Text-Independent Voice Speaker Identification
NLP ML Final Project

Polina Turishcheva
Innopolis University / Innopolis, Russia
`p.turischeva@innopolis.university`

Abstract

This is a course project for text-independent speaker verification based on x-vectors. We tried to replicate a DeepSpeaker for Russian speech but faced problems due to a limited dataset and were forces to change DeepSpeaker training paradigm a bit. Our model showed EER = 0.0700 and Accuracy = 0.9451 on a validation dataset.

1 Credits

This work is a replication of Li et. al. (2017). *Deep speaker: an end-to-end neural speaker embedding system* for Russian language. I have also looked through 2 repositories with DeepSpeaker implementation during creating my own.

- *Keras Implementation*
- *PyTorch Implementation*

Please note that I have neither tested if all of these repositories are working nor copypasted the code from them. Also both repositories have mismatches between their implementations and the article they are implementing, nevertheless, looking at some code examples was useful for me.

2 Introduction

There are 2 types of speaker identification: text-dependent and text-independent. For text dependent classification we need several utterances for each person and for all of them there should be same phrase/phrases they pronounce. I have not managed to find big enough Russian dataset meeting those conditions, hence, switched to text-independent speech recognition. The only one open-source dataset for speech-dependent recognition is the one collected by *NN HSE* but it contains only 46 speakers and less than 100 hours of speech, which is not enough to train a model. To compare, in [5] they used a dataset with around 150M utterances from around 630K speakers, and there were only 2 phrases in it.

3 Background

During the course we have studied embeddings. For speakers there 2 most common types of embeddings: i-vectors and d-vectors. I-vectors are usually extracted with Gaussian mixture modeling (GMM) from the training data followed by an adaptation using maximum-aposteriori (MAP) rule. The i-vector framework assumes that the i-vector, or the latent variable has a Gaussian distribution. A typical speaker identification framework with i-vectors ooks like : data preprocessing -¿ i-vectors extraction -¿ PLDA (Probabilistic Linear Discriminant Analysis).
For x-vectors there are no initial assumptions. DNN model that takes stacked filterbank features (Log-Filterbank or MFCC) and generates the one-hot speaker label (or the speaker probability) on the output is trained. D-vector is the averaged activation from the last hidden layer of this DNN. DeepSpeaker is a model to create x-vector embeddings. An article by Snyder et al. [4] shows that x-vectors perform better that i-vectors.

4 Method

4.1 Data Preprocessing

Voice utterances is a non-standard data type yet, hence, there is no common preprocessing pipeline, however, frequency filtering, LogFilterrbank, MFCC (Mel-Frequency Cepstral Coefficients) are the most widespread methods. We will use MFCC as they were used in thee original article but here will be a small introduction about other approaches and argumentation why MFCC is the most rational choice.
LogFilterbank is an idea to miimic the human ear using one-dimensional triangular (with growing period) convolutions, called Mel-scales, as a rule there are 40 of those. The problem of those scales is that a significant part of Mel coefficients are highly correlated. To avoid information redundancy, Discrete Cosine Transform is applied to LogFilterbank coefficients and new compressed coefficients are called MFCC.
As for filtering, the problem is that there fundamental frequencies of human voice are in range of [50, 350] Hz, however, our voice is richer as it contains overtones (soprano singing may reach 11000 Hz), and they may be also useful, and there are many exceptions from the stated range, especially children. Anyway, filtering is classically followed by MFCC and removing a significant part of information will highly influence the MFCC transformations.

We have not applied any data augmentation due to limited amount of time and resources. However, pink noise augmentation or Urban

4.2 Baseline

We used SincNet (*repository used*) as a baseline model. The idea of SincNet is to avoid any spesific preprocessing. The authors say that MFCC extraction may be not optimal for a particular case, hence, a network can learn coefficients for convolution on its own - this is the core idea of SincNet. It showed EER = 0.0365 and Accuracy = 0.9965.

4.3 Datasets

For speaker recognition there are 2 important features for a dataset: the number of unique speakers and the length of the dataset. Since, there are few open-source datasets with Russian speech, we have selected 2 of them - one for training and one for evaluation.

Common Voice Russian (Training Dataset)

- 927 unique speakers
- 105-validated hours(116 total)
- has subtitles
- 73% of records are for male subjects, 18% are female, the rest are undefined
- 3Gb in a compressed representation, 77 utterances
- highly imbalanced (one speaker my have one utterance, while the other one may have a hundred)

VoxForge2 Dataset (Evaluation Dataset)

- 207 unique speakers
- 24.8 hours
- has subtitles
- no gender statistics
- more or less balanced

- for evaluation purposes we used only 3 utterances per speaker

Some well-known datasets like Gutenberg are not suitable because they contain few number of speakers. The canonical dataset for English is LibriSpeech, containing about 2500 speakers. We have 2.7 times less in our training data, hence, worse quality is expected.

4.4 Our Approach

We have absolutely followed the DeepSpeech article in terms of architecture and the majority of training details. For architectural details see Fig.1.

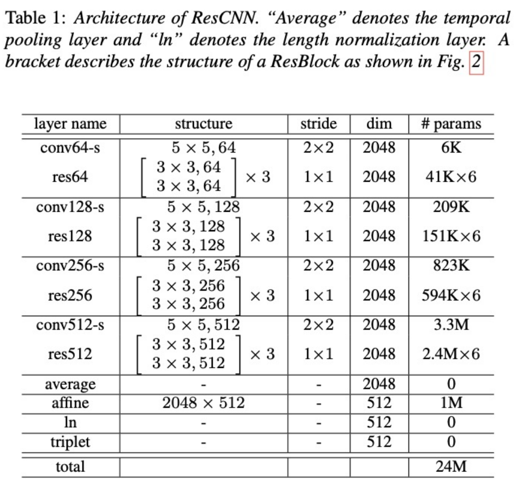

Table 1: *Architecture of ResCNN. "Average" denotes the temporal pooling layer and "ln" denotes the length normalization layer. A bracket describes the structure of a ResBlock as shown in Fig.* 2

layer name	structure	stride	dim	# params
conv64-s	$5 \times 5, 64$	2×2	2048	6K
res64	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	1×1	2048	41K×6
conv128-s	$5 \times 5, 128$	2×2	2048	209K
res128	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 3$	1×1	2048	151K×6
conv256-s	$5 \times 5, 256$	2×2	2048	823K
res256	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 3$	1×1	2048	594K×6
conv512-s	$5 \times 5, 512$	2×2	2048	3.3M
res512	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	1×1	2048	2.4M×6
average	-	-	2048	0
affine	2048×512	-	512	1M
ln	-	-	512	0
triplet	-	-	512	0
total				24M

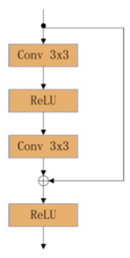

Figure 2: *Detailed view of ResBlock. A convolution block Conv* 3×3 *is parameterized by the filter size* 3×3*, the zero padding 1 in both directions and the consecutive striding* 1×1

Figure 1: Architecture of DeepSpeaker from the original article. Square brackets in the left picture are ResBlocks, illustrated in the right. "ln" layer is l2 normalization and affine layer is just a fully-connected layer. There is also one additional fully-connected layer [512 x num-of-speakers] during the first part of training.

Training Details: 2 stage learning - first with additional classification layer, speaker labels and CrossEntropy loss for 10 epochs, than for 15 epochs without classification layer and with triplet loss. SGD optimizer with 0.99 momentum was used in both cases, however, learning rate scheduler and initial learning rate were ignored in the second case as embeddings collapsed to a single point.

5 Analyses

5.1 What was good in terms of implementation:

The framework was chosen correct. Some of DeepSpeaker models are not standard, like normalization one. Defining user layers is expensive, but PyTorch it is much cheaper than Lamda layers in Keras.

5.2 What can be improved in terms of implementation:

One of the most time-consuming (10 hours) and difficult part was creating a dataset because of infrastructure latency (Google Colab and Google Drive) and slow MFCC extraction. Also, Drive sometimes fails to work with directories with big amount of files ($\approx$ 76 000 in our case).
There also were problems with triplet loss as it converged to margin, hence, all vectors collapsed to 1 point and it is difficult to control such behaviour.

5.3 Other improvements:

Due to limited amount of data triplet loss appeared to be non-optimal here and resulted in vectors collapsing after a couple of epochs even with low learning rate (0.0001 or 0.00001). Hence, we used the result of first epoch with $\alpha = 0.0001$ to calculate embeddings, however, I believe that bigger and more balanced data would help to solve this issue with triplet loss convergence to margin.

5.4 Model Evaluation

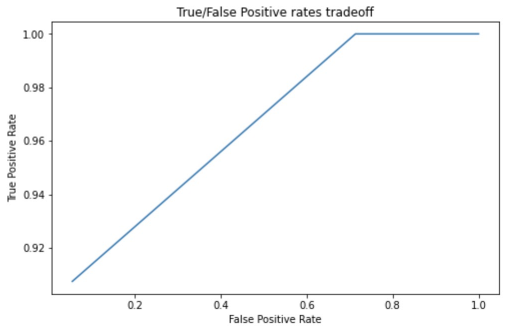

Figure 2: Analog of ROC-AUC but for True positive and False positive rates

For model evaluation we used 2 metrics, one is classical accuracy, another is EER. EER is equal error rate. Let assume that FP - false positive, FN - false negative, TP - true positive, TN - true negative, FAR - false acceptance rate, FRR - false reject rate. Then $FAR = \frac{FP}{FP+TN}$ and $FRR = \frac{FN}{TP+FN}$. EER is the value where FRR and FAR intersect. To identify if a user is correctly identified we take 2 embeddings, one with a known speaker id, calculate cosine similarity between 2 vectors and check if the distance is less or above a threshold. If less than we assume that this is the same speaker for both embeddings and we know the speaker, otherwise it is a different speaker and we continue searching. For evaluation purposes we compared all possible couples of vectors.Our model showed EER = 0.0700 and Accuracy = 0.9451 on a validation dataset (all of the speakers were new for a model). Please note that validation dataset was not a part of training one, hence, there were some new noises and could be a small distribution shift.

6 Conclusion

Our model showed worse results that SincNet, which is not surprisingly as during the second stage of training triplet loss collapsed embeddings. Moreover, SincNet is a newer model - DeepSpeaker is from 2017 and SincNet is from 2019, hence, it is logical that more recent models are better. However, I would like to continue trying and compare both models performances on a big enough dataset.

References

1. Li, C., Ma, X., Jiang, B., Li, X., Zhang, X., Liu, X., Zhu, Z. (2017). Deep speaker: an end-to-end neural speaker embedding system. arXiv preprint arXiv:1705.02304, 650.

2. Haytham Fayek. (2016) Speech Processing for Machine Learning: Filter banks, Mel-Frequency Cepstral Coefficients (MFCCs) and What's In-Between.
Available at https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html

3. Ravanelli, M., Bengio, Y. (2018, December). Speaker recognition from raw waveform with sincnet. In 2018 IEEE Spoken Language Technology Workshop (SLT) (pp. 1021-1028). IEEE.

4. Snyder, D., Garcia-Romero, D., Sell, G., Povey, D., Khudanpur, S. (2018, April). X-vectors: Robust dnn embeddings for speaker recognition. In 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 5329-5333). IEEE.

5. Rahman Chowdhury, F. R., Wang, Q., Moreno, I. L., Wan, L. (2018, April). Attention-based models for text-dependent speaker verification. In 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 5359-5363). IEEE.