Proceedings Paper:

Shi, Y. and Hain, T. orcid.org/0000-0003-0939-3464 (2021) Supervised speaker embedding de-mixing in two-speaker environment. In: 2021 IEEE Spoken Language Technology Workshop (SLT). 2021 IEEE Spoken Language Technology Workshop (SLT), 19-22 Jan 2021, Shenzhen, China. Institute of Electrical and Electronics Engineers , pp. 758-765. ISBN 9781728170671

https://doi.org/10.1109/slt48900.2021.9383580

SUPERVISED SPEAKER EMBEDDING DE-MIXING IN TWO-SPEAKER ENVIRONMENT

Yanpei Shi, Thomas Hain

Speech and Hearing Research Group
Department of Computer Science, University of Sheffield
{YShi30, t.hain}@sheffield.ac.uk

ABSTRACT

Separating different speaker properties from a multi-speaker environment is challenging. Instead of separating a two-speaker signal in signal space like speech source separation, a speaker embedding de-mixing approach is proposed. The proposed approach separates different speaker properties from a two-speaker signal in embedding space. The proposed approach contains two steps. In step one, the clean speaker embeddings are learned and collected by a residual TDNN based network. In step two, the two-speaker signal and the embedding of one of the speakers are both input to a speaker embedding de-mixing network. The de-mixing network is trained to generate the embedding of the other speaker by reconstruction loss. Speaker identification accuracy and the cosine similarity score between the clean embeddings and the de-mixed embeddings are used to evaluate the quality of the obtained embeddings. Experiments are done in two kind of data: artificial augmented two-speaker data (TIMIT) and real world recording of two-speaker data (MC-WSJ). Six different speaker embedding de-mixing architectures are investigated. Comparing with the performance on the clean speaker embeddings, the obtained results show that one of the proposed architectures obtained close performance, reaching 96.9% identification accuracy and 0.89 cosine similarity.

Index Terms— Speaker Embeddings, Speech Source Separation, Speaker De-mixing, Speaker Identification, Two-Speaker Signal.

1. INTRODUCTION

In recent years, speech source separation becomes an active research area. Speech source separation separates mixture speech signal in signal space. Traditionally, speech source separation is viewed as a signal processing problem, different approaches are proposed such as CASA [1]. Matrix factorization methods are also widely used in speech source separation, such as Non-Negative Matrix Factorization (NMF) [2, 3] and Independent component analysis (ICA) [4, 5, 6, 7, 8]. With the rapid growth of deep learning, some deep learning approaches are used to separate speech signals, such as supervised separation [9, 10, 11, 12], deep clustering and deep attractor network [13, 14, 15].

However, separating speech signal from two-speaker signal is still a challenging task. Speech signals are high dimensional, and different speaker properties in two-speaker signals are highly co-related to each other, which would influence the quality of the output [15, 7].

Instead of separating speech signal in signal space, de-mixing different speaker properties from two-speaker signal in embedding space might be more efficient. Speaker embedding is low dimensional, and it can project variable length acoustic signal into fixed length embedding space [16]. This property of speaker embedding makes it convenient to be further used comparing with that in signal space. The obtained speaker embeddings might be beneficial for downstream tasks such as speaker identification [17, 18, 19, 20] and speech recognition [21, 22].

In this work, a speaker embedding de-mixing approach for separating speaker embeddings in two-speaker signal is proposed. The proposed approach contains two steps: in step one, a residual TDNN network is used to learn high quality speaker embeddings from clean speech data. After training, the embedding of each speaker are extracted and collected. In step two, a speaker embedding de-mixing network is trained. Suppose the input data contains a target speaker and an interfering speaker. The proposed approach takes the two-speaker signal as input, as well as the embedding of the interfering speaker. The output would be the embedding of the target speaker; or inversely, the proposed approach takes the two-speaker mixture signal and the embedding of the target speaker as input, the obtained embedding would be the embedding of the interfering speakers. When the embedding of one of the speaker is available, the system will generate the embedding of the other speaker that appears in the input signal.

To the best of our knowledge, the proposed approach is the first that trying to directly de-mix speaker embedding from two-speaker signal. This is also the main contribution of this work. The benefits of the proposed approach is manifold: Suppose in a home device, the embedding of some speakers might be available. The proposed approach might be beneficial for obtaining the embedding of the other speaker in two-speaker signal. The de-mixed speaker embedding might be further used for some downstream tasks, such as speaker verification [23, 24] and speech recognition [25].

The rest of this report is organized as follow: Section 2 introduces the model architectures in both step one and step two. Section 3 introduces the experiments design, including data and use, and experiment setup. In Section 4, results are shown, followed by discussion and analysis. Section 5 introduces the conclusion and the future work plan.

2. MODEL ARCHITECTURE

In this section, the model structure in this work is introduced, which consists of two steps. Step one: learning clean speaker representation; Step two: using the learned speaker embedding to train a speaker embedding de-mixing network. The goal for step one is to learn high quality embeddings for each speaker in the dataset. In step two, the two-speaker signal is firstly projected into embedding space, resulting in mixture embedding e_{mix}. The mixture embedding and the embedding of one of the speakers e_2 are put into a de-mixing function. The output is the estimation of the embedding from the other speaker $e_1^{'}$.

2.1. Step One: Learning High Quality Speaker Representations

In step one, the clean speech signal is input to a speaker embedding extractor $\boldsymbol{A}$. After training $\boldsymbol{A}$, the embedding for each speaker is extracted from the bottleneck layer of $\boldsymbol{A}$. A classifier $\boldsymbol{B}$ is used to evaluate the quality of the learned speaker embeddings.

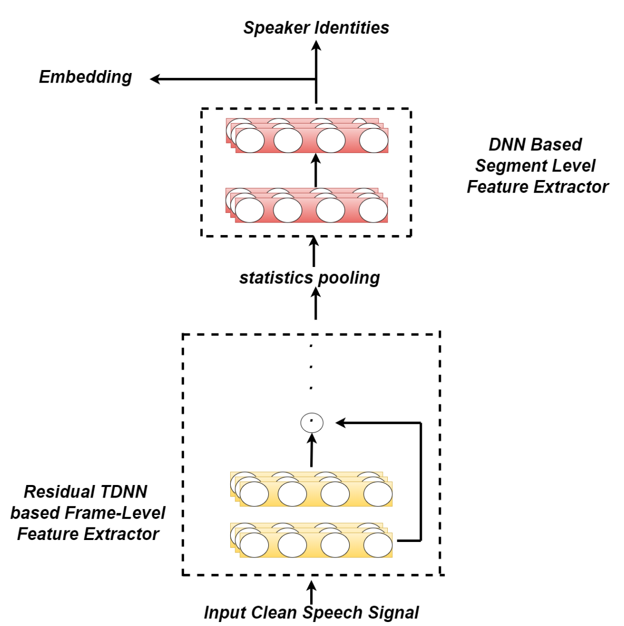

Fig. 1. The architecture of speaker embedding extractor $\boldsymbol{A}$.

Figure 1 and Table 1 show the architecture of $\boldsymbol{A}$. In order to learn high quality and robust speaker embeddings, $\boldsymbol{A}$ is designed based on TDNN architecture, as TDNN architecture shows high robustness and it can better capture time relevant

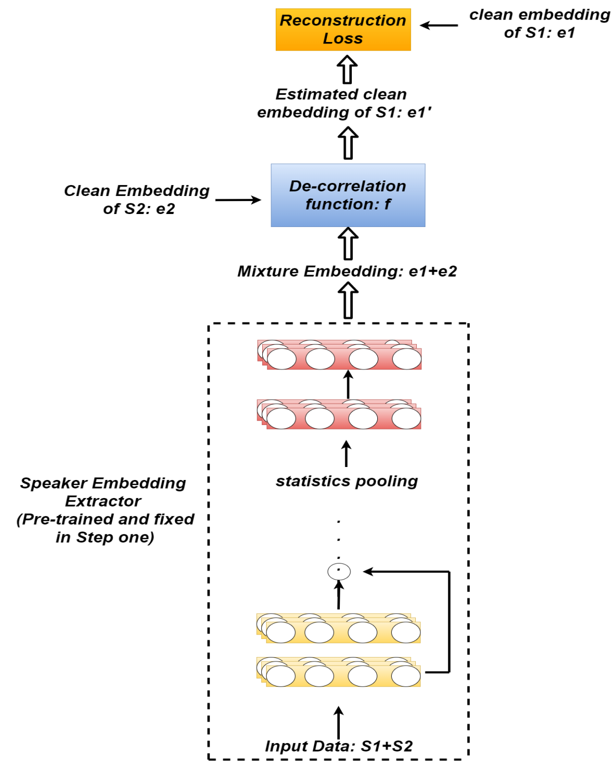

Fig. 2. Model Architecture of speaker embedding de-mixing network $\boldsymbol{C}$. $\boldsymbol{C}$ consists of pre-trained speaker embedding extractor and the de-mixing function.

information [26]. There are three parts within the architecture of $\boldsymbol{A}$: frame-level feature extractor, statistics pooling and segment-level feature extractor.

In frame-level feature extractor, the network consists of TDNN layers and residual TDNN blocks. The input data is firstly passed through into two TDNN layers. Then, three residual TDNN blocks are used. The last TDNN layer transforms the feature dimension into 1500. The use of residual TDNN blocks instead of using normal TDNN layers like X-vectors might increase the robustness of the learned embeddings [27].

Statistics pooling operation is then used, the output is feed into the segment-level feature extractor. There are two fully-connected layers in segment-level feature extractor. The speaker embedding is extracted from the last fully-connected layer.

For the architecture of classifier $\boldsymbol{B}$, a simply architecture is chosen: a fully connected network with one hidden layer with 512 nodes.

2.2. Step Two: The De-mixing of Speaker Representations in Embedding Space

After collecting the high quality embeddings for each speaker in step one, step two learns the de-mixing function of the mixture embeddings.

Suppose the input data contains two speakers: s_1 and s_2.

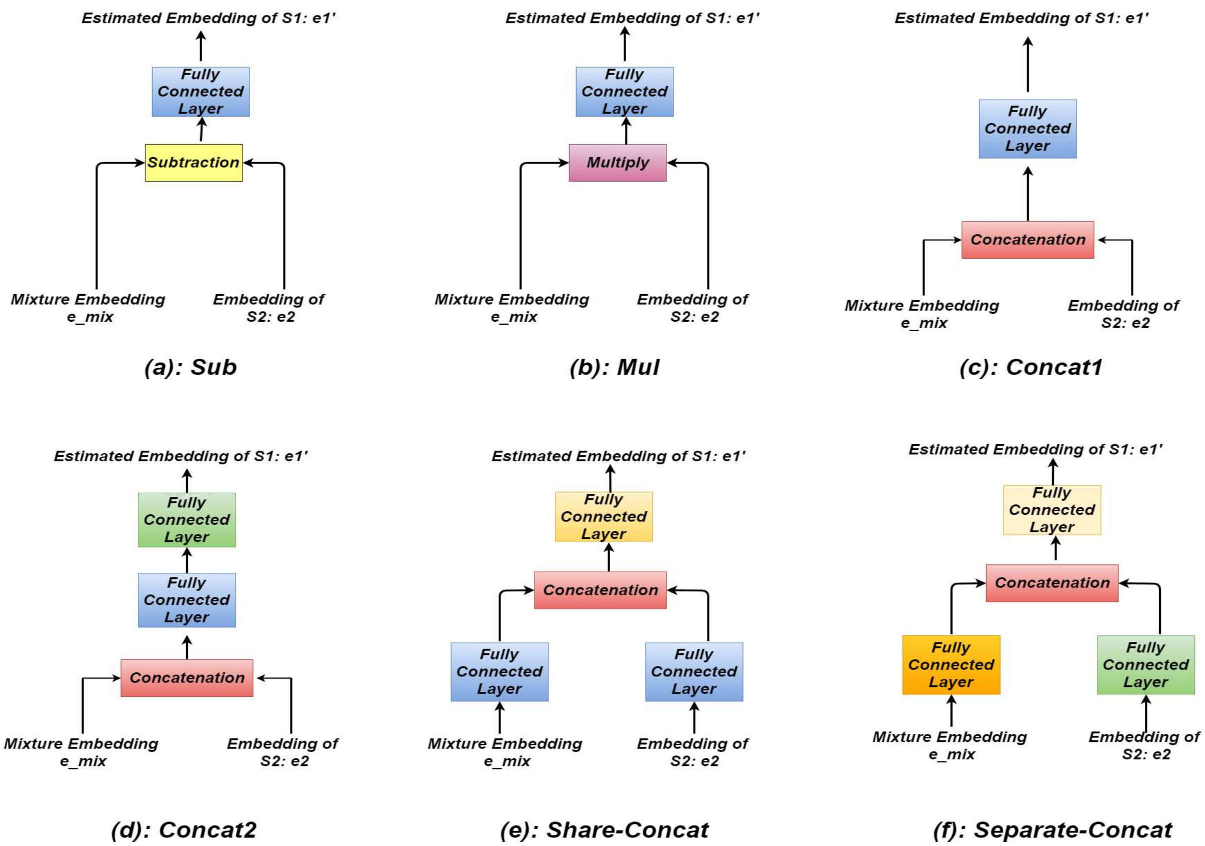

Fig. 3. Different architecture of de-mixing function $\boldsymbol{f}$: (a) Subtraction; (b) Multiplication; (c) Concatenation with one fully-connected layer (d) Concatenate with two fully-connected layers; (e) Shared Fully-Connected Layer with Concatenation and (f) Separated Fully-Connected Layer with Concatenation.

Layer	Context	Output
TDNN Layer1	$[t-1, t, t+1]$	512
TDNN Layer2	[t]	512
TDNN-Res1	$[t-2, t-1, t, t+1t+2]$ [t]	512
TDNN-Res2	$[t-2, t-1, t, t+1, t+2]$ [t]	512
TDNN-Res3	$[t-2, t-1, t, t+1, t+2]$ [t]	512
TDNN Layer3	[t]	1500
Statistics Pooling	T	3000
Segment-Level	T T	512 512

Table 1. Architecture of the speaker embedding network $\boldsymbol{A}$

In step one, both of the high quality embeddings of $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$ are learned and obtained, which are denoted as $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$. Given the input mixture data, the speaker de-mixing network $\boldsymbol{C}$ firstly transforms it in embedding space, results in mixture embedding $\boldsymbol{e}_{mix}$. Then, a de-mixing function $\boldsymbol{f}$ is learned to remove the information of the speakers and remains the other.

More specifically, Figure 2 illustrates the architecture of de-mixing network $\boldsymbol{C}$. The input mixture data contains $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$. $\boldsymbol{C}$ contains two-parts: the first part contains the pre-trained speaker embedding extractor in step one, the goal is to project the input data in embedding space. The output of the pre-trained speaker embedding extractor is $\boldsymbol{e}_{mix}$, which consists of the mixture embedding of two speakers: $\boldsymbol{e}_1 + \boldsymbol{e}_2$. Then, $\boldsymbol{e}_{mix}$ and the clean embedding $\boldsymbol{e}2$ (trained and collected from step one) are input to a de-mixing function $\boldsymbol{f}$ (shows in Equation 1). The output is estimated embedding of the other speaker $\boldsymbol{e}1^{'}$.

$$\boldsymbol{e}_1^{'} = \boldsymbol{f}(\boldsymbol{e}_{mix}, \boldsymbol{e}_2) \tag{1}$$

A reconstruction loss $\mathcal{L}$ (shows in Equation 2) is applied between $\boldsymbol{e}_1^{'}$ and $\boldsymbol{e}_1$. In this work, mean absolute error [28] is applied.

$$\mathcal{L} = ||\boldsymbol{e}_1 - \boldsymbol{e}_1^{'}|| \tag{2}$$

2.3. The architecture of the de-mixing function $\boldsymbol{f}$

The de-mixing function $\boldsymbol{f}$ might have different choices. In this work, six possible methods are investigated. Figure 3 illustrates the six different methods of $\boldsymbol{f}$: (a) Subtraction; (b) Multiplication; (c) Concatenation with one fully-connected layer (d) Concatenate with two fully-connected layers; (e)

Shared Fully-Connected Layer with Concatenation and (f) Separated Fully-Connected Layer with Concatenation.

2.3.1. Subtraction

The first one is a subtraction operation of $\boldsymbol{e}_{mix}$ and $\boldsymbol{e}_2$ (shows is Equation 3 and Figure 3 (a)). After subtraction, the subtracted embedding vector is passed through a fully-connected layer without activation function (could be viewed as a linear transformation). This method is further referred to "Sub". The embedding dimension is denoted as d. $\boldsymbol{W} \in \mathcal{R}^{d\times d}$ and $\boldsymbol{b} \in \mathcal{R}^{1\times d}$ are the parameters of the fully-connected layer.

$$\boldsymbol{f}(\boldsymbol{e}_{mix}, \boldsymbol{e}_2) = (\boldsymbol{e}_{mix} - \boldsymbol{e}_2)\boldsymbol{W} + \boldsymbol{b} \tag{3}$$

2.3.2. Multiplication

Multiplication approach (further referred to "Mul") is similar with "Sub" method. The only difference is $\boldsymbol{e}_{mix}$ is multiplied with $\boldsymbol{e}_2$ instead of subtracted. Figure 3 (b) and Equation 4 shows the architecture of "Mul" method. $\odot$ denotes element-wise multiplication.

$$\boldsymbol{f}(\boldsymbol{e}_{mix}, \boldsymbol{e}_2) = (\boldsymbol{e}_{mix} \odot \boldsymbol{e}_2)\boldsymbol{W} + \boldsymbol{b} \tag{4}$$

2.3.3. Concatenate with one fully-connected layer

In the third method, $\boldsymbol{e}_{mix}$ and $\boldsymbol{e}_2$ are firstly concatenated together, and then feeded into a fully connected layer (shows in Equation 5 and Figure 3 (c)). $[\boldsymbol{e}_{mix}, \boldsymbol{e}_2]^T \in \mathcal{R}^{1\times 2d}$ denotes the concatenated vector of $\boldsymbol{e}_{mix}$ and $\boldsymbol{e}_2$. This method is further referred to "Concat1". $\boldsymbol{W} \in \mathcal{R}^{2d\times d}$ and $\boldsymbol{b} \in \mathcal{R}^{1\times d}$ are parameters for the fully connected layer, $\times$ denotes matrix multiplication.

$$\boldsymbol{f}(\boldsymbol{e_{mix}}, \boldsymbol{e_2}) = [\boldsymbol{e}_{mix}, \boldsymbol{e}_2]^T \times \boldsymbol{W} + \boldsymbol{b} \tag{5}$$

2.3.4. Concatenate with two fully-connected layers

The next method is concatenate with two fully-connected layers. Similar with the previous method, $\boldsymbol{e}_{mix}$ and $\boldsymbol{e}_2$ are firstly concatenated together, and then feed into two fully connected layers instead of one (shows in Equation 6 and Figure 3 (d)). The first fully-connected layer uses $Relu$ activation function while there are no activation function after the second layer.

This method is further referred to "Concat2". $\boldsymbol{W} \in \mathcal{R}^{2d\times d}$ and $\boldsymbol{b} \in \mathcal{R}^{1\times d}$ are parameters for the fully connected layer.

$$\boldsymbol{f}(\boldsymbol{e_{mix}}, \boldsymbol{e_2}) = \mathrm{Relu}(([\boldsymbol{e}_{mix}, \boldsymbol{e}_2]^T \times \boldsymbol{W}_0 + \boldsymbol{b}_0)\boldsymbol{W}_1) \tag{6}$$

2.3.5. Shared Fully-Connected Layer with Concatenation

The last two methods are different from the above methods. In the fifth method, $\boldsymbol{e}_{mix}$ and $\boldsymbol{e}_2$ are firstly input to two fully connected layers respectively, the two fully connected layer share parameters. The output $\boldsymbol{k}_{mix}$ and $\boldsymbol{k}_2$ are then concatenated and feed into another fully connected layer (shows in Equation 7 and Figure 3 (e)). This method is further referred to "Share-Concat". $\boldsymbol{W}_0 \in \mathcal{R}^{d\times d}$, $\boldsymbol{b}_0 \in \mathcal{R}^{1\times d}$, $\boldsymbol{W}_1 \in \mathcal{R}^{2d\times d}$ and $\boldsymbol{b}_1 \in \mathcal{R}^{1\times d}$ are parameters for the fully connected layers.

$$\begin{aligned} \boldsymbol{f}(\boldsymbol{e}_{mix}, \boldsymbol{e}_2) &= \mathrm{Relu}([\boldsymbol{k}_{mix}, \boldsymbol{k}_2]^T \boldsymbol{W}_1 + \boldsymbol{b}_1) \\ \boldsymbol{k}_{mix} &= \mathrm{Relu}(\boldsymbol{e}_{mix}\boldsymbol{W}_0 + \boldsymbol{b}_0) \\ \boldsymbol{k}_2 &= \mathrm{Relu}(\boldsymbol{e}_2\boldsymbol{W}_0 + \boldsymbol{b}_0) \end{aligned} \tag{7}$$

2.3.6. Separated Fully-Connected Layer with Concatenation

The last one is similar with "Share-Concat" method. $\boldsymbol{e}_{mix}$ and $\boldsymbol{e}_2$ are firstly input to two fully connected layers respectively, the two fully connected layers are separated, which means they do not share parameters. The output $\boldsymbol{k}_{mix}$ and $\boldsymbol{k}_2$ are then concatenated and input to another fully connected layer (shows in Equation 8 and Figure 3 (f)). This method is further referred to "Separate-Concat". $\boldsymbol{W}_{0,0} \in \mathcal{R}^{d\times d}$, $\boldsymbol{b}_{0,0} \in \mathcal{R}^{1\times d}$, $\boldsymbol{W}_{0,1} \in \mathcal{R}^{d\times d}$, $\boldsymbol{b}_{0,1} \in \mathcal{R}^{1\times d}$, $W_1 \in \mathcal{R}^{2d\times d}$ and $\boldsymbol{b}_1 \in \mathcal{R}^{1\times d}$ are parameters of the fully connected layers.

$$\begin{aligned} \boldsymbol{f}(\boldsymbol{e}_{mix}, \boldsymbol{e}_2) &= \mathrm{Relu}([\boldsymbol{k}_{mix}, \boldsymbol{k}_2]^T \boldsymbol{W}_2 + \boldsymbol{b}_2) \\ \boldsymbol{k}_{mix} &= \mathrm{Relu}(\boldsymbol{e}_{mix}\boldsymbol{W}_{0,0} + \boldsymbol{b}_{0,0}) \\ \boldsymbol{k}_2 &= \mathrm{Relu}(\boldsymbol{e}_2\boldsymbol{W}_{0,1} + \boldsymbol{b}_{0,1}) \end{aligned} \tag{8}$$

3. EXPERIMENTS

3.1. Data

In this work, TIMIT corpus [29] is used. The TIMIT corpus of read speech is designed to provide speech data for acoustic-phonetic studies and for the development and evaluation of automatic speech recognition systems. There are a total of 6300 utterances, 10 sentences spoken by each of 630 speakers from 8 major dialect regions of the United States. The train and test set are re-split. Six utterances from each speaker are randomly selected for training and the other two utterances are for testing. Hence there are 3780 utterances in the training set and 1260 utterances in the test set.

In order to evaluate the performance in real world conditions, the multi-channel wall street journal audio visual corpus (MC-WSJ) [30] is also used in this work. MC-WSJ contains a total number of 40 speakers reading WSJ sentences in three scenarios: single speaker stationary: A single speaker reading sentences from six positions in a meeting room; Single speaker moving: a single speaker moving between six positions while reading sentences; Overlapping speakers: two speakers reading sentences from different position. There are no speaker overlap between these three conditions.

In this work, the overlapping speaker audio scenario is used. In the overlap version, there are 9 pairs of speakers containing 10 unique speakers. For each speaker pairs, there are 700 utterances in average. There are three different recording techniques: two microphone arrays, lapel and headset microphones worn on all of the speakers.

For all of the experiments in this work, the 20 dimensional MFCC features are used [26].

	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	86.0	92.9
Separate-Concat	0.78	**0.86**	**0.89**	82.5	**93.0**	**96.9**
Clean	1.0			98.5		

Table 2. The cosine similarity and speaker identification accuracy of using the estimated embedding of target speaker $\boldsymbol{e}_1^{'}$. Before denotes the cosine similarity or speaker identification directly using $\boldsymbol{e}_{mix}$. Clean denotes the cosine similarity or speaker identification using $\boldsymbol{e}_1$ that 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	93.7	87.0	59.5
Separate-Concat	**0.87**	**0.79**	0.70	**97.1**	**93.8**	83.6
Clean	1.0			98.5		

Table 3. The cosine similarity and speaker identification accuracy of using the estimated embedding of target speaker $\boldsymbol{e}_2^{'}$. Before denotes the cosine similarity or speaker identification directly using $\boldsymbol{e}_{mix}$. Clean denotes the cosine similarity or speaker identification using $\boldsymbol{e}_2$ that extracted from clean speech.

3.2. Experiment Setup

For TIMIT experiments, in step one, the speaker embeddings are learned using clean TIMIT training set. After training model $\boldsymbol{A}$, for each speaker, 200 segments are randomly sampled and feeded into $\boldsymbol{A}$. The clean speaker embeddings are the average of the embeddings from each segments belonging to the same speaker. $\boldsymbol{B}$ is trained using the same training data as A.

In step two, as TIMIT data contains clean speech only, in order to generate mixture speech signal, each utterance in TIMIT dataset are randomly mixed with another utterance from the other speaker. More specifically, when generating mixture speech signal, one utterance contains target speaker S_1 is chosen, and an utterance from interfering speaker $\boldsymbol{S}_2$ is randomly chosen. $\boldsymbol{S}_1$ is viewed as the target speaker, and $\boldsymbol{S}_2$ is the interfering speaker. The target speaker and the interfering speaker are mixed with a certain SNR (signal-to-noise ratio). Training data will only be mixed with training data, test data will only be mixed with test data. This is to avoid bias problem, as when training the separation model C, the model will not get access to any utterances from test set.

TIMIT experiment is separated into two parts: the first one is to use $\boldsymbol{e}_2$ to obtain $\boldsymbol{e}_1$, in other words, this experiment using the embedding of the interfering speaker to obtain that of the target speaker. The second one is using $\boldsymbol{e}_1$ to obtain $\boldsymbol{e}_2$, which is using the embedding of target speaker to obtain the embedding of interfering speaker.

For MC-WSJ experiments, in step one, the speaker embeddings are learned using the headset recorded audios from the overlapping speakers scenario. The headset recorded audios are close to the corresponding speaker, as a result, the audios in this kind of recording has the close quality of the clean signal [30]. The same technique is used to generated and collect embedding for each speaker and training of classifier $\boldsymbol{B}$. In step two, the model $\boldsymbol{C}$ is trained and tested on two microphones recorded speech (microphone1 and microphone2). For each speaker pair, 70 utterances are randomly selected as the test utterances. Speaker identification accuracies are computed on this test set.

3.3. Evaluation Metric

In this work, two evaluation metrics are used: speaker identification accuracy and cosine similarity.

The speaker identification accuracy is obtained from the classifier $\boldsymbol{B}$. After training $\boldsymbol{A}$, $\boldsymbol{B}$ is also trained and the parameters are fixed. When the de-mixing network $\boldsymbol{C}$ is trained, the embeddings from the test set are extracted. $\boldsymbol{B}$ is used to obtain the speaker identification accuracy of the test set.

The cosine similarity score [31] is directly computed between the clean embedding (e.g. $\boldsymbol{e}_1$) and the de-mixed embedding (e.g. $\boldsymbol{e}_1^{'}$). The final cosine similarity score is computed as the average of the cosine similarity scores for each sample. There is no post-processing techniques used such as PLDA [32], as any post-processing technique used might influence the performance the evaluation process.

3.4. Implementation

In this work, the dimension of all of the fully connected layers is set to 512. Each layer is followed by a batch normalisation layer [33] except for the embedding layer. ReLU activation

	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	

Table 4. The cosine similarity and speaker identification results on MC-WSJ dataset.

[34] is used for each layer except for the embedding layer. The Adam optimiser [35] 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}

4. RESULTS AND DISCUSSION

Table 2 shows the results of using $\boldsymbol{e}_2$ to obtain $\boldsymbol{e}_1$. In Table 2, the cosine similarity and speaker identification results of all of the six speaker de-mixing functions $\boldsymbol{f}$ in different SNR levels are shown.

Comparing without using $\boldsymbol{f}$ (directly evaluate on mixture embeddings $\boldsymbol{e}_{mix}$), most of the architectures of $\boldsymbol{f}$ obtained better performance. This shows that the speaker de-mixing process removed some of the influences of the information from the interfering speakers. The "Separate-Concat" method obtained the best performance when SNR at 0dB and 5 dB, which is close to the results of clean speech. Even the SNR is -5 dB (the power of the interfering speaker $\boldsymbol{S}_2$ is larger than the target speaker ($\boldsymbol{S}_1$), the "Separate-Concat" method can still reach 82.5% test accuracy and 0.78 cosine similarity.

The reason of why "Separate-Concat" method worked better in most of the cases might be the inputs are in different embedding spaces. $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$ are pre-trained and collected, and they contain the properties of a single speaker. But $\boldsymbol{e}_{mix}$ contains the properties of two overlapped speakers, so it contains more complex patterns. "Separate-Concat" method firstly used different fully-connected layers to transform them into another embedding space, and then concatenated them. This operation might make the model to better separate different speaker properties.

"Sub" method obtained best performance when the SNR is -5 dB. "Sub" method, reaching 86.2% in speaker identification and 0.80 cosine similarity score when the SNR is -5 dB. This shows that a simple mathematical operation and a linear transformation can be applied on the speaker embeddings to filter out some information of the interfering speaker. "Mul" method uses another mathematical operation (multiplication), and the performance obtained are still close to the that of clean speech.

The "Concat1", "Concat2" and "Share-Concat" methods obtained lower results. The reason why the "Concat1" and "Concat2" obtained lower performances might be because directly concatenating $\boldsymbol{e}_{mix}$ and $\boldsymbol{e}_2$ might influence the model C to distinguish different speaker properties. The low performance of "Share-Concat" might have the same reason.

Table 3 shows the results of using $\boldsymbol{e}_1$ and $\boldsymbol{e}_{mix}$ to obtain $\boldsymbol{e}_2$, which is using the embedding of the target speaker to obtain the embedding of the interfering speaker. Note the SNR value is the signal-to-noise ratio of the target speaker (S1) and interference speaker (S2). So the results when SNR is 5dB is lower than the results when SNR is -5dB in Table 3. All of the results of six methods shows lower but close performance of that of using $\boldsymbol{e}_2$ to reconstruct $\boldsymbol{e}_1$. It shows that the "Share-Concat" and Sub methods also have the ability to obtain high quality embedding of the interfering speaker from two-speaker environment.

Table 4 shows the experiments result of microphone1 (M1) and microphone2 (M2) in MC-WSJ dataset. The "Share-Concat" method obtain the best results, reaching 93.9% and 90.9% test accuracies and 0.83 and 0.80 socine similarities in M1 and M2. The reason why the results of M2 is lower than that of M1 might be the distance of the speakers and microphones. The M1 is closer to speakers while M2 is far from speakers [30].

Comparing with the results of headset recording, which reaches 99.1% test accuracy, the results obtained by the "Separate-Concat" method still have a gap. The reason might be in real world conditions, the two speakers are moving, the SNR between the target speaker and interfering speakers might be different at different time. It might be more difficult for the model to de-mix the embedding of two speakers.

5. CONCLUSION AND FUTURE WORK

In conclusion, in this work, a speaker embedding de-mixing approach is proposed. The proposed approach reconstructs the embedding of target speaker from the embedding of interfering speaker and mixture embedding, or inversely, obtain the embedding of interfering speaker from that of target speaker and mixture embedding. The quality of embeddings are evaluated by speaker identification accuracy and cosine similarity score on the reconstructed embeddings and the clean embeddings. Results on TIMIT (artificially augmented two-speaker signal) and MC-WSJ (real world two-speaker signal) datasets show that within the six different demixing architectures, the "Share-Concat" method obtain better results, which is close to the results of clean speech.

In this future work, more speaker mixture scenarios will be investigated, such as three-speaker mixture. Different model architectures might be investigated, and larger dataset might be used such as voxceleb1 and 2.

6. ACKNOWLEDGEMENTS

Funding for this research was provided by Huawei Innovation Research Program (HIRP).

7. REFERENCES

[1] AS Bregman, "Auditory scene analysis: The perceptual organization of sound. cambridge, ma, us," 1990.

[2] Kevin W Wilson, Bhiksha Raj, Paris Smaragdis, and Ajay Divakaran, "Speech denoising using nonnegative matrix factorization with priors," in *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2008, pp. 4029–4032.

[3] Bhiksha Raj, Tuomas Virtanen, Sourish Chaudhuri, and Rita Singh, "Non-negative matrix factorization based compensation of music for automatic speech recognition," in *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[4] Hiroshi Saruwatari, Satoshi Kurita, Kazuya Takeda, Fumitada Itakura, Tsuyoki Nishikawa, and Kiyohiro Shikano, "Blind source separation combining independent component analysis and beamforming," *EURASIP Journal on Advances in Signal Processing*, vol. 2003, no. 11, pp. 569270, 2003.

[5] Michael A Casey and Alex Westner, "Separation of mixed audio sources by independent subspace analysis.," in *ICMC*, 2000, pp. 154–161.

[6] Jen-Tzung Chien and Bo-Cheng Chen, "A new independent component analysis for speech recognition and separation," *IEEE transactions on audio, speech, and language processing*, vol. 14, no. 4, pp. 1245–1254, 2006.

[7] Soo-Young Lee, "Blind source separation and independent component analysis: A review," *Neural Information Processing-Letters and Reviews*, vol. 6, no. 1, pp. 1–57, 2005.

[8] Shoji Makino, Hiroshi Sawada, and Shoko Araki, "Frequency-domain blind source separation," in *Blind Speech Separation*, pp. 47–78. Springer, 2007.

[9] Po-Sen Huang, Minje Kim, Mark Hasegawa-Johnson, and Paris Smaragdis, "Deep learning for monaural speech separation," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 1562–1566.

[10] Jun Du, Yanhui Tu, Yong Xu, Lirong Dai, and Chin-Hui Lee, "Speech separation of a target speaker based on deep neural networks," in *2014 12th International Conference on Signal Processing (ICSP)*. IEEE, 2014, pp. 473–477.

[11] Yanhui Tu, Jun Du, Yong Xu, Lirong Dai, and Chin-Hui Lee, "Speech separation based on improved deep neural networks with dual outputs of speech features for both target and interfering speakers," in *The 9th International Symposium on Chinese Spoken Language Processing*. IEEE, 2014, pp. 250–254.

[12] Jun Du, Yanhui Tu, Li-Rong Dai, and Chin-Hui Lee, "A regression approach to single-channel speech separation via high-resolution deep neural networks," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 8, pp. 1424–1437, 2016.

[13] John R Hershey, Zhuo Chen, Jonathan Le Roux, and Shinji Watanabe, "Deep clustering: Discriminative embeddings for segmentation and separation," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 31–35.

[14] Zhuo Chen, Yi Luo, and Nima Mesgarani, "Deep attractor network for single-microphone speaker separation," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 246–250.

[15] Yi Luo, Zhuo Chen, and Nima Mesgarani, "Speaker-independent speech separation with deep attractor network," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 4, pp. 787–796, 2018.

[16] Yu-An Chung and James Glass, "Speech2vec: A sequence-to-sequence framework for learning word embeddings from speech," *arXiv preprint arXiv:1803.08976*, 2018.

[17] David Snyder, Daniel Garcia-Romero, Gregory Sell, Alan McCree, Daniel Povey, and Sanjeev Khudanpur, "Speaker recognition for multi-speaker conversations using x-vectors," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 5796–5800.

[18] Herman Kamper, Weiran Wang, and Karen Livescu, "Deep convolutional acoustic word embeddings using word-pair side information," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 4950–4954.

[19] David Snyder, Pegah Ghahremani, Daniel Povey, Daniel Garcia-Romero, Yishay Carmiel, and Sanjeev Khudanpur, "Deep neural network-based speaker embeddings for end-to-end speaker verification," in *2016 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2016, pp. 165–170.

[20] Koji Okabe, Takafumi Koshinaka, and Koichi Shinoda, "Attentive statistics pooling for deep speaker embedding," *arXiv:1803.10963*, 2018.

[21] Junzo Watada et al., "Speech recognition in a multi-speaker environment by using hidden markov model and mel-frequency approach," in *2016 Third International Conference on Computing Measurement Control and Sensor Network (CMCSN)*. IEEE, 2016, pp. 80–83.

[22] Shruti Palaskar, Vikas Raunak, and Florian Metze, "Learned in speech recognition: Contextual acoustic word embeddings," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 6530–6534.

[23] Joon Son Chung, Arsha Nagrani, and Andrew Zisserman, "Voxceleb2: Deep speaker recognition," *arXiv preprint arXiv:1806.05622*, 2018.

[24] Weidi Xie, Arsha Nagrani, Joon Son Chung, and Andrew Zisserman, "Utterance-level aggregation for speaker recognition in the wild," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 5791–5795.

[25] Pavel Denisov and Ngoc Thang Vu, "End-to-end multi-speaker speech recognition using speaker embeddings and transfer learning," *arXiv preprint arXiv:1908.04737*, 2019.

[26] David Snyder, Daniel Garcia-Romero, Gregory Sell, Daniel Povey, and Sanjeev Khudanpur, "X-vectors: Robust dnn embeddings for speaker recognition," in *ICASSP*. IEEE, 2018.

[27] Hossein Zeinali, Shuai Wang, Anna Silnova, Pavel Matějka, and Oldřich Plchot, "But system description to voxceleb speaker recognition challenge 2019," *arXiv preprint arXiv:1910.12592*, 2019.

[28] Cort J Willmott and Kenji Matsuura, "Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance," *Climate research*, vol. 30, no. 1, pp. 79–82, 2005.

[29] John S Garofolo, Lori F Lamel, William M Fisher, Jonathan G Fiscus, and David S Pallett, "Darpa timit acoustic-phonetic continous speech corpus cd-rom. nist speech disc 1-1.1," *NASA STI/Recon technical report n*, 1993.

[30] Mike Lincoln, Iain McCowan, Jithendra Vepa, and Hari Krishna Maganti, "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, 2005.* IEEE, 2005, pp. 357–362.

[31] Hieu V Nguyen and Li Bai, "Cosine similarity metric learning for face verification," in *Asian conference on computer vision*. Springer, 2010, pp. 709–720.

[32] Patrick Kenny, Themos Stafylakis, Pierre Ouellet, Md Jahangir Alam, and Pierre Dumouchel, "Plda for speaker verification with utterances of arbitrary duration," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013, pp. 7649–7653.

[33] Sergey Ioffe and Christian Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv:1502.03167*, 2015.

[34] Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus, "Regularization of neural networks using dropconnect," in *International conference on machine learning*, 2013, pp. 1058–1066.

[35] Diederik P Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *arXiv:1412.6980*, 2014.