



THE HONG KONG
POLYTECHNIC UNIVERSITY
香港理工大學

Department of Electronic and Information Engineering

Final Year Project Final Report **(2022/23)**

Deep Speaker Embedding Across Languages

Student Name:	FANG Jiaying
Student ID:	19078474D
Programme Code:	42470
Supervisor(s):	Prof. Mak Man-Wai
Submission Date:	April 07, 2023

Contents

1	Introduction	4
2	Background and Related Works	4
2.1	Speaker Verification Systems	4
2.1.1	X-vector Architecture	5
2.1.2	Existing Speaker Verification Systems	5
2.2	Domain Adaptation in Speaker Verification Systems	7
2.2.1	Domain Adaptation in Backend	7
2.2.2	DNN Domain Adaptation	8
3	Methodology	13
3.1	System Pipeline	13
3.2	Model Architecture	14
3.3	Additive Angular Margin Softmax	14
3.4	Maximum Mean Discrepancy	15
3.5	Loss Function	16
3.6	VAE-based Method Exploration	16
3.6.1	VAE-based System Pipeline	16
3.6.2	VAE Model Architecture	17
3.6.3	VAE Loss	17
4	Implementation	18
4.1	Dataset	18
4.2	Data Loader	19
4.3	Iterating Over Two Data Loaders	20
4.4	MMD Loss	20
4.5	Loss Function	21
4.6	Multi-GPU Training	22
4.7	VAE-based Method Implementation	22

4.7.1	Reparameterization	22
4.7.2	VAE Loss	23
5	Experiments and Evaluation	23
5.1	Evaluation Metrics	23
5.1.1	Equal Error Rate	23
5.1.2	Minimum Decision Cost Function	24
5.2	Experiments	24
5.2.1	Results	24
5.2.2	Discussion	26
5.2.3	Demonstration	27
6	Conclusion	28
7	References	29
	Summary and Declaration Form	32

1 Introduction

The main objective of this project is to deal with the language mismatch problem, which is a long-existing and challenging problem that hinders the further development and application of speaker verification systems. This project incorporates language-mismatch loss into the deep neural network (DNN) models to achieve language-independent speaker verification. The new model is designed based on current state-of-the-art models like ECAPA-TDNN [1] and trained on large datasets with Mandarin and English speech data. It is then evaluated on popular benchmark datasets for performance evaluation and comparison. The project's outcomes are a new language-invariant speaker embedding model and a new loss function that leverages maximum mean discrepancy.

Besides, another Variational-Auto-encoder-based (VAE-based) method for alleviating the language mismatch problem in speaker verification systems has also been explored in this project. The implementation and experiments of this VAE-based method will also be introduced.

The remainder of this report is structured as follows. Section 2 is about the background, related works, and the difference between existing methods and our method. Section 3 presents the methodology proposed in this project. Section 4 and Section 5 present the implementation details and experiment results. Section 6 is the conclusion.

2 Background and Related Works

This project mainly involves two research fields: speaker verification and domain adaptation. It will focus on language mismatch, one of the domain mismatches in speaker verification systems.

2.1 Speaker Verification Systems

In most speaker verification systems, time delay neural networks (TDNNs) are trained on speaker identification tasks to enable the network to output speaker embeddings. Speaker verification can be accomplished by comparing the two embeddings from enrollment and test recordings. Recently, x-vector [2] based systems have achieved good performance on speaker verification tasks. The following will introduce the original x-vector architecture and current state-of-the-art methods.

2.1.1 X-vector Architecture

X-vector [2] is a speaker embedding extracted from a layer of an embedding DNN. The network is first trained to classify different speakers. After convergence, two x-vectors are extracted from the network with enrollment input and test recording input, respectively, and then serve as the embeddings needed for the speaker verification procedure mentioned above. A detailed structure of the original x-vector system can be seen in Fig. 1.

Layer	Layer context	Total context	Input x output
frame1	$[t - 2, t + 2]$	5	120x512
frame2	$\{t - 2, t, t + 2\}$	9	1536x512
frame3	$\{t - 3, t, t + 3\}$	15	1536x512
frame4	$\{t\}$	15	512x512
frame5	$\{t\}$	15	512x1500
stats pooling	$[0, T)$	T	1500T x 3000
segment6	$\{0\}$	T	3000x512
segment7	$\{0\}$	T	512x512
softmax	$\{0\}$	T	512xN

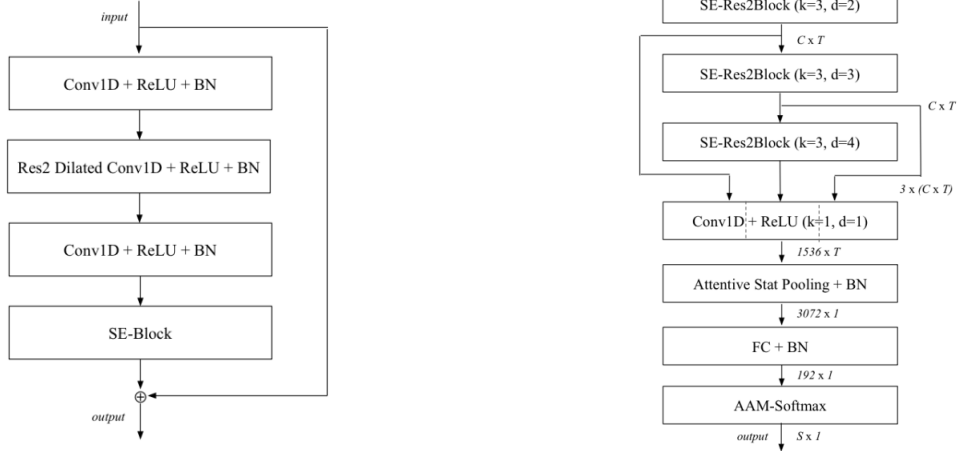
Figure 1: The detailed architecture of the embedding DNN. X-vectors are extracted from *segment 6*, the layer after *stats pooling* [2].

The statistics pooling layer shown in Fig. 1 is worthy of attention. This layer allows x-vectors to be of the same size regardless of the lengths of input utterances. This is done by aggregating all outputs from *frame 5* in frame level and computing the mean as well as standard deviation of the aggregation result. With statistics pooling, comparing embeddings would be easy and unbiased.

2.1.2 Existing Speaker Verification Systems

ECAPA-TDNN [1] has achieved good results on VoxCeleb Speaker Recognition Challenge [3]. It utilizes the x-vector architecture while adding three main enhancements: statistics pooling with channel and context attention, Squeeze-Excitation Res2Blocks (SE block), and Multi-layer feature aggregation.

In ECAPA-TDNN [1], the attention mechanism is used in statistics pooling with channel dimension attention to enable the network to pay more attention to certain characteristics that



(a) The architecture of SE-Res2Block [1].

(b) The network architecture of ECAPA-TDNN [1].

Figure 2: ECAPA-TDNN architecture.

may not activate on similar time instances. The equation of this attention mechanism is shown in Equation 1:

$$e_{t,c} = \mathbf{v}_c^T f(\mathbf{W}\mathbf{h}_t + \mathbf{b}) + k_c, \quad (1)$$

where \mathbf{h}_t contains the activations in the last convolutional layer of frame level of t , \mathbf{W} and \mathbf{b} project \mathbf{h}_t to a lower dimension for efficiency and avoidance of overfitting, $f(\cdot)$ is a non-linear function, \mathbf{v}_c^T and k_c are weights and bias. Then the channel-dependent attention score $e_{t,c}$ will be normalized using the softmax function to get a normalized attention score, indicating the importance of different frames with a given channel.

Another contribution of ECAPA-TDNN [1] is its use of Squeeze-and-Excitation-block (SE-block) [4] in one-dimension. The successful applications of SE-block in 2D have proved its effectiveness. In ECAPA-TDNN, the network consists of several 1D-modified SE blocks called SE-Res2Block, which emphasizes the important channel features and weakens the unimportant channel features. The multi-layer aggregation and summation in ECAPA-TDNN [1] is also effective, which utilizes the feature maps of not only the last frame layer but also more shallow layers. The architecture of SE-Res2Block and the overall network architecture are shown in Fig. 2a and Fig. 2b.

Some other speaker verification methods which achieve good results combine ResNet and

TDNN to achieve better results [5]. However, they all have a relatively similar structure to the ECAPA-TDNN [1].

2.2 Domain Adaptation in Speaker Verification Systems

Data is biased if certain characteristics are shown more frequently than they would be if uniformly randomly selected. In real-world applications, generalizing the biased data to a target subpopulation is more meaningful than generalizing it to the whole population [6]. This process is called domain adaptation. For example, the neural network in a speaker verification system is trained on English-speaking speech data, but it needs to be used on Chinese-speaking speech data. Directly using the system will result in bad performance since there will be a language mismatch. In order to alleviate this problem, a system trained with data in the target domain is required. However, labeled data in the target domain is hard to get. Therefore, methods only utilize unlabeled data will be introduced.

2.2.1 Domain Adaptation in Backend

Most of the current state-of-the-art speaker verification systems perform domain adaptation in the backend [2]. The two commonly used backends in speaker verification systems are Probabilistic Linear Discriminant Analysis [7] and Correlation Alignment [8].

2.2.1.1 Probabilistic Linear Discriminant Analysis

Probabilistic linear discriminant analysis (PLDA) [7] is a probabilistic version of linear discriminant analysis (LDA). Similar to LDA, PLDA projects embeddings like x-vectors [2] and i-vectors [9] to a new vector space. It extends the distance between classes and reduces the distance within classes. Then, it determines whether the two embeddings are from the same class. In PLDA, an embedding \mathbf{x}_{ij} , which is the j^{th} embedding in class i , can be represented as,

$$\mathbf{x}_{ij} = \boldsymbol{\mu} + \mathbf{V}\mathbf{y}_i + \boldsymbol{\epsilon}_{ij}, \quad (2)$$

where $\boldsymbol{\mu}$ is the global mean, \mathbf{V} is the inter-class variability matrix, \mathbf{y}_i is the speaker coordinate in space defined by \mathbf{V} and $\boldsymbol{\epsilon}_{ij}$ is sampled from a gaussian distribution with the intra-class covariance.

PLDA uses the log-likelihood ratio to perform the evaluation. It provides a log-likelihood ratio between two hypotheses: one is the two embeddings are from the same class, and the other one is

the two embeddings are from different classes.

2.2.1.2 Correlation Alignment

An effective and popular domain adaptation method is Correlation Alignment (CORAL) [8], which was introduced in speaker verification in [10]. The main idea of CORAL is whitening the Out-of-Domain (OOD) vector and then re-coloring it. Given \mathbf{C}_o and \mathbf{C}_I as the covariance matrices of the Out-of-Domain vector and In-Domain (InD) vector, ϕ as the OOD vector, CORAL is accomplished as follows:

$$\phi' = \mathbf{C}_I^{\frac{1}{2}} \mathbf{C}_o^{-\frac{1}{2}} \phi, \quad (3)$$

where

$$\mathbf{C}_o^{-\frac{1}{2}} = \mathbf{Q}_o \mathbf{\Lambda}_o^{-\frac{1}{2}} \mathbf{Q}_o^\top \quad (4)$$

represents the whitening process,

$$\mathbf{C}_I^{\frac{1}{2}} = \mathbf{Q}_I \mathbf{\Lambda}_I^{\frac{1}{2}} \mathbf{Q}_I^\top \quad (5)$$

represents the re-coloring process. \mathbf{Q} and $\mathbf{\Lambda}$ are the eigenvectors and eigenvalues of the covariance matrices.

In speaker verification systems, CORAL is often used on OOD embeddings. It transforms them into pseudo-in-domain vectors. Then, Probabilistic Linear Discriminant Analysis (PLDA) [7], a popular backend, will be re-trained on pseudo-in-domain vectors.

2.2.2 DNN Domain Adaptation

There are two main types of DNN domain adaptation methods in speaker verification systems, one is the metric-based method, and the other one is the generative-model-related method.

2.2.2.1 Metric-based Method

One of the most commonly used metrics in metric-based domain adaptation methods is the maximum mean discrepancy (MMD) [11].

- **Maximum Mean Discrepancy**

The Maximum Mean Discrepancy is used to measure how different the two distributions are.

It is shown as follows,

$$\text{MMD}[\mathcal{F}, p, q] := \sup_{f \in \mathcal{F}} \left(\mathbf{E}_{x \sim p} [f(x)] - \mathbf{E}_{y \sim q} [f(y)] \right). \quad (6)$$

The details of MMD will be introduced in the Methodology section.

- **Maximum Mean Discrepancy in Domain Adaptation**

Tzeng et al. [12] proposed a domain adaptation system that combines the MMD with the deep neural network. The MMD is used as a loss term during the training of the deep neural network. Fig. 3 shows the architecture of the proposed system.

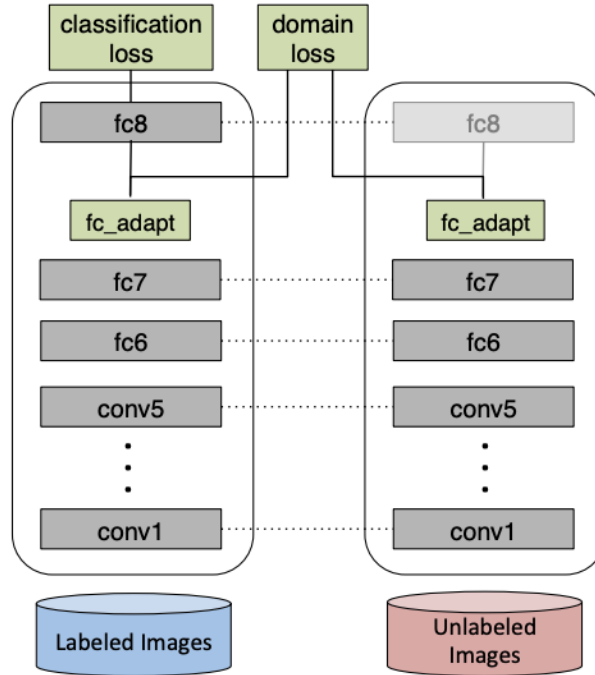


Figure 3: The architecture of [12]. It utilizes the MMD as a loss term during training.

Another metric-based method is [13], where the author proposed a framework using the MMD to accomplish speaker verification across languages. Similar to [12], when training the network, apart from the classification loss, a domain loss is added. In order to use data augmentation in the system, the author adds an additional consistency regularization loss. This system is able to perform unlabeled domain adaptation with the MMD loss.

2.2.2.2 Generative-model-related Method

As for generative models, there are some works using Generative Adversarial Networks (GAN) [14]. Besides, Variational Auto-encoders (VAEs) are also used, like in EDITNet [15], the author

uses a Conditional Variational Auto-encoder (CVAE) [16] to achieve domain adaptation. This report will mainly focus on the VAE solution. Details of the Variational Auto-encoder (VAE) are introduced in the following.

- **Variational Auto-encoder**

The Variational Auto-encoder (VAE) is a generative model with a prior and a noise probability distribution [17]. Its structure is similar to Auto-encoders. It has an encoder and a decoder. However, instead of using a latent vector like in Auto-encoders, VAE uses a latent distribution. This ensures the VAE can have a good property for output generation. Fig. 4 is a diagram illustrating the difference between simple Auto-encoders and Variational Auto-encoders.

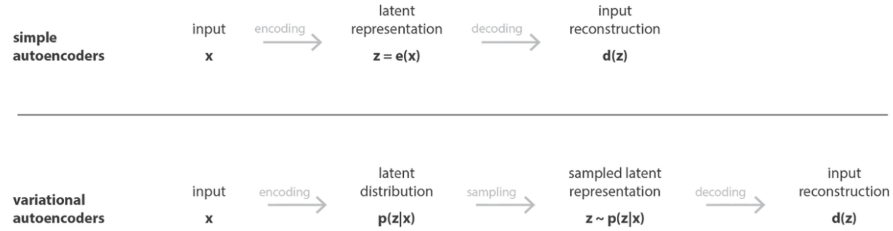


Figure 4: The comparison between simple Auto-encoders and Variational Auto-encoders [18].

The typical loss of a VAE contains two terms, one is the reconstruction loss, and the other one is the KL divergence loss. The reconstruction loss is similar to the loss of Auto-encoders.

It is calculated as follows:

$$\mathcal{L}_{recon} = \frac{1}{N} \sum_{n=1}^N |\mathbf{x}_n - \hat{\mathbf{x}}_n|, \quad (7)$$

where the N is the batch size, \mathbf{x}_n is the original input and $\hat{\mathbf{x}}_n$ is the reconstructed input.

The Kullback-Leibler divergence (KL divergence) measures how a probability distribution is different from a reference probability distribution [19]. In VAE, the KL divergence loss is used to measure how the latent distribution is different from a standard normal distribution, i.e., $\mathcal{N}(\mathbf{0}, \mathbf{I})$. This loss term is used to ensure the generation property of the VAE. It is calculated as follows:

$$\mathcal{L}_{KL} = -\frac{1}{N} \sum_{n=1}^N \frac{1}{2} \sum_{j=1}^J (1 + \log(\sigma_{n,j}^2) - \mu_{n,j}^2 - \sigma_{n,j}^2). \quad (8)$$

- **Variational Auto-encoders in Domain Adaptation**

There has been a number of applications of Variational Auto-encoders in domain adaptation. EDITNet [15] is a transform network that works on speaker embeddings extracted from the pre-trained network in the source domain. Based on a Conditional Variational Auto-encoder structure, it can transfer embeddings from the target domain to the source domain and does not require fine-tuning. The architecture of EDITNet [15] is shown in Fig. 5. EDITNet can transfer the target domain embedding to the source domain by first obtaining the mean and variance of the target domain embedding distribution and the corresponding latent vector through the encoder of the VAE, then performing the prior transfer, and then inputting the transferred latent vector to the decoder to get the transferred embedding. This is shown in Fig. 6.

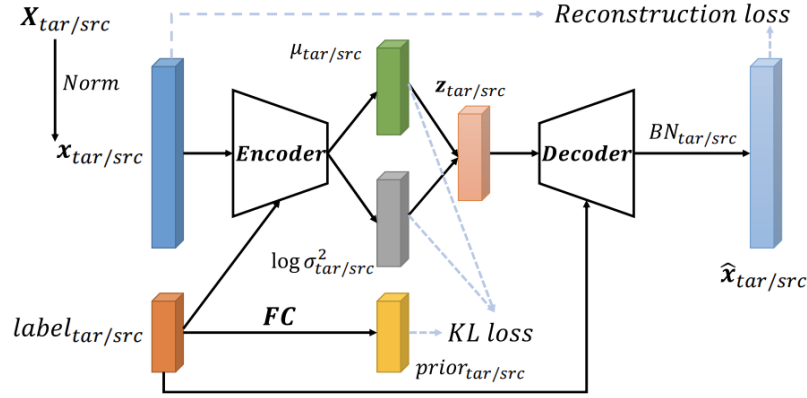


Figure 5: The architecture of EDITNet [15].

Algorithm 1 Transfer embedding \mathbf{x}_{tar} from target domain to source domain

- 1: $\mu_{tar}, \log \sigma_{tar}^2 = \text{Encoder}(\mathbf{x}_{tar}, \text{label}_{tar})$
 - 2: **if** training **then**
 - 3: $\mathbf{z}_{tar} \sim \mathcal{N}(\mu_{tar}, \sigma_{tar}^2)$ ▷ sampling
 - 4: **else**
 - 5: $\mathbf{z}_{tar} = \mu_{tar}$ ▷ for evaluation
 - 6: **end if**
 - 7: $\tilde{\mathbf{z}}_{tar} = \mathbf{z}_{tar} - \text{prior}_{tar} + \text{prior}_{src}$ ▷ transfer the latent variable into source domain
 - 8: $\tilde{\mathbf{x}}_{tar} = \text{BN}_{src}(\text{Decoder}(\tilde{\mathbf{z}}_{tar}, \text{label}_{src}))$
 - 9: **return** $\tilde{\mathbf{x}}_{tar}$ as the transferred embedding of \mathbf{x}_{tar}
-

Figure 6: The EDITNet algorithm of transferring the target domain embedding to source domain [15].

Another way of using VAE to achieve domain adaptation in speaker verification systems is

proposed by Wang et al. [20]. Since the commonly-used backend - PLDA requires training, there can be a domain mismatch in PLDA if the input data is not from the same domain as the training data. Therefore, the solution can be: ensuring the input domain data and the training domain data can be transferred into latent vectors which share the same latent distribution. This can be done by Variational Auto-encoders. Then, there will be less domain mismatch if the latent vector is inputted into the PLDA. The architecture of this system is shown in Fig. 7.

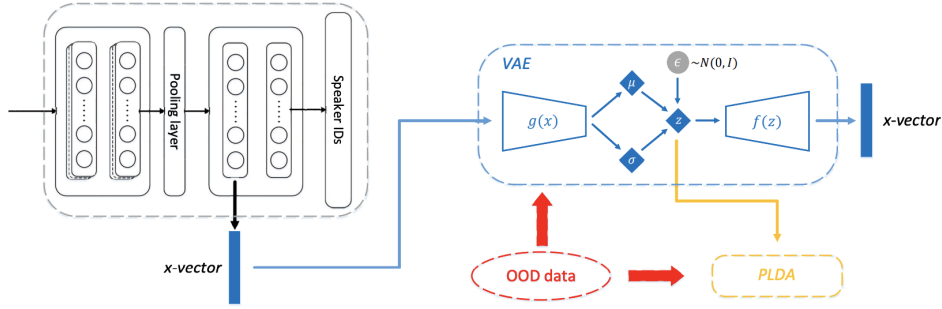


Figure 7: The architecture of the proposed VAE [20].

Lee et al. [21] proposed another way of achieving domain adaptation using Variational Auto-encoders. It proposed two separate losses for the encoder and the decoder, respectively. Although the authors proposed this method in the mechanical engineering field, some of their ideas can be applied to speaker verification systems. The structure of the model is shown in Fig. 8. It is worth noting that the training of this VAE system also consists of MMD loss, which is used for matching the two domains.

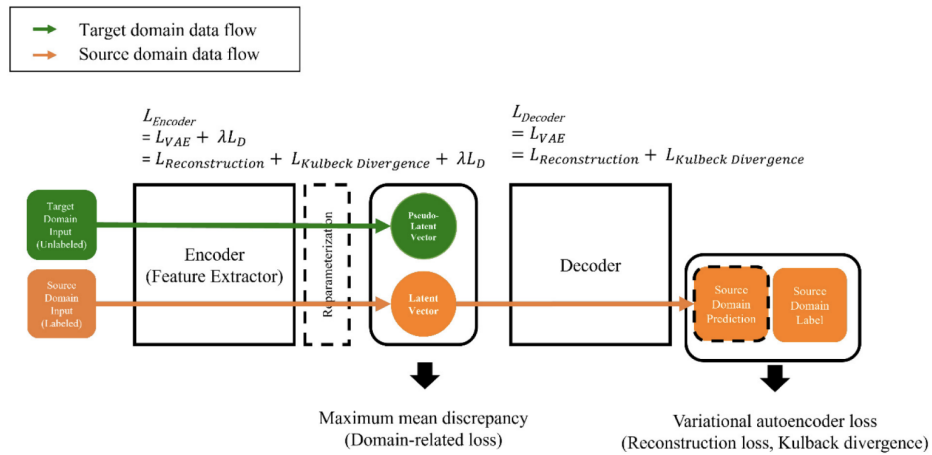


Figure 8: The training scheme and pipeline of [21].

3 Methodology

This project aims to utilize the MMD-based domain loss during the training of ECAPA-TDNN to alleviate the language mismatch problem in speaker verification systems. In the following sections, this proposed method will be focused on and explained in detail. Besides, a VAE-based method has also been explored but is not the main focus of this project. Therefore, the VAE-based method is explained only in one section, Section 3.6.

3.1 System Pipeline

In this project, ECAPA-TDNN [1], which is the current state-of-the-art speaker verification system, is used as the baseline model, and on top of that, metric-based domain adaptation is used to deal with language mismatch problems. The main steps for achieving the proposed system are as follows:

1. The model is based on the ECAPA-TDNN network with English-speaking speech data as the source domain, Mandarin-speaking speech data as the target domain;
2. The classification loss is the AAM-softmax [22]. To perform domain adaptation, the utterance-level domain loss and the frame-level domain loss using the maximum mean discrepancy metric are added;
3. The model is trained and evaluated on the VoxCeleb1 [23], VoxCeleb2 [24], CN-Celeb1 [25] and CN-Celeb2 [26] datasets.

The proposed system pipeline is shown in Fig. 9.

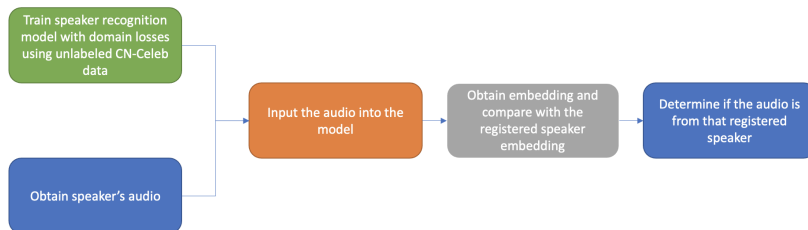


Figure 9: The proposed system pipeline for performing speaker verification.

3.2 Model Architecture

The model architecture of the system is based on ECAPA-TDNN. The system does not require a complex backend like PLDA [7] since the AAM-softmax loss is applied. Simple cosine similarity is sufficient for the scoring. In addition to the AAM-softmax, two extra domain losses are added. This loss structure is similar to the structure in [13], but without consistency regularization, as shown in Fig. 10.

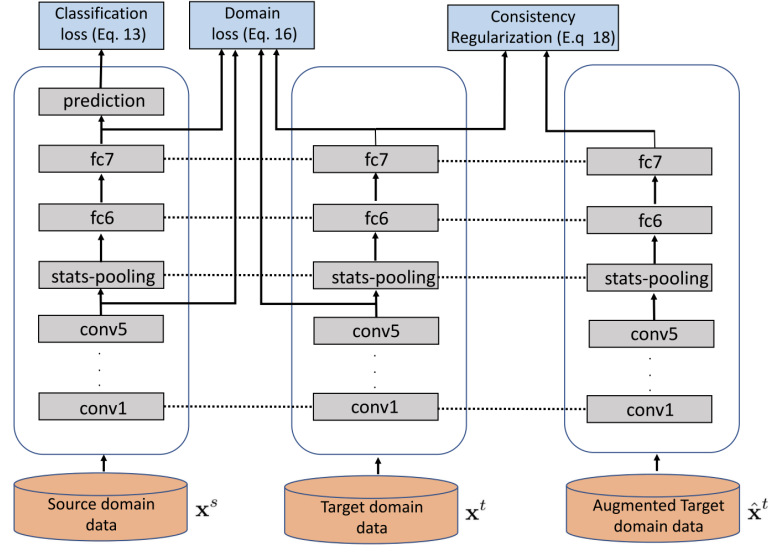


Figure 10: The network architecture in [13]. There are four terms in the loss function proposed in [13], the AAM-softmax, utterance-level MMD, frame-level MMD, and consistency regulation loss. In our proposed system, only AAM-softmax, utterance-level MMD, and frame-level MMD are used.

3.3 Additive Angular Margin Softmax

A modified softmax cross-entropy loss is:

$$\mathcal{L}_{softmaxCE} = -\frac{1}{N} \sum_{i=1}^N \log \frac{e^{\mathbf{s} \cdot \cos(\theta_{y_i})}}{\sum_{j=1}^C e^{\mathbf{s} \cdot \cos(\theta_j)}}, \quad (9)$$

where N is the batch size, C is the number of classes, \mathbf{s} is the normalized input embedding and θ is the angle between \mathbf{s} and the weight matrix \mathbf{W} .

In order to enhance the speaker-discriminative power, the inter-class margin should be maximized, and the intra-class margin should be minimized. Therefore, margin penalties can be added. Additive Angular Margin Softmax (AAM-softmax) [22] uses this idea. Given m is the margin

penalty, it is calculated as follows:

$$\mathcal{L}_{AAM-softmax} = -\frac{1}{N} \sum_{i=1}^N \log \frac{e^{\mathbf{s} \cdot \psi(\theta_{y_i})}}{e^{\mathbf{s} \cdot \psi(\theta_{y_i})} + \sum_{j=1, j \neq i}^C e^{\mathbf{s} \cdot \psi(\theta_j)}}, \quad (10)$$

where

$$\psi(\theta_{y_i}) = \cos(\theta_{y_i} + m). \quad (11)$$

With increased inter-class variability and reduced intra-class variability, the AAM-softmax is a good choice for the classification task. Using AAM-softmax can also avoid the complex PLDA [7] in the backend, and cosine similarity would be good enough. Therefore, following the ECAPA-TDNN [1], AAM-softmax is used as the classification loss in this project.

3.4 Maximum Mean Discrepancy

As mentioned before, the Maximum Mean Discrepancy is defined as:

$$\text{MMD}[\mathcal{F}, p, q] := \sup_{f \in \mathcal{F}} \left(\mathbf{E}_{x \sim p} [f(x)] - \mathbf{E}_{y \sim q} [f(y)] \right). \quad (12)$$

Its estimation is usually written as:

$$\text{MMD}[\mathcal{F}, X, Y] := \sup_{f \in \mathcal{F}} \left(\frac{1}{m} \sum_{i=1}^m f(x_i) - \frac{1}{n} \sum_{i=1}^n f(y_i) \right). \quad (13)$$

MMD can be represented by kernel functions $k(\cdot, \cdot)$:

$$\text{MMD}[X, Y] = \frac{1}{N^2} \sum_{i=1}^N \sum_{i'=1}^N k(x_i, x_{i'}) - \frac{2}{NM} \sum_{i=1}^N \sum_{j=1}^M k(x_i, y_j) + \frac{1}{M^2} \sum_{j=1}^M \sum_{j'=1}^M k(y_j, y_{j'}). \quad (14)$$

Some examples of the kernels in Equation 14 are quadratic kernels and Gaussian kernels.

MMD can be used to test the difference between two distributions p and q . The value of the smooth function $f(\cdot)$ is large when a sample is drawn from p and is small when it is drawn from q . The difference between the two distributions can be estimated by the difference between the mean function values of the two sample sets. When MMD is large, the samples are more likely to come from different distributions; when it is small, the samples are more likely to be of the same distribution.

In our project, multi-gaussian-kernel MMD is used as the cross-language loss. Given that Gaussian kernels can match up to infinitely many moments of the source and the target distributions, the Gaussian kernels are more powerful than the quadratic kernels [13]. Therefore, Gaussian kernels are used in the MMD loss instead of the quadratic kernels. The reason for using multiple kernels is to improve the chance of finding the appropriate kernels that are near the ideal ones. Besides, the loss function will not only consider the final embedding but also feature maps from other frame-level layers to optimize the effect [13]. By considering the frame-level feature maps, the divergence caused by duration can be mitigated.

3.5 Loss Function

The loss function of this network includes three parts, the classification loss in the source domain, the domain loss between source domain data and target domain data at the utterance level, and the domain loss at the frame level. The two domain losses all utilize the MMD metric. The loss function is as follows:

$$\mathcal{L} = \alpha \mathcal{L}_{\text{AAM-softmax}} + \beta \mathcal{L}_{\text{utterance-levelMMD}} + \gamma \mathcal{L}_{\text{frame-levelMMD}}, \quad (15)$$

where α , β and γ are the hyper-parameters.

3.6 VAE-based Method Exploration

In addition to the method proposed above, I also explored the VAE-based method in this final-year project.

3.6.1 VAE-based System Pipeline

The designed VAE-based system takes the embeddings of the target domain data extracted using the ECAPA-TDNN [1] as inputs. Then, the VAE transfers them into the source domain. These two embeddings will then be scored using cosine similarity.

The pipeline for training the VAE-based system is as follows:

1. Source domain audios and target domain audios are randomly selected and inputted into the ECAPA-TDNN network;

2. The output embeddings are then inputted into the variational auto-encoder network;
3. Reconstruction loss and KL divergence loss are calculated using the source domain data, and the domain loss is calculated using both the source domain data and the target domain data;
4. During training, only the weights of the VAE model are updated. The weights of the ECAPA-TDNN are fixed.

3.6.2 VAE Model Architecture

Similar to the architecture of EDITNet [15], the designed VAE architecture is a simple and classic one. It is shown in Fig. 11. The dimension of the input embeddings is 192, and the dimension of the latent vector is designed to be 128.

```
VAE(
  (encoder): Encoder(
    (MLP): Sequential(
      (L0): Linear(in_features=192, out_features=256, bias=True)
      (A0): ReLU()
      (BN0): BatchNorm1d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (L1): Linear(in_features=256, out_features=128, bias=True)
      (A1): Tanh()
    )
    (linear_means): Linear(in_features=128, out_features=128, bias=True)
    (linear_log_var): Linear(in_features=128, out_features=128, bias=True)
  )
  (decoder): Decoder(
    (MLP): Sequential(
      (L0): Linear(in_features=128, out_features=256, bias=True)
      (A0): ReLU()
      (BN0): BatchNorm1d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (L1): Linear(in_features=256, out_features=512, bias=True)
      (A1): ReLU()
      (BN1): BatchNorm1d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (L2): Linear(in_features=512, out_features=192, bias=True)
      (BN2): BatchNorm1d(192, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
  )
)
```

Figure 11: The designed VAE model architecture.

3.6.3 VAE Loss

As mentioned before, the usual VAE loss contains two parts, one is the reconstruction loss, and the other one is the KL divergence loss. The designed VAE loss in this project is only calculated using the source domain data. They are calculated as,

$$\mathcal{L}_{recon} = \frac{1}{N} \sum_{n=1}^N |\mathbf{x}_{src,n} - \hat{\mathbf{x}}_{src,n}|, \quad (16)$$

where the N is the batch size, $\mathbf{x}_{src,n}$ is the original source domain embedding and $\hat{\mathbf{x}}_{src,n}$ is the reconstructed source domain embedding.

$$\mathcal{L}_{KL} = -\frac{1}{N} \sum_{n=1}^N \frac{1}{2} \sum_{j=1}^J (1 + \log(\sigma_{src,n,j}^2) - \mu_{src,n,j}^2 - \sigma_{src,n,j}^2), \quad (17)$$

where $\mu_{src,n,j}$ is the mean of the source domain latent distribution and $\sigma_{src,n,j}$ is the standard deviation.

However, since the goal of this VAE is to achieve domain adaptation, a latent domain loss should be added. The designed domain loss utilizes the MMD metric. The MMD loss can be represented as,

$$\mathcal{L}_{latentMMD} = \text{MMD}(\mathbf{z}_{tgt}, \mathbf{z}_{src}), \quad (18)$$

where \mathbf{z}_{tgt} is the target domain latent vector produced by the encoder and \mathbf{z}_{src} is the source domain latent vector produced by the encoder.

The total loss is:

$$\mathcal{L} = \mathcal{L}_{recon} + \mathcal{L}_{KL} + \gamma_1 \mathcal{L}_{latentMMD}, \quad (19)$$

where γ_1 is the hyper-parameter.

4 Implementation

The implementation of the proposed system will be introduced first. In addition to the proposed system, the VAE-based method has also been explored. The implementation of the VAE-based will be introduced in one separate section, Section 4.7.

4.1 Dataset

Currently, the two main datasets required are VoxCeleb2 [24] and CN-Celeb1 [25]. The VoxCeleb2 training set with labels was used as the source domain data because it contains much more labeled data than the CN-Celeb1 dataset, and the original ECAPA-TDNN [1] model was trained on VoxCeleb2. The CN-Celeb1 training set without labels was used as the target domain data. For the evaluation process, the CN-Celeb1 evaluation set was used.

Since the VoxCeleb2 is in AAC format [27], FFmpeg [28] was used to convert AAC files to

WAV format [29] files. Since CN-Celeb1 is in FLAC format [30], FFmpeg was also used here to convert FLAC files to WAV files.

In addition, following the implementation code of ECAPA-TDNN, the MUSAN Dataset [31] and the Room Impulse Response and Noise Database [32] were used to perform data augmentation for VoxCeleb2 training.

4.2 Data Loader

The implementation of the proposed system was based on the code of ECAPA-TDNN [1]. In order to process the target domain data, I added a Data Loader for the CN-Celeb1 dataset. The detailed code is shown in Fig. 12.

```
class target_loader(object):
    def __init__(self, target_train_list, target_train_path, num_frames, **kwargs):
        self.train_path = target_train_path
        self.num_frames = num_frames
        # Load data & labels
        self.data_list = []
        lines = open(target_train_list).read().splitlines()
        for line in lines:
            file_name = os.path.join(target_train_path, line.split()[0])
            for filename in os.listdir(file_name):
                if filename.endswith("wav"):
                    fn = os.path.join(file_name, filename)
                    self.data_list.append(fn)

    def __getitem__(self, index):
        # Read the utterance and randomly select the segment
        audio, sr = soundfile.read(self.data_list[index])
        length = self.num_frames * 160 + 240
        if audio.shape[0] <= length:
            shortage = length - audio.shape[0]
            audio = numpy.pad(audio, (0, shortage), 'wrap')
        start_frame = numpy.int64(random.random()*(audio.shape[0]-length))
        audio = audio[start_frame:start_frame + length]
        audio = numpy.stack([audio], axis=0)
        return torch.FloatTensor(audio[0])

    def __len__(self):
        return len(self.data_list)
```

Figure 12: The detailed code of the Data Loader of CN-Celeb1.

4.3 Iterating Over Two Data Loaders

Since the proposed system needs to perform domain adaptation, the training process requires two data loaders, one for the source domain data - VoxCeleb2, and the other one for the target domain data - CN-Celeb1. Therefore, the code needs to iterate over the two data loaders during training. Two methods have been explored for achieving this goal.

The logic of the first method is shown in Fig. 13. The *dataloader1* should be the data loader of VoxCeleb2, and the *dataloader2* should be the data loader of CN-Celeb1. Since the CN-Celeb1 dataset is smaller than the VoxCeleb2 dataset, *cycle()* function is used here to iterate over the CN-Celeb1 again so that the *dataloader2* will never run out of data. However, this method did not work out. There are mainly two reasons for failure:

1. The data loader of CN-Celeb1 will not shuffle again if using the *cycle()* function.
2. The usage of *cycle()* and *zip()* might have memory leakage.

Therefore, I gave up this method and tried to find a new solution.

```
for item1, item2 in zip(dataloader1, cycle(dataloader2)):
    data1, label1 = item1
    data2, label2 = item2
```

Figure 13: The logic of the first method.

After searching for solutions online, following the method proposed on StackOverflow [33], the logic of the second method for iterating over two data loaders is shown in Fig. 14. Using *iter()* and *enumerate()* worked perfectly in this case. This method was chosen as the final solution to ensure the proposed method works smoothly.

4.4 MMD Loss

Following the code of [13], the Maximum Mean Discrepancy Loss was implemented as in Fig. 15. Since multiple Gaussian kernels are used, different σ s are chosen, referring to the σ s in the code of [13]. As mentioned before, the MMD loss contains two parts, one is the utterance-level MMD loss, and the other one is the frame-level MMD loss. The utterance-level MMD loss simply takes

```

for num, data1 in enumerate(source_loader):
    try:
        data2 = next(data_loader_iterator)
    except StopIteration:
        data_loader_iterator = iter(target_loader)
        data2 = next(data_loader_iterator)

```

Figure 14: The logic of the second method.

the output of the last layer, and the frame-level MMD loss takes the flattened output of the last layer before statistics pooling.

4.5 Loss Function

Currently, the implemented loss function is as follows:

$$\mathcal{L} = \alpha \mathcal{L}_{\text{AAM-softmax}} + \beta \mathcal{L}_{\text{utterance-levelMMD}} + \gamma \mathcal{L}_{\text{frame-levelMMD}}, \quad (20)$$

where α , β and γ are the hyper-parameters.

The loss function, as mentioned before, has multiple terms. In order to achieve satisfactory learning results for both speaker verification and domain adaptation, some fine-tuning of the hyper-parameters of loss terms is necessary. The most straightforward way is to add different weights to loss terms. At this stage, the hyper-parameter for AAM-softmax, α , is set to 1; the hyper-parameter for two MMD losses, β and γ , are both set to 100.

```

def mmd(self, x, y, sigmas=SIGMAS):
    cost = gaussian_kernel(x, x, sigmas) \
          + gaussian_kernel(y, y, sigmas) \
          - 2 * gaussian_kernel(x, y, sigmas)
    return cost

```

Figure 15: The Maximum Mean Discrepancy Loss.

4.6 Multi-GPU Training

Due to the limited memory of one single GPU, multi-GPU training is performed. In the experiments, the GPU used is GeForce RTX 2080 Ti, and its memory is only 11G. By using two 2080 Ti GPUs, the memory available can achieve 22G, and the project can now have a batch size of 160. *DataParallel()* method is used to allow multi-GPU training. The related code is shown in Fig. 16.

```
## ECAPA-TDNN
self.speaker_encoder = nn.DataParallel(ECAPA_TDNN(C = C)).to(device)
```

Figure 16: Multi-GPU training code.

There are some points worth mentioning in using *DataParallel()* to perform multi-GPU training. First of all, the pre-trained model may be trained using a single GPU. Therefore, the pre-trained model may have different names if the system now uses *DataParallel()*. For example, in the pre-trained model provided by the ECAPA-TDNN code, since it was trained using one GPU, its weights have names like *"speaker_encoder.layer1.convs.0.weight"*. But if we use *DataParallel()* in our code now, the names of our model become something like *"speaker_encoder.module.layer1.convs.0.weight"*. Therefore, I used the following code in Fig. 17 to rename the names in the pre-trained model, so that it can be successfully loaded.

```
if name not in self_state:
    name = name.replace("speaker_encoder.", "speaker_encoder.module.")
```

Figure 17: Code for loading a pre-trained model trained using one GPU in a multi-GPU setting.

4.7 VAE-based Method Implementation

4.7.1 Reparameterization

To enable backpropagation in VAE, the reparameterization trick should be used. At first, an ϵ is randomly sampled from a standard normal distribution. Based on the mean and standard deviation obtained from the encoder, the latent vector is the mean plus standard deviation times the ϵ . The code in Fig. 18 shows the implementation of reparameterization.

```
def reparameterize(self, mu, log_var):

    std = torch.exp(0.5 * log_var)
    eps = torch.randn_like(std)

    return mu + eps * std
```

Figure 18: Code for reparameterization in VAE.

4.7.2 VAE Loss

As mentioned in Section 3.6, VAE loss contains the reconstruction loss and the KL divergence loss. They are implemented as shown in Fig. 19.

```
def loss_vae(recon_x, x, mean, log_var):
    MSE = torch.nn.MSELoss(reduction="sum")
    recon_x = recon_x.view(args.batch_size, -1)
    x = x.view(args.batch_size, -1)
    loss_recons = MSE(x, recon_x)

    KLD = torch.sum(-0.5 * (1 + log_var - mean.pow(2) - log_var.exp()))
    return loss_recons/args.batch_size, KLD/args.batch_size
```

Figure 19: Code for implementing VAE loss.

5 Experiments and Evaluation

5.1 Evaluation Metrics

For speaker verification systems, multiple evaluation metrics can be used. In this project, the equal error rate (EER) and minimum decision cost function (minDCF) will be mainly used for evaluation.

5.1.1 Equal Error Rate

The equal error rate is closely related to the false acceptance rate (FAR) and the false rejection rate (FRR).

$$\text{False acceptance rate (FAR)} = \frac{\text{Number of false acceptance}}{\text{Number of trials}}, \quad (21)$$

$$\text{False rejection rate (FRR)} = \frac{\text{Number of false rejection}}{\text{Number of trials}}. \quad (22)$$

The point at which the FAR = FRR is known as the equal error rate (EER) [34]. The smaller the EER is, the better the performance is. The EER assumes that false alarms and missed alarms have the same cost to the system, and it does not contain the prior probability of the target. Therefore, considering the limitations of the EER, the decision cost function is added to the evaluation metrics used in this project.

5.1.2 Minimum Decision Cost Function

The FRR and FAR are weighted together to form the decision cost function (DCF) [35]. The equation is:

$$DCF = C_{miss} \cdot P_{target} \cdot FRR + C_{FalseAlarm} \cdot (1 - P_{target}) \cdot FAR, \quad (23)$$

where C_{miss} is the cost of false rejection, $C_{FalseAlarm}$ is the cost of false acceptance and P_{target} is the prior probability of being a target. Besides, the FRR and FAR here are related to the thresholds. MinDCF is the minimum DCF among all DCFs with different pairs of FRR and FAR values.

5.2 Experiments

5.2.1 Results

5.2.1.1 Proposed System

The experiments were conducted using VoxCeleb2 and CN-Celeb1 training sets as training data, and the CN-Celeb1 evaluation set as evaluation data. Besides, the pre-trained model provided by the implementation code of ECAPA-TDNN [1] was used as the initial model. This pre-trained model was trained only using the VoxCeleb2 data, i.e., English-speaking data. The training was performed based on this initial model.

The EER and minDCF results of directly applying the pre-trained ECAPA-TDNN model to the CN-Celeb1 evaluation data are shown in Table 1. It can be seen that the results of directly applying the model trained on VoxCeleb2 are unsatisfactory.

The results of the proposed system (currently implemented version) are shown in Table 2.

Metric	ECAPA-TDNN (pre-trained on VoxCeleb2)
EER	13.74%
minDCF	0.4996

Table 1: The evaluation results of the pre-trained ECAPA-TDNN model on the CN-Celeb1 evaluation set

Compared with Table 1, it is obvious that the proposed model has better performance.

Training Epoch	Metric	Proposed System (currently implemented)
1	EER	12.67%
	minDCF	0.4670
2	EER	12.63%
	minDCF	0.4603
3	EER	12.77%
	minDCF	0.4738
4	EER	12.75%
	minDCF	0.4659
5	EER	12.62%
	minDCF	0.4559

Table 2: **Proposed System:**The evaluation results of the proposed model on the CN-Celeb1 evaluation set

5.2.1.2 VAE-based System

In addition to the proposed system, a VAE-based method has also been explored. The experiments of the VAE-based method were conducted using VoxCeleb2 and CN-Celeb1 training sets as training data, and the CN-Celeb1 evaluation set as evaluation data. The embedding extraction model is the pre-trained ECAPA-TDNN model. The pre-trained model was trained only using the VoxCeleb2 data.

The EER and minDCF results of the initial ECAPA-TDNN model and the implemented VAE-based system are shown in Table 3.

Metric	ECAPA-TDNN	ECAPA-TDNN + VAE
EER	13.74%	45.15%
minDCF	0.4996%	1.0000

Table 3: **VAE-based System:** The evaluation results of the pre-trained ECAPA-TDNN model and the ECAPA-TDNN + VAE system on the CN-Celeb1 evaluation set

It can be seen that the VAE-based system performs poorly. This will be discussed in the

following section.

5.2.2 Discussion

There are several points to be discussed and further investigated in this project.

5.2.2.1 Proposed System

1. It can be observed from the above Table 2 that the performance does not necessarily improve with more training epochs. This is probably because the training for domain adaptation quickly converges, and the training becomes more and more toward the classification task. Therefore, it is important to decide when to stop training to ensure good performance.
2. More ablation studies need to be conducted to evaluate the effect of each module. For example, to evaluate the effect of multilevel aggregation in domain adaptation, the performance of the system with and without aggregation needs to be recorded.
3. Data augmentation is crucial in x-vector systems [2]. Therefore, it is also beneficial to conduct data augmentation for the target domain data. To make sure the augmented data are consistent with the unlabeled target domain data, consistency regularization is also needed for data augmentation. This can be further explored and implemented in the future.

5.2.2.2 VAE-based System

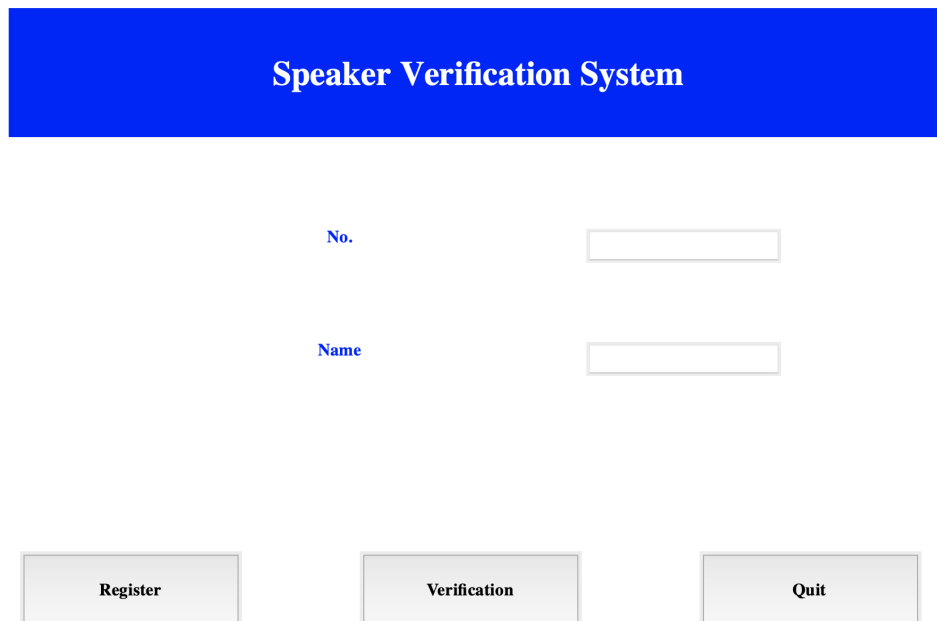
From Table 3, it can be observed that by adding a VAE module, the system performs poorly. There are several possible reasons.

1. The first possible reason is that the designed VAE model is not sufficient for this task. The designed VAE model only contains Multilayer Perceptrons. The performance may be improved if CNNs are used. Besides, the learning rate, latent vector dimension, and other hyper-parameters need to be carefully selected.
2. The second possible reason is the MMD metric may not be appropriate for the latent vector. The possible solutions are changing to a different metric or "learning" the metric directly using the generative model.

3. There might be some unnoticed errors or bugs in the current implementation code of the VAE. This requires a careful code review in the future.
4. The performance may be improved if the training of the encoder and decoder of the VAE is separated, just as the training scheme mentioned in [21]. This is because the decoder needs to reconstruct the source domain embedding from the latent vector. The decoder does not need to consider the target domain embedding. Therefore, the decoder should be optimized only using the VAE loss, i.e., the reconstruction loss and KL divergence loss of the source domain data. On the other hand, the encoder needs to transfer the source domain embedding as well as the target domain embedding into the same latent distribution. Therefore, the MMD loss should be added during the training of the encoder, i.e., the encoder should be optimized based on the VAE loss and the MMD loss.

5.2.3 Demonstration

For demonstration, I implemented a speaker verification system using GUI. It takes the microphone inputs as the enrollment audios and test audios. The speaker verification model used in this demonstration is the proposed model, which was trained using the AAM-softmax and proposed domain losses for five epochs.



Speaker Verification System

No.

Name

Figure 20: The interface of the speaker verification system.

The interface of this speaker verification system is shown in Fig. 20. The user is first asked to

input an ID and a name. These will be used to keep track of the speaker's information. The user can choose either to register the system or to verify the speaker. If the user chooses to register, then he/she will be required to speak to the microphone for 5 seconds. This will be recorded and saved in the "Enrollment" folder. Fig. 21 shows the code for achieving this. If the user chooses to verify the speaker, then he/she will be required to speak to the microphone for 5 seconds as well. This will be recorded and saved in the "Testing" folder. Then the system will input the testing audio and corresponding enrollment audio into the speaker verification model and compare the output embeddings. A score will then be calculated. If the score is higher than the threshold, the system will mark that the test audio is from the same speaker as the corresponding enrollment audio. This threshold can be obtained from the validation set of the CN-Celeb1 dataset. It is the corresponding threshold at the EER point.

```
showinfo("Message", "Once you press OK, the system will start recording.")
fs=16000
duration = 5 # seconds
myrecording = sd.rec(int(duration * fs), samplerate=fs, channels=1)
sd.wait()
showinfo("Message", "Done recording.")

WAVE_OUTPUT_FILENAME = "Enrollment/"+name + "_" + Id + ".wav"

soundfile.write(WAVE_OUTPUT_FILENAME, myrecording, fs)
```

Figure 21: The code for recording audio and saving it.

6 Conclusion

This project proposed a new language-independent speaker verification system. Based on the state-of-the-art ECAPA-TDNN [1] model, the proposed model is trained with two extra different-level domain losses. While the initial model is trained on English-speaking data, the proposed model is trained on the labeled English-speaking data and unlabeled Chinese-speaking data. The performance of the proposed system is shown to be better than the original ECAPA-TDNN model on Chinese-speaking data.

An extra exploration of the VAE-based model is also worth mentioning. Although the designed VAE-based model performs poorly, it is still beneficial to make such an attempt. Further adjustments to the architecture or the training scheme of the VAE-based model can be made.

7 References

- [1] B. Desplanques, J. Thienpondt, and K. Demuynck, “Ecapa-tdnn: Emphasized channel attention, propagation and aggregation in tdnn based speaker verification,” *arXiv preprint arXiv:2005.07143*, 2020.
- [2] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, “X-vectors: Robust dnn embeddings for speaker recognition,” in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5329–5333.
- [3] A. Nagrani, J. S. Chung, J. Huh, A. Brown, E. Coto, W. Xie, M. McLaren, D. A. Reynolds, and A. Zisserman, “Voxsrc 2020: The second voxceleb speaker recognition challenge,” *arXiv preprint arXiv:2012.06867*, 2020.
- [4] J. Hu, L. Shen, and G. Sun, “Squeeze-and-excitation networks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7132–7141.
- [5] A. Brown, J. Huh, J. S. Chung, A. Nagrani, and A. Zisserman, “Voxsrc 2021: The third voxceleb speaker recognition challenge,” *arXiv preprint arXiv:2201.04583*, 2022.
- [6] W. M. Kouw and M. Loog, “A review of domain adaptation without target labels,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 3, pp. 766–785, 2019.
- [7] S. J. Prince and J. H. Elder, “Probabilistic linear discriminant analysis for inferences about identity,” in *Proceedings of the IEEE 11th International Conference on Computer Vision*, 2007, pp. 1–8.
- [8] B. Sun, J. Feng, and K. Saenko, “Return of frustratingly easy domain adaptation,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, 2016.
- [9] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, “Front-end factor analysis for speaker verification,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788–798, 2010.
- [10] K. A. Lee, Q. Wang, and T. Koshinaka, “The coral+ algorithm for unsupervised domain adaptation of plda,” in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 5821–5825.
- [11] A. Gretton, K. Borgwardt, M. Rasch, B. Schölkopf, and A. Smola, “A kernel method for the two-sample-problem,” *Advances in Neural Information Processing Systems*, vol. 19, 2006.
- [12] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell, “Deep domain confusion: Maximizing for domain invariance,” *arXiv preprint arXiv:1412.3474*, 2014.
- [13] W. Lin, M.-W. Mak, N. Li, D. Su, and D. Yu, “A framework for adapting dnn speaker embedding across languages,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 2810–2822, 2020.

- [14] J. Rohdin, T. Stafylakis, A. Silnova, H. Zeinali, L. Burget, and O. Plchot, “Speaker verification using end-to-end adversarial language adaptation,” in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 6006–6010.
- [15] J. Li, W. Liu, and T. Lee, “Editnet: A lightweight network for unsupervised domain adaptation in speaker verification,” *arXiv preprint arXiv:2206.07548*, 2022.
- [16] K. Sohn, H. Lee, and X. Yan, “Learning structured output representation using deep conditional generative models,” *Advances in Neural Information Processing Systems*, vol. 28, 2015.
- [17] L. Pinheiro Cinelli, M. Araújo Marins, E. A. Barros da Silva, and S. Lima Netto, “Variational autoencoder,” in *Variational Methods for Machine Learning with Applications to Deep Networks*. Springer, 2021, pp. 111–149.
- [18] J. Rocca. Understanding variational autoencoders (vae). [Online]. Available: <https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73>.
- [19] S. Kullback and R. A. Leibler, “On information and sufficiency,” *The annals of mathematical statistics*, vol. 22, no. 1, pp. 79–86, 1951.
- [20] X. Wang, L. Li, and D. Wang, “Vae-based domain adaptation for speaker verification,” in *2019 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. IEEE, 2019, pp. 535–539.
- [21] S. M. Lee, S.-Y. Park, and B.-H. Choi, “Application of domain-adaptive convolutional variational autoencoder for stress-state prediction,” *Knowledge-Based Systems*, vol. 248, p. 108827, 2022.
- [22] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, “Arcface: Additive angular margin loss for deep face recognition,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 4690–4699.
- [23] A. Nagrani, J. S. Chung, and A. Zisserman, “Voxceleb: a large-scale speaker identification dataset,” *arXiv preprint arXiv:1706.08612*, 2017.
- [24] J. S. Chung, A. Nagrani, and A. Zisserman, “Voxceleb2: Deep speaker recognition,” *arXiv preprint arXiv:1806.05622*, 2018.
- [25] Y. Fan, J. Kang, L. Li, K. Li, H. Chen, S. Cheng, P. Zhang, Z. Zhou, Y. Cai, and D. Wang, “Cn-celeb: a challenging chinese speaker recognition dataset,” in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 7604–7608.
- [26] L. Li, R. Liu, J. Kang, Y. Fan, H. Cui, Y. Cai, R. Vipplera, T. F. Zheng, and D. Wang, “Cn-celeb: multi-genre speaker recognition,” *Speech Communication*, vol. 137, pp. 77–91, 2022.

- [27] K. Brandenburg, “Mp3 and aac explained,” in *Audio Engineering Society Conference: 17th International Conference: High-Quality Audio Coding*. Audio Engineering Society, 1999.
- [28] Ffmpeg. [Online]. Available: <http://ffmpeg.org>.
- [29] Wikipedia. Wav. [Online]. Available: <https://en.wikipedia.org/wiki/WAV>.
- [30] ——. Flac. [Online]. Available: <https://en.wikipedia.org/wiki/FLAC>.
- [31] D. Snyder, G. Chen, and D. Povey, “MUSAN: A Music, Speech, and Noise Corpus,” 2015, arXiv:1510.08484v1.
- [32] T. Ko, V. Peddinti, D. Povey, M. L. Seltzer, and S. Khudanpur, “A study on data augmentation of reverberant speech for robust speech recognition,” in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 5220–5224.
- [33] Afroditi. How to iterate over two dataloaders simultaneously using pytorch? [Online]. Available: <https://stackoverflow.com/questions/51444059/how-to-iterate-over-two-dataloaders-simultaneously-using-pytorch>.
- [34] M.-W. Mak and J.-T. Chien, *Machine learning for speaker recognition*. Cambridge University Press, 2020.
- [35] A. Martin, G. Doddington, T. Kamm, M. Ordowski, and M. Przybocki, “The det curve in assessment of detection task performance,” National Inst of Standards and Technology Gaithersburg MD, Tech. Rep., 1997.

The Hong Kong Polytechnic University
Department of Electronic and Information Engineering

EIE4430 / EIE4433 Honours Project

1. Student Name: FANG Jiaying (Student No.: 19078474D)
2. Programme Code: 42470 / 42477
3. Project Title: Deep Speaker Embedding Across Languages
4. Supervisor Name: Prof. Mak Man-Wai
5. Project summary (State clearly the project objectives and results achieved by yourself.)
[Please list in point form where appropriate]

This project aims to achieve language-independent speaker verification by incorporating language-mismatch loss into deep neural network (DNN) models.

In this project, I achieved the following results:

1. Explored different techniques in speaker verification and domain adaptation.
2. Learn about the state-of-the-art speaker recognition and verification model – ECAPA-TDNN.
3. Based on the ECAPA-TDNN, proposed a speaker verification model with language-mismatch loss.
4. Implemented the language-mismatch loss, data loaders for different datasets and the training function with two data loaders (source domain data loader and target domain data loader).
5. Trained the proposed model on the CN-Celeb (Chinese-speaking) and the VoxCeleb (English-speaking) datasets.
6. Evaluated the model on the CN-Celeb dataset.
7. Explored another approach of alleviating language-mismatch problem in speaker verification systems – Variational Auto-encoder approach.
 - a. Designed a simple VAE-based speaker verification system.
 - b. Implemented and evaluated the VAE-based system on the CN-Celeb and VoxCeleb datasets.
8. Discussed the evaluation results.

DECLARATION OF ORIGINALITY

Except where reference is made in the text of this report, I declare that this report contains no material published elsewhere or extracted in whole or in part from any works or assignments presented by me or any other parties for another subject. In addition, it has not been submitted for the award of any other degree or diploma in any other tertiary institution.

No other person's work has been used without due acknowledgement in the main text of the Report.

I fully understand that any discrepancy from the above statements will constitute a case of plagiarism and be subject to the associated.

FANG Jiaying

Signature