Speaker Identification Challenge

SCRIBO

TechChallenge WS18/19
at the Technical University Munich

Piotr Tatarczyk
tatarczyk.p@gmail.com

Janis Postels
janis.postels@tum.de

Vladimir Fomenko
vlad.fomenko@tum.de

Isa Usmanov
isa.usmanov@tum.de

Calin Buruian
calinburuian@gmail.com

Abstract

In this work, we propose a speaker identification pipeline that utilizes a neural embedding network based on LSTM cells. Our mobile architecture can be efficiently run on Huawei mobile phones equipped with the Kirin 970 Neural-network Processing Unit (NPU) and to the best of our knowledge is the first one designed specifically for this hardware configuration and speaker identification task. Furthermore, we prove that the NPU allows for the use of neural networks capable of achieving state-of-the-art performance in speech-related problems and compare it to a large-scale architecture. The proposed neural networks are trained with triplet loss to tell two speakers apart, being useful in a handful of speech-related tasks. This is achieved by embedding speech utterances into a fixed dimensional euclidean space, while maximizing the distance between samples of different speakers and closely grouping samples of the same speaker. This approach builds a foundation for offline mobile deep learning based voice applications, e.g. on-the-phone speech recognition systems as a speaker change detector or voice password applications.

1 Introduction

Speaking is a natural form of human communication and allows for hands-free information exchange between users and devices [6]. With growing popularity of automatic information processing tools, the role of identifying a speaking person in such systems increases. Speaker recognition is the ability to identify who is speaking. This task is often confused with speech recognition, which is the ability to recognize what words have been said. A system capable of telling any two speakers apart with high accuracy finds its use in numerous tasks in the speech area, e.g. speaker identification, verification, speaker change detection.

Speaker identification is a task of finding the identity of an unknown speaker by comparing his/her voice with voices of the enrolled speakers. Speaker verification deals with judging whether two samples at hand belong to the same speaker [6]. To solve these tasks, we propose a deep neural network that learns to embed speech utterances onto a fixed dimensional euclidean space while maximizing the distance between samples from different speakers and closely grouping speech segments of the same speaker. As shown in Figures 1 and 2, our embedding network enforces stronger separation of distinct speakers. Thus, our neural embedding architecture can be used for multiple tasks. In our experiments, we propose a pipeline and show that it can achieve high performance on both speaker identification and verification.

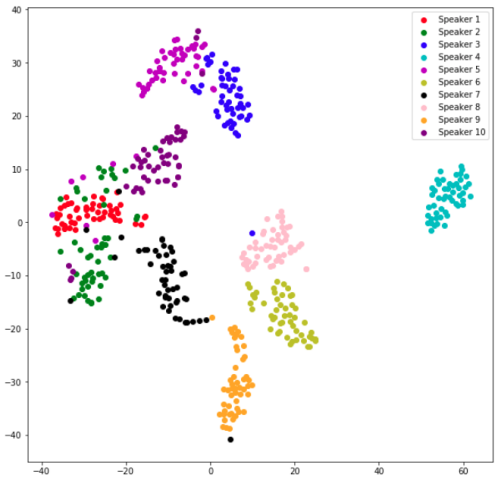

Figure 1: Visualization of a 2-D projection of MFCC feature vectors. Each point is an average 16-dim MFCC feature vector computed for 3.5 sec speech utterances a single speaker. Some structure is visible but clusters are not always easily separable.

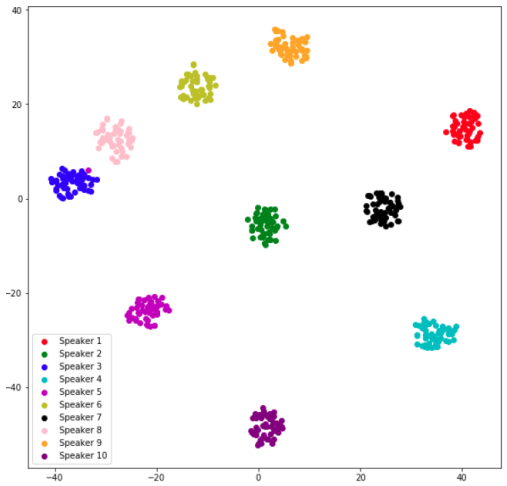

Figure 2: Visualization of 2-D projection of speaker embedding vectors computed by our proposed mobile architecture. Embeddings are computed for 50 speech utterances for each of the 10 speakers from the test set. Each utterance was 3.5 sec long. Clusters are clearly separable.

2 Related works

Our method builds upon successes of deep learning in the last years. Recently, a neural embedding network TristouNet [7] was shown to achieve state-of-the-art results in speaker verification using LSTM layers. Also GRU and ResCNN architectures were proven to work well on short utterances, as proposed by Li et.al [6]. There exist proven speaker identification algorithms that do not utilize deep learning. One of the very successful system design was proposed by Qi [6] and was a combination of verbal information verification and speaker verification with Hidden Markov Models as the underlying technology achieving 1.2% equal error rate. Zeinali et.al [9] showed that data augmentation can improve performance of deep learning architectures in related tasks, increasing strongly computational effort due to enlarger data set. Even though task-specific algorithm can achieve higher accuracy, due to the computational and memory cost, when used in mobile applications, an embedding network can outperform the classic approaches in its flexibility.

3 Architecture

Recurrent deep neural networks, mostly LSTM, have been successfully used for various speech related tasks [5]. The have also been utilized for acoustic modeling of speakers [7, 8]. Inspired by those achievements, we choose LSTMs for our architecture.

Mobile architecture

Huawei NPU hardware imposed strict constraint regarding the architecture. Those constraints had a great impact on the performance of the system. Each dimension on the network and in the data batch has to be divisible by 16. For the mobile architecture, we hence build a network consisting of one static LSTM layer and two fully connected layers with tanh activation at the last layer. Using ReLu would cause many terms in the embedding vector to be zero. The hidden dimension of the LSTM is 128 and the number of nodes in the fully connected layers is 128 and 32 yielding a 32-dimensional embedding.

Large-scale architecture

The large-scale neural architecture is not constraint by Huawei hardware limitations. Thus, a free choice could be made and was inspired by TristouNet [7] regarding the type of layers. To embed the extracted MFCC features, we use two bi-directional LSTM layers with a hidden dimension of 512 and subsequently add two fully-connected layers, with the final tanh activation function as in the mobile architecture. The number of hidden nodes in the fully connected layers is 256 and 128, yielding a 128-dimensional embedding.

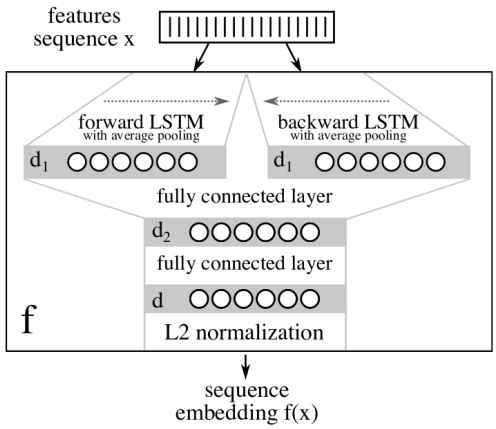

Figure 3: Overview of the used neural network for large-scale architecture. Source: [7]

4 Training data

The data included LibriSpeech [2] subsets: “clean-train-100” and “clean-train-360”, both having together 1105 English speakers; Clarin Polish data set with 552 speakers and an additional dataset for this challenge containing 165 English speakers.

Data preprocessing

All audio data was converted to wave format and resampled to 16kHz using ffmpeg [3]. Silence in the data set can heavily influence the performance of a voice embedding neural network [4]. Thus, speech activity detection was done offline before training. For non-speech segment removal, a pre-trained support vector machine (SVM) classifier is used. Speech removal reduced the overall size of the data set from 67.4GB to 55GB.

Feature extraction

Librosa library [2] was used as MFCC feature extractor. The main motivation was the availability of this library in Python and Java. Python was used for training and Java to implement and test the architecture on a Huawei mobile phone with NPU. Librosa Java package demands the FFT window and the step size for MFCCs to be powers of 2. We hence have the windows equal to 128ms and the step size of 16ms (25ms and 10ms respectively for the large-scale architecture). Additionally, the requirement from the Huawei NPU is that each input dimension is divisible by 16 so we stick to 16 MFCC coefficients only. We also do not include energies and its derivatives to optimize computational cost. For the large-scale architecture, we use 19 MFCC coefficients with their first and second derivatives that constitute a 59 dimensional feature vectors. Other parameters are set equal to the ones used for the mobile architecture. We stack feature vectors for one forward pass so that they cover 3.58s for mobile architecture seconds of speech and 2s for the large network. Finally, the input to the LSTM is of the size (224, 16) for mobile architecture and (224, 59) for the large architecture. To increase the training speed, we compute speech features for the whole data set after silence removal, but before training to reduce the load. An important distinction in feature preparation for the two networks is that for the mobile network we do not normalize feature batches over whole utterance. Only single feature vectors are normalized. For the large scale network, we do normalize over the whole duration of utterance. It will be worth trying to train a large network without this kind of normalization as it is hypothesized to worsen the overall performance.

5 Training

Inspired by TristouNet [7], we train both networks to optimize for triplet loss. The hypothesized goal is that the network learns to tell apart any two speakers from the data set.

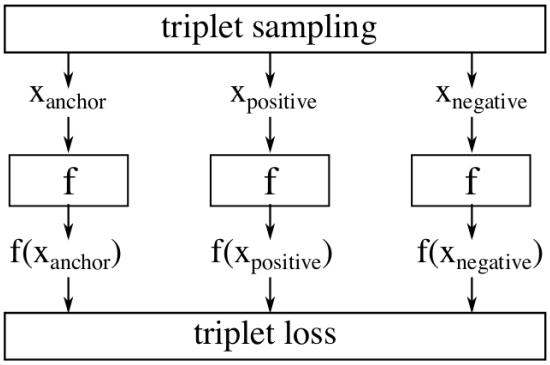

Figure 4: Triplet loss training. In each iteration a network computes a forward path of three samples. Two of them are from the same speaker (anchor and positive) and one is from a different speaker (negative). Source: [7]

The anchor and positive samples belong to one speaker and the negative sample to a different speaker. One iteration of the forward pass includes all those three vectors passing the network. If the distance between the anchor and the negative sample is not larger by the alpha margin or more from the distance between the anchor and the positive sample, the triplet is not contributing to the training and makes training slower.

Mobile architecture training

We leave 300 speakers as the validation set, and train on the rest. The pre-computed batches contain features of randomly chosen speaker from the dataset. Opposed to the large-scale training, we do not apply any tactic to sample only valid triplets of speakers that contribute to training. We set the alpha parameter of the triplet loss to 0.2 and train the architecture on the constant learning rate 0.001. We built our own training pipeline in Tensorflow as the HiAI IDE allows for model adding only in .pb format. The architecture was trained on a Intel i5 CPU with 8GB of RAM and took 3 days to reach 19 epochs. As small as it sounds, low number of epochs was shown to yield convergence and was also proposed by McLaren et. al [4]. The training could have been continued as we did not see any rise in the validation loss yet. The batch size was set to 128.

Large-scale training

A specific tactic for sampling appropriate triplet was used for training of this network and was inspired by FaceNet [11]. Large network seemed very sensitive to the appropriate choice of the triplets. We left only 200 speakers for validation to have more training data. This network was trained on a Google Virtual Machine with 32GB RAM, and Nvidia K80 graphics card and took around 24 hours to converge. The learning rate varied from$1.6 \cdot 10^{-4}$ to $1.6 \cdot 10^{-3}$ and the used optimizer was Adam with time-varying momentum between 0.94 and 0.85. We implemented the training in Pytorch 0.4 based on Pyannote Library [10].

6 Inference pipeline

When using an embedding network for verification and identification, we split the pipeline into the so called enrollment and query stages. Enrollment speakers, in the case of a voice password application, will be allowed to enter the system. Query speakers contain the enrolled speakers and the unwanted intruders. The goal is to let in the enrolled speakers and catch the unwanted ones. So during querying we can formulate the problem as a binary classification based on the highest similarity to the enrolled speakers and an experimentally found threshold.

For enrollment, we compute average embeddings of randomly chosen, 2 second (or 3.58 second for mobile) utterances. As the enrollment is done just once once, it is in our opinion reasonable to request a longer speech sequence and more samples from the user not harming the usability of the application. For the query time, utterances of 2 seconds (or 3.58 seconds for mobile) are averaged. Averaging embeddings of more utterances, even strongly overlapping if the audio at hand is short, is shown in our experiments to deliver more precise speaker embeddings.

For speaker verification, we can experimentally find an appropriate threshold for the smallest distance between the query speaker and enrolled speakers. In this task, it is reasonable to decrease the false positives (unknown speaker logs in successfully) raising the false negatives rate (enrolled speakers cannot log in) if security is more important then usability. Nevertheless, some balance has to be found. In speaker identification, we will be computing distances from the query speaker to all enrolled speakers and by choosing the smallest distance, we point to the predicted target class.

7 Experiments

Having the perspective of the potential use of the network in mobile applications, we tested the approach in the following scenarios: personal password app (speaker verification) and classifica-

tion (speaker identification). Equal error rate (EER), is a widely accepted performance measure of speaker verification architectures. It corresponds to a threshold at which the false rejection rate is equal to the false acceptance rate. The EER point corresponds to the intersection of the ROC curve with the straight line of 45 degrees. The EER performance measure is not the regular point of operation, but it is commonly used as a performance measure in speaker verification systems. In production, it is often moved towards lower false acceptance rate to increase safety [12].

Password application (speaker verification)

We have a test set of 80 speaker recordings (libri-dev and libri-test subsets) with removed silence. Those speakers have not been seen by the networks during training. We enroll 40 speakers, and build queries for all 80 test speakers. So one half of the query should ideally be accepted. We plot receiver operating characteristic, that shows acceptance (true positive) and rejection (false positive) rates with respect to several threshold values which are chosen among all computed similarities. It is important to note that the number of enrolled speakers influences the accuracy as the query speaker has a higher chance of having a similar voice to an enrolled speaker.

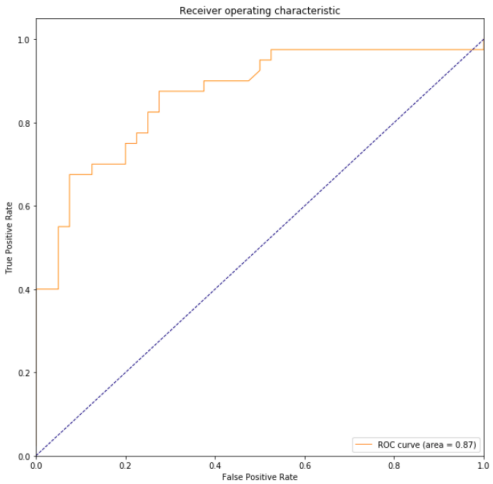

Figure 5: ROC-curve for personal password application scenario for large-scale architecture. 40 enrolled speakers and 80 query speakers from the test set (unseen). Randomly sampled and averaged five 2s segments for enrollment and two 2s for query. EER equal to ˜17%.

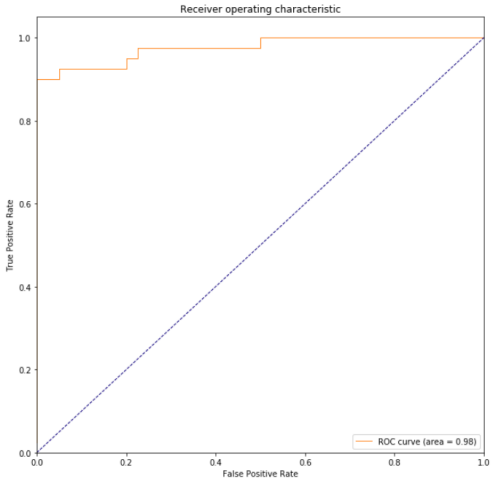

Figure 6: ROC-curve for personal password application scenario for mobile architecture. 40 enrolled speakers and 80 query speakers from the test set (unseen). Randomly sampled and averaged five 3.58s segments for enrollment and two 3.58s for query. EER equal to ˜3%.

Speaker identification (classification)

We build models for each of the 80 speakers by computing average embedding vectors of randomly choose utterances, each 2 seconds long (3.58 seconds for mobile architecture). We again sample from the set of the same speakers (not the same utterances), and try to correctly classify those audio snippets by computing average embedding vectors and finding a speaker model with the

highest cosine similarity. We report performance on the test set with unseen 80 classes in the form of classification accuracy, showing also the influence of the number of sampled utterances for enrollment and query on the results.

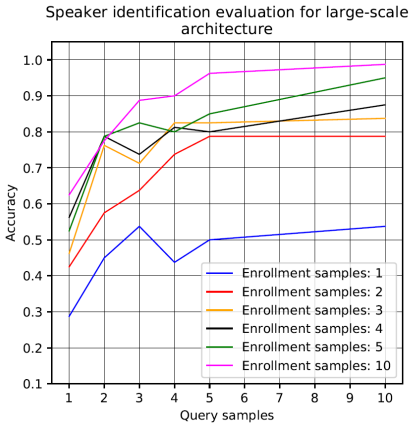

Figure 7: Identification accuracy of the large-scale architecture versus number of sampled query utterances for different numbers of enrollment utterances. Short utterance length (2 seconds) causes low accuracy for low sample numbers. We approach 100% identification accuracy for 10 enrollment samples and 10 query samples.

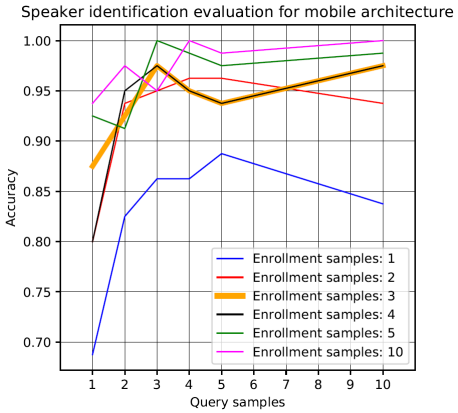

Figure 8: Identification accuracy of the mobile architecture versus number of sampled query utterances for different numbers of enrollment utterances. Due to longer utterance length, this network achieves high accuracy already at lower numbers samples. This network approaches 100% for already 5 enrollments and 3

Due to time constraints, we did not train networks on the same speech utterance lengths. We can still compare results looking at the points where network input has equal lengths e.g 2 query and enrollment samples in the large-scale model that correspond to 1 query and enrollment sample in the mobile model. They both correspond to inputs about 4 seconds long and yield similar accuracy of around 65-70%. Nevertheless, smaller network with longer utterances achieves 100% accuracy demanding much less speech data. There is a general trend that the accuracy is strongly influenced by the number of sampled sequences. As in the case of speaker verification, it is reasonable to have more samples for the model, that is computed just once. For convenient usability, we would like the needed voice recordings to be possibly short. Thus, we could suggest having 10 samples for the enrollment and 2 or 3 samples for the query.

8 Conclusions and overview

The main conclusion is that a neural embedding network can achieve high performance on multiple tasks like speaker verification and identification. Utterance number and length is decisive in both tasks. Due to the fact that during experiments we used the mobile network in a speech recognition system, utterances had to be longer. This seems to allow the mobile network to achieve such high performance relative to the large scale network. The mobile architecture and our own training pipeline achieve very high EER on the test set, while being perfect on the

training set. This leads to the conclusion that generalization is still an issue but having the possibility to train the network on the speakers that later will be enrolled/queried could boost performance. The training of the mobile network was very sensitive to the batch size. Low batch size caused loss stagnation, due to unfulfilled criteria for triplets as any triplet mining method was not used for this network. Further, different architectures (deeper, wider) can be tested as well as data augmentation with reverbation, music and white noise. Moreover, in cases where such a flexible network is not needed, a network tailored for a specific tasks can achieve overall higher results.

During the challenge, we did not build any password app around the mobile architecture on the phone. This network was used for speaker change detection in our transcription application. For the sake of speaker identification challenge, we evaluated the mobile architecture on alternative tasks. Trained models and code for testing and to reproduce results is publicly available on: https:// github.com/PiotrTa/Huawei-Challenge-Speaker-Identification. Code for the training of the mobile architecture will be released soon.

References

[1] V. Sharma, P. Bansal, "A Review On Speaker Recognition Approaches And Challenges", in International Journal of Engineering Research & Technology, 2013.

[2] Panayotov, V., Chen, G., Povey, D., and Khudanpur, S. (2015). Librispeech: an asr corpus based on public do- main audio books. In Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on , pages 5206–5210. IEEE. 4, 4

[3] FFmpeg Developers. (2016). ffmpeg tool (Version be1d324) [Software]. Available from http://ffmpeg.org/ 4

[4] M. McLaren, D. Castan, M. K. Nandwana, L. Ferrer, and E. Yilmaz, "How to train your speaker embeddings extractor," in Proc. Speaker Odyssey: The Speaker and Language Recognition Work- shop , 2018. 4, 5

[5] A. Graves, A. Mohamed, and G. Hinton, "Speech Recognition with Deep Recurrent Neural Networks," in 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) , 2013. 3

[6] Li, Qi. Speaker Authentication, Springer, 2011. ProQuest Ebook Central. 1, 2

[7] H. Bredin, "TristouNet: Triplet Loss for Speaker Turn Embedding," in ICASSP 2017, IEEE International Conference on Acoustics, Speech, and Signal Processing , New Orleans, USA, March 2017 2, 3, 3, 3, 5, 4

[8] Ji, Ruifang et al. "An End-to-End Text-Independent Speaker Identification System on Short Utterances." Interspeech (2018). 3

[9] Zeinali, Hossein & Burget, Lukas & Rohdin, Johan & Stafylakis, Themos & Cernocky, Jan. (2018). How to Improve Your Speaker Embeddings Extractor in Generic Toolkits. 2

[10] Yin, Ruiqing & Bredin, Hervé & Barras, Claude. (2017). Speaker Change Detection in Broadcast TV Using Bidirectional Long Short-Term Memory Networks. 3827-3831. 10.21437/Interspeech.2017-65. 5

[11] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: a Unified Embedding for Face Recognition and Clustering," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 815– 823. 5

[12] S. Furui, "Speech and Speaker Recognition Evaluation", Springer, 2007, Department of Computer Science, Tokyo Institute of Technology Tokyo, Japan.

7

9 Appendix

Large-scale training details

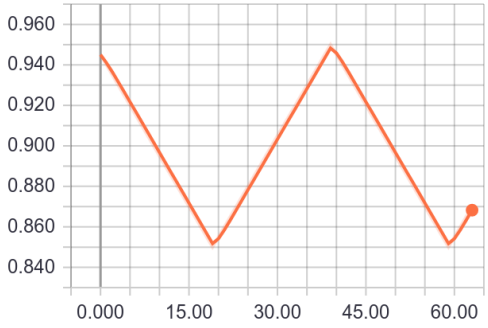

Figure 9: Momentum (Adam optimizer) scheduling over epochs.

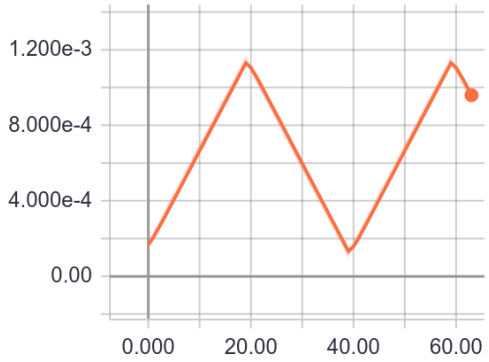

Figure 10: Learning rate scheduling over epochs.

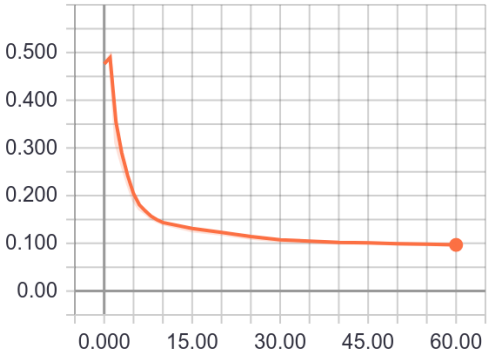

Figure 11: Equal error rate on training data over epochs.

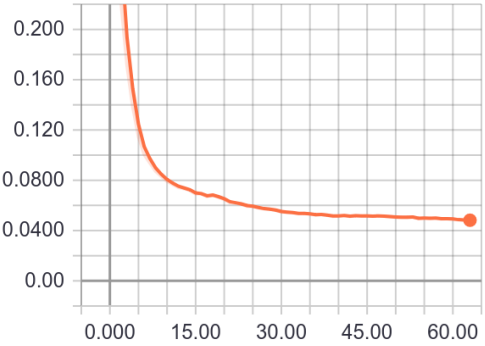

Figure 12: Visualization of the training loss.

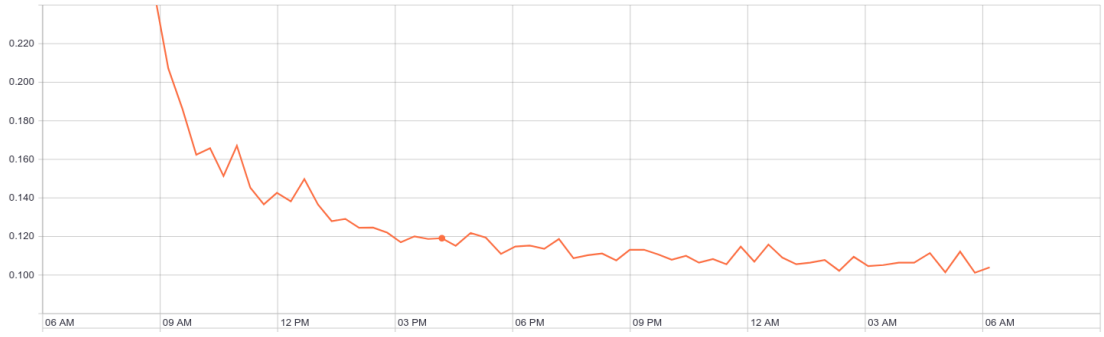

Figure 13: Equal error rate on the validation set.

Mobile network training details

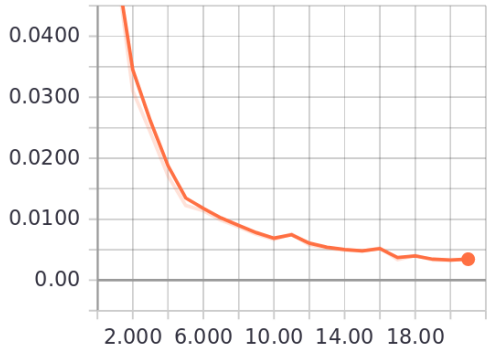

Figure 14: Training loss over epochs.

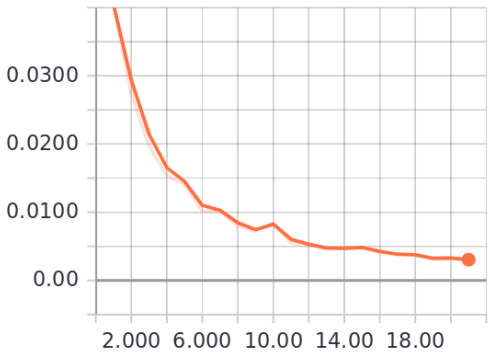

Figure 15: Validation loss over epochs.

Further speaker verification experiments

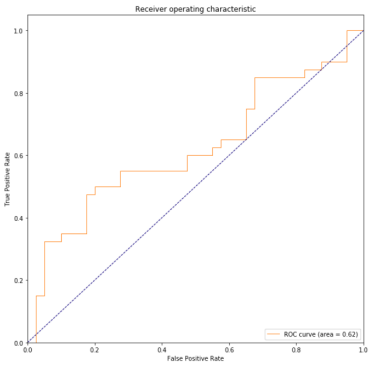

Figure 16: ROC-curve for personal password application scenario for large-scale architecture. 40 enrolled speakers and 80 query speakers from the test set. Randomly sampled and averaged one 2s segment for enrollment and one 2s for query. Estimated EER equal to ˜40%.

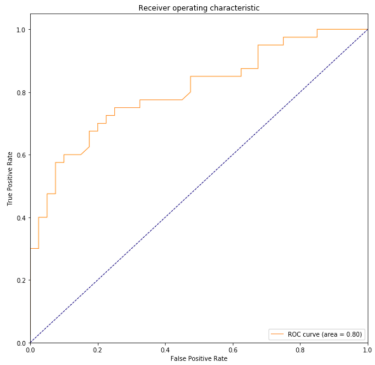

Figure 17: ROC-curve for personal password application scenario for mobile architecture. 40 enrolled speakers and 80 query speakers from the test set (unseen). Randomly sampled and averaged one 3.58s segment for enrollment and one 3.58s for query. EER equal to ˜20%.

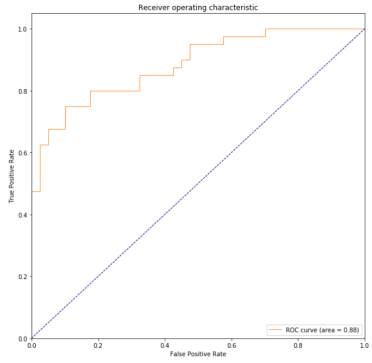

Figure 18: ROC-curve for personal password application scenario for large-scale architecture. 40 enrolled speakers and 80 query speakers from the test set. Randomly sampled and averaged ten 2s segments for enrollment and two 2s for query. Estimated EER equal to ˜15%.

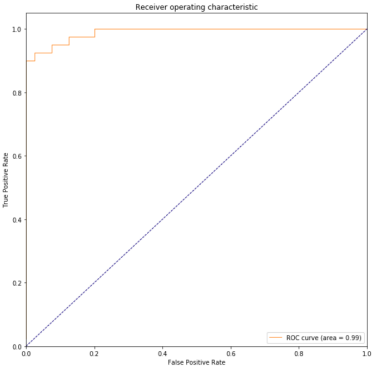

Figure 19: ROC-curve for personal password application scenario for mobile architecture. 40 enrolled speakers and 80 query speakers from the test set (unseen). Randomly sampled and averaged ten 3.58s segments for enrollment and two 3.58s for query. EER equal to 3%.