

FRAME-LEVEL SPEAKER EMBEDDINGS FOR TEXT-INDEPENDENT SPEAKER RECOGNITION AND ANALYSIS OF END-TO-END MODEL

Suwon Shon, Hao Tang, James Glass

Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02139 USA

{swshon, haotang, glass}@mit.edu

ABSTRACT

In this paper, we propose a Convolutional Neural Network (CNN) based speaker recognition model for extracting robust speaker embeddings. The embedding can be extracted efficiently with linear activation in the embedding layer. To understand how the speaker recognition model operates with text-independent input, we modify the structure to extract frame-level speaker embeddings from each hidden layer. We feed utterances from the TIMIT dataset to the trained network and use several proxy tasks to study the networks ability to represent speech input and differentiate voice identity. We found that the networks are better at discriminating broad phonetic classes than individual phonemes. In particular, frame-level embeddings that belong to the same phonetic classes are similar (based on cosine distance) for the same speaker. The frame level representation also allows us to analyze the networks at the frame level, and has the potential for other analyses to improve speaker recognition.

Index Terms— speaker recognition, embedding, frame-level representation, text-independent

1. INTRODUCTION

Deep neural networks (DNNs) have been actively used in speaker recognition to discriminate speakers' identity. In most settings, DNNs are used as a replacement for Gaussian mixture models (GMMs) to improve the conventional i-vector approach [1] by having a more phonetically aware Universal Background Model (UBM) [2, 3, 4]. Other subsequent method based on DNN were introduced for noise-robust and domain-invariant i-vector [5, 6, 7] However, the process of training the GMM-UBM and extracting i-vectors largely remained the same.

More recently, many studies have begun to explore end-to-end DNN speaker recognition to extract robust speaker embeddings using large datasets as well as data augmentation [8, 9]. These end-to-end models directly operate on spectrograms, log Mel features, or even waveforms [10, 11]. Among the end-to-end approaches, x-vectors have been the

most effective for text-independent scenarios [8]. Compared to i-vectors and bottleneck feature-based i-vectors, x-vectors have achieved better results by taking advantage of data augmentation with noise and reverberation. Due to neural networks large learning capacity, data augmentation has been shown to be a cheap and effective approach to improve performance and robustness. The gap between x-vectors and i-vectors is expected to widen as the amount of data increases and end-to-end networks continue to be improved.

The i-vector approach is based on the assumption that each individual mean vector in a GMM is a shift from a mean vector of the UBM, and that the shifts of all the means are controlled by a single vector, the i-vector. The model has been studied extensively and is well understood [1]. In contrast, it is difficult to understand why and how speaker embedding networks work, which hinders the development of better end-to-end speaker recognition models.

In this paper, we introduce a speaker embedding extracted from a 1-dimensional convolution and linear activation from an end-to-end model. The use of linear activation is inspired by previous studies [12, 13], where reducing non-linearities has been shown to improve performance. The embeddings are compared to two strong baselines, x-vectors and an approach based on the VGG network. We then analyze the networks behavior by modifying the network structure and extracting frame-level representations from the hidden layers. We feed utterances from the TIMIT dataset into the model and monitor the behavior of the representations at different training epochs. We hypothesize that the networks' ability to recognize speakers is based on how the phonemes are pronounced and that the networks pay more attention to certain phonemes or broad class than others. For text-independent input, since it is unlikely that the same set of phonemes appeared in both the enrollment and test utterances, we believe the speakers' identity is less likely to be decided at the phonetic level but more likely at a higher level based on the phonetic classes. Identifying speakers at the broad-class level allows the networks to assess the speaker's voice even without the presence of the exact same phoneme.

Table 1: Results on the Voxceleb1 test set. ReLU is applied after every layer.

	EER	DCF _{p=0.01}	DCF _{p=0.001}
fc1 (LDA+PLDA)	6.2	0.53	0.70
fc2 (LDA+PLDA)	6.9	0.55	0.65

To verify this hypotheses, we conduct phoneme recognition and broad-class classification tasks using frame-level representation of speaker embeddings, and then we visualize and analyze the frame-level cosine similarity measurements from the same and different speaker pairs. From these proxy tasks, we examine how phonetic information is encoded in the network. We also investigate which phoneme or broad-classes are more important for text-independent speaker recognition using frame-level speaker embeddings extracted from TIMIT data.

2. SPEAKER EMBEDDINGS WITH LINEAR ACTIVATION

Previous work has shown that the layer immediately following the statistics pooling layer performs well in combination with latent discriminant analysis (LDA) and probabilistic LDA (PLDA) as backends [8]. It is perhaps not surprising that the layer closest to the output layer contains the most discriminative information about the output labels. We follow a similar approach to analyze our end-to-end speaker recognition model.

Our network structure is similar to the VGG network but we use 1-dimensional convolutions to consider all frequency bands at once. The network structure consists of 4 1d-CNN (i.e., filters of size 40×5 , 1000×7 , 1000×1 , 1000×1 with strides 1, 2, 1, and 1 and numbers of filters 1000, 1000, 1000, and 1500) with two fully connected (FC) layers (of size 1500 and 600) as shown in Figure 1. We use statistics pooling as in [8]. We use a 1-d CNN instead of a 2-d CNN commonly used for computer vision, because, unlike images, a spectrogram carries different meanings in each axis, namely time versus frequency. A person’s voice can be shifted in time but it cannot be shifted in frequency. For this reason, the width of the 1-d CNN is the entire frequency axis.

We use 40-dimensional Mel-Frequency Cepstral Coefficients (MFCCs) with the standard 25ms window size and 10ms shift to represent the speech signal. The features are normalized to have zero mean. We use the Voxceleb1 development dataset, including 1,211 speakers and 147,935 utterances, to train the networks. The Voxceleb1 test set has 18,860 verification pairs for each positive and negative test, i.e., a combination of 4,715 utterances from 40 speakers not included in the development set. The networks are trained from random initialization. The SGD learning rate is 0.001, and is decayed by a factor of 0.98 after every 50,000 updates.

Our first experiments use ReLU nonlinearities after each

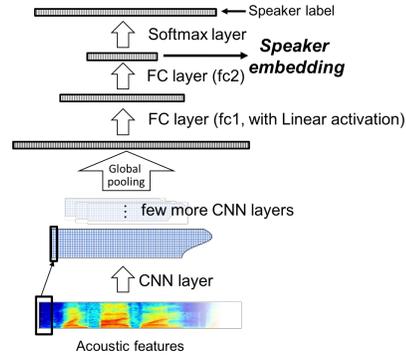


Fig. 1: DNN structure to extract speaker embeddings

Table 2: Results on the Voxceleb1 test set. ReLU is applied after every layer except after fc1.

	EER	DCF _{p=0.01}	DCF _{p=0.001}
fc1 (LDA+PLDA)	6.2	0.51	0.69
fc2 (LDA+PLDA)	5.9	0.50	0.62

layer in the network. Results are shown in Table 1. We find similar results as in [3] that the vectors from the first fully connected layer (fc1) have better performance than those from the second fully connected layer (fc2).

We remove the nonlinear activation function before the last hidden layer (fc2), and the results are shown in Table 2. From the result, we observe better performance when removing the activation function, and this observation has also been made in previous studies [13, 12]. We subsequently use the vectors from the fc2 layer as speaker embeddings.

The differences between recent speaker embedding approaches are summarized in Table 3. Using the same setting as in Table 2, we compare the speaker embeddings with i-vectors, x-vectors, and the approach based on the VGG network. Results are shown in Table 4. We augment the dataset as in [8] with reverberation and different noise types, such as babble noise and background music. The number of utterances is 147,935 and we augment an additional 140,000 utterances. Without data augmentation, the proposed speaker embedding method is slightly worse than i-vectors but significantly outperforms the VGG approach and x-vectors. We find that the i-vectors in [9] are worse because they use a small number of Gaussian components for the GMM-UBM. After using 2,048 components, the i-vectors perform the best. With data augmentation, the EER improves by 15% for x-vectors but is still worse than i-vectors. Our embeddings also benefit from data augmentation and is able to match the i-vector results.

Table 3: Three recent speaker embedding approaches.

	x-vector [3]	VGG [9]	Ours
Input for training	MFCC	Spectrogram with fixed length(3sec)	MFCC with fixed length (2sec)
Input normalization	CMN	CMVN	CMN
Structure	TDNN	2d-CNN (VGG-M)	1d-CNN
Parameters	4.4m	64m	13m
Global Pooling	Statistics	Average	Statistics
Embedding layer	First fully connected layer	Last fully connected layer	Last fully connected layer
Nonlinearity	All layers	All layers	All layers except before embedding layer
Embedding Dimension	512	1024	600
Backend Processing	Zero-mean norm.+LDA +length norm.+PLDA	Euclidean Distance with Siamese network	Zero-mean norm.+LDA +length norm.+PLDA

Table 4: Results on the Voxceleb1 test set. Systems trained with data augmentation are labeled with *.

	EER	DCF _{p=0.01}	DCF _{p=0.001}
i-vector	5.4	0.45	0.63
i-vector*	5.5	0.48	0.61
VGG [9]	7.8	0.71	-
x-vector (Cosine)	11.3	0.75	0.81
x-vector (PLDA)	7.1	0.57	0.75
x-vector* (Cosine)	9.9	0.69	0.85
x-vector* (PLDA)	6.0	0.53	0.75
fc2 (Cosine)	7.3	0.56	0.64
fc2 (PLDA)	5.9	0.50	0.62
fc2* (Cosine)	7.0	0.58	0.68
fc2* (PLDA)	5.3	0.45	0.63

3. FRAME-LEVEL REPRESENTATION OF SPEAKER EMBEDDING AND ITS ANALYSIS

Conventional speaker recognition approaches, such as i-vectors, require many steps that are carefully designed for learning a robust representation of speaker identity from acoustic features. Representations learned from deep networks, however, are optimized for this particular task purely from data [14]. The output vectors produced by the intermediate layers, the hidden representation, could be the key to understand what end-to-end speaker recognition model implicitly learns from voice inputs. We adopt the approach used in [15, 16] where several proxy tasks are used to analyze the hidden representations. In speaker recognition, several studies have assessed which phoneme contributes the most to discriminating speakers [17], and which phonetic classes are more important than others [18]. Both papers conclude that vowels and nasals provide the most useful information for identifying speakers. However, the experiments are limited to a single phoneme or a single phonetic class, so it is difficult to draw similar conclusions when the networks can make use of an entire utterance. Wang et al. [19] analyze speaker embeddings in a text-dependent speaker recognition system, and use an approach similar to [15]. However, text-dependent speaker

recognition is easier to analyze than the text-independent case because the enrollment and test utterances always have the same phoneme statistics. We aim to understand how the embedding representation encodes phoneme information at the frame level when text-independent input is given.

In this section, we analyze what phonetic information is encoded in the end-to-end speaker recognition model and how they capture and discriminate between talkers given text-independent input. We use phoneme recognition and phonetic classification as proxy tasks and monitor the behavior over the course of training to answer these questions. We also identify phonemes that are critical for text-independent speaker verification. For the proxy tasks, we assume that the ability of a classify to predict a certain property depends on how well the property has been encoded in the representation (as in previous studies [15, 16, 19]). Poor accuracy does not necessarily mean the information is not present however.

3.1. Frame-level speaker embeddings

To obtain frame-level representations, we need to modify the structure of our models. For training, we substitute the statistics pooling layer to be an average pooling layer. After the modification, the EER increased from 7.0% to 8.4% using cosine similarity, and from 5.3% to 6.0% with PLDA. After training, we moved the average pooling layer to be after the fc2 layer but just before ReLU activation function. This modification does not change the final results, because multiplying the average with a matrix is the same as the average of the individual vectors multiplied by the matrix. The output vectors produced by all layers before the average pooling yields frame-level representations not only for CNN layers but also for FC layers. Specifically, suppose u is a segment-level representation (i.e. a speaker embedding extracted at the fc2 layer in Figure 1). The embedding u can be represented as the frame-level representation \bar{u}_t which is extracted from the modified model structure as $u = (1/T) \sum_t \bar{u}_t$ where T is the number of frames in the utterance. We extracted this frame-level representation at all CNN and FC layers for the proxy tasks.

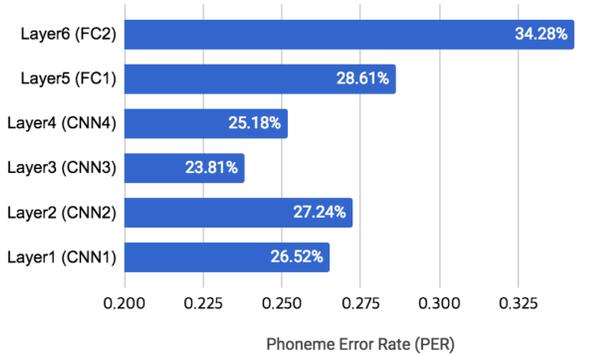


Fig. 2: Phoneme error rates of segmental models on the TIMIT test set using frame-level representations from different layers. Lower is better.

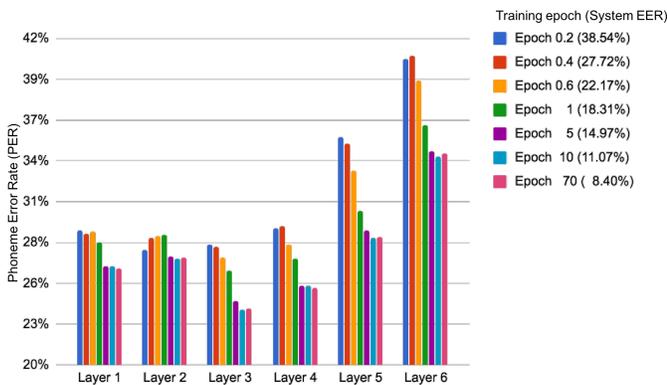


Fig. 3: Phoneme error rates of segmental models on the TIMIT test set using frame-level representation from different layers over the course of training. Lower is better.

3.2. Phoneme Recognition

Given the trained model with the modified structure, we use TIMIT utterances to examine the ability of the speaker recognition system at distinguishing phonemes. The first CNN layer produces an output every 10ms while subsequent layers produce an output every 20ms because of the second layer stride by 2 that reduces the effective analysis rate to 20ms. The stride should not affect the analyses, because previous work has shown that similar, if not better, results can be achieved with one-fourth of the original frame rate [20].

We train a phoneme recognizer with the frame-level representation extracted from each of the layers and analyze their performance. We use an end-to-end discriminative segmental model [20] with a 2-layer LSTM model as our phoneme recognizer. The input to the LSTM consists of frame-level representations from different layers from the end-to-end speaker recognition model. In each layer, the output vectors of the LSTM are sub-sampled by half. The final segment scores are based on the output vectors of the LSTM within the seg-

Table 5: TIMIT broad phonetic classes.

Class	Symbol
Affricate	jh, ch
Closures	bcl,dcl,gcl,pcl,tck,kcl
Fricative	s,sh,z,zh,f,th,v,dh
Nasals	m,n,ng,em,en,eng,nx
Semivowels and Glides	l, r, w, y, hh, hv, el
Vowels	iy, ih, eh, ey, ae, aa, aw, ay, ah, ao, oy, ow, uh, uw, ux, er, ax, ix, axr, ax-h
Stops	b, d, g, p, t, k, dx, q
others	pause(pau), epenthetic silence(epi), start and end silence (h)

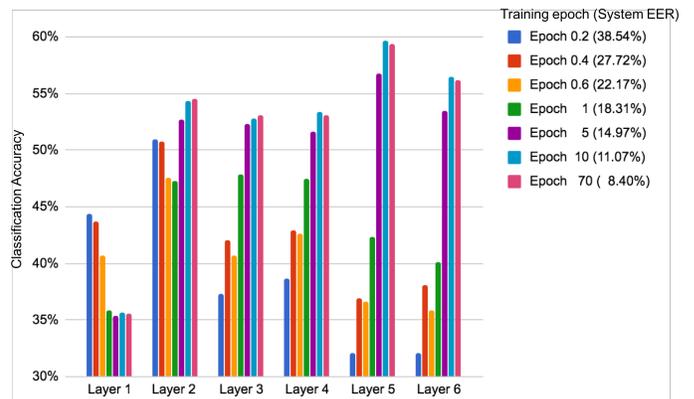


Fig. 4: Phonetic class classification accuracy on TIMIT Test set using features from each layer over the course of training. Higher is better.

ment and the duration of the segments (the FCB feature set in [21]). We allow a segment to have a maximum duration of 120 frames. The segmental models are trained with marginal log loss [22] for 20 epochs with vanilla stochastic gradient descent (SGD), a step size of 0.1. The batch size is one utterance, and the gradients are clipped to norm 5.

Figure 2 shows the phoneme error rate (PER) for different layers, and indicates that the embeddings from higher layers give higher PERs. Judging from the PERs over the course of training, the training error, in general, stops improving after epoch five. At layer 6 for example, the PER plateaus at 34%. From this observation, the frame-level representation contains less information about phonemes, and the phoneme identify does not seem to be important for discriminating speakers.

3.3. Broad-Class Phonetic Classification

For broad-class phonetic classification, we collect all phoneme segments in the TIMIT dataset based on the ground truth segmentation. The broad phonetic classes we use are shown in Table 5. The segment embedding of each segment is computed by averaging the frame-level embedding obtained from the trained speaker recognition system. We create a naive classifier by averaging the embeddings of the same

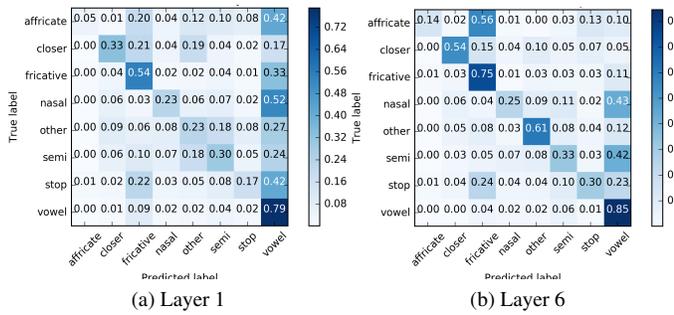


Fig. 5: Confusion matrix for broad-class phonetic classification comparing the features produced by layer 1 and layer 6 at epoch 70.

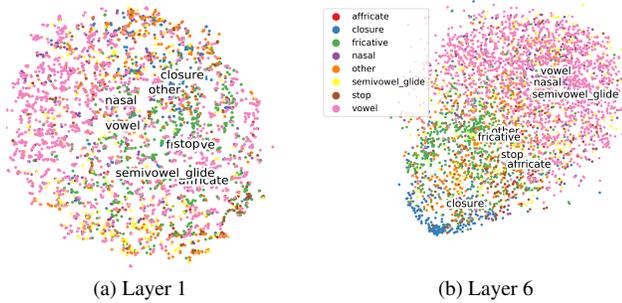
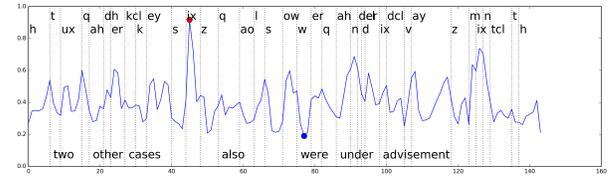


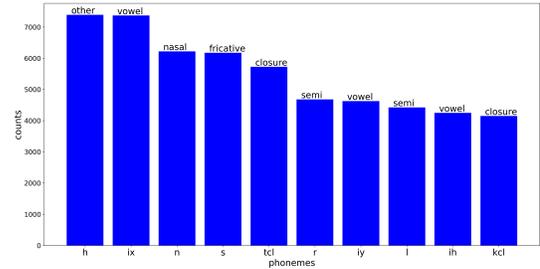
Fig. 6: Low-dimensional t-SNE projection of speaker embeddings from layer 1 and 6 at epoch 70 using utterances from the TIMIT core test set. The broad-class label is printed in the figure.

phonetic classes. Specifically, we compute a vector $U^b = \sum_{i \in P_b} u_i / |P_b|$ for phonetic class b , where P_b contains the segments of class b , and u_i is the segment embedding of segment i by averaging the frame embedding in segment i . Given a new segment j , we compute its segment embedding u_j by averaging the frame embeddings computed from the trained network. Classification can be done with $\text{argmax}_b(\cos(U^b, u_j))$. As shown in Figure 4, the system at the early stage of training does not distinguish the phonetic classes well, and is worst in the higher layer. After training, the model learns to distinguish phonetic classes well. In particular, the representation in the higher layers performs significantly better than the ones in the lower layers.

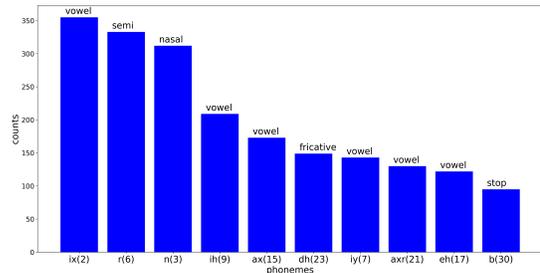
The confusion matrix of the broad phonetic classes is shown in Figure 5. The diagonal of the confusion matrix shows that the accuracy of the embedding from layer 6 performs better than the one from layer 1. An interesting observation is that some categories are still confusable at layer 6. For example, affricates and stops are predicted as fricatives, and nasals and semivowels are predicted as vowels. This suggests that the model might be classifying segments into



(a) Example of frame-level cosine similarity between the same speaker's segment-level and frame-level speaker embedding



(b) Original phoneme occurrence histogram on TIMIT training set



(c) Histogram of highest cosine similarity phonemes in each utterance on TIMIT training set (numbers after words : order from the original histogram)

Fig. 7: Stats in terms of phoneme

even broader categories, such as obstruents and sonorants. This phenomenon is also observed in the t-SNE plots of the phoneme segments shown in Figure 6¹.

3.4. Critical phonemes and phonetic classes

In this section, we analyze cosine similarity at the frame level. In the TIMIT dataset there are 10 utterances per speaker. We calculated cosine similarity for all utterances using frame-level speaker embeddings from the modified structure. For enrollment, for each speaker nine out of ten utterances are used by averaging the segment embeddings to create a single speaker embedding. Figure 7 (a) shows the frame-level cosine similarity with the single speaker embedding. In the example, the phoneme /ix/ shows the highest cosine similarity (blue dot) and the phoneme /w/ shows the lowest (red dot). Figure 7 (b) shows the histogram of phoneme frequency in

¹GIF animation also Available on <https://people.csail.mit.edu/swshon/supplement/slt18.html>

the training set, and (c) shows how often each phonetic class achieves the highest cosine similarity. From the histogram, we made similar observation as previous studies [17, 18] that vowels and nasals are important for discriminating speakers. These observations suggests that using an attention mechanism at the pooling layer may improve the performance of speaker recognition.

Figure 8 shows the frame-level similarity matrix between two utterances from the TIMIT training set. In (a), the two utterances are spoken by the same speaker and (b) by different speakers. For comparison, the same text content is chosen in both figures. From the figures, the same phoneme shows relatively high similarity even for different speakers. However, in the case of (a) which is spoken by the same speaker, the phonemes that share the same phonetic classes show high similarity. For example, /s/ and /er/ show high similarity with /z/ and /aa/, respectively. From this result, we observe that the network could also potentially measure a speaker’s similarity across different phonemes. Interestingly, obstruents, such as fricatives and closures, have relatively high similarity scores even for different speakers.

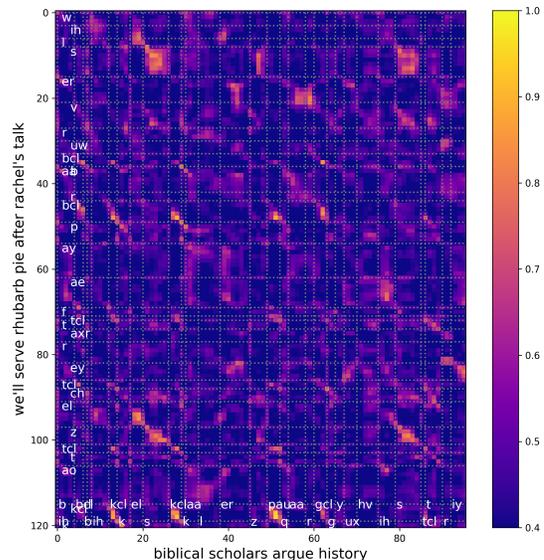
3.5. Discussion

Using both frame-level and segment-level representations, we observe that the representations from higher network layers corresponds to broader classes, while the representation from lower layers correspond to classes that are more specific. Particularly, the representation appears to converge towards obstruent and sonorant categories at higher layers. This behavior suggests that the model computes similarity at a more abstract level than that at the phonetic level. It potentially could provide an advantage when the input is text-independent and short because the model has fewer phonemes to compare with the enrollment data. This is consistent with the results found in the similarity matrix in Figure 8.

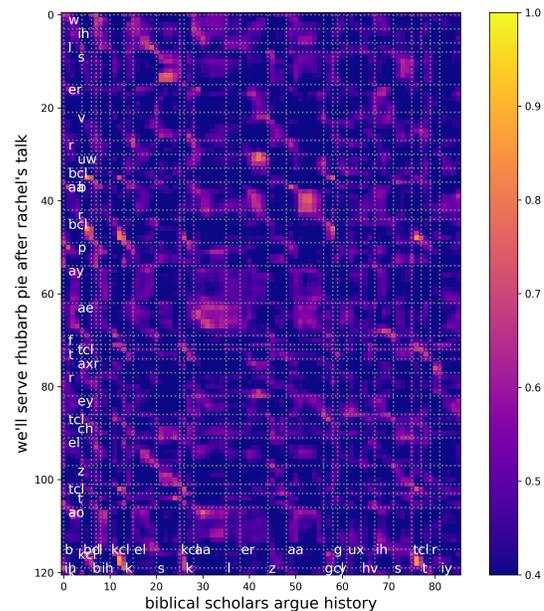
4. CONCLUSION

In this paper, we proposed a robust speaker embedding for speaker verification. Embeddings are extracted without non-linear activation and are compared to other approaches to verify their effectiveness. On the Voxceleb1 dataset with only 1.2k speakers, the proposed approach shows superior performance compared to x-vectors. In this framework, there is still room for improvement, such as exploring a larger speaker dataset, a different loss function, such as the angular Softmax loss, or adding an attention layer. We leave these as future work.

From the analysis, we attempt to better understand how the speaker recognition model extracts discriminative embeddings. The analysis provides some insight on the model behavior, and the frame-level analysis provides an important tool to assess the quality of the trained models.



(a) Same speaker (speaker id 'fame0' in TIMIT)



(b) Different speakers (speaker id 'faem0' for y-axis and 'mrpc1' for x-axis in TIMIT)

Fig. 8: Frame-level cosine similarity matrix between two sentences spoken by the same speaker and by different speakers.

5. REFERENCES

- [1] Najim Dehak, Patrick J Kenny, Reda Dehak, Pierre Dumouchel, and Pierre Ouellet, “Front-End Factor Analysis for Speaker Verification,” *IEEE Trans. on Audio, Speech, and Lang. Process.*, vol. 19, no. 4, pp. 788–798, may 2011.
- [2] Fred Richardson, Senior Member, Douglas Reynolds, and Najim Dehak, “Deep Neural Network Approaches to Speaker and Language Recognition,” *Ieee Signal Processing Letters*, vol. 22, no. 10, pp. 1671–1675, 2015.
- [3] David Snyder, Pegah Ghahremani, Daniel Povey, Daniel Garcia-Romero, and Yishay Carmiel, “Deep Neural Network Embeddings for Text-Independent Speaker Verification,” in *Interspeech*, 2017, pp. 999–1003.
- [4] Yun Lei, Nicolas Scheffer, Luciana Ferrer, and Mitchell McLaren, “A Novel Scheme for Speaker Recognition using a Phonetically-aware Deep Neural Network,” in *IEEE ICASSP*, 2014, pp. 1714–1718.
- [5] Omid Ghahabi and Javier Hernando, “Deep belief networks for i-vector based speaker recognition,” in *IEEE ICASSP*, 2014, number 1, pp. 1700–1704.
- [6] Timur Pekhovsky, Sergey Novoselov, Aleksei Sholohov, and Oleg Kudashov, “On autoencoders in the i-vector space for speaker recognition,” in *Proc. Odyssey 2016 The Speaker and Language Recognition Workshop*, 2016, pp. 217–224.
- [7] Suwon Shon, Seongkyu Mun, Wooil Kim, and Hanseok Ko, “Autoencoder based Domain Adaptation for Speaker Recognition under Insufficient Channel Information,” in *Interspeech*, 2017, pp. 1014–1018.
- [8] David Snyder, Daniel Garcia-Romero, Gregory Sell, Daniel Povey, and Sanjeev Khudanpur, “X-Vectors: Robust Dnn Embeddings for Speaker Recognition,” in *IEEE ICASSP*, 2018.
- [9] Arsha Nagrani, Joon Son Chung, and Andrew Zisserman, “VoxCeleb: A large-scale speaker identification dataset,” in *Interspeech*, 2017, pp. 2616–2620.
- [10] Suwon Shon, Ahmed Ali, and James Glass, “Convolutional neural network and language embeddings for end-to-end dialect recognition,” in *Proc. Odyssey 2018 The Speaker and Language Recognition Workshop*, 2018, pp. 98–104.
- [11] Jee-Weon Jung, Hee-Soo Heo, Il-Ho Yang, Hye-Jin Shim, and Ha-Jin Yu, “A Complete End-to-end Speaker Verification System using Deep Neural Network: from Raw Signals to Verification Result,” in *IEEE ICASSP*, 2018.
- [12] Yu Zhang, Ekapol Chuangsuwanich, and James Glass, “Extracting Deep Neural Network Bottleneck Features Using Low-Rank Matrix Factorization,” in *IEEE ICASSP*, 2014, pp. 185–189.
- [13] Karel Veselý, Martin Karafiát, and František Grézl, “Convolutional bottleneck network features for LVCSR,” in *2011 IEEE Workshop on Automatic Speech Recognition and Understanding, ASRU 2011, Proceedings*, 2011, pp. 42–47.
- [14] Yann Lecun, Yoshua Bengio, and Geoffrey Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
- [15] Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg, “Fine-grained analysis of sentence embeddings using auxiliary prediction tasks,” in *International Conference on Learning Representations (ICLR)*, 2017.
- [16] Yonatan Belinkov and James Glass, “Analyzing Hidden Representations in End-to-End Automatic Speech Recognition Systems,” in *Advances in Neural Information Processing Systems*, 2017, pp. 2438–2448.
- [17] J. P. Eatock and J. S. Mason, “A quantitative assessment of the relative speaker discriminating properties of phonemes,” in *IEEE ICASSP*, 1994, vol. 1, pp. 133–136.
- [18] Margit Antal and Gavril Todorean, “Broad Phonetic Classes Expressing Speaker,” *Knowledge Creation Diffusion Utilization*, vol. LI, no. 1, pp. 49–58, 2006.
- [19] Shuai Wang, Yanmin Qian, and Kai Yu, “What does the speaker embedding encode?,” in *Interspeech*, 2017, vol. 2017-Augus, pp. 1497–1501.
- [20] Hao Tang, Liang Lu, Lingpeng Kong, Kevin Gimpel, Karen Livescu, Chris Dyer, Noah A. Smith, and Steve Renals, “End-to-end neural segmental models for speech recognition,” *IEEE Journal of Selected Topics in Signal Processing*, 2017.
- [21] Hao Tang, *Sequence Prediction with Neural Segmental Models*, Ph.D. thesis, Toyota Technological Institute at Chicago, 2017.
- [22] Hao Tang, Weiran Wang, Kevin Gimpel, and Karen Livescu, “End-to-end training approaches for discriminative segmental models,” in *IEEE Workshop on Spoken Language Technology (SLT)*, 2016.