amari et al. 2003, Goodfellow et al. 2016, Khan et al. 2020).

In a CNN, the convolutional layer contains a set of convolutional kernels, each acting as a single neuron which, according to (Gál et al. 2004), divide the input matrix into small slices, often known as receptive fields, in order to better extract feature motifs. Each kernel convolves with the input matrix by multiplying its elements with those corresponding elements of the receptive field and using a specific set of weights. Figure 2.4 shows an example of the convolutional operation in a two-dimensional CNN. The values in the kernel multiply with the corresponding values in the input matrix and the result is a summation of the values. The convolutional kernel moves horizontally and vertically with a certain step size (known as a stride) to process the whole input matrix. After the activation function, the result is a new matrix called a feature map (Goodfellow et al. 2016). A convolutional layer contains multiple kernels, and a CNN contains a stack of multiple convolutional layers, at the end of which is the DNN that aggregates the learnings from the convolutional layers (LeCun et al. 2015).

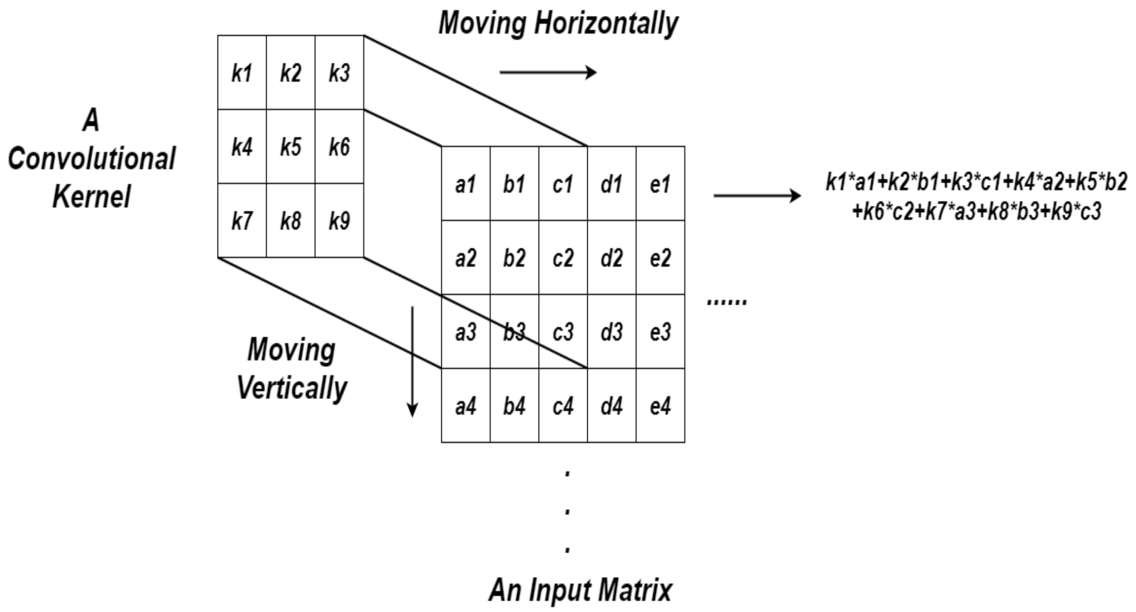


Figure 2.4: *An example of the convolutional operation in CNN.*

The two-dimensional CNN was firstly designed for image processing, and many studies have shown that the convolutional kernels can learn hierarchical features from low to high levels (Goodfellow et al. 2016, LeCun et al. 2015). The receptive fields of the convolutional kernels were shown to be larger when in the deeper layers of a CNN, which helps it to deliver state-of-the-art performance in many tasks, such as image classification and image segmentation (Krizhevsky et al. 2012, Alam et al. 2015). In speech processing, as a one-dimensional speech signal can be represented by two-dimensional spectrograms, with time and frequency axes using the Fourier transform (Oppenheim et al. 2001), the spectrograms can be viewed as two-dimensional images and the convolutional operation introduced above can be applied (Oppenheim et al. 2001, Davis & Mermelstein 1980). Many studies show the two-dimensional CNN was also successfully used for speech processing and reached the state-of-the-art performance in speaker recognition (Xie et al. 2019, Yu et al. 2019).

2.1.3 Recurrent Neural Network

A recurrent neural network (RNN) is a feed forward neural network that models the time series information (Hochreiter & Schmidhuber 1997, Mikolov et al. 2010, Lipton et al. 2015) by including hidden states spanning adjacent time steps. In order to capture the sequence information, the RNN stores the features along each step from the input data into the hidden states, which store the representations of the sequence information over time, and these representations are updated when the input of a new time step comes in. Figure 2.5 shows a diagram of a single layer RNN and Equation 2.7 (from Medsker & Jain (2001)) shows the computation process of one RNN unit, where $x_t \in \mathcal{R}^{1\times F}$, $\boldsymbol{W}_{xh} \in \mathcal{R}^{F\times F}$, $\boldsymbol{W}_{hh} \in \mathcal{R}^{F\times F}$, $\boldsymbol{W}_{hy} \in \mathcal{R}^{F\times F}$ and $\boldsymbol{b}_h \in \mathcal{R}^{1\times F}$ are the trainable weights and bias. F denotes the feature dimension of $\boldsymbol{X}$, σ denotes the activation function, for which Sigmoid and Tanh are two most commonly used. The single layer RNN can also be organized into multiple layers (Mikolov et al. 2010).

$$\begin{aligned}\boldsymbol{h}_t =& \sigma(\boldsymbol{x}_t\boldsymbol{W}_{xh} + \boldsymbol{h}_{t-1}\boldsymbol{W}_{hh} + \boldsymbol{b}_h)\\ \boldsymbol{y}_t =& \boldsymbol{h}_t\boldsymbol{W}_{hy} + \boldsymbol{b}_y\end{aligned} \tag{2.7}$$

The input signal (i.e. a spectrogram) is denoted as $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_T\}$, $\boldsymbol{x}_t \in \mathcal{R}^{1\times F}$ denotes a feature vector of the input signal at time t, $\boldsymbol{h}_t$ and $\boldsymbol{h}_{t-1}$ denotes the hidden states that contain the sequence information in an RNN in time t and the previous time step $t-1$. $\boldsymbol{y}_t$ denotes the output at time t. The RNN receives, as inputs, feature vector $\boldsymbol{x}_t$ at time t and hidden states $\boldsymbol{h}_{t-1}$ in the network's previous state. At each time t, the output $\boldsymbol{y}_t$ is derived from the hidden state $\boldsymbol{h}_t$ at time t, though output $\boldsymbol{y}_t$ can be influenced by input $\boldsymbol{x}_{t-1}$ at time $t-1$ by the recurrent connections.

Schuster & Paliwal (1997) were the first to propose the bidirectional recurrent neural network in which there are two layers of hidden states, both of which are connected to both input and output, but they differ in that the first layer has recurrent connections from past time steps, while the second has the recurrent connections reversed, enabling it to pass activation back along the sequence. This is now one of the most widely used

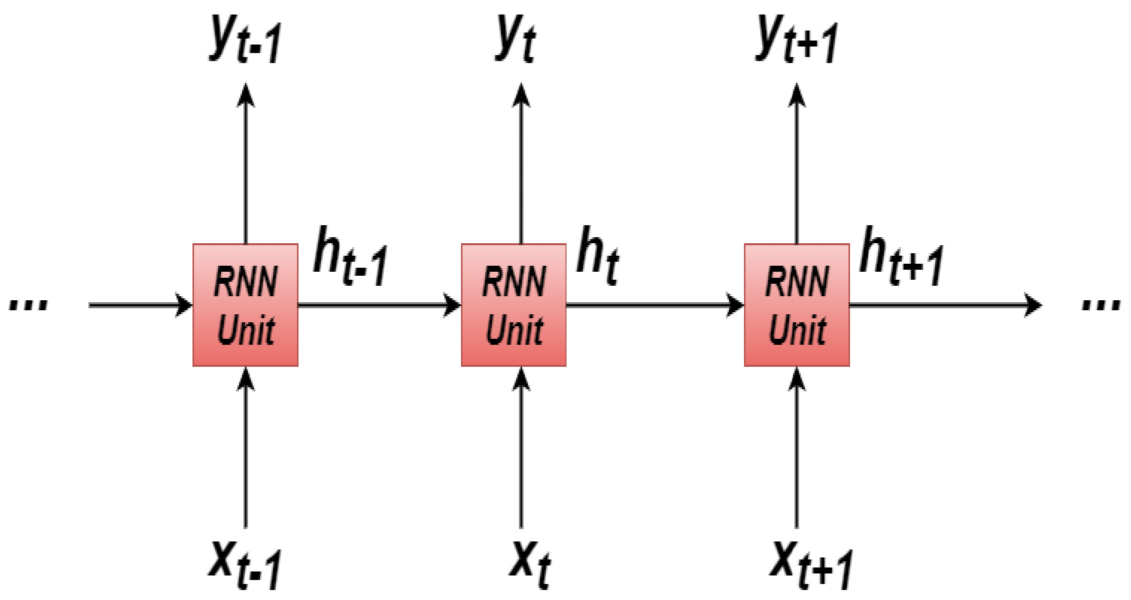


Figure 2.5: *A diagram of a single layer RNN, it can be organized into multiple layers.*

of the RNN architectures.

According to Hochreiter & Schmidhuber (1997), Bengio et al. (1993) and Schaefer et al. (2008), the long-range dependencies built in to RNN architectures create issues around vanishing and exploding gradients. These are caused by the proliferation of backpropagating errors across multiple time steps which, in turn, lead to reduced gradient values over long time sequences, making RNNs particularly difficult to train. This problem is referred to as the vanishing gradient problem. In order to solve this problem, the long short-term memory (LSTM) was intuitively proposed by Hochreiter & Schmidhuber (1997). Simple RNNs already have both long-term memory, in the form of weights that change gradually during the training sequence and encoding general information about the training dataset, and ephemeral activations, that act as short-term memory, passing between successive nodes. LSTM captures the information by splitting the hidden states of the RNN into long-term and short-term memory. This has been shown to out-perform the RNN in many tasks (Prabowo et al. 2018). Another variation of RNN is the gated recurrent unit (GRU) proposed by Chung et al. (2014). The GRU reduced the computational cost of LSTM while retaining the advantages, and it has now become one of the most widely used methods for modelling time series data (Kamath et al. 2019, Khandelwal et al. 2016).

Another widely used method to model sequential information, proposed by (Waibel et al. 1989), is the time delay neural network (TDNN), which is a DNN with the addition of multiple, interconnected layers of perceptrons. Neurons on each layer of a TDNN send their outputs to neurons at the layer above, but there are two key differences to a standard DNN. First, every unit in a TDNN, at every layer, receive inputs from a contextual window of outputs from the layer below; and second, in addition to these connections from the layer below, each unit is also connected to the time-delayed outputs from the same lower units, thereby providing a model of the temporal pattern and trajectory of each unit. Two-dimensional input signals, such as spectrograms, produce a 2-dimensional context window at each layer and, since the inputs into the higher layers come from wider context windows, these higher layers are able to model coarser levels of abstraction. Peddinti et al. (2015) has indicated that TDNN is an efficient method compared to RNN-based models and it operates a one-dimensional convolution along the temporal dimension to capture temporal context information of the sequence inputs. Several studies show that TDNN-based deep neural network models perform well in speaker recognition (Snyder et al. 2018, Wang, Okabe, Lee, Yamamoto & Koshinaka 2018).

2.2 Deep Learning For Speaker Recognition

Due to the advantages of the deep neural network models, deep learning has been widely used in speaker recognition and is now the state-of-the-art method (Bai & Zhang 2021, Hansen & Hasan 2015). This section reviews the techniques used in deep learning based speaker recognition, including the input features, loss functions, evaluation metrics and the typical neural network architectures.

2.2.1 Input Features

It has been found that taking the raw waveform directly as input to a neural network creates issues around dimensionality, as well as slowing the training process. Given that frequencies in the waveform are resolved in a non-linear manner by the human ear, it is better practice to extract acoustic features for the network to process, rather than the full waveform (Davis & Mermelstein 1980, Zhang 2017).

In order to achieve this, an audio waveform is usually split into short segments using a sliding window (25ms length and 10ms step are used in all the experiments in this thesis). Then, a short-term Fourier transform (STFT) (Oppenheim et al. 2001) is applied to each segment and the magnitudes spectrogram are remained. In some cases, the spectrograms can be used directly as input features (He et al. 2016, Yu et al. 2019). To further reduce the dimensionality, the spectrogram is then passed over a set of triangular Mel-scale frequency filters (Davis & Mermelstein 1980) which is shown in Equation 2.8 (Zhang 2017, Davis & Mermelstein 1980).

$$\mathrm{Mel}(f) = 2595\log_{10}(1 + \frac{f}{700}) \tag{2.8}$$

The result of this process are filterbank features. The log amplitudes the filterbank features are also widely used for neural network inputs in speaker recognition (Zeinali et al. 2019).

The log amplitudes of the filterbank features can be further computed with a discrete cosine transform (DCT) (Oppenheim et al. 2001). The DCT linearly de-correlates the filterbank features and results in the cepstral vectors (Zhang 2017, Oppenheim et al. 2001, Davis & Mermelstein 1980). The MFCC features is one of the most widely used input features for neural networks in speech recognition (Zhang 2017) and in speaker recognition (Snyder et al. 2018).

2.2.2 Loss Functions

The performance of deep neural networks is highly determined by the loss functions. The loss function measures the difference of the model predictions and the ground truth labels, and the back propagation algorithm relies on the the loss function. Many studies shown that a good loss function can improve the performance of the model (Wang, Cheng, Liu & Liu 2018, Deng et al. 2019).

According to Irum & Salman (2019) and Bai & Zhang (2021), one of the most commonly used workflows for obtaining speaker embeddings is to firstly train a network using a multi-class classification manner with a softmax function, combined with cross-entropy loss. Then, the speaker embeddings are extracted from the trained model. Usually, as introduced in Section 2.1, the softmax function is used in the last layer of a neural network to convert the scores for each of the classes to between zero and one. It also guarantees that the scores for all of the classes can be summed up into one score to simulate the probability process (Bouchard 2007).

It has been indicated by Irum & Salman (2019) that a good speaker embedding extractor must maximise the inter-class distance between the embeddings that belong to different speakers, while minimising the intra-class distance between the embeddings that belong to the same speaker. In speaker identification or verification, maximising the inter-class distance can make the learned speaker embeddings easier to separate, and minimising the intra-class distance can reduce speaker variance within different embeddings from one speaker (Bai & Zhang 2021).

However, Irum & Salman (2019) and Huang et al. (2018) identified that the deep neural networks that trained using the softmax loss function can maximise the inter-class distance but without any constraint on the intra-class distance. In order to minimise the intra-class distance, several extensions, based on the softmax function, were proposed. For example Li et al. (2018) proposed an Angular softmax (A-softmax) function that added a margin to control the decision boundary by controlling the angle between the embeddings during the training stage. It have shown that embeddings learned through this A-softmax function have smaller intra-class distances, thereby performing better in speaker verification tasks. Other loss functions were inspired by this work, including the additive margin softmax (AM softmax) proposed by Wang, Cheng, Liu & Liu (2018) and the additive angular margin softmax (AAM softmax) from Deng et al. (2019), both of which made use of the different margins to control the intra-class distances in the speaker embeddings.

2.2.3 Evaluation Metrics

Evaluation metric measures the performance of the model and is used after the training stage. This section introduces the commonly used evaluation metrics for speaker recognition that will be used in most of the works in the following chapters.

Prediction Accuracy

At the beginning of this chapter, the difference between speaker identification and verification was discussed. Speaker identification can be viewed as a multi-class classification task, where the different speakers represent different class labels. Thus, the evaluation metric in speaker identification is the same as that in other classification tasks, such as audio classification. The most commonly used metric is classification accuracy (Rosenfield & Fitzpatrick-Lins 1986, Aronoff et al. 1982). Classification accuracy or prediction accuracy indicates the rate of the correct predictions in the total test samples. In some cases, classification accuracy can be further divided into top-1 accuracy and top-K accuracy. In top-1 accuracy, for each data sample, when the highest probability of the model prediction matches the class label, the prediction can be viewed as being correct. While in top-K accuracy, when the ground truth label is in the Kth highest probability predictions of the model, the prediction can be viewed as being correct. The top-K accuracy is greater or equal to the top-1 accuracy, the latter of which is most commonly used in speaker identification.

In speaker verification, on the other hand, the goal is to identify whether or not two input utterances belong to the same speaker. The output of the speaker verification system will be a similarity score and a threshold will determine whether the output is positive or negative. In this case, rather than using classification accuracy, equal error rate (EER) is better way to evaluate the system (Oglesby 1995, Sztahó et al. 2019).

Equal Error Rate (EER)

In the evaluation of speaker verification, if an error is made, it will either be a false accept (FA) or a false reject (FR). If the speaker verification system is looking to authenticate speakers, then an FA error effectively grants access to an imposter speaker, while an FR error denies access to a legitimate one (Hansen & Hasan 2015). Measuring the number of FA errors from a given number of imposter attempts gives the false accept rate (FAR) while the number of false reject errors in a given number of legitimate attempts gives the false reject rate (FRR). In a speaker verification system, a FA error is known as a false alarm, while a FR error is known as a miss error.

The output from a typical speaker verification system is a scalar score between the two utterances, as is the case for most two-class recognition/binary detection problems. The higher the score value, the more similar they are. According to (Oglesby 1995), the key variable here is the threshold, τ, which, if too high, creates a high number of FA or false alarm errors; too low, and it conversely creates a high number of FR or miss errors. By adjusting the threshold, it is possible to reach a point at which FAR = FRR, and that is known as the equal error rate (EER). Although the FAR = FRR state of equilibrium that produces the EER may seem a compelling point at which to operate a system, it may not always be ideal from a practical perspective. For example, in a high security setting, like a bank, it makes sense to set a higher threshold, reducing FA errors in favour of FR ones. It is better, in this case, to deny legitimate users access (absent some other form of ID) rather than grant access to an imposter. Conversely, where convenience is more important than security, such as in an automated customer service system, the threshold can be set lower, as granting access to an imposter does not have any dire or long-term consequences (Hansen & Hasan 2015).

A widely used technique for visualising the performance of a binary class classification task is the receiver operating characteristic (ROC) curve. The true positive (TP) rate and the false negative (FP) rate are visualised on a two dimensional plane. However, in speaker verification, the FAR and FRR are usually smaller than 10%, it is inconvenient to observe the ROC curve. As a result, the most common method of visualising the performance of a speaker verification system is the detection error trade-off curve (DET) that plots FA errors against FR errors (Martin et al. 1997, Garcia-Perera et al. 2012). The two axes of DET curve are not linear, but scaled by the the inverse function of the standard Gaussian cumulative density function.

Detection Cost Function (DCF)

While the EER has its uses, it is not always the preferred performance measure as it does not distinguish between FA and miss errors, whereas the DCF measures numerical costs and penalties for both error types. DCF was introduced by the NIST SRE 2008 Martin & Greenberg (2009) challenge.

Equation 2.9, taken from Martin & Greenberg (2009), shows how the DCF is computed across the entire range of decision threshold values, where C_{FR} is the cost of a miss/FR error, C_{FA} is the cost of an FA error, P_{Target} is the a priori probability of target speaker, E_{FR} is the probability of $(\text{Miss}|\text{Target}, \text{Threshold} = \tau)$, E_{FA} is the probability of $(\text{FA}|\text{Nontarget}, \text{Threshold} = \tau)$.

$$DCF = C_{FR}E_{FR}P_{Target} + C_{FA}E_{FA}(1 - P_{Target}) \tag{2.9}$$

C_{FR}, C_{FA} and P_{Target} are pre-defined and they can be set according to different application environments. For example, in the military access control system, it is desirable to strictly control the entry and exit of people, thus the probability of false acceptance is relatively small, then the value of C_{FA} can be increased; When monitoring the voice of criminals, it requires not to miss possible target voices, so the value of C_{FR} needs to be increased. When C_{FA}, C_{FR}, P_{Target}, and $1-P_{Target}$ are set, a certain group of E_{FR} and E_{FA} values makes DCF the smallest, and the DCF at this time becomes the minDCF (Hansen & Hasan 2015, Martin & Greenberg 2009). This thesis deploys the settings from NIST SRE 2008 Martin & Greenberg (2009) and Shon et al. (2019), which are the average minDCF from two sets of parameters: $C_{FR} = 10$, $C_{FA} = 1$, $P_{Target} = 0.01$ and $C_{FR} = 10$, $C_{FA} = 1$, $P_{Target} = 0.001$.

2.2.4 Deep Neural Network Architectures

This section discusses three commonly used network architectures that are used for the baselines for the proposed model in the following chapters.

D-vector

Variani et al. (2014) proposed one of the earliest deep learning models for speaker recognition, called the "d-vector", as shown in Figure 2.6 (from Variani et al. (2014)). The network is based on a feed-forward neural network and is trained using a supervised manner with frame-level inputs. The output of the last hidden layer of the network are collected and averaged as the embedding of the utterance. A speaker verification task was conducted using the extracted embeddings. The results show that the i-vector had better performance, but the sum of the d-vector and i-vector out-performed the i-vector, with 14% to 25% relative improvement under both clean and noisy conditions.

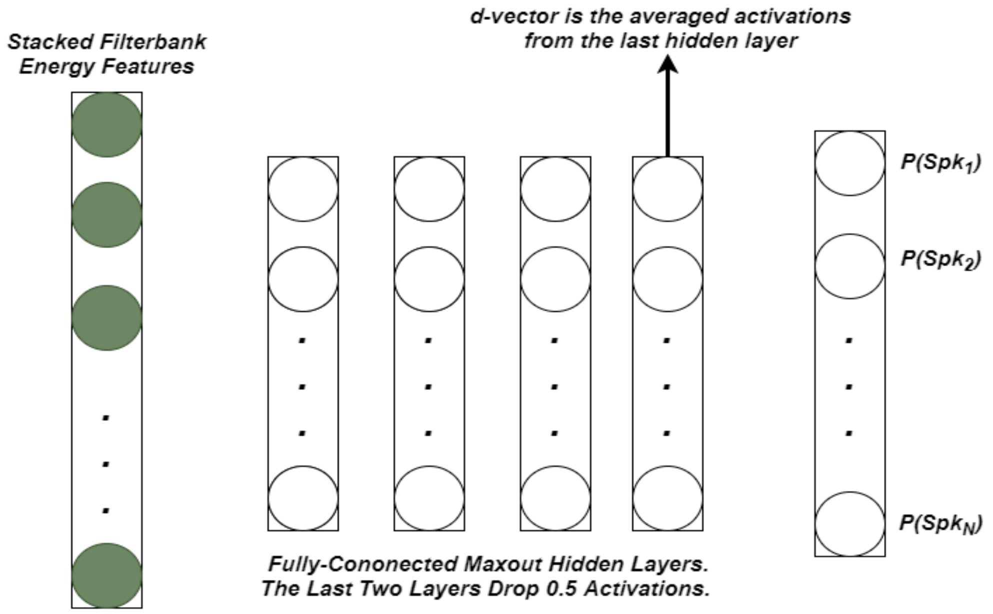

d-vector is the averaged activations from the last hidden layer
Stacked Filterbank Energy Features
P(Spk$_1$)
P(Spk$_2$)
P(Spk$_N$)
Fully-Cononected Maxout Hidden Layers. The Last Two Layers Drop 0.5 Activations.

Figure 2.6: *The architecture of the d-vector model, from Variani et al. (2014).*

X-vector

The d-vector model made use of deep neural networks but it did not out-perform the i-vector in some scenarios, possibly because the frame-level training of the speaker embedding loses some global features of the speakers (Variani et al. 2014). In order to solve this problem, Snyder et al. (2018) proposed an X-vector model that learns the speaker embeddings at the segment level. Figure 2.7, shows the architecture of the X-vector model, which consists of three parts: the frame-level feature extractor, the statistics pooling operation and the segment-level feature extractor. The input sequence is denoted as $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_T\}$, where $\boldsymbol{x}_t$ denotes the temporal frame at time t, and T denotes the sequence length. In the frame-level feature extractor, the model makes use of the time-delay neural network (TDNN) that operates at a different time step from the input utterance. The output of the TDNN layers are the frame-level features, which are denoted as $\boldsymbol{H} = \{\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_L\}$, where $\boldsymbol{h}_t$ denotes the feature vector at time t, L denotes the sequence length of $\boldsymbol{H}$.

After the frame level feature extractor, the statistics pooling operates on a time

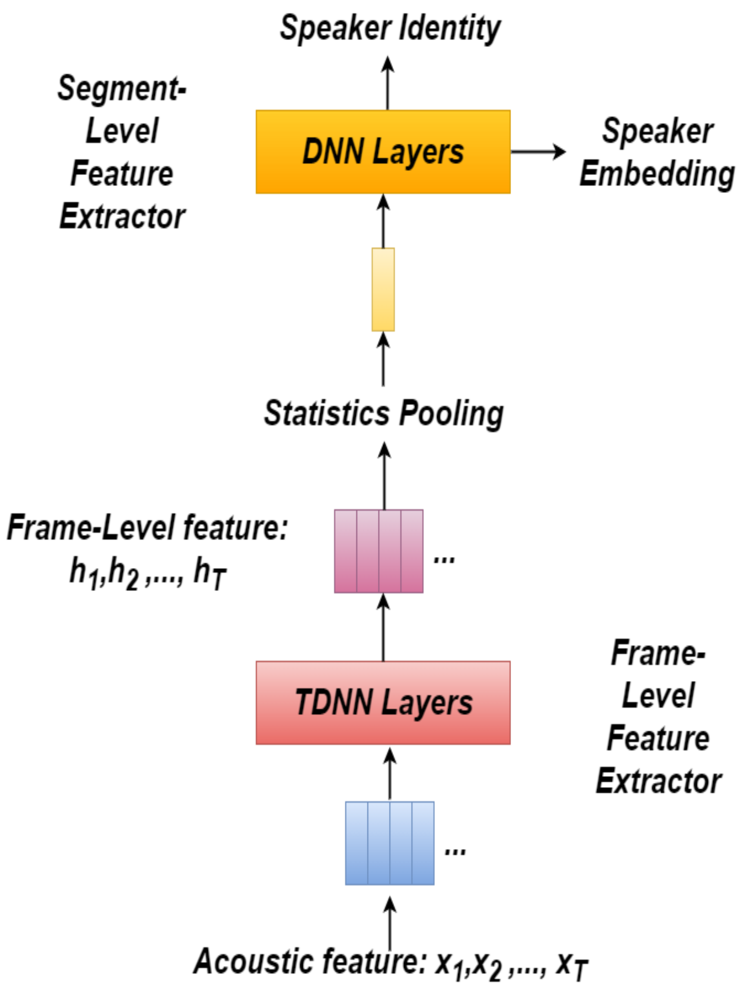

Speaker Identity
Segment-Level Feature Extractor
DNN Layers
Speaker Embedding
Statistics Pooling
Frame-Level feature: $h_1, h_2, \ldots, h_T$
...
TDNN Layers
Frame-Level Feature Extractor
...
Acoustic feature: $x_1, x_2, \ldots, x_T$

Figure 2.7: *The architecture of the X-vector model, from Snyder et al. (2018).*

axis that compresses the sequence output into a single vector (Snyder et al. 2018). The mean and standard deviation (std) vectors ($\boldsymbol{\mu}$ and $\boldsymbol{s}$) are the output of this process. The $\boldsymbol{\mu}$ and $\boldsymbol{s}$ are concatenated into a single vector and fed into the segment-level feature extractor that consists of two fully-connected layers that fuse the learned features and map them into embeddings (Snyder et al. 2018). The speaker embeddings were extracted from the first DNN. The experiments were conducted on the NIST SRE 2016 and the Voxceleb datasets. The results show that the X-vector out- performs the i-vector system, reaching a relative improvement of more than 10%.

ResNet

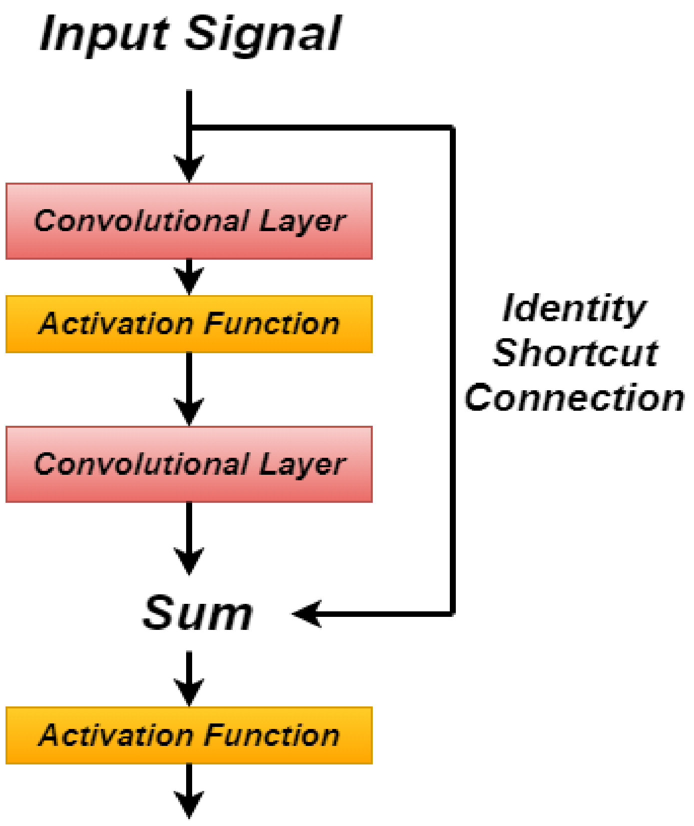
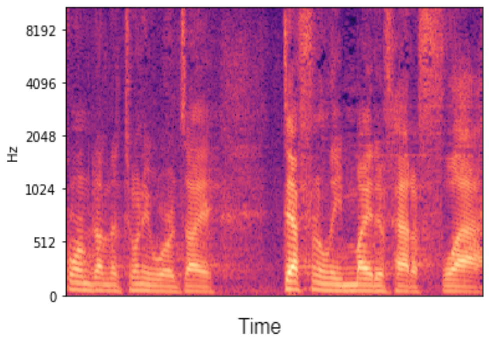


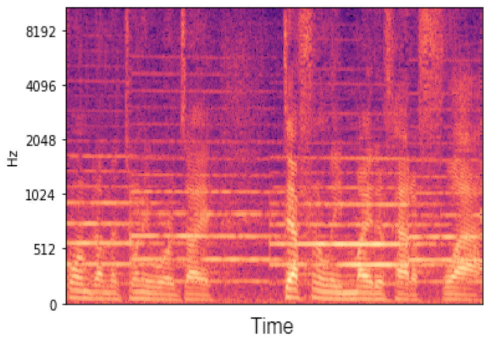
Figure 2.8: *The architecture of a residual block, from (He et al. 2016).*

Another option for speaker recognition is using a CNN model instead of either a DNN or a TDNN. The most widely used model is ResNet, which was first proposed by He et al. (2016) for image classification. The core idea of the ResNet is the identity shortcut connection that skips one or more layers. Figure 2.8 shows an example of a residual block. The input signal is processed by several convolutional layers with activation function, which is referred to as the main path. The identity shortcut connection (or the residual connection) skips the convolutional layers, thus the input signal directly sums with the output of the main path. According to He et al. (2016), the residual connection can make the network more stable in the training step and it can learn more complex feature patterns from the input data.

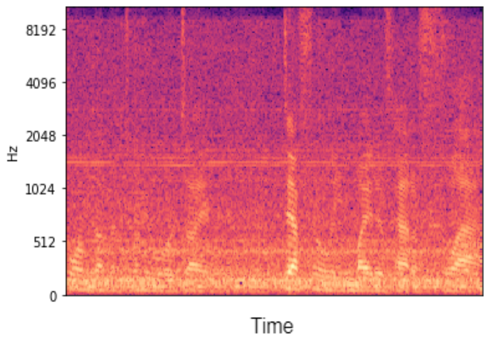
A ResNet contains multiple residual blocks connected end-to-end, based on the number of convolutional layers, and different versions have been proposed, such as ResNet-34 and ResNet-50 (where the number indicates the number of convolutional layers) (Aloysius & Geetha 2017). In speaker verification, various versions of ResNet have been shown to attain state-of-the-art performance. For example, Cai et al. (2018) used ResNet-34 and successfully reached 4.46% EER on the Voxceleb1 test set, while Hajibabaei & Dai (2018) adapted the ResNet-20 architecture for speaker verification and reached 4.30% EER.



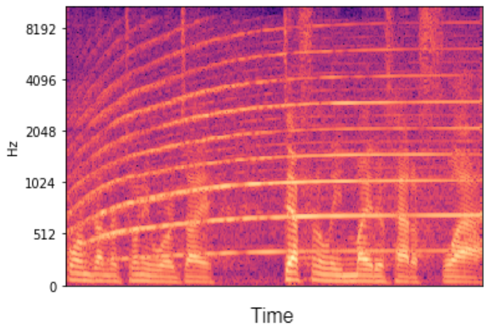
(a) Original



(b) Ambulance whistle



(c) Aircraft engine



(d) Air defence alert

Figure 2.9: *The example spectrograms of (a) speech and the noisy spectrogram corrupted by (b) ambulance whistle; (c) aircraft engine; (d) air defense alert.*

2.3 Noise Robust Speaker Recognition

Although deep neural networks improved the performance of speaker recognition systems, they still suffer from real-world noise interference, making robustness an increasingly essential metric and property (Zhao et al. 2014). This section firstly discusses the background noise types and their various influences on the speech signals. All of the noises discussed in this thesis are additive noises. Then, attention mechanism and speech enhancement methods will be introduced. For each of these methods, recent works will be reviewed and discussed.

2.3.1 Noise Interference

Speech signals are always influenced by background (or environmental) noise in real-world conditions, such distortion can take many forms, such as additive noise and reverberations (Ming et al. 2007). Le Prell & Clavier (2017) investigated the impact of various types of background noise on the speech signal. In their work, they indicated that background noise can potentially cover up (or mask) the information of another sound (e.g. the speaker's voice) and they identified two categories, namely steady noise and fluctuating noise. Steady noise contains the stable spectral and temporal characteristics of features, such as the noise from an idling vehicle. This kind of noise can mask some words spoken by the speaker. However, when the spectrum of the speech and the spectrum of the noise are not completely overlapped, the speech can also be recognised, at least partially.

The other type of noise is fluctuating noise, such as from road traffic, which contains two variant components: the spectral fluctuation where the frequency components changes rapidly over time, and the temporal envelope variation where the noise level varies over time.

Typical background noise in the real world contains both types, but the fluctuating noise is the most common. Figure 2.9 shows some examples of noise corrupted spectrograms. Figure 2.9(a) illustrates the speech and Figure 2.9(b), 2.9(c) and 2.9(d) are the spectrograms after they have been corrupted by different types of background noise: an ambulance siren, an aircraft engine, and an air defense alert. Note that the noises discussed in this section are additive noise, the effects of high level noise on the speech produced by a speaker (Lombard effect) is not included (Lane & Tranel 1971).

According to Le Prell & Clavier (2017), there is also a typical background noise type that caused by the speech shaped noise. This kind of noise is referred to as the babble noise, and it represents background noise that contains energy in the same frequency band as the speech.

Both steady and fluctuating background noise corrupts the speech signal across frequency and time domains in ways that can significantly and negatively affect the performance of deep neural networks. Zhao et al. (2014) and Ming et al. (2007) investigated how the performance of the DNN model can be affected by various types of noise conditions. In Ming et al. (2007), different noise types such as engine noise, restaurant noise and pop songs, all at different levels ranging from the signal-to-noise-ratio (SNR) is 10dB to 20dB (Johnson 2006), are artificially mixed with the clean signal. Then, a DNN model was used to evaluate the performance. In the clean condition, the speaker identification accuracy can reach 90.64% using the TIMIT dataset. However, with a noise level of 20dB, the accuracy can reduce to between 87% and 83%, depending on noise type. When the noise level is 10dB, the accuracy in some noise conditions (e.g. a mobile phone ringing) can be lower than 50%.

2.3.2 Attention Mechanism

A method that may improve the robustness of deep neural networks in various noise conditions is the so called attention mechanism (Bahdanau et al. 2015), which is used to allocate different weights to different parts of the input data, which can then highlight the information that is relevant to the targets (Hao et al. 2019). The method allows the model to pay more attention to the region that contains more information of the target speaker, while paying less attention to the region that is distorted by background noise (Zacarias-Morales et al. 2021, Yuan et al. 2020).

Bahdanau et al. (2015) were the first to propose the use of an attention mechanism within a neural network that was specifically designed for a sequence-to-sequence modelling task, a task which, according to (Cho et al. 2014), is based on a encoder-decoder architecture, both of which are RNNs. The encoder RNN accepts an input series of tokens, and the decoder RNN takes a single, fixed length vector as its input and generates an output sequence from it (see Figure 2.10(a)).

The first challenge with this architecture is that the compression by the encoder of all the long and detailed input information into a single, fixed length vector can

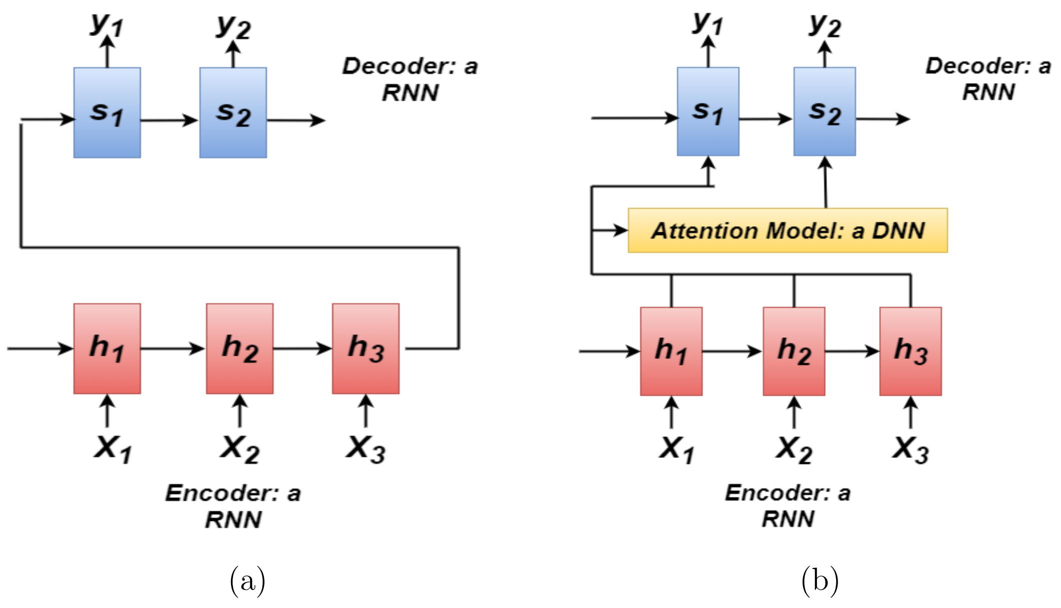


Figure 2.10: *Encoder-decoder architecture: (a) traditional (b) with attention model*

lead to a loss of information (Cho et al. 2014). The second challenge is the inability of the framework to model alignment between the input and output sequences, thereby making it unusable for tasks requiring a structured output, such as translation or summarisation.

An attention mechanism allows the decoder to access the entire encoded input sequence which, in theory, serves to remove, or at least reduce, the impact of these two challenges by inducing attention weights over the input sequence. This allows the system to prioritise the set of positions in which there is relevant information and then use this prioritisation in the generation of the next output token. The way in which an attention mechanism is incorporated into the encoder-decoder architecture is shown in Figure 2.10(b).

In order for the attention weights to be learned, an additional feed forward neural network is incorporated into the encoder-decoder architecture, known as the alignment function, which scores the input relevancy with regard to the output state. The output of this alignment function are energy scores that are then converted into attention weights by the distribution function which is, in most cases, the softmax function

(Bahdanau et al. 2015).

After the success of the attention mechanism in the sequence to sequence task, different types of attention mechanisms were developed. One of the most commonly used variation is the so-called transformer model (Vaswani et al. 2017) that captures global dependencies between input and output using a self-attention mechanism. This architecture has been shown to be capable of significant parallel processing, requiring shorter training times and delivering greater levels of machine translation accuracy without any recurrent components

In essence, the transformer architecture consists of a stack of six identical layers of encoders and decoders and two sub-layers, namely the feed forward network layer and the multi-head self-attention layer. According to Vaswani et al. (2017), the 'multi-head' attention refers to the fact that the self-attention layer is actually several attention layers stacked in parallel, allowing for the capture of more aspects of the input and enabling the model to be more expressive.

Self Attention in Speaker Recognition

Outside of the sequence-to-sequence task, the attention mechanism can also be used in speaker recognition. One of the first attention mechanisms to be used in speaker recognition was the attentive X-vector model (Wang, Okabe, Lee, Yamamoto & Koshinaka 2018, Zhu et al. 2018) which uses a self-attention layer between the frame-level feature extractor and the segment-level feature extractor within the X-vector architecture (Snyder et al. 2018). It can highlight the important part of the input speech signal and generate an attention weight vector that contains different score numbers for each part of the input sequence. It is then multiplied by the original input sequence and the output is referred to as the refined sequence or feature vectors.

Figure 2.11 shows a diagram of the self-attention mechanism that is used in the attentive X-vector. As shown in Section 2.2.4, the output of the frame-level feature extractor is $\boldsymbol{H} \in \mathcal{R}^{L \times E} = \{\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_L\}$, $\boldsymbol{h}_t \in \mathcal{R}^{1 \times E}$ denotes one feature vector in

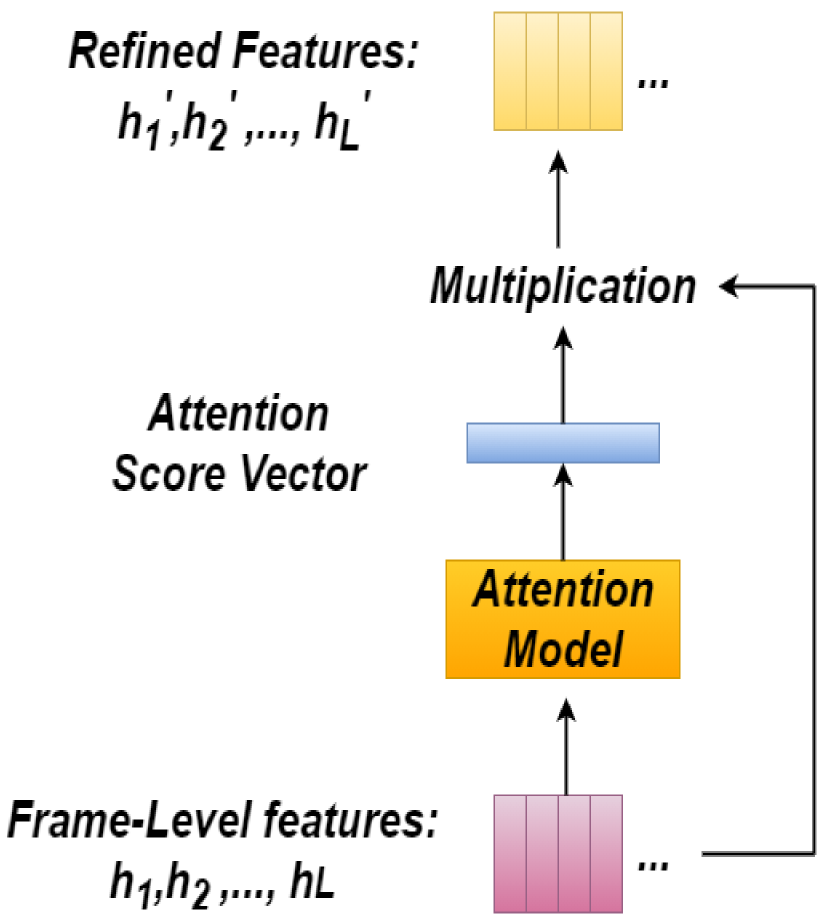


Figure 2.11: *The architecture of the attentive X-vector, from Zhu et al. (2018) and Wang, Okabe, Lee, Yamamoto & Koshinaka (2018).*

time t, L denotes the length of the sequence $\boldsymbol{H}$, E denotes the dimension of each feature vector of $\boldsymbol{H}$. Equation 2.10 shows the computation process of the attention mechanism.

$$\begin{aligned} c_t =& \mathrm{ReLU}(\boldsymbol{h}_t \boldsymbol{W} + \boldsymbol{b})\boldsymbol{v} \\ \alpha_t =& \frac{\exp(c_t)}{\sum_{i=0}^{L} \exp(c_i)} \end{aligned} \tag{2.10}$$

Equation 2.10 computes the score c_t for each feature vector using a feed forward network with $\boldsymbol{W} \in \mathcal{R}^{E\times L}$, $\boldsymbol{b} \in \mathcal{R}^{1\times L}$ and $\boldsymbol{v} \in \mathcal{R}^{L\times 1}$ being its weights and bias. The scores ($\boldsymbol{c}$) are normalised by a softmax function and the result is $\boldsymbol{\alpha} = \{\alpha_1, \alpha_2, ..., \alpha_L\}$, where each element in $\boldsymbol{\alpha}$ represents the attention weight value (a scalar number) to the corresponding frame. For example, one feature vector (e.g. $\boldsymbol{h}_t$) will be multiplied with the attention weight value for the corresponding time step (α_t), where $t \in \{1, 2, ..., L\}$

Another way to understand the use of the attention weights is the data broadcasting rule. The attention weight vector $\boldsymbol{\alpha}$ contains the dimensionality of $L \times 1$, and the input feature map $\boldsymbol{H}$ contains the dimensionality of $L \times E$. In order to apply the attention weight vector to the feature map, the vector $\boldsymbol{\alpha}$ is firstly repeated into the same dimensionality with $\boldsymbol{H}$. Then, element wise multiplication is applied between the attention weights and $\boldsymbol{H}$, resulting in the refined feature map $\boldsymbol{H}^{'}$, this is also known as the data broadcasting step.

The above mentioned data broadcasting rule is represented as a function: Broad(.) further in this thesis. In order to apply the attention weights to each frame of $\boldsymbol{H} \in \mathcal{R}^{L\times E}$, it is necessary to repeat the elements of $\boldsymbol{\alpha} \in \mathcal{R}^{L\times 1}$ E times along the frequency dimension. Equation 2.11 shows the computational process of the data broadcasting process, where $\text{Broad}(\boldsymbol{\alpha})^{L\times E}$ denotes the broadcasting function with $\boldsymbol{\alpha}$ being its input, the target dimension is $L \times E$. $\boldsymbol{q} \in \mathcal{R}^{1\times E}$ is a one's vector, where all of the elements in $\boldsymbol{q}$ are ones, $\times$ denotes the outer product.

$$\text{Broad}(\boldsymbol{\alpha})^{L\times E} = \boldsymbol{\alpha} \times \boldsymbol{q} \tag{2.11}$$

This data broadcasting rule can be flexibly implemented for other dimensions. Suppose $\boldsymbol{\alpha} \in \mathcal{R}^{1\times 1\times C}$ denotes the attention weights for the output of a CNN layer $\boldsymbol{H} \in \mathcal{R}^{T\times F\times C}$. Equation 2.12 shows the computation process that repeats $\boldsymbol{\alpha}$ into the same dimension as $\boldsymbol{H}$, where $\boldsymbol{q}_1 \in \mathcal{R}^{T\times 1}$ and $\boldsymbol{q}_2 \in \mathcal{R}^{1\times C}$ are vectors that all of the element values are ones. $\text{Trans}(.)^{T\times F\times C}$ denotes the transpose function that transpose the matrix or tensor into the target dimension (e.g. $T \times F \times C$). More details can be found later in Chapter 4.

$$\text{Broad}(\boldsymbol{\alpha})^{T\times F\times C} = \text{Trans}(\text{Trans}(\boldsymbol{q}_1 \times \boldsymbol{\alpha})^{T\times C\times 1} \times \boldsymbol{q}_2)^{T\times F\times C} \tag{2.12}$$

According to Wang, Okabe, Lee, Yamamoto & Koshinaka (2018) and Zhu et al.

(2018), the attentive X-vector performs better than the X-vector, reaching a 16% improvement in EER on the NIST 2016 speaker recognition evaluations dataset. Following the success of the attentive X-vector, it became widely used in speaker recognition systems, while other versions of the attention were developed. rahman Chowdhury et al. (2018) used K-max pooling (Kalchbrenner et al. 2014) on the attention weight vector $\boldsymbol{\alpha}$. K-max pooling selects the largest attention weights and discards the less important frames on the learned attention weights of the attentive X-vector model, reducing computational cost and making the model focus on the parts most relevant to the target speaker identity. According to rahman Chowdhury et al. (2018), K-max pooling can slightly improve on the performance of the attentive X-vector in text dependent speaker verification tasks, and the selection of different K values allows the enhanced model to out-perform the original model.

In addition to the X-vector architecture, attention mechanisms can also be applied in the CNN-based models for speaker recognition. An et al. (2019) experimented with adding a self-attention mechanism into some popular CNN architectures, such as ResNet. The attention mechanism used in the ResNet architecture can achieve approximately a 2% improvement in speaker identification on the Voxceleb1 dataset compared with the original ResNet model. India Massana et al. (2019) used a multi-head self-attention mechanism in a CNN model. As discussed above, the multi-head attention mechanism computes the attention weights $\boldsymbol{\alpha}$ multiple times in parallel, each of them focusing on different parts of the input signal. The results show the attention mechanism used in the CNN model can achieve an 18% relative improvement over the original CNN model.

In the attentive X-vector architecture, the dimensionality of the attention weight vector $\boldsymbol{\alpha}$ is $L \times 1$, thus the attention values are allocated for the time dimension of $\boldsymbol{H} \in \mathcal{R}^{L\times E}$, which is also referred to as the time attention (Miao et al. 2019). It can also be applied in the frequency dimension, where the attention model computes the attention weights $\boldsymbol{\alpha} \in \mathcal{R}^{1\times E}$ for each element of the frequency dimension of $\boldsymbol{H}$. According to Yadav & Rai (2020), both the time and frequency dimensions of the speech signal contains important information for speaker recognition, and the use of the combination of frequency and time attentions can achieve more than 3% relative improvement than the time attention in the ResNet model. This work was inspired by the convolutional block attention module (CBAM) in image classification, developed by (Woo et al. 2018), where the attention weights were computed on the time, frequency and channel dimensions of the input feature, thus important features in all three dimensions can be highlighted by the model.

However, the attention mechanism described above has two potential problems. Firstly, the self-attention model computes the attention weights for each frame of the whole sequence, which is the so-called global attention. Longer sequences of, say, three seconds, may contain multiple relevant features to the target speaker and, since the softmax function based global attention can only capture some of the important features, the model is likely to lose some significant information. This is due to the fact that global attention computes the importance weight for each frame in the whole sequence. The softmax function requires the attention weights sum to one (discussed in Section 2.2.2) and, as the sequence becomes longer, the importance of each frame is diluted (Wang, Okabe, Lee, Yamamoto & Koshinaka 2018, Okabe et al. 2018). For example, when there are two important parts in the sequence that the model needs to focus on, one of the parts is highlighted by the attention mechanism and a high weight value is assigned (e.g. larger than 0.5). The remainder of the sequence can only share the remainder of the weighting (e.g. less than 0.5), so the second important part will be incorrectly weighted, and may lead to an incorrect decision by the model.

The second problem is that the global self-attention pays insufficient attention to local features due to the computation process discussed above, and this phenomenon was also discussed by Wu et al. (2018). This can affect the noise robustness of the model. In noisy conditions, as illustrated in Figure 2.9 and discussed in Section 2.3.1, different types of noises (including fluctuating and steady noise) can affect the speech signal locally (Le Prell & Clavier 2017, Ming et al. 2007).

These issues are dealt with in Chapter 3 with a novel neural network architecture, based on a hierarchical attention mechanism.

2.3.3 Speech Enhancement

Another potential solution for the noise corruption problem in speaker recognition is to employ the speech enhancement method. In order to reduce the influences of the noises in the input signal and to obtain better recognition performance, speech enhancement is a natural choice (Ortega-García & González-Rodríguez 1996). Speech enhancement can be viewed as an independent preprocessing module that filters out noise information, create a clean signal that is then fed into the speaker recognition model. Approaches to noise reduction can be divided into two categories; methods that are mask-based, methods that are mapping-based (Bai & Zhang 2021).

The objective of the masking-based methods is to generate a time-frequency mask which contains a weight for each time-frequency (T-F) unit. The weights in the T-F mask reduces the noise impact by allocating a different weight for each T-F unit, with those corrupted by interference being allocated a lower weight than the clean units (Zhao et al. 2014). This is similar to the procedure of the attention mechanism discussed above, the main difference being that the attention mechanism is usually built into a neural network architecture to help the network better filter out the irrelevant information, while speech enhancement is usually used independently as a front-end module ahead of the speaker recognition module.

Zhao et al. (2014) firstly used a DNN-based speech enhancement model to generate a time-frequency ratio mask, which was then used to reduce the noise in the input signal. The clean output signal is then used in a speaker identification task. The 2008 NIST Speaker Recognition Evaluation dataset was used and a large number of noise types were tested. The speaker identification results show the DNN-based speech enhancement module can improve the noise robustness of the speaker recognition model in a wide range of noise types and reached 84.8% speaker identification accuracy. Later on, Kolbœk et al. (2016) applied long short term memory (LSTM) to the speech enhancement module as it can better capture the temporal information, thereby resulting in better speaker recognition performance. Shon et al. (2019) went a step further, integrating a speech enhancement module and a speaker recognition module into a system they called VoiceID Loss, in which the speaker recognition module is pre-trained and fixed when optimising the joint system. Unlike other approaches, the objective of the speech enhancement module is to improve speaker identification accuracy, rather than predict a clean signal. The results show that the quality of the enhanced signal was poor but the performance of the speaker recognition module was improved because the speech enhancement module not only reduced noise interference, but also discarded some irrelevant features for speaker recognition. Under different noise conditions, the VoiceID Loss method performs better in speaker verification than when the speech enhancement module is not used, reached 6.79% EER in the Voxceleb1 test set.

Rather than generating a time-frequency mask, mapping-based methods directly learn the mapping from the noisy signal to the clean signal, filtering out the noise information and retaining the useful information by reconstructing the clean signal. Most mapping-based methods are based on denoising auto encoder architectures (Lu et al. 2013), which have been shown to have good noise reduction abilities through the compression and reconstruction of the input signal. In the compression process, the model only remembers the key features of the input signal and the noise information is reduced.

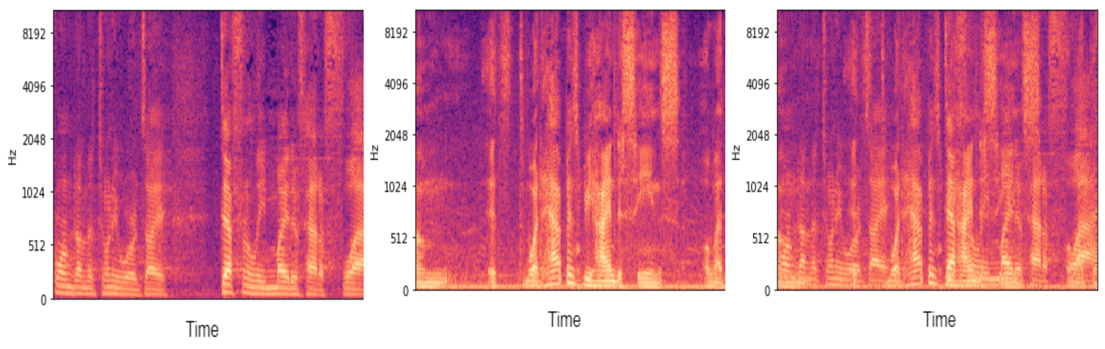
Plchot et al. (2016) proposed a DNN-based denoising auto-encoder architecture for reducing noise influence in the input speech signal. The model was trained in a supervised manner that reconstructed the clean signal from the noisy input in such a way as to minimise the error between the generated signal and the clean signal. The experimental results show the use of the speech enhancement module can deliver relative improvements of up to 50% for the text-dependent system and up to 48% for the text-independent one in the Fisher database. Pandey & Wang (2019) proposed adding skip connects into the auto-encoder architecture to obtain a better quality of output signal. The skip connection enabled the model to better retain relevant features when predicting the clean signal, and the results show that the quality of the signal was improved as compared with the baselines under various noise conditions. The speech quality of the output of the speech enhancement model was shown to be better than that from the original auto-encoder on the TIMIT dataset.

The MisMatch Problem

In the literature, the speech enhancement model is mainly used independently, and the enhanced output signal is then used for the back-end speech recognition and speaker recognition systems. In this way, the noises are filtered by the speech enhancement model, benefiting the back-end system and improving noise robustness. However, this does create a mismatch problem between the speech enhancement model and the back-end model. Sadjadi & Hansen (2010), Wang & Wang (2016) and Shon et al. (2019) indicated that this problem was caused by the speech enhancement module distorting some of the useful features and effectively creating new interference which, in turn, negatively impacts the back-end model.

The reason for this is the difference between the training targets of the two models. For example, in the mapping-based speech enhancement model, the objective of the speech enhancement model is to learn the mapping from the noisy speech to the clean signal. The objective functions are mainly based on the reconstruction of the loss function. A speech enhancement system often fails to make a good distinction between a useful feature and noise interference in a reconstructed signal because there is no constraint in the loss function to guarantee the speech enhancement model will retain the features useful to the back-end model, as the model is trained independently.

In Chapter 4, this problem will be further discussed and new models will be proposed to address this problem, specifically a joint training framework and new speech enhancement model architectures.



(a) Target speaker (b) Interfering speaker (c) Mixed speech

Figure 2.12: *Spectrograms of (a) Target speaker, (b) Interfering speaker and (c) Mixed speech.*

2.4 Speaker Recognition In a Multi-Speaker Environment

In some real world scenarios such as conversation, the interference not only comes from background noise, but also from other speakers. These overlapped interfering speakers are challenging to overcome as the features and patterns of the interfering speakers are similar to the target speaker, which makes it difficult for the speaker recognition system to separate speakers and recognise the target speaker (Wang & Chen 2018). This section will discuss the influence of interfering speaker in speech signals, and the potential methods to solve the problem, including target speaker extraction and weakly supervised speaker identification.

2.4.1 Interfering Speaker

In this thesis, the term "interfering speaker" refers to the speaker that occurs and temporally overlaps in the same recording as the target speaker. The target speaker is the speaker that the speaker recognition system needs to recognise. The voices from the interfering speaker can be viewed as the competing speech. It is when listeners are unable to distinguish the elements of the target speech from the similar-sounding

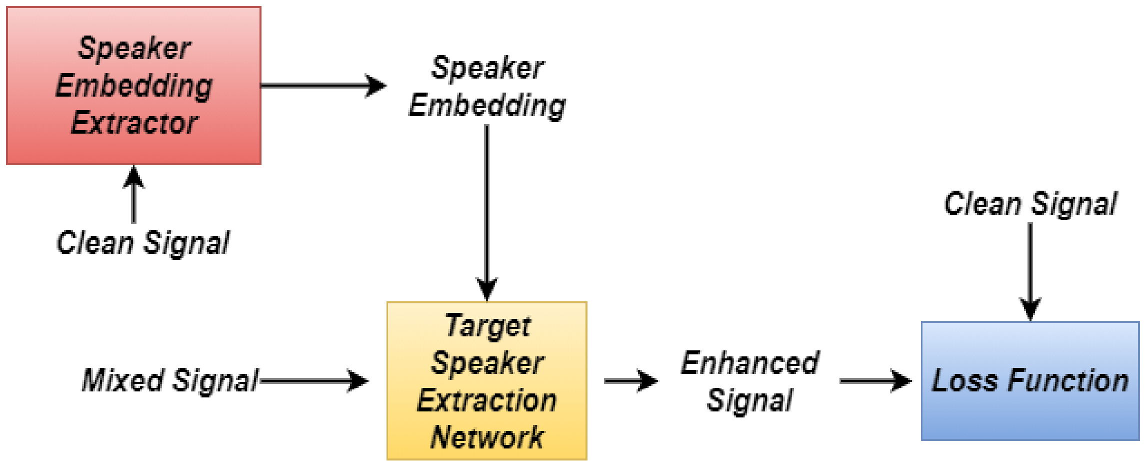


Figure 2.13: *The workflow of the target speaker extraction, from Wang et al. (2019).*

distracter (Le Prell & Clavier 2017). A typical problem in this field is the "cocktail party" problem, where multiple speakers are talking at the same time and the listener cannot distinguish between them (Haykin & Chen 2005).

Figure 2.12 shows an example of the influence of interfering speaker. The three spectrograms are, from left to right, from the target speaker, the interfering speaker and the mixed speech. It is obvious that from the mixed speech, it is difficult to distinguish between the target speaker and the interfering speaker.

For the purposes of this thesis, the experiments in Chapters 3 and 4 evaluate the performance of the proposed model with various background noises, including the babble type. In Chapters 5 and 6, different ways are proposed to address the noise caused specifically by interfering speaker.

2.4.2 Target Speaker Extraction

Target speaker extraction, also known as speaker dependent speech separation, is a methodology designed to overcome the interfering speakers by isolating the voice of the target speaker and treating the voices of all the other speakers as background noise. After removing the influences from other speakers, the extracted signal of the target speaker can then be used for speaker recognition (Zhao et al. 2019). Figure 2.13 shows the workflow of the target speaker extraction method in which the embedding of the target speaker is trained using a deep neural network known as a speaker embedding extractor, which takes the clean reference audio for the target speaker to obtain the corresponding speaker embedding. Then, the target speaker extraction network takes the mixed signal and the pre-trained speaker embedding as input, and outputs the enhanced signal that contains the voice of the target speaker only. The network is trained to minimise the error between the reference clean signal and the enhanced signal Wang et al. (2019).

In recent years, one of the most popular methods for target speaker extraction is the "Voice Filter" method proposed by Wang et al. (2019) who based their speaker embedding network on the d-vector structure (Variani et al. 2014), where the DNN used in the d-vector model is replaced by a three-layer LSTM to better capture the sequence information. Then, the Voice Filter network takes the learned d-vector for the target speaker and the mixed speech signal as the input. A soft mask is generated and multiplied with the original noisy speech signal. The Voice Filter method was firstly used for speech recognition and reached a better performance in noisy and multi-speaker environments. Later on, Rao et al. (2019) adapted this method for speaker recognition, using the enhanced speech signal in a speaker verification task. The experimental results show that the target speaker extraction method can improve the quality of the signal, while speaker verification is improved, reaching a 65.7% EER reduction over not using target speaker extraction.

In the above discussion and citations, target speaker extraction extracts the voice of the target speaker from the mixed signal, and that is then used as input to the back-end speaker recognition module. In order to reduce the impact of interfering speaker in the speech signal and improve speaker recognition performance, there may be another option that separates the embedding of the target speaker rather than the signal. In this approach, the enhanced signal in Figure 2.13 can be replaced by the embeddings that contain the information of the target speaker. In other words, the target speaker extraction network does not need to output the signal of the target speaker; instead, the embedding of the target speaker is the output of the target speaker extraction module. This approach simplifies the network architecture as it is only separating the target speaker embedding without having to construct it in the signal space. The other advantage is picked up by Bai & Zhang (2021) who point out that, since the embeddings are low-dimensional and fixed in length, the back-end speaker recognition module is easier to train.

This can be called the embedding de-mixing approach and is one that does not appear in any of the literature to date. In Chapter 5, the first approach that separates the speaker information in the embedding space is proposed in order to achieve the above goal.

2.4.3 Weakly Supervised Speaker Recognition

Although the target speaker extraction method can isolate the voice of the target speaker and improve speaker recognition performance, it may not work in some scenarios. According to Karu & Alumäe (2018), there is a specific situation when the input signal contains an unknown number of speakers, such as meetings or conversations, the system needs to recognise all of the speaker identities in the recording, instead of separating the voice of the target speaker. This is a specific task where there is no target speaker in the multi-speaker signal, and the goal is to recognise all of the speaker identities that occur in the input utterance.

In order to achieve this, the speaker diarisation method is one option (Anguera et al. 2012). The speaker diarisation method can segment a multi-speaker signal into several segments, based on the speaker identities, in order to answer the question "who speaks when?" (Anguera et al. 2012, Bai & Zhang 2021). However, in the situation described above, the only requirement is to answer the question "who speaks?"; the question of "when" is not required.

As indicated by Anguera et al. (2012), the supervised training of a speaker diarisa-

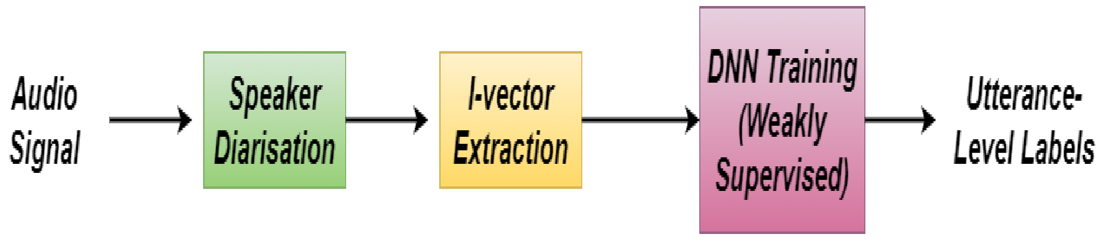


Figure 2.14: *The diagram of the weakly supervised training of the DNN, from Karu & Alumäe (2018).*

tion system requires the label of both speaker identities (who) and positions (when). Obtaining these labels is time consuming and expensive as it requires manually annotated identities and positions of each speaker. Karu & Alumäe (2018) indicated that weakly supervised learning is another option to achieve the goal and this does not require the positional labels. The field of weakly supervised learning was reviewed by Zhou (2018) and Yuan & Zhang (2010), who demonstrated that there is a scenario they called inexact supervision, in which the training data contains only coarse grained labels. Specifically, the labels for the multi-speaker signal in the training set contain only the set of speaker identities, and no positional information is provided, providing so-called coarse grained labels or utterance-level labels. In this scenario, the model can be trained to directly use the utterance-level labels to recognise the speakers who occurred in the input utterance (see Section 6.2 for more details), this is referred to as the weakly supervised speaker identification task.

The study by Karu & Alumäe (2018) is the only one that proposes a weakly supervised speaker identification approach, which is shown in Figure 2.14, above. It uses a pre-trained speaker diarisation system to find unique speakers in each recording, and a pre-trained i-vectors extractor to project the speech of each speaker to a fixed-dimensional vector. Both the speaker diarisation system and the i-vector system were pre-trained using background data. The speaker diarisation system segments the input signal into several short segments. The segments that belongs to the same speaker are clustered together, but speaker identities are not provided.

A DNN is then trained to map i-vectors to the score for each of the speakers in a multi-speaker signal. The output of the weakly supervised training of the DNN is the similarity score (between 0 and 1) for each speaker in the training set. The score shows the likelihood of a particular speaker occurring in the input utterance. The experiments were conducted on the augmented Voxceleb1 dataset and the results show the weakly supervised training of the DNN can reach 94.5% speaker identification accuracy, compared with the supervised CNN baseline (80.5%). The reason why the weakly supervised training can significantly outperform the supervised training baseline is that it does not require manually annotated labels, making it suitable for handing large quantities of data.

This approach of applying weakly supervised training to large amounts of coarse grained labelled data, proposed by Karu & Alumäe (2018) is not without problems. The speaker diarisation and i-vector systems need to be pre-trained, which also requires position labels for each of the speakers as input. In other words, only the training from the i-vector to the utterance-level labels are in the weakly supervised manner, and the remainder of the approach illustrated in Figure 2.14 deploys traditional supervised training.

In the literature, an approach that can directly make use of the coarse grained labeled data for speaker recognition is as yet un-developed. Chapter 6 proposes the first end-to-end weakly supervised speaker identification system that can directly learn the mapping from the multi-speaker signal to the utterance-level labels, with two novel neural network architectures.

2.5 Datasets

This section introduces the datasets that will be used in the following chapters for speaker identification or verification.

Table 2.1: *Part of the state-of-the-art speaker verification results on the Voxceleb1 test set when using Voxceleb1 for training only.*

	EER %
Nagrani et al.(Nagrani et al. 2017)	10.2
Nagrani et al.(Nagrani et al. 2017)	7.8
Shon et al. (Shon et al. 2019)	6.79
Cai et al. (Cai et al. 2018).	4.46
Hajibabaei and Dai (Hajibabaei & Dai 2018)	4.40
Hajibabaei and Dai (Hajibabaei & Dai 2018)	4.30

2.5.1 Voxceleb1

The Voxceleb1 dataset is a large and widely used dataset that was first proposed by Nagrani et al. (2017) for speaker identification and verification. It contains short clips of human speech, extracted from interview videos uploaded to YouTube. In order to extract the audio signals from the videos, a list of speaker identities were collected in advance, each of which is referred to as the person of interest (POI). Then, face tracking was used to segment the video segments for each of the POIs, and the audio signal was extracted from the segmented videos. According to the authors, the utterances in the dataset contain a wide range of noises, thus this dataset can be used for evaluating the noise robustness of the models.

The Voxceleb1 dataset contains 1,251 POIs with more than 150,000 utterances and 350 hours of speech. The dataset was split into two scenarios, one for speaker identification and the other for verification. In the speaker identification scenario, both training and test sets contain the 1,251 speakers, with 145,265 utterances in the training set and 8,251 utterances in the test set. In the speaker verification scenario, 1,211 speakers with 148,642 utterances were selected for the training set, and the remaining 40 speakers with 4,874 utterances were allocated into the test set. The test set was organised into 37,720 pairs for speaker verification.

The Voxceleb1 dataset has become the benchmark dataset for speaker identification and verification. The authors of the Voxceleb1 dataset (Nagrani et al. 2017) used a CNN based method, named VGG-M, which reached 80.5% identification accuracy and 7.8% EER in speaker verification. These results were then used as popular baselines for many subsequent studies. For example, Shon et al. (2019) proposed a speech enhancement and speaker verification joint system on Voxceleb1 and reached 6.79% EER; Cai et al. (2018) used a ResNet-34 model to conduct speaker verification and reached a EER of 4.46%; and Hajibabaei & Dai (2018) ran a ResNet-20 model and reached 4.30% EER. With regard to speaker identification, Ding et al. (2020) proposed a CNN based architecture and reached 87.66% identification accuracy, and Hajibabaei & Dai (2018) used the ResNet-20 architecture that reached 89.7% identification accuracy. Table 2.1 lists parts of the the state-of-the-art speaker verification results on the Voxceleb1 test set when using Voxceleb1 for training only. More details can be found in Section 3.5.3.

Based on the above baselines, the following chapter will develop new neural network architectures, and the proposed models will conduct both speaker identification and verification tasks on the Voxceleb1 dataset, and their performances with be compared to the baselines listed above.

2.5.2 MUSAN

The MUSAN dataset from Snyder et al. (2015) contains three categories of signals from real-world recordings; general noise, music and speech signals. There are 6 hours of general noise that includes technical noises such as DTMF tones, dial tones, fax machine noises, and ambient sounds such as wind, footsteps, paper rustling, rain, animal noises and so on. The music category contains 42 hours of music from popular Western genres. The speech signals contain a total of 60 hours of speech, including 20 hours of read speech and 40 hours of recordings from US government hearings, committees and debates.

Data augmentation technique has been widely used in speaker recognition in order to increase the noise robustness of the model (Wang et al. 2020, Snyder et al. 2018).

The MUSAN dataset contains a wide range of noise types and a large number of data collections, covering all of the noise types discussed in Section 2.3.1. It has been widely used for evaluating the noise robustness of speaker recognition models. Shon et al. (2019) comprehensively evaluated the noise robustness of a CNN-based speaker verification model under different noise types and levels, using utterances from the Voxceleb1 dataset mixed with the noise samples from the MUSAN dataset at different levels. The model reached 6.79% EER on the original Voxceleb1 test set, falling to 16.56%, 16.24% and 37.96% EER under the noise, music and babble noise types at 0dB respectively. Due to the comprehensive experimental setup in the study of Shon et al. (2019), the experiments in this thesis to evaluate the noise robustness of the proposed model all follow the settings from their study.

2.5.3 SRE08, SWBC and CHE

The 2008 NIST Speaker Recognition Evaluation Training Set Part 1 (SRE08) (Martin & Greenberg 2009) was developed by the Linguistic Data Consortium (LDC) and the National Institute of Standards and Technology (NIST). It contains multilingual telephone speech and English interview speech, which contains 640 hours of speech from 1,336 individual speakers. The interview speech signals appear in segments of approximately 3 minutes that have been lifted from longer conversations.

The Switchboard Cellular Part 1 (SWBC) dataset contains a total of 109 hour-long phone calls with 254 speakers, of whom 129 are male and 125 are female (David Graff 2001). Unlike the SRE08, this dataset was recorded under various environmental conditions, specifically indoors, outdoors and in moving vehicles. CALLHOME American English Speech (CHE) (Alexandra Canavan 2001) contains 120 telephone conversations between a total of 120 native English speakers, of which 90 of the calls are to various locations outside of North America.

The properties of these three datasets are explained in details in Section 3.4.1.

2.5.4 TIMIT and MC-WSJ

The TIMIT corpus of read speech (Garofolo et al. 1993) is designed to provide clean speech data for acoustic-phonetic studies and for the development and evaluation of automatic speech recognition systems or speaker recognition systems. There are a total of 6300 utterances, each consisting of 10 sentences spoken by each of the 630 speakers from 8 major dialect regions across the United States.

The second dataset is the Multi-Channel Wall Street Journal Audio Visual corpus (MC-WSJ) collected by Lincoln et al. (2005). It is used to evaluate the proposed system under real-world conditions. This dataset contains a total of 40 speakers reading WSJ sentences in three scenarios; single speaker stationary, where a single speaker reads sentences from six positions in a meeting room; single speaker moving, where a single speaker moves between the six positions while reading sentences; and overlapping speakers, where two speakers are reading sentences from different positions at the same time. The recordings from this latter scenario are used in this setup as it is the only one offering simultaneous speakers. It contains 10 unique speakers in 9 pairs, where each pair contains an average of 700 utterances. There is no speaker overlap between the three scenarios. There are three different recording sources: two microphone arrays, lapel and headset microphones worn by all of the speakers.

These two datasets are used for evaluating the performance of the model under multi-speaker environment, details can be found in Section 5.3.2.

2.5.5 Speech Command Dataset

Speech Command dataset (Warden 2018) contains isolated spoken English words. Each utterance contains one isolated word, and the duration of each is one second. There are 35 unique words, spoken by more than 2,000 different people. For each isolated word, there are more than 1,000 utterances on average. The details for this dataset can be found in Section 5.4.2 and Appendix B.3.

2.6 Baseline System

Previous sections discussed how deep neural network models can be affected by background noise. In this section, a baseline system is designed to show the influence of background noise from experimental results.

2.6.1 Experiment Setup

The baseline system makes use of the X-vector model (Snyder et al. 2018) (discussed in Section 2.2.4) because it was developed relatively recently and is widely used. This model architecture is also used as the baseline model in future experiments in Chapters 3 and 6.

As discussed in Section 2.5.2, in order to comprehensively observe the influence of the noise, both speaker identification and verification experiments were conducted using the Voxceleb1 dataset and, for both experiments, the training set is augmented by mixing Voxceleb1 data with noise signals at random chosen SNR levels (0, 5, 10, 15 and 20dB, randomly chosen from the 5 SNR levels). In this setting, each training utterance will be mixed with one noise utterance at one of the five SNR levels. The test utterances are mixed with a certain kind of noise at one of the five SNR levels, thereby delivering clearer results. This experimental setup will be used for the future experiments in Chapter 3 and Chapter 4, and more details can be found in Appendix B.1.

2.6.2 Results and Discussion

Table 2.2 shows the results obtained. The X-vector model can reach 88.2% speaker identification accuracy and a 5.47% equal error rate in speaker verification under clean conditions.

However, in the condition of the general noise type, the performances for both speaker identification and speaker verification is reduced, reaching 74.6% accuracy and

Table 2.2: *Speaker identification accuracy (%) and verification EER (%) for different noise types (Noise, Music and Babble) at different SNR (0-20 dB), and the original Vox-celeb1 test set using the X-vector model.*

Noise Type	**SNR**	**X-vectors**	
		Top1	EER
General	0	74.6	12.26
	5	79.5	10.01
	10	83.1	8.33
	15	85.0	7.25
	20	87.9	6.91
Music	0	68.2	14.15
	5	72.0	11.03
	10	79.4	9.35
	15	84.2	8.41
	20	86.1	6.79
Babble	0	64.1	30.02
	5	70.5	16.46
	10	77.4	13.26
	15	83.5	9.10
	20	86.6	7.95
Original		88.2	5.47

12.26% EER when the SNR is 0dB. These results show that noise can significantly influence the performance of the baseline model in both speaker identification and speaker verification. With music as interference, the model behaves similarly and delivering even worse results, with accuracy of 68.2% and an ERR of 14.15%. The worst results can be observed from the interfering speaker experiments. When the SNR is 0dB, speaker identification accuracy is only 64.1% and the ERR is 30.02%. In each of the SNR levels, the interfering speaker shows the worst results.

Compared with the work of Ming et al. (2007) and Shon et al. (2019), a similar phenomenon can be observed. Firstly, the deep neural network model can be highly influenced by noise interference, where the higher the noise level, the worse the performance, and the worst performance comes from the experiments with babble-type interference. The reason for this is that the interfering speakers compete for a similar frequency band as the target speaker so that most of the features of the target speaker are corrupted, meaning that the neural network has a limited amount of clean information on the target speaker (see Section 2.4.1). With other noise types such as general noise and music, even when the features of the target speaker are corrupted by both fluctuating and steady noise, there are still enough clean features for the network to capture and use for recognition (Le Prell & Clavier 2017).

2.7 Summary

At the beginning of this chapter, the concepts of speaker identification and speaker verification were introduced. Speaker identification is a closed-set task while speaker verification is an open-set task. Deep neural networks now deliver state-of-the-art performance in speaker recognition, compared to the conventional GMM-UBM based i-vector systems. Three types of neural network architectures were introduced and discussed in Section 2.1, including the feed-forward neural network (DNN), the convolutional neural network (CNN) and the recurrent neural network (RNN). Section 2.2 presents a review of the recent literature, including works on input features, loss functions and evaluation metrics. The neural network architectures (d-vector, X-vector and ResNet) for speaker recognition make use of the advantages of different neural networks and reached the state-of-the-art method in this domain.

Although deep neural networks have significantly improved the performance of speaker recognition, they are still affected by noise interference. In Section 2.3, the background noise were firstly discussed, and the influence of noise interference on the speech signal was shown in Figure 2.3.1. Then, to overcome this problem, two potential solutions was discussed, which are the attention mechanism and speech enhancement. The attention mechanism can be used in noise reduction because it can highlight the important features of the input signal and discard the corrupted ones. This can be viewed as a simulation of the procedure of designing a noise filter. Another method considered was speech enhancement, which filters out the noise information and outputs a clean speech signal. The key difference between these two methods is that the attention mechanism that was usually used within a neural network architecture to help the model select relevant features, whereas speech enhancement was an independent module deployed ahead of the speaker recognition system.

Interfering speaker was also considered. This presents a new challenge in that other speakers can compete in the same frequency band as the target speaker, making it difficult for the model to extract features of the target speaker (see Section 2.4). Target speaker extraction was discussed as the potential solution. The target speaker extraction method isolates the voice of the target speaker and treats the interference from other speakers as background noise. This property can be used for the back-end speaker recognition system. There is another specific situation that when the input utterance contains unknown number of speakers, the goal is to recognize all of the speaker identities in one input utterance. This is a specific task and weakly supervised learning can be deployed, whereby a large amount of data is used with coarse grained labels.

Finally, datasets recorded were introduced in Section 2.5, specifically Voxceleb1 and MUSAN datasets. These datasets will be used to evaluate the performance of the proposed models in the following chapters. A baseline system was designed in Section 2.6 to provide the evidence on how the performance of the deep neural networks can be influenced by noise interference.

Chapter 3

Hierarchical Attention Network

3.1 Introduction

Section 2.3.2 introduced the attention mechanism, discussed various attention-based approaches to speaker recognition, and laid out the properties of the mechanism. Briefly, the attention mechanism used for the speaker recognition model (e.g. attentive X-vector (Wang, Okabe, Lee, Yamamoto & Koshinaka 2018) and ResNet (He et al. 2016)) was a global self-attention layer. It can highlight the most relevant part from the input signal to the target that can improve the performance of the model (rahman Chowdhury et al. 2018). This property allowed for noise reduction methods to be developed for both image and speech signals, whereby the "corrupted features" were allocated lower weights to ensure that the model focuses on the clean features. In this way, excess noise can be reduced and the robustness of the model can be improved (Hao et al. 2019).

However, as discussed in Section 2.3.2, the attention mechanism used in current speaker recognition models has two potential problems. Firstly, the self-attention mechanism computes the attention weights for the whole input sequence using the softmax function. When there are multiple important parts in the input sequence, some of them may be incorrectly weighted. The second problem is that, while noise interferences occur at a local scale, a global attention mechanism may be unable to capture sufficient local information.

In order to address the two problems discussed above, one possible solution is to develop a new neural network architecture that can capture both local and global features in one framework. The attention mechanism needs to be used in both local and global scenarios, and this is something that can be achieved through the use of hierarchical structures such as the document classification approach proposed by (Yang et al. 2016). In this approach, the network firstly uses multiple word level encoders, each one of which captures the local features between words in each sentence and the attention mechanism is used to assign weights for each word within each sentence. Each sentence is then summarised in a single sentence vector. At a higher level, the generated sentence vectors form the input to a sentence level encoder which focuses on the global information between each sentence. The attention mechanism was used to allocate weights between each of the sentence vectors. The sentence level encoder then compresses different sentence vectors to generate a document vector, which is then used for the final prediction.

The hierarchical structure introduced above splits the attention mechanism into two levels, where the local and global information can be captured by the word-level and sentence-level encoders respectively. This property can be applied to speaker recognition to address the problems mentioned above. The local and global features from the input utterance can be captured by the hierarchical structure, and it may also be able to avoid the problem that some parts of the input are incorrectly weighted by the global attention mechanism (discussed above). To achieve this, the utterance can be viewed as a document, the segments are sentences and the frames are viewed as the words. The hierarchical attention network splits the input signal into different segments. The frame-level encoder computes the attention weights between each frame within the segment. Then, the segment-level encoder measures the importance between each segment and generates the utterance vector for the final prediction of the speaker identities.

3.1.1 Chapter Outline

The remainder of this chapter is organised as follows:

- Section 3.2 introduces the hierarchical structure at a high level.
- Section 3.3 introduces the detailed model architecture of the hierarchical attention network.
- Sections 3.4 and 3.5 shows and discusses the results of experiments designed to evaluate the performance of the hierarchical attention network. Section 3.4 in particular focuses on the model generalisation comparison between the proposed hierarchical attention network and the baseline models.
- Section 3.5 shows the evaluation of the noise robustness of the proposed hierarchical attention network.
- Section 3.6 is the conclusion.

3.2 Hierarchical Structure

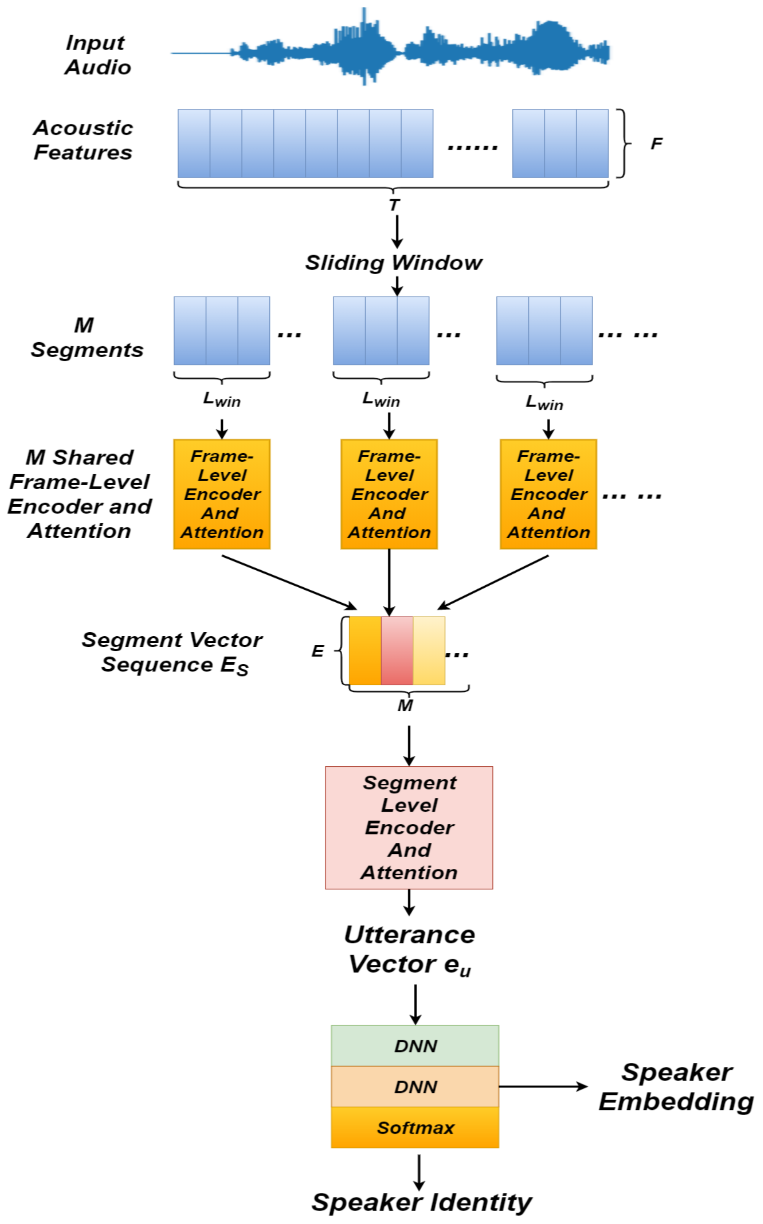

Input Audio
Acoustic Features
F
T
Sliding Window
M Segments
L_{win}
M Shared Frame-Level Encoder and Attention
Frame-Level Encoder And Attention
Segment Vector Sequence E_S
E
M
Segment Level Encoder And Attention
Utterance Vector e_u
DNN
DNN
Softmax
Speaker Embedding
Speaker Identity

Figure 3.1: *High level illustration of the hierarchical architecture.*

Figure 3.1 shows the high level architecture of the hierarchical structure. The proposed hierarchical attention network is based on this structure and will be introduced in the next section.

Firstly, the acoustic features are computed from the waveform signal input (MFCC features (Davis & Mermelstein 1980) are used in all of the experiments in this chapter). In order to apply the attention mechanism at the frame and segment levels, the input feature vector sequence is divided into several short segments using a sliding window. Specifically, the input sequence $\boldsymbol{X}$ is divided into M segments: $\{\mathbf{S}_1, \mathbf{S}_2, \cdots, \mathbf{S}_M\}$ using a sliding window with length L_{win} and step size L_{step}. Each segment $\mathbf{S}_m \in \mathcal{R}^{L_{win} \times F}$ contains L_{win} F-dimensional acoustic frame vectors.

The frame level encoder takes each of the short segments as input. It computes the attention weights for each frame within the segment, and compresses it into a single feature vector called a segment vector. There are M frame-level encoders, each of them compresses the corresponding segment sequence $\boldsymbol{S}_m$ into a segment vector $\boldsymbol{e}_{S_m}$. The output vectors of the M frame-level encoders are concatenated together into a new sequence called the segment vector sequence $\boldsymbol{E}_S = \{\boldsymbol{e}_{S_1}, \boldsymbol{e}_{S_2}, \cdots, \boldsymbol{e}_{S_M}\}$.

In the segment level encoder, each element of the segment vector sequence $\boldsymbol{E}_S$ contains the information of the corresponding segment. The segment level encoder and attention mechanism is applied to the segment vector sequence $\boldsymbol{E}_S$, and compresses it into a single vector $\boldsymbol{e}_u$ called the utterance vector.

The final speaker identity classifier is constructed using a two-layer MLP followed by a softmax activation function with $\boldsymbol{e}_u$ being its input. The final speaker identities are the output vector which contains the scores (between 1 and 0) for each speaker. The model is trained using cross entropy loss, as discussed in Section 2.2.2.

It is obvious that the frame level attention only concerns the relationship of the frames within a short segment, while the segment level attention pays attention to the speaker features between each of the segment vectors; in other words, it captures global features between each segment. It is assumed that this network architecture will be able to capture more speaker-relevant features, leading to greater accuracy in terms of speaker recognition. Furthermore, when dealing with noise interference in the input signal, it is assumed that the hierarchical structure will be more robust than the X-vector and attentive X-vector architectures.

3.3 Network Architecture

The previous section described the overall structure of the hierarchical structure; in this section, the details of the proposed hierarchical attention network will be introduced.

3.3.1 Frame-Level Encoder

The architecture of the frame-level encoder for the mth segment is shown in Figure 3.2, which takes a segment $\boldsymbol{S}_m$ as input. A TDNN layer (as described in Section 2.1.2) is firstly applied to the input sequence. Followed by the TDNN layer is a bi-directional GRU layer. As discussed in Section 2.1.3, a GRU can capture both long-term and short-term memory better than the standard RNN model, and it can also reduce the computational cost compared to LSTM. As the adjacent frames within one segment have a strong dependency over time, the speaker-related features (mainly local features at the frame level) are expected to be located in those frames. Using a GRU can help the model to better capture this information. In order to get information from both directions of acoustic frames and contextual information, the bi-directional version of GRU is also used (discussed in Section 2.1.3).

The output of the mth frame-level encoder is denoted as $\boldsymbol{H}_m \in \mathcal{R}^{L_{win} \times E}$, $\boldsymbol{H}_m = \{\boldsymbol{h}_{m,1}, \boldsymbol{h}_{m,2}, \cdots, \boldsymbol{h}_{m,L_{win}}\}$. In the attention layer, a frame-level attention mechanism takes $\boldsymbol{H}_m$ as input, the output is the attention score vector $\boldsymbol{\alpha}_m \in \mathcal{R}^{L_{win} \times 1}$. The scalar elements of the attention vector $\boldsymbol{\alpha}_m$ are used to scale the vector elements of $\boldsymbol{H}_m$. In implementation, the weight vector $\boldsymbol{\alpha}_m$ is repeated to the same dimension as $\boldsymbol{H}_m$ and element wise multiplication is used to generate the weighted output $\boldsymbol{A}_m$. The computation process of $\boldsymbol{\alpha}_m$ is the same as that used in the attentive X-vector, which can be found in Equation 2.10, Section 2.3.2.

A statistics pooling operation is applied on the weighted output sequence $\boldsymbol{A}_m$ to compute its mean vector ($\boldsymbol{\mu}_m \in \mathcal{R}^{1 \times E}$) and standard deviation ($\boldsymbol{\sigma}_m \in \mathcal{R}^{1 \times E}$) vector over time. A segment vector $\boldsymbol{e}_{S_m} \in \mathcal{R}^{1 \times 2E}$ is then obtained by concatenating the two

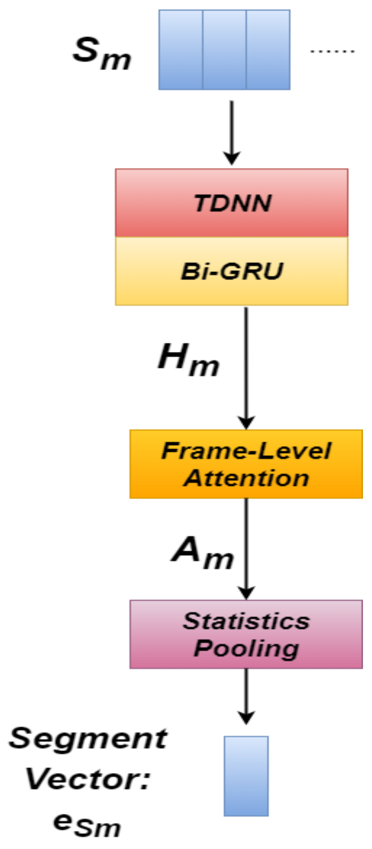


Figure 3.2: *The architecture of frame-level encoder with attention to the mth segment.*

vectors. The computation process of the statistics pooling operation is the same as that used in the X-vector model, which is introduced in Section 2.2.4.

3.3.2 Segment Level Encoder

For the segment-level encoder and attention, the same steps used in the frame-level encoder and attention are implemented except for the bi-directional GRU layer. It contains a TDNN layer, a segment-level attention layer and a statistics pooling operation. The bi-directional GRU is not used in the segment-level encoder; this has the dual advantage of a) reducing the computational cost, thereby b) accelerating the training when processing a large number of samples. Furthermore, the use of the GRU in the frame-level encoder can help the model to capture more contextually relevant features but, at the segment level, the co-relevance between segments may be relatively small in comparison with the contextual information between frames occurring within a segment. This is because the segment vectors come from the compression of the frames within each segment.

The output of the frame level encoder is $\boldsymbol{E}_S \in \mathcal{R}^{M \times E_S} = \{\boldsymbol{e}_{S_1}, \boldsymbol{e}_{S_2}, \cdots, \boldsymbol{e}_{S_M}\}$. After a TDNN layer (with $\boldsymbol{E}_S$ being its input), the weight vector $\boldsymbol{\alpha}_S \in \mathcal{R}^{M \times 1}$ of the segment level attention can be computed using the same method as that in the frame-level encoder. The statistics pooling operation is applied to the weighted output of the segment-level attention, the output is the utterance-level feature vector $\boldsymbol{e}_u$. The final speaker identity classifier is constructed using a two-layer MLP with $\boldsymbol{e}_u$ as its input. The output of the first fully connected layer can be used as the final utterance embedding.

Although the computational process of the attention mechanism at the segment level is the same as that at the frame level, they have different inputs. The segment-level attention mechanism measures the importance between each segment while the frame-level mechanism only focuses on the frames within each segment. Instead of only applying a global attention such as the attentive X-vector, the two levels of the attention mechanism may help the model to capture more relevant features and avoid any dilution of the features. The training details of the model can be found in the next section.

3.4 The Generalization of the Hierarchical Attention Network

In addition to the assessment of noise robustness, the assessment of the model generalisation is important. According to Bai & Zhang (2021) and Garcia-Romero & McCree (2014), the different properties of the training and test sets can cause the domain mismatch problem, which indicates the training and test sets contain different distributions. This becomes an issue when evaluating a model's ability to adapt properly to new, previously unseen data that is drawn from the same distribution as the one used to create the model (Goodfellow et al. 2016), otherwise known as generalisability. If the evaluation is done with a test set that does not match the training set, this is when a domain mismatch problem can arise.

In order to comprehensively test the performance of the proposed hierarchical attention network, the experiments are split into two parts: the first part will focus on the model generalisation of the hierarchical attention network compared with other strong baseline speaker recognition models that have been recently developed. The second part is to evaluate the noise robustness of the hierarchical attention network. These will be the focus of the following two sections.

3.4.1 Experimental Setup

Dataset

In this work, SRE08, SWBC and CHE datasets are used. The introduction of these three datasets can be found in Section 2.5.3.

The properties of the three datasets are shown in Table 3.1. Clearly, these three datasets have specific and distinct properties. The SRE08 dataset is a large, multilingual dataset containing both telephone and interview speech, whereas the SWBC and CHE are smaller and only offer telephone speech. In order to evaluate the model generalisation of the proposed model, the SRE08 can be used to train a neural network

Table 3.1: *The details of three speech datasets: Part1 of Sre2008 (SRE08), CallHome(CHE) and Switchboard(SWBC)*

Dataset	Type	#Speaker	Size (hour)	#Utterance (1s)	#Utterance (3s)
SRE08	Telephone+Interview	1336	640	3,528,326	1,176,453
CHE	Telephone	120	60	252,224	84,460
SWBC	Telephone	254	130	1,008,901	336,417

and both the SWBC and CHE datasets can be used to extract speaker embeddings from the trained model and evaluations.

Experimental Setup

In order to evaluate the generalisation of the proposed model, the models are firstly trained using the SRE08 dataset. Then, speaker embeddings are extracted from the trained model using SWBC and CHE datasets. Both speaker identification and verification experiments are conducted. The detailed experimental setup including model training details can be found in Appendix A.1.1, including the detailed data processing strategies.

Both the window size (L_{win}) and step size (L_{step}) of the proposed hierarchical attention network are fixed at 30 frames. This means there is no overlap between each segment. For the performance of different window and step sizes, further experiments are outlined in Section 3.5.

In order to evaluate the performance of the proposed model with short and long utterances, all the experiments described above are split into two scenarios, one using an utterance length of one second, the other using an utterance length of three seconds.

Baseline Selection

The purpose of the experiments in this section is to evaluate the performance of the proposed hierarchical attention network compared to current, widely used baselines.

In the experiments, two baselines were selected, namely X-vectors (Snyder et al. 2018) and attentive X-vectors (Wang, Okabe, Lee, Yamamoto & Koshinaka 2018, Zhu et al. 2018). The proposed model is henceforth referred to as H-vector.

X-Vectors (Snyder et al. 2018) is based on a TDNN architecture and is described in more detail in Section 2.2.4. This model architecture contains a five-layer TDNN-based frame-level feature extractor, with each layer operating on certain time steps. A statistics pooling operation is applied to the output of the frame-level feature extractor to summarise the output sequence into a vector. Then, a DNN-based segment-level feature extractor is used to generate the final speaker embedding.

The reason for selecting X-vector as one of the baselines is that it is widely used for speaker recognition and is effective in speaker embedding extraction. Snyder et al. (2018) demonstrated that the X-vector model has strong generalisation to different datasets. In their work, the X-vector model is trained using the SRE16 dataset, and Voxceleb1 data was used as one of the test sets. The results show the X-vector can obtain better results compared with an i-vector based system. Another reason to use X-vectors as one of the baselines is that it uses no attention mechanism in the architecture. The effectiveness of the attention mechanism used in the H-vector model can therefore be evaluated by comparing the performance of the X-vector architecture and the H-vector model.

The attentive X-vectors model, combines a global attention mechanism with X-vectors (Wang, Okabe, Lee, Yamamoto & Koshinaka 2018, Zhu et al. 2018), is the second baseline deployed in this study as it is one of the most widely used variations of the X-vector architecture, and is first introduced in Figure 2.11 in Section 2.3.2. In addition to the frame-level feature extractor, the statistics pooling operation and the segment-level feature extractor, the attentive X-vectors model uses a global attention mechanism on the output of the frame-level feature extractor before the statistics pooling operation. The attention mechanism used in attentive X-vectors directly computes weights for each frame, which is different from the proposed approach.

Selecting the attentive X-vector model as one of the baselines allows for a direct comparison between the proposed hierarchical attention mechanism and the global attention mechanism used in attentive X-vector. As discussed in Section 3.1, the potential problems of the global attention mechanism are addressed by the proposed hierarchical attention mechanism . Comparing the performance of the two models is an obvious way to show which attention mechanism performs best.

3.4.2 Results and Discussion

Table 3.2: *Speaker identification accuracy (%) on the SRE08 test set with the utterance length is 1s or 3s. L_{win} and L_{step} are set to 30 frames. Improve represents the relative improvement (%) of the speaker identification accuracy compared to the X-vector model in 1 or 3 second scenarios.*

Utterance Length	Model	Accuracy	Improve
1 Second	X-vector	90.1	0
	X-vector+Att	92.1	2.2
	H-vector	94.5	4.8
3 Seconds	X-vector	95.2	0
	X-vector+Att	96.7	1.5
	H-vector	98.5	3.4

Performance On SRE08

Table 3.2 shows the speaker identification accuracy on the SRE08 test set using the proposed approach and the two baselines. As the models are trained using the SRE08 dataset, the identification accuracy on its test set is firstly shown. Overall, the accuracy for all of the three models is over 90% in both the one-second and three-second

Table 3.3: *Speaker identification accuracy (%) and speaker verification EER (%) on CHE dataset when the utterance length is 1s or 3s. L_{win} and L_{step} are set to 30 frames. "Improve Acc" represents the relative improvement (%) of the speaker identification accuracy compared to the X-vector model in 1 or 3 second scenarios. "Improve EER" represents the relative improvement (%) of the speaker verification EER compared to the X-vector model in 1 or 3 second scenarios.*

Utterance Length	Model	Accuracy	EER	Improve Acc	Improve EER
1 Second	X-vector	84.8	1.86	0.0	0.00
	X-vector+Att	87.5	1.53	3.1	17.74
	H-vector	89.1	1.36	5.0	26.88
3 Seconds	X-vector	89.4	1.39	0.0	0.00
	X-vector+Att	91.0	1.18	1.7	15.10
	H-vector	92.8	1.01	3.8	27.33

scenarios, though accuracy with the three-second utterances is better than that for the one-second ones. This probably indicates that a longer utterance may contain more information relevant to a target speaker than a short one.

For the one-second utterances, the H-vector showed a relative improvement of 4.8% compared to the X-vector model, and 2.6% compared to the attentive X-vector model. For the three-second scenario, the relative improvements delivered by the H-vector model were 3.4% and 1.8% respectively.

These relative improvements may demonstrate that, with regard to X-vector model, the H-vector is superior as a result of its attention mechanism, which the X-vector lacks and, with regard to the attentive X-vector model, the H-vector performs better by deploying both local and global attention, as opposed to global attention only.

Table 3.4: *Speaker identification accuracy (%) and speaker verification EER (%) on SWBC dataset when the utterance length is 1s or 3s. L_{win} and L_{step} are set to 30 frames. "Improve Acc" represents the relative improvement (%) of the speaker identification accuracy compared to the X-vector model in 1 or 3 second scenarios. "Improve EER" represents the relative improvement (%) of the speaker verification EER compared to the X-vector model in 1 or 3 second scenarios.*

Utterance Length	Model	Accuracy	EER	Improve Acc	Improve EER
1 Second	X-vector	78.2	2.17	0.0	0.00
	X-vector+Att	81.0	2.02	3.5	6.90
	H-vector	83.7	1.90	7.0	12.44
3 Seconds	X-vector	81.3	1.98	0.0	0.00
	X-vector+Att	84.0	1.79	3.3	9.59
	H-vector	86.2	1.61	6.0	18.6

Generalisations On Out-of-Domain Datasets

Having shown the performance in the in-domain SRE08 dataset, Tables 3.3 and 3.4 show the identification accuracy and verification equal error rate when using the embeddings learned on the SWBC and the CHE dataset, respectively.

Overall, both of the identification accuracies in the following two tables are lower than that those in Table 3.2. This is because SWBC and CHE have different properties compared with SRE08

More specifically, for all three models, the accuracies on SWBC are lower than CHE, perhaps due to the wide range of environmental conditions (indoors, outdoors and moving vehicles) that may affect the identification rate. For the CHE dataset, the overall accuracy did not reach the training accuracy on SRE08. A possible reason is that CHE contains only voices from native English speakers, while SRE08 contains multilingual data, and so the speaker similarity on the CHE dataset might affect the identification rate.

Compared with the results obtained by X-vector and attentive X-vector, the results obtained by the H-vector show that a better generalisation. The H-vector consistently outperforms the two baselines, regardless of whether the length of utterances is 1 second or 3 seconds. In the CHE dataset, the H-vector approach reaches an 89.1% prediction accuracy and 1.44% equal error rate, with an improvement of over 3% in the speaker identification task as compared to the baselines. In the speaker verification task, the H-vector achieved relative improvement over 20% and 10% compared with the X-vector and attentive X-vector models respectively. Similar to the results with the SRE08 dataset, the results obtained with the three-second utterance length is better than the performance with the one-second utterances.

For both the in-domain and out-of-domain datasets, the H-vector performed better than both the X-vector and attentive X-vector baselines, thereby demonstrating that the proposed hierarchical attention mechanism has better model generalisation and, specifically, performs better than the global attention in the attentive X-vector model which, in turn, performs better than the X-vector model. This is due to the use of global attention that highlight important parts. The lack of attention mechanism in the X-vector architecture means it treats each frame as being equally important to the target speaker identities

3.4.3 Analysis

Visualisation of the DET Curve

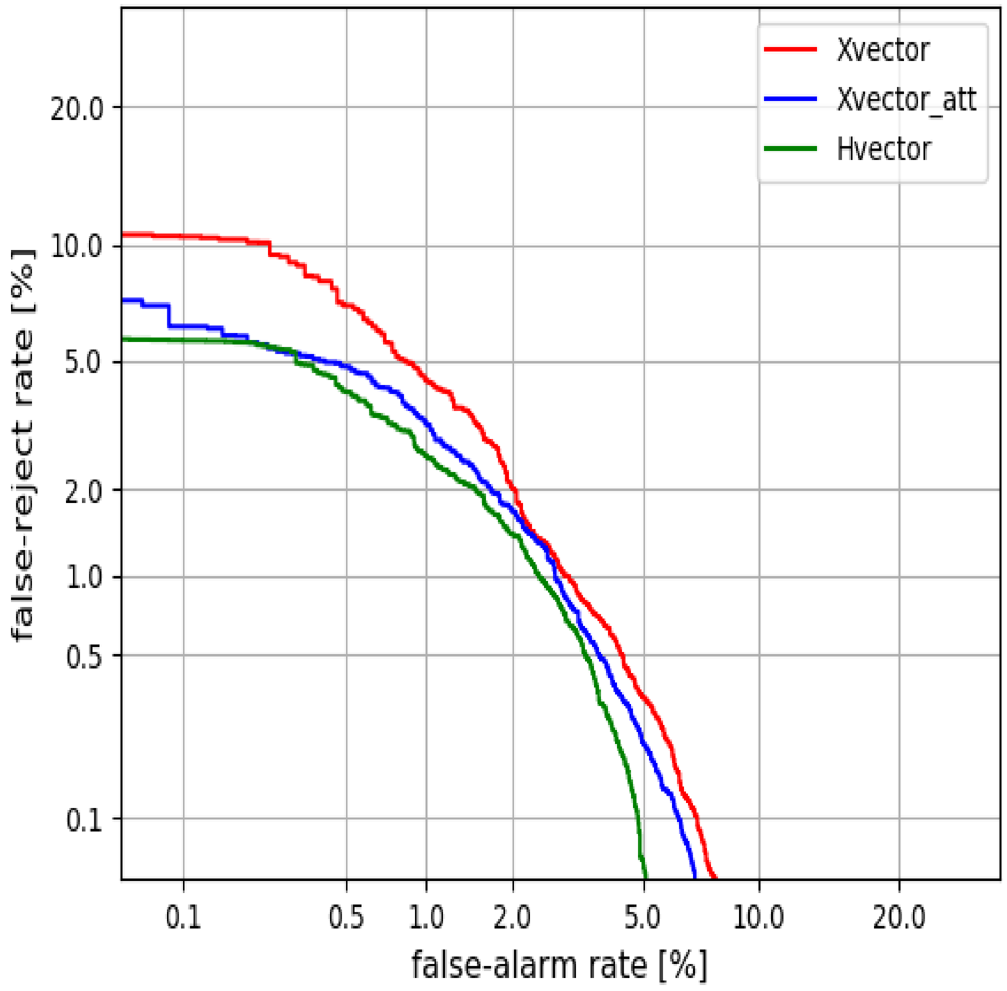


Figure 3.3: *The DET curve (introduced in Section 2.2.3) on the SWBC dataset when the utterance length is 3 seconds.*

In order to intuitively show and compare the proposed H-vector model and the two baselines, this section visualises the performance of the three models in two different ways.

Figure 3.3 shows the detection error trade-off (DET) curve (Martin et al. 1997, Garcia-Perera et al. 2012) that was introduced in Section 2.2.3. The figure shows the curves for all three models (X-vector, attentive X-vector and H-vector) on the SWBC dataset when the utterance length is 3 seconds. The DET curve plots the false alarm (FA) rate and false rejection (FR) rate into one graph to illustrate the model performance and the equal error rate (when the FR and FA rates are equal).

From Figure 3.3, it is clear that the H-vector model had a lower false rejection rate and a lower false alarm rate, as well as a lower equal error rate. The attentive X-vector obtained higher false rejection and false alarm rates, though these were still lower than those obtained by the X-vector model. This is mainly due to the use of attention mechanism; the attentive X-vector uses global attention that allocates different weights to each frame, which can highlight the importance of different frames. For example, when the false alarm rate is fixed at 1 %, the false reject rate of H-vector, attentive X-vector and X-vector are 2.9%, 3.5% and 4.7%. When the false reject rate is fixed at 1%, the false alarm rate of H-vector, attentive X-vector and X-vector are 2.6%, 3.1% and 3.4%.

Visualisation of Embeddings Using the t-SNE Algorithm

The next step was to visually evaluate further the quality of the extracted utterance-level embeddings. This was achieved by applying the t-SNE algorithm, developed by Maaten & Hinton (2008). This is a widely used visualisation technique, shows the distribution of the embeddings by projecting the high-dimensional vectors onto a 2D plane. The algorithm can project high dimensional data points onto a low dimension space. For example, the data points that are close to one another in high dimensional space can be projected into the low dimensional space and remain close. In this way, the distances between the speaker embeddings can be visualised, and the intra-class and inter-class distances can be plotted in a two dimensional plane.

In order to plot the data points for the SWBC dataset, 10 speakers were selected

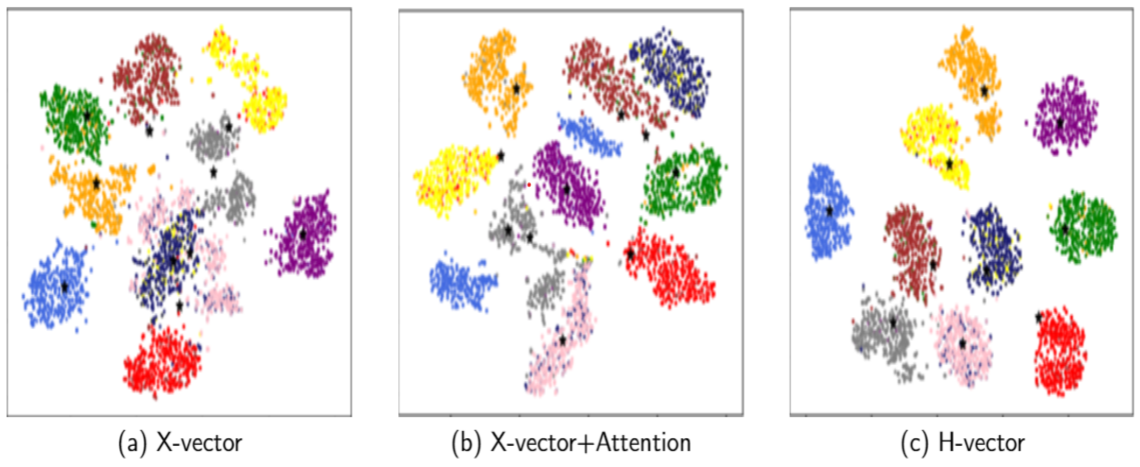


Figure 3.4: *Embedding visualization using t-SNE. In the SWBC dataset, 10 speakers are selected and 500 three-second segment are randomly sampled for each speaker. Each color represents a speaker, and each point indicates an utterance.*

and 500 three-second segment were randomly sampled for each speaker.

Figures 3.4 (a), (b) and (c) show the distribution of the selected samples of the 10 speakers after using the X-vector, the attentive X-vector, and the H-vector, respectively. Each colour represents a distinct speaker and each point represents an utterance. The black mark represents the centre point for each speaker class. Figure 3.4 (a) shows the distribution of the embeddings obtained by the X-vector model, and a certain lack of discrimination between speakers is obvious in that there are overlaps between speaker classes. Due to the use of an attention mechanism in attentive X-vector, Figure 3.4 (b) shows a better sample distribution than Figure 3.4 (a), though it clearly had issues with the speaker shown in blue as these data points are not well grouped. In Figure 3.4 (c), the embedding obtained by the H-vector model creates a better separation than either of the baseline models.

The visualisations in Figure 3.4 clearly shows that the H-vector model has the smallest intra-class distance and the largest inter-class distance demonstrating, once again, that the H-vector model is superior to either of the baseline models.

3.4.4 Summary

In order to evaluate the model generalisations, three datasets with different properties were selected, namely SRE08, SWBC and CHE. The two baseline models, the X-vector and the attentive X-vector models, were used to compare with the proposed hierarchical model. The X-vector was used to compare the effectiveness of the attention mechanism used in the proposed model, while the attentive X-vector was used to compare the global attention mechanism with the proposed hierarchical attention mechanism and, against each one, the H-vector model performed better and had a better level of generalisation. Specifically, the results in Table 3.2 shows the H-vector model performed best with the in-domain dataset, while Tables 3.3 and 3.4 show the proposed H-vector model can obtain a better model generalisation with the out-of-domain dataset.

In order to intuitively show the performance comparison, Figure 3.3 shows the DET plot, indicating that the H-vector model performed better than the two baseline models. Finally, Figure 3.4 visualises the sample of speaker embeddings in two dimensional space, and it shows that the proposed H-vector model can achieve both a lower intra-class distance and a larger inter-class distance than the two baselines.

The results obtained by the H-vector model and the attentive X-vector model are better than the X-vector model, this phenomenon shows the attention mechanism can highlight important parts of the input signal and improve the performances. While the comparison of the attentive X-vector model and the proposed H-vector model shows that the hierarchical attention can capture both local and global information by the hierarchical structure, which may lead to the better performances than the global attention mechanism.

3.5 Noise Robustness of the Hierarchical Attention Network

Following on from the evaluation of the model generalisation, this section will focus on the evaluation of the noise robustness of the H-vector model compared with the two baselines.

3.5.1 Experiment Setup

In order to evaluate the noise robustness of the proposed model, the Voxceleb1 dataset (introduced in Section 2.5.1) is used. The data augmentation process is the same as described and used in Section 2.6.1, details can be found in Appendix B.1. For both speaker identification and speaker verification tasks, the training utterances from the training sets are augmented by mixing Voxceleb1 data with random noise signals from the MUSAN dataset at random SNR levels (0, 5, 10, 15 and 20dB). The test utterances are mixed with a certain kind of noise at one of the five SNR levels. The ability to evaluate and compare particular values with specific noise types across the test dataset allows for a comprehensive evaluation of the noise robustness of the three models.

For the experiments described above, both the window size (L_{win}) and step size (L_{step}) of the proposed hierarchical attention network are fixed to 30 frames. There is no overlap between segments, and experiments to determine the influence of different window and step sizes were also conducted. The implementation details of the H-vector model for the experiments in this section are the same for the previous sections.

3.5.2 Results and Discussion

Performance On Voxceleb1

In this section, the goal is to evaluate the noise robustness of the proposed H-vector model. The Voxceleb1 dataset was recorded "in the wild" and therefore the original

Table 3.5: *Speaker identification accuracy (%) and speaker verification EER (%) on Voxceleb1 test set with the utterance length is 1s or 3s. L_{win} and L_{step} are set to 30 frames.*

Utterance Length	Model	Accuracy	EER
1 Second	X-vector	85.8	5.75
	X-vector+Att	86.9	5.22
	H-vector	88.7	4.97
3 Seconds	X-vector	88.2	5.13
	X-vector+Att	89.2	4.79
	H-vector	90.4	4.64

utterances already had a wide range of other noises included. Not all of the experiments in this section conducts the AM-Softmax loss function as mentioned in Section 2.2.2.

Table 3.5 (above) shows the speaker identification accuracy and equal error rate on the Voxceleb1 dataset and, similarly to the earlier experiments, the H-vector model shows better performances than the two baselines. For the speaker identification task, the H-vector achieved an 88.7% accuracy with utterance lengths of 1 second and a 90.4% accuracy with utterance lengths of 3 seconds. This is a relative improvement of over 3% over the two baseline models. In the speaker verification task, H-vector reached a 4.97% equal error rate on 1-second utterance lengths and 4.64% on 3-second utterance lengths.

The performance of the models in 1-second scenario is lower than that with the 3-second scenario utterances, again because the longer utterances contain more speaker-related information. In the following experiments, the utterance length is kept at three seconds.

Noise Robustness With the Augmented Voxceleb1 Dataset

In order to evaluate the robustness of the proposed model in noisy conditions, additional noises from the MUSAN dataset are mixed with the utterances from the original Voxceleb1 dataset. As discussed in the previous section, the test set is mixed with

Table 3.6: *Speaker identification accuracies (%) for different noise types (Noise, Music and Babble) at different SNRs (0-20 dB), and the original Voxceleb1 test set. The utterance length is 3 seconds. L_{win} and L_{step} are set to 30 frames.*

Noise Type	SNR	X-vectors	Att-X-vectors	H-vectors
General	0	74.6	75.8	76.9
	5	79.5	79.4	81.3
	10	83.1	84.0	86.0
	15	85.0	86.3	87.2
	20	87.9	87.8	88.9
Music	0	68.2	70.1	72.3
	5	72.0	73.5	74.8
	10	79.4	81.0	82.9
	15	84.2	86.6	87.8
	20	86.1	88.0	89.3
Babble	0	64.1	65.2	67.9
	5	70.5	71.4	74.0
	10	77.4	77.0	78.7
	15	83.5	84.5	86.2
	20	86.6	86.9	88.1
Original		88.2	89.2	90.4

certain noise types at certain noise levels.

Tables 3.6 and 3.7 show the speaker identification accuracy and speaker verification equal error rate under different noise conditions. Three noise types are used: general noise, music and speech noise. The noise level is changed from 0dB to 20dB. The utterance length is kept at three seconds.

From the results, the proposed H-vector outperforms the two baselines in under noise conditions (general noise, music and babble). When the noise level becomes larger, such as babble and music at 0 and 5 dB, the H-vector model obtains a larger improvement. Even with babble at a level of 0dB, the proposed model achieves a prediction accuracy of 67.9% and a relative improvement of more than 5% over the X-vector model and 3% over the attentive X-vector model.

The performances of the three models are much lower than those generated with the

Table 3.7: *Speaker verification EER (%) for different noise types (Noise, Music and Babble) at different SNRs (0-20 dB), and the original Voxceleb1 test set. The utterance length is 3 seconds. L_{win} and L_{step} are set to 30 frames.*

Noise Type	SNR	X-vectors	Att-X-vectors	H-vectors
General	0	12.26	11.32	10.92
	5	10.01	9.26	9.03
	10	8.33	7.77	7.28
	15	7.25	6.76	6.50
	20	6.91	6.02	5.95
Music	0	14.15	12.92	12.68
	5	11.03	10.04	9.83
	10	9.35	8.64	8.33
	15	8.41	8.08	7.62
	20	6.79	6.25	6.17
Babble	0	30.02	27.77	26.82
	5	16.46	15.32	14.58
	10	13.26	12.53	12.38
	15	9.10	8.31	8.14
	20	7.95	7.22	7.04
Original		5.47	5.06	4.64

original dataset, confirming the discussion in Section 2.3.1 and the experimental results in Section 2.6.2 showing that background noise can corrupt the speaker-related features, thereby significantly disrupting the performance of deep neural network models. Of all the noise types, all three models performed worst with babble, at all levels from 0dB to 20dB. This confirms the observation in Section 2.4.1 that interference from other speakers, unlike that from all other noise types, contains similar features or patterns as the target speaker, all occupying the same frequency band as the target speaker.

3.5.3 Analysis

The Effect of Different Window and Step Sizes

Two key hyper-parameters of the proposed H-vector model are the window size (L_{win}) and the step size (L_{step}) and the effect of these two hyper-parameters were not evaluated

Table 3.8: *Speaker identification accuracy (%) and speaker verification EER (%) on Voxceleb1 dataset when the window size L_{win} is changed from 15 to 35 frames.*

Utterance Length	Window Size	Accuracy	EER
1 Second	15	86.4	5.24
	20	87.3	5.01
	25	89.2	4.82
	30	88.7	4.97
	35	88.3	5.11
3 Seconds	15	88.7	4.72
	20	89.6	4.43
	25	91.0	4.28
	30	90.4	4.64
	35	89.5	4.79

Table 3.9: *Speaker identification accuracy (%) and speaker verification EER (%) on Voxceleb1 dataset when the step size L_{step} is changed from 15 to 35 frames.*

Utterance Length	Step Size	Accuracy	EER
1 Second	15	87.5	4.93
	20	89.6	4.86
	25	89.4	4.92
	30	88.7	4.97
	35	87.1	5.12
3 Seconds	15	90.1	4.61
	20	91.0	4.43
	25	90.6	4.37
	30	90.4	4.64
	35	88.3	4.90

in the previous sections. The window size controls the number of frames in each segment. It is the sequence length to the input of the frame-level encoder and attention. The step size controls the overlap rate between two adjacent segments and it can also affect the number of segments delivered to the segment-level encoder.

In order to evaluate the performance of the proposed H-vector model when using different window sizes (L_{win}) and step sizes (L_{step}), Tables 3.8 and 3.9 show the prediction accuracy and equal error rate from Voxceleb1 dataset when

- a) the window size changes from 15 to 35 frames and the step size is fixed at 30 frames, and

- b) the step size changes from 15 to 35 frames and the window size is fixed at 30.

It can be observed that the model is more sensitive to a change in the window size. The model performs best with the 1-second utterance lengths when $L_{win} = 20$. The performance for both speaker identification and verification results becomes better and then drops, which may be due to the fact that the window size controls the number of frames that input to the frame-level encoder (this can be observed from Figure 3.1, Section 3.2). When the number of frames is small (e.g. 15 frames), there is not enough information for each frame-level encoder to capture, but when there are more than 35 frames in one segment, the relationships between the beginning and end frames of the current segment may be small, as they are not close to each other. This may not help the frame-level encoder to focus on the key features.

A similar phenomenon occurs when changing the step size in that the performance increases and then drops away from a peak. This is because the step size controls the overlap rate between two segments. A smaller step size of, say, 15 frames, results in more redundant information and more segment vectors, making it more difficult for the frame-level encoders to distinguish the important features for the target speaker. On the other hand, when the step size is larger than the window size, there are some frames that are not input into the network, meaning that the model will lose information.

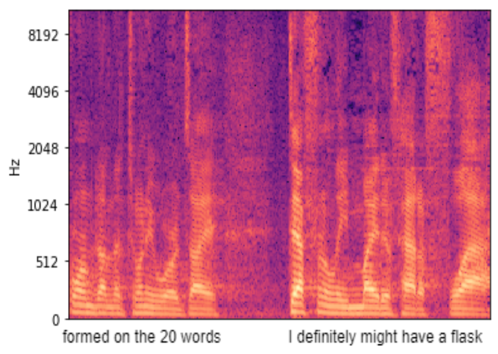
Comparison With State-of-the-Art

In the previous sections, the proposed hierarchical attention network demonstrated better model generalisation and robustness compared with the baselines. Table 3.10 shows the comparison of the H-vector model with some of the state-of-the-art techniques using the same Voxceleb1 training set and it is clear that the proposed H-vector model achieves comparable or better results. Among the models listed, the VGG-M model is based on a two-dimensional CNN architecture while the CNN+TDNN method combines the CNN with TDNN architectures. Compared with those three models, the H-vector model achieves significantly better results which is probably due to better

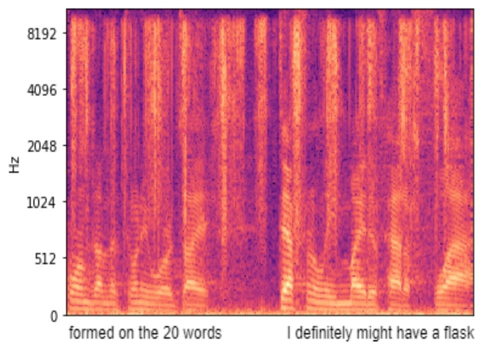
Table 3.10: *Comparison of the proposed approach with the state-of-the-art speaker verification results on the Voxceleb1 test set when using Voxceleb1 for training only.*

	Model	Loss	EER
Nagrani et al.(Nagrani et al. 2017)	VGG-M	softmax	10.2
Nagrani et al.(Nagrani et al. 2017)	VGG-M	softmax+contrastive	7.8
Shon et al. (Shon et al. 2019)	CNN+TDNN	softmax	6.79
Cai et al. (Cai et al. 2018).	ResNet-34	A-softmax+PLDA	4.46
Hajibabaei and Dai (Hajibabaei & Dai 2018)	ResNet-20	A-softmax	4.40
Hajibabaei and Dai (Hajibabaei & Dai 2018)	Retnet-20	AM-softmax	4.30
Ours	H-vector	AM-softmax	4.28

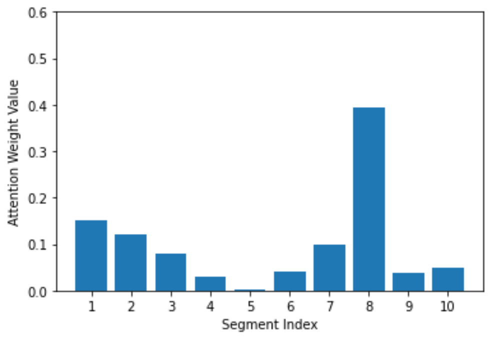
modelling and the use of AM-softmax. The hierarchical attention structure captures local and global information and thus can be more robust in the Voxceleb1 dataset which contains noisy data recorded in the real world conditions. The other three methods are based on the ResNet architecture described in Section 2.2.4, and these performed better due to the fact that residual connection is better at capturing local features. The H-vector model can deliver a slightly better performance compared to the ResNet architecture, demonstrating that capturing both local and global features by the hierarchical structure is helpful.



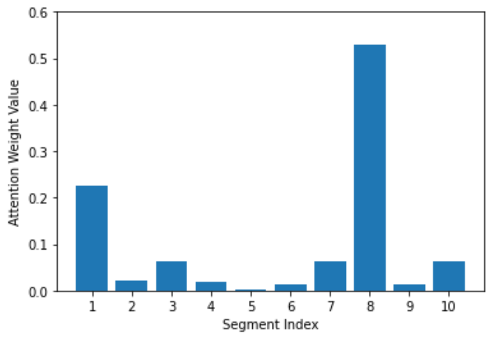
(a) Original Spectrogram



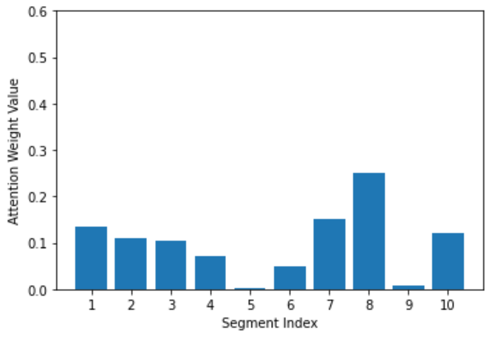
(b) Noise Corrupted Spectrogram



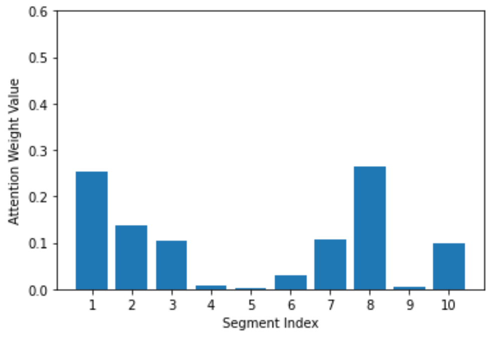
(c) Global Attention Weights from (a)



(d) Global Attention Weights from (b)



(e) H-vector Attention Weights from (a)



(f) H-vector Attention Weights from (b)

Figure 3.5: *Visualisation of attention weights. (a) the original spectrogram, (b) the noise corrupted spectrogram, (c) the global attention weights for the original spectrogram, (d) the global attention weights for the corrupted spectrogram, (e) the H-vector attention weights for the original spectrogra and (f) the H-vector attention weights for the corrupted spectrogram. Note that the number of the attention weights in the attentive X-vector is 300 (there are 300 frames in the input data) and the number of the segment-level attention weights in H-vector is 10 (10 segment vectors). In order to compare the attention weights, the attention weights of the attentive X-vector are divided into 10 groups using a sliding window (with window size L_{win} and step size L_{step}). The values for each group are averaged, results in 10 attention weight values.*

Visualisation of Attention Weights

In order to show how the attention mechanism works, Figure 3.5 provides the visualisation of the attention weights. Figure 3.5 (a) is the spectrogrm of a 3s utterance randomly selected from the Voxceleb1 dataset. Figure 3.5 (b) shows the noise corrupted spectrogram (at 0dB). The corresponding text information are marked at the bottom of the figures. For better visualisation, here demonstrate spectrograms, instead of MFCCs. Figures 3.5 (c) and 3.5 (d) show the attention weights obtained when using the attentive X-vector (global attention) on the original utterance and the noise corrupted utterance, respectively. Figures 3.5 (e) and 3.5 (f) show the attention weights obtained by using the H-vector in the same conditions. Note that the number of the attention weights in the attentive X-vector is 300 (there are 300 frames in the input data) and the number of the segment-level attention weights in H-vector is 10 (10 segment vectors). In order to compare the attention weights, the attention weights of the attentive X-vector are divided into 10 groups using a sliding window (with window size L_{win} and step size L_{step}). The values for each group are averaged, results in 10 attention weight values.

Although the weight distributions displayed in Figure 3.5(c) and (d) show that the use of both attentive X-vector and H-vector can learn the importance of features in different parts of an utterance recording, the attentive X-vector assigned a high weight value, about 0.5 to the segment 8. This means the contribution of the segment 8 is dominant over the remaining 9 segments. This might easily cause an overestimate of some class labels (or speaker identities), and thus probably leads to an incorrect decision. As a comparison, although the H-vector model allocated the highest weight to the segment 8, it is close to 0.3 as shown in Figure 3.5(e) and (f), and other segments segments are also allocated a relatively reasonable attention values. It shows the H-vector model can highlight feature contributions from multiple regions of an utterance recording.

It may be that the global attention process within the attentive X-vector model may tend to favour few number of regions over others of a recording, whereas the hierarchical structure of the H-vector model is able to highlight contributions from more regions by computing the attention weights within a small segment, and then computing the attention weights over all segments. In this way, the local features located in each individual segments can be captured by the corresponding frame-level encoders, while the global features at a higher level can be captured by the segment-level encoder.

3.5.4 Summary

In this section, the robustness of the hierarchical attention network was evaluated using the Voxceleb1 dataset combined with additional noise signals from the MUSAN dataset. In Table 3.5 in Section 3.5.2, the proposed H-vector model performs better with the Voxceleb1 dataset. In Tables 3.6 and 3.7 of Section 3.5.2, the H-vector model performed better than the baselines in almost all of the noise conditions.

The window size and the step size are the key hyper-parameters, and Tables 3.8 and 3.9 of Section 3.5.3 show how the variations of the window and step sizes were tested (from 15 frames to 30 frames). The results show the performance can be improved when the window size becomes larger, and drops away after reaching a peak. Changing the step size shows similar behaviour.

In order to compare the proposed H-vector model with the state-of-the-art models, Table 3.10 in Section 3.5.3 compared the proposed model with the state-of-the-art results that used the same training data as the H-vector model. The proposed H-vector model can deliver comparable results to the state-of-the-art models.

Figure 3.5 in Section 3.5.3 shows visualisations of the attention weights allocated by the attentive X-vector and H-vector models in order to assess how the attention mechanism works. The results show that the H-vector model can highlight more than one important feature, even in noisy conditions.

The comparison of the proposed hierarchical attention network and the two baselines shows the hierarchical structure can obtain a better noise robustness. The reason may be the hierarchical structure provides a better way to highlight important parts in the input signal by splitting the attention mechanism into two levels, each level focuses on different information.

3.6 Conclusion

In this chapter, the hierarchical attention network was proposed and evaluated. For obtaining better performance and robustness in noisy conditions, the widely used global attention has two problems. Firstly, it can only focus on a small number of features when the input utterances become longer. The second is that the global attention is unable to pay sufficient attention to the local features. The proposed hierarchical H-vector model is designed to address these two issues and improve overall performance. The hierarchical attention network splits the input utterance into several short segments. The frame-level encoder is then applied to each of the small segments, capturing the local features of the adjacent frames within each segment. The segment-level attention applies attention at a higher level, focusing on the global features by allocating different weights to the segment vectors. The details of the architecture of the hierarchical attention network are given in Sections 3.2 and 3.3.

In order to evaluate the proposed H-vector model, two baselines were used. The X-vector model represents the most widely used model for speaker recognition and it does not contain an attention mechanism, this it useful for demonstrating the effectiveness of the attention mechanism. The second baseline is the attentive X-vector model with a global attention mechanism, and this serves to compare the performance of the proposed hierarchical attention and the global attention.

The experiments were designed to evaluate model generalisation and noise robustness. Section 3.4 covered model generalisation and three datasets were used: SRE08, SWBC and CHE. SRE08 was used for training, and the SWBC and CHE datasets provided the out-of-domain data that were used for evaluation. The experimental results show the H-vector model out-performed the two baselines. The comparison of the H-vector and X-vector models shows the use of the attention mechanism is more effective than the X-vector model that treats each frame equally. The comparison of H-vector and attentive X-vector shows the proposed hierarchical attention delivers better performance compared to the global attention. In order to show the performance comparison, a DET plot and an embedding visualisation using the t-SNE algorithm were conducted. The DET plot clearly shows the performance comparison of the three models. The embedding visualisation clearly shows that the embedding generated by the H-vector model has a smaller intra-class distance and larger inter-class distance.

Section 3.5 focuses on noise robustness evaluation using the Voxceleb1 dataset, augmented by additional noises from MUSAN dataset. The results with the original Voxceleb1 dataset and the augmented Voxceleb1 dataset shows the H-vector model performs well in almost all noise conditions. Even with babble, the most challenging of the three noise conditions, the proposed model out-performs the two baselines.

The performance of the H-vector model was also compared with published state-of-the-art results that use the same training set (Voxceleb1) and the proposed model stands up well to all of them. The results from the H-vector model were slightly better than the ResNet-based models, showing that the hierarchical attention structure is comparable to ResNet. In order to show how the attention works, the attention weights were visualized. The comparison of the attention weights obtained by global attention and hierarchical attention indicates that the hierarchical attention can highlight multiple important features, as well as capturing features somewhat corrupted by noise.

In conclusion, the proposed hierarchical attention network splits the attention mechanism between frame level and segment level to capture both local and global features. The experimental results show this architecture can deliver better model generalisation and noise robustness as compared to the widely used X-vector and attentive X-vector models.

Chapter 4

Speech Enhancement For Speaker Recognition

4.1 Introduction

In Section 2.3, different types of noise interferences were discussed, it was pointed out that background noise can contain fluctuating or steady noise and that significantly affect the quality of the speech signal. The performance of speaker recognition models can also be influenced as different types of noise can corrupt features within the signal.

Section 2.3.3 discussed speech enhancement as one of the potential solutions to this problem. In contrast to the attention mechanism discussed in Section 2.3.2, a speech enhancement model can be built independently as a front-end pre-processing model to filter out noise interference by either generating a time frequency mask or directly predicting the clean signal from the noisy input. However, the independent training of speech enhancement model may cause mismatch issues. The mismatch problem arises when the speech enhancement model not only filters out the noise interference from the noisy input, but also some of the features required by the back-end speaker recognition model can also be corrupted (see Section 2.3.3). The reason for the mismatch problem is that the training target of the speech enhancement model is to reduce the noise by either generating a mask or directly producing a clean signal, whereas there are no constraints on the features that are used for the back-end model (e.g. speaker related features for the back-end speaker recognition system) (Sadjadi & Hansen 2010, Shon et al. 2019, Wang & Wang 2016).

In order to overcome this problem, a joint training framework of the speech enhancement and speaker recognition models may be a solution. It can make the speech enhancement model filter out the noise information and remain the useful features for the speaker recognition model at the same time. From review of recent relevant literature in Section 2.3.3, it emerged that most speech enhancement systems are trained independently. Shon et al. (2019) developed a method that integrates the speech enhancement and speaker recognition models but, even in this model, the two models still had to be trained separately.

In this chapter, a joint training framework of speech enhancement and speaker recognition is proposed, with two novel model architectures, namely the joint training of speech enhancement for speaker recognition, and speaker dependent speech enhancement for speaker recognition. In the first proposed model, the speech enhancement and speaker recognition models are trained using one objective function. Within the model architecture, a novel multi-stage attention mechanism is proposed to improve the performance of the speech enhancement model. The second proposed approach aims to further improve the performance using speaker embedding that is pre-trained and used in the speech enhancement model. Such speaker dependent speech enhancement in the joint training framework can improve the performance under various noise conditions.

4.1.1 Chapter Outline

The rest of this chapter is organized as follows:

- Section 4.2 introduces the overall structure of the joint training framework.
- Section 4.3 includes the model architecture of the joint system, as well as the

experimental setup and the results, followed by a discussion.

- Section 4.4 proposes a speaker-dependent speech enhancement model based on the joint training framework. In this section, the experimental setup, results and discussion are also included.
- A conclusion of this chapter is provided in Section 4.5.

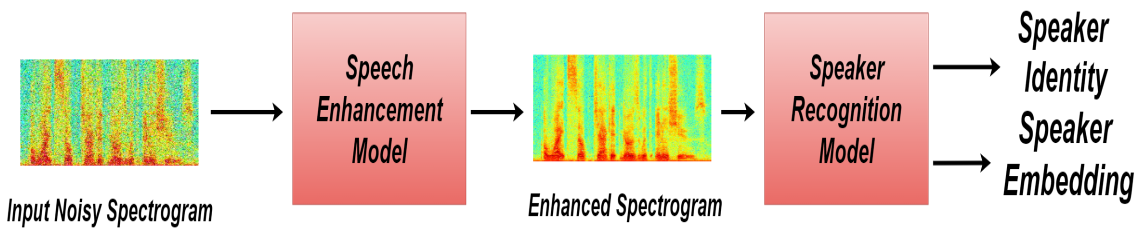


Figure 4.1: *The overall structure of the joint training framework of the speech enhancement and speaker recognition models.*

4.2 A Joint Training Framework

In this section, the overall structure of the proposed joint training framework is introduced. The proposed joint training system (as detailed in Section 4.3) and the speaker dependent speech enhancement system (as detailed in Section 4.4) are all based on the structure described in this section. The following section will introduce the detailed model architectures for each of the models within the framework.

Figure 4.1 shows the overall architecture. In this framework, $\boldsymbol{X}_N \in \mathcal{R}^{T\times F}$ denotes the input noisy spectrogram, where T and F denote the time and frequency dimensions respectively. The speech enhancement model is denoted as "SE-Net" and the speaker recognition model is denoted as "SR-Net" in this section. The input to the speech enhancement model is $\boldsymbol{X}_N$. The model learns a time-frequency mask which is then multiplied with $\boldsymbol{X}_N$. The result is the estimated clean spectrogram $\boldsymbol{X}_C^{'} \in \mathcal{R}^{T\times F}$, which has the same dimension as $\boldsymbol{X}_N$. The estimated clean spectrogram is then fed to the SR-Net. The speaker identities are the output of the SR-Net and the speaker embeddings are extracted from the SR-Net for speaker verification.

The objective function contains two parts; the first is the so called reconstruction loss and the second is the cross-entropy loss. Equation 4.1 shows the reconstruction loss of the SE-Net, where x_{ij} and $x_{ij}^{'}$ denote each element in the clean and denoised spectrogram, T and F denote the dimension on time and frequency axes, respectively. The mean absolute error (MAE) (Willmott & Matsuura 2005) is used to measure the difference between the reference clean spectrogram $\boldsymbol{X}_C$ and the estimated clean spectrogram $\boldsymbol{X}_C^{'}$. This is a constraint on the SE-Net to filter out the noise information.

$$\mathcal{L}_{SE} = \frac{1}{TF}\sum_{i=1}^{T+1}\sum_{j=1}^{F+1}|x_{ij} - x_{ij}^{'}| \tag{4.1}$$

The second part of the objective function is shown in Equation 4.2 in which y and $y^{'}$ denote the ground truth speaker label and model prediction from the SR-Net (discussed in Section 2.1.1). N is the total number of speakers in the training set. It is the cross-entropy loss that was discussed in Section 2.2.3. Minimising the cross-entropy loss can restrict the joint system from learning features related to the target speakers (Shon et al. 2019).

$$\mathcal{L}_{SR} = -\sum_{i=1}^{N+1} y_i \log y_i^{'} \tag{4.2}$$

In order to make the joint training model reduce the noise influence and retain speaker related features at the same time, the joint framework shown in Figure 4.1 is trained using Equation 4.3, which is a combination of Equations 4.1 and 4.2. In practice, and in order to accelerate the training process, the SE-Net and the SR-Net are pre-trained independently using Equations 4.1 and 4.2. Then, the two models are integrated together and fine-tuned using Equation 4.3.

$$\mathcal{L} = \mathcal{L}_{SE} + \mathcal{L}_{SR} \tag{4.3}$$

4.3 Joint Training of Speech Enhancement and Speaker Recognition

4.3.1 Model Architecture

This section builds upon the architecture depicted in Figure 4.1 and shows the details of the various components of the joint system, including the model architecture of the speech enhancement model, the speaker recognition model and the proposed multi-stage attention mechanism.

Speech Enhancement Model

The architecture of the proposed speech enhancement and speaker recognition joint system consists of a speech enhancement model and a speaker recognition model. The noisy spectrogram input is denoted as $\boldsymbol{X}_N \in \mathcal{R}^{T \times F \times C}$, where T, F, and C represent the time dimension, frequency dimension, and channel dimension, respectively. In contrast to the description in Section 4.2, here the channel dimension is written as it will be used in the multi-stage attention mechanism (later in this section). The value of C of the input spectrogram $\boldsymbol{X}_N$ is one. Note that the term channel in this Chapter is different from that from Chapter 2. It is not the number of microphones of the input signal, but the number of features or kernels of a convolutional layer. This will be explained in detail in the next sections.

The speech enhancement model consists of multiple convolution and multi-stage attention (CONV-MS) blocks. Each of them contains a dilated convolution layer followed by a multi-stage attention mechanism. The dilated convolution (Yu & Koltun 2015) is a type of convolution that has a larger receptive field (discussed in Section 2.1.2). It was shown by Tan et al. (2018) and Pandey & Wang (2020) to have a better performance in speech enhancement. For each CONV-MS block, the output of the dilated convolutional layer is denoted as $\boldsymbol{H}_k \in \mathcal{R}^{T_k \times F_k \times C_k}$, where k means the kth CONV-MS block. The term k used here is to distinguish different CONV-MS blocks, as a neural network may have multiple CONV-MS blocks. This will be explained later in this section. $\boldsymbol{H}_k$ is then input to a multi-stage attention (MS) block. The output $\boldsymbol{H}_k'''$ denotes the refined features of the kth Conv-MS block, whose dimension is the same as $\boldsymbol{H}_k$ (the reason for the three superscripts can be found in the following section). The estimated clean spectrogram is the output of the last CONV-MS block. The detailed model architecture of SE-Net can be found in Table A.2, Appendix A.2.

Speaker Recognition Model

The speaker recognition model consists of multiple residual convolutional layers and a multi-stage attention mechanism (RES-MS block). Similar to the description in Section 4.3.1, the input of the kth residual block is denoted as $\boldsymbol{H}_k \in \mathcal{R}^{T_k \times F_k \times C_k}$, and the final refined feature map of the kth RES-MS block is $\boldsymbol{H}_k'''$. Within each RES-MS block, a multi-stage attention mechanism is applied. The last RES-MS block is followed by fully-connected layers, by which the predictions of speaker identities are finally computed. The notations of the input from, and output to, the CONV-MS block and the RES-MS block ($\boldsymbol{H}_k$ and $\boldsymbol{H}_k^{'''}$) are the same for both SE-Net and SR-Net.

The ResNet-20 architecture (as discussed in Section 2.2.4) is used, because it contains residual connections that can make the model easier to train using a large amount of data. The detailed model architecture can be found in Table A.3, Section A.2.

Multi-Stage Attention Mechanism

One of the key components of the speech enhancement model is the multi-stage attention mechanism (MS). It can work as a noise filter, filtering out interference by assigning attention weights to the input data. It was shown by Yadav & Rai (2020) that applying attention mechanism in different dimensions of the input speech signal can achieve better noise robustness in speaker recognition. Thus, in order to better filter out noise interference in different dimensions, the multi-stage attention mechanism can be applied for different dimensions of the input data. The following describes the architecture of the multi-stage attention mechanism.

The output of the CNN layer in CONV-MS or RES-MS blocks contains the three dimensional feature map (time, frequency and channel dimensions). As shown in the previous section, it is denoted as $\boldsymbol{H}_k$ and the time, frequency and channel dimensions are denoted as T_k, F_k and C_k respectively. All three dimensions may contain noise information from the input signal, propagating through the layers in the neural network. Therefore it is necessary to filter them out in each dimension.

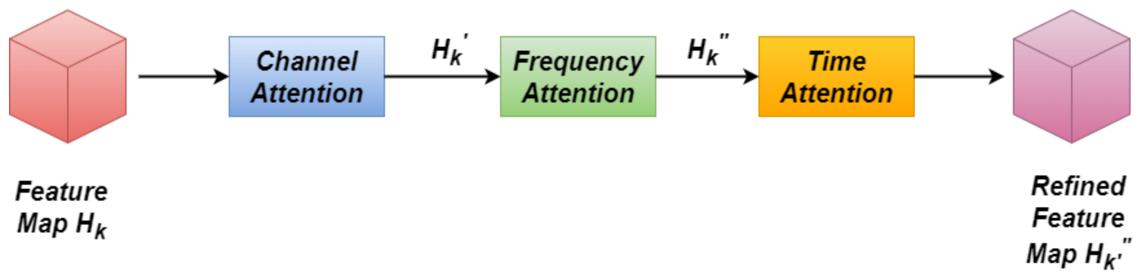


Figure 4.2: *The multi-stage attention mechanism.*

In order to achieve this, the proposed multi-stage attention mechanism works in all three dimensions. Figure 4.2 shows the diagram of the multi-stage attention mechanism, and its computational process is shown in Equation 4.4.

$$\begin{aligned} \boldsymbol{H}_k^{'} &= \mathrm{Broad}(\boldsymbol{\alpha}_{C,k}) \odot \boldsymbol{H}_k \\ \boldsymbol{H}_k^{''} &= \mathrm{Broad}(\boldsymbol{\alpha}_{F,k}) \odot \boldsymbol{H}_k^{'} \\ \boldsymbol{H}_k^{'''} &= \mathrm{Broad}(\boldsymbol{\alpha}_{T,k}) \odot \boldsymbol{H}_k^{''} \end{aligned} \tag{4.4}$$

The attention mechanism is first used in the channel dimension. The attention weight vector $\boldsymbol{\alpha}_{C,k} \in \mathcal{R}^{1\times 1\times C_k}$ is obtained and repeated to the same dimension as $\boldsymbol{H}_k \in \mathcal{R}^{T_k\times F_k\times C_k}$, and then element-wise multiplication (denoted as $\odot$) is used to obtain the refined feature map $\boldsymbol{H}_k^{'}$. The broadcasting function Broad(.) is defined in Section 2.3.2, the super-script that denotes the target dimension $(T_k \times F_k \times C_k)$ are omitted for simplicity.

The output is then processed to a similar attention mechanism in the frequency dimension. As per the previous step, the refined feature map is then processed to the final attention mechanism, the time attention. The output of the time attention is the final refined feature map $\boldsymbol{H}_k^{'''}$, in which the noise influences across the channel, frequency and time dimensions may be reduced. The multi-stage attention mechanism can be viewed as an independent block that can be flexibly added after each convolutional layer in either the SE-Net or the SR-Net (details see Section 4.3.4). The computational process of the attention weight vectors $\boldsymbol{\alpha}_{C,k}$, $\boldsymbol{\alpha}_{F,k}$ and $\boldsymbol{\alpha}_{T,k}$ can be found in Appendix A.2.1.

4.3.2 Experiments

Data Processing

Rather than computing the MFCC feature that was used in Chapter 3, the experiments in this chapter take the spectrogram as the input. Spectrograms retain the detailed local features in the frequency domain that are widely used as the input for the speech enhancement model (Shon et al. 2019, Yadav & Rai 2020, Ming et al. 2007). A second reason is that, from the spectrogram input, it is convenient to observe the effectiveness of the noise reduction. In Section 4.4 in this chapter, the denoised spectrograms are visualised in order to compare different techniques.

Experimental Setup

The experiments focus on noise robustness and use the Voxceleb1 dataset and the MUSAN dataset. The experimental setup in this section is the same as that in Section 3.5, and details can be found in Appendix B.1. To evaluate the recognition performance, Top-1 and Top-5 accuracies are employed for speaker identification, as discussed in Section 2.2.3. The evaluation metrics for speaker verification are the equal error rate and the minimum Detection Cost Function.

Baselines

In order to comprehensively evaluate the proposed model, the selection of the baselines is important. The following description shows the descriptions of the three baselines and the three configurations of the proposed models:

SR is the speaker identification baseline using the speaker recognition model only.

SEP is the baseline that the speech enhancement model and speaker recognition model are trained separately.

VoiceID is the baseline from Shon et al. (2019), where the speech enhancement and speaker recognition models are cascaded, but the two models are not trained jointly.

SE+SR is the proposed model that jointly optimises speech enhancement and speaker recognition without a multi-stage attention.

SE-MS+SR is the proposed model using joint optimisation and a multi-stage attention in the speech enhancement model.

SE+SR-MS is the proposed model using joint optimisation and a multi-stage attention in speaker recognition model.

A comparison of SR, SEP and VoiceID can show the effectiveness of the joint training strategy. A comparison of SE+SR, SE-MS+SR and SE+SR-MS can demonstrate the effectiveness of the proposed multi-stage attention. Note that the softmax loss function is used as the loss function for all of the models, rather than the AM-softmax function used in Chapter 3, because of its use in the work on VoiceID; keeping all the models the same in this respect makes for a fairer and more transparent comparison. Further experiments using AM-softmax are conducted in Section 4.4.4.

4.3.3 Results and Discussion

The Noise Robustness of the Joint System

In order to show the effectiveness of the joint training strategy and the noise robustness of the joint system, Table 4.1 shows speaker identification results obtained using the models listed in Section 4.3.2. The best result in each row is highlighted to facilitate observation.

Compared to the SR baseline and the SEP baseline, SE+SR yields better performance for speaker identification. After using multi-stage attention models, SE+SR-MS and SE-MS+SR, about 2% to 3% further improvements on Top-1 and Top-5 accuracy

Table 4.1: *Speaker identification accuracies (Top-1 (red) and Top-5 (yellow) accuracies %) on the Voxcebe1 test data, corrupted by three types of noise (Noise, Music and Babble) at different SNR (0-20 dB) levels. Five different scenarios are tested: SR, SEP, SE+SR, SE-MS+SR and SE+SR-MS.*

Noise Type	**SNR**	**SR**		**SEP**		**VoiceID**		**SE+SR**		**SE-MS +SR**		**SE+SR-MS**	
		Top1	Top5	Top1	Top5	Top1	Top5	Top1	Top5	Top1	Top5	Top1	Top5
General	0	74.1	86.9	73.2	84.3	75.6	88.0	76.3	88.9	**78.5**	**90.0**	77.7	89.2
	5	79.2	90.0	78.8	88.2	80.4	90.8	81.1	91.8	**83.4**	**92.1**	81.9	91.8
	10	83.2	93.2	81.4	89.7	84.7	94.3	86.0	94.7	**87.3**	**95.6**	86.7	95.1
	15	84.9	94.6	84.3	92.0	85.6	95.1	87.3	95.8	**89.5**	**96.7**	88.8	96.0
	20	87.9	95.4	88.0	93.4	88.7	96.0	89.1	96.6	**90.9**	**97.5**	90.2	97.0
Music	0	65.8	82.0	63.9	79.8	67.1	83.3	67.7	83.7	**70.3**	**84.1**	69.5	83.5
	5	76.9	89.1	75.1	88.4	78.2	89.9	80.0	91.0	**81.6**	**91.5**	80.6	90.8
	10	83.8	93.5	83.6	92.5	84.6	94.2	85.2	94.7	**86.3**	**95.3**	85.8	94.7
	15	86.1	93.9	85.4	92.7	87.3	95.0	88.4	95.6	**89.1**	**96.7**	88.2	95.4
	20	87.4	94.7	87.9	93.0	88.9	95.6	89.1	96.0	**90.2**	**97.1**	89.5	96.6
Babble	0	62.4	80.2	59.8	77.2	63.8	82.1	65.7	81.5	**67.5**	**83.0**	66.6	81.9
	5	76.2	87.3	73.2	85.3	77.6	88.7	78.6	88.9	**80.6**	**89.9**	79.3	89.6
	10	81.4	92.2	79.4	90.5	82.3	93.5	84.6	93.6	**86.6**	**94.5**	85.3	83.2
	15	84.0	92.6	81.2	90.8	86.1	94.0	86.8	93.9	**88.3**	**94.7**	87.6	94.0
	20	85.8	92.9	84.0	91.4	86.6	95.1	87.1	94.6	**89.0**	**95.5**	88.8	95.2
Original		88.5	95.9	86.4	92.9	89.7	96.4	89.8	96.5	**91.9**	**97.6**	90.8	97.3

are obtained in comparison to the baseline in all noise conditions. Compared to SE+SR, the use of the attention model can also show about 1% to 2% relative improvement even if the SNR is at 0dB level. This may probably because the use of an attention mechanism can highlight the speaker-related information and reduce the interference caused by irrelevant noise signals. Note that the results of SR in Table 4.1 is different from the speaker identification accuracies in Table 2.2. This is because the different input features and model architectures. The experiments in this chapter uses the spectrogram as the input feature and ResNet-20 as the model architecture. While the experiments in Section 2.6.2 uses 20 dimensional MFCC as the input feature, and X-vector as the model architecture.

With regard to speaker verification, the tests revealed similar tendencies, as shown

Table 4.2: *Speaker verification EERs (%, blue) and DCFs (green) on Voxceleb1 test data, corrupted by different types of noise (Noise, Music and Babble) at different SNRs (0-20 dB). Six different scenarios are tested: SR, SEP, VoiceID, SE+SR, SE-MS+SR, SE+SR-MS.*

Noise Type	**SNR**	**SR**		**SEP**		**VoiceID**		**SE+SR**		**SE-MS+SR**		**SE+SR-MS**	
		EER	DCF	EER	DCF	EER	DCF	EER	DCF	EER	DCF	EER	DCF
General	0	16.94	0.933	17.88	0.946	16.56	0.938	16.20	0.912	**15.95**	**0.901**	16.13	0.908
	5	12.48	0.855	14.02	0.902	12.26	0.830	11.99	0.819	**11.76**	**0.805**	11.78	0.812
	10	10.03	0.760	11.94	0.828	9.86	0.747	9.54	0.732	**9.17**	**0.717**	9.29	0.727
	15	8.84	0.648	9.28	0.734	8.69	0.686	8.48	0.665	**8.08**	**0.639**	8.10	0.641
	20	7.96	0.594	8.86	0.699	7.83	0.639	7.52	0.629	**7.07**	**0.615**	7.09	0.623
Music	0	17.04	0.940	20.25	0.949	16.24	0.913	15.96	0.901	**15.58**	**0.899**	15.89	0.904
	5	11.54	0.828	17.26	0.925	11.44	0.818	11.15	0.805	**10.93**	**0.791**	11.04	0.801
	10	9.69	0.749	15.32	0.878	9.13	0.733	9.12	0.731	**8.87**	**0.714**	8.97	0.725
	15	8.40	0.689	12.16	0.811	8.10	0.677	8.08	0.643	**7.62**	**0.621**	7.77	0.629
	20	7.70	0.665	10.42	0.724	7.48	0.635	7.39	0.619	**7.13**	**0.607**	7.26	0.614
Babble	0	38.90	1.000	46.50	1.000	37.96	1.000	37.53	0.999	37.55	0.999	**37.46**	**0.998**
	5	28.04	0.998	35.28	1.000	27.12	0.996	26.97	0.979	26.42	0.981	**26.35**	**0.977**
	10	17.34	0.917	21.25	0.958	16.66	0.926	16.44	0.911	**16.30**	**0.907**	16.36	0.911
	15	11.31	0.795	18.87	0.927	11.25	0.807	11.24	0.801	**10.89**	**0.795**	10.94	0.801
	20	9.12	0.720	12.46	0.852	8.99	0.705	8.77	0.695	**8.39**	**0.677**	8.51	0.688
Original		6.92	0.565	9.29	0.697	6.79	0.574	6.41	0.541	**6.18**	**0.528**	6.26	0.535

in Table 4.2. It is clear that SE+SR with the use of joint optimisation performs better than VoiceID which uses only a pre-trained speaker identification model, instead of joint optimisation. In comparison to the speaker identification results, the verification improvements obtained using SE-MS+SR and SE+SR-MS are relatively small. For speaker verification, the use of an attention model in the speech enhancement model can yield better results in almost all conditions, except when speech is corrupted by babble at SNR levels of 0dB and 5dB. This exception could be due to the fact that babble signals are relatively complex as they have characteristics similar to speech. The use of an attention mechanism in the speaker recognition model might be more suitable for extracting speaker-relevant information than using it in the speech enhancement model when the acoustic environment is poor.

From the results presented in Tables 4.1 and 4.2, several points can be highlighted. Firstly, from the comparison of the SR, SEP ,VoiceID and SE+SID, it is obvious that the joint training system (SE+SR) proposed in this section can deliver better results than those obtained by the SR, SEP and VoiceID. This indicates the joint training system can reduce the mismatch problem discussed in Section 2.3.3.

The results obtained by SEP are worse than that from SR. This phenomenon confirms the discussion in Section 2.3.3 of the mismatch problem, where the separate training target of the speech enhancement network may corrupt some useful information to the speaker recognition model when reducing the noise.

Secondly, the best results are mainly obtained with the SE-MS+SR model. This indicates that the multi-stage attention mechanism used in the speech enhancement model is more efficient than using it in the speaker recognition model. The reason may because the SE-Net can access the original noisy spectrogram, whereas the multi-stage attention models in the speech enhancement model are closer to the noisy data than those in the speaker recognition model. The design of the multi-stage attention model is to filter out the noise interference in time, frequency and channel dimensions. The output feature maps in the speech enhancement model contains more noise information, therefore the multi-stage attention models in the speech enhancement model may filter out more noise information and thereby contribute more to the final prediction.

Finally, compared with the results obtained by the H-vector model in Table 3.7 of Section 3.5.2 which were obtained using the same data construction and experimental setup, one can observe that the results obtained in this experiments are worse. This is because the use of the AM-softmax in Section 3.5.2 reduces the intra-class distance and enlarges the inter-class distance, while the softmax function can only enlarge the inter-class distance (as discussed in Section 2.2.2). This property can significantly influence the performance of the speaker verification task which relies on the similarity scores between two speaker embeddings.

Table 4.3: *The comparison of the speaker identification accuracies (top-1 accuracy %) obtained by the different attention mechanisms in the multi-stage attention mechanism. T, F, C represents the independent computation of time, frequency and channel attentions respectively. The C-F-T indicates the order of the three attention mechanisms, being the channel attention first, then the frequency attention, and finally the time attention. The SNR of the noises are 0dB.*

Noise Type	C-F-T	T	F	C
General	78.5	76.6	77.5	77.7
Music	70.3	67.6	68.9	69.8
Babble	67.5	65.8	66.7	67.3

4.3.4 Analysis

The Effectiveness of the Time, Frequency and Channel Attentions

The previous results shown the multi-stage attention played an important role in the joint system. In order to further evaluate the its effectiveness, the time, frequency and channel attentions are used independently in the speech enhancement model. In Table 4.3, the results obtained are shown. The T, F, C represents the time, frequency and channel attentions respectively in the speech enhancement model. The results are compared with the C-F-T, which represents the SE-MS+SR model in Table 4.1 that is using the order of the attention mechanisms as channel, frequency and time attention. The models are evaluated on different noise types at 0dB. The speaker identification results in Table 4.3 show the channel attention contributes most to the final results (reached 77.7% in the general noise scenario compared to 78.5% obtained by the C-F-T). Frequency attention provides the second largest contribution while the time attention contributes less to the final prediction.

This phenomenon indicates channel dimension contains more noise information. As discussed in Section 2.1.2, in a two dimensional CNN architecture, the channel dimension represents the features learned from the convolutional kernels. For each dimension of the feature map within a CNN architecture, each channel dimension represents the summation of a learned feature map by a convolutional kernel. When the layers becomes deeper, the time and frequency dimensions are smaller while the channel dimension becomes larger. Thus, the channel dimension may contain more information about the input data, as well as more noise interference, than the other two dimensions. The attention mechanism used in channel dimension may filter out more noise, thereby contributing more to the final prediction.

It can also be observed from Table 4.3 that the frequency attention obtained better results than those obtained by the time attention, indicating that there is more noise information in the frequency dimension. As discussed in Sections 2.3.1 and 2.4.1, and shown in Figures 2.9 and 2.12, noise influences the signal mostly in the frequency dimension as fluctuating noise is the major background noise type and it is fluctuating in the frequency dimension. Thus, the attention mechanism used in the frequency dimension can provide a greater contribution to the final prediction than the time attention.

Different Orders of the Time, Frequency and Channel Attentions

In order to verify the above findings, the multi-stage attention is organised in different orders. Table 4.4 shows the speaker identification results obtained by the different multi-stage attention mechanisms. In each version, the order of the three attention computing processes are changed. For simplicity, the different version of the multi-stage attention is denoted by the order in which the symbols appear (e.g. C-F-T denotes channel-frequency-time).

Although the results obtained by C-F-T are better than those obtained by C-T-F, the difference is relatively small. For the babble tests, C-F-T reaches 67.5% accuracy and C-T-F reaches 67.2%, suggesting that switching the time / frequency order makes little difference to the final outcome. The best results come from the C-F-T order and, when considering those where the channel attention moves away from first place, it is

Table 4.4: *The comparison of the speaker identification accuracies (top-1 accuracy %) obtained by different orders of the time, frequency and channel attentions in the multi-stage attention mechanism. T, F, C represents the independent computation of time, frequency and channel attentions. The C-F-T indicates the order of the three attention mechanisms in the multi-stage attention and, in this case, it is the channel attention first, then the frequency attention, and finally the time attention. The SNR of the noises are 0dB.*

Noise Type	C-F-T	C-T-F	F-C-T	F-T-C
General	78.5	77.9	76.4	75.5
Music	70.3	69.8	67.2	65.9
Babble	67.5	67.2	65.8	64.3

clear that the further it is from the noisy input, the worse the results become. For example, when the channel attention is applied first in C-F-T, it can reach, 78.5% with general noise, but when it is moved to the end, the identification accuracy drops to 75.5%. This verifies the findings in Table 4.4 that the channel attention can capture and filter out more noise information. The performance of the model is clearly sensitive to the order of the attention mechanisms.

The results from testing the F-C-T and F-T-C combinations were worse than when the attention mechanism is not used (SE+SR reached 67.7% with music noise while F-C-T reached 67.2% and F-T-C reached 65.9%). Again, it is clear that the order of the attention mechanisms affects performance, and this is because the three mechanisms are deployed sequentially, rather than in parallel, which means that errors in the output of the first mechanism can propagate to the other two. The most effective order, then, is C-F-T, which follows the ordering of importance shown in Table 4.3.

4.3.5 Summary

In this section, a joint training system of speech enhancement and speaker recognition was proposed to address the mismatch problem. In Section 4.2, a high level introduction of the joint training system was provided, and the detailed architecture was in Section 4.3. The joint training system contains a speech enhancement model and a speaker recognition model, trained using one loss function that consists of a reconstruction loss and a classification loss. The reconstruction loss constrains the model to filter out the noise information while the classification loss constrains the model to retain speaker related features at the same time. A multi-stage attention mechanism is proposed and applied in both models. The multi-stage attention mechanism computes attention weights for each dimension of the output feature map within the CNN architecture in both models.

In Section 4.3.2, the experiments were conducted on the augmented Voxceleb1 dataset to evaluate the robustness of the proposed model compared to multiple baselines. The results are shown in Section 4.3.3. Comparing the proposed joint training system with the baseline system that does not use a speech enhancement system shows the proposed system provides better noise robustness. When the proposed system was compared to the baseline that separately trained the two models indicates the joint training system can reduce the mismatch problem and obtain better results, reaching 91.9% identification accuracy and 6.18% EER with the original Voxceleb1 test set without using the AM-softmax function. The combined results of deploying the multi-stage attention in both the speech enhancement and speaker recognition models shows that using it in speech enhancement model is more efficient as it is closer to the original noisy input. Further experiments in Section 4.3.4 shows that the time, frequency and channel attentions contribute differently to the final prediction, with the latter contributing the most. The ordering of these three attention mechanisms is also important, and the best results come from applying the channel attention first.

4.4 Speaker Dependent Speech Enhancement For Speaker Recognition

In the previous section, a joint training framework that integrates the speech enhancement and speaker recognition models was proposed. The results show that joint training system can deliver better noise robustness compared to not using the joint training framework or not using the speech enhancement model.

The performance of the joint training framework may be further improved by the pre-trained speaker embeddings. As shown in Wang et al. (2019), discussed in Section 2.4.2, for the target speaker extraction approach, pre-trained speaker embeddings play an important role as they contains prior information of the target speaker, which helps the model retain more information of the target speaker when reducing the noise impact. Thus, the performance of the target speaker extraction network can benefit from the speaker embedding.

Based on the above discussion, it is assumed that the performance of the speech enhancement model in the proposed joint training framework can also be improved by the pre-trained speaker embeddings. In order to prove this hypothesis, this section proposes an architecture that uses pre-trained speaker embeddings to improve the performance of the joint system. The proposed approach in this section is further referred to as the speaker dependent speech enhancement method for speaker recognition. The speech enhancement model reduces the noise for specific target speakers, and the speaker recognition model benefits from the speaker dependent noise reduction process.

4.4.1 Model Architecture

Figure 4.3 shows the architecture of the proposed approach, consisting of two steps (Step1 and Step2), where each step contains both a speech enhancement model (SE-Net1 and SE-Net2) and a speaker recognition model (SR-Net and SR-Net2). Given an

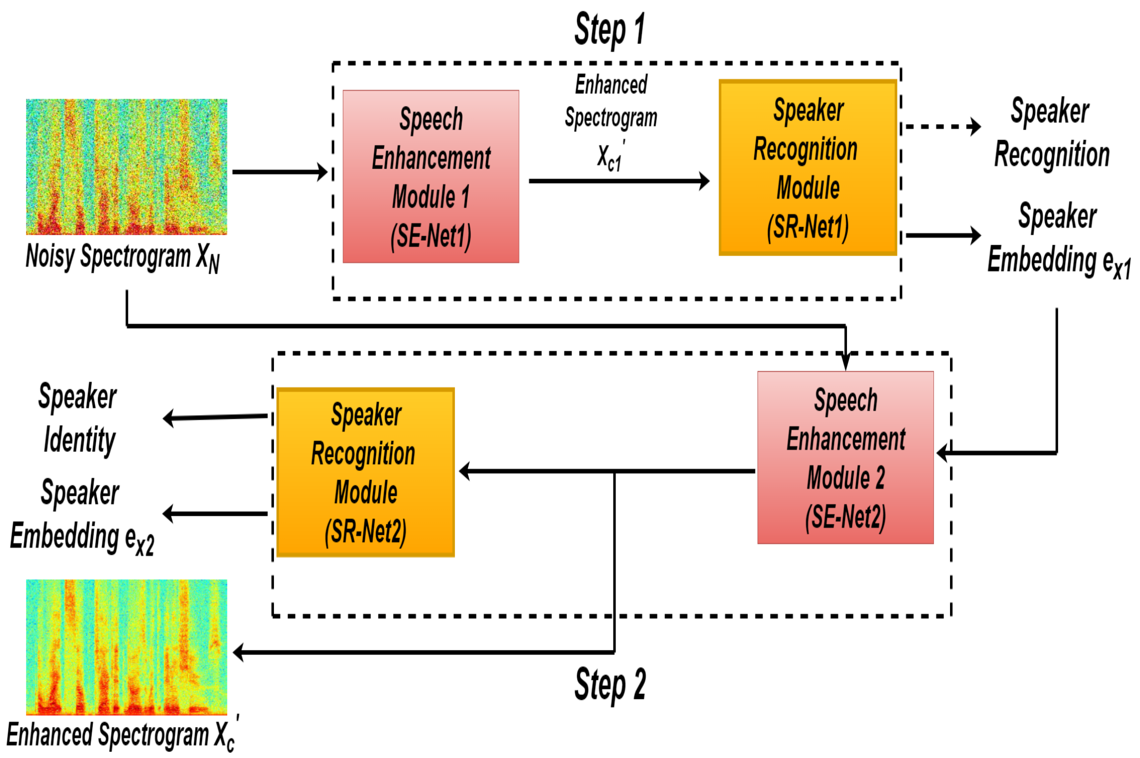

Step 1
Noisy Spectrogram X_N
Speech Enhancement Module 1 (SE-Net1)
Enhanced Spectrogram X_{c1}'
Speaker Recognition Module (SR-Net1)
Speaker Recognition
Speaker Embedding e_{x1}
Speech Enhancement Module 2 (SE-Net2)
Speaker Recognition Module (SR-Net2)
Speaker Identity
Speaker Embedding e_{x2}
Enhanced Spectrogram X_c'
Step 2

Figure 4.3: *Architecture of the proposed approach consisting of two steps (Step1 and Step2), each of which contains a speech enhancement model (SE-Net1 and SE-Net2) and a speaker recognition model (SR-Net1 and SR-Net2). The input is a spectrogram corrupted by noise. A speaker identity and an enhanced spectrogram are the output.*

input spectrogram $\boldsymbol{X_N}$, the goal for Step1 is to generate a speaker embedding ($\boldsymbol{e}_{x1}$) using SE-Net1 and SR-Net1 that were developed in Section 4.2. In Step2, the speaker embedding $\boldsymbol{e}_{x1}$ is used as the prior information to improve the speaker recognition and speech enhancement performances. The architecture of the speech enhancement model (SE-Net2) and speaker recognition model (SR-Net2) have similar architectures to the SE-Net1 and SR-Net1 in Step1. The only difference is that SE-Net2 takes $\boldsymbol{e}_{x1}$ into account. The details of the training strategy and the model architectures for each component are introduced in the following sections.

Speaker Embeddings (Step1)

As shown in Figure 4.3, in the first step, SE-Net1 and SR-Net1 are cascaded. The noise corrupted spectrogram $\boldsymbol{X}_N$ is denoised using SE-Net1, and a speaker embedding is then obtained by the first fully connected layer of SR-Net1. The training strategy in Step1 is the same as that described in Section 4.2, where the speech enhancement and speaker recognition models are firstly pre-trained using the reconstruction loss and the classification loss functions respectively. Then, the joint training system is trained using the combination of these two loss functions. The experimental results in Section 4.3 show that the speaker embeddings extracted from the joint system (used for speaker verification) deliver better performances in noisy conditions. As a result, the joint system is used as a robust speaker embedding extractor in this section, the extracted speaker embedding $\boldsymbol{e}_{x1}$ is used as the prior knowledge for Step2.

Speaker-Dependent Speech Enhancement (Step2)

In Step2, both the speech enhancement and speaker recognition models use similar structures to those in Step1. However, unlike SE-Net1, the SE-Net2 concatenates the speaker embedding vector $\boldsymbol{e}_{x1}$, with the output of one of its bottleneck layers and enhances the quality of $\boldsymbol{X}_N$.

The optimisation of Step1 and Step2 are independent of each other. The parameters of SE-Net1 and SR-Net1 used in Step1 are fixed when training SE-Net2. SR-Net2 shares weights with SR-Net1. A joint optimisation is implemented for the two models in Step2 by using the Equation 4.5, where the computation of $\mathcal{L}_{SE}$ and $\mathcal{L}_{SR}$ are the same as that in Section 4.2 (Equations 4.1 and 4.2).

$$\mathcal{L} = \mathcal{L}_{SE} + \mathcal{L}_{SR} \tag{4.5}$$

There are several important points in Step2. Firstly, although it contains its own speech enhancement (SE-Net2) and speaker recognition (SR-Net2) models, the weights are shared. The only difference between SE-Net1 and SE-Net2 is that the speaker embedding extracted from Step1 concatenates with the output of one of the bottle layers of SE-Net2. Most of the parameters are trained in Step1 and fine-tuned in Step2.

Secondly, SR-Net1 and SR-Net2 share parameters. It is trained in Step1, and fixed in Step2. The training in Step2 only fine-tunes the parameters in SE-Net2. The reason to fix the parameters of SR-Net2 and only train SE-Net2 in Step2 is to make the model focus more on the speaker-dependent denoising process. The SR-Net2 in Step2 can be viewed as a pre-trained classifier to evaluate the performance of the speaker dependent noise reduction process in SE-Net2. As a result, in Step2, SE-Net2 is trained using the loss function that contains both the reconstruction loss and the classification loss (as shown in Equation 4.5). The input to SE-Net2 is the noisy input together with the pre-trained speaker embedding from Step1.

A similar training strategy in Step2 can be found in the VoiceID Loss approach (Shon et al. 2019) and was discussed in Section 2.3.3. The speech enhancement model and the pre-trained speaker recognition model are integrated and are then trained while the speaker recognition model is fixed. One of the key differences between this proposal and the work in Shon et al. (2019) is the use of both reconstruction loss and classification loss functions to train the model. VoiceID Loss only used the classification loss function. By way of comparison, the VoiceID Loss model is used here as a baseline, referred to as VoiceID that previously appeared in Section 4.3 and will soon be discussed further in Section 4.4.2.

Model Architecture

Figure 4.4 shows the architecture of the speech enhancement model (SE-Net2) in Step2, which is the same as in Step1, except for the concatenation of the speaker embeddings. It is based on the structure of a convolutional residual auto-encoders that was proposed for image processing by Dong et al. (2018) and applied to speech enhancement by Pandey & Wang (2019). The residual auto-encoder is used so that the model is better

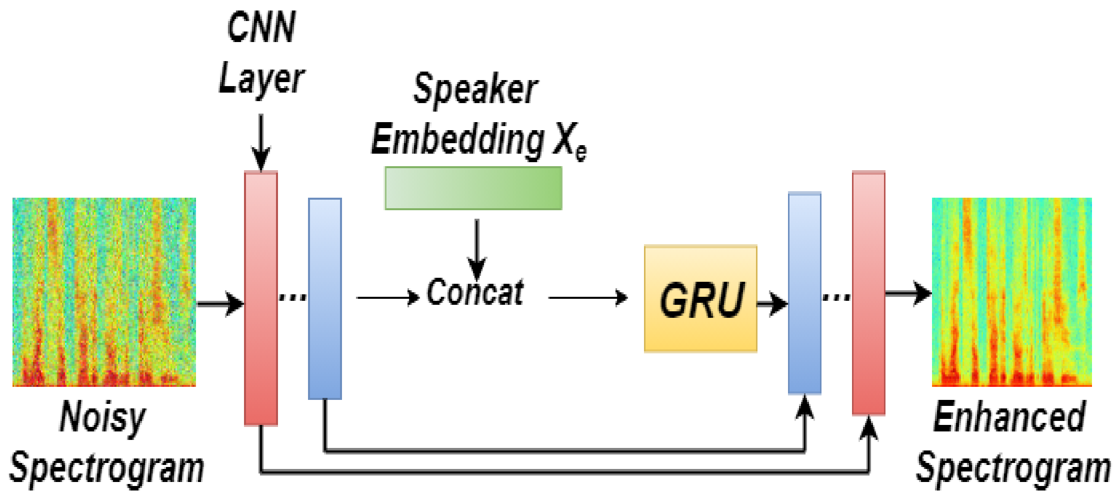


Figure 4.4: *Structure of the speech enhancement model, which is built on a residual/skip auto-encoder networks and is used in both Step1 and Step2. The speaker embedding is used only in SE-Net2.*

able to reconstruct the clean signal. As discussed in Section 2.3.3, the standard auto-encoder can denoise the input by compressing the input and then reconstructing it, but while this compression process can make the model discard the noise information and retain the most relevant information for reconstruction, it can lead to useful features being discarded. This, in turn, can make it difficult for the model to distinguish and separate some of the speaker related information and the noise interference. The residual connection provides a shortcut connection from the encoder to the decoder, allowing the decoder to capture more useful speaker related information, not only from the compressed bottleneck layer but also directly from the encoder.

The GRU layer is used to capture more contextual features through the capture of both long term and short term information while, at the same time, reducing computational costs (as discussed in Section 2.1.3).

4.4.2 Experiments

The datasets, the data processing and the experimental setup here are identical to those in Section 4.3. This is to make comparison more straightforward.

In this section, two baselines and two configurations of the proposed approach were tested:

SR is the baseline method using only the speaker recognition model (SR-Net1) without any pre-processing or post-processing.

VoiceID is the baseline published in Shon et al. (2019), where the speech enhancement and speaker recognition models are cascaded, but without joint training and without the use of speaker embeddings.

SESR-Step1 is the proposed model where the SE-Net1 and SR-Net1 models are jointly trained, but speaker embedding vectors are not being used. This is equivalent to the joint training approach that introduced in Section 4.4.

SESR-Step2 : is the proposed model where the SE-Net2 and SR-Net2 models are jointly trained with the learned speaker embedding vector being used in SE-Net2.

As in Section 4.3, the use of the SR and VoiceID models is to compare the proposed model with one that does not use a speech enhancement model and does not use a joint training strategy.

The proposed model SESR-Step1 is the Step1 that uses a speaker independent speech enhancement model in the joint training framework, while SESR-Step2 is the speaker dependent speech enhancement and the speaker recognition model. The comparison with these two models is to evaluate the effectiveness of the pre-trained speaker embedding used in Step2. Furthermore, SESR-Step1 used the same training strategy as that shown in Section 4.3, the comparison of SESR-Step2 with SESR-Step1 can also be viewed as a comparison of the proposed model in this section and that proposed in Section 4.3.

In addition, in order to explore the noise robustness of the speaker dependent speech enhancement method, the quality of the enhanced signal is also assessed using the Perceptual Evaluation of Speech Quality (PESQ) (Rix et al. 2001) and the Short-Time Objective Intelligibility (STOI) (Taal et al. 2010) scores. PESQ is one of the most widely used methods for evaluating speech quality. It measures the difference between the enhanced signal and the reference clean signal by mapping the differences in the acoustic parameters of the two signals into a PESQ score whose value is between -0.5 and 4.5. A higher score means the higher the quality. STOI is a state-of-the-art evaluation technique for speech intelligibility that is computed by linear correlation of speech temporal envelopes. The STOI score is between 0 and 1, and again, higher is better.

4.4.3 Results and Discussion

Noise Robustness

Table 4.5 shows the speaker identification performances obtained using the two baselines (SR and VoiceID) and the two proposed approaches (SESR-Step1, SESR-Step2). It is clear that the two proposed methods can yield better performance than the baselines under various noise conditions, even if the SNR is 0dB. Moreover, after using the speaker information learned by Step1, the proposed SESR-Step2 approach can further improve the identification performance (91.1%) in comparison with SESR-Step1 (90.2%). These improvements can be put down to two factors, the first of which is the use of speech enhancement before speaker identification and a joint optimisation, by which some noise interference might be filtered out. The second factor is the implementation of the speaker dependent speech enhancement in Step2. Unlike speaker-independent speech enhancement, the use of speaker information can not only recover the noise-corrupted speech signals to some extent, but may also highlight speaker-specific features which may turn out to be the key to subsequent speaker recognition.

Table 4.6 shows the speaker verification performance obtained using the four differ-

Table 4.5: *Comparison of Top-1 (red) and Top-5 (yellow) accuracies % obtained using four different methods (SR, VoiceID, SESR-Step1 and SESR-Step2) under various noise conditions.*

Noise Type	**SNR**	**SR**		**VoidID**		**SESR-Step1**		**SESR-Step2**	
		Top1	Top5	Top1	Top5	Top1	Top5	Top1	Top5
General	0	74.1	86.9	75.6	88.0	76.4	88.9	**77.4**	**89.4**
	5	79.2	90.0	80.4	90.8	81.8	91.2	**83.6**	**91.6**
	10	83.2	93.2	84.7	94.3	85.4	94.7	**87.1**	**95.7**
	15	84.9	94.6	85.6	95.1	86.3	95.8	**88.7**	**95.9**
	20	87.9	95.4	88.7	96.0	89.5	96.4	**90.3**	**96.8**
Music	0	65.8	82.0	67.1	83.3	69.2	85.2	**70.4**	**85.8**
	5	76.9	89.1	78.2	89.9	80.1	**90.6**	**81.3**	**90.6**
	10	83.8	93.5	84.6	94.2	85.9	95.1	85.9	**95.5**
	15	86.1	93.9	87.3	95.0	88.4	95.7	**89.0**	**96.3**
	20	87.4	94.7	88.9	95.6	89.2	96.6	**90.4**	**97.0**
Babble	0	62.4	80.2	63.8	82.1	65.7	83.5	**66.6**	**83.9**
	5	76.2	87.3	77.6	88.7	79.4	89.1	**80.1**	**90.3**
	10	81.4	92.2	82.3	93.5	84.0	**94.9**	**84.8**	94.5
	15	84.0	92.6	86.1	94.0	87.2	95.2	**89.1**	**95.7**
	20	85.8	92.9	86.6	95.1	88.4	95.7	**90.3**	**96.2**
Original		88.5	95.9	89.7	96.4	90.2	96.8	**91.1**	**97.7**

ent methods. Similar to Table 4.5, SESR-Step2 achieves the best results under most conditions. When evaluating the verification performance, any other techniques that are designed to reduce the intra-class distance, such as AM-softmax (discussed in Section 2.2.2), were not employed. This might be the reason that the improvement of using SESR-Step2 over SESR-Step1 on the speaker verification task is relative small.

Speech Quality Assessment of the Enhanced Signal

While evaluation using speaker recognition metrics is important, so is the quality of the enhanced speech. In the proposed joint training framework, the speech enhancement model provides the enhanced signal to the speaker recognition model, which is therefore highly dependent on the quality of the incoming, enhanced signal.

Tables 4.7 and 4.8 show the speech enhancement performance evaluated using PESQ

Table 4.6: *Comparison of speaker verification EERs (%, blue) and DCFs (green) obtained using four different methods (SR, VoiceID, SESR-Step1 and SESR-Step2) under various noise conditions.*

Noise Type	**SNR**	**SR**		**VoidID**		**SESR-Step1**		**SESR-Step2**	
		EER	minDCF	EER	minDCF	EER	minDCF	EER	minDCF
General	0	16.94	0.993	16.56	0.938	16.02	**0.885**	**15.89**	0.886
	5	12.48	0.855	12.26	0.830	11.87	0.794	**11.83**	0.**786**
	10	10.03	0.760	9.86	0.747	9.21	0.695	**9.17**	0.695
	15	8.84	0.648	8.69	0.686	8.18	0.625	**7.99**	**0.616**
	20	7.96	0.594	7.83	0.639	7.06	0.590	**6.85**	**0.589**
Music	0	17.04	0.940	16.24	0.913	**15.69**	**0.893**	15.70	0.904
	5	11.54	0.828	11.44	0.818	10.88	**0.754**	**10.78**	0.770
	10	9.69	0.749	9.13	0.733	**8.76**	**0.690**	8.94	0.704
	15	8.40	0.689	8.10	0.677	7.81	0.631	**7.65**	**0.621**
	20	7.70	0.665	7.48	0.635	7.09	0.606	**7.03**	**0.592**
Babble	0	38.90	1.000	37.96	1.000	**37.18**	0.999	37.52	**0.994**
	5	28.04	0.998	27.12	0.996	26.84	0.991	**26.69**	0.991
	10	17.34	0.917	16.66	0.926	**16.38**	**0.878**	16.93	0.901
	15	11.31	0.795	11.25	0.807	10.87	0.781	**10.84**	**0.780**
	20	9.12	0.720	8.99	0.705	8.76	**0.679**	**8.72**	0.685
Original		6.92	0.565	6.79	0.574	6.52	0.548	**6.48**	**0.537**

and STOI, respectively. The second column in both tables shows the quality of input speech corrupted by music noise at five different SNR levels. The third column indicates the obtained speech quality after using VoiceID. Of the two proposed approaches, SESR-Step2 shows clear advantages over VoiceID and SESR-Step1 under various noise conditions, reached 1.90 PESQ and 0.63 STOI scores when the SNR is 0dB.

An interesting observation is that the speech quality obtained by the VoiceID is lower than that obtained by the noisy speech. The reason, as discussed in Section 4.4.1, is that the VoiceID model only takes the classification loss as the training target when training the joint model while in the proposed model, both the reconstruction loss and the classification loss are used to train the joint system (Shon et al. 2019). From the speech quality assessment results in Tables 4.7 and 4.8, and the speaker recognition

Table 4.7: *Comparison of PESQ scores obtained using the proposed approaches and baselines under various music noise conditions.*

SNR	Noisy	VoiceID	SESR-Step1	SESR-Step2
0	1.53	1.48	1.62	1.90
5	1.78	1.72	1.89	2.14
10	1.86	1.83	1.97	2.35
15	2.16	2.06	2.21	2.58
20	2.39	2.20	2.53	2.89

Table 4.8: *Comparison of STOI scores obtained using the proposed approaches and baselines under various music noise conditions.*

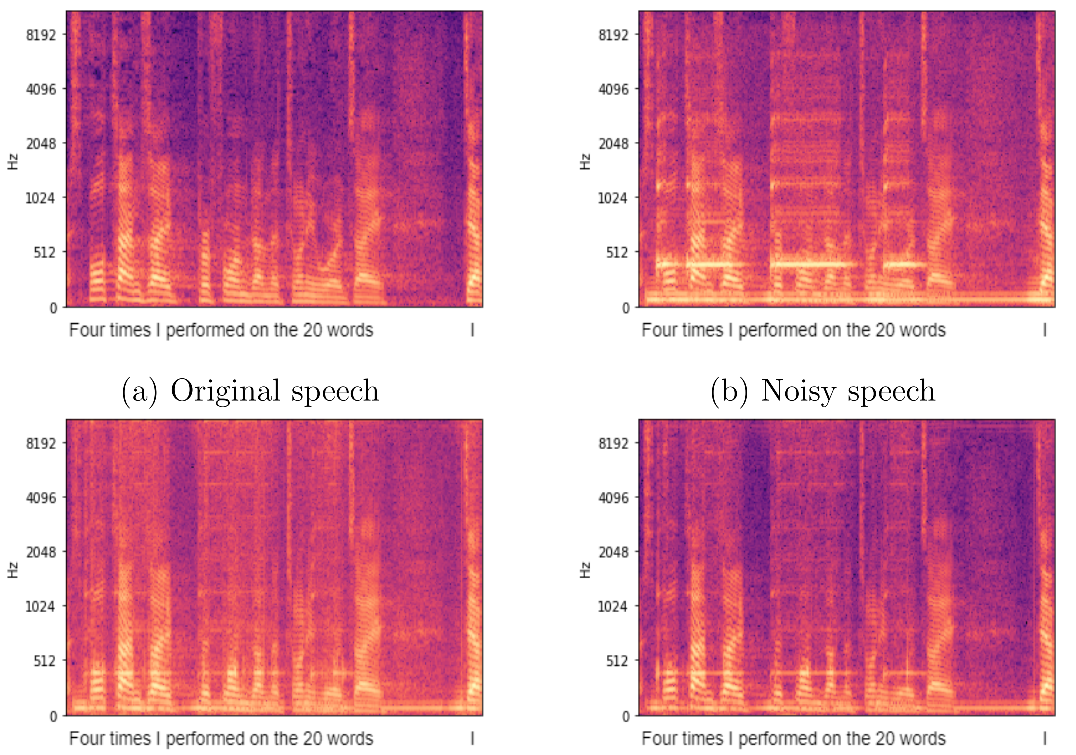
SNR	Noisy	VoiceID	SESR-Step1	SESR-Step2
0	0.53	0.50	0.56	0.63
5	0.60	0.58	0.64	0.71
10	0.65	0.61	0.67	0.75
15	0.67	0.62	0.69	0.77
20	0.68	0.64	0.70	0.78

results in Tables 4.5 and 4.6, the proposed SESR-Step2 model performs better for the speaker recognition task and obtains a better quality enhanced signal which is due to the pre-trained speaker embedding that help the model to better focus on the specific target speakers when inputting a noisy spectrogram. The fact that the joint training system uses both the reconstruction and classification losses also boosts the quality of the enhanced signal and delivers more robust speaker embeddings.

4.4.4 Analysis

An Illustration of the Enhanced Signal

In order to intuitively see the enhanced signal, and to further verify the robustness of the proposed approach with regard to noise interference, Figure 4.5 shows four spectrograms;



(a) Original speech (b) Noisy speech

(c) Enhanced speech using SE-Net1 (d) Enhanced speech using SE-Net2

Figure 4.5: *Comparison of spectrograms (a) original speech; (b) speech corrupted by noise; (c) enhanced spectrogram obtained using the SE-Net1; (d) enhanced spectrogram obtained using the SE-Net2.*

- the original speech;
- speech corrupted by music noise at 0dB;
- enhanced speech obtained using SESR-Step1;
- enhanced speech obtained using SESR-Step2.

It can be observed from both Figures 4.5(c) and 4.5(d) that the music noise can be removed from the spectrograms to a certain extent after using speech enhancement, and that the two joint training models developed in this chapter clearly reduce the noise impact. From Figures 4.5(a) and 4.5(b), it is clear that the noise contains both fluctuating and steady components that was discussed in Section 2.3.1. Both of the

SESR-Step1 and SESR-Step2 can reduce the influence. Especially the noises from the high frequency part can be remarkably reduced.

From the spectrogram shown in Figures 4.5(c) and 4.5(a), it is obvious that the some speaker information are corrupted by SESR-Step1. This may because the speaker independent mode of the joint system does not have the information about the target speaker. This makes the model difficult to distinguish the information between the noise and the target speaker. While the spectrogram shown in Figure 4.5(d) is close to the original one shown in Figure 4.5(a). Moreover, the spectrogram in Figure 4.5(d) is clearer than that in Figure 4.5(c). This may due to the pre-trained speaker embedding helps the model to focus on the specific target speakers. The speaker embeddings provide strong prior information about the target speaker, which makes the model emphasis the information of the target speaker thus deliver better noise reduction performance.

Comparison of State-of-the-art Methods

The previously developed model in Section 4.3 shows the effectiveness of the joint training framework and the multi-stage attention. The work in Chapter 3 showed the effectiveness of the AM-softmax to reduce the intra-class distance of the speaker embeddings to improve the performance of the speaker verification task.

The experiments in this section shows that speaker dependent speech enhancement can further improve the performance of the joint system. Here, all of the advantages of the developed models are put together and compared with the state-of-the-art models in the literature.

Table 4.9 shows the comparison of the SESR-Step2 with the state-of-the-art models that were trained using the Voxceleb1 dataset and evaluated using the Voxceleb1 test set. Other than the model architectures discussed in Section 4.4.1, two speech enhancement models make use of the multi-stage attention model, with channel attention coming first in the order (C-F-T, as discussed in Section 4.3.4). The model is trained

Table 4.9: *Comparison of the speaker dependent speech enhancement approach (SESR-Step2) with the state-of-the-art with the Voxceleb1 test set, using Voxceleb1 for training only.*

	Model	Loss	EER (%)
Nagrani et al. (2017)	VGG-M	softmax	10.2
Nagrani et al. (2017)	VGG-M	softmax+contrastive	7.8
Shon et al. (2019)	CNN+TDNN	softmax	6.79
Cai et al. (2018).	ResNet-34	A-softmax+PLDA	4.46
Hajibabaei & Dai (2018)	ResNet-20	A-softmax	4.40
Hajibabaei & Dai (2018)	RetNet-20	AM-softmax	4.30
Chapter 3	H-vector	AM-softmax	4.28
Chapter 4	SESR-Step2	AM-softmax	4.15

using AM-softmax instead of standard softmax to reduce the intra-class distance of the speaker embeddings. The results show that the proposed speaker dependent speech enhancement model can out-perform the state-of-the-art models, as well as the H-vector model that was proposed in Chapter 3. This clearly demonstrates that when the three proposed elements – the multi-stage attention, the joint training framework and the speaker dependent speech enhancement, and the AM-softmax function – are combined to work together, they collectively deliver better performance that the current state-of-the-art models.

4.4.5 Summary

In this section, a speaker dependent speech enhancement model was proposed, based on the developed joint training framework in Section 4.3. In Section 4.4, the proposed method was divided into two steps:

- Step1 is based on the developed joint training framework, and the purpose is to generate the speaker embeddings for each input signal.
- Step2 has the pre-trained speaker embeddings used as prior knowledge into the

speech enhancement model to better focus on the target speakers when reducing the noise impact.

The architecture of the speech enhancement model makes use of the residual auto-encoder architecture, which can build a shortcut connection between the decoder layers and the corresponding encoder layers that means the decoder retains more speaker related features when predicting a clean signal from a noisy input signal.

The experimental setup in Section 4.4.2 shows the experiments were done with the augmented Voxceleb1 dataset. The results in Section 4.4.3 show the proposed model can deliver better noise robustness compared with the baselines, especially the baseline that does not use the speaker dependent speech enhancement scenario. This indicates that the speaker dependent mode of the speech enhancement model helps to deliver better performance. Further experiments were conducted, designed to evaluate the quality of the enhanced signal, and the results show that the proposed models both deliver a higher quality enhanced signal than either the noisy input or the baseline that does not employ the joint training strategy. This indicates that the proposed model not only improves the performance of the speaker recognition task in noisy conditions, but also can improve the quality of the noisy spectrogram. Furthermore, although the SESR-Step2 contains two speech enhancement models and two speaker recognition models, the two speaker recognition models are the same in the two steps, it is trained in Step1 and fixed at step2. The speech enhancement model in Step2 shares most of the parameters, the only difference is that it concatenates the pre-trained speaker embeddings. As a result, the number of parameters of SESR-Step2 is similar to that of SESR-Step1.

In Section 4.4.4, the denoised spectrograms are shown and indicates the speaker dependent mode of the speech enhancement model can reduce more noise impact and get closer to the clean signal. Finally, the multi-stage attention, joint training strategy, speaker dependent speech enhancement and AM-softmax are combined in one model that can out-perform the state-of-the-art models in the literature, reached 4.15% EER using the same training and test dataset.

4.5 Conclusion

Speech enhancement is one of the potential ways to overcome noise interference in the speech signal and improve the noise robustness of the speaker recognition models. However, as discussed in Section 2.3.3. there is the mismatch between the independently trained speech enhancement model and the speaker recognition model, where the features learned by the former are not a good match for the input of the latter. This is due to the different target functions and the separate training strategy.

In order to address the mismatch problem, this chapter firstly proposed a joint training framework in Section 4.2. The joint training framework trains the speech enhancement and speaker recognition models with one loss function. The loss function contains the reconstruction loss, which makes the joint system better reduce noise, and the classification loss that helps the joint system retain speaker related features at the same time.

In order to implement the joint training framework, Section 4.3 discussed the architecture of the joint training system. The speech enhancement model consists of multiple dilated convolutional layers that can capture more local features from the noisy input spectrogram. The ResNet-20 architecture is used as the speaker recognition model. As discussed in Section 2.2.4, the ResNet architecture is widely used in speaker recognition. It has shown good performance in several studies and it is easier to compare with other models.

The other key component is the novel multi-stage attention mechanism. As discussed in Section 2.3.2, the attention mechanism can also reduce the noise impact by allocating weights to different parts or regions of the input data. In order to achieve higher robustness, the attention mechanism is computed in time, frequency and channel dimensions in order to filter out the noise impact in all dimensions from the output feature map of a CNN model. Both speech enhancement and speaker recognition models contain the multi-stage attention mechanism.

The experimental results in Section 4.3.3 show the joint system can deliver better robustness compared to the baseline that is not using the speech enhancement model and that is not using the joint training strategy. The proposed joint training system can obtain 2% to 3% relative improvement compared to those baselines. When using the multi-stage attention mechanism in either the speech enhancement or speaker recognition models, an additional improvement can be obtained compared with the baseline without the multi-stage attention mechanism. The results obtained also indicate the attention used in the speech enhancement model contributes more to the final prediction. This is because the speech enhancement model is close to the noisy input, so the attention mechanism can filter out more noise interference in the speech enhancement model.

The additional experiments in Section 4.3.4 investigate the effectiveness of the time, frequency and channel attention and the order in which they appear. The results show that the channel attention contributes more to the final prediction, and that the performance of the frequency attention is lower than that of the channel attention but higher than that of the time attention. This suggests the most noise information resides in the channel dimension, followed by the frequency dimension, with the least in the time dimension. This is due to the computational process of the two-dimensional CNN that learns and represents features into the channel dimension. The conclusion from this work is that the most effective ordering of the three attention mechanisms follows the pattern of most- to least-important dimensions, i.e., channel attention first, then frequency attention, and finally time attention.

In order to further improve the noise robustness of the proposed model, Section 4.4 proposed a second joint system called the speaker dependent speech enhancement model, based on the joint system proposed in Section 4.3. It can be divided into two steps. Step1 is similar to the joint system in that a speech enhancement model (SE-Net1) and a speaker recognition model (SR-Net1) are jointly trained to generate the speaker embedding of the input signal. Then, in Step2, the generated speaker embedding is used as prior knowledge into the speech enhancement model (SE-Net2) to help it better focus on the target speaker. Step2 also contains a speaker recognition model (SR-Net2), which is identical to SR-Net1, shares weights and is fixed when training in Step2. SE-Net2 has a similar architecture to SE-Net1 but it takes the speaker embedding as input. The speech enhancement models in this step make use of the residual auto-encoder architecture to generate a better enhanced signal. The experimental results in Section 4.4.3 show the speaker dependent speech enhancement system not only delivers better speaker recognition performance, but also delivers a better quality enhanced signal when compared to the enhanced signal after only Step1. The visualisation of the enhanced spectrograms verify this effect.

Finally, the joint training strategy, the multi-stage attention (developed in Section 4.3), the speaker dependent speech enhancement (developed in Section 4.4) and the AM-softmax (discussed in Section 2.4 and used in Section 3.4) are combined. The results in Section 4.4.4 show that the combination of these developed techniques can out-perform the state-of-the-art models using Voxceleb1 as the training set, reaching an EER of 4.15% on the test set of Voxceleb1.

Chapter 5

Embedding De-Mixing In a Two-Speaker Signal

5.1 Introduction

Section 2.4.1 discussed the speaker interference in the speech signals, and particularly the way in which interference can compete the same frequency band with the target speaker to make it difficult for the back-end speaker recognition model to distinguish between the target speaker and the interfering speaker. Section 2.4.2 discussed a solution called target speaker extraction, where the aim is to isolate the target voice in a mixed signal. A typical workflow for the target speaker extraction approach was discussed, where a speaker embedding extractor is pre-trained using the clean signals, the clean speaker embeddings are obtained and the target speaker extraction network then takes the mixed speech signal and the pre-trained embedding as the input, outputting the clean signal or spectrogram that contains the target speaker only. The separated clean signal can then be used for the back-end speaker recognition model.

One key property of the target speaker extraction approach is that the target speaker's voice is separated in signal space. There may be another option to separate the interfering speaker in embedding space, which may have several benefits. The embedding is low dimensional and it can project variable length acoustic signals into fixed length embeddings. It has shown in Wang et al. (2019) that the embedding contains the information of the target speaker and it can help the separation network to better filter out the interfering speaker and remain the useful information. Secondly, when separating the information of the target speaker in embedding space, the back-end model (e.g. speaker recognition model) can directly learn and recognize the speaker properties from the estimated embeddings, rather than from the reconstructed signals. This may make the back-end model easier to train.

The separation process that extracts the target information from a mixed signal in embedding space is further referred to as the embedding de-mixing process. Judging by the extant literature, this approach remains undeveloped. In this chapter, a novel embedding de-mixing approach is proposed that contains two aspects: a filtering process and an extraction process. For the filtering process, the model can use a pre-trained speaker embedding to filter out the influence of that speaker in a two-speaker signal. For example, when the embedding of one speaker is obtained, the model can filter out the influence of this speaker and output the embedding of the other speaker. The filtering process can also be applied using the spoken content embeddings, where the model can filter out the influence of the spoken content from one speaker by the content embedding. In this case, the output is the spoken content embedding of the other speaker.

For the extraction process, the model can use a pre-trained speaker embedding to extract the spoken content embedding for the same speaker. This process can also be applied inversely, where the model can use a pre-trained spoken content embedding to extract the speaker embedding.

Based on the filtering process and the extraction process above, the proposed embedding de-mixing approach offers three scenarios;

- The speaker embedding de-mixing scenario: the model removes the influence of one speaker using the speaker embedding for this speaker, so that the output is the embedding of the other speaker;

- The content embedding de-mixing scenario: the model removes the content that is spoken by one speaker using the corresponding content embedding;
- The speaker and content embedding de-mixing scenario: the model extracts one speaker embedding by the content embedding of the same speaker, or inversely, one content embedding can be obtained by its speaker embedding.

5.1.1 Chapter Outline

The remainder of this chapter is organised as follows:

- Section 5.2 gives an overall description of the embedding de-mixing approach, including the definition of the embedding de-mixing functions.
- Section 5.3 discusses the detailed model architecture and implementations of the speaker embedding de-mixing step. The experimental results and discussion are also provided.
- Section 5.4 discusses the detailed model architecture and experimental results of the content embedding de-mixing step.
- Section 5.5 discusses the speaker and content embedding de-mixing step.
- Section 5.6 provides a conclusion.

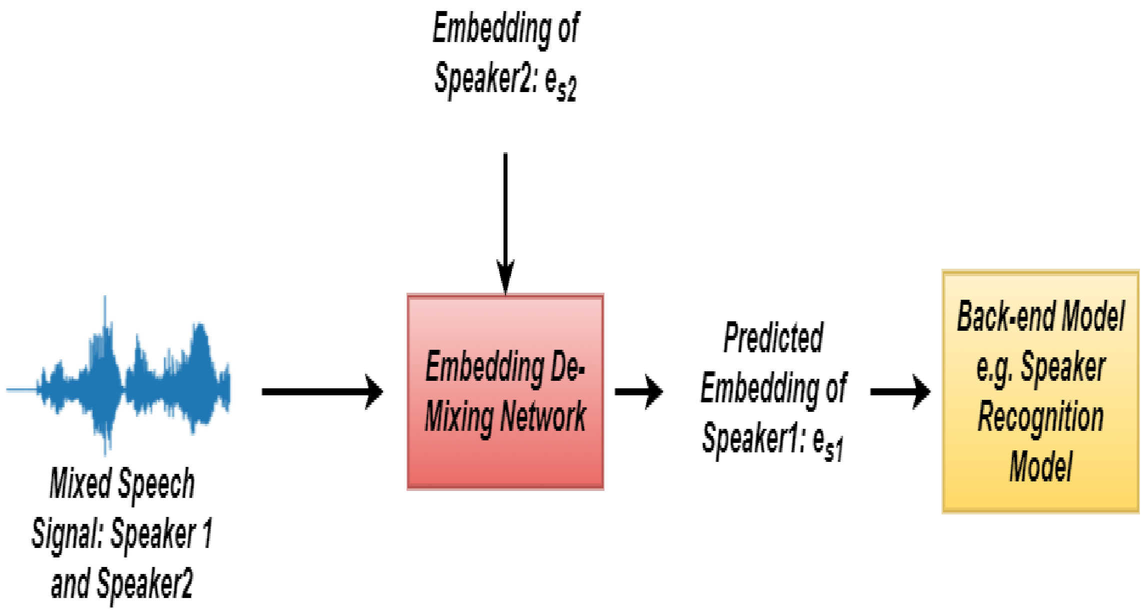

Embedding of Speaker2: e_{s2}
Mixed Speech Signal: Speaker 1 and Speaker2
Embedding De-Mixing Network
Predicted Embedding of Speaker1: e_{s1}
Back-end Model e.g. Speaker Recognition Model

Figure 5.1: *The diagram of the embedding de-mixing approach.*

5.2 The Embedding De-Mixing Approach

In this section, the structure of the embedding de-mixing approach is introduced. Figure 5.1 shows an example of the embedding de-mixing process. Suppose the input mixed speech signal $\boldsymbol{X}_{mix}$ contains two speakers, and the information from each speaker includes the properties of that speaker, and the spoken content from that speaker. When representing this information in embedding space, the speaker embeddings are referred to as $\boldsymbol{e}_{s1}$ and $\boldsymbol{e}_{s2}$. The content embeddings are represented as $\boldsymbol{e}_{c1}$ and $\boldsymbol{e}_{c2}$ in embedding space. The term $\boldsymbol{e}_{mix}$ represents the embedding of the mixed signal, which is further referred to as the mixed embedding. The terms Speaker1 and Speaker2 represent the two speakers; Speaker1 is viewed as the target speaker, while Speaker2 represents the interfering speaker.

In Figure 5.1, the mixed signal $\boldsymbol{X}_{mix}$ is input to an embedding de-mixing network that also takes the embedding of Speaker2 ($\boldsymbol{e}_{s2}$) as input. The output of the network is the predicted speaker embedding of Speaker1 ($\boldsymbol{e}^{'}_{s1}$). The obtained embedding $\boldsymbol{e}^{'}_{s1}$ can be used by a back-end speaker recognition model. In this example, the embedding de-mixing network filters out the influence of Speaker2 on $\boldsymbol{e}_{mix}$ using $\boldsymbol{e}_{s2}$. The output is assumed to contain only the information of Speaker1. This is the speaker embedding de-mixing mentioned in Section 5.1.

The content embedding de-mixing is similar to the speaker embedding de-mixing that was introduced above. The de-mixing network takes the content embedding of Speaker2 ($\boldsymbol{e}_{c2}$) and the mixed speech signal as input, and the output is the predicted clean content embedding $\boldsymbol{e}^{'}_{c1}$. In this way, the spoken content information from one speaker in the mixed speech signal is filtered by the de-mixing network using the corresponding content embedding. In this thesis, the spoken content denotes the word or phonemes spoken by the speaker. The content embedding refers to the spoken content that represented in embedding space.

The third scenario, the speaker and content embedding de-mixing, is different from the above two scenarios. In this scenario, the speaker embedding $\boldsymbol{e}_{s1}$ and the mixed signal are provided to the embedding de-mixing network. The desired output is the corresponding content embedding $\boldsymbol{e}_{c1}$. In this way, the embedidng de-mixing network extracts $\boldsymbol{e}_{c1}$ using $\boldsymbol{e}_{s1}$ from the mixed signal. The de-mixing network can also extract the speaker embedding using the content embedding, where input can be the content embedding $\boldsymbol{e}_{c1}$ and the mixed signal, and the output is the corresponding speaker embedding $\boldsymbol{e}_{s1}$.

Figure 5.2 shows an example of the work flow of the embedding de-mixing approach. Similar to the Figure 5.1, it also uses the speaker embedding de-mixing scenario as the example. The embedding de-mixing network consists of three parts; the speaker recognition model that is used for obtaining the clean embeddings, the embedding extractor that converts the mixed speech signal into the embedding space. The output is the mixed embedding $\boldsymbol{e}_{mix}$. The final component is the de-mixing function, as shown in Equation 5.1.

$$\boldsymbol{e}^{'}_{s1} = f_{demix}(\boldsymbol{e}_{mix}, \boldsymbol{e}_{s2}) \tag{5.1}$$

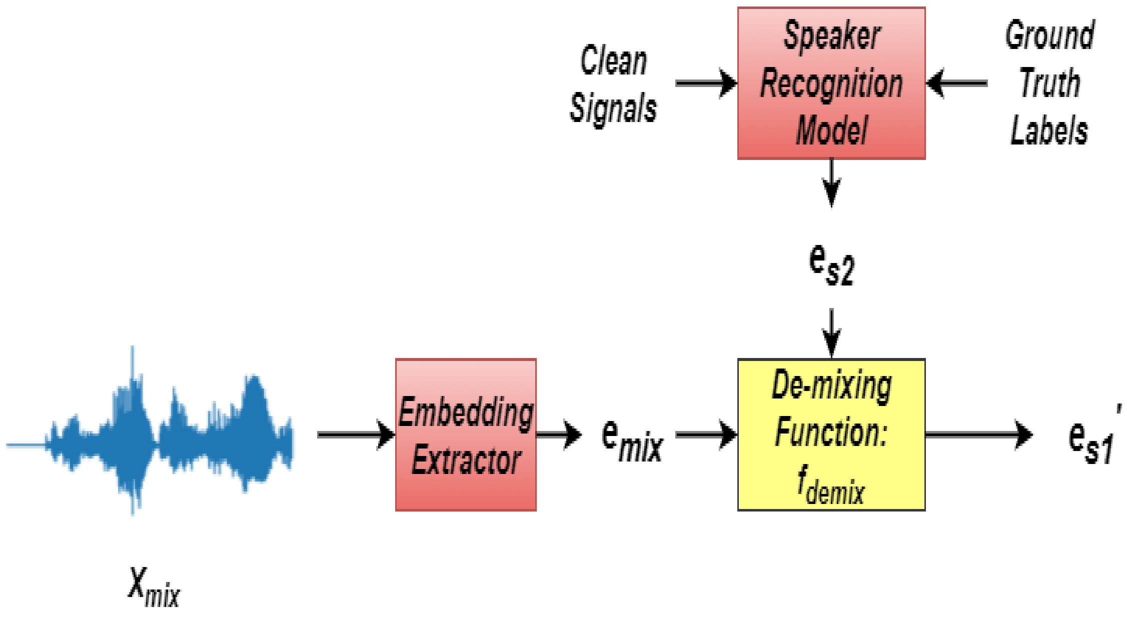


Figure 5.2: *The work flow of the embedding de-mixing approach.*

Equation 5.1 takes $\boldsymbol{e}_{mix}$ and the pre-trained clean embedding $\boldsymbol{e}_{s2}$ as input and outputs the predicted embedding $\boldsymbol{e}'_{s1}$. The embedding de-mixing model is trained in a supervised manner, which uses the reconstruction loss of the predicted embedding $\boldsymbol{e}'_{s1}$ and the reference clean embedding $\boldsymbol{e}_{s1}$ to form the training target (Willmott & Matsuura 2005).

The embedding de-mixing process is performed by the de-mixing function $f_{demix}(.)$. The next section will introduce and discuss the different possible architectures of $f_{demix}(.)$.

5.2.1 The Embedding De-Mixing Function

The embedding de-mixing function $f_{demix}(.)$ takes the $\boldsymbol{e}_{mix}$ and one speaker embedding $\boldsymbol{e}_{s2}$ as input, the output is the estimated clean embedding $\boldsymbol{e}'_{s1}$. There are different possible choices of $f_{demix}(.)$. As there is no baseline system in the literature with which a proposed model can be compared, six possible architectures are investigated in this section. Figure 5.3 illustrates the six different implementation strategies for $f_{demix}(.)$: (a)

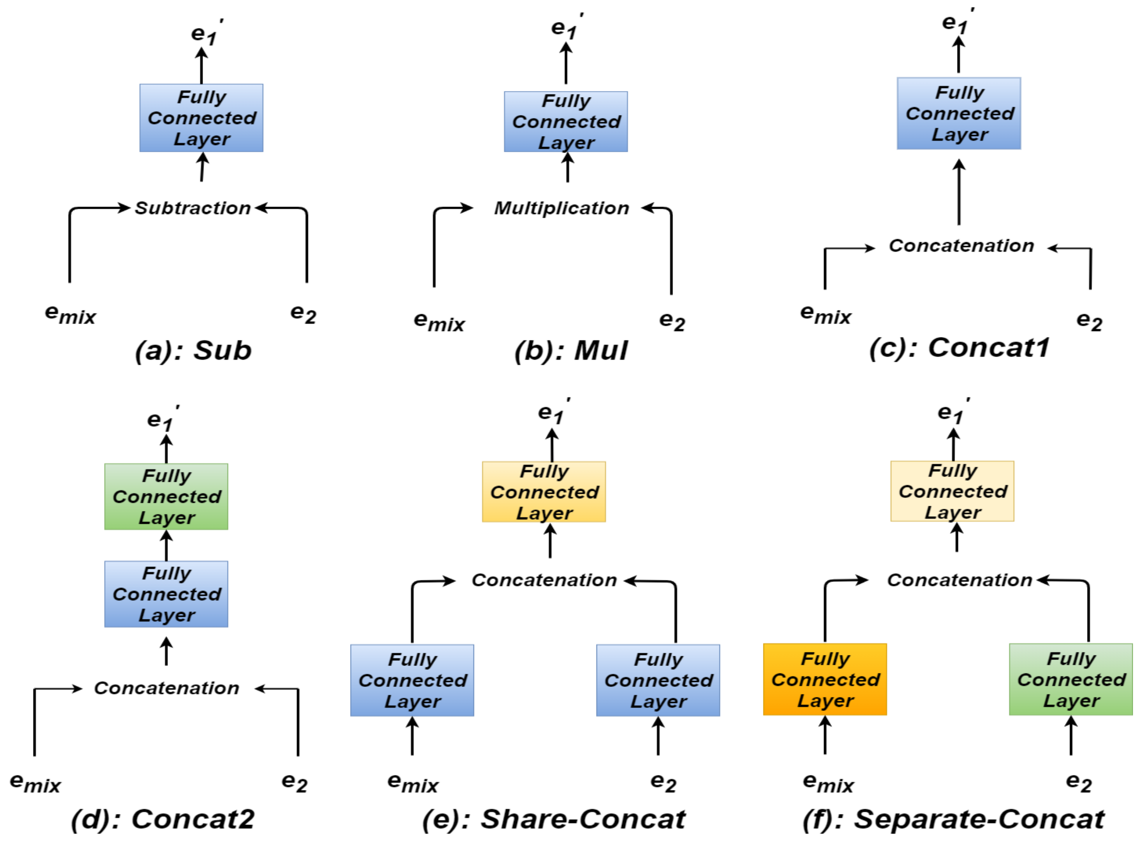


Figure 5.3: *Different implementation strategies of the de-mixing function f_{demix}: (a) subtraction; (b) multiplication; (c) concatenation with one fully-connected layer (d) concatenation with two fully-connected layers; (e) shared fully-connected layer with concatenation; (f) separated fully-connected layer with concatenation.*

subtraction (Sub); (b) multiplication (Mul); (c) concatenation with one fully-connected layer (Concat1) (d) concatenation with two fully-connected layers (Concat2); (e) shared fully-connected layer with concatenation (Share-Concat); (f) separated fully-connected layer with concatenation (Separate-Concat). In this section, the subscripts s and c are omitted for simplicity.

Subtraction

The first implementation strategy is a subtraction operation (Sub) of $\boldsymbol{e}_{mix}$ and $\boldsymbol{e}_2$, shown in Figure 5.3(a) above and Equation 5.2 below. After subtraction, the subtracted embedding vector is passed to a fully-connected layer without an activation function. That can be viewed as a linear transformation. The embedding dimension is denoted as d for all of the embeddings in this section. $\boldsymbol{W} \in \mathcal{R}^{d \times d}$ and $\boldsymbol{b} \in \mathcal{R}^{1 \times d}$ are the parameters of the fully-connected layer.

$$\boldsymbol{e}_1^{'} = (\boldsymbol{e}_{mix} - \boldsymbol{e}_2)\boldsymbol{W} + \boldsymbol{b} \tag{5.2}$$

Multiplication

The multiplication approach (Mul) is similar to the Sub method except that $\boldsymbol{e}_{mix}$ is multiplied element-wise with $\boldsymbol{e}_2$. Figure 5.3(b) and Equation 5.3 show the architecture of the Mul method, where $\odot$ denotes element-wise multiplication.

$$\boldsymbol{e}_1^{'} = (\boldsymbol{e}_{mix} \odot \boldsymbol{e}_2)\boldsymbol{W} + \boldsymbol{b} \tag{5.3}$$

Concatenate With One Fully-Connected Layer

The third method is referred to as Concat1. $\boldsymbol{e}_{mix} \in \mathcal{R}^{1 \times d}$ and $\boldsymbol{e}_2 \in \mathcal{R}^{1 \times d}$ are concatenated, and then input to a fully connected layer, as shown in Figure 5.3(c) and Equation 5.4. $[\boldsymbol{e}_{mix}; \boldsymbol{e}_2] \in \mathcal{R}^{1 \times 2d}$ denotes the concatenated vector of $\boldsymbol{e}_{mix}$ and $\boldsymbol{e}_2$, $\boldsymbol{W} \in \mathcal{R}^{2d \times d}$ and $\boldsymbol{b} \in \mathcal{R}^{1 \times d}$ are the parameters for the fully connected layer, which does not have any activation function.

$$\boldsymbol{e}_1^{'} = [\boldsymbol{e}_{mix}; \boldsymbol{e}_2]\boldsymbol{W} + \boldsymbol{b} \tag{5.4}$$

Concatenate With Two Fully-Connected Layers

Concatenation with two fully-connected layers (Concat2) is similar to the Concat1 in that $\boldsymbol{e}_{mix}$ and $\boldsymbol{e}_2$ are concatenated and then fed into two fully connected layers, as shown in Figure 5.3(d) and Equation 5.5. The first fully-connected layer uses a

ReLU activation function while there is no activation function after the second layer. $\boldsymbol{W} \in \mathcal{R}^{2d\times d}$ and $\boldsymbol{b} \in \mathcal{R}^{1\times d}$ are the parameters for the fully connected layer.

$$\boldsymbol{e}_1^{'} = \text{ReLU}([\boldsymbol{e}_{mix}; \boldsymbol{e}_2]\boldsymbol{W}_0 + \boldsymbol{b}_0)\boldsymbol{W}_1 \tag{5.5}$$

Shared Fully-Connected Layer With Concatenation

These final two strategies differ from the previous four. In the fifth method, (Share-Concat), $\boldsymbol{e}_{mix}$ and $\boldsymbol{e}_2$ are input into two fully connected layers that share parameters. The outputs $\boldsymbol{k}_{mix}$ and $\boldsymbol{k}_2$ are then concatenated and passed to another fully connected layer, as shown in Figure 5.3(e) and Equation 5.6. $\boldsymbol{W}_0 \in \mathcal{R}^{d\times d}$, $\boldsymbol{b}_0 \in \mathcal{R}^{1\times d}$, $\boldsymbol{W}_1 \in \mathcal{R}^{2d\times d}$ and $\boldsymbol{b}_1 \in \mathcal{R}^{1\times d}$ are the parameters for the fully connected layers.

$$\begin{aligned} \boldsymbol{e}_1^{'} &= \text{ReLU}([\boldsymbol{k}_{mix}; \boldsymbol{k}_2]\boldsymbol{W}_1 + \boldsymbol{b}_1) \\ \boldsymbol{k}_{mix} &= \text{ReLU}(\boldsymbol{e}_{mix}\boldsymbol{W}_0 + \boldsymbol{b}_0) \\ \boldsymbol{k}_2 &= \text{ReLU}(\boldsymbol{e}_2\boldsymbol{W}_0 + \boldsymbol{b}_0) \end{aligned} \tag{5.6}$$

Separated Fully-Connected Layer With Concatenation

The final strategy, Separate-Concat, is similar to Share-Concat, in that $\boldsymbol{e}_{mix}$ and $\boldsymbol{e}_2$ are input into two fully connected layers but, in this case, the two fully connected layers do not share parameters. The outputs $\boldsymbol{k}_{mix}$ and $\boldsymbol{k}_2$ are then concatenated and input into another fully connected layer (as shown in Figure 5.3(f) and Equation 5.7). $\boldsymbol{W}_{0,0} \in \mathcal{R}^{d\times d}$, $\boldsymbol{b}_{0,0} \in \mathcal{R}^{1\times d}$, $\boldsymbol{W}_{0,1} \in \mathcal{R}^{d\times d}$, $\boldsymbol{b}_{0,1} \in \mathcal{R}^{1\times d}$, $W_1 \in \mathcal{R}^{2d\times d}$ and $\boldsymbol{b}_1 \in \mathcal{R}^{1\times d}$ are the parameters of the fully connected layers.

$$\begin{aligned} \boldsymbol{e}_1^{'} &= \text{ReLU}([\boldsymbol{k}_{mix}; \boldsymbol{k}_2]\boldsymbol{W}_2 + \boldsymbol{b}_2) \\ \boldsymbol{k}_{mix} &= \text{ReLU}(\boldsymbol{e}_{mix}\boldsymbol{W}_{0,0} + \boldsymbol{b}_{0,0}) \\ \boldsymbol{k}_2 &= \text{ReLU}(\boldsymbol{e}_2\boldsymbol{W}_{0,1} + \boldsymbol{b}_{0,1}) \end{aligned} \tag{5.7}$$

These six embedding de-mixing functions can be classified into three categories, namely mathematical operations (Sub and Mul), concatenation before processing (Concat1 and Concat2), and concatenation after processing (Share-Concat and Separate-Concat). These will now be discussed in further detail.

The mathematical operation methods investigate whether the information in the embedding space can be controlled with a simple mathematical operation, such as subtraction or multiplication. This idea is originally from the word embeddings in natural language processing. When the word embedding of “France” is subtracted by the embedding of “Paris”, and the embedding of “Rome” is added, the result will be close to the embedding of “Italy”. This indicates mathematical operations can be applied to the semantics that is encoded in the word embeddings (Mikolov, Chen, Corrado & Dean 2013, Mikolov, Sutskever, Chen, Corrado & Dean 2013). For the embeddings from speech signals considered here, it is difficult to evaluate the semantics of different speaker properties. However, if the subtraction or multiplication strategies can benefit the de-mixing process, it shows the learned embeddings contain high level information that can be separated and controlled using a simple mathematical operation.

The methods in the second category (concatenation before processing) apply a concatenation operation before neural network layers, while those in the third category (concatenation after processing) apply a concatenation operation after processing with neural network layers. Concatenation is an information fusion technique widely used in neural networks and is employed here to fuse two input embeddings in the neural network layers. The embeddings ($\boldsymbol{e}_{mix}$ and $\boldsymbol{e}_1$) contains different information. Merging the two embeddings before or after the network layers may provide a strong prior information to the neural network, and it may affect the performance of the de-mixing process. In order to evaluate whether or not this prior information is important to the embedding de-mixing process, the two categories mentioned above are designed.

5.3 Speaker Embedding De-Mixing

As discussed in Section 5.2, the first scenario of the embedding de-mixing approach is the speaker embedding de-mixing. The embedding de-mixing network in this scenario takes the mixed embedding $\boldsymbol{e}_{mix}$ and the embedding of one speaker $\boldsymbol{e}_{s2}$ to predict the embedding of the other speaker $\boldsymbol{e}_{s1}$. This is a filtering process whereby the information of Speaker2 is removed from the mixed embedding, when the model has $\boldsymbol{e}_{s2}$ being its input. This section will introduce the detailed model architectures of the embedding de-mixing network for speaker embedding de-mixing, as well as the experimental results.

5.3.1 Model Architecture

Clean Embedding Extraction

In order to achieve the goal for the speaker embedding de-mixing process, as shown in Figure 5.2 and discussed in Section 5.2, the workflow contains three components: the speaker recogntion model to obtain the clean embeddings, the embedding extractor for converting the mixed signal into the embedding space, and the embedding de-mixing function.

In order to learn high quality and robust speaker embeddings, the speaker recognition model is designed based on the X-vector architecture which, as discussed in Section 2.2, delivers a high robustness and can better capture time-relevant information (Snyder et al. 2018). The detailed model architecture of the speaker recognition model can be found in Appendix A.4.

Embedding De-mixing

After collecting the embeddings for each speaker using the trained speaker recognition network, the embedding de-mixing network is trained. The embedding extractor takes the mixed signal as input, the output is the mixed embedding $\boldsymbol{e}_{mix}$. After obtaining the $\boldsymbol{e}_{mix}$, the embedding de-mixing function takes the clean embedding $\boldsymbol{e}_{s2}$ and the

$\boldsymbol{e}_{mix}$ as input, outputs the predicted clean embedding $\boldsymbol{e}'_{s1}$. Different implementation strategies of the de-mixing function can be found in Section 5.2.1, the detailed model architecture of the embedding extractor can be found in Appendix A.4.

As discussed in Section 5.2, the de-mixing function is trained using the reconstruction loss that measures the difference between the clean embedding $\boldsymbol{e}_{s1}$ and the predicted embedding $\boldsymbol{e}'_{s1}$ (shown in Equation 5.8) in which a reconstruction loss is used between $\boldsymbol{e}'_{s1}$ and $\boldsymbol{e}_{s1}$. In this work, the mean absolute error (MAE) is applied Willmott & Matsuura (2005), where $e_{s1,i}$ and $e'_{s1,i}$ denote the value of the ith element from $\boldsymbol{e}_{s1}$ and $\boldsymbol{e}'_{s1}$, respectively. $|.|$ denotes the absolute value.

$$\mathcal{L} = \frac{1}{d}\sum_{i=1}^{d} |e_{s1,i} - e'_{s1,i}| \tag{5.8}$$

The MAE is used here as the reconstruction loss function in order to avoid the gradient vanishing problem, discussed in Section 2.1.3, that may be created by the more widely used MSE. In the embedding de-mixing process, the difference between the predicted embedding and the reference clean embedding may be very small, and using MSE may make the difference even smaller, thereby creating the gradient vanishing issues.

5.3.2 Experimental Setup

Data Processing

In order to comprehensively test the performance of the speaker embedding de-mixing model, two datasets are used; the TIMIT dataset (Garofolo et al. 1993) and the MC-WSJ dataset (Lincoln et al. 2005). The introduction of these two datasets can be found in Section 2.5.4.

The experiments are split into two parts; speaker embedding de-mixing with artificially augmented data (TIMIT), and with the real-world dataset (MC-WSJ). As discussed in Section 3.5, it is convenient to control the energy level of the interfering speaker in an artificially augmented dataset, and it is better to observe the behaviour of the proposed model under different levels of interference. Under real-world conditions in the MC-WSJ dataset, two microphone arrays recorded two moving speakers. This situation is more difficult for the de-mixing network as the energy for each speaker changes throughout any given utterance. More details of the data processing of the augmented TIMIT dataset and the MC-WSJ dataset can be found in Appendix B.2.

Evaluation Metric

As the embedding de-mixing approach in this thesis is the first approach that separates the speaker information in embedding space rather than in signal space, there is no standard evaluation metric. For the purposes of this study, two evaluation metrics were selected; speaker identification accuracy and a cosine similarity score.

Speaker identification accuracy is obtained using an additional classier. As mentioned in Section 5.3, the speaker recognition model is trained using the clean signals. The final DNN layers are reused here as the classifier to evaluate the quality of the predicted embeddings. It takes the embedding as input and outputs the predicted speaker identity scores. A detailed model architecture can be found in Appendix A.4.

$$\mathrm{Cos}(\boldsymbol{e}_{s1}, \boldsymbol{e}_{s1}^{'}) = \frac{\boldsymbol{e}_{s1}\boldsymbol{e}_{s1}^{'}}{||\boldsymbol{e}_{s1}||||\boldsymbol{e}_{s1}^{'}||} \tag{5.9}$$

The other evaluation metric is the cosine similarity score that computes the distance between the clean embedding $\boldsymbol{e}_{s1}$ and the predicted embedding $\boldsymbol{e}_{s1}^{'}$, as shown in Equation 5.9. In contrast to speaker the identification accuracy that computed by a back-end model, the cosine similarity score directly measures the distance between two embeddings and thereby indicates the quality of the de-mixed embeddings.

Table 5.1: *The cosine similarity score and speaker identification accuracy when using the estimated embedding of target speaker $\boldsymbol{e}'_{s1}$. "Before" denotes the cosine similarity or speaker identification directly using $\boldsymbol{e}_{mix}$. "Clean" de- notes the cosine similarity or speaker identification using $\boldsymbol{e}_{s1}$ that is extracted from clean speech.*

	Cosine Similarity			**Identification Accuracy (%)**		
SNR	**-5dB**	**0dB**	**5dB**	**-5dB**	**0dB**	**5dB**
Before	0.22	0.48	0.59	36.5	58.4	72.5
Sub	**0.80**	0.82	0.84	**86.2**	89.9	95.2
Mul	0.68	0.73	0.78	83.7	88.8	94.8
Concat1	0.44	0.47	0.52	52.9	56.8	68.8
Concat2	0.51	0.55	0.60	64.5	70.3	88.5
Share-Concat	0.46	0.62	0.69	58.9	76.0	82.9
Separate-Concat	0.78	**0.86**	**0.89**	82.5	**93.0**	**96.9**
Clean	1.0			98.5		

5.3.3 Results and Discussion

The Performance on the Augmented TIMIT Dataset

The first experiment is conducted on the augmented TIMIT dataset. The interfering speakers are mixed with a known SNR level in order to better observe the performance of the six different de-mixing functions.

Table 5.1 shows the results of using $\boldsymbol{e}_{s2}$ to obtain $\boldsymbol{e}_{s1}$ from the mixed embedding emix. The cosine similarity scores and speaker identification accuracies for all of the six embedding de-mixing functions $f_{demix}(.)$ at different SNR levels are shown.

Comparing with not using $f_{demix}(.)$ (directly evaluating mixed embeddings $\boldsymbol{e}_{mix}$), most of the architectures of $f_{demix}(.)$ performed better. This shows that the speaker embedding de-mixing process removed some of the influences of the information from the interfering speakers. The Separate-Concat method delivered the best performance with SNR level of 0dB and 5dB. Even when the SNR is -5dB (The power of the interfering Speaker2 is greater than that of the target Speaker1), the Separate-Concat method still reached 82.5% identification accuracy and 0.78 cosine similarity.

The Sub method delivered the best performance when the SNR is -5dB, with a speaker identification accuracy of 86.2% and a cosine similarity score of 0.80. This shows that a simple mathematical operation and a linear transformation can be applied on the speaker embeddings to filter out some of the information of the interfering speaker. The Mul method is another mathematical operation (multiplication).

The speaker embedding de-mixing approach can be applied symmetrically, where the speaker embedding de-mixing network can use $\boldsymbol{e}_{s1}$ and $\boldsymbol{e}_{mix}$ to obtain $\boldsymbol{e}_{s2}$, which is using the embedding of the target speaker to obtain the embedding of the interfering speaker. Table 5.2 shows the experimental results. The SNR value is the signal-to-noise ratio of the target speaker (Speaker1) to the interfering speaker (Speaker2) and it is clear that, when the SNR equals to 5dB, the performance worse than when the SNR is -5dB. All the results for six methods show slightly lower performances than when using $\boldsymbol{e}_{s2}$ to reconstruct $\boldsymbol{e}_{s1}$ (shown in Table 5.1). This is due to the data construction process. When constructing artificially augmented utterances, the target speakers are fixed but the interfering speakers are randomly chosen. This means that each of the target speakers occurred a certain number of times in the training dataset, but for the interfering speakers, some speakers occur many times while some are rarely appeared. Thus, the de-mixing network cannot fit each of the interfering speakers well.

From observation of the results, it is clear that all six implementation methods for the embedding de-mixing functions delivered different performance levels. As discussed in Section 5.2.1, these six methods can be categorised into three classes: simple mathematical operation, concatenation before processing and concatenation after processing. From the results in Tables 5.1 and 5.2, the results obtained by the two simple mathematical operation methods (Sub and Mul) are better than those obtained by the mixed signals, indicating that these operations impact on the speaker properties encoded in

Table 5.2: *The cosine similarity and speaker identification accuracy when using the estimated embedding of interfering speaker $\boldsymbol{e}'_{s2}$. "Before" denotes the cosine similarity or speaker identification directly using $\boldsymbol{e}_{mix}$. "Clean" denotes the cosine similarity or speaker identification using $\boldsymbol{e}_{s2}$ extracted from clean speech.*

	Cosine Similarity			**Identification Accuracy (%)**		
SNR	**-5dB**	**0dB**	**5dB**	**-5dB**	**0dB**	**5dB**
Before	0.60	0.46	0.28	72.0	58.4	31.7
Sub	0.78	0.74	**0.72**	95.9	90.0	**87.1**
Mul	0.70	0.66	0.62	95.5	88.4	83.2
Concat1	0.45	0.42	0.38	65.1	56.0	51.7
Concat2	0.52	0.47	0.42	89.2	70.9	64.1
Share-Concat	0.65	0.53	0.47	83.7	77.0	59.5
Separate-Concat	**0.87**	**0.79**	0.70	**97.1**	**93.8**	83.6
Clean	1.0			98.5		

the embeddings, particularly the Sub method. This indicates that the overlapped information of the two speakers in the signal space can be reduced in the embeddings space.

The results obtained by the "concatenation before processing" methods (Concat1 and Concat2) are far from those obtained by the clean results, suggesting that the embedding de-mixing network needs the information of the separate inputs. As discussed in Section 5.2.1, Concat1 and Concat2 both concatenate the embeddings before feeding them into the network layers, an operation that does not provide the prior information that $\boldsymbol{e}_{mix}$ and $\boldsymbol{e}_{s2}$ are separate embeddings that contain different information to the model. This information may not be learned by the model in the training process.

The results obtained with "the concatenation after processing" methods (Share-Concat and Separate-Concat) behaved differently. The results obtained by the Share-Concat method are far from that obtained from the clean signal, while the Separate-

Concat method obtained the best results of all six methods in most of the conditions. The reason may be similar to that in the previous categories. The Share-Concat merged the two embeddings after the neural network layer, but it uses two shared layers and the prior information may still not provided to the model. Separate-Concat, on the other hand, uses separate layers to process the two embeddings, which provides the model with strong prior information that the two embeddings contain separate information. The model can learn this information well and thereby deliver the best performance.

Performance on the Real-World Dataset

The experiments with the artificially augmented dataset have shown that the speaker embedding de-mixing network can filter out the interference information in the embedding space. This experiment will evaluate whether it can also obtain similar performance under real-world conditions.

Table 5.3 shows the experimental result from microphone1 (M1) and microphone2 (M2) in the MC-WSJ dataset. Similar to the results obtained on the augmented TIMIT dataset, the Separate-Concat method obtained the best results, reaching 93.9% and 90.9% test accuracies and 0.83 and 0.80 cosine similarities with M1 and M2 respectively. The reason why the results from M2 are lower than that from M1 might be the distance between the speakers and the microphones since M1 is closer to the speakers while M2 is further away (Lincoln et al. 2005).

Compared with the results from the headset recordings, which reached a 99.1% test accuracy, the results obtained by the Separate-Concat method still show a gap because, under real-world conditions, the two speakers are moving and the SNR between the target speaker and the interfering speaker changes over time. It may be more difficult for the de-mixing network to filter out the information of the interfering speaker with fluctuating energy levels.

Table 5.3: *The cosine similarity and speaker identification results on MC-WSJ dataset in microphone M1 and M2.*

	Cosine Similarity		**Identification Accuracy (%)**	
	M1	**M2**	**M1**	**M2**
Before	0.46	0.41	52.1	47.1
Sub	0.74	0.69	87.2	83.9
Mul	0.71	0.66	84.4	82.1
Concat1	0.39	0.33	50.2	41.7
Concat2	0.64	0.60	79.1	72.4
Share-Concat	0.60	0.53	65.1	55.4
Separate-Concat	0.83	0.80	**91.3**	**90.9**
Headset	1.0		99.1	

5.3.4 Summary

In this section, a speaker embedding de-mixing network is introduced that can filter out the information of one speaker in the embedding space by the corresponding pre-trained clean speaker embedding. Specifically, it can filter out the information of the interfering speaker or the target speaker by the corresponding speaker embeddings. In Section 5.3.1 the detailed model architecture of the speaker embedding de-mixing network was introduced, which consists of the speaker recognition model, the embedding extractor and the embedding de-mixing function. The speaker recognition model is used to obtain the clean speaker embeddings. The embedding extractor projects the mixed signal into the mixed embedding $\boldsymbol{e}_{mix}$, which is then used by the embedding de-mixing function.

In order to comprehensively evaluate the performance of the speaker embedding de-mixing network, experiments were conducted using two different datasets: the artificially augmented TIMIT dataset and the MC-WSJ dataset. In Section 5.3.2, data construction process was introduced. The augmented TIMIT dataset was used to evaluate the performance of the proposed model with different levels of interference, while the MC-WSJ dataset was used to evaluate the proposed model under real-world conditions.

The results given in Section 5.3.3 demonstrate that the embedding de-mixing network is able to filter out interference using the corresponding speaker embedding. From the comparison of the different speaker de-mixing network functions, it can be observed that simple mathematical operations can be applied in the embedding space to reduce the influence of the interfering speaker.

5.4 Content Embedding De-Mixing

As Section 5.3 shows that the proposed embedding de-mixing network can filter out the information of the interfering speaker in the embedding space, it is interesting to test whether the same architecture can also separate spoken content information.

Following the architecture of the speaker embedding de-mixing network, this section introduces the content embedding de-mixing network. Similar to the de-mixing process of the speaker embeddings, the content embedding de-mixing network takes the mixed embedding $\boldsymbol{e}_{mix}$ and embedding of the spoken content of the interfering speaker $\boldsymbol{e}_{c2}$ to obtain the embedding of the spoken content of the target speaker $\boldsymbol{e}_{c1}$.

5.4.1 Model Architecture

The content embedding de-mixing network contains the same components: the content recognition network, the embedding extractor and the embedding de-mixing network.

The content recognition network has the same architecture as the speaker recognition network introduced in Section 5.3.1, this time with the goal of extracting a clean embedding of the spoken content.

The embedding extractor maps the mixed signal to the mixed embedding $\boldsymbol{e}_{mix}$. The embedding de-mixing function takes $\boldsymbol{e}_{mix}$ and $\boldsymbol{e}_{c2}$ as inputs, the output is the predicted content embedding $\boldsymbol{e}^{'}_{c1}$. The de-mixing network is trained using the reconstruction loss shown in Equation 5.10.

$$\mathcal{L} = \frac{1}{d}\sum_{i=1}^{d} |e_{c1,i} - e^{'}_{c1,i}| \tag{5.10}$$

5.4.2 Experiment Setup

The data used for separating the content information is difficult to select because while the speaker identities remain constant across the whole utterance, while the content information changes throughout each utterance. As a result, in order to use the same model architecture for the speaker embedding de-mixing network to evaluate the separation of the content information, the Speech Command dataset (Warden 2018), a dataset containing isolated spoken English words, was selected. As introduced in Section 2.5.5, each utterance contains one isolated word, and the duration of each is one second. There are 35 unique words, spoken by more than 2,000 different people. For each isolated word, there are more than 1,000 utterances on average. The details for this dataset can be found in Appendix B.3.

As a result, the content in this section denotes the spoken words in the input utterance. The model is trained to predict the spoken word for each input signal. The extracted embeddings are assumed to contain the information of the spoken words.

Apart from the data selected, the experimental setup of the content embedding de-mixing network is the same as that for the speaker embedding de-mixing network discussed in Section 5.3.2.

5.4.3 Results and Discussion

Table 5.4 shows the spoken word recognition accuracy using the estimated content embedding $\boldsymbol{e}'_{c1}$. For simplicity, the spoken content of the interfering speaker is further referred to as the interfering content, and the spoken content of the target speaker is further referred to as the target content. In this scenario, The embedding of the interfering content $\boldsymbol{e}_{c2}$ and the mixed embedding $\boldsymbol{e}_{mix}$ are used to obtain the embedding of the target content $\boldsymbol{e}_{c1}$. Similar to the experiments with speaker embedding de-mixing, the Separate-Concat method obtained the best results among all of the de-mixing functions. The test accuracy is much higher than the classification accuracy using the mixed signal. When the SNR is 5dB, the classification accuracy of the Separate-Concat method reached 93.3%.

Other de-mixing functions, such as Share-Concat, also delivered good performance, reaching 91.1% test accuracy at an SNR of 5dB, which is in stark contrast to its much lower performance in speaker embedding de-mixing.

Table 5.4: *Spoken word identification accuracy on the Speech Command dataset when using the estimated embedding of target content $\boldsymbol{e}'_{c1}$. "Before" denotes the word identification directly using $\boldsymbol{e}_{mix}$. "Clean" denotes word identification using $\boldsymbol{e}_{c1}$ that is extracted from clean speech.*

	Cosine Similarity			**Test Accuracy (%)**		
SNR	**-5dB**	**0dB**	**5dB**	**-5dB**	**0dB**	**5dB**
Before	0.21	0.33	0.39	28.5	39.2	41.4
Sub	0.50	0.68	0.71	59.4	76.1	81.1
Mul	0.46	0.60	0.67	54.4	71.2	79.6
Concat1	0.49	0.58	0.65	57.7	64.1	77.1
Concat2	0.51	0.66	0.73	59.5	73.9	81.8
Share-Concat	0.59	0.72	0.80	75.3	85.8	91.1
Separate-Concat	**0.71**	**0.78**	**0.86**	**81.1**	**87.5**	**93.3**
Clean	1.0			96.5		

Table 5.5 shows the symmetrical experimental results that using the estimated interference content embedding $\boldsymbol{e}'_{c2}$. In this scenario, the embedding of the target content $\boldsymbol{e}_{c1}$ and the mixed embedding $\boldsymbol{e}_{mix}$ are used to obtain the embedding of the interfering content $\boldsymbol{e}_{c2}$. Similar behaviour can be observed in this scenario with the experimental results in Table 5.4. The Separate-Concat method also obtained the best performance among all of the de-mixing methods, demonstrating that this method performs better either using the target embedding to obtain the interfering embedding, or using the interfering embedding to obtain the target embedding.

It is clear that the content embedding de-mixing network can filter out some of the information of the interference content, despite the fact that this is a much more challenging task, given that the content spoken by different people may have different properties, affecting the content information and making it harder for the system to distinguish. As a result, there is still a gap between the best results obtained from the

Table 5.5: *Spoken word identification accuracy on the Speech Command when using the estimated embedding of interfering content* $\boldsymbol{e}'_{c2}$*. "Before" denotes the word identification directly using* $\boldsymbol{e}_{mix}$*. "Clean" denotes word identification using* $\boldsymbol{e}_{c2}$ *that was extracted from clean speech.*

	Cosine Similarity			**Test Accuracy (%)**		
SNR	**-5dB**	**0dB**	**5dB**	**-5dB**	**0dB**	**5dB**
Before	0.33	0.25	0.12	39.5	36.4	25.3
Sub	0.56	0.52	0.44	79.7	73.5	58.4
Mul	0.54	0.49	0.38	77.5	70.7	52.4
Concat1	0.61	0.48	0.42	75.5	63.2	56.6
Concat2	0.66	0.52	0.43	80.4	72.5	59.4
Share-Concat	0.77	0.68	0.56	**90.4**	83.6	74.7
Separate-Concat	**0.78**	**0.71**	**0.64**	90.2	**86.4**	**80.1**
Clean	1.0			96.5		

content embedding (80.1% by the Separate-Concat method) and those from the clean data.

Another phenomenon is that the Sub method was unable to obtain better results than other methods such as Separate-Concat, again due to the complexity of the content information mentioned above. Unlike the results from speaker embedding de-mixing process, the simple mathematical operations struggle to process the information of the spoken content.

5.4.4 Summary

This section moved on to content, with the goal of establishing whether or not the same embedding de-mixing network can also filter out spoken content information in the embedding space. As introduced in Section 5.4.1, the content embedding de-mixing network takes the mixed embedding $\boldsymbol{e}_{mix}$ and the interfering content embedding $\boldsymbol{e}_{c2}$ to obtain the embedding of the target content $\boldsymbol{e}_{c1}$.

In the experiments laid out in Section 5.4.2, a speech command dataset was selected which contains a single isolated spoken word in each utterance. This is to ensure that the speaker identity stays the same for the entire utterance while the content information changes. Only utterances that contain single words were selected in order to control this variance and to simplify the evaluation process.

The results in Section 5.4.3 show that the content embedding de-mixing network can filter out some information of the interfering speakers, but there remains a gap between these results and those obtained from the clean signal, due to the data complexity mentioned above. There is no improvement from using the simple mathematical operations. The Separate-Concat method delivered the best results among all of the six embedding de-mixing functions, reaching 80.1% identification accuracy at a level of 5dB.

5.5 Speaker and Content Embedding De-Mixing

In Sections 5.3 and 5.4, it was demonstrated that the proposed methods are able to separate both speaker and content information in a two-speaker environment by using the de-mixing network when the embedding of the interfering speaker is available. This section proposes the embedding extraction method whereby the content embedding $\boldsymbol{e}_{c1}$ can be extracted using the speaker embedding $\boldsymbol{e}_{s1}$, or inversely the speaker embedding $\boldsymbol{e}_{s1}$ can be extracted using the content embedding $\boldsymbol{e}_{c1}$ in a two-speaker environment. This method is further referred to as the speaker and content embedding de-mixing.

5.5.1 Model Architecture

Before de-mixing the information in the embedding space by using the embedding de-mixing network, the clean speaker and the content embeddings need to be extracted. The model architecture of the clean embedding de-mixing process is the same as that introduced in Sections 5.3 and 5.4. The clean speaker embeddings are extracted using the speaker recognition network, and the clean content embeddings are extracted from the content recognition network.

After the clean speaker and content embeddings are obtained, the de-mixing network is trained, the model architecture of the de-mixing network in this scenario can be found in Appendix A.5.

Equation 5.11 shows the loss function of using $\boldsymbol{e}_{c1}$ to obtain $\boldsymbol{e}_{s1}$ and using $\boldsymbol{e}_{s1}$ to obtain $\boldsymbol{e}_{c1}$ respectively.

$$\begin{aligned}\mathcal{L}_1 =& \frac{1}{d}\sum_{i=1}^{d} |\boldsymbol{e}_{s1,i} - \boldsymbol{e}^{'}_{s1,i}| \\ \mathcal{L}_2 =& \frac{1}{d}\sum_{i=1}^{d} |\boldsymbol{e}_{c1,i} - \boldsymbol{e}^{'}_{c1,i}|\end{aligned} \tag{5.11}$$

5.5.2 Experiments

Experimental Setup

In order to evaluate the performance of the speaker and content extraction at the same time, TIMIT is used in this section. As discussed in Section 5.3.2, TIMIT contains both ground truth speaker labels and time-aligned phone labels. This is suitable for this experiment because the time-aligned phone labels can help the model learn the content information.

In order to extract the clean embeddings, the speaker embedding extraction process is the same as that described in Section 5.3.2. The speaker recognition model is trained using the clean signals. The embeddings for each utterance are extracted, and the final speaker embedding is the average of the utterance-level embeddings that belong to the same speaker.

The clean phone embeddings are learned using a phone classification network in a manner similar to the work in (Shi & Hain 2021). The utterances are segmented using a sliding window with 20 frames length and 10 frames step. The phone label for each segment is represented as the center frame's phone label, as the phone labels in TIMIT are accurate to the frame. After training, the final phone embedding is the average of the obtained embeddings that belong to the same phone. A total of 58 unique phone embeddings are obtained, with silences excluded.

After the clean embedding extraction, two separate embedding de-mixing networks are trained. The first network takes the mixed signal and $\boldsymbol{e}_{s1}$ as input, the output is $\boldsymbol{e}^{'}_{c1}$. When training this network, as the content changes over time in the utterances while the speaker identity keeps the same. The training process is at the segment level (the segment length is 20 frames). The de-mixing network is trained to predict the phone embedding from the input segment. The output phone embeddings for one utterance can be organised as a sequence by concatenating all of the predicted phone embeddings from one input utterance, which is denoted as $\boldsymbol{E}^{'}_{c1}$.

The second de-mixing network takes the mixed signal and $\boldsymbol{e}_{c1}$ as input, the output is $\boldsymbol{e}'_{s1}$. In this case, the input utterance is also segmented using a sliding window with 20 frames length and 10 frames step size. For each segment, the network will predict the speaker embedding $\boldsymbol{e}'_{s1}$. As the speaker identity stays the same across the whole sequence, the final predicted embedding takes the average of the outputs from all of the segments. Two de-mixing networks trained; the first takes the mixed signal and $\boldsymbol{e}_{s1}$ as the input to predict $\boldsymbol{e}_{c1}$, while the second takes the mixed signal and $\boldsymbol{e}_{c1}$ as the input to predict $\boldsymbol{e}_{s1}$. The output of the first de-mixing network is a sequence of the predicted phone embeddings, while the output of the second network is a single predicted speaker embedding.

Evaluation

In order to evaluate the quality of the predicted speaker and content embeddings, the cosine similarity score is used. As discussed in Section 5.3.2, this is a direct way to measure the distance between the predicted embedding and the reference clean embedding.

For the predicted speaker embeddings, the speaker identification task is also conducted to evaluate the performance of the predicted embeddings from the back-end speaker recognition model. This is also the same as that in Section 5.3.2.

For the evaluation of the content embeddings, in addition to the cosine similarity score, two tasks are conducted, namely isolated phone classification and continuous phone recognition. For the isolated phone classification, the output of the predicted phone embedding sequence can be divided into several segments, each of which contains a unique phone label. A phone classifier is used to evaluate the quality of the obtained phone embeddings. The phone classifier is pre-trained when obtaining the clean phone embeddings.

The isolated phone classification can be used to evaluate the quality of each output sequence phone embeddings, while the continuous phone recognition task evaluates the quality of the sequence predicted phone embeddings. In practice, the sequence of the predicted phone embeddings ($\boldsymbol{E}'_{c1}$) is taken as the input. The recognition network architecture largely follows that developed by Michalek & Vaněk (2018), and the detailed architecture can be found in Appendix A.5.

In addition to the above evaluation tasks, the additional noise information is mixed with the input sequence to evaluate the performance of the proposed speaker and content embedding de-mixing network in a complex environment. As the model does not have any prior information about the noise interference, the robustness of the proposed method can be evaluated. Similar to the experimental setups in Section 3.5, noise signals are taken from the MUSAN dataset (Snyder et al. 2015), which are mixed with the input mixed signals at specific SNR levels. In this case, only the general noise and the music noise types are selected. It is not necessary to use the babble type as there is already speaker interference in the mixed signal.

5.5.3 Results and Discussion

Extracting Speaker Embedding $\boldsymbol{e}'_{s1}$ from Content Embedding $\boldsymbol{e}_{c1}$

Table 5.8 shows the cosine similarity and speaker identification accuracy results when using the content embedding $\boldsymbol{e}_{c1}$ to obtain the speaker embedding $\boldsymbol{e}_{s1}$. Similar to the previous results in Section 5.3, the Separate-Concat method delivered the best results, reaching a 0.76 cosine similarity score and a speaker identification accuracy of 82.2%, compared with an accuracy of 98.5% on the clean embeddings. This is a larger gap than emerged with the speaker embedding de-mixing network (shown in Tables 5.1 and 5.2). Using the content embedding to obtain the speaker embedding is much more difficult than using the embedding for one speaker to obtain the other. The speaker embeddings comes from the same embedding space, and are trained together, while the content embeddings and speaker embeddings represent different information and are trained separately, which means they come from different embedding spaces. The results using the Sub method were comparable to those from the Separate-Concat, but still lower than those from the speaker embedding de-mixing network. This shows

Table 5.6: *Speaker identification and cosine similarity results when using the content embeddings to obtain the speaker embeddings. "Before" denotes the speaker identification directly using $\mathbf{e}_{mix}$. "Clean" denotes speaker identification using $\mathbf{e}_{s2}$ that was extracted from clean speech.*

	Cosine Similarity (%)			**Test Accuracy (%)**		
SNR	**-5dB**	**0dB**	**5dB**	**-5dB**	**0dB**	**5dB**
Before	0.22	0.45	0.64	36.5	58.4	72.5
Sub	0.62	0.67	0.71	72.7	76.5	78.9
Mul	0.51	0.58	0.60	65.5	62.4	64.7
Concat1	0.27	0.32	0.38	39.7	42.2	46.4
Concat2	0.48	0.53	0.56	58.7	60.2	63.9
Share-Concat	0.43	0.48	0.50	50.2	54.3	56.5
Separate-Concat	0.65	0.71	0.76	77.7	78.4	82.2
Clean	1.0			98.5		

the mathematical operation can help the model to extract the speaker information from the corresponding content embeddings, but the performance is affected due to the complexity of the different embedding spaces discussed above.

Table 5.7 shows the results when using the content embedding to extract the corresponding speaker information under different noise interference conditions. The SNR level of the the interfering speaker is fixed at 5dB. This situation is more difficult than the two-speaker environment as the model not only needs to separate the speaker properties, but it also need to filter out the complex noise information.

It is obvious that when the energy level of the noises is larger, there is a larger gap between the results obtained and those derived from the clean conditions. However, the Separate-Concat method achieved a 70.3% identification accuracy compare with not using the de-mixing network (39.6%). This shows that it has a better noise robustness when separating information from different speakers. The results obtained by the Sub method are lower than those obtained by the Separate-Concat method, which indicates the performance of the mathematical operation in the embedding space is affected by

Table 5.7: *Speaker identification and cosine similarity results when using the content embeddings to obtain the speaker embeddings. The level of the speaker interference is fixed at 5dB. "Before" denotes the speaker identification directly using $\mathbf{e}_{mix}$. "Clean" denotes speaker identification using $\mathbf{e}_{s2}$ that was extracted from clean speech.*

Model	**SNR**	**Noise**		**Music**	
		Cosine	**Accuracy (%)**	**Cosine**	**Accuracy (%)**
Before		0.33	43.7	0.26	39.6
Sub	5	0.52	64.7	0.47	59.4
	10	0.56	68.6	0.50	62.3
Mul	5	0.47	51.3	0.42	45.7
	10	0.51	56.2	0.49	50.0
Concat1	5	0.25	32.5	0.22	29.4
	10	0.28	37.6	0.25	33.6
Concat2	5	0.41	51.6	0.36	46.2
	10	0.48	55.7	0.42	49.7
Share-Concat	5	0.37	47.8	0.31	45.0
	10	0.43	50.2	0.38	48.6
Separate-Concat	5	0.61	70.6	0.54	67.6
	10	0.66	72.7	0.59	70.3
Clean		1.0	98.5	1.0	98.5

the noise interference, and the noise robustness of the mathematical operation is lower than for other network architectures. This is because when operating the subtraction operation in the embedding space, it can only subtract some of the influence of the interfering speakers, but not the noise interference, as there is no prior information provided about the noises.

Extracting Content Embedding $\boldsymbol{e}_{c1}^{'}$ from Speaker Embedding $\boldsymbol{e}_{s1}$

Isolated Phone Classification

After evaluating the obtained speaker embeddings, the obtained content embeddings were then evaluated. Table 5.8 shows the isolated phone classification results of using $\boldsymbol{e}_{s1}$ to obtain $\boldsymbol{e}_{c1}$. Once again, the Separate-Concat method delivered the best performance, with a 0.60 similarity score and 63.2% accuracy, compared with 74.1% from

Table 5.8: *Phone classification and cosine similarity results when using the speaker embeddings to obtain the phone embeddings. "Before" denotes the phone classification directly using $\mathbf{e}_{mix}$. "Clean" denotes phone classification using $\mathbf{e}_{c2}$ that was extracted from clean speech.*

	Cosine Similarity			**Test Accuracy (%)**		
SNR	**-5dB**	**0dB**	**5dB**	**-5dB**	**0dB**	**5dB**
Before	0.11	0.16	0.24	19.7	21.4	29.8
Sub	0.30	0.42	0.49	48.7	52.5	58.8
Mul	0.25	0.37	0.40	42.5	49.7	53.2
Concat1	0.11	0.16	0.23	22.0	27.1	30.8
Concat2	0.20	0.31	0.38	38.7	40.2	42.7
Share-Concat	0.32	0.38	0.44	47.6	49.8	52.2
Separate-Concat	0.45	0.52	0.60	55.8	57.6	63.2
Clean	1.0			74.1		

the clean phone classification results. It is important to note that phone classification differs from the widely reported phone recognition experiments on TIMIT (continuous phone recognition). Classification uses phone boundaries which are assumed to be known. However, no contextual information is available, which is typically used in the recognition setups by means of triphone models, or bigram language models. Therefore the task is often more difficult than recognition.

Similar to the results in the previous experiments, the Sub method generate reasonable results. However, the gaps between the clean results and the results obtained by the Sub method are larger than those from the Separate-Concat method. As discussed in Section 5.3, the reason is that the model cannot capture the information from the separate embeddings by concatenating them before passing them through into the network layers.

The Sub and Mul methods obtained higher results compared with those when not using the de-mixing network, indicating that the mathematical operations can capture the isolated content information in the embedding space. Combined with the results

Table 5.9: *Phone classification and cosine similarity results when using the speaker embeddings to obtain the phone embeddings in different noise conditions, the interference from the other speaker is fixed at 5dB. "Before" denotes the phone classification directly using $\mathbf{e}_{mix}$. "Clean" denotes phone classification using $\mathbf{e}_{c2}$ that was extracted from clean speech.*

Model	SNR	Noise		Music	
		Cosine	Accuracy (%)	Cosine	Accuracy (%)
Before		0.10	17.4	0.08	15.9
Sub	5	0.31	42.4	0.28	41.1
	10	0.36	47.1	0.33	46.2
Mul	5	0.28	36.2	0.26	34.2
	10	0.33	39.4	0.30	37.4
Concat1	5	0.07	20.3	0.05	18.4
	10	0.12	23.4	0.09	21.2
Concat2	5	0.20	29.6	0.18	26.6
	10	0.27	32.6	0.22	29.2
Share-Concat	5	0.24	40.4	0.19	37.4
	10	0.29	43.7	0.23	39.6
Separate-Concat	5	0.43	51.1	0.37	49.6
	10	0.48	54.7	0.44	52.7
Clean		1.0	74.1	1.0	74.1

obtained in Section 5.3.2, the mathematical operations can not only help the model to filter out the speaker information in the embedding space, but can also extract the content information using the speaker information. However, as discussed in Section 5.4.2, the mathematical operations deliver lower results in the content embedding de-mixing process. This indicates that the speaker information is necessary for the mathematical operations to capture content information in the embedding space, as both the speaker de-mixing network in Section 5.3.2 and the de-mixing network in this section take the speaker embeddings as input.

Table 5.9 shows the performance of the isolated phone classification under noisy conditions. A similar phenomenon can be observed in that the noise interference can affect the performance for all of the six methods. However, the Separate-Concat method delivered a higher level of noise robustness, reaching 52.7% accuracy at 10dB of music noise compared with 63.2% obtained in clean conditions. The Sub method has lower noise robustness, reaching 46.2% in 10dB of music noise compared with 58.8% obtained in clean conditions, but this is still higher than the results obtained by other de-mixing functions. This phenomenon is similar to the observations from Table 5.7. It shows the Separate-Concat method not only delivers better results when separating the speaker or content information in the embedding space, but can also reach a high noise robustness level when different noise types affect the quality of the input signal.

Phone Recognition On the Obtained Content Embedding Sequence $\boldsymbol{E}_{c1}^{'}$

Having evaluated the performance using the isolated phone classification, the continuous phone recognition is also important as it evaluates the contextual information captured by the model. Table 5.10 shows the results of the continuous phone recognition using the obtained phone embedding sequence $\boldsymbol{E}_{c1}^{'}$. The phone error rates (PER) are reported instead of the accuracy as it is a widely used measurement in speech recognition. The Separate-Concat method achieved 30.2% PER, as compared to the clean phone recognition results of 16.6% PER.

The Sub method reached 37.6% PER in continuous phone recognition, which is the second lowest PER among all of the six architectures. Mul, the other mathematical method, also performed better than when not using the de-mixing functions. The Share-Concat method shows different performances in this section compared with that in Sections 5.3.2 and 5.4.2. In the previous sections, it obtained lower results because it uses two shared DNN layers to process the two input embeddings, which makes it difficult for the model to distinguish the information from the separate embeddings. However, in this experiment, it reaches 38.2% PER which is comparable with the best result (30.2%) obtained by the Separate-Concat method. The reason why the Share-

Table 5.10: *Continuous phone recognition results using the phone embeddings obtained by the speaker embeddings. The phone error rate (PER) is reported. "Before" denotes the phone recognition directly using $\boldsymbol{e}_{mix}$. "Clean" denotes phone recognition using $\boldsymbol{E}_{c2}$ that was extracted from clean speech.*

	Phone Error Rate(%)		
SNR	**-5dB**	**0dB**	**5dB**
Before	87.4	79.3	72.5
Sub	42.5	40.2	37.6
Mul	50.2	47.9	44.7
Concat1	60.2	58.8	55.4
Concat2	55.8	53.5	50.1
Share-Concat	46.1	40.5	38.2
Separate-Concat	40.1	35.9	30.2
Clean	16.6		

Concat method performs well in the continuous phone recognition task but less so in the isolated phone classification task may be that the contextual information of the input sequence helps the model to distinguish the information from different input embeddings.

5.5.4 Summary

In this section, a speaker and content embedding de-mixing approach is proposed that aims to extract the information of the target speaker from the corresponding spoken content information or, inversely, extract the information of the spoken content of the target speaker using the corresponding speaker embeddings. In Section 5.5.1, the model architectures are discussed. The clean embedding extraction network makes use of the model architecture developed in Sections 5.3 and 5.4. Then, after the clean content and speaker embeddings are extracted, they are used for the embedding de-mixing network.

In Section 5.5.2, the experimental setup is discussed. The TIMIT dataset is used as it contains both the ground truth speaker labels and the time-aligned content labels (phone labels). When designing the experiments, the obtained speaker embeddings are evaluated using the cosine similarity score and speaker identification accuracy, while the content embeddings are evaluated in two different ways: the isolated phone classification task and the continuous phone recognition task. The isolated phone classification task directly evaluates the quality of the obtained phone embeddings, and the continuous phone recognition task further evaluates the contextual information that is captured by the de-mixing network. Both the speaker and the content embeddings are also evaluated under different noise interferences to reveal the levels of robustness.

The results shown and discussed in Section 5.5.3 show that both the speaker and the content information can be extracted from the embeddings space. However, there is a gap between these results and those derived from the clean conditions which is due to the complexity of the speaker and content embeddings. These two types of embeddings encode different information and are trained separately using different networks. It is difficult for the module to capture the information from the speaker embedding by use of the content embedding.

Among the six proposed de-mixing functions, the Separate-Concat method provided the best results, and demonstrated the best robustness under noisy conditions. The two mathematical operations can obtain results comparable to the Separate-Concat method, but when the input contains noise information, the performance of these methods is reduced.

5.6 Conclusion

In this chapter, an embedding de-mixing approach was proposed. As discussed in Section 5.1, in order to filter out the speaker interference and improve the performance of the back-end module, target speaker extraction methods are widely used that isolate the voice of the target speaker and construct it to a clean signal.

However, when the purpose is to improve the performance of the back-end module, such as on speaker recognition model, it may not be necessary to construct the information of the target speaker into the signal space, as the decoding process (or reconstruction process) can also affect the performance. Instead, separating the information of the target speaker into the embedding space may be more efficient. Firstly, embeddings are low dimensional, and it can project signals of variable length into fixed length. Moreover, the back-end speaker recognition model can also be simplified as it can learn a mapping from the embeddings to the speaker identity scores rather than from the signals to the speaker identity scores.

Following a review of the literature, ways to separate different speaker properties into the embedding space have yet to be developed and, with this being the case, the first method was presented here and is referred to as the embedding de-mixing network. Section 5.2 shows its high level architecture of three components;

- the recognition network that obtains the clean embeddings;
- the embedding extractor that maps the mixed signal into the embedding space;
- the de-mixing function that separates the information in the embedding space.

In order to comprehensively evaluate the proposed approach, six different embedding de-mixing functions were designed. They can be classified into three categories: simple mathematical operations (the Sub and Mul methods) which are based on subtraction and multiplication operations; concatenation before processing (the Concat1 and Concat2 methods) which merge the information of the two input embeddings by a concatenation operation before passing them through the network layers; and concatenation after processing (the Share-Concat and Separate Concat methods) which merge the information of the two input embeddings by a concatenation operation after passing them through the network layers.

Based on the architecture introduced above, the embedding de-mixing method can be separated into three steps: speaker embedding de-mixing, content embedding de-mixing and the speaker and content embedding de-mixing.

In the speaker embedding de-mixing method in Section 5.3, the goal is to filter out the influence of one speaker using the embedding of the other speaker in the same two-speaker signal. This step has two scenarios: using the embedding of the interfering speaker to obtain the embedding of the target speaker, and using the embedding of the target speaker to obtain that of the interfering speaker. The clean embeddings are pre-trained and collected using a speaker recognition network. Then, a de-mixing network is trained which takes the mixed signal and one of the embeddings to obtain the embedding of the other speaker. The model is evaluated on two datasets: the augmented TIMIT dataset and the real-world MC-WSJ dataset. The results show that the speaker embedding de-mixing network can filter out the information of one speaker using the embedding of the other speaker.

Section 5.4 introduced the content embedding de-mixing network which filters out the spoken content information of one speaker by the corresponding content embedding. Similar model architectures are used for this step, and the Speech Command dataset is used. The results show that the content embedding de-mixing network can filter out the content information from one speaker, but not as efficiently as it does with the clean data. This is due to that the content information may be complex, with different speakers speaking the same content, thereby constantly changing the content information and making it difficult for the module to capture that information.

Section 5.5 introduced the third scenario, referred to as the speaker and content embedding de-mixing. In this step, the model uses the speaker embedding to extract the corresponding content information or, inversely, uses the content embedding to extract the corresponding speaker information. The TIMIT dataset was used as it contains the speaker labels and the aligned phone labels. The experimental results covered on both the obtained speaker embeddings and the obtained content embeddings. The speaker identification accuracies show the obtained speaker embedding can be extracted from the corresponding content embedding, although the quality is lower than in the speaker embedding de-mixing step. The speaker and content embeddings are in different embedding spaces, which may affect the performance. The isolated phone classification and continuous phone recognition results show the content embeddings can be extracted from the corresponding speaker embeddings, but there is still a gap to those obtained by the clean signal for same reasons as discussed earlier. The additional experiments use the noise signal mixed with the data to evaluate the noise robustness of the proposed model. The results show the noise information can affect the performances of all of the six de-mixing functions, but the Separate-Concat method still obtains reasonable results.

Comparing the six proposed embedding de-mixing functions, the Separate-Concat method out-performed all the others in all three steps described above. The Sub method can deliver results that are comparable to the Separate-Concat method, except for when the speaker information is not available, such as in the content embedding de-mixing approach. It also obtained a lower noise robustness when there are noise interferences in the input signals. The reason may be that the content information or the noise interference is so complex that the simple architecture of the mathematical operations (Sub and Mul) can not capture them. Other methods, such as Concat1 and Concat2, delivered worse results in almost all conditions due to the concatenation process being applied before the network layers so that the model cannot distinguish different types of information from the different input embeddings.

Chapter 6

Weakly Supervised Speaker Identification

6.1 Introduction

Section 2.4.3 discussed a scenario in which overlapped speakers, engaged in a conversation, in a single signal. There is no target speaker, the number of speakers is unknown, and all the speakers need to be recognised by the system. In such a situation, neither target speaker extraction techniques (discussed in Section 2.4.2) nor the embedding de-mixing approach (as proposed in Chapter 5) will be of any utility.

As discussed in Section 2.4.3, one possible solution for this situation is to manually annotate each speaker, as well as the time in which each speaker occurs in the signal. However, this is expensive and time consuming. An alternative solution is to use a front-end speaker diarization model before recognising all the speakers, though this may still require manually annotated labels for training purposes.

In order to achieve the goal discussed above, weakly supervised learning may be a potential solution. As discussed in Section 2.4.3, one scenario in weakly supervised learning, known as inexact supervision, can learn directly from the coarse grained labels. When the input signal contains an unknown number of speakers, and the goal is to recognise all of the speaker identities, the inexact supervision in weakly supervised learning can make direct use of the speaker labels without the need for the time information for each speaker. In this way, it is not necessary to manually annotate all of the speaker labels with the time information as only a set of speaker identities is needed.

From a review of the literature, there appears to be only one study (Karu & Alumäe 2018) that has employed weakly supervised learning to speaker recognition, using a pre-trained speaker diarization system and a pre-trained i-vector extraction model. Those two systems still need manually annotated time information for each speaker. An end-to-end weakly supervised speaker identification approach that can directly learn the mapping from the signal to the set of speaker labels is as yet undeveloped.

In this chapter, an approach is proposed that uses weakly supervised training for speaker identification through the use of two neural network architectures: the hierarchical attention network and the hierarchical transformer network. The hierarchical attention network is based on the network architecture that was proposed in Chapter 3. In this chapter, it is trained in a weakly supervised manner because the hierarchical structure is shown to have a good noise robustness. As discussed in Section 2.4.1, the interfering speakers affects the speech signal in a similar but stronger way. When there are multiple speakers overlapping, the frame-level encoders of the hierarchical attention network are assumed to capture the local information that may be useful for the model to capture the speaker related information.

However, the hierarchical attention network has two disadvantages. The attention mechanism used in the frame-level encoders may not be able to capture the properties from multiple speakers in a single input utterance. The second problem is that there are no connections between each of the frame-level encoders, which may negatively impact performance. To address these problems, a hierarchical transformer network (T-vector) is proposed. The T-vector model also contains a frame-level encoder and a segment-level encoder. The model makes use of transformer encoder blocks in both the frame-level and segment-level encoders to better capture multiple speaker information and process the sequence in parallel instead of the GRU layer. The multi-head attention mechanism (Vaswani et al. 2017) used in the transformer block may better capture different speaker properties from the multi-speaker input signal. An additional memory mechanism is used between each frame-level encoder to capture long-term information and improve the performance (Dai et al. 2019).

6.1.1 Chapter Outline

The remainder of this chapter is organised as follows:

- Section 6.2 defines and discusses the end-to-end weakly supervised speaker identification task.
- Section 6.3 adapts the developed hierarchical attention network to weakly supervised learning, and the experimental results are also shown.
- Section 6.4 introduces and discussed the proposed hierarchical transformer network, combined with the experimental results and discussion.
- Section 6.5 provides the conclusion.

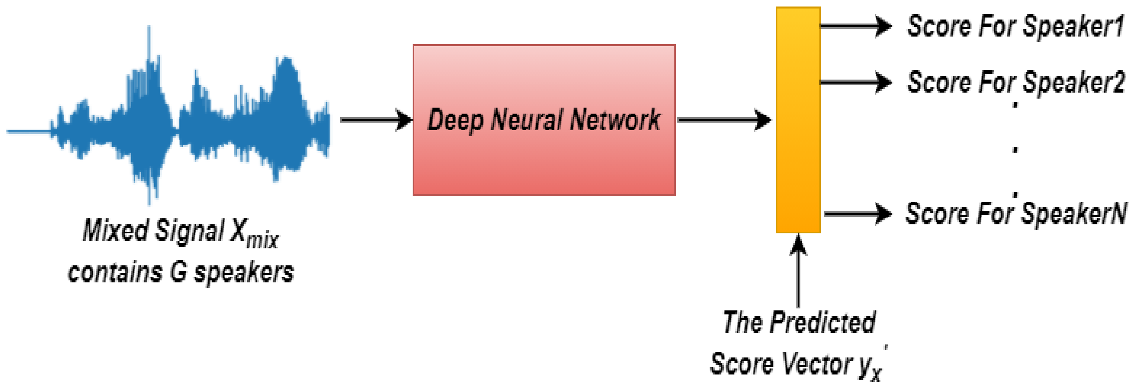


Figure 6.1: *A diagram of the end-to-end weakly supervised speaker identification task.*

6.2 Task Definition

The goal for the weakly supervised speaker identification is to recognise all the speaker identities within an input utterance (or in a given period of time) when the utterance contains an unknown number of speakers.

Figure 6.1 shows a diagram of the weakly supervised speaker identification task. Suppose the total number of speakers in the training set is N, the input data $\boldsymbol{X}_{mix}$ contains a random number of G speakers. The number $G <= N$ that indicates the speakers that occurred in the input utterance are all from the training set, but the number G is unknown for each training utterance. For each input data $\boldsymbol{X}_{mix}$, the label is a set of speaker identities without any time information. $\boldsymbol{X}_{mix}$ contains the mixed time domain signals, and it will be converted into MFCC features before passing through the neural network model. Specifically, the label for $\boldsymbol{X}_{mix}$ is denoted as $\boldsymbol{y_x} = \{y_{x,1}, y_{x,2}, ..., y_{x,N}\}$. It is organised as a vector of dimension N, which is the same as the total number of speakers. Each element of $\boldsymbol{y}_x$ denotes one speaker from the training set, and the value is either one or zero, where number one represents the corresponding speaker who occurs in $\boldsymbol{X}_{mix}$ while the number zero represents the speaker who is not present in $\boldsymbol{X}_{mix}$. It is obvious that the number of elements in $\boldsymbol{y}_x$ whose values are equal to one is G. For example, suppose $\boldsymbol{y}_x = \{0, 1, 0, 0, 1, 0\}$ denotes a label vector for an input signal which contains two speakers (G equals to 2), and the total number of speakers is six (N equals to 6).

A neural network is trained to predict the speakers who occurs in $\boldsymbol{X}_{mix}$. For example, when $\boldsymbol{X}_{mix}$ is input to the neural network, the output is the predicted score vector $\boldsymbol{y}_x^{'}$, which contains the estimated scores for each speaker. The score values are between zero and one.

From the above description, one can observe that the weakly supervised speaker identification can make direct use of the weakly labelled data as the input data contains a random number of speakers, but only the utterance-level labels are available. As discussed in Section 2.4.3, the definition of the weakly supervised speaker identification in this section can satisfy the requirement to omit the need for manually annotated speaker labels. The proposed task definition does not require the time information of the speakers.

Another key point in the task definition discussed above is that it is a closed-set task, where the total number of speakers N is fixed. As discussed in the beginning of Chapter 2, this it is similar to the closed-set speaker identification task. However, it is convenient here to convert it to an open set task. An extra dimension that represents the unseen speaker can be added to the score vector $\boldsymbol{y}_x$. This dimension represents the speaker that is not in the training set, and will change the dimension $\boldsymbol{y}_x$ and $\boldsymbol{y}_x^{'}$ to $N + 1$. In this chapter, the focus is on the closed-set scenario, though additional experiments for adapting it into an open-set scenario are discussed in Section 6.4.3.

$$\mathcal{L} = -\sum_{i=1}^{N} y_i \text{log} y_i^{'} + (1 - y_i)\text{log}(1 - y_i^{'}) \tag{6.1}$$

The most important part of the weakly supervised speaker identification task is the training target or loss function. In this task, as the labels for each speaker are binary labels, thus, the binary cross-entropy loss is used as the loss function, which is shown in Equation 6.1, where $y_i^{'}$ and y_i denote the predicted score and the ground truth label for the ith speaker, N denotes total number of speakers in the training set.

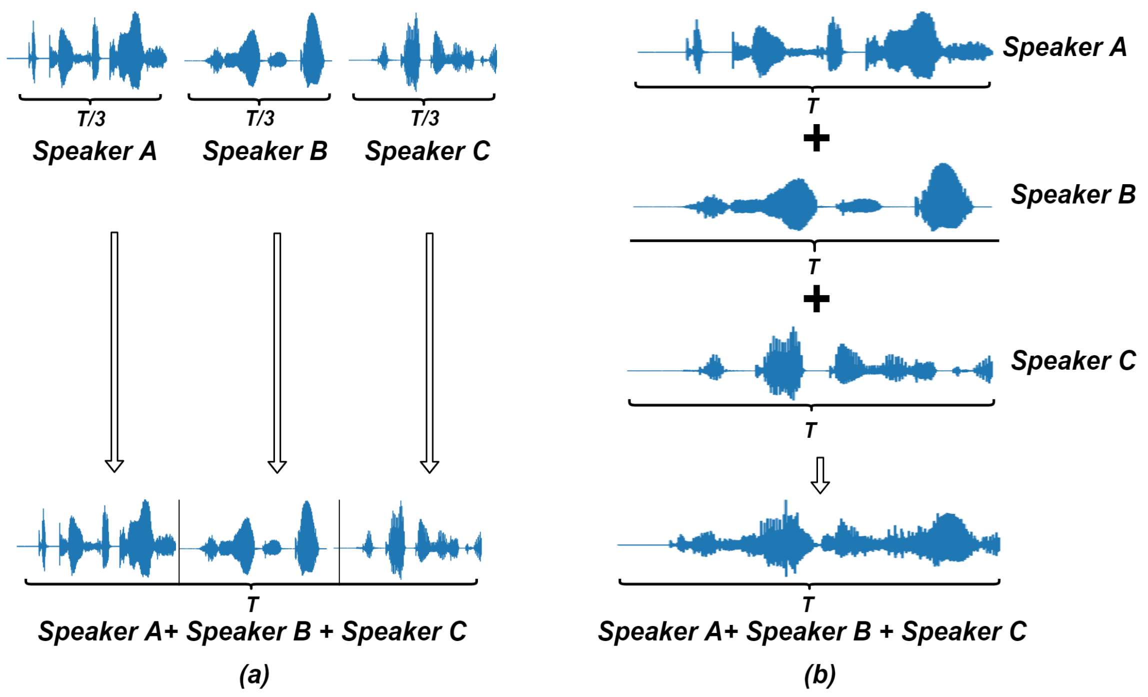


Figure 6.2: *The illustration of the data construction process. (a): Concat; (b): Overlap.*

6.3 Weakly Supervised Training of the Hierarchical Attention Network

6.3.1 Experimental Setup

Data Construction

There is no ready-made data available for this task. Consequently, new datasets are constructed from the Voxceleb1 and SWBC datasets that were first introduced in Sections 2.5 and 3.4.1.

In order to conduct weakly supervised training, two data scenarios were designed, namely Concat and Overlap. Figure 6.2(a) shows an example of the Concat scenario where three speakers' voices are concatenated without any overlap, while Figure 6.2(b) shows an example of the Overlap scenario where the three speakers' voices are completely overlapped.

Table 6.1: *Details for the construction of the four datasets: SWBC-S, Vox-L, SWBC-S and Vox-L.*

Name	Original Dataset	Type	#Select Speaker	#Utterance Train	#Utterance Test
SWBC-S	SWBC	Telephone	254	6000	20,000
SWBC-L	SWBC	Telephone	254	100,000	20,000
Vox-S	Voxceleb1	Interview	1000	15000	30,000
Vox-L	Voxceleb1	Interview	1000	150,000	30,000

There are different reasons for constructing these two data scenarios. From Figure 6.2(a), it is obvious that the overlap rate for the three speakers is zero. As there is no published baseline for this task, it is unknown what performance to expect when identifying unknown number of speakers from one input utterance. Thus, the Concat scenario provides a simple situation which can reveal the upper limits for the model in this task. The Overlap scenario provides the most difficult situation, where all the speakers speaker at the same time. In this case, the purpose is to evaluate the lower limit of the model. Having evaluated the proposed model in these two scenarios, additional experiments are shown in Section 6.4.3 that use the data containing a random overlap rate for multiple speakers.

In order to test the robustness of the proposed approach, for each of the two scenarios, four datasets were generated based on SWBC and Voxceleb1. The details are shown in Table 6.1. SWBC-S (where S stands for small) is derived from the SWBC dataset. Each speaker occurs 30 times on average within the training set. SWBC-L, where L stands for large, contains more training data, with each speaker occurring an average of 200 times in the training set, while the amount of test data remains the same. These small and large versions of the datasets are used to explore the robustness of the proposed model with small and large amounts of training data. The Voxceleb1 was configured into small and large datasets in a similar way. For Vox-S, 1,000 speakers were randomly selected from the dataset, each occurs 30 times in the training set. For

Vox-L, each speaker occurs 300 times in the training set, while the test set is the same as for Vox-S. For each of the eight datasets, the number of speakers in each signal (G) is randomly chosen from one to three in all datasets. When the number of speakers is one, the generated utterance is the same as the original utterance. When the number of speakers are two or three, the output utterance contains multiple speakers, with or without overlap. The method for mixing the utterances can be found in Appendix B.1.

Baseline

As the experiments here used the developed H-vector model, the selected baselines are the same as those in Chapter 3, namely the X-vectors from Snyder et al. (2018) and the attentive X-vector (or Att-Xvector) from Zhu et al. (2018), Okabe et al. (2018), Wang, Okabe, Lee, Yamamoto & Koshinaka (2018), and rahman Chowdhury et al. (2018). The reason is that although the H-vector has already been shown to deliver better performance under noisy conditions than the X-vector and the attentive X-vector, it is still not clear how it will perform in a multi-speaker environment. As discussed in Section 2.4.1, interfering speaker in the speech signal is much more difficult to process than simple background noise.

As discussed in Section 3.4, the X-vector is a TDNN-based model which contains a TDNN-based frame-level feature extractor, a statistics pooling operation and a segment-level feature extractor. A comparison between X-vector and H-vector can evaluate the effectiveness of the attention mechanism used in the H-vector model. The attentive X-vector model contains an additional global self-attention mechanism to allocate different weights for different frames ahead of statistics pooling operation. Thus, the attentive X-vector is used to compare the different attention mechanisms: the global attention in attentive X-vector and the hierarchical attention in H-vector.

As shown in Section 3.5.2, different window and step sizes can control the amount of information provided to the frame-level and segment-level encoders, thereby affecting the performance. In order to evaluate the relationships between the segmentation settings and performance, similar to that in Section 3.5.2, it is split into two scenarios: H-vector+sliding window and H-vector+static window. In H-vector+sliding window, the window length L_{win} is set to 20 frames, and the step length L_{step} is set to 10 frames. In H-vector+static window, the L_{win} is set to 20 frames, and the L_{step} is set to the same as L_{win}, which means there is no overlap for each local segment. The implementation details can be found in Appendix A.6.

Evaluation Metric

As discussed in Section 2.2.3, prediction accuracy is used as the evaluation metric for the speaker identification task. However, this is not suitable for the weakly supervised speaker identification task because the number of speakers in each training utterance (G) is unknown. When using the prediction accuracy as the evaluation metric, the model prediction comes from the prediction with the highest probability. It can only be used when there are a certain number of positive labels in one training sample. When there are multiple unknown numbers of labels, it is impossible to determine which predictions have the highest score (Xu et al. 2017).

Therefore, the equal error rate (EER), as discussed in Section 2.2.3, is employed here, as it does not require G to be fixed. In this scenario, each dimension of the predicted score vector $\boldsymbol{y}_x^{'}$ can be viewed as the output of a binary classification task. The EER is firstly computed for each predicted score vector, the EER for the whole test set is computed as the average of the EERs of all of the predicted score vectors.

6.3.2 Results and Discussion

Performance for the Concat Scenario

The first experiment focuses on the performance of the models in the simple Concat scenario, where the speakers do not overlap in the input utterance. Figure 6.3 shows the results obtained using the four models (X-vector, attentive X- vector, H-vector with static window and H-vector with sliding window) under different test conditions (1, 2,

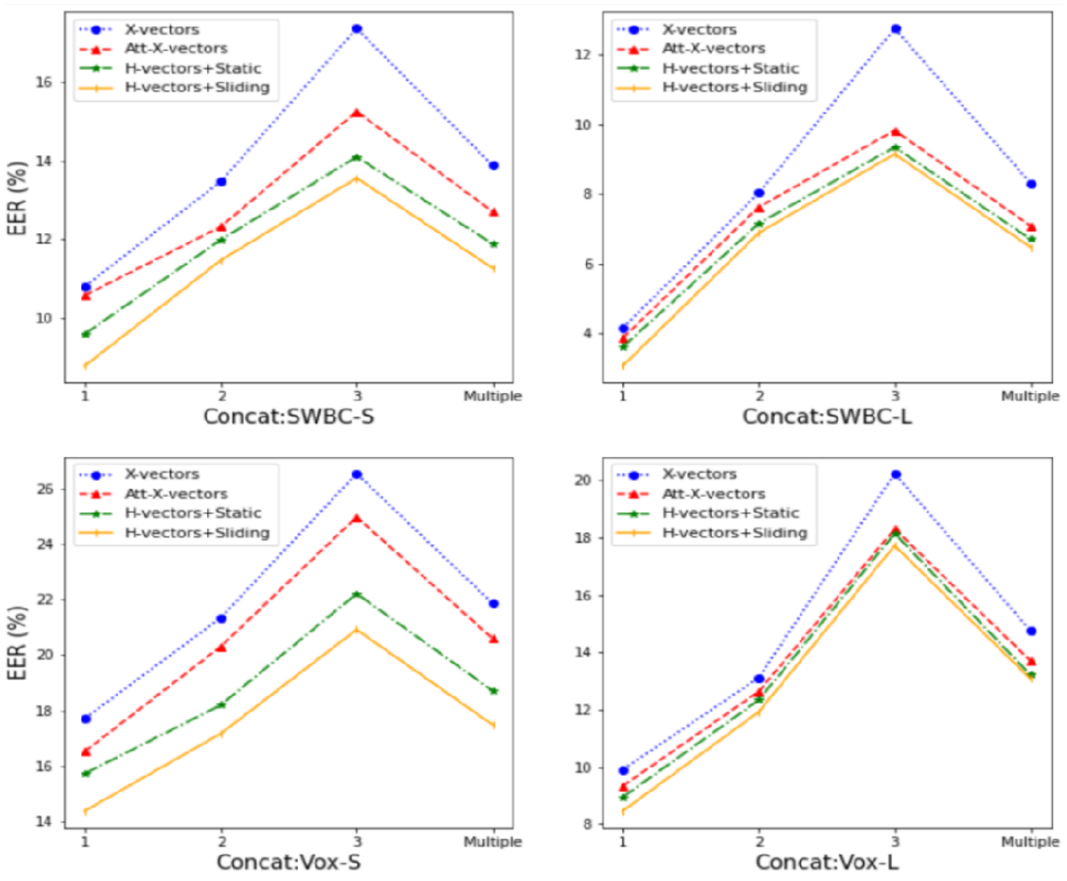


Figure 6.3: *The EERs (%) obtained using four models: X-vectors, attentive X-vectors, H-vector with static window and H-vector with sliding window under different test conditions on the four designed datasets for the Concat scenario.*

3 or multiple speakers) on the four designed datasets (SWBC-S, SWBC-L, Vox-S and Vox-L). In each figure, the X-axis represents the number of speakers in an utterance (1, 2 and 3), "Multiple" means the combination of all three cases, which means it is the average number of the three situations (1,2, and 3 speakers). With regard to the H-vector with static window, the window size L_{win} is 20 frames; in the H-vector with sliding window, the window size L_{win} is 20 frames and the step size L_{step} is 10 frames.

Notice that in the training sets, the number of speakers in each input utterance ranges randomly from one to three. For the test set, there are four different conditions, where the number of speakers are fixed at one, two or three. The "Multiple" scenario means the number of speakers are fixed randomly at either one, two or three, as in the training set.

From the results, it is clear that the H-vector model out-performed both the X-vector and the attentive X-vector. The attention mechanism gives the H-vector model the ability to highlight the important parts. It delivers better results than the X-vector and it out-performs the attentive X-vector. Because the hierarchical attention structure in H-vector captures both local and global features, compared to the global features captured by the global attention mechanism.

H-vector+sliding window performs better in almost all conditions, while the H-vector+static window performs better than the two baselines. This indicates the overlap between each segment for the H-vector can help the model better capture the information from different speakers.

Among all of the test conditions, the best results are obtained when the number of speakers in each utterance is one, and the worst case is when each utterance contains three speakers. Of course, it is easier for the models to capture the properties of a single speaker than multiple speakers speaking simultaneously. This is also the difficulty of the weakly supervised speaker identification task.

Moreover, when the training data is small, the proposed H-vector+sliding window still performs better than the baselines and the H-vector+static window, reaching 11.5% and 3.4% relative improvement over X-vector and Att-Xvector respectively with the SWBC-S dataset in the Concat scenario. This shows the robustness of the proposed H-vector+sliding window when there is insufficient training data.

The Performance for the Overlap Scenario

Figure 6.4 shows the results obtained for the four models on Overlap scenario. As discussed in Section 6.3.1, this is a difficult scenario in which all of the speakers are overlapped in the utterance signal with an overlap rate is 100%.

The results obtained from the Concat scenario are much better than those for the

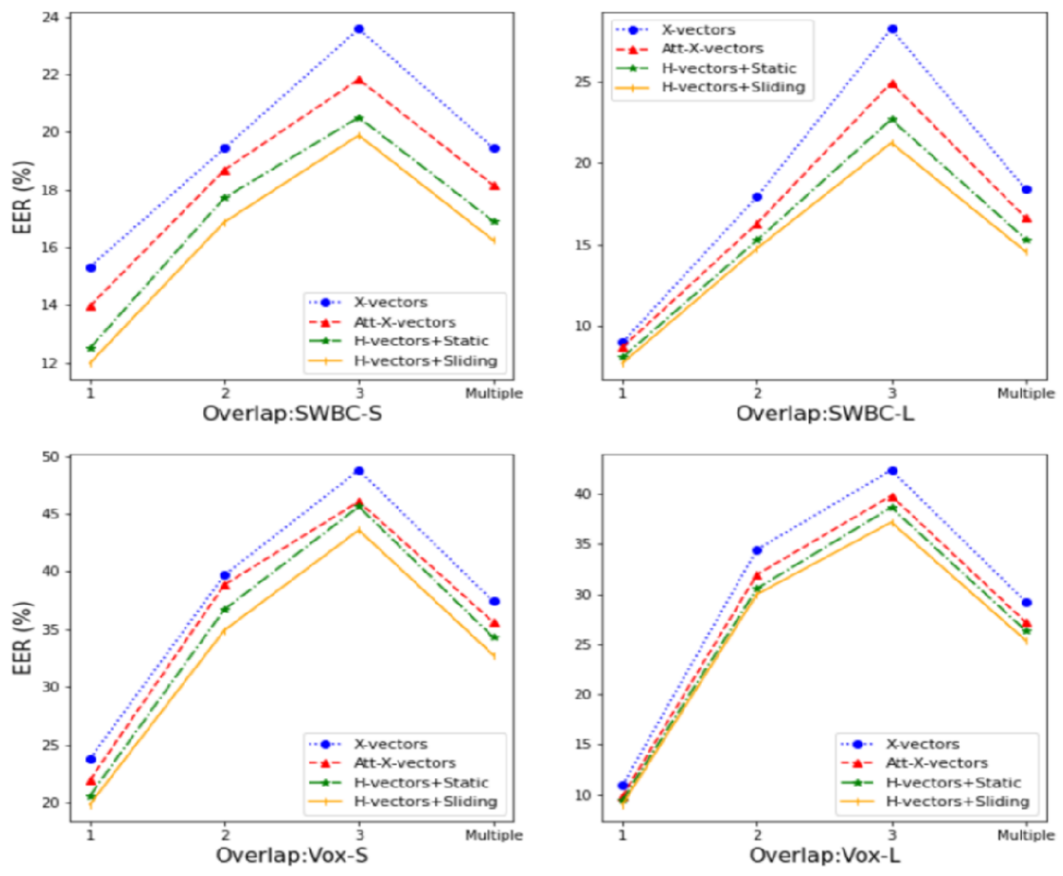


Figure 6.4: *The EERs (%) obtained using four models: X-vectors, attentive X-vectors, H-vector with static window and H-vector with sliding window in different test conditions on the four designed datasets in the Overlap scenario.*

Overlap scenario, confirming the discussion in Section 2.4.1 that interference from other speakers can compete for the same frequency band as other speakers, thus making it more difficult for the model to distinguish the elements of each speaker. The worst case is on the Vox-S dataset in the Overlap scenario, where the overlap rate is the highest and the there is only a small amount of training data available.

Although the Vox-S dataset in Overlap scenario is the most difficult one among all of the eight datasets, the H-vector+sliding window still out-performed the other models, reaching 43.0% EER, compared with the X-vector (48.5%) and attentive X-vector (45.6%) when there are three overlapped speakers in the test utterance.

Table 6.2: *The EERs (%) of the proposed H-vector architecture using different window sizes (from 10 to 30 frames), while the step size is kept at 10 frames.*

Data Type	**Window Size**	**EER (%)**			
		SWBC-S	SWBC-L	Vox-S	Vox-L
Concat	10	12.56	7.15	18.29	13.69
	15	11.87	6.85	18.08	13.34
	20	**11.27**	**6.47**	**17.48**	**13.08**
	25	11.69	6.59	17.81	13.29
	30	12.11	6.92	18.21	13.66
Overlap	10	17.81	15.71	34.37	26.46
	15	16.89	15.05	33.48	25.85
	20	16.24	**14.56**	32.77	**25.39**
	25	**15.99**	15.58	**32.26**	25.94
	30	16.59	16.02	32.86	26.17

Effectiveness of Different Window and Step Sizes

The window and step sizes are, in effect, hyperparameters that control the length of the signal input to the frame-level and segment-level encoders. It is therefore necessary to evaluate the performance of the H-vector model at different window and step sizes, In a similar way to the experiments discussed in Section 3.5.3.

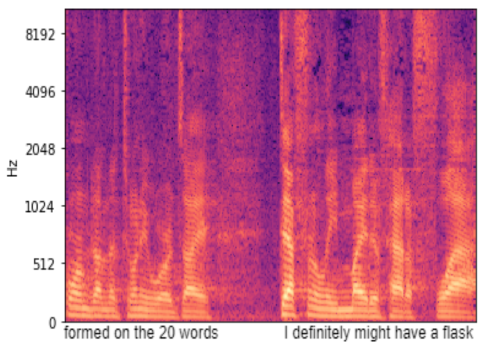
Tables 6.2 and 6.3 show the results obtained using the proposed H-vector+sliding window when using different window and step sizes. For the Overlap scenario, the EER is more sensitive to the change of window size and step size than in the Concat scenario. In most cases, the best results are obtained when the window size is 20 frames and the step size is 10 frames, where the step size is set to the half-size of the window size. A similar behaviour can be observed in Section 3.5.3, where the performance using different window and step sizes can be improved from a small size, but eventually, performance peaks and then declines. This confirms the findings in Section 3.5.3 stating that a larger window size leads to a longer segment being input into the frame-level encoder. The information between the beginning and the end frames of the segment

Table 6.3: *The EERs (%) of the proposed H-vector architecture using different step sizes (from 5 to 25 frames), while the window size is kept at 20 frames.*

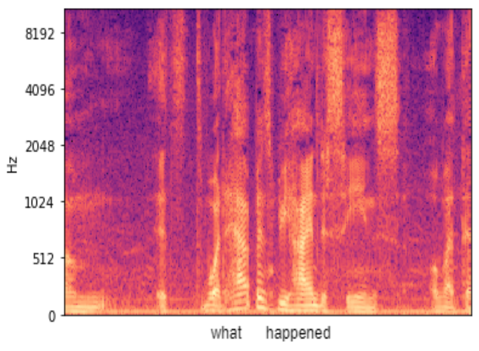
Data Type	**Step Size**	**EER (%)**			
		SWBC-S	SWBC-L	Vox-S	Vox-L
Concat	5	11.95	6.74	18.01	13.65
	10	**11.27**	6.47	**17.48**	13.08
	15	11.34	**6.29**	17.98	**12.82**
	20	11.45	6.96	18.21	13.15
	25	11.86	6.84	18.56	13.42
Overlap	5	16.49	14.92	33.87	25.51
	10	**16.24**	14.56	**32.77**	25.39
	15	16.88	**14.13**	33.53	**24.86**
	20	17.22	14.82	33.92	25.46
	25	17.78	15.11	34.25	25.81

may contain fewer relationships. When the window size is smaller, the segment length is small and there may be insufficient information provided to the frame-level encoder.

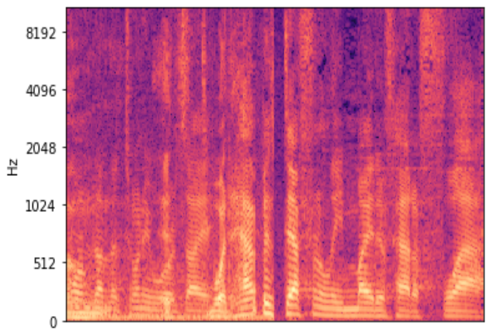
Step size also plays an important role with regard to the prediction results, even though the experiments in Section 3.5.3 demonstrate that the H-vector is more sensitive to the window size rather than the step size. This may be due to the complexity of the data structure. The step size controls the number of segments, thus it controls the information input to the segment-level encoder. The segment-level encoder captures the global features, which indicates that global information is also important in the weakly supervised speaker identification task. This is because when the input utterance contains multiple speaker, the model has to capture all the speaker properties and predict them. Even though it can capture local features using the frame-level encoder, the captured local information needs to be integrated at a higher level. If the step size is large, there will be insufficient information provided to the segment-level encoder, leaving it less able to capture the global features for all the speakers in a multi-speaker utterance.



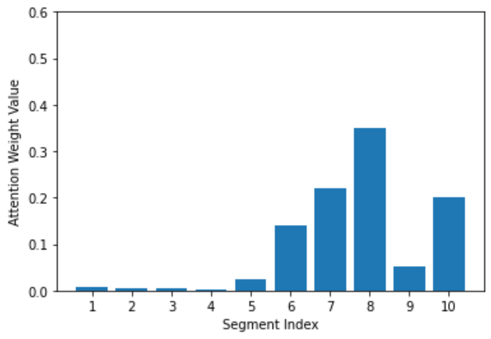
(a) Spectrogram of Speaker One



(b) Spectrogram of Speaker Two



(c) Spectrogram of Mixed Signal



(d) Attention Weights in H-vector

Figure 6.5: *The visualisation of the attention weights: (a) the spectrogram of Speaker One, (b) the spectrogram of Speaker Two, (c) the spectrogram of Mixed Signal, (d) the attention weights of the segment-level attention in H-vector model.*

Another finding is that the results are more sensitive to both the window size and the step size when in the Overlap scenario, where both local and global information are difficult to extract. Thus both the frame-level and segment-level encoders need more information in order to make a prediction.

6.3.3 Analysis

Visualisation of the Attention Weights

Following the analysis in Section 3.5.3, in order to observe how the hierarchical attention network works, the segment-level attention weights are visualised in Figure 6.5.

In Figure 6.5(a), the spectrogram of one speaker is shown; Figure 6.5(b) is the spectrogram of the second speaker; Figure 6.5(c) is a combination of the first two spectrograms in (a) and (b), with an overlap rate is 50%. Only the first half of the signal of Speaker One is affected by Speaker Two with 0dB, as the second half of the signal of Speaker One is not affected.

From the results, it is clear that the H-vector model allocates higher weight for the segments 5 to 10, and lower weights for the segments 1 to 5. This confirms the discussion in Section 2.4.1 that the interference from the second speaker can significantly affect the signal by competing with the target speaker for the same frequency band, which makes it difficult for the model to separate the elements of the two speakers. As a result, the higher weights are allocated to the clean features (in segments 5 to 10), and lower weights are allocated to the segments that are highly distorted by Speaker Two (segments 1 to 5).

However, the H-vector model in this chapter is trained to recognise all of the speakers from the input signal and, as shown in Figure 6.5(d) the H-vector model failed to capture the features for Speaker Two because the Speaker Two only occurs in segments 1 to 5, but these segments are allocated with very low weights.

The reason why the H-vector model cannot capture the features for the second speaker may be that the attention mechanism in the H-vector model cannot focus on two speakers. As discussed in Section 3.3, the computation process of both of the frame-level and segment-level attention mechanisms only provides one group of attention weights for the signal, and the attention weight values are computed by the softmax function so that they can be summed to one. It may be difficult for the model to capture multi-speaker information when the training target is to recognise all of the present speakers. In the next section, this problem will be addressed by a new model architecture.

6.3.4 Summary

This section proposed an adaption of the developed hierarchical attention network to the weakly supervised speaker identification task. The reason for using the H-vector model for the weakly supervised speaker identification task is that the hierarchical structure is shown to have greater robustness in noisy conditions. It is assumed that the H-vector model can also deliver better performance in a multi-speaker environment.

In Section 6.3.1, new datasets are constructed from the clean signals from the Voxceleb1 and SWBC datasets. In order to comprehensively evaluate the performance of the model, the constructed datasets represent two scenarios: Concat and Overlap. The Concat scenario contains a random numbers of speakers in the same input utterance without any overlap. The Overlap scenario contains a random number of speakers totally overlapped in a single input utterance. The model is evaluated with both small and large amounts of training data from both of the parent datasets, leading to the construction of a total of eight child datasets. The X-vector and attentive X-vector are selected as the baselines. The goal is to evaluate the effectiveness of the attention mechanism and the hierarchical structure. The results obtained shown in Section 6.3.2 reveal that the H-vector with a sliding window obtained the best results, reaching 6.48% and 14.88% EERs in the Concat and Overlap scenarios of SWBC-L dataset ("Multiple" test condition). This is due to the effectiveness of the hierarchical structure and the use of the sliding window allows the model to capture more information. The results in the Concat scenario are better than those obtained in the Overlap scenario due to the fact that overlapped speakers can significantly affect the features, making it difficult for the model to distinguish between the elements. When there is minimum training data available, the H-vector can still perform better than the baselines.

In order to intuitively observe and analyse how the attention mechanism works, Section 6.3.3 offers a visualisation of the segment-level attention weights. From the comparison of the visualised attention weights in Section 3.5.3, the H-vector model clearly failed to capture the features from the second speaker that occurred in the input utterance. This is due to the fact that the computational process of the attention mechanism in the H-vector model may not capture the information of multi-speaker features.

6.4 The Hierarchical Transformer Network

In the previous section, the H-vector model was used for weakly supervised speaker identification. However, the results in Sections 6.3.2 and 6.3.3 show that the attention mechanism in the H-vector model cannot capture multiple speaker features from a single input utterance. In this section, a new network architecture, called the hierarchical transformer network, is suggested as a way to address this issue.

6.4.1 Model Architecture

Figure 6.6 shows the architecture of the proposed hierarchical transformer network (T-vector). The model consists of a global TDNN layer, a positional encoding operation which is based on the sinusoidal positional encoding (Vaswani et al. 2017) that will be discussed later in this section, frame-level encoders, a segment-level encoder, and two fully connected layers as a classifier.

Given the input acoustic frame vector sequence, a TDNN layer is used, and the output is then added to the positional encoding of the original input, the output is denoted as $\boldsymbol{S} \in \mathcal{R}^{T \times F}$, where T represents the sequence length and F is the dimension of the frequency axis.

In the frame-level encoders, similar to the hierarchical attention network, the input sequence $\boldsymbol{S}$ is divided into M segments: $\{\mathbf{S}_1, \mathbf{S}_2, \cdots, \mathbf{S}_M\}$ using a sliding window with length L_{win} and step size L_{step}. Then, M frame-level encoders are used, compressing M segments into M segment vectors $\boldsymbol{e}_{S_m}, m \in \{1, 2...M\}$. Each frame-level encoder contains L transformer encoder blocks and a statistics pooling operation. The frame-level encoder blocks share weights, and are connected using memories of hidden states, which will be discussed later in this section.

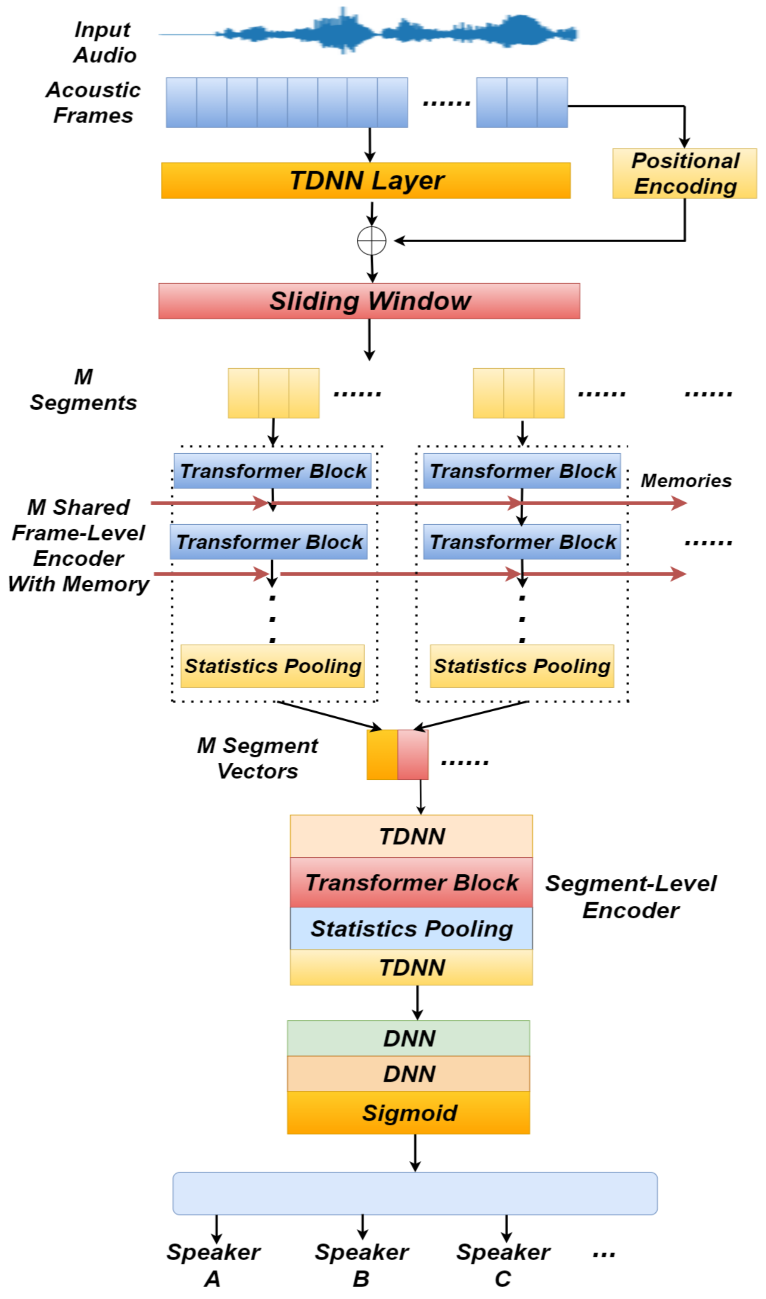


Figure 6.6: *The architecture of the hierarchical transformer network.*

After obtaining the segment vector sequence $\boldsymbol{E}_S \in \mathcal{R}^{M \times E_S}$, the segment-level encoder uses a TDNN layer followed by a transformer encoder block, memory is not used in the segment-level encoder. Another TDNN layer and a statistics pooling are used to compress the segment vector sequence into a single vector that represents the whole input sequence, which is referred to as the utterance vector. The speaker identity classifier is constructed using a two-layer MLP followed by a sigmoid activation function.

The speaker identity scores are the output vectors which contains the scores (between 1 and 0) for each speaker. The model is trained using the loss function that was introduced in Section 6.2.

From the above description, one can observe some key attributes of the architecture of the T-vector model. Firstly, it still makes use of the hierarchical structure that was introduced in Section 3.2. The experimental results show that the hierarchical structure can capture the local and global features of the input signal, rather than just the global attention mechanism that only focuses on the global features. As a result, the hierarchical structure is reused, but different architectures for each component within the structure are applied.

The key differences between the proposed hierarchical transformer network and the hierarchical attention network are the transformer block and the memory mechanism. The transformer block makes use of the multi-head attention that may help the model to better capture the features for multiple speakers. The memory mechanism can build a connection between each frame-level encoder by sharing the information across the whole input sequence. The details of each component are introduced in the following sections.

Transformer Encoder Block

As shown in Figure 6.6, the most important part of the T-vector model is the replacement of the TDNN and GRU layers with the transformer encoder block. This section and the next will introduce the details of the transformer encoder block.

The architecture of the transformer was introduced in Section 2.3.2. One of the most important components is the transformer encoder block. Figure 6.7 shows the architecture of one transformer encoder block where, for each block, a multi-head

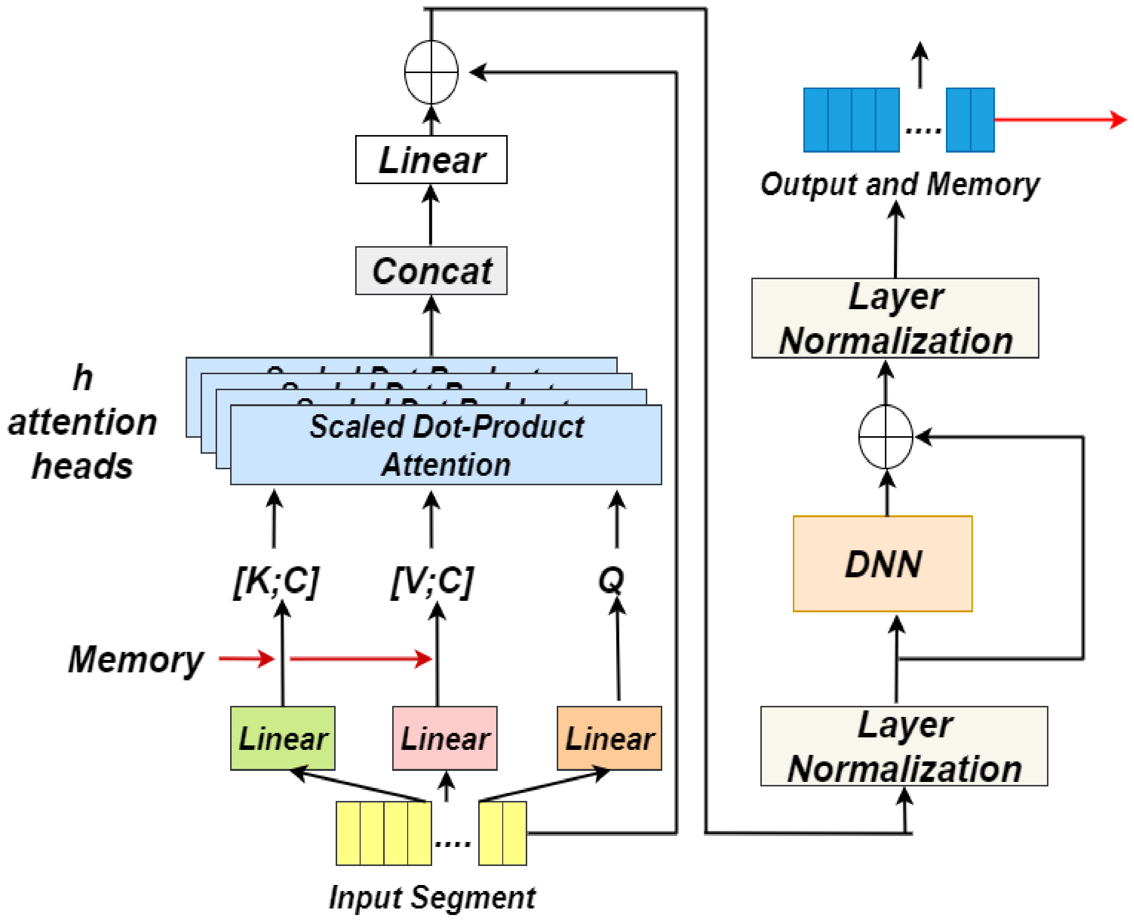


Figure 6.7: *The architecture of the transformer encoder block with memories (Vaswani et al. 2017, Dai et al. 2019).*

attention layer is used for the input segment (Vaswani et al. 2017). Layer normalisation (Ba et al. 2016) is used after the residual connection (He et al. 2016). The output is fed into a DNN layer, and then the same layer normalisation is used. The output is used as the input to the next block and the memory for the next frame-level encoder.

As discussed in Section 6.3.3, the reason why the attention mechanism in the hierarchical attention network cannot capture the features from multiple speakers maybe that the computation of the attention mechanism can only focus on one speaker. In order to address this issue, the multi-head attention mechanism is used.

As discussed in Section 2.3.2, the multi-head attention and the transformer were first developed in the field of neural machine translation. A fundamental feature of the multi-head attention mechanism is that it computes the attention weights for the input signal several times in parallel. Each attention mechanism, called an attention head, focuses on one part of the input, allowing the array to cover all the different features in the input signal. For speaker recognition, especially in weakly supervised speaker identification, the multi-head attention mechanism is assumed to be able to capture different speaker features with different attention heads.

To achieve this, the input signal is initially linearly transformed into three different sequences: Query ($\boldsymbol{Q}$), Key ($\boldsymbol{K}$) and Value ($\boldsymbol{V}$), each of them have the same dimensionality as the input signal. The purpose is to compute the attention weights multiple times in parallel. Equation 6.2 (from Vaswani et al. (2017)) shows an example of the the computational process for one attention head.

$$\boldsymbol{h}_i = \text{Attention-Head}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{Softmax}(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d}})\boldsymbol{V} \tag{6.2}$$

In order to compute multiple attention heads, each attention head is computed using $\boldsymbol{K}$, $\boldsymbol{Q}$ and $\boldsymbol{V}$ with the corresponding weight parameters $\boldsymbol{W}_i^K$, $\boldsymbol{W}_i^Q$ and $\boldsymbol{W}_i^V$. This is shown in Equation 6.3. The results of each attention head are then combined using a parameter matrix $\boldsymbol{W}^O$.

$$\begin{aligned} \text{MultiHead}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) &= [\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_h]\boldsymbol{W}^O \\ \boldsymbol{h}_i &= \text{Attention-Head}(\boldsymbol{Q}\boldsymbol{W}_i^Q, \boldsymbol{K}\boldsymbol{W}_i^K, \boldsymbol{V}\boldsymbol{W}_i^V) \end{aligned} \tag{6.3}$$

Figure 6.7 also shows how the layer normalisation (Ba et al. 2016) and the residual connection (He et al. 2016) are used. The layer normalisation normalises the learned features, a process that has been shown to have a greater level of robustness when processing noisy speech signals (India Massana et al. 2019). The residual connection, discussed in Section 2.2.4, can make the training of the model faster and more accurately.

Combining all of the components discussed above, the transformer blocks are used as the basic component of the hierarchical transformer network. Another important property of the T-vector model is the memory mechanism, which will be introduced in the next section.

Multi-Head Attention With Memory

The memory mechanism provides the information that can be shared across the whole sequence. The idea comes from the RNN architecture that was discussed in Section 2.1.3. Different frame-level encoders take each segment as input, computing the attention weights and summarising them into a single vector. When processing each segment, the input of the frame-level encoders take only the current segment, no information from the previous segments is provided. In the weakly supervised speaker identification task, the input utterances may contain multiple overlapped speakers, which means that the features in the early segments may be difficult to capture, though this becomes easier in the later segments. The memory mechanism provides a connection between the frame-level encoders to capture those features that are located in different segments and share them across all the frame-level encoders. This is similar to the hidden states in the RNN model that store and share the information across all of the time steps to make the model better capture long term memory. The detailed computation process of the memory mechanism can be found in Appendix A.7.2.

The last component is the positional encoding (Takase & Okazaki 2019), as illustrated in Figure 6.7 and discussed in Section 2.3.2, which is deployed before the sequence is split into segments in order to provide positional information for each frame to the frame-level encoders. For example, when splitting the input utterance into segments, originally there is no positional information provided. The frame-level encoders, built based on the transformer block, use each segment as input. Within each frame-level encoder, as the transformer encoder block processes the sequence information in parallel, it does not have the sequential mechanism, such as that in the RNN, to deal with the sequence input. This effect is also indicated by Vaswani et al. (2017). As a result, the sequence information within each segment will not be captured.

A simple but efficient way to solve this problem is to assign a unique code for each frame so that the transformer encoder block in each of the frame-level encoders will make use of the sequence information. The T-vector model makes use of the sinusoidal positional encoding (Vaswani et al. 2017) technique. The computational process can be found in Appendix A.7.2.

6.4.2 Experiment Setup

The T-vector model is an updated architecture from the H-vector. Therefore the experimental setups here are identical to those laid out in Section 6.3.1.

In terms of baseline selection, the H-vector is used in addition to the X-vector and attentive X-vectors, as a comparison between H-vector and T-vector can reveal the performance of the multi-head attention and the memory mechanisms. An additional baseline, the S-vector, is also selected (Katta et al. 2020). The S-vector makes use of the same transformer encoder block as the basic component of the model. The difference between S-vector and T-vector is that S-vector does not have the hierarchical structure and the memory mechanism, but is rather based on the X-vector architecture, replacing the TDNN layers with multiple transformer blocks. This makes the S-vector suitable for evaluating the performance of the hierarchical structure and the memory mechanism. The implementation details of the T-vector model can be found in Appendix A.7.1.

6.4.3 Results and Discussion

Performance for the Concat Scenario

Similar to the evaluation process laid out in Section 6.3.2, the Concat scenario is used first, as it is an simpler scenario. Figure 6.8 shows the results obtained using the five models (X-vector, attentive X-vector, H-vector, S-vector, T-vector) under different test conditions (1, 2, 3 or multiple speakers) on the four designed datasets (SWBC-S, SWBC-L, Vox-S and Vox-L). For all of the figures, the x-axis represents the number of

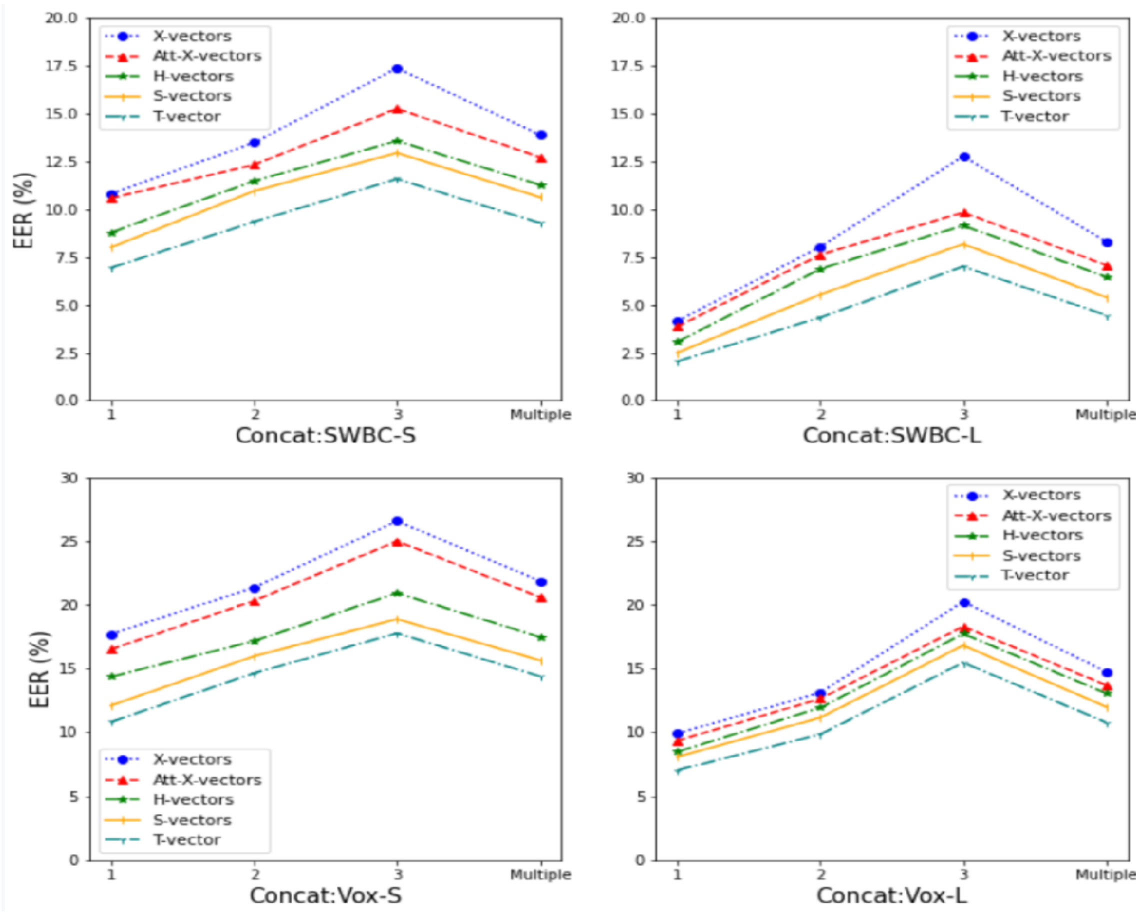


Figure 6.8: *The EERs (%) obtained using the five models in different test conditions on the four designed datasets: SWBC-S, SWBC-L, Vox-S and Vox-L in Concat Scenario. For all of the figures, the x-axis represents the number of speakers in test utterance.*

speakers in a test utterance. In the T-vector model, the window size L_{win} is 20 frames, the step size L_{step} is 10 frames. For the H-vector model, the window size L_{win} is also 20 frames, the step size L_{step} is 10 frames.

The T-vector model performed better than the four baselines under all test conditions, showing robustness when the training data is small by reaching 13.3% and 10.56% relative improvements over H-vector and S-vector respectively, in the SWBC-S dataset in the Concat scenario. Compared with the results obtained by the H-vector, the improvement of the T-vector may come from the use of multi-head attention and the memory mechanism.

Compared with the results obtained by the S-vector, the improvement of T-vectors may come from the use of a hierarchical structure with the memory mechanism. Instead of processing the whole input sequence, the T-vector makes use of the hierarchical structure that can better capture both local and global features. The memory mechanism guarantees the information of each speaker is shared across all frame-level encoders. Compared with the X-vector and attentive X-vector baselines, the improvements delivered by the T-vectors comes from the use of the multi-head attention mechanism, which captures overlapped speaker information better than the TDNN layers.

Similar to the results in Figure 6.3, the best results for any of the models are obtained when the number of speakers is one, and the worst case is when there are three speakers. This is due to complexity of the data. When the number of speakers increases, the models are difficult to capture their features.

Performance for the Overlap Scenario

The Overlap scenario is a more difficult scenario as the speakers are completely overlapped in the input utterances. Figure 6.9 shows the results obtained using the five models (X-vector, attentive X-vector, H-vector, S-vector, T-vector) under different test conditions (1, 2, 3 or multiple speakers) on the four datasets (SWBC-S, SWBC-L, Vox-S and Vox-L). For all figures, the x-axis represents the number of speakers in the test utterance. Once again, the settings for the T-vector model are: window size L_{win} is 20 frames, step size L_{step} is 10 frames.

It is visible that the results obtained in the Overlap scenario are worse than those from the Concat scenario. However, there remains a gap in performance between the T-vector model and the baselines, particularly for the SWBC-L dataset where it attained an EER of 28.1%. The gap between X-vector and the other models is significant because no attention mechanism is used in the X-vector model. It treats each frame as being equally important. In the Overlap scenario, where the speaker features are overlapped, the attention mechanism is necessary in order to discard the features that

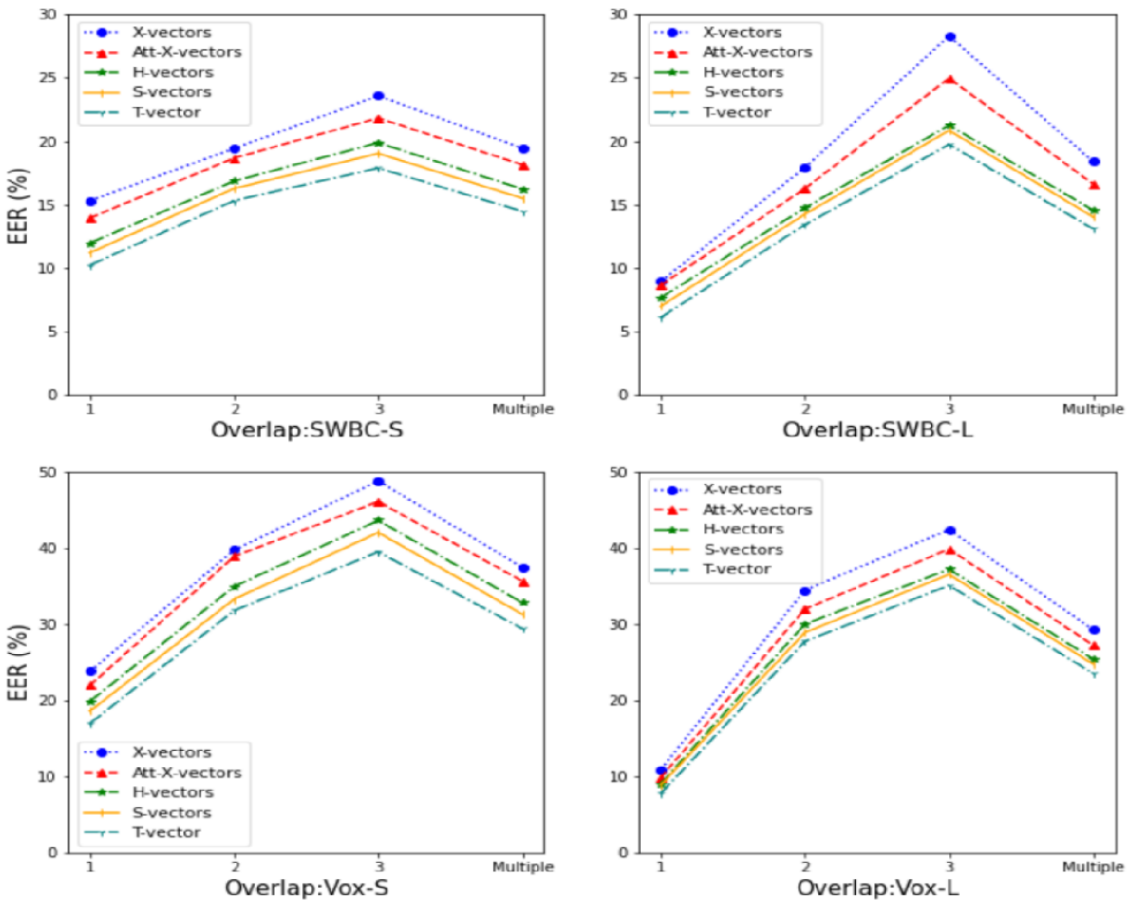


Figure 6.9: *The EERs (%) obtained using the five models under different test conditions on the four designed datasets: SWBC-S, SWBC-L, Vox-S and Vox-L in Overlap scenario. For all the figures, the x-axis represents the number of speakers in the test utterance.*

are highly distorted.

It is obvious that all results from the SWBC based dataset are better than those from Voxceleb1, an effect that can be observed in Figures 6.3, 6.4, 6.8 and 6.8. As discussed in Section 6.3.1, the total number of speakers are different in that the SWBC dataset contains only 254 speakers, while Voxceleb1 contains 1,000 speakers. When the number of speakers increases, it is more difficult for the model to be trained.

Table 6.4: *The EERs (%), with and without using the memory mechanism in the T-vector model. The window size ranges from 20 to 30 frames.*

Data Type	**Memory**	**Window Size**	**EER (%)**			
			SWBC-S	SWBC-L	Vox-S	Vox-L
Concat	**With**	20	9.28	4.46	14.41	10.79
		25	**8.97**	**4.04**	**13.97**	**10.25**
		30	9.05	4.20	14.30	10.49
	Without	20	10.31	5.52	16.02	11.64
		25	10.04	5.18	15.54	10.98
		30	10.34	5.37	15.85	11.47
Overlap	**With**	20	14.48	13.10	29.38	23.51
		25	**14.07**	12.96	**28.91**	23.25
		30	14.13	**12.77**	29.04	**23.08**
	Without	20	15.85	14.25	31.04	24.48
		25	15.25	13.99	30.24	23.80
		30	15.71	13.75	30.09	23.66

The Effectiveness of the Memory Mechanism

The memory mechanism in the T-vector model is one of the most important components. In order to evaluate the effectiveness of this memory mechanism, Table 6.4 shows the results from both with and without the use of the memory mechanism when using different window sizes (L_{win}). Universally and across both scenarios, the best results are obtained when using the memory mechanism. It significantly improves the performance of the T-vector, reaching 10.6% and 7.7% relative improvement for the two scenarios respectively, with the SWBC-S dataset when the window size is 25 frames. It shows that reusing information from previous segments allows the model to better capture long-term speaker information.

The results reveal a similar effect as seen in Section 6.3.2 whereby the window size affects performance; a larger window size leads to a higher level of performance, though this does peak eventually, from whence it declines. This is because the window size controls the amount of information input to the frame-level encoders; a small window size leads to insufficient information, while a window size that is too large leads to irrelevant information, thereby negatively influencing the ability of the network to capture pertinent features.

In addition, the window size can also affect the performance of the memory mechanism as the window size not only controls the amount of information that is input to, and captured by, the frame-level encoder, but also controls the amount of information that is stored and shared in memory. When the window size is large, irrelevant information is stored and shared across all of the frame-level encoders that will reduce the ability of some of the encoders to extract useful information. Conversely, when the window size is small, there will be insufficient useful information that is stored and shared.

The results shown in Table 6.4 confirm that the connection between the different frame-level encoders provided by the memory mechanism can improve the performance in a weakly supervised speaker identification task as the connections allow each frame-level encoder to share information of difference segments that can allow the model to better capture the speaker features located in different segments.

6.4.4 Analysis

The Open-Set Mode

As discussed in Section 6.2, the weakly supervised speaker identification task can be adapted into an open-set scenario in which the total number of speakers is not fixed, and the final layer of the network has an additional dimension to predict the scores for the unknown speakers. In order to evaluate the T-vector and H-vector models in this open-set scenario, 200 utterances spoken by 50 speakers are selected from the TIMIT dataset. When training the two models, the 200 utterances were part of the data construction process for the test set, and the labels for the 50 speakers are set to the same label, namely the unknown speaker.

Table 6.5 shows the results for the closed-set and open-set scenarios for the two models. It is obvious that the results obtained in the open-set scenario are worse

Table 6.5: *The EERs (%) of the T-vector and the H-vector models in close- and open-set weakly supervised speaker identification tasks. The window size for the two models is kept at 25 frames, and the step size is kept at 10 frames.*

Data Type	**Model**	**Mode**	**EER (%)**			
			SWBC-S	SWBC-L	Vox-S	Vox-L
Concat	**H-vector**	**Open**	13.07	8.03	18.85	15.02
		Close	11.69	6.59	17.81	13.29
	T-vector	**Open**	9.98	5.25	15.11	11.48
		Close	8.97	4.04	13.97	10.25
Overlap	**H-vector**	**Open**	17.22	16.91	34.58	27.14
		Close	15.99	15.58	32.26	25.94
	T-vector	**Open**	15.43	24.52	30.62	25.17
		Close	14.07	23.96	28.91	23.25

than those for the closed-set scenario, simply. In the original training set, there is a fixed number of speakers, all with speaker identity labels. However, in the open-set scenario, the model needs to identify all of the original speakers and the extra speakers, and classify all of the 50 speakers into the unknown category.

Compared to the results obtained with the T-vector and H-vector models, it is obvious that the T-vector can obtain better results than the H-vector in all of the datasets in both open-set and closed-set scenarios. For example, with the SWBC-S dataset in the Concat scenario, even though there are only 6,000 utterances in the closed-set scenario, the T-vector can provide an EER of 11.69%; in the open set scenario, the results are not as good, but the T-vector model still managed a 13.07% EER, compared with the H-vector model that reached 17.22%. As discussed in the previous sections, the reason is that the T-vector contains the multi-head attention mechanism that can capture the properties of multiple speakers, and the memory mechanism allows each frame-level encoder to connect and share information across all frame-level encoders.

Table 6.6: *The EERs (%) of the proposed T-vector model and the H-vector model in Concat, Overlap scenarios, as well as the scenario that the input utterances have random overlap rate for each speaker.*

Model	**Mode**	**EER (%)**			
		SWBC-S	SWBC-L	Vox-S	Vox-L
H-vector	**Concat**	11.69	6.59	17.81	13.29
	Overlap	15.99	15.58	32.26	25.94
	Random	13.25	10.88	25.72	19.28
T-vector	**Concat**	8.97	4.04	13.97	10.25
	Overlap	14.07	12.96	28.91	23.25
	Random	11.44	9.24	23.20	18.79

The proposed weakly supervised training strategy for a speaker identification task can be adapted into an open-set scenario and, although the results obtained are worse than those for the closed-set scenario, the proposed T-vector model can still outperform the H-vector model.

Multiple Speakers With Random Overlap Rates

As discussed in Section 6.3.1, the Concat and Overlap scenarios of the constructed datasets offer an easy and a difficult scenario, for the models to recognise multiple speakers. The Concat scenario is easy in that none of the speakers overlap in the input signal, whereas the Overlap scenario is difficult because multiple speakers are totally overlapped in the signal. These two scenarios are used to evaluate the upper and lower bounds of the proposed models, because the weakly supervised speaker identification task is a newly proposed task.

In order to simulate the situation in real-world conditions, another scenario also needs to be considered in which multiple speakers have random overlap rates in the input signal. This scenario, known as the Random scenario, is closer to a real-world situation and can be used to evaluate the performance of the models in close-to-real-world conditions.

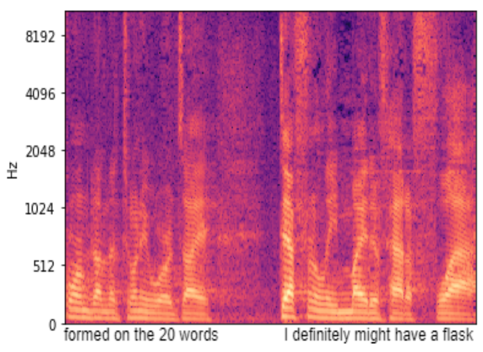
Table 6.6 shows the results obtained with the H-vector and T-vector models in the Concat, Overlap and Random scenarios with the four constructed datasets. It is obvious that the results in the Random scenario are better than those in the Overlap scenario, but worse than those in the Concat scenario. This is because in the Random scenario, the speakers are overlapped in some segments but not in others, and the model can capture more speaker features from the segments where there is no overlapping. The T-vector delivered better performance than the H-vector model in the Random scenario. For example, in the Vox-S dataset in the Random scenario, the T-vector reached 23.20% EER while the H-vector reached 25.72% EER for reasons discussed in previous sections, that the T-vector has the advantage of the multi-head attention mechanism and the memory mechanism to better capture the information of multiple speakers in a single utterance.

Overall, the results in Table 6.6 show the T-vector model performs better than the H-vector model in all of the three data construction scenarios.

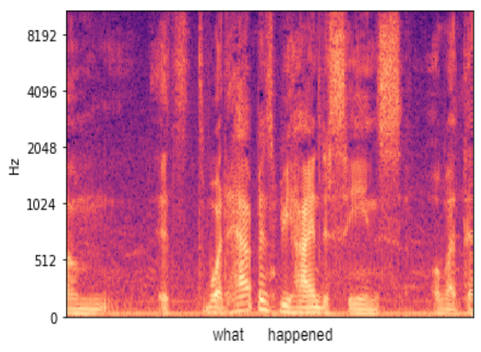
Analysis of the Multi-Head Attention

Following the analysis techniques in Sections 6.3.3 and 3.5.3, Figure 6.10 shows a visualisation of the multi-head attention weights. In Figures 6.10(a), (b) and (c), the signals of Speaker One, Speaker Two and the mixed speech signals are the same as those in Figure 6.5, Section 6.3.3, thereby providing a comparison between the multi-head attention in the T-vector and the attention mechanism used in the H-vector models.

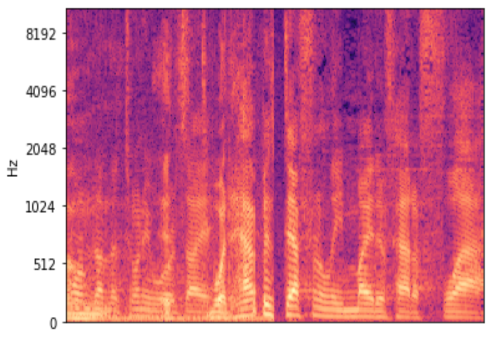
Recall that the first half of the mixed signal in Figure 6.10(c) is a mixture of the Speaker One signal (Figure 6.10(a)) and the first half of the Speaker Two signal (Figure 6.10(b)). The second half of the signal in Figure 6.10(c) only contains the second half of the signal of Speaker One. In other words, the mixed signal in Figure 6.10(c) only overlaps the signal from Speaker Two in the first half, while the second half contains



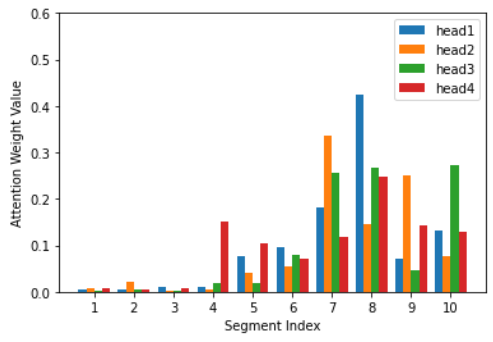
(a) Spectrogram of Speaker One



(b) Spectrogram of Speaker Two



(c) Spectrogram of Mixed Signal



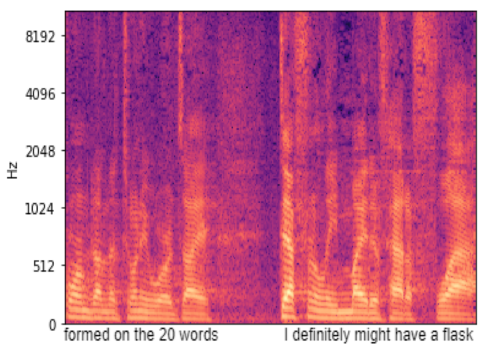
(d) Attention Weights in T-vector

Figure 6.10: *Multi-head attention weights. (a) the spectrogram of Speaker One, (b) the spectrogram of Speaker Two, (c) the spectrogram of Mixed Signal, (d) the multi-head attention weights of the in T-vector model.*

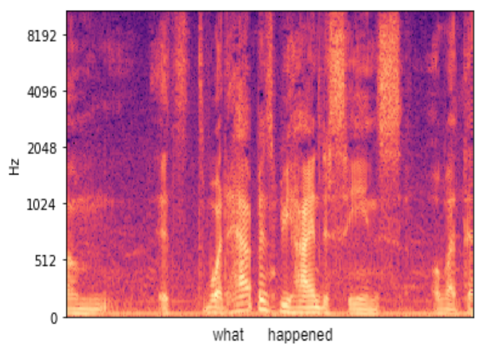
the clean signal of Speaker One. This setting is used to evaluate whether the attention mechanism can capture the features of multiple speakers.

The multi-head attention weights are visualised in Figure 6.10(d), with each color representing the weight of one of the four attention heads. In this illustration, the same technique is used as in Sections 6.3.3 and 3.5.3 in that the attention values are averaged using a sliding window into ten weight values. This is for better observation and comparison with the attention weights shown in Sections 6.3.3 and 3.5.3 as they use the same target speaker (Speaker One, shown in Figure 6.10(a)).

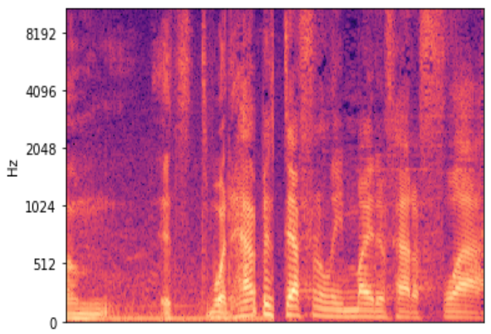
It is obvious from Figure 6.10(d) that the different attention heads concentrate on different parts of the signal. Attention head 1 (blue) assigned the highest weight to the



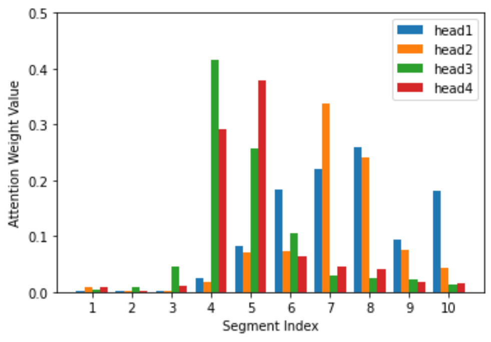
(a) Spectrogram of Speaker One



(b) Spectrogram of Speaker Two



(c) Spectrogram of Mixed Signal



(d) Attention Weights in T-vector

Figure 6.11: *Attention weights. (a) the spectrogram of Speaker One, (b) the spectrogram of Speaker Two, (c) the spectrogram of Mixed Signal, (d) the multi-head attention weights of in T-vector model.*

segment 8, while attention head 2 (orange) assigned the segment 7 with the highest weight. Combined with the analysis in Section 6.3.3, this is because segments 7 and 8 contain the most important features about Speaker One. This phenomenon can also be confirmed by the previous illustration in Figures 3.5 and 6.5. Attention head 3 (green) not only allocates high weights to the segments 7 and 8, but also the segment 10. This shows that the 10th segment also contains information from Speaker One, and it was captured by attention head 3. All of these three attention heads captures the information of Speaker One.

Attention head 4 (red) also allocated the highest weight to the segment 7, but the segment 4 also shares a larger weight compared with the other attention heads. The segment 4 is located in the mixed signal of Speakers One and Two, which suggests that, while attention head 4 clearly captured useful information from the segment 4, the fact that it comes from the segment 4 within the mixed signal means this alone is insufficient to prove that the captured information is from Speaker Two.

In order to analyse what information is captured by attention head 4, and whether the multi-head attention mechanism can capture multiple speaker information from the signal, Figure 6.11 shows the Concat scenario of the two signals. Figure 6.11(c) contains the mixed signal of the two speakers in the Concat scenario, which means the first half of the signal in Figure 6.11(c) only contains the signal from Speaker Two, and the second half of the signal in Figure 6.11(c) only contains the signal from Speaker One. In other words, the mixed part of the signal in Figure 6.10(c) (the first half of it) is replaced with the signal of Speaker Two only.

From the visualisation of the multi-head attention weights in Figure 6.11(d), the attention head 1 and head 2 are mostly concentrated on segments 5 to 10; head 3 and head 4 are mostly concentrated on the information in the segments 4 and 5 that only carry the information from Speaker Two. This shows that the multi-head attention can capture the features from different speakers in one input signal and, furthermore, this confirms the assumption made in Section 6.4.1 that the different attention heads can concentrate on different speakers. This property of the multi-head attention mechanism used in the T-vector model is one of the reasons why the T-vector model can obtain better results in the weakly supervised speaker identification task.

6.4.5 Summary

In this section, in order to improve performance in the weakly supervised speaker identification task, the hierarchical transformer network is introduced. Even though the H-vector model can obtain better results compared to the X-vector and attentive X-vector models, it is unable to capture features from multiple speakers in one input utterance.

In order to address this problem, the proposed T-vector makes several improvements based on the architectural structure of the H-vector. As discussed in Section 6.4.1, the transformer encoder block is used as the basic component of the architecture of the T-vector model. The most important part of the transformer encoder block is the multi-head attention mechanism, used to compute multiple attention weights for the same input signal in parallel and assumed to be able to capture information from multiple speakers from a single input signal. A memory mechanism is used connect each of the frame-level encoders to build a link to store and share information.

As discussed in Section 6.4.2, the experimental setup is the same as that used in Section 6.3.1, including the data construction process in order to provide a comparison with the H-vector model. The S-vector model is used as an additional baseline. It makes use of the multi-head attention mechanism, but it has no hierarchical structure. It can be used to compare with the T-vector model in order to demonstrate the effectiveness of the hierarchical structure and the memory mechanism.

The experimental results in Figures 6.8 and 6.9 shows that the T-vector outperforms all four baselines (X-vector, attentive X-vector, S-vector and H-vector) in both the Concat and Overlap scenarios. It can reach 4.89% and 13.97% EERs in Concat and Overlap scenarios for SWBC-L dataset ("Multiple" test condition). Furthermore, Table 6.4 shows that the memory mechanism is essential for the T-vector model in that a relative improvement of over 3% can be obtained using the memory mechanism in almost all of the constructed datasets. Different window sizes also affect the performance, as the length of the sliding window controls the amount of information input to the frame-level encoders and the amount of information that is shared by the memory mechanism.

The results in Table 6.5 show the proposed weakly supervised speaker identification task can be adapted into an open-set scenario in which the T-vector can obtain better results than the H-vector model in the open-set scenario. To simulate the scenario in real-world conditions, Table 6.6 shows the proposed T-vector model can also out-perform the H-vector model in the Random scenario, in which the speakers are overlapped with a random overlap rate in the input utterance. The visualisation of the multi-head attention weights shown in Figures 6.10 and 6.11 confirms that the multi-head attention mechanism can capture the features of different speakers from a single input utterance. This property is the reason why the T-vector model can out-perform the H-vector model in almost all test conditions.

6.5 Conclusion

As discussed in Section 2.4.3, in some real-world scenarios, when there is an unknown number of speakers in one input signal, it may be useful to identify all the speakers. A way to achieve this is to manually annotate all of the speakers with their time information from the mixed signal. However, this is time consuming and expensive. It is desirable instead to develop an approach that can directly make use of the coarse grained labels of the multi-speaker data. This was called weakly supervised speaker identification.

As discussed in Section 2.4.3, previous work addressed this problem but that still required the pre-training of a speaker diarisation model and an i-vector model. The end-to-end weakly supervised speaker identification approach was thus undeveloped. This chapter proposed an approach that can directly train a neural network model for speaker identification in a weakly supervised manner.

Since there are no previous baselines that can be referenced, Section 6.2 defined the weakly supervised speaker identification task. The data contains the utterances of a random number of speakers and only the utterance-level labels are available. The goal for this task is to recognise all of the speaker identities. This task can be separated as either a closed-set or open-set scenario. In the closed-set scenario, the total number of speakers is fixed, while in the open-set scenario, the extra speakers can also be recognised and classified as unknown speakers.

The hierarchical attention network, the H-vector model developed in Chapter 3, is adapted to serve for the weakly supervised speaker identification task. The H-vector model was shown to have better noise robustness in speaker recognition under noisy conditions compared to the X-vector and the attentive X-vectors. The hierarchical structure can capture both local and global features, instead of only focusing on the global features in the attentive X-vector. In order to comprehensively evaluate the H-vector model in this task, eight datasets were constructed into Concat and Overlap scenarios, based on the SWBC and Voxceleb1 datasets.

The Concat scenario contains multiple speakers in the input utterance with no overlap, while the Overlap scenario contains multiple speakers who are completely overlapped in the input signal. These two scenarios are designed to test the upper and lower bounds of the model performance when applied to the weakly supervised speaker identification task. The results obtained show that the H-vector model can out-perform the X-vector and attentive X-vector models for both the Concat and Overlap scenarios. The window and step sizes can both affect the performance of the H-vector as they control the information that is input to the frame-level and segment-level encoders. Although the H-vector delivers a better performance than the two baselines, when visualising the attention weights, it can be observed that the attention mechanism used in the H-vector model cannot effectively capture the features of multiple speakers because the attention weights vector is only computed once in the H-vector model, so it is difficult for the attention mechanism to focus on multiple speakers at the same time.

In order to address this issue, the hierarchical transformer network (T-vector) model was proposed in Section 6.4. The T-vector model makes use of the multi-head attention mechanism, which computes the self-attention weight vectors for one input signal multiple times in parallel. This property is assumed to be able to capture the features of multiple speakers in a single input utterance. Another important component in the T-vector model is the memory mechanism, which is used to build a connection between each of the frame-level encoders, allowing the encoders to both store and share information across the network to better capture different speaker features. The experimental setup is the same as that in Section 6.3.1 to enable a better comparison. An additional baseline, named S-vector, is used to compare the effectiveness of the hierarchical structure used in the T- vector model. The results show that the T-vector model can obtain better performance compared with all four baselines in almost all conditions. The memory mechanism is shown to have a significant impact on the results obtained, which confirms the assumption that the memory mechanism can store and share information between each of the frame-level encoders to improve the performance. The experiments also show that the proposed weakly supervised speaker identification task can also be adapted into an open-set scenario. To simulate real-world conditions, the utterances are mixed with random overlap rates. The T-vector still out-performs the H-vector in these two scenarios. Finally, to show how the multi-head attention works, the attention weights of different heads are visualised. The results show that the different attention heads can concentrate on different speakers in one input utterance, confirming the assumption that this can improve performance further by having different attention heads focusing on different speakers at the same time.

Chapter 7

Conclusion and Future Work

Speaker recognition systems aim to recognise the speaker identities from their voices. Due to the large model capacity and strong feature extraction ability, deep neural networks out-perform the conventional GMM based i-vector system in speaker recognition. In Chapter 2, a literature review was conducted covering deep neural network architectures, input features and loss functions for speaker recognition systems, and recently developed deep neural network architectures for speaker recognition.

As discussed in Section 2.3.1, speech signals can be influenced by background noise, and the performance of deep neural networks can be affected by noisy conditions. One potential solution is the attention mechanism that can be built into a neural network to emphasise the important parts of the input while discarding irrelevant information. However, the current widely-used global attention mechanism is unable to overcome noisy interference. As discussed in Section 2.3.2, the global attention mechanism can only focus on some of the important features, and fails to pay enough attention to the local features.

In Chapter 3, a novel network architecture named the hierarchical attention network was proposed. It has a hierarchical structure that splits the attention mechanism into two levels: the frame level and the segment level. The frame-level encoder applies the frame-level attention that focuses on the local features, while the segment-level attention focus on the global features. The results show that the hierarchical attention network can deliver better performance than the X-vector and attentive X-vector models under various noisy conditions at different levels. The contributions and findings of Chapter 3 are:

- The proposed hierarchical attention network that can capture both local and global features to improve the model generalisation of the speaker recognition system. It can reach better performance when using the out-of-domain test set compared to the X-vector and attentive X-vector baselines.

- The hierarchical attention network can reach better performance compared to the baselines under various noise conditions.

- The Proposed hierarchical attention network delivers results comparable to the published state-of-the-art model, reached 4.28% EER on the Voxceleb1 test set.

Speech enhancement is another approach to overcome the noise interference. As discussed in Section 2.3.3, current speech enhancement methods are typically trained independently. This may cause a mismatch problem. The learned features of the speech enhancement model may not well match that are required by the speaker recognition model. In order to address the problem and improve the noise robustness of the speaker recognition system, Chapter 4 proposes two novel model architectures. The first method combined the speech enhancement model and the speaker recognition model, and a novel multi-stage attention mechanism is proposed to better filter out the noise influence. The second system further uses speaker embeddings to build a speaker dependent speech enhancement method to reach better noise robustness for specific target speakers. The contributions and findings of Chapter 4 are:

- The joint training framework can reach better noise robustness compared to baselines, with 91.1% speaker identification accuracy and 6.18% EER on the Voxceleb1 test set.

- The multi-stage attention mechanism can improve the noise robustness of the joint system in either the speech enhancement model or the speaker recognition model. The multi-stage attention used in the speech enhancement model can obtain better results, and the channel attention contributes the most among the three attention mechanisms.
- The speaker dependent speech enhancement model contains a novel residual auto-encoder architecture that concatenates the pre-trained speaker embeddings. It can reach better noise robustness compared to the speaker independent mode of the joint system. It is able to outperform the state-of-the-art methods, reaching 4.15% EER on the Voxceleb1 test set.

As discussed in Section 2.4.1, the interference from other speakers in the input signal can also affect the performance of the speaker recognition model. The widely used target speaker extraction method extracts the signal of the target speaker to overcome this problem. However, as discussed in Section 2.4.2, this may not be necessary, and extracting the speaker properties in an embedding space may be more efficient. Chapter 5 proposed an embedding de-mixing approach that not only filters out the influence of the interfering speaker, but can also extract the content or speaker properties using the pre-trained speaker or content embeddings. The contributions and findings of Chapter 5 are:

- The embedding de-mixing approach separates speaker and content properties in an embedding space rather than in a signal space.
- The proposed speaker embedding de-mixing approach can reduce the influence of the interfering speaker in a two-speaker signal by the pre-trained embeddings. The best configuration of the proposed approach can reach 96.9% and 91.3% speaker identification accuracies on TIMIT and MC-WSJ datasets, compared to the results obtained by the clean signals, which are 98.5% and 99.1%.

- The proposed content embedding de-mixing approach can reduce spoken content information from the interfering speaker using the corresponding pre-trained embedding. The best results can reach 93.3% spoken word identification accuracy on the Speech Command dataset, which is close to that obtained from the clean signals (96.5%).

- The proposed speaker and content de-mixing approach extracts speaker or content properties of the target speaker by the use of the pre-trained content or speaker embeddings. When extracting the speaker embeddings, the obtained embeddings can reach 82.2% speaker identification accuracy compared to that obtained from the clean signals (98.5%). When extracting the content embeddings, the obtained embeddings can reach 63.2% phone classification accuracy and 30.2% phone recognition error rate, compared to the clean results, which are 74.1% and 16.6%.

- One of the six proposed embedding de-mixing functions, named Separate-Concat method, achieved the best performances in almost all of the three scenarios. It can reach close embedding quality compared to the clean signals.

As discussed in Section 2.4.3, in some real-world scenarios, such as conversations or meetings, recognising all of the speaker identities from the multi-speaker signal is required. To achieve this, instead of hand annotating the speaker and positional labels for each speaker, a more efficient method is to use weakly supervised learning that can directly make use of the coarse grained labels. The current weakly supervised speaker identification approach requires a pre-trained speaker diarisation system that still requires manually annotated labels. Chapter 6 proposed the first end-to-end weakly supervised training of speaker identification approach, as well as two neural network architectures. The contributions and findings of Chapter 6 are:

- The first end-to-end weakly supervised speaker identification approach can directly learn from the utterance-level labels of the multi-speaker input signals.

- The proposed hierarchical attention network can be trained in a weakly supervised manner. It can reach better performance (equals or more than 3% relative improvements) than the X-vector and attentive X-vector baselines in all of the augmented datasets and test configurations.

- The proposed hierarchical transformer network can further improve the performance of the proposed H-vector by the multi-head attention mechanism and novel memory mechanism. The multi-head attention mechanism can capture the speaker features from multiple speakers in one input utterance. The memory mechanism can improve the performance by connecting different frame-level encoders.

7.1 Future Work

A straightforward extension to the work in Chapters 3 and 4 is to build the developed hierarchical attention network and the joint training system into one framework. As shown in these two chapters, the hierarchical structure can improve the noise robustness of the model by focusing on both the local and global features. The speech enhancement model in the joint training system can filter out noise interference and then feed the enhanced signal into the speaker recognition model. One possible extension is to use the hierarchical structure into the speech enhancement model and train it jointly with a speaker recognition model. In this way, the hierarchical structure can capture both local and global features, thus can improve the quality of the noise reduction process in the speech enhancement model.

A possible extension of the work in Chapter 5 is to explore the effectiveness of the de-mixing approach with three or more speakers. The proposed approach only considers a scenario with two speakers. However, in some real-world conditions, there may be more speakers speaking at the same time. De-mixing the speaker properties of all of the speakers into the embedding space may be useful for many downstream tasks.

As possible extension of the work in Chapter 6 is to consider when the input utterance contains more speakers. The proposed approach only evaluates up to a maximum of three speakers, whereas in a real-world situation, the number of speakers in a single utterance or a given period of time may be larger than three. Developing a technique that can recognise all of them will be helpful in many applications.

Bibliography

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M. et al. (2016), TensorFlow: a system for large-scale machine learning, *in* 'Proceedings of the 12th USENIX conference on Operating Systems Design and Implementation', pp. 265–283.

Alam, M., Vidyaratne, L. & Iftekharuddin, K. M. (2015), Novel hierarchical cellular simultaneous recurrent neural network for object detection, *in* '2015 International Joint Conference on Neural Networks (IJCNN)', IEEE, pp. 1–7.

Alexandra Canavan, David Graff, G. Z. (2001), 'CALLHOME American English Speech', `https://catalog.ldc.upenn.edu/LDC97S42`.

Aloysius, N. & Geetha, M. (2017), A review on deep convolutional neural networks, *in* '2017 International Conference on Communication and Signal Processing (ICCSP)', IEEE, pp. 0588–0592.

Amari, S. et al. (2003), *The handbook of brain theory and neural networks*, MIT press.

An, N. N., Thanh, N. Q. & Liu, Y. (2019), 'Deep CNNs with self-attention for speaker identification', *IEEE Access* **7**, 85327–85337.

Anguera, X., Bozonnet, S., Evans, N., Fredouille, C., Friedland, G. & Vinyals, O. (2012), 'Speaker diarization: A review of recent research', *IEEE Transactions on Audio, Speech, and Language Processing* **20**(2), 356–370.

Anwar, S. & Barnes, N. (2019), Real image denoising with feature attention, *in* 'Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)', pp. 3155–3164.

Aronoff, S. et al. (1982), 'Classification accuracy: a user approach', *Photogrammetric Engineering and Remote Sensing* **48**(8), 1299–1307.

Ba, J. L., Kiros, J. R. & Hinton, G. E. (2016), 'Layer normalization', *arXiv preprint arXiv:1607.06450* .

Bahdanau, D., Cho, K. H. & Bengio, Y. (2015), Neural machine translation by jointly learning to align and translate, *in* '3rd International Conference on Learning Representations, ICLR 2015'.

Bai, Z. & Zhang, X.-L. (2021), 'Speaker recognition based on deep learning: An overview', *Neural Networks* .

Bengio, Y., Frasconi, P. & Simard, P. (1993), The problem of learning long-term dependencies in recurrent networks, *in* 'IEEE international conference on neural networks', IEEE, pp. 1183–1188.

Bottou, L. (2012), Stochastic gradient descent tricks, *in* 'Neural networks: Tricks of the trade', Springer, pp. 421–436.

Bouchard, G. (2007), Efficient bounds for the softmax function, applications to inference in hybrid models, *in* 'Presentation at the Workshop for Approximate Bayesian Inference in Continuous/Hybrid Systems at NIPS-07', Citeseer.

Bridle, J. S. (1989), Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters, *in* 'Proceedings of the 2nd International Conference on Neural Information Processing Systems (NIPS)', pp. 211–217.

Brownlee, J. (2017), 'Why one-hot encode data in machine learning', *Machine Learning Mastery* .

Cai, W., Chen, J. & Li, M. (2018), Exploring the encoding layer and loss function in end-to-end speaker and language recognition system, *in* 'Proc. Odyssey 2018 The Speaker and Language Recognition Workshop', pp. 74–81.

Campbell, J. P. (1997), 'Speaker recognition: A tutorial', *Proceedings of the IEEE* **85**(9), 1437–1462.

Campbell, W. M., Sturim, D. E. & Reynolds, D. A. (2006), 'Support vector machines using GMM supervectors for speaker verification', *IEEE signal processing letters* **13**(5), 308–311.

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

Chung, J., Gulcehre, C., Cho, K. & Bengio, Y. (2014), Empirical evaluation of gated recurrent neural networks on sequence modeling, *in* 'Neural Information Processing Systems (NIPS)'.

Cortes, C. & Vapnik, V. (1995), 'Support-vector networks', *Machine learning* **20**(3), 273–297.

Dai, Z., Yang, Z., Yang, Y., Carbonell, J. G., Le, Q. & Salakhutdinov, R. (2019), Transformer-XL: Attentive language models beyond a fixed-length context, *in* 'Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)', pp. 2978–2988.

David Graff, Kevin Walker, D. M. (2001), 'Switchboard Cellular Part 1 Audio', `https://catalog.ldc.upenn.edu/LDC2001S13`.

Davis, S. & Mermelstein, P. (1980), 'Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences', *IEEE transactions on acoustics, speech, and signal processing* **28**(4), 357–366.

Dehak, N., Dehak, R., Kenny, P., Brümmer, N., Ouellet, P. & Dumouchel, P. (2009), Support vector machines versus fast scoring in the low-dimensional total variability space for speaker verification, *in* 'Tenth Annual conference of the international speech communication association (Interspeech)'.

Dehak, N., Kenny, P. J., Dehak, R., Dumouchel, P. & Ouellet, P. (2010), 'Front-end factor analysis for speaker verification', *IEEE Transactions on Audio, Speech, and Language Processing* **19**(4), 788–798.

Deng, J., Guo, J., Xue, N. & Zafeiriou, S. (2019), Arcface: Additive angular margin loss for deep face recognition, *in* 'Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)', pp. 4690–4699.

Ding, S., Chen, T., Gong, X., Zha, W. & Wang, Z. (2020), 'Autospeech: Neural architecture search for speaker recognition', *Proc. Interspeech 2020* pp. 916–920.

Dong, L.-F., Gan, Y.-Z., Mao, X.-L., Yang, Y.-B. & Shen, C. (2018), Learning deep representations using convolutional auto-encoders with symmetric skip connections, *in* '2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)', IEEE, pp. 3006–3010.

Fontaine, V. & Bourlard, H. (1997), Speaker-dependent speech recognition based on phone-like units models-application to voice dialling, *in* '1997 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)', Vol. 2, IEEE, pp. 1527–1530.

Gál, V., Hámori, J., Roska, T., Bálya, D., Borostyánkői, Z., Brendel, M., Lotz, K., Négyessy, L., Orzó, L., Petrás, I. et al. (2004), 'Receptive field atlas and related CNN models', *International Journal of Bifurcation and Chaos* **14**(02), 551–584.

Garcia-Perera, L. P., Nolazco-Flores, J. A., Raj, B. & Stern, R. (2012), Optimization of the DET curve in speaker verification, *in* '2012 IEEE Spoken Language Technology Workshop (SLT)', IEEE, pp. 318–323.

Garcia-Romero, D. & McCree, A. (2014), Supervised domain adaptation for i-vector based speaker recognition, *in* '2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)', IEEE, pp. 4047–4051.

Garofolo, J. S., Lamel, L. F., Fisher, W. M., Fiscus, J. G. & Pallett, D. S. (1993), 'DARPA TIMIT acoustic-phonetic continous speech corpus CD-ROM. NIST speech disc 1-1.1', *NASA STI/Recon technical report n* **93**.

Goodfellow, I., Bengio, Y., Courville, A. & Bengio, Y. (2016), *Deep learning*, Vol. 1, MIT press Cambridge.

Greig, D. M., Porteous, B. T. & Seheult, A. H. (1989), 'Exact maximum a posteriori estimation for binary images', *Journal of the Royal Statistical Society: Series B (Methodological)* **51**(2), 271–279.

Hajibabaei, M. & Dai, D. (2018), 'Unified hypersphere embedding for speaker recognition', *arXiv preprint arXiv:1807.08312* .

Hansen, J. H. & Hasan, T. (2015), 'Speaker recognition by machines and humans: A tutorial review', *IEEE Signal processing magazine* **32**(6), 74–99.

Hao, X., Shan, C., Xu, Y., Sun, S. & Xie, L. (2019), An attention-based neural network approach for single channel speech enhancement, *in* 'ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)', IEEE, pp. 6895–6899.

Haykin, S. & Chen, Z. (2005), 'The cocktail party problem', *Neural computation* **17**(9), 1875–1902.

He, K., Zhang, X., Ren, S. & Sun, J. (2016), Deep residual learning for image recognition, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)', pp. 770–778.

Hinton, G. E. & Salakhutdinov, R. R. (2006), 'Reducing the dimensionality of data with neural networks', *science* **313**(5786), 504–507.

Hochreiter, S. & Schmidhuber, J. (1997), 'Long short-term memory', *Neural computation* **9**(8), 1735–1780.

Hojo, N., Ijima, Y. & Mizuno, H. (2016), An investigation of DNN-based speech synthesis using speaker codes., *in* 'Interspeech', pp. 2278–2282.

Huang, X. & Lee, K.-F. (1993), 'On speaker-independent, speaker-dependent, and speaker-adaptive speech recognition', *IEEE Transactions on Speech and Audio processing* **1**(2), 150–157.

Huang, Z., Wang, S. & Yu, K. (2018), Angular softmax for short-duration text-independent speaker verification., *in* 'Interspeech', pp. 3623–3627.

India Massana, M. À., Safari, P. & Hernando Pericás, F. J. (2019), Self multi-head attention for speaker recognition, *in* 'Interspeech 2019: the 20th Annual Conference of the International Speech Communication Association: 15-19 September 2019: Graz, Austria', International Speech Communication Association (ISCA), pp. 4305–4309.

Ioffe, S. & Szegedy, C. (2015), Batch normalization: Accelerating deep network training by reducing internal covariate shift, *in* 'International conference on machine learning (ICML)', PMLR, pp. 448–456.

Irum, A. & Salman, A. (2019), 'Speaker verification using deep neural networks: A review', *International Journal of Machine Learning and Computing* **9**(1).

Ito, Y. (1991), 'Representation of functions by superpositions of a step or sigmoid function and their applications to neural network theory', *Neural Networks* **4**(3), 385–394.

Jin, M. & Yoo, C. D. (2010), Speaker verification and identification, *in* 'Behavioral Biometrics for Human Identification: Intelligent Applications', IGI Global, pp. 264–289.

Johnson, D. H. (2006), 'Signal-to-noise ratio', *Scholarpedia* **1**(12), 2088.

Kalchbrenner, N., Grefenstette, E. & Blunsom, P. (2014), A convolutional neural network for modelling sentences, *in* '52nd Annual Meeting of the Association for Computational Linguistics (ACL)', Association for Computational Linguistics.

Kamath, U., Liu, J. & Whitaker, J. (2019), *Deep learning for NLP and speech recognition*, Vol. 84, Springer.

Karlik, B. & Olgac, A. V. (2011), 'Performance analysis of various activation functions in generalized MLP architectures of neural networks', *International Journal of Artificial Intelligence and Expert Systems* **1**(4), 111–122.

Karu, M. & Alumäe, T. (2018), Weakly supervised training of speaker identification models, *in* 'Proc. Odyssey 2018 The Speaker and Language Recognition Workshop', pp. 24–30.

Katta, S. V., Umesh, S. et al. (2020), 'S-vectors: Speaker embeddings based on transformer's encoder for text-independent speaker verification', *arXiv preprint arXiv:2008.04659* .

Kenny, P. (2005), 'Joint factor analysis of speaker and session variability: Theory and algorithms', *CRIM, Montreal,(Report) CRIM-06/08-13* **14**, 28–29.

Kenny, P., Mihoubi, M. & Dumouchel, P. (2003), New map estimators for speaker recognition, *in* 'Eighth European Conference on Speech Communication and Technology'.

Khan, A., Sohail, A., Zahoora, U. & Qureshi, A. S. (2020), 'A survey of the recent architectures of deep convolutional neural networks', *Artificial Intelligence Review* **53**(8), 5455–5516.

Khandelwal, S., Lecouteux, B. & Besacier, L. (2016), Comparing GRU and LSTM for automatic speech recognition, PhD thesis, LIG.

Kingma, D. P. & Ba, J. (2015), Adam: A method for stochastic optimization, *in* 'The International Conference on Learning Representations (ICLR)'.

Kinnunen, T. & Li, H. (2010), 'An overview of text-independent speaker recognition: From features to supervectors', *Speech communication* **52**(1), 12–40.

Kolbœk, M., Tan, Z.-H. & Jensen, J. (2016), Speech enhancement using long short-term memory based recurrent neural networks for noise robust speaker verification, *in* '2016 IEEE spoken language technology workshop (SLT)', IEEE, pp. 305–311.

Krizhevsky, A., Sutskever, I. & Hinton, G. E. (2012), 'Imagenet classification with deep convolutional neural networks', *Advances in neural information processing systems (NeurIPS)* **25**, 1097–1105.

Kuhn, R., Nguyen, P., Junqua, J.-C., Goldwasser, L., Niedzielski, N., Fincke, S., Field, K. & Contolini, M. (1998), Eigenvoices for speaker adaptation, *in* 'Fifth International Conference on Spoken Language Processing (ICSLP)'.

Lane, H. & Tranel, B. (1971), 'The lombard sign and the role of hearing in speech', *Journal of Speech and Hearing Research* **14**(4), 677–709.

Le Prell, C. G. & Clavier, O. H. (2017), 'Effects of noise on speech recognition: Challenges for communication by service members', *Hearing research* **349**, 76–89.

LeCun, Y., Bengio, Y. & Hinton, G. (2015), 'Deep learning', *nature* **521**(7553), 436–444.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. & Jackel, L. D. (1989), 'Backpropagation applied to handwritten zip code recognition', *Neural computation* **1**(4), 541–551.

Lei, Y., Scheffer, N., Ferrer, L. & McLaren, M. (2014), A novel scheme for speaker recognition using a phonetically-aware deep neural network, *in* '2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)', IEEE, pp. 1695–1699.

Li, M., Zhang, T., Chen, Y. & Smola, A. J. (2014), Efficient mini-batch training for stochastic optimization, *in* 'Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining', pp. 661–670.

Li, Y., Gao, F., Ou, Z. & Sun, J. (2018), Angular softmax loss for end-to-end speaker verification, *in* '2018 11th International Symposium on Chinese Spoken Language Processing (ISCSLP)', IEEE, pp. 190–194.

Lincoln, M., McCowan, I., Vepa, J. & Maganti, H. K. (2005), The multi-channel Wall Street Journal audio visual corpus (MC-WSJ-AV): Specification and initial experiments, *in* 'IEEE Workshop on Automatic Speech Recognition and Understanding (SRU).', IEEE, pp. 357–362.

Lipton, Z. C., Berkowitz, J. & Elkan, C. (2015), 'A critical review of recurrent neural networks for sequence learning', *arXiv preprint arXiv:1506.00019* .

Lu, X., Tsao, Y., Matsuda, S. & Hori, C. (2013), Speech enhancement based on deep denoising autoencoder., *in* 'Interspeech', Vol. 2013, pp. 436–440.

Maaten, L. v. d. & Hinton, G. (2008), 'Visualizing data using t-SNE', *The Journal of Machine Learning Research (JMLR)* pp. 2579–2605.

Markel, J., Oshika, B. & Gray, A. (1977), 'Long-term feature averaging for speaker recognition', *IEEE Transactions on Acoustics, Speech, and Signal Processing* **25**(4), 330–337.

Martin, A., Doddington, G., Kamm, T., Ordowski, M. & Przybocki, M. (1997), The DET curve in assessment of detection task performance, Technical report, National Inst of Standards and Technology Gaithersburg MD.

Martin, A. F. & Greenberg, C. S. (2009), NIST 2008 speaker recognition evaluation: Performance across telephone and room microphone channels, *in* 'Tenth Annual Conference of the International Speech Communication Association'.

McCulloch, W. S. & Pitts, W. (1943), 'A logical calculus of the ideas immanent in nervous activity', *The bulletin of mathematical biophysics* **5**(4), 115–133.

Medsker, L. R. & Jain, L. (2001), 'Recurrent neural networks', *Design and Applications* **5**.

Miao, X., McLoughlin, I. & Yan, Y. (2019), 'A new time-frequency attention mechanism for TDNN and CNN-LSTM-TDNN, with application to language identification', *Proc. Interspeech 2019* pp. 4080–4084.

Michalek, J. & Vaněk, J. (2018), A survey of recent dnn architectures on the TIMIT phone recognition task, *in* 'International Conference on Text, Speech, and Dialogue', Springer, pp. 436–444.

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

Mikolov, T., Karafiát, M., Burget, L., Černockỳ, J. & Khudanpur, S. (2010), Recurrent neural network based language model, *in* 'Eleventh annual conference of the international speech communication association (Interspeech)'.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. & Dean, J. (2013), 'Distributed representations of words and phrases and their compositionality', *arXiv preprint arXiv:1310.4546* .

Ming, J., Hazen, T. J., Glass, J. R. & Reynolds, D. A. (2007), 'Robust speaker recognition in noisy conditions', *IEEE Transactions on Audio, Speech, and Language Processing* **15**(5), 1711–1723.

Nagrani, A., Chung, J. S. & Zisserman, A. (2017), 'VoxCeleb: A large-scale speaker identification dataset', *Proc. Interspeech 2017* pp. 2616–2620.

Nair, V. & Hinton, G. E. (2010), Rectified linear units improve restricted boltzmann machines, *in* 'International Conference on Machine Learning (ICML)'.

Oglesby, J. (1995), 'What's in a number? moving beyond the equal error rate', *Speech communication* **17**(1-2), 193–208.

Okabe, K., Koshinaka, T. & Shinoda, K. (2018), 'Attentive statistics pooling for deep speaker embedding', *Interspeech* pp. 2252–2256.

Oppenheim, A. V., Buck, J. R. & Schafer, R. W. (2001), *Discrete-time signal processing. Vol. 2*, Upper Saddle River, NJ: Prentice Hall.

Ortega-García, J. & González-Rodríguez, J. (1996), Overview of speech enhancement techniques for automatic speaker recognition, *in* 'Proceeding of Fourth International Conference on Spoken Language Processing. ICSLP'96', Vol. 2, IEEE, pp. 929–932.

Pandey, A. & Wang, D. (2019), 'A new framework for CNN-based speech enhancement in the time domain', *IEEE/ACM Transactions on Audio, Speech, and Language Processing* **27**(7), 1179–1188.

Pandey, A. & Wang, D. (2020), Densely connected neural network with dilated convolutions for real-time speech enhancement in the time domain, *in* 'ICASSP 2020-

2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)', IEEE, pp. 6629–6633.

Pang, J. (2017), Spectrum energy based voice activity detection, *in* '2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC)', IEEE, pp. 1–5.

Peacocke, R. D. & Graf, D. H. (1995), An introduction to speech and speaker recognition, *in* 'Readings in Human–Computer Interaction', Elsevier, pp. 546–553.

Peddinti, V., Povey, D. & Khudanpur, S. (2015), A time delay neural network architecture for efficient modeling of long temporal contexts, *in* 'Sixteenth annual conference of the international speech communication association (Interspeech)'.

Plchot, O., Burget, L., Aronowitz, H. & Matejka, P. (2016), Audio enhancing with dnn autoencoder for speaker recognition, *in* '2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)', IEEE, pp. 5090–5094.

Poddar, A., Sahidullah, M. & Saha, G. (2017), 'Speaker verification with short utterances: a review of challenges, trends and opportunities', *IET Biometrics* **7**(2), 91–101.

Prabowo, Y. D., Warnars, H. L. H. S., Budiharto, W., Kistijantoro, A. I., Heryadi, Y. et al. (2018), LSTM and simple RNN comparison in the problem of sequence to sequence on conversation data using bahasa indonesia, *in* '2018 Indonesian Association for Pattern Recognition International Conference (INAPR)', IEEE, pp. 51–56.

rahman Chowdhury, F. R., Wang, Q., Moreno, I. L. & Wan, L. (2018), Attention-based models for text-dependent speaker verification, *in* '2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)', IEEE, pp. 5359–5363.

Rao, W., Xu, C., Chng, E. S. & Li, H. (2019), 'Target speaker extraction for multi-talker speaker verification', *Proc. Interspeech 2019* pp. 1273–1277.

Reynolds, D. A. (2002), An overview of automatic speaker recognition technology, *in* '2002 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)', Vol. 4, IEEE, pp. IV–4072.

Reynolds, D. A., Quatieri, T. F. & Dunn, R. B. (2000), 'Speaker verification using adapted Gaussian Mixture Models', *Digital signal processing* **10**(1-3), 19–41.

Reynolds, D. A. & Rose, R. C. (1995), 'Robust text-independent speaker identification using Gaussian Mixture speaker models', *IEEE transactions on speech and audio processing* **3**(1), 72–83.

Rix, A. W., Beerends, J. G., Hollier, M. P. & Hekstra, A. P. (2001), Perceptual evaluation of speech quality (PESQ) -a new method for speech quality assessment of telephone networks and codecs, *in* '2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (ICASSP)', Vol. 2, IEEE, pp. 749–752.

Rosenfield, G. H. & Fitzpatrick-Lins, K. (1986), 'A coefficient of agreement as a measure of thematic classification accuracy.', *Photogrammetric engineering and remote sensing* **52**(2), 223–227.

Ruder, S. (2016), 'An overview of gradient descent optimization algorithms', *arXiv preprint arXiv:1609.04747* .

Sadjadi, S. O. & Hansen, J. H. (2010), Assessment of single-channel speech enhancement techniques for speaker identification under mismatched conditions, *in* 'Eleventh Annual Conference of the International Speech Communication Association (Interspeech)'.

Schaefer, A. M., Udluft, S. & Zimmermann, H.-G. (2008), 'Learning long-term dependencies with recurrent neural networks', *Neurocomputing* **71**(13-15), 2481–2488.

Schuster, M. & Paliwal, K. K. (1997), 'Bidirectional recurrent neural networks', *IEEE transactions on Signal Processing* **45**(11), 2673–2681.

Shi, Y. & Hain, T. (2021), Contextual joint factor acoustic embeddings, *in* '2021 IEEE Spoken Language Technology Workshop (SLT)', IEEE, pp. 750–757.

Shon, S., Tang, H. & Glass, J. (2019), 'VoiceID Loss: Speech enhancement for speaker verification', *Proc. Interspeech 2019* pp. 2888–2892.

Singh, S. P., Kumar, A., Darbari, H., Singh, L., Rastogi, A. & Jain, S. (2017), Machine translation using deep learning: An overview, *in* '2017 international conference on computer, communications and electronics (ICCC)', IEEE, pp. 162–167.

Snyder, D., Chen, G. & Povey, D. (2015), 'MUSAN: A music, speech, and noise corpus', *arXiv preprint arXiv:1510.08484* .

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

Soltau, H., Liao, H. & Sak, H. (2016), 'Neural speech recognizer: Acoustic-to-word LSTM model for large vocabulary speech recognition', *arXiv preprint arXiv:1610.09975* .

Sreenu, G., Girija, P., Prasad, M. N. & Nagamani, M. (2004), A human machine speaker dependent speech interactive system, *in* 'Proceedings of the IEEE INDICON 2004. First India Annual Conference, 2004.', IEEE, pp. 349–351.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. (2014), 'Dropout: a simple way to prevent neural networks from overfitting', *The journal of machine learning research* **15**(1), 1929–1958.

Sztahó, D., Szaszák, G. & Beke, A. (2019), 'Deep learning methods in speaker recognition: a review', *arXiv preprint arXiv:1911.06615* .

Taal, C. H., Hendriks, R. C., Heusdens, R. & Jensen, J. (2010), A short-time objective intelligibility measure for time-frequency weighted noisy speech, *in* '2010 IEEE international conference on acoustics, speech and signal processing (ICASSP)', IEEE, pp. 4214–4217.

Takase, S. & Okazaki, N. (2019), Positional encoding to control output sequence length, *in* 'Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (NAACL-HLT)', pp. 3999–4004.

Tan, K., Chen, J. & Wang, D. (2018), 'Gated residual networks with dilated convolutions for monaural speech enhancement', *IEEE/ACM transactions on audio, speech, and language processing* **27**(1), 189–198.

Variani, E., Lei, X., McDermott, E., Moreno, I. L. & Gonzalez-Dominguez, J. (2014), Deep neural networks for small footprint text-dependent speaker verification, *in* 'IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)', IEEE.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. & Polosukhin, I. (2017), Attention is all you need, *in* 'Advances in neural information processing systems (NeurIPS)', pp. 5998–6008.

Waibel, A., Hanazawa, T., Hinton, G., Shikano, K. & Lang, K. J. (1989), 'Phoneme recognition using time-delay neural networks', *IEEE transactions on acoustics, speech, and signal processing* **37**(3), 328–339.

Wan, L., Zeiler, M., Zhang, S., Le Cun, Y. & Fergus, R. (2013), Regularization of neural networks using dropconnect, *in* 'International conference on machine learning (ICML)', pp. 1058–1066.

Wang, D. & Chen, J. (2018), 'Supervised speech separation based on deep learning:

An overview', *IEEE/ACM Transactions on Audio, Speech, and Language Processing* **26**(10), 1702–1726.

Wang, F., Cheng, J., Liu, W. & Liu, H. (2018), 'Additive margin softmax for face verification', *IEEE Signal Processing Letters* **25**(7), 926–930.

Wang, Q., Muckenhirn, H., Wilson, K., Sridhar, P., Wu, Z., Hershey, J., Saurous, R. A., Weiss, R. J., Jia, Y. & Moreno, I. L. (2019), 'Voicefilter: Targeted voice separation by speaker-conditioned spectrogram masking', *Proc. Interspeech 2019* pp. 2728–2732.

Wang, Q., Okabe, K., Lee, K. A., Yamamoto, H. & Koshinaka, T. (2018), Attention mechanism in speaker recognition: What does it learn in deep speaker embedding?, *in* '2018 IEEE Spoken Language Technology Workshop (SLT)', IEEE, pp. 1052–1059.

Wang, S., Yang, Y., Wu, Z., Qian, Y. & Yu, K. (2020), 'Data augmentation using deep generative models for embedding based speaker recognition', *IEEE/ACM Transactions on Audio, Speech, and Language Processing* **28**, 2598–2609.

Wang, Z.-Q. & Wang, D. (2016), 'A joint training framework for robust automatic speech recognition', *IEEE/ACM Transactions on Audio, Speech, and Language Processing* **24**(4), 796–806.

Warden, P. (2018), 'Speech commands: A dataset for limited-vocabulary speech recognition', *arXiv preprint arXiv:1804.03209* .

Willmott, C. J. & Matsuura, K. (2005), 'Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance', *Climate research* **30**(1), 79–82.

Woo, S., Park, J., Lee, J.-Y. & Kweon (2018), Cbam: Convolutional block attention module, *in* 'Proceedings of the European conference on computer vision (ECCV)', pp. 3–19.

Wu, M. & Chen, L. (2015), Image recognition based on deep learning, *in* '2015 Chinese Automation Congress (CAC)', IEEE, pp. 542–546.

Wu, Y., Mao, H. & Yi, Z. (2018), 'Audio classification using attention-augmented convolutional neural network', *Knowledge-Based Systems* **161**, 90–100.

Xie, W., Nagrani, A., Chung, J. S. & Zisserman, A. (2019), Utterance-level aggregation for speaker recognition in the wild, *in* 'ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)', IEEE, pp. 5791–5795.

Xu, Y., Huang, Q., Wang, W., Foster, P., Sigtia, S., Jackson, P. J. & Plumbley, M. D. (2017), 'Unsupervised feature learning based on deep models for environmental audio tagging', *IEEE/ACM Transactions on Audio, Speech, and Language Processing* **25**(6), 1230–1241.

Yadav, S. & Rai, A. (2020), Frequency and temporal convolutional attention for text-independent speaker recognition, *in* 'ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)', IEEE, pp. 6794–6798.

Yang, Z., Yang, D., Dyer, C., He, X., Smola, A. & Hovy, E. (2016), Hierarchical attention networks for document classification, *in* 'Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies (NAACL)', pp. 1480–1489.

Yu, F. & Koltun, V. (2015), 'Multi-scale context aggregation by dilated convolutions', *arXiv e-prints* pp. arXiv–1511.

Yu, Y.-Q., Fan, L. & Li, W.-J. (2019), Ensemble additive margin softmax for speaker verification, *in* 'ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)', IEEE, pp. 6046–6050.

Yuan, C.-M., Sun, X.-M. & Zhao, H. (2020), 'Speech separation using convolutional neural network and attention mechanism', *Discrete Dynamics in Nature and Society* .

Yuan, J. & Zhang, B. (2010), Weakly supervised learning: What could it do and what could not?, *in* 'Proceedings of the Annual Meeting of the Cognitive Science Society', Vol. 32.

Zacarias-Morales, N., Pancardo, P., Hernández-Nolasco, J. A. & Garcia-Constantino, M. (2021), 'Attention-inspired artificial neural networks for speech processing: A systematic review', *Symmetry* **13**(2), 214.

Zeinali, H., Wang, S., Silnova, A., Matějka, P. & Plchot, O. (2019), 'BUT system description to voxceleb speaker recognition challenge 2019', *arXiv preprint arXiv:1910.12592* .

Zhang, C. (2017), Joint Training Methods for Tandem and Hybrid Speech Recognition Systems using Deep Neural Networks, PhD thesis, University of Cambridge.

Zhang, L., Wang, M., Zhang, Q. & Liu, M. (2020), 'Environmental attention-guided branchy neural network for speech enhancement', *Applied Sciences* **10**(3), 1167.

Zhao, F., Li, H. & Zhang, X. (2019), A robust text-independent speaker verification method based on speech separation and deep speaker, *in* 'ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)', IEEE, pp. 6101–6105.

Zhao, X., Wang, Y. & Wang, D. (2014), 'Robust speaker identification in noisy and reverberant conditions', *IEEE/ACM Transactions on Audio, Speech, and Language Processing* **22**(4), 836–845.

Zhou, Z.-H. (2018), 'A brief introduction to weakly supervised learning', *National Science Review* **5**(1), 44–53.

Zhu, Y., Ko, T., Snyder, D., Mak, B. & Povey, D. (2018), Self-attentive speaker embeddings for text-independent speaker verification., *in* 'Interspeech', pp. 3573–3577.

Appendices

Appendix A

Implementation Details

This appendix describes the implementation details of the experiments mentioned in the previous chapters. All of the models are implemented using Tensorflow version 1.5 Abadi et al. (2016).

A.1 Hierarchical Attention Network

A.1.1 Experimental Setup For Section 3.4.1

This section describes the detailed experimental setup for the speaker identification and verification experiments in Section 3.4.1.

The CHE and SWBC datasets contain out-of-domain data compared with the SRE08 dataset, which contains fewer speakers but in a greater variety of acoustic conditions. For the extracted speaker embeddings from the CHE and SWBC datasets, both speaker identification and verification tasks are conducted. For the speaker identification task, datasets are randomly split into a training set and a test set at a 9:1 ratio, with both sets having the same number of speakers. The training set here is to train the classifier of the embeddings, rather than training the model, using SRE08 dataset. Prediction accuracy is used as the evaluation metric. For the speaker verification task, in SWBC, there are 50 speakers in the enrolment set and 120 speakers in the evaluation set, with 10 utterances for each speaker. In the CHE, there are 30 speakers in the enrolment set and 60 speakers in the evaluation set. Each speaker has 10 utterances. For this task, the evaluation metric is the equal error rate (EER).

Energy based VAD (Pang 2017) is used to remove the unvoiced signals. VAD is applied directly to the signals for the interviews and, for the telephone speech signals, the two channels are firstly separated into two individual signals, and then VAD is applied to each channel. Then, fixed length sliding windows (either one second or three seconds) with a half-size shift are employed to divide speech streams into short segments, each of which is viewed as an utterance. The total number of utterances of the three datasets are listed in Table 3.1. Each segment is further segmented into frames using a 25ms sliding window with a 10ms shift. All frames are converted into 20-dimensional MFCC feature vectors. In creating a hierarchical structure, each utterance is viewed as a document; each fragment as a sentence; and each frame vector as a word (Yang et al. 2016).

A.1.2 Implementation and Training Details

This section describes the implementation and training details of the hierarchical attention network in Chapter 3. In order to improve the reproducibility of the experiments, Table A.1 shows the detailed configuration of the proposed hierarchical attention network. Between each layer, there are batch normalisation layers (Ioffe & Szegedy 2015) and dropout layers (Srivastava et al. 2014), where the dropout rate is set to 0.2. An Adam optimiser (Kingma & Ba 2015) is used for all experiments with $\beta_1 = 0.95$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. The initial learning rate is 10^{-4}.

As discussed in Section 2.2.2, variations of the softmax functions, such as the AM-softmax (additive margin softmax) (Wang, Cheng, Liu & Liu 2018), can reduce the intra-class distance of the embeddings. In order to obtain better results, all of the models in this study, including the baselines, use the AM-softmax loss function for training, where m is set to 0.35, and s is set to 40. Cosine similarity is used to measure

Table A.1: *The implementation details of the proposed hierarchical attention network, where L_{win} denotes the segment length, M denotes the number of segments in one utterance.*

Level	Model	Input	Output
Frame-Level	CNN	(L_{win},20,1)	(L_{win},1,512)
	Bi-GRU	(L_{win},512)	(L_{win},1024)
	Attention	(L_{win},1024)	(L_{win},1024)
	Statistics Pooling	(L_{win},1024)	(1,2048)
Segment-Level	CNN	(M,2048,1)	(M,1,1500)
	Attention	(M,1500)	(M,1500)
	Statistics Pooling	(M,1500)	(1,3000)
Utterance-Level	DNN (512)	(1,3000)	(1,512)
	DNN (512)	(1,512)	(1,512)

the distance of the two embeddings in the speaker verification task.

A.2 Speech Enhancement For Speaker Recognition

A.2.1 Implementation of the Multi-Stage Attention Mechanism

This section describes the computation process of the multi-stage attention mechanism that proposed in Section 4.3.1. The working flow of channel attention is shown in Equation A.1, where $\boldsymbol{W_0} \in \mathcal{R}^{C_k \times 100}$, $\boldsymbol{b_0} \in \mathcal{R}^{1\times 100}$ and $\boldsymbol{W_1} \in \mathcal{R}^{100\times C_k}$ are the parameters of the kth channel attention block.

$$\begin{aligned}
\boldsymbol{H}^C_{k,max} &= \max{}^{T_k\times F_k\times 1}(\boldsymbol{H}_k) \\
\boldsymbol{H}^C_{k,avg} &= \text{avg}^{T_k\times F_k\times 1}(\boldsymbol{H}_k) \\
\boldsymbol{s}_{max} &= \text{ReLU}(\boldsymbol{H}^C_{k,max}\boldsymbol{W}_0 + \boldsymbol{b}_0)\boldsymbol{W}_1 \\
\boldsymbol{s}_{avg} &= \text{ReLU}(\boldsymbol{H}^C_{k,avg}\boldsymbol{W}_0 + \boldsymbol{b}_0)\boldsymbol{W}_1 \\
\boldsymbol{\alpha}_{C,k} &= \text{Sigmoid}(\boldsymbol{s}_{avg} + \boldsymbol{s}_{max})
\end{aligned} \tag{A.1}$$

In the implementation of channel attention, max pooling and average pooling (with the kernel size of $T_k \times F_k \times 1$) are firstly applied on both the time and frequency dimension of $\boldsymbol{H_k}$. Their output $\boldsymbol{H}^{C}_{k,avg} \in \mathcal{R}^{1\times1\times C_k}$ and $\boldsymbol{H}^{C}_{k,max} \in \mathcal{R}^{1\times1\times C_k}$ are then used as the input of two fully connected layers sharing the same parameters and followed by a ReLU activation. The channel attention weight vector $\boldsymbol{\alpha}_{C,k} \in \mathcal{R}^{1\times1\times C_k}$ is finally obtained after a sigmoid activation which is applied to the summation of $\boldsymbol{s}_{avg}$ and $\boldsymbol{s}_{max}$. After repeating $\boldsymbol{\alpha}_{C,k}$ to the same dimension as $\boldsymbol{H}_k$, the attention map is multiplied by the original feature map $\boldsymbol{H}_k$ to generate the refined feature map $\boldsymbol{H}'_k$.

One important operation is the attention weight that is computed here by the sigmoid function, rather than a softmax function. As discussed in Sections 2.1.1 and 2.2.2, both the softmax and sigmoid functions project the input to the numbers that are between zero and one. The difference is that the output numbers of the softmax function can be summed to one, a constraint that the sigmoid function does not have. As discussed in Section 3.1 and shown in Section 3.5.3, the softmax based attention weight computing method may dilute some features when there is more than one important feature. In order to avoid this problem, the softmax function is replaced by the sigmoid function.

$$
\begin{aligned}
\boldsymbol{H}^{C'}_{k,max} &= \max{}^{1\times1\times C_k}(\boldsymbol{H}'_k) \\
\boldsymbol{H}^{C'}_{k,avg} &= \mathrm{avg}^{1\times1\times C_k}(\boldsymbol{H}'_k) \\
\boldsymbol{H}^{C'}_{k,pool} &= [\boldsymbol{H}^{C'}_{k,avg}; \boldsymbol{H}^{C'}_{k,max}] \\
\boldsymbol{H}^{T'}_{k,max} &= \max{}^{T_k\times1\times1}(\boldsymbol{H}^{C'}_{k,pool}) \\
\boldsymbol{H}^{T'}_{k,avg} &= \mathrm{avg}^{T_k\times1\times1}(\boldsymbol{H}^{C'}_{k,pool}) \\
\boldsymbol{H}'_{k,pool} &= [\boldsymbol{H}^{T'}_{k,avg}; \boldsymbol{H}^{T'}_{k,max}] \\
\boldsymbol{\alpha}_{F,k} &= \mathrm{Sigmoid}(\mathrm{Conv}^{2\times7\times2}(\boldsymbol{H}'_{k,pool}))
\end{aligned}
\tag{A.2}
$$

The frequency and time attention works in a similar structure. Equation A.2 shows the implementation of the frequency attention. In the kth CONV-MS or RES-MS block, a max pooling and an average pooling are firstly applied to the channel dimension of the input data $\boldsymbol{H}'_k$, and the corresponding outputs are $\boldsymbol{H}^{C}_{k,max} \in \mathcal{R}^{T_k \times F_k \times 1}$ and $\boldsymbol{H}^{C}_{k,avg} \in \mathcal{R}^{T_k \times F_k \times 1}$, respectively. $\boldsymbol{H}^{C}_{k,pool} \in \mathcal{R}^{T_k \times F_k \times 2}$ is obtained by concatenating the outputs after using poolings. On the time dimension, the same max pooling and average pooling steps are applied on $\boldsymbol{H}^{C}_{k,pool} \in \mathcal{R}^{T_k \times F_k \times 2}$ and the corresponding outputs are $\boldsymbol{H}^{T}_{k,avg} \in \mathcal{R}^{1 \times F_k \times 2}$ and $\boldsymbol{H}^{T}_{k,max} \in \mathcal{R}^{1 \times F_k \times 2}$. Again, the output after concatenating them on the time dimension is $\boldsymbol{H}_{k,pool} \in \mathcal{R}^{2 \times F_k \times 2}$. The frequency attention weight vector $\boldsymbol{\alpha}_{F,k}$ is computed using a convolution operation with a $2 \times 7 \times 2$ kernel followed by a sigmoid activation. The stride value is 1 on the frequency dimension during convolution. The size of $\boldsymbol{\alpha}_{F,k}$ is then expanded to the same as $\boldsymbol{H}''_k$ by data broadcast. The frequency refined feature map $\boldsymbol{H}''_k$ is finally obtained by the product of $\boldsymbol{\alpha}_{F,k}$ and $\boldsymbol{H}'_k$.

$$
\begin{aligned}
\boldsymbol{H}^{C''}_{k,max} &= \max{}^{1\times 1\times C_k}(\boldsymbol{H}''_k) \\
\boldsymbol{H}^{C''}_{k,avg} &= \text{avg}^{1\times 1\times C_k}(\boldsymbol{H}''_k) \\
\boldsymbol{H}^{C''}_{k,pool} &= [\boldsymbol{H}^{C''}_{k,avg}; \boldsymbol{H}^{C''}_{k,max}] \\
\boldsymbol{H}^{F''}_{k,max} &= \max{}^{1\times F_k\times 1}(\boldsymbol{H}^{C''}_{k,pool}) \\
\boldsymbol{H}^{F''}_{k,avg} &= \text{avg}^{1\times F_k\times 1}(\boldsymbol{H}^{C''}_{k,pool}) \\
\boldsymbol{H}''_{k,pool} &= [\boldsymbol{H}^{F''}_{k,avg}; \boldsymbol{H}^{F''}_{k,max}] \\
\boldsymbol{\alpha}_{T,k} &= \text{Sigmoid}(\text{Conv}^{7\times 2\times 2}(\boldsymbol{H}''_{k,pool}))
\end{aligned}
\tag{A.3}
$$

The computation of the time attention is similar to that for the frequency attention. Equation A.2 shows the computational flow. The final feature representation is obtained by the multiplication of the previous frequency refined feature map and the time attention weights $\boldsymbol{\alpha}_{T,k}$.

A.2.2 Model Architectures and Training Details

Table A.2: *Architecture of the speech enhancement network (SE-Net) that consists of 11 blocks. In each block, a dilated convolutional layer is followed by a multi-stage attention (MS) layer.*

Layer Name	Structure	Dilation
CONV-MS Block1	7x1x48 MS	1x1
CONV-MS Block2	1x7x48 MS	1x1
CONV-MS Block3	5x5x48 MS	1x1
CONV-MS Block4	5x5x48 MS	1x2
CONV-MS Block5	5x5x48 MS	1x4
CONV-MS Block6	5x5x48 MS	1x8
CONV-MS Block7	5x5x48 MS	1x1
CONV-MS Block8	5x5x48 MS	2x2
CONV-MS Block9	5x5x48 MS	4x4
CONV-MS Block10	5x5x48 MS	8x8
CONV-MS Block11	1x1x1 MS	1x1

Table A.3: *Architecture of SR-Net consists of 8 blocks. Within each block, the multiple convolutional layers are followed by a multi-stage attention (MS) layer before a residual connection.*

Block Name	Structure	Output
RES-MS Block1	3x3x64 3x3x64 3x3x64 MS	150x129
RES-MS Block2	3x3x128 3x3x128 3x3x128 MS	75x65
RES-MS Block3	3x3x128 3x3x128 MS	75x65
RES-MS Block4	3x3x256 3x3x256 3x3x128 M	38x33
RES-MS Block5	3x3x256 3x3x256 MS	38x33
RES-MS Block6	3x3x256 3x3x256 MS	38x33
RES-MS Block7	3x3x256 3x3x256 MS	38x33
RES-MS Block8	3x3x512 3x3x512 3x3x128 MS	19x17
Pool	19x1	1x17x512
FC	512	

This section describes the detailed model architectures and training details of the joint system proposed in Section 4.3. In order to improve the reproducibility of the proposed model in this work, detailed model architecture of the speech enhancement and speaker recognition models are shown in Tables A.2 and A.3, and their architectures are introduced in Section 4.3.1.

In the speech enhancement model, 11 dilated convolutional layers are employed and each layer is followed by a multi-stage attention module (MS) that is inserted into each residual block. Each of these two models are trained independently, and are then fine-tuned by a joint optimisation. During the training, an Adam optimiser (Kingma & Ba 2015) is used with the initial learning rate being set to $1e-3$ and the decay rate being set to 0.9 for each epoch.

The speaker recognition model shown in Table A.3 uses the ResNet-20 architecture (He et al. 2016), due to its effectiveness in speaker recognition (Hajibabaei & Dai 2018).

A.3 Speaker Dependent Speech Enhancement Model

This section describes the detailed model architectures and training details of the speaker dependent speech enhancement model proposed in Section 4.4. Table A.4 shows the encoder architecture of the skip/residual auto-encoder employed by the speech enhancement model used in Step1 and Step2. Its decoder structure mirrors the encoder.

For the speaker recognition model, as discussed in Section 4.4.1, the ResNet-20 architecture is used and the detailed model architecture can be found in Section A.2.

For optimisation, the Adam optimizer (Kingma & Ba 2015) is used with the initial learning rate being set to $1e-3$ and the decay rate being set to 0.9 for each epoch.

Table A.4: *The encoder architecture of the proposed speaker dependent speech enhancement approach, where T, F, C represents the time, frequency and feature dimensions. The number of features and strides on each dimension are shown as Feature/Strides.*

Operation	Structure	Input (T, F, C)	Output (T, F, C)
Encoder	16/(1,2)	(300,257,1)	(300,129,16)
	32/(2,2)	(300,129,16)	(150,65,32)
	64/(2,2)	(150,65,32)	(75,33,64)
	128/(2,2)	(75,33,64)	(38,17,128)
	256/(2,4)	(38,17,128)	(19,5,256)
Reshape	-	(19,5,256)	(19,1280)
Concatenation	-	(19,1280)	(19,1536)
DNN	512	(19,1536)	(19,512)
Bi-GRU	640	(19,512)	(19,1280)
Reshape	-	(19,1280)	(19,5,256)

A.4 Speaker Embedding De-mixing

A.4.1 Model Architecture

This section describes the model architecture of the speaker embedding de-mixing network proposed in Section 5.3. Table A.5 show the architecture of the speaker recognition model used in this work. The speaker recognition model consists of three elements, which are the frame-level feature extractor, statistics pooling and the segment-level feature extractor. For the frame-level feature extractor, the network consists of the TDNN layers and the residual TDNN blocks. As discussed in Section 2.2 the residual connection can make the model easy to train and more robust and is therefore added to the frame-level extractor feature. The input data is first passed into two TDNN layers. Then, three residual TDNN blocks are used. The last TDNN layer maps the feature dimension into 1500. The use of residual TDNN blocks instead of normal TDNN layers, like X-vectors, can increase the robustness of the learned embeddings (Zeinali et al. 2019). A statistics pooling operation is then used, the output from which is fed into a segment-level feature extractor which contains two fully-connected layers. The speaker embedding is obtained from the last fully-connected layer. The softmax function, com-

Table A.5: *Architecture of the speaker recognition module that is used for learning clean embeddings.*

Layer	Context	Output
TDNN Layer1	$[t-1, t, t+1]$	512
TDNN Layer2	[t]	512
TDNN-Res1	$[t-2, t-1, t, t+1t+2]$	512
	[t]	
TDNN-Res2	$[t-2, t-1, t, t+1, t+2]$	512
	[t]	
TDNN-Res3	$[t-2, t-1, t, t+1, t+2]$	512
	[t]	
TDNN Layer3	[t]	1500
Statistics Pooling	T	3000
Segment-Level	T	512
	T	512

bined with cross-entropy loss (as discussed in Section 2.2.2), is used as the loss function to train the speaker recognition network.

For the architecture of the de-mixing network, it puts the feature extractor and the embedding de-mixing function together. The input mixed data contains Speaker1 and Speaker2. The embedding extractor converts the mixed signal into the embedding space and results in $\boldsymbol{e}_{mix}$. The embedding extractor for the mixed signal is built using the same model architecture as the speaker recognition model (excluding the softmax layer). The embedding extractor for the mixed signal actually takes the pre-trained network of the speaker recognition model, as it already learns the mapping from the signal space to the embedding space. In order to reduce the training complexity, the trained network is reused to extract the mixed embedding $\boldsymbol{e}_{mix}$.

A.4.2 Implementation Details

With regard to implementation, for all of the models used in the speaker embedding de-mixing method, the dimension of all of the fully connected layers is set to 512. Each layer is followed by a batch normalisation layer (Ioffe & Szegedy 2015) except for the embedding layer. ReLU activation (Wan et al. 2013) is used for each layer except for the embedding layer. The Adam optimiser (Kingma & Ba 2015) is used in training, with β_1 set to 0.95, β_2 to 0.999, and ϵ is 10^{-8}. The initial learning rate is 10^{-3}. For both of the experiments conducted using the TIMIT and MC-WSJ datasets, 20 dimensional MFCC features are used.

A.5 Speaker and Content Embedding De-Mixing

This section describes the detailed model architecture of the speaker and content embedding de-mixing network proposed in Section 5.5.

For the architecture of the phone recognition network used for evaluation in Section 5.5.2, it contains a sequence-to-sequence architecture that was introduced in Section 2.3.2. The encoder contains a two-layer Bi-LSTM, each with 256 units for each direction. For the decoder, the beam search decoder is used. CTC loss (Soltau et al. 2016) is used for the continuous phone recognition task. It is pre-trained using the clean TIMIT dataset for the purpose of phone recognition.

For the implementation of the speaker and content de-mixing network, it is shown in Figure A.1. The acoustic feature sequence is input to two TDNN layers. The output sequence is then the input to a two-layer GRU. The output vector from the second bi-directional GRU layer is $\boldsymbol{e}_{mix}$. Then, the same de-mixing function $f_{demix}(.)$ is used as was introduced in Section 5.2. $\boldsymbol{e}_{mix}$ and $\boldsymbol{e}_{s1}$ are input to $f_{demix}(.)$, the output $\boldsymbol{e}_{c1}^{'}$ is the estimated content embedding of the corresponding speaker. Or, inversely, the $\boldsymbol{e}_{mix}$ and $\boldsymbol{e}_{s1}$ are input to $f_{demix}(.)$, and the output $\boldsymbol{e}_{s1}^{'}$ is the estimated speaker embedding of the corresponding content.

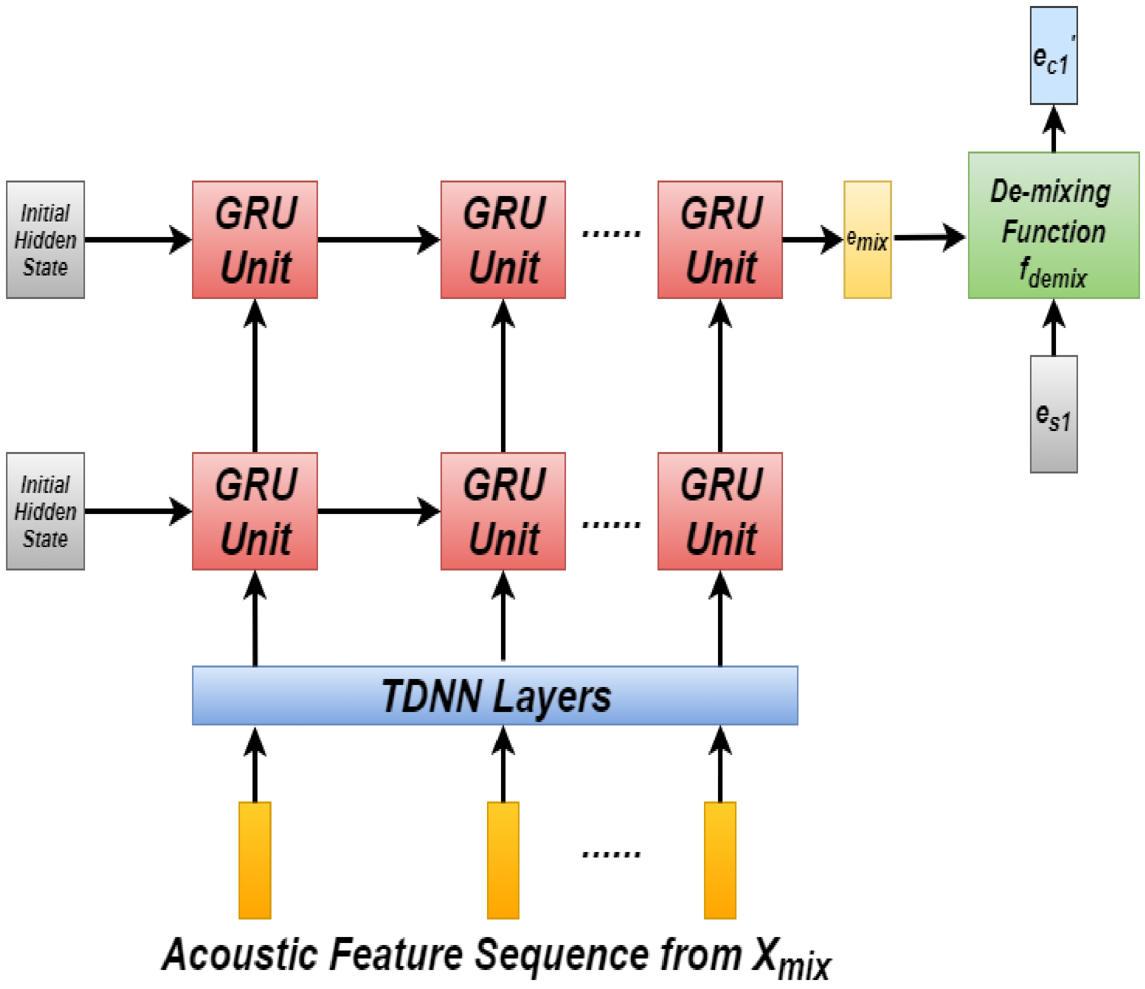


Figure A.1: *The architecture of the speaker and content de-mixing network.*

There are two TDNN layers, the first of which operates on $[t-2, t, t+2]$, the second on the current time step t only. Each of the two layers have 512 dimensional output. Then, a two-layer bi-directional GRU is applied on the output of the TDNN layers. The GRU layers has a 256 dimension for each direction. The embedding is extracted from the second layer of the GRU.

A.6 Weakly Supervised Training of the Hierarchical Attention network

A.6.1 Implementation and Training Details

Table A.6: *Architecture of the proposed hierarchical attention network architecture, where N denotes the total number of speakers, L_{win} denotes the segment length, M denotes the number of segments in one utterance.*

Level	Model	Input	Output
Frame-Level	TDNN	(L_{win},20)	(L_{win},256)
	Bi-GRU	(L_{win},256)	(L_{win},512)
	Attention	(L_{win},512)	(L_{win},512)
	Statistics Pooling	(L_{win},512)	(1,1024)
Segment-Level	TDNN1	(M,1024)	(M,512)
	TDNN2	(M,512)	(M,512)
	TDNN3	(M,512)	(M,1500)
	Attention	(M,1500)	(M,1500)
	Statistics Pooling	(M,1500)	(1,3000)
Utterance-Level	DNN (512)	(1,3000)	(1,512)
	DNN (K)	(1,512)	(1,N)

This section describes the implementation and training details of the weakly supervised training for the hierarchical attention network that proposed in Section 6.3. In order to improve the reproducibility, Table A.6 shows the details of the h-vector model, which is the same as that in Section 3.4 except for the last layer. The TDNN in both frame-level and segment-level encoder operates at the current time step. Batch normalizations (Ioffe & Szegedy 2015) are added after each layer except for attention layer. Adam optimiser (Kingma & Ba 2015) is used for all experiments with $\beta_1 = 0.95$,

$\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. The initial learning rate is 10^{-4}.

A.7 Hierarchical Transformer Network

A.7.1 Implementation Details

Table A.7: *The architecture of the proposed hierarchical transformer network, where N denotes the total number of speakers, T represents the whole sequence length, Transformer represents one transformer encode block, L_{win} denotes the segment length, M denotes the number of segments in one utterance, d denotes the output dimension of each transformer encoder block.*

Level	Model	Input	Output
Global	TDNN	(T,20)	(T,d)
Frame-Level	4xTransformer	(L_{win},d)	(L_{win},d)
	Statistics Pooling	(L_{win},d)	(1,2d)
Segment-Level	TDNN	(M,2d)	(M,d)
	Transformer	(M,d)	(M,d)
	TDNN	(M,D)	(M,1500)
	Statistics Pooling	(M,1500)	(1,3000)
Utterance-Level	DNN (512)	(1,3000)	(1,512)
	DNN (K)	(1,512)	(1,N)

This section describes the implementation and training details of the hierarchical transformer network that proposed in Section 6.4. Table A.7, shows the parameter configuration of the proposed network architecture, The value of d is set to 512, and the dimension of the DNN within the transformer encoder block is set to 2048, and the number of the attention heads h is set to h in all transformer layers. The TDNN layers in both frame-level and segment-level encoders operate at the current time step. The Adam optimiser (Kingma & Ba 2015) is used for all experiments with $\beta_1 = 0.95$,

$\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. The initial learning rate is 10^{-4}.

A.7.2 The Memory Mechanism In the Multi-Head Attention Mechanism

This section describes the computation process of the memory mechanism that used in the hierarchical transformer network proposed in Section 6.4.1. Equation A.4 shows the memory mechanism that is built into the multi-head attention mechanism. The most important part of the transformer block is the multi-head attention mechanism that captures different speaker properties, therefore the memory mechanism shares the information from each segment in the multi-head attention mechanism, allowing it to capture more speaker features.

$$\begin{aligned} &\text{MultiHead}([\boldsymbol{Q};\boldsymbol{C}],[\boldsymbol{K};\boldsymbol{C}],\boldsymbol{V}) = [\boldsymbol{h}_1,\boldsymbol{h}_2,...,\boldsymbol{h}_h]\boldsymbol{W}^O \\ &\boldsymbol{h}_i = \text{Attention-Head}(\boldsymbol{Q}\boldsymbol{W}_i^Q,[\boldsymbol{K};\boldsymbol{C}]\boldsymbol{W}_i^k,[\boldsymbol{V};\boldsymbol{C}]\boldsymbol{W}_i^V) \end{aligned} \tag{A.4}$$

In Equation A.4, the input segment is transformed into Queries ($\boldsymbol{Q}$), Keys ($\boldsymbol{K}$) and Values($\boldsymbol{V}$) using the linear transformation ($\boldsymbol{K},\boldsymbol{Q},\boldsymbol{V} \in \mathcal{R}^{M\times d}$) (Vaswani et al. 2017). Then, memory ($\boldsymbol{C}$, has the same dimensionality as $\boldsymbol{Q}$, $\boldsymbol{K}$ and $\boldsymbol{V}$) from the last block is concatenated with $\boldsymbol{K}$ and $\boldsymbol{V}$ respectively, resulting in $[\boldsymbol{K};\boldsymbol{C}],[\boldsymbol{Q};\boldsymbol{C}] \in \mathcal{R}^{2M\times d}$. For each attention head $\boldsymbol{h}_i$, $\boldsymbol{W}_i^Q,\boldsymbol{W}_i^Q,\boldsymbol{W}_i^Q \in \mathcal{R}^{d_k\times M}$ are the parameter matrices, $d_k = \frac{d}{h}$, h is the number of attention heads. $\boldsymbol{h}_i$ is computed using Equation 6.2. The results for each attention head are concatenated together and $\boldsymbol{W}^O \in \mathcal{R}^{d\times M}$ is used to fuse the output of each attention head into a single output. The values of the initial memories are set to zeros when processing the first segment of the input sequence; the gradients are not computed for the all of the memories during training process to reduce the computational cost (Dai et al. 2019).

Equation A.5 shows the sinusoidal positional encoding technique used in the T-vector model. This was introduced by Vaswani et al. (2017). The "pos" represents the position of the frame in the whole sequence (not the segment) (Takase & Okazaki 2019).

$$\begin{aligned}\mathrm{PE}(\mathrm{pos}, 2i) &= \sin(\mathrm{pos}/10000^{\frac{2i}{d}}) \\ \mathrm{PE}(\mathrm{pos}, 2i+1) &= \cos(\mathrm{pos}/10000^{\frac{2i}{d}})\end{aligned} \tag{A.5}$$

From Equation A.5 one can observe that the positional encoding provides a vector to represent the position of each frame in the whole sequence. The positional encoding vector is then concatenated with the original frames. When processing each segment, the transformer block in each frame-level encoder can access the unique position of each frame. Combined with the memory mechanism, it is assumed that this can make the model better able to capture the sequence information and provide better performance.

Appendix B

Experimental Setup and Data Description

B.1 Experimental Setup for The Noise Robustness Evaluation

This section describes the experimental setup for the evaluation of the noise robustness of the proposed models. The experiments in Sections 3.5, 4.3 and 4.4 all make use of the experimental setup below.

For the speaker identification task, in the VoxCeleb1 dataset, both training and test sets contain 1,251 speakers (Nagrani et al. 2017). The training set contains 145,265 utterances and the test set contains 8,251 utterances. In order to reduce possible bias, the MUSAN dataset is also split into two parts for training and test to ensure that the noise signals used for training will not be reused for test. Each training utterance is mixed with a type of noise at one of five SNR levels. For the test set, the same data configuration is set.

For the speaker verification task, there are 148,642 utterances from 1,211 speakers in the VoxCeleb1 development dataset, and 4,874 utterances from 40 speakers in the test dataset. There are a total 37,720 test pairs. The data configuration is the same for both tasks.

$$\begin{aligned}
\boldsymbol{x}_{mix} &= \boldsymbol{x} + \text{scale}\boldsymbol{n} \\
\text{scale} &= \sqrt{\frac{\text{power}(\boldsymbol{n}')}{\text{power}(\boldsymbol{n})}} \\
\text{power}(\boldsymbol{n}) &= \frac{1}{T}\sum_{t=1}^{T} n_t^2 \\
\text{power}(\boldsymbol{n}') &= \frac{\text{power}(\boldsymbol{x})}{10^{\text{SNR}/10}} \\
\text{power}(\boldsymbol{x}) &= \frac{1}{T}\sum_{t=1}^{T} x_t^2
\end{aligned} \tag{B.1}$$

Equation B.1 shows the computational process of the procedure of mixing signals, where $\boldsymbol{x}$ and $\boldsymbol{n}$ are audio signal and noise from Voxceleb1 and MUSAN dataset respectively, T represents the length of the two signals, and x_t and n_t represents the tth sample point in $\boldsymbol{x}$ and $\boldsymbol{n}$.

B.2 Data Processing for the Embedding De-mixing Approach

This section describes the data processing step (using TIMIT and MC-WSJ) for the embedding de-mixing approach, which are used for the experiments in Sections 5.3 and 5.5.

As discussed in Section 5.3.2, TIMIT contains clean speech data. In the clean embedding extraction step, the clean embeddings are directly learned from the clean signal from the TIMIT dataset. After training the speaker recognition model, for each speaker, 200 segments are randomly sampled and fed into the speaker recognition model, resulting in a corresponding 200 embeddings for each segment. The clean speaker embeddings are the average of the embeddings from each segment-level embedding belonging to the same speaker.

TIMIT consists only of clean speech so, in order to generate a mixed speech signal for the purposes of speaker embedding de-mixing, each utterance in the TIMIT dataset is randomly mixed with another utterance from another speaker, at a certain signal-to-noise ratio (SNR), who thereby becomes the interfering speaker. Speaker1 is the target speaker, and Speaker2 is the interfering speaker. Training data will only be mixed with training data, and test data will only be mixed with test data. This is to avoid bias, as when training the de-mixing network, the model will not get access to any utterances from the test set.

As discussed in Section 5.3.2, the overlapping scenario contains the overlapped speakers and it is used for this setup. The clean speaker embeddings are trained using the signals recorded by headset microphones worn by the speakers. This is because the recorded signal from the headset is closest to the clean signal and there is minimal energy from the interfering speaker. In the embedding de-mixing process, the learned clean embeddings are used to filter out the interference information from the interference signal. The de-mixing network is trained and tested on the signal from two types of microphones (array1 and array2). For each speaker pair, 70 utterances are randomly selected as the test utterances.

B.3 The Speech Command Dataset

This section describes the property of the speech command dataset (Warden 2018) that was used for the experiments in Section 5.4.2.

The speech command dataset contains 105,829 utterances of 35 unique words. The length for each utterance is one second. There are 2618 unique speakers in this dataset. The 35 unique words and the number of occurrences can be found in Table B.1.

Table B.1: *The 35 unique words and the number of occurrences in the speech command dataset, from Warden (2018).*

Word	Number of Utterances
Backward	1,664
Bed	2,014
Bird	2,064
Cat	2,031
Dog	2,128
Down	3,917
Eight	3,787
Five	4,052
Follow	1,579
Forward	1,557
Four	3,728
Go	3,880
Happy	2,054
House	2,113
Learn	1,575
Left	3,801
Marvin	2,100
Nine	3,934
No	3,941
Off	3,745
On	3,845
One	3,890
Right	3,778
Seven	3,998
Sheila	2,022
Six	3,860
Stop	3,872
Three	3,727
Tree	1,759
Two	3,880
Up	3,723
Visual	1,592
Wow	2,123
Yes	4,044
Zero	4,052