Speaker identification

Kučera Adam, Bc.
xkucer95@stud.fit.vutbr.cz

Sabo Jozef, Bc.
xsaboj00@stud.fit.vutbr.cz

June 7, 2021

Abstract

In this paper, we examine the impact of 3 different loss functions, used as an objective function, when training models with x-vector architecture. X-vector architecture is deep neural network (DNN) architecture, whose main goal is to extract distinctive representations of a speakers from speaker utterances. Extracted representations of speakers from x-vector architecture are high dimensional vectors called x-vectors. Although x-vector architecture is implemented in some publicly available frameworks, we have worked with our own pytorch implementation. In the project, we have evaluated the trained models with a speaker verification metric: equal error rate (EER). As a backend to compare pairs of x-vectors, we used cosine similarity. Performance of the x-vector model was the best, when we used our own implementation of the Angular Softmax with cross-entropy as a loss function for training the model ($EER = 8.04\%$). On the other hand, performance was the worst when we used the Triplet Margin Loss ($EER = 16.46\%$).

1 Introduction

Machine learning tasks connected with audio (speaker identification, speaker verification) had for a really long time a good performing basis in statistical models called Gaussian Mixture Models (GMM) [1]. More precisely, for the speaker embeddings, the universal background model (UBM) [2] based on this GMM is trained with maximum likelihood maximization, then for the target utterance the means of this UBM are maximum aposteriori (MAP) adapted and the vector of those means called supervector is the final embedding. Significant performance improvement over GMM came with vector representations of speakers called i-vectors (i-vectors are vectors produced by Joint Factor Analysis over GMM mean supervectors) [3]. In our project, we have worked with vector representations of the speakers called x-vectors. Using x-vectors, for speaker verification or identification, is not a state of the art approach anymore, but x-vectors offers more discriminative features between different speakers than i-vectors [4].

For producing x-vectors, we have used DNN architecture, which is shown in the figure 1. Most interesting part of architecture, statistical pooling layer, is located between convolution and fully connected layers. This layer is necessary for producing speaker embeddings from variable-length utterances. Statistical pooling layer aggregates frame-level information into one embedding by concatenating mean and standard deviation of outputs from convolution layers. Models with this architecture can be simply trained to correctly classify speakers from a training dataset with multi-class cross entropy as an objective function (see subchapter 2). This way the model learns how to extract discriminative features from variable length utterances of the speakers. After training the model, any speaker utterance can be passed as input to the model and outputs from the first or second fully connected layer are desired embeddings.

Authors of the architecture found out in their experiments, that output from first fully connected layer gives best results for the speaker verification (in figure 1 marked as 'embedding A') [4].

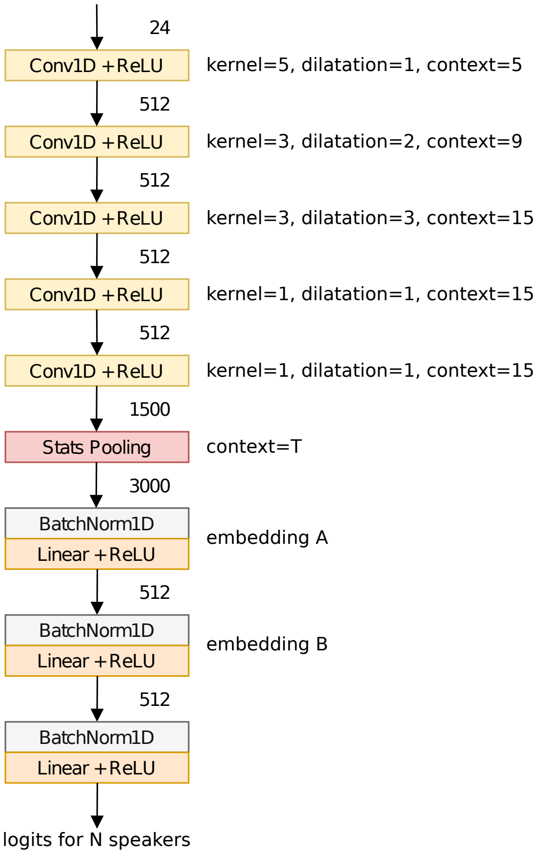


Figure 1: DNN architecture for extracting x-vectors

2 Our work

The x-vector architecture has significantly outperformed the traditional statistical approaches and is very popular model to the date, although it is not state of the art anymore. Besides different newer architectures, the original x-vector model performance was further improved using different training methods and loss functions. In [5], they used multi-task training to train the model and had visible improvements in performance metrics compared with original training method. They enhanced cross-entropy loss from softmax activation with direct optimization on L2 distances between embeddings from hidden layer using triplet margin loss. They also used small neural network to extract more complex distance metric.

The goal of our work was to test different loss functions separately to see their potential to train x-vector architecture and infere their contribution in multi-task losses. As a backend scoring metric, we used cosine similarity as a very strong and popular metric. Another popular backend in recent years is PLDA[1], but lately it lost its popularity due to the increasing capabilities of DNN models. The main reason why we didn't choose PLDA is that it is a classifier and has hyperparameters, which might shadow the actual DNN model performance.

Implementation and dataset

We build the model using pytorch library in order to get more flexibility, despite the fact the model used in production is already implemented in popular C++ Kaldi library. We decided to use the smaller version of the new VoxCeleb [6] dataset, which consists of 148 642 utterances from 1211 speakers in the training set, and 4874 utterances from 40 speakers in the test set, all from YouTube videos. We preprocessed each utterance before the training step. We used single channel audio with 16kHz sampling rate as a raw input. Preemphasis with coeficient 0.97 has been then applied and the audio has been divided into overlapping frames each of length of 25 milliseconds and with step of 15 milliseconds between adjacent frames. For each frame, the STFT[2] has been computed and finally the 128-dimensional spectrogram has been extracted using Mel frequency filter banks. In traditional approaches, the high correlation in this spectrogram used to be undesirable, but in DNN approaches more dimensionality has been shown to yield better results [7]. However, we used a decorrelated spectrogram – the MFCC[3], because in the original x-vector paper they used those. Initially we wanted to test both, but soon we realized that our hardware might not let us to do so, and we decided to use a 24-dimensional MFCC features only as a input to the model. For all these audio processing tasks, we used python framework librosa. We've implemented and used 3 training methods, each defined by different loss function.

Softmax with cross-entropy

We trained the first model using original training method, i.e. closed set classification trying to classify respective utterances to each speaker in the training dataset. The multiclass cross-entropy loss function on softmax activation of the last layer (eq. 1) has been optimized.

$$\mathcal{L}_{softmax} = \frac{1}{N}\sum -\log\left(\frac{e^{\boldsymbol{x}\cdot\boldsymbol{W}_i^T+\boldsymbol{b}_i}}{\sum_j e^{\boldsymbol{x}\cdot\boldsymbol{W}_j^T+\boldsymbol{b}_j}}\right) \tag{1}$$

We trained the network using Adam optimizer with default parameters while it was seemingly converging and then tuned it using SGD with momentum 0.9 and small $\eta = 1e^{-4}$. The training has been done on variable length utterances with minibatch size of 64. We used a cross-validation set created from the training set to determine the convergence of the optimizer.

[1] https://towardsdatascience.com/probabilistic-linear-discriminant-analysis-plda-explained-253b5effb96

[2] Short Time Fourier Transform

[3] https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html

Triplet margin loss – direct optimization

The second approach we've tested is direct optimization using standard triplet margin loss (eq. 2) with L2 distance, where $\boldsymbol{A}$, $\boldsymbol{P}$ and $\boldsymbol{N}$ are embedding vectors for anchor, positive (from the same speaker), and negative (from a different speaker) utterances respectively. The triplet loss is minimized, when for every speaker the intra-class distances are all smaller than inter-class distances by a margin of α.

$$\mathcal{L}_{triplet} = \frac{1}{N} \sum \max\left(\|\boldsymbol{A} - \boldsymbol{P}\|^2 - \|\boldsymbol{A} - \boldsymbol{N}\|^2 + \alpha, 0\right) \tag{2}$$

The one benefit of this approach over classification is obvious – it does not require exact class annotations as closed set classification does. But from the performance perspective, it is not so obvious. One might argue that direct training should lead to better results, but softmax with more than thousand classes is forced to learn very discriminative representation and the optimization takes all those classes into account in every learning step. It is known that whereas triplet learning for embedding extraction is capable to outperform other approaches, it is crucial to employ good sampling technique and use hard-negative mining [8].

In our work, we've implemented naive sampling where all triplets are sampled randomly with uniform probability during the whole training. Another important thing about triplet margin loss is its hyperparameter α. We used the value of 0.8 as recommended in [5]. The ReLU activation from the embedding layer has been removed and also the output embeddings were normalized in order to prevent network from degrading into extremely increasing or decreasing the embedding magnitude without changing its orientation (and cosine similarity backend score). We trained the model in a similar manner as the previous one.

Angular Softmax with cross-entropy

The third and the last loss function we've tested is so called Angular Softmax loss [9] or A-softmax. It is a relatively recent approach to the problem of classic softmax activation used for embedding extraction. In classification using softmax, there is no direct way of maximizing the inter-class variability and minimizing the intra-class variability, which is desired property for a good embedding model. The final cross-entropy loss is not forced to keep any margins, it only cares about classifying the samples into the classes correctly in a discreet sense. The angular softmax can be viewed as an enhanced softmax idea based on the geometric definition of the dot product, which implies the equivalence (eq. 3), where θ_i is the angle between the class weights $\boldsymbol{W}_i$ (column of the linear layer matrix for row vector convention) and the input vector $\boldsymbol{x}$.

$$\frac{e^{\boldsymbol{x} \cdot \boldsymbol{W}_i^T + \boldsymbol{b}_i}}{\sum_j e^{\boldsymbol{x} \cdot \boldsymbol{W}_j^T + \boldsymbol{b}_j}} = \frac{e^{\|\boldsymbol{x}\| \cdot \|\boldsymbol{W}_i\| \cos(\theta_i) + \boldsymbol{b}_i}}{\sum_j e^{\|\boldsymbol{x}\| \|\boldsymbol{W}_j\| \cos(\theta_j) + \boldsymbol{b}_j}} \tag{3}$$

The main idea is to multiply the angle θ_i, i.e. angle between x and weights of the ground-truth class by some value m. Because the cosine function is monotonically decreasing in the interval $\langle 0, \pi \rangle$, the relation $j \neq i : \cos(\theta_i) > m \cdot \cos(\theta_i) > \cos(\theta_j)$ holds true in this interval. In [9] it is also shown that this new restriction creates angular margin between weight vectors of different classes. This margin forces the model to learn more discriminative features, because it has to minimize the angles within classes and maximize the angles between classes. Recently, there have been more losses with such margins, e.g. Additive Angular Margin Loss [10] where m is incorporated using addition instead of multiplication, but in [11] they used the hyper-spherical loss based on multiplication for a speaker verification task, so we did the same.

We've implemented a linear layer without bias and with weights L2 normalization (as recommended). We used the value $m = 3$, which has been suggested in [11] to be the optimal. They also recommended the integral value of m, so the multi-angle formula could be used to simplify backward gradient computations. In our case of $m = 3$, it can be simplified using the formula (eq. 4). Finally, to remove the restriction of θ_i being in interval $\langle 0, \frac{\pi}{m} \rangle$, the new function has been introduced to replace the cosine (eq. 5).

$$\cos(3\theta) = 4\cos(\theta)^3 - 3\cos(\theta) = 4\left(\frac{\boldsymbol{x}\cdot\boldsymbol{W}}{\|\boldsymbol{x}\|}\right)^3 - 3\left(\frac{\boldsymbol{x}\cdot\boldsymbol{W}}{\|\boldsymbol{x}\|}\right) \tag{4}$$

$$\phi\left(\theta_i\right) = (-1)^k \cos\left(m\theta_i\right) - 2k \tag{5}$$

Now the final A-softmax loss function can be formulated by an expression (eq. 6).

$$\mathcal{L}_{A-softmax} = \frac{1}{N}\sum -\log\left(\frac{e^{\|\boldsymbol{x}\|\phi(\theta_i)}}{e^{\|\boldsymbol{x}\|\phi(\theta_i)} + \sum_{j\neq i} e^{\boldsymbol{x}\cdot\boldsymbol{W}_j^T}}\right) \tag{6}$$

The training process was not trivial in this case. We found out, that such restrictive loss metric is very prone to divergence at the beginning of the optimization (as also mentioned in [10]). However, in [11] they didn't mention this problem, probably because they were able to use very large batches (1000) as they were not training the network on variable length utterances but only on chunks, and they also had better hardware than we did. To tackle this problem, we trained our network with help of softmax activation combined with A-softmax (eq. 7), where we initially started with $\alpha = 0.2$ and $\beta = 0.8$.

$$\mathcal{L} = \alpha \cdot \mathcal{L}_{A-softmax} + \beta \cdot \mathcal{L}_{softmax} \tag{7}$$

Then we slowly moved the coeficients towards $\alpha = \beta = 0.5$ (1 epoch for every step of 0.1) and then we finally removed the helper softmax loss and trained only using A-softmax until the convergence. We used the same strategy as before, i.e. Adam to converge fast somewhere and then SGD with reasonably small η to settle down in the local minima more precisely.

3 Results

The trained models were tested using mentioned cosine similarity metric on the testing dataset. The embeddings were computed for each utterance and then all embeddings were paired into a 4874×4874 score matrix. Finally, the DET[4] curves were plotted and the scalar metric EER [5] has been computed. This metric is a standard way to describe the performance of the binary hypothesis tests. It is very popular in evaluation of biometric systems such as speaker verification model. The metric represents the threshold for which the FPR[6] and FNR[7] values are equal. The threshold is not meant in an absolute way, but normalized into interval $\langle 0, 1 \rangle$ in order to be invariant. Our results are summarized in the table 1.

[4] Detection Error Tradeoff

[5] Equal Error Rate

[6] False Positive Ratio

[7] False Positive Ratio

Training Loss	**EER**
softmax	12.34 %
softmax with ReLU	10.66 %
A-softmax	8.04 %
triplet margin L2	16.46 %

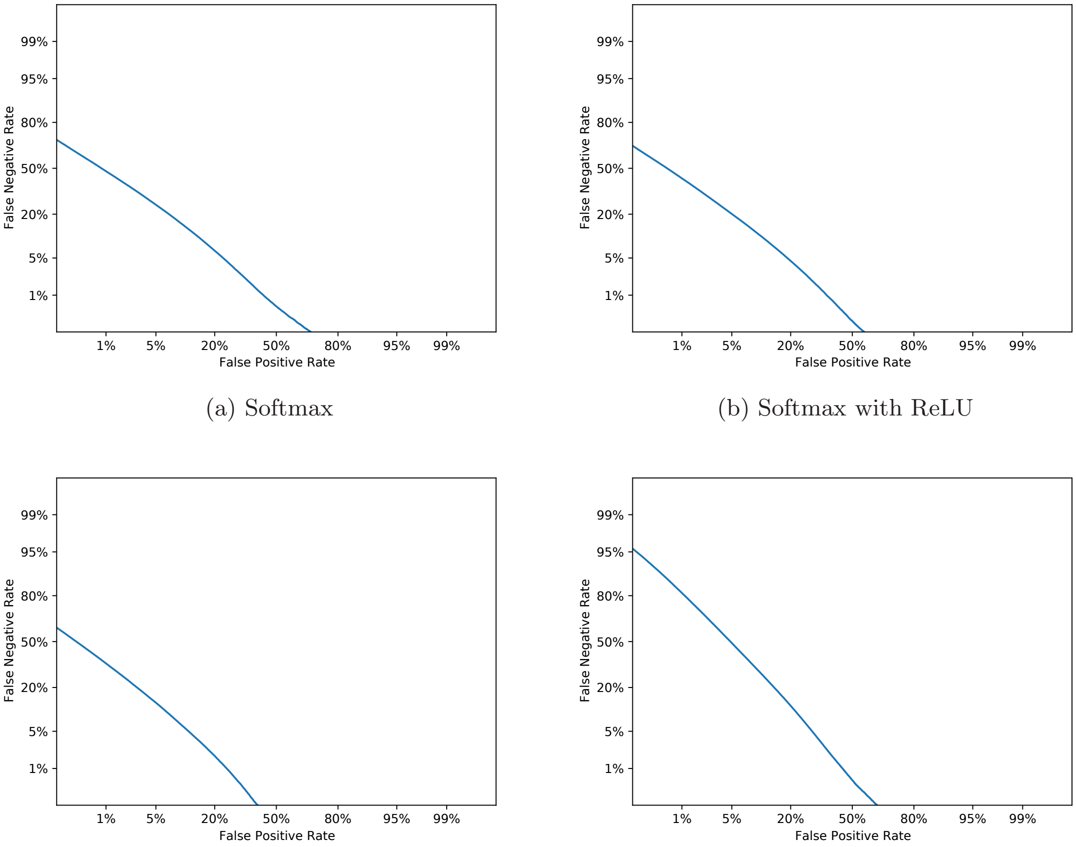
Table 1: Equal Error Rate values for different embedding models

As can be seen, the triplet margin loss with naive sampling performed the worst given the scoring backend. It might definitely improve the performance of the softmax in multi-task training, but on its own it would require avanced sampling technique and also experimentation with the margin α. Also, the L2 distance on normalized vectors is still slightly different than the cosine distance used for scoring, so maybe exploiting our backend decision and using cosine distance in triplet loss would yield better results. However, to test this hypothesis is out of the scope of our work. On the other hand, the Angular Softmax loss function has gained the best performance. The power of the central softmax based loss with large number of classes combined with additional angular margin led to superior results.

There is an open debate about whether to use only linearly activated layer or with some non-linear activation on top of it as an embedding. In triplet margin loss the answer is clear, but in classification based embeddings we tried both variants and found interesting results. While the A-softmax with ReLU activation kept performed much worse, even worse than softmax, the plain softmax performed significantly better with ReLU than with ReLU removed. We suppose that the reason for this is that the ReLU discards information contained in the negative values. Therefore it is not propagated directly to the final minimized loss, so it can be expected that using the negative part would more likely confuse the results rather than improve them. But this explanation ostensibly doesn't apply for the A-softmax function. Interesting future work would be to test whether some other activations that keep also the negative part (e.g. tanh) would lead to even better results than classical ReLU, which is extensively used due to its great gradient properties, but those are not so significant after batch normalization anyway.

Finally, we've also visualized the results in the form of DET curves (fig. 2) and scatter plots of the test utterances embeddings in 512 dimensional space reduced into 2 dimensions mantaining the L2 distances using t-SNE[8] algorithm (fig. 3).

[8] https://towardsdatascience.com/an-introduction-to-t-sne-with-python-example-5a3a293108d1



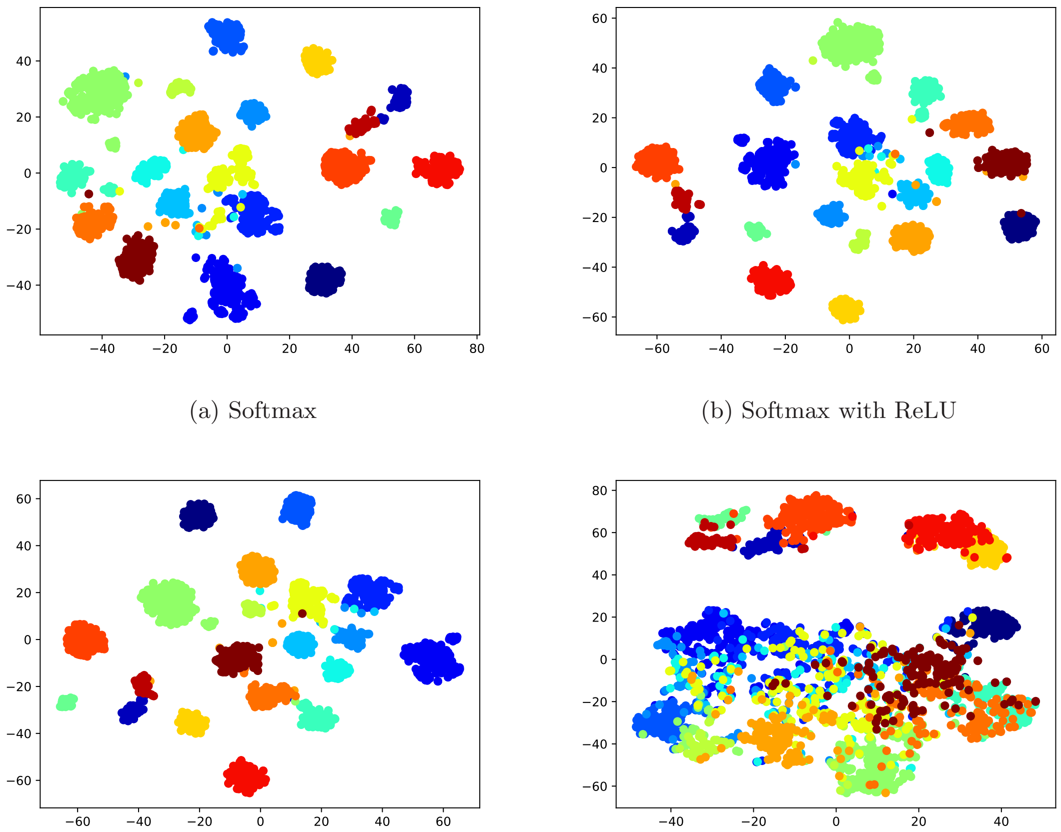
(a) Softmax

(b) Softmax with ReLU

(c) Angular Softmax

(d) Triplet margin L2

Figure 2: DET curves



(a) Softmax

(b) Softmax with ReLU

(c) Angular Softmax

(d) Triplet margin L2

Figure 3: Embeddings of same 20 speakers but produced with 4 different models

References

[1] D. Reynolds and R. Rose, “Robust text-independent speaker identification using gaussian mixture speaker models,” *Speech and Audio Processing, IEEE Transactions on*, vol. 3, pp. 72 – 83, 02 1995.

[2] W. Campbell, D. Sturim, and D. Reynolds, “Support vector machines using gmm supervectors for speaker verification,” *IEEE Signal Processing Letters*, vol. 13, no. 5, pp. 308–311, 2006.

[3] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, “Front-end factor analysis for speaker verification,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788–798, 2011.

[4] D. Snyder, D. Garcia-Romero, D. Povey, and S. Khudanpur, “Deep neural network embeddings for text-independent speaker verification,” pp. 999–1003, 08 2017.

[5] Z. Ren, Z. Chen, and S. Xu, “Triplet based embedding distance and similarity learning for text-independent speaker verification,” in *2019 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pp. 558–562, 2019.

[6] A. Nagrani, J. S. Chung, and A. Zisserman, “Voxceleb: a large-scale speaker identification dataset,” *CoRR*, vol. abs/1706.08612, 2017.

[7] Y. Lukic, C. Vogt, O. Dürr, and T. Stadelmann, “Speaker identification and clustering using convolutional neural networks,” in *2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1–6, 2016.

[8] Y. Yuan, K. Yang, and C. Zhang, “Hard-aware deeply cascaded embedding,” *CoRR*, vol. abs/1611.05720, 2016.

[9] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song, “Sphereface: Deep hypersphere embedding for face recognition,” *CoRR*, vol. abs/1704.08063, 2017.

[10] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, “Arcface: Additive angular margin loss for deep face recognition,” in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4685–4694, 2019.

[11] Y. Li, F. Gao, Z. Ou, and J. Sun, “Angular softmax loss for end-to-end speaker verification,” in *2018 11th International Symposium on Chinese Spoken Language Processing (ISCSLP)*, pp. 190–194, 2018.