REAL-WORLD SPEAKER RECOGNITION ON VOXCELEB2 USING ANGULAR MARGIN LOSSES

Claude Lehmann and Thilo Stadelmann

ZHAW Datalab
Zurich University of Applied Sciences
Winterthur, Switzerland
{lehl, stdm}@zhaw.ch

ABSTRACT

While speaker recognition for studio recordings is considered solved, there is still room for improvement on speech *in the wild*. The addition of background noise and otherwise irrelevant signals poses a more difficult challenge to overcome. Deep neural networks have been acting as a backbone for this application for years now. Our goal is to use a state of the art architecture for studio recordings and scale it up to speech in the wild. In this paper we show the effectiveness of a RNN architecture on the VoxCeleb2 dataset, a now defacto standard benchmark for real-world speaker recognition. Moreover, we use the highly efficient angular margin losses to further improve performance, propose a sweet spot for the segment length and investigate effects of different clustering optimizations. Our model reaches a performance of 5.53 % EER on the official VoxCeleb1 evaluation list.

Index Terms— Deep Learning, Speech Recognition, VoxCeleb2, Angular Margin Loss

1. INTRODUCTION

The field of speaker recognition can be separated in several key areas which fundamentally address tasks around comparing speaker identities for a given audio recording. The most prominent areas are as follows:

- *Speaker verification* decides if a given sample contains a known speaker identity as a binary classification task. When unlocking your phone using your voice, an algorithm performs speaker verification for authentication.
- *Speaker identification* complicates speaker verification by turning it into a multiclass classification problem where the speaker identification is compared against multiple known identities.
- *Speaker clustering* instead identifies and groups unique speaker identities together without prior knowledge about any of the speakers or how many distinct identities are present in the recording. *Speaker diarization* (assigning speaker identities to segments of a recording) is sometimes also referred to as speaker clustering, but usually involves segmenting the recording as well. Compared to both speaker verification and identification, both supervised tasks, speaker clustering is an unsupervised task with a higher complexity, as both the number of clusters (speakers) and cluster affiliation need to be decided.

Speaker recognition is not a new topic, some of the earliest attempts were already made in the 1950's and 1960's with Bell Lab's AUDREY or IBM's Shoebox [1], long before computers became as prevalent and powerful as today. Traditionally, Mel Frequency Cepstral Coefficients (MFCC) [2] have been used as acoustic features for Gaussian Mixture Models (GMM) [3, 4]. Furthermore, so called i-vectors [5] emerged as an abstract representation of a speaker model.

With growing computational power, deep neural networks have also shown success as feature extractors, producing x-vectors [6] or d-vectors [7], similar to the i-vectors. These vector types can be grouped as speaker embeddings. Due to the success on image datasets [8], convolutional neural network (CNN) architectures are dominating the market for deep speaker embeddings [9, 10, 11]. Their success stems from the ability of the CNN kernels to read temporal information. Moreover, the emergence of the residual network (ResNet) architecture [12] further cemented the use of computer vision building blocks for speaker recognition [13, 14, 15].

Other than CNNs, recurrent neural networks (RNNs) such as long-short term memory networks (LSTMs) [16] are directly built to include the temporal aspect in their calculations and are as such also used for speaker recognition tasks, showing great performance [17, 18, 19].

While speaker recognition on studio quality datasets such as TIMIT [20] is mostly solved [21], the same can not yet be said for speech in the wild. However, the emergence of benchmark datasets such as VoxCeleb and VoxCeleb2 [10, 22] has led to steady progress [14, 23, 24] in the field.

While we were unable to exceed current state of the art's [24] equal error rate (EER) of 1.02 %, we achieve an EER of

5.53 % and show the performance of an upscaled LSTM architecture on a real-world dataset using angular margin losses and propose improvements for the clustering step.

1.1. Related Work

The release of the VoxCeleb1 [22] and later VoxCeleb2 [10] datasets allowed speech in the wild to finally have decent benchmark dataset for comparability. Xie et al. [14] managed to define a new state-of-the-art result on the VoxCeleb2 dataset using the angular margin losses combined with a thin ResNet architecture, achieving an equal error rate of **3.22** % on the VoxCeleb1 test dataset. The recent VoxSRC challenge [25] winner [24] in late 2019 further pushed state-of-the-art with reported equal error rates as low as **1.02** %.

ZHAW Deep Voice

Previous works at ZHAW Datalab[1] on the TIMIT dataset have shown great progress both using CNN [26] and LSTM [27] approaches for studio recordings, but need to be scaled up for use in real-world scenarios. Moreover, it was shown that the Pairwise Kullback-Leibler Divergence (PKLD) used as loss function is unsuitable for a growing dataset, such as VoxCeleb2 [28]. Angular Margin Losses, (e.g. CosFace, ArcFace and SphereFace [29, 30, 31]), on the other hand are known from the domain of facial recognition, a domain where large number of classes common. These loss functions were shown to be more suitable for the task of speaker recognition [32] and we will investigate how to apply them in the wild.

2. METHODS

2.1. Angular Margin Losses

After their success in the field of facial recognition [31], Angular Margin Losses quickly became the standard for speaker recognition in real-world settings [33] due to their ability to maximize the angular distance between class centres in high-dimensional space, while keeping class clusters condensed.

These losses are based upon the softmax loss, which is heavily used as a classification loss function, shown here:

$$L_1 = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{W_{y_i}^T x_i + b_{y_i}}}{\sum_{j=1}^{n} e^{W_j^T x_i + b_j}} \tag{1}$$

In Equation 1, N denotes the batch size, n the total number of classes. x_i belong to class y_i and is the calculated feature of the *i*-th sample in a given batch.

Deng et al. [31] state that there exists an issue for tasks where intra-class samples can be very diverse due to the fact that the softmax loss function "*does not explicitly optimize the feature embedding to enforce higher similarity for intra-class samples and diversity for inter-class samples*". This applies to speaker recognition, as voices can take many different shapes ranging from murmurs to shouting and head voice.

[1]`https://www.zhaw.ch/datalab`

Wang et al. [34] propose a fix to this issue transforming $W_j^T x_i + b_j$ by setting the bias b_j to zero and rephrasing the remaining term as $\|W_j\|\|x_i\|\cos(\theta_j)$, where θ_j denotes the angle between the weights and the feature W_j and x_i. By normalizing both the weights and features, they are able to simplify the expression further as $1 = \|W_i\| = \|x_i\|$. The resulting formula $W_j^T x_i = cos(\theta_j)$ builds the foundation for the Angular Margin Losses, where additional margin parameters m are added as a margin m_c to the cosine space for CosFace. For SphereFace and ArcFace, the margins m_s and m_a are added and multiplied in the angular space, respectively. The final resulting formula can be seen as follows:

$$L_2 = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{s(cos(m_s\theta_{y_i}+m_a)-m_c)}}{e^{s(cos(m_s\theta_{y_i}+m_a)-m_c)} + \sum_{j=1,j\neq y_i}^{n} e^{s\cos\theta_j}} \tag{2}$$

Note how the CosFace margin m_c is subtracted from the result of the cosine function, whereas both ArcFace's m_a and SphereFace's m_s act upon the angle provided to the cosine function. Unintuitively, these margins can be combined as seen fit (see Deng et al. [31] for more details).

2.2. Network Architecture

For our model we combine bidirectional LSTM layers with fully connected dense layers, as shown in Figure 1. The bidirectional LSTM cells have a network width of 512 neurons, but due to the bidirectional nature the output is always twice its size, i.e. 1'024. The dense layers match this dimensionality and shrink down for the second dense layer to 512 neurons, acting as a bottleneck layer (see [30, 31] for more details). The final dense layer has a width equal to the number of classes used, in our case 5'994. Dropout layers help reduce the overfitting problem, as L_1 and L_2 regularization cannot be used in conjunction with LSTMs. Bengio et al. [35] state that "*this prevents the model to learn generator networks, nor can it exhibit long term memory traces*". We could confirm this statement during our own training optimizations.

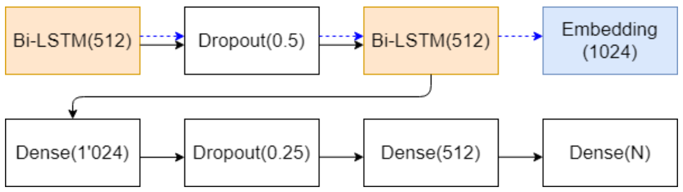

Fig. 1. Bidirectional LSTM network architecture showing both training (black) and inference (blue, dotted) data paths.

2.3. Proxy Learning Task

Ideally, we want our model to learn to model *any* (unseen) speaker into a generalized embedding. During training however, all the speakers are known and part of our dataset. As such, our goal is to train the network on a proxy task that will allow the intermediate layers to form a generalized representation of the speaker model (our speaker embedding). To achieve this, the model is trained on speaker identification using the training dataset. At inference, the dense layers are removed and we extract our speaker embedding directly from the output of the second bidirectional LSTM layer (see blue arrows in Figure 1). This allows us to use the dataset to model all known speakers through the LSTM layers and classify it through the dense layers, but keep the LSTM structure as a generalized speaker embedding feature extractor, generating a speaker embedding that uses a highly generalized representation of the speaker's prosodic features.

2.4. Linkage Criteria Intuition

In speaker clustering, observations representing utterances are to be grouped according to their speaker identity. Hierarchical clustering uses the linkage criterion to measure the proximity between observations and decide, which observations are merged into a cluster. The choice of this criterion is crucial, since the proximity can vary wildly for different criteria. In the following paragraph, we will give a brief overview of linkage criteria and their intuition. We consider the choice whether a cluster C_a should be added to C_b, where both C_a and C_b can be single observations or clusters of multiple observations. Observations are denoted as x.

Single Linkage, also referred to as *nearest neighbor*, defines the proximity as the minimum distance between any of the observations x in clusters C_a and C_b. As such, it is prone to building chain-like clusters, only considering the local surroundings.

Complete Linkage is the opposite of single linkage, comparing the distance between the two observations that are furthest apart in C_a and C_b. This leads to globally compact clusters that are forced to generate spherical clusters and at the same time avoids building elongated chain clusters.

Average Linkage (or UPGMA) uses the distance $d(x_i, x_j)$ between all observations $x_i \in C_a$ and $x_j \in C_b$ and takes the average. This produces clusters to be connected with compactness in mind, then the lowest distance can be scored if all $d(x_i, x_j)$ are minimized. With this criterion, clusters with small variance are joined with priority and outliers affect the distance less (especially compared to single and complete linkages).

Weighted Average (or WPGMA) is remarkably close to average linkage and only differs as each observation x in both clusters C_a and C_b is summed equally, compared to average linkage where the average can be shifted through the amount of observations at each clustering attempt.

Group Median (or UPGMC) and Weighted Centroid (or WPGMC) behave very similarly to average linkage, again using weighted and unweighted variants. These centroid based criteria behave very similar to K-means, but introduce so called inversions, where the centroid positional change after merging can decrease the minimum distance to remaining clusters in time step t_n below the minimum distance of time step t_{n-1}.

Ward's method is an outlier in this list. Instead of measuring the proximity between clusters, Ward's method calculates the changes in variance for each potential cluster merge and chooses the pairing which minimizes the total deviation after a merge.

3. EXPERIMENTS

3.1. Datasets

Our models were trained using the VoxCeleb2 [10] dataset, using the 'dev' part for training and the 'test' part to report model performance. The VoxCeleb1 [22] dataset was then used for final evaluations (see Section 4.1 for more details). Table 1 shows a comparison of the two datasets below:

	VoxCeleb1 (Dev and Test)	VoxCeleb2 (Dev)
Unique speakers	1'251	5'994
# utterances	153'516	1'092'009
Avg. # / speaker	116	185
Utterance length	4.0 / 8.2 / 145.0	4.0 / 7.8 / 220.2

Table 1. Comparison of VoxCeleb datasets 1 and 2. Utterance length denotes min, mean and max length, respectively

3.2. Training Loss

All our models were trained using the Angular Margin Losses outlined in Section 2.1, with a focus on the CosFace [29] and ArcFace [31] variants. We settled at values: $m_{cosface} = 0.0001$, $m_{arcface} = 0.05$ and $s = 30$. The original PKLD loss function used in [27] was not further investigated, as the Angular Margin Loss directly outperformed it by 7 EER percentages. Furthermore, the PKLD calculations were slower by over an order of magnitude.

3.3. Training Details

The audio files are converted to Mel Spectrograms with 128 frequency bins using a sample rate of 16'000 and a window length of 1'024. All audio files are pre-processed to their respective spectrogram to speed up training performance. Before storing them in `h5py` format, dynamic range compression is performed as proposed by Dieleman et al. [36]. As the network expects a constant width spectrogram, a fixed-length spectrogram segment is extracted with a randomized offset during training. In case of utterances that are shorter than the segment length used, the utterance is looped. This segment is then normalized using the mean and standard deviation of the full spectrogram. Intuitively, looping an utterance acts as if a speaker is repeating what he just said, which can occur naturally. We do not remove silence of any kind and apply no voice activity detection. We use a batch size of 256 in conjunction with the Adam optimizer [37], using default standard parameters with a learning rate of 0.0003. The model was trained for a maximum of 32 epochs, with early stopping terminating after a lack of improvement over 3 epochs. Most models converged between 20 and 25 epochs.

The code for our experiments can be found online[2] and is based upon the original ZHAW Deep Voice repository[3].

4. RESULTS

In the following section we show the performance of our LSTM architecture using Angular Margin Losses. We compare our results against current state-of-the-art and take a closer look at the performance of varying clustering techniques to further improve upon the results. Moreover, we propose potential applications for gender identification.

4.1. Evaluation on VoxCeleb1

The performance of our models is measured on the VoxCeleb1 [22] evaluation lists provided online[4]. The `original` evaluation list is based upon the 'test' part of the VoxCeleb1 dataset, including just 40 unique speakers. Because the two VoxCeleb datasets are completely disjunct and we train only using VoxCeleb2, two additional evaluation lists can be used. The `extended` list includes all speakers from both the training and test split of the VoxCeleb1 dataset. Moreover, the `hard` evaluation list was created using pairs of the same gender and nationality, further increasing the difficulty.

During evaluation, random fixed-length segment of the evaluation utterances are extracted and fed into our partial model (see Figure 1). The resulting speaker embeddings are then compared using cosine similarity. We also experiment with the dot product as a distance measure, as well as multiple clustering algorithms such as Agglomerative Hierarchical Clustering (AHC) and Dominant Sets [21] (see Section 4.4).

Table 2 shows a comparison of current state of the art systems against our model using the VoxCeleb1 evaluation lists. Note that shortly after their release cleaned versions of the lists were published, removing duplicates from the original, extended and hard lists. Results followed by a † symbol denote results that were achieved using the non-cleaned versions of the respective evaluation lists. Noteworthy are the results from the VoxSRC challenge [25] taking place in late 2019, which drastically improved the performance on the VoxCeleb2 dataset from the 3 % EER region down to as low as 1.02 % EER, as shown by Zhou et al. [38], Garcia-Romero et al. [39] and Zeinali et al. [24]. Even the hard evaluation list can now be mostly considered solved, with an EER of 2.12 %. The architectures of the VoxSRC winners shows a focus on much larger models (ResNet-150 and 256 instead of ResNet-34 and 50), the addition of time delay neural networks (TDNN) and the heavy use of ensembles to further increase model performance. The overall winning model by Zeinali et al. [24] combines all these aspects.

4.2. Network Complexity

The ZHAW Deep Voice bidirectional LSTM network used on the TIMIT dataset produced state-of-the-art results on studio recordings, but clearly the model complexity had to be increased to fit the additional data complexity of a speech in the wild dataset. As such, we explore different layer widths for the bidirectional LSTM blocks. The pre-existing classification dense layers were already expanded to fit this larger classification task with 5'994 classes instead of the 100 and 470 classes of the TIMIT dataset respectively. This increase in dense layer width is already represented in the network architecture depicted in Figure 1. For reference we show the performance of an unchanged LSTM capacity and double it until performance decreases. As shown in Figure 2[5], the EER can be improved from 33.8 % down to 26.1 % by using a layer width of 512 instead of 256. Further doubling the width to 1'024 decreases the performance slightly to 26.9 % EER. These measurement suggest that the ideal LSTM width lies between 1'024 and 256. We decided to go further using the $LSTM_{512}$ architecture, without conducting more fine-grained experiments. As an added benefit, the $LSTM_{1024}$ architecture is be much more prone to overfit due to the number of trainable parameters. While overfitting remains an issue, using a network half the size helps reduce this risk.

[2]`https://github.com/edualc/ZHAW_deep_voice`

[3]`https://github.com/stdm/ZHAW_deep_voice`

[4]`http://www.robots.ox.ac.uk/~vgg/data/voxceleb/vox2.html`

[5]Note that the network architectures are denoted as LSTM followed by the width of both bidirectional LSTM layers, e.g. $LSTM_{512}$ has network width of 512 neurons.

	Model Architecture	Loss	Dims	Traning Dataset	EER (%)		
					Vox1-O	Vox1-E	Vox1-H
Nagrani et al. [22]	I-vectors + PLDA	-	-	VoxCeleb1	8.80 †	-	-
Nagrani et al. [22]	CNN + Embedding	-	-	VoxCeleb1	7.80 †	-	-
Chung et al. [10]	ResNet-34	Softmax + Contrastive	512	VoxCeleb2	5.04 †	-	-
Chung et al. [10]	ResNet-50	Softmax + Contrastive	512	VoxCeleb2	4.19 †	-	-
Xie et al. [14]	Thin ResNet-34	AM-Softmax	512	VoxCeleb2	3.23 †	4.42 †	7.33 †
Xie et al. [14]	Thin ResNet-34	Softmax	512	VoxCeleb2	3.24	3.13	5.06
Cai et al. [40]	ResNet-34	Softmax	128	VoxCeleb2	2.00	-	-
Zhou et al. [38]	ResNet	Softmax	128	VoxCeleb2	1.85	1.70	3.13
Zhou et al. [38]	ResNet	A-Softmax	128	VoxCeleb2	4.43	4.05	8.46
Zhou et al. [38]	ResNet	L_2-Softmax	128	VoxCeleb2	1.81	1.68	3.12
Garcia-Romero et al. [39]	TDNN + PLDA	Softmax	512	VoxCeleb2	-	1.93	-
Garcia-Romero et al. [39]	TDNN	AM-Softmax	256	VoxCeleb2	-	1.61	-
Garcia-Romero et al. [39]	Ensemble	-	-	VoxCeleb2	-	1.22	-
Zeinali et al. [24]	ResNet-256	Softmax + AAM	256	VoxCeleb2	1.42	1.35	2.48
Zeinali et al. [24]	ResNet-160	Softmax + AAM	160	VoxCeleb2	1.31	1.38	2.50
Zeinali et al. [24]	TDNN	Softmax + AAM	512	VoxCeleb2	1.46	1.57	2.89
Zeinali et al. [24]	TDNN	Softmax	512	VoxCeleb2	1.94	2.03	3.97
Zeinali et al. [24]	Ensemble	-	-	VoxCeleb2	**1.02**	**1.14**	**2.12**
Ours	BiLSTM	AAM	1024	VoxCeleb2	5.53	5.84	9.49

Table 2. Results of our bidirectional LSTM system compared against multiple state of the art systems, including winners of the VoxSRC challenge [25]. The listed EER results are evaluated using the VoxCeleb1 evaluation lists `Vox1-O`, `Vox1-E` and `Vox1-H`, denoting original, extended and hard respectively. † indicates a result on the non-cleaned evaluation list.

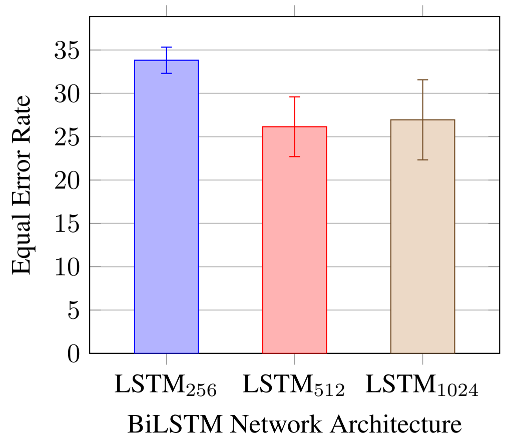

Fig. 2. Network complexity comparison using the bidirectional LSTM approach with varying network width.

4.3. Segment Length

Stadelmann et al. [27] showed a sweet spot for the segment length on the TIMIT dataset, which is comprised of studio recordings. For the VoxCeleb2 dataset, this experiment had to be repeated under *speech in the wild* conditions. Furthermore, Xie et al. [14] compared different segment lengths for evaluation and concluded that longer segment lengths deliver superior performance, which can be explained since more information is given to the model. Our next set of experiments takes this idea and applies it to our $LSTM_{512}$ model by examining the performance at various segment length thresholds. The LSTM architecture proposed by Stadelmann et al. [27] used a segment length of 400ms on studio quality recordings, while Xie et al. [14] experimented with segment lengths up to 6 seconds on VoxCeleb2. In our experiment we sample between 400ms and 5 seconds, as any longer segments overburdened our available resources.

Figure 3 shows the performance at varying segment lengths. During experiments we used both cosine similarity and the dot product as a distance metric to compare speaker embeddings. This figure clearly shows a downwards trend towards longer segment lengths, with most configurations at or above 2 second segment length achieving an EER of 10 % or below. Due to the nature of the angular margin loss function it is to be expected that the cosine similarity performs better than the dot product for a loss function that maximizes intra-class angles. Moreover, as segment lengths increase the additional performance gain is diminished and the EER for 4 and 5 seconds is almost identical (7.770 % and 7.787 % respectively) using the cosine similarity. We believe to have found a sweet spot for segment length at 4 seconds.

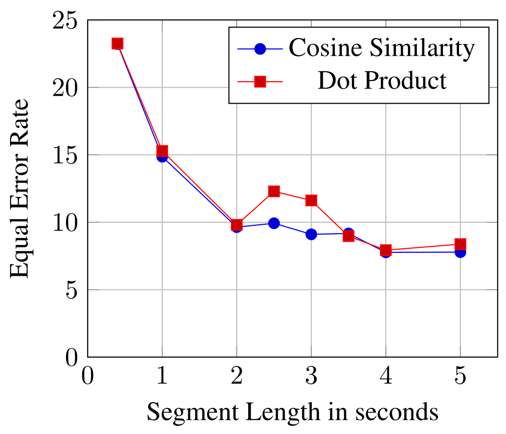

Fig. 3. Equal error rate as a function of the input segment length on the VoxCeleb1 cleaned evaluation list, using the cosine similarity and the dot product as a distance metric.

4.4. Speaker Clustering Optimizations

When comparing if two utterances are from the same speaker, currently we generate two speaker embeddings and measure the distance using the cosine similarity or another distance metric. What if instead we apply clustering algorithms such as agglomerative hierarchical clustering (AHC)? The idea we propose is to extract overlapping spectrograms from an utterance and create multiple embeddings per utterance. The spectrograms are overlapping by 50%, experiments with 75% overlap showed almost identical results. When we now compare two utterances, instead of measuring a distance between two points in the embedding space, we are able to apply clustering algorithms as a different distance metric, for example the distance between the biggest two clusters. Furthermore, as clustering does not require a direct domain knowledge and acts unsupervised, the choice of algorithm is not evident.

In this next set of experiments we focus on AHC and Dominant Sets introduced by Hibraj et al. [21]. For comparison we will keep both the cosine similarity and dot product, although applied to the mean of the overlapping spectrograms. For AHC we measure the distance between the last two clusters that are combined, which is equivalent to the maximum height in a dendrogram. The intuition behind this choice is, that for a comparison between two different speakers this distance is expected to be high while for the same speaker we expect it to be low. We further experimented with different secondary and tertiary cluster distances beyond the last two clusters without success.

The approach using Dominant Sets generated dominant sets to reduce the utterance spectrogram clusters to a more refined set of embedding vectors such that for every dominant set a new utterance is made by creating a weighted average of the embeddings in each dominant set cluster. These new embeddings are then fed into the same AHC methods used above, as well as the dot product and cosine similarity.

For comparison with an established speaker recognition system, we perform these experiments not only on our $LSTM_{512}$ model but also the Thin ResNet-34 model proposed by Xie et al. [14]. Figures 4 and 5 show the performance of these clustering approaches on our $LSTM_{512}$ and the Thin ResNet-34 model using the best performing clustering approaches among AHC and Dominant Sets (DS). The Dominant set variants perform very poorly, with an EER of 6.42 % and above for the $LSTM_{512}$ and 3.62 % and above for the Thin ResNet-34 model. On the $LSTM_{512}$ model, we could improve the performance from 5.84 % (cosine similarity) to 5.54 % EER using AHC with average linkage, while extracting just one fixed-length segment during evaluation yielded an EER of 7.52 % on the LSTM model. This improvement through clustering could not be recreated on the Thin ResNet-34 architecture, where the dot product remained the best performing similarity measure. We are unsure why the cosine similarity is outperformed by the dot product for this model, as the additive softmax should in theory behave better using a distance metric that focuses on the angle between cluster centers. This anomaly should be further investigated, but we are unable to follow up at this moment due to time constraints.

Since we do not create just one speaker embedding per utterance, comparisons are based upon a plethora of utterances where outliers exists due to small disturbances in the audio recording (e.g. background noise). These disturbances seem to be crucially responsible for model performance, as both the LSTM and ResNet models perform best on centroid- and average-based linkage criteria using AHC (see Figures 4 and 5). Both these criteria are much less affected by outliers compared to single or complete linkage, which points at a significant presence of outliers in these embeddings. These outliers explain the performance drop for complete linkage with AHC (6.60 % EER) against centroid and average linkages (5.66 % and 5.54 % respectively). However, through the use of dominant sets this can be improved to 6.42 %. As we discussed criteria intuitions in Section 2.4, complete linkage aims to produce spherical clusters. In the embedding space of the whole VoxCeleb datasets, embeddings of the same utterance are expected to be compact clusters with few outliers (where short disruptions in a recording skew embeddings away from the mean). Dominant sets are able to filter out this effect and work hand in hand with the complete linkage criterion. Note how for the ResNet model both complete linkage and Ward's method without dominant sets perform the worst, compared to other linkage criteria. This seems to be caused by the arrangement of embeddings in the embedding space. Since clusters appear to be spread out, both criteria struggle to form reasonable clusters. A more conclusive table of results is available in the Appendix in Section A.1.

An issue of these clustering algorithms is the quality of the speaker embeddings itself. When looking at the dendro-

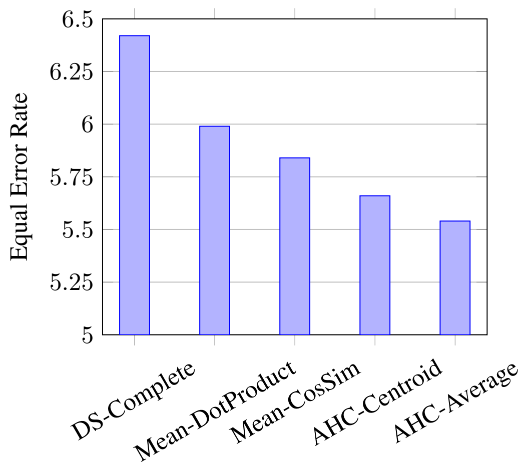

Fig. 4. Comparison of clustering techniques against our best $LSTM_{512}$ model.

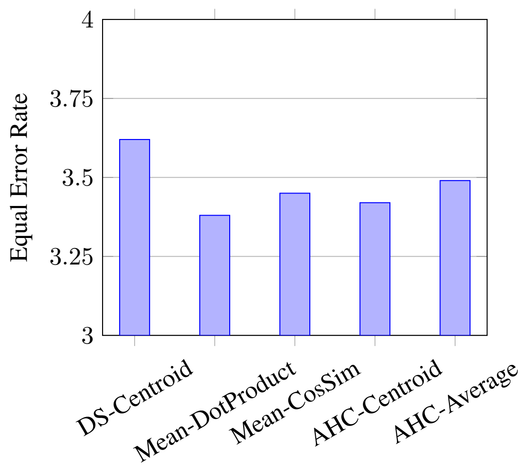

Fig. 5. Comparison of clustering techniques against the Thin ResNet-34 model proposed by Xie et al. [14].

grams of such speaker embeddings we could identify about 2% of comparisons in the VoxCeleb1 original evaluation list where the comparison between the same speaker and two different speakers look almost identical. Figure 6 shows two such examples where it is impossible to determine if the two utterances are by the same speaker or not.

4.5. Gender Identification Potential

Visualizing the speaker embeddings generated by our models we realized that our $LSTM_{512}$ model can distinguish between male and female voices remarkably well. This is characterized by the formation of two distinct clusters for our $LSTM_{512}$ model, while the Thin ResNet-34 model clusters the genders much more closely. Figure 7 shows the PCA plots for both models using the first two components. Note

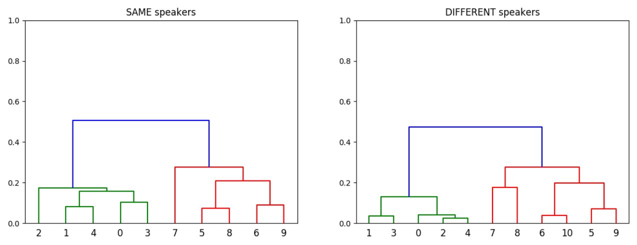

Fig. 6. Two dendrograms from comparisons of the same (left) and different (right) speakers using overlapping spectrograms to generate multiple embeddings.

that the embeddings for the LSTM model are spread between $x, y \in [-2, 4]$, while the Thin ResNet-34 creates embeddings that are clumped much tighter with $x, y \in [-0.4, 0.4]$. We believe that this shows a preference for gender when identifying a given voice from our $LSTM_{512}$ model. This is much less present for the Thin ResNet-34 model. We believe while our model was unable to surpass the performance of current state of the art, it might see use for gender identification purposes. Additional t-SNE plots and enlarged versions of the PCA plots are included in the appendix (see Section A.3).

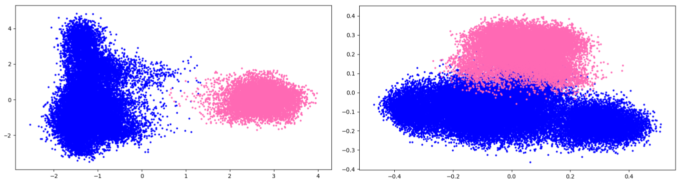

Fig. 7. PCA plot of speaker embeddings from the VoxCeleb1 Original evaluation list generated by the $LSTM_{512}$ (left) and Thin ResNet-34 (right) models, colored by gender.

Figure 7 shows how the features of the LSTM and ResNet models are very different; the LSTM model clearly separates gender and uses it heavily to identify the speaker model for given utterances. The same LSTM architecture was able to solve the task of speaker recognition on the clean studio audio recordings [27] using comparable embeddings, generating concrete features to perform *optimally under optimal audio conditions*. The ResNet model instead captures noise and additional data complexity better, but at a cost of interpretability.

5. CONCLUSION

In this work we introduced an upscaled version of an LSTM architecture that is able to use the VoxCeleb2 dataset for speaker recognition. We also introduce a clustering improvement step for the LSTM architecture that might be applicable to different architectures as well. The network is not able to achieve state of the art performance, but performs similarly to state of the art systems that are from early 2019. We further showed how the segment length affects the performance and that a longer length in general is desirable.

5.1. Future Work

We propose to investigate larger networks such as used by the VoxSRC winners, as well as the use of ensembles to further refine the performance. Furthermore, while the clustering steps might not be applicable to all network architectures, we see a lot of potential still for improvements in this segment. It could even be useful to train a different network on speaker embeddings that replaces the distance metrics and instead learns the best comparisons for a model's embeddings.

The usefulness of ensembles cannot be ignored. Another idea would be to train some models on pure male and female split datasets, where models would be able to focus more on differences within gendered voices. We believe it could be beneficial to create an ensemble using a trio of the same model trained on three different datasets: the original dataset, a male-only and a female-only version.

The surprising separability for the $LSTM_{512}$ to be separated by gender leads us to believe that there might be more information hidden in the speaker embeddings. With official labels of the nationality available as well as the gender, further experiments in this direction could uncover interesting results perhaps even around the interpretability of these embeddings.

During training we observed a certain degree of overfitting and variance. Data augmentation could be useful in the sense of adding random noise, as included in the Kaldi toolkit by Povey et al. [41]. This would allow a network to see the VoxCeleb2 dataset with a richer spectrum of background noise, reverberations, et cetera while reducing variance.

6. REFERENCES

[1] Richard S Glantz, "Shoebox: a personal file handling system for textual data," in *Proceedings of the November 17-19, 1970, fall joint computer conference*, 1970, pp. 535–545.

[2] Steven Davis and Paul Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE transactions on acoustics, speech, and signal processing*, vol. 28, no. 4, pp. 357–366, 1980.

[3] Douglas A Reynolds and Richard C Rose, "Robust text-independent speaker identification using gaussian mixture speaker models," *IEEE transactions on speech and audio processing*, vol. 3, no. 1, pp. 72–83, 1995.

[4] Douglas A Reynolds, Thomas F Quatieri, and Robert B Dunn, "Speaker verification using adapted gaussian mixture models," *Digital signal processing*, vol. 10, no. 1-3, pp. 19–41, 2000.

[5] Najim Dehak, Patrick J Kenny, Réda Dehak, Pierre Dumouchel, and Pierre Ouellet, "Front-end factor analysis for speaker verification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788–798, 2010.

[6] David Snyder, Daniel Garcia-Romero, Gregory Sell, Daniel Povey, and Sanjeev Khudanpur, "X-vectors: Robust dnn embeddings for speaker recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5329–5333.

[7] Oren Barkan and Hagai Aronowitz, "Diffusion maps for plda-based speaker verification," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013, pp. 7639–7643.

[8] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[9] Yann LeCun, Yoshua Bengio, et al., "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, vol. 3361, no. 10, pp. 1995, 1995.

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

[11] Sarthak Yadav and Atul Rai, "Learning discriminative features for speaker identification and verification.," in *Interspeech*, 2018, pp. 2237–2241.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[13] Amirhossein Hajavi and Ali Etemad, "A deep neural network for short-segment speaker recognition," *arXiv preprint arXiv:1907.10420*, 2019.

[14] 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.

[15] Ya-Qi Yu, Lei Fan, and Wu-Jun Li, "Ensemble additive margin softmax for speaker verification," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 6046–6050.

[16] Jürgen Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85–117, 2015.

[17] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 369–376.

[18] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE, 2013, pp. 6645–6649.

[19] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent, "Audio chord recognition with recurrent neural networks.," in *ISMIR*. Citeseer, 2013, pp. 335–340.

[20] 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*, vol. 93, 1993.

[21] Feliks Hibraj, Sebastiano Vascon, Thilo Stadelmann, and Marcello Pelillo, "Speaker clustering using dominant sets," in *2018 24th International Conference on Pattern Recognition (ICPR)*. IEEE, 2018, pp. 3549–3554.

[22] Arsha Nagrani, Joon Son Chung, and Andrew Zisserman, "Voxceleb: a large-scale speaker identification dataset," *arXiv preprint arXiv:1706.08612*, 2017.

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

[24] 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.

[25] Joon Son Chung, Arsha Nagrani, Ernesto Coto, Weidi Xie, Mitchell McLaren, Douglas A Reynolds, and Andrew Zisserman, "Voxsrc 2019: The first voxceleb speaker recognition challenge," *arXiv preprint arXiv:1912.02522*, 2019.

[26] Yanick Lukic, Carlo Vogt, Oliver Dürr, and Thilo Stadelmann, "Speaker identification and clustering using convolutional neural networks," in *2016 IEEE 26th international workshop on machine learning for signal processing (MLSP)*. IEEE, 2016, pp. 1–6.

[27] Thilo Stadelmann, Sebastian Glinski-Haefeli, Patrick Gerber, and Oliver Dürr, "Capturing suprasegmental features of a voice with rnns for improved speaker clustering," in *IAPR Workshop on Artificial Neural Networks in Pattern Recognition*. Springer, 2018, pp. 333–345.

[28] Jan Sonderegger, Patrick Walter, and Thilo Stadelmann, "Benchmarking of Classical and Deep Learning Speaker Clustering Approaches," 2019.

[29] Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Dihong Gong, Jingchao Zhou, Zhifeng Li, and Wei Liu, "Cosface: Large margin cosine loss for deep face recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5265–5274.

[30] Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song, "Sphereface: Deep hypersphere embedding for face recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 212–220.

[31] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou, "Arcface: Additive angular margin loss for deep face recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4690–4699.

[32] Daniel Neururer and Thilo Stadelmann, "AngleVoice: Leveraging Angular Margin Losses for Real World Speaker Recognition, Clustering and Diarization," 2019.

[33] Mahdi Hajibabaei and Dengxin Dai, "Unified hypersphere embedding for speaker recognition," *arXiv preprint arXiv:1807.08312*, 2018.

[34] Feng Wang, Jian Cheng, Weiyang Liu, and Haijun Liu, "Additive margin softmax for face verification," *IEEE Signal Processing Letters*, vol. 25, no. 7, pp. 926–930, 2018.

[35] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio, "On the difficulty of training recurrent neural networks," in *International conference on machine learning*, 2013, pp. 1310–1318.

[36] Sander Dieleman and Benjamin Schrauwen, "End-to-end learning for music audio," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 6964–6968.

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

[38] Tianyan Zhou, yong zhao, Jinyu Li, Yifan Gong, and Jian Wu, "Cnn with phonetic attention for text-independent speaker verification," in *Automatic Speech Recognition and Understanding Workshop*. IEEE, December 2019.

[39] Daniel Garcia-Romero, David Snyder, Gregory Sell, Alan McCree, Daniel Povey, and Sanjeev Khudanpur, "X-vector dnn refinement with full-length recordings for speaker recognition," in *Proc. Interspeech*, 2019, pp. 1493–1496.

[40] Danwei Cai, Xiaoyi Qin, Weicheng Cai, and Ming Li, "The dku system for the speaker recognition task of the 2019 voices from a distance challenge," *arXiv preprint arXiv:1907.02194*, 2019.

[41] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, et al., "The kaldi speech recognition toolkit," in *IEEE 2011 workshop on automatic speech recognition and understanding*. IEEE Signal Processing Society, 2011, number CONF.

[42] Christian Lauener, Claude Lehmann, and Thilo Stadelmann, *Speaker Clustering for Real-World Data using Deep Learning*, Bachelor thesis, 2019.

[43] Yanyao Shen, Hyokun Yun, Zachary C Lipton, Yakov Kronrod, and Animashree Anandkumar, "Deep active learning for named entity recognition," *arXiv preprint arXiv:1707.05928*, 2017.

A. APPENDIX

A.1. Speaker Clustering Results

The following Tables 3 and 4 show a detailed overview of the clustering results on the $LSTM_{512}$ and Thin ResNet-34 models discussed in Section 4.4. Dominant Set results are denotes with the prefix `DS-`, agglomerative hierarchical clustering results with the prefix `AHC-`. Due to poor performance, the results using dominant sets were only performed on the VoxCeleb1-Original list.

	$LSTM_{512}$		
	Vox1	Vox1-E	Vox1-H
Mean-DotProduct	5.99	6.45	11.19
Mean-CosSim	5.84	6.09	9.97
AHC-Average	**5.54**	**5.84**	**9.49**
AHC-Centroid	5.66	5.85	9.56
AHC-Complete	6.60	6.81	11.06
AHC-Median	5.74	6.02	9.81
AHC-Single	6.17	6.36	10.29
AHC-Ward	5.81	6.01	9.77
AHC-WeightedAvg	5.61	5.97	9.74
DS-Average	6.73	-	-
DS-Centroid	6.48	-	-
DS-Complete	6.42	-	-
DS-Median	7.62	-	-
DS-Single	16.33	-	-
DS-Ward	8.17	-	-
DS-WeightedAvg	8.23	-	-

Table 3. Results of the $LSTM_{512}$ model using different clustering approaches for the comparison distance metric.

A.2. Active Learning

We explore an Active Learning (AL) approach, initially proposed in a bachelor thesis [42] to reduce the total amount of data shown to a network and in return decrease training times while still reaching similar results. AL is usually used when a dataset is not fully labelled and the network can be used to determine which subsets of unlabelled data is most useful for further training. Shen et al. [43] reached 99% performance compared to current state of the art system in named entity recognition while only actively using up to 25% of training data. Our idea is to only train on a small subset of the VoxCeleb2 dataset and act as if the remainder is unlabelled and thus unusable data. During multiple iterations the remainder would be passed through the network and compared against the desired output. The n results that exhibit the furthest distance from the desired output are chosen to be added to the

	$ThinResNet-34$		
	Vox1	Vox1-E	Vox1-H
Mean-DotProduct	**3.38**	**3.46**	**5.47**
Mean-CosSim	3.45	3.57	5.67
AHC-Average	3.49	3.60	5.70
AHC-Centroid	3.42	3.55	5.63
AHC-Complete	5.31	5.53	8.38
AHC-Median	3.75	3.88	6.16
AHC-Single	4.25	4.39	6.86
AHC-Ward	5.65	5.61	8.17
AHC-WeightedAvg	3.78	3.92	6.23
DS-Average	6.58	-	-
DS-Centroid	6.32	-	-
DS-Complete	6.37	-	-
DS-Median	7.15	-	-
DS-Single	12.74	-	-
DS-Ward	8.63	-	-
DS-WeightedAvg	7.50	-	-

Table 4. Results of the Thin ResNet-34 model using different clustering approaches for the comparison distance metric.

dataset. This process is repeated until convergence. In our experiments we set $n = 2^{13} = 8'192$ and performed the uncertainty sampling after every epoch. The initial dataset was comprised of 10% of each speaker's utterances and due to rounding equates to about 17% of the VoxCeleb2 'dev' dataset. This ensured that all classes are available in the initial dataset. Models were trained up to 32 epochs with early stopping after 3 epochs of no improvement. The baseline model without AL reached an EER of 30.3, with AL enabled 32.9.

As shown by this experiment, the AL approach does not bring an impressive advantage and instead performs as we would expect: If we are only using a smaller subset of the full dataset, the model is not able to reach the same performance as when using the full dataset. Moreover, training times were similar, while the uncertainty sampling added an additional step that prolonged total training time further instead of reducing it. While this approach can be useful for a hardware limited system, enabling multiprocessing to our models renders this approach obsolete. Furthermore, the act of adding more data during training on a performance basis introduces a shifting class balance throughout training, which might interfere with model performance.

A.3. Speaker Embedding Visualizations

As discussed in Section 4.5, our LSTM model is able to better separate genders than the Thin ResNet-34 model. While this needs further investigating, the PCA and t-SNE plots are added below. Each dot represents a speaker embedding that was extracted from a spectrogram. Spectrograms were extracted with 50% overlap and multiple utterances were used per speaker, meaning there are multiple dots *per speaker and utterance*.

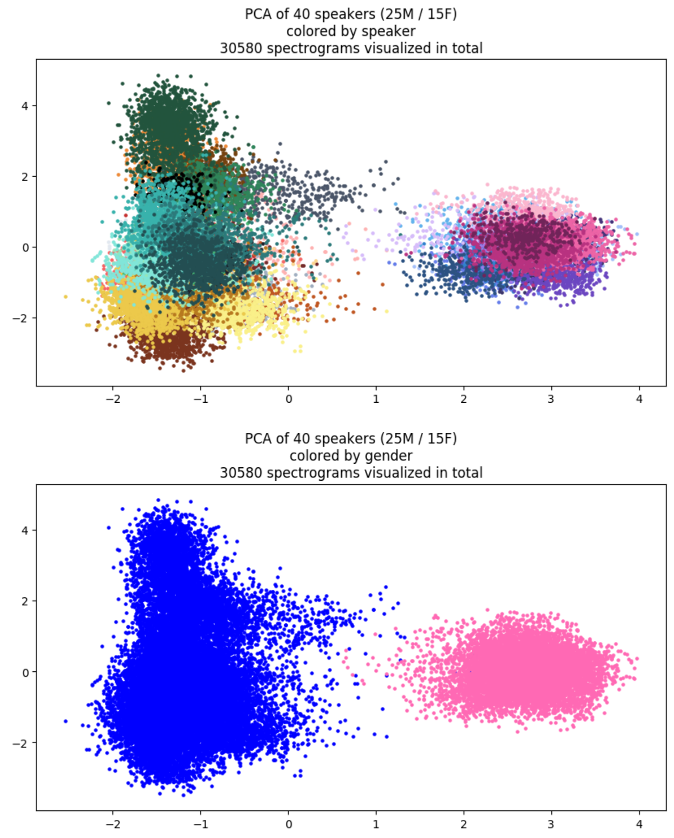

Fig. 8. PCA plot of the $LSTM_{512}$ network's speaker embeddings on the original VoxCeleb1 evaluation list. Each dot denotes a speaker embedding. Embeddings are colored by their speaker (top) and gender (bottom).

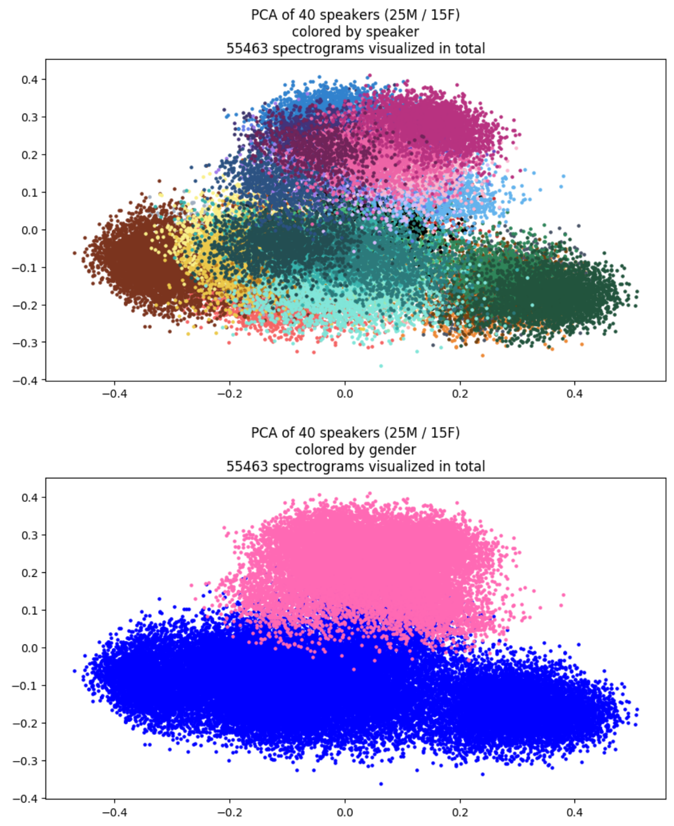

Fig. 9. PCA plot of the Thin ResNet-34 network's speaker embeddings on the original VoxCeleb1 evaluation list. Each dot denotes a speaker embedding. Embeddings are colored by their speaker (top) and gender (bottom).

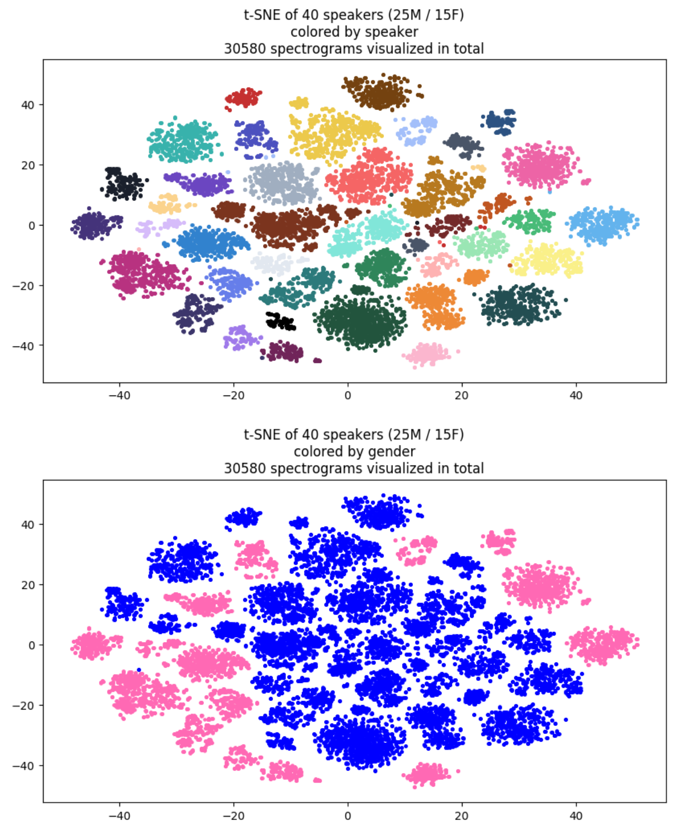

Fig. 10. t-SNE plot of the $LSTM_{512}$ network's speaker embeddings on the original VoxCeleb1 evaluation list. Each dot denotes a speaker embedding. Embeddings are colored by their speaker (top) and gender (bottom).

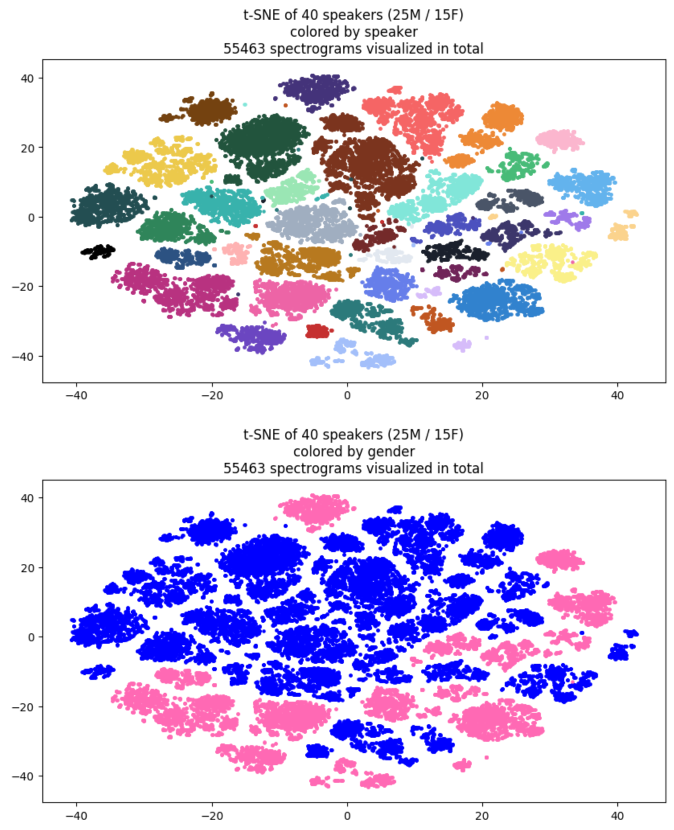

Fig. 11. t-SNE plot of the Thin ResNet-34 network's speaker embeddings on the original VoxCeleb1 evaluation list. Each dot denotes a speaker embedding. Embeddings are colored by their speaker (top) and gender (bottom).