Zurich University
of Applied Sciences

School of
Engineering

InIT Institute of Applied
Information Technology

Project Work

Speaker Recognition & Diarization on Realistic Data

Author	Flurin Gishamer Philippe Hürlimann
Main supervisor	Thilo Stadelmann

Date	21.12.2018

Zurich University
of Applied Sciences

School of
Engineering

DECLARATION OF ORIGINALITY

Project Work at the School of Engineering

By submitting this project work, the undersigned student confirm that this work is his/her own work and was written without the help of a third party. (Group works: the performance of the other group members are not considered as third party).

The student declares that all sources in the text (including Internet pages) and appendices have been correctly disclosed. This means that there has been no plagiarism, i.e. no sections of the project work have been partially or wholly taken from other texts and represented as the student's own work or included without being correctly referenced.

Any misconduct will be dealt with according to paragraphs 39 and 40 of the General Academic Regulations for Bachelor's and Master's Degree courses at the Zurich University of Applied Sciences (Rahmenprüfungsordnung ZHAW (RPO)) and subject to the provisions for disciplinary action stipulated in the University regulations.

City, Date: Signature:

....

.....

.....

The original signed and dated document (no copies) must be included after the title sheet in the ZHAW version of all project works submitted.

Zusammenfassung

In unserer Projektarbeit haben wir versucht auf anderen an der ZHAW entwickelten Systemen der Speaker Diarization aufzubauen und zu verbessern. Die Speaker Diarization beantwortet die Frage: «Wer sprach wann?». Man sollte in der Lage sein einem Speaker Diarization System eine annotationsfreie Audiospur zu geben und das System gibt einem zurück; wie viele Personen auf der Aufnahme sprechen, wann eine Person spricht und wie lange sie das tat. Man könnte z.B. eines Tages in der Lage sein, während eine Besprechung mit dem Smartphone aufzunehmen, um anschliessend ein vollautomatisch generiertes und mit korrekten Sprechern annotiertes Protokoll zu erhalten. Solche Systeme wurden früher meistens mit klassischen statistischen Methoden, wie z.B. «Hidden Markov Models» gemacht. In jüngster Zeit haben auf Deep Learning basierende Systeme an grosser Popularität gewonnen und auch in diesem Bereich bereits Ergebnisse erzielen können. In dieser Projektarbeit haben wir uns mit solchen Deep Learning basierten Systemen auseinandergesetzt. Wir waren nicht in der Lage mit der Loss Funktion des existierenden Systems aussagekräftige Resultate zu erzielen und somit auch nicht in der Lage die Resultate der vorhergehenden Arbeit zu reproduzieren. Stattdessen haben wir mehrere Möglichkeiten erforscht wie man das bestehende System verbessern könnte und davon zwei konkrete Verbesserungen implementiert. Ausserdem haben wir die Charakteristiken von drei anderen Systemen angeschaut, welche mit dem VoxCeleb Dataset arbeiten. Dies hat uns ermöglicht, zusätzliche Vorschläge zur Verbesserung des bestehenden Systems vorzuschlagen. Des Weiteren schlagen wir eine neuartige Art und Weise vor wie man das VoxCeleb Dataset aufteilen kann.

Abstract

In this Thesis, we tried to build upon and improve the results of previous speaker diarization systems developed at ZHAW. Speaker diarization deals with the task of automatically determining when which person has spoken and for how long, without prior knowledge of the characteristics of individual subjects with respect to their voices. [1] Speaker diarization technology could serve a wide range of applications, for example, recordings could be made during a conference using a smartphone, and the speaker diarization system would then automatically label the data to support the creation of transcripts. Traditionally, speaker diarization systems were built using statistical methods such as Hidden Markov Models. More recently [1], deep learning methods such as Convolutional and Recurrent Neural Network have also found application in this field. In this Thesis, we examined a system based on a recurrent neural network. The VoxCeleb dataset was used to train the model to improve its ability to deal more accurately with real-world applications. However, we were not able to achieve meaningful results with the loss function of the existing codebase and therefore were not able to replicate the results of the previous clustering experiments. Instead, we researched multiple new approaches and improved upon the existing codebase. We looked at the characteristics of the existing system and were able to implement two concrete approaches to optimization. Additionally, we examined three existing systems dealing with speaker identification and verification on the VoxCeleb dataset, which allowed us to define a set of further enhancements to the original system. We also propose a novel procedure for splitting the VoxCeleb dataset, to better adapt it to the requirements of the Clustering Experiment.

Table of Contents

1 Aim

In this thesis, we will discuss possible measures to improve the quality of speaker embeddings using the system previously developed by Lukic, Vogt, Dürr and Stadelmann [2], which builds upon the research conducted by Stadelmann, Glinski-Haefeli, Gerber and Dürr in [3]. As Lukic et al. [2] suggest, further research is necessary to enable the current system to accurately cluster speakers from a larger dataset than the 40-speaker subset from TIMIT. To conduct our experiments, we used the VoxCeleb dataset developed by Nagrani, Chung and Zisserman in [4]. Our Goal was to improve the current system in such a way that it is possible to firstly work with the new, much larger dataset which is partly concerned with reducing training time while secondly improving its ability to detect the necessary features responsible for extracting speaker discriminative embeddings which could be used during the clustering stage.

2 Introduction

2.1 Siamese Networks

Siamese networks were first introduced by Bromley, Guyon, and LeCun [5] in the context of signature verification. In their paper they explain the basic concept of the Siamese network as follows: “The Siamese network has two input fields to compare two patterns and one output whose state value corresponds to the similarity between the two patterns” [5]. The two inputs, in turn, lead into a so-called sub-network, which is identical for both inputs. In the original paper, a convolutional network was used, but this pattern can also be transferred to other architectures.

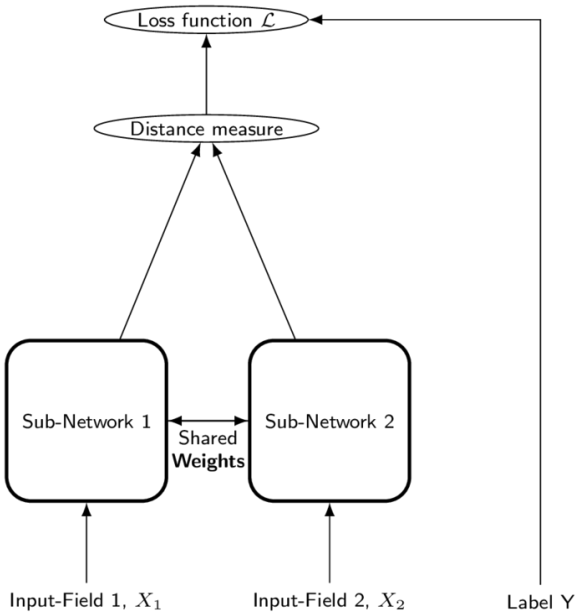


Figure 1-Basic architecture of a Siamese network

Another fundamental feature of the Siamese network architecture is that the output of both subnetworks is used as a feature vector, and a distance metric is calculated between them.

The task of the loss function is to minimize this distance as much as possible if the two inputs correspond to the same class, respectively to maximize it if they correspond to different classes.

In [6] Chopra Hadsell and LeCun present a procedure which should enable a Siamese network to learn a similarity metric discriminately. Their report refers to the application in the domain of facial recognition. However, Nagrani et al. [4] refer to the contrastive loss function presented in this report when using the Siamese network, which indicates that it is also suited for the task of speaker verification. Chopra et al. [6] describe the general loss function as follows:

$$\mathcal{L}(W) = \sum_{\{i=1\}}^{P} L(W, Y, X_1, X_2) \qquad (1)$$

And:

$$L(W, Y, X_1, X_2) = (1 - Y)L_{G(E_W);} + ; \; Y \, L_{1(E_W)} \qquad (2)$$

Where $L(Y, X_1, X_2)$ is a single training example, consisting of label Y, the first input X_1, and the second input X_2. E_w denotes the energy between X_1 and X_2, i.e. a metric computed between X_1 and X_2, Y is a binary prediction, and P denotes the total number of training examples.

If the two input fields belong to the same class, then Y equals 0, if they belong to a different class Y equals 1.

In [7] Hadsell, Sumit and LeCun define the actual loss function differently than in [6], which we found easier to comprehend and sufficient to explain the general principle. The exact loss function then is:

$$L(W, Y, \vec{X_1}, \vec{X_2}) = (1 - Y)\frac{1}{2}(D_W)^2 + (Y)\frac{1}{2}\{max(0, m - D_W)\}^2 \qquad (3)$$

In this case, D_W plays a similar role to that of E_W in the loss function of equation (2) i.e., a distance metric, but in equation (3) D_W is simply the Euclidean distance between the outputs of the sub-networks, given feature vectors X_1 and X_2 respectively. In [7] Hadsell et al. explain that $m > 0$ is a margin that defines a radius around the vector computed from X_1 and X_2. Intuitively one could think of this m setting as a minimum distance, which should not be undercut by the network.

2.2 ResNet

Recent trends in deep learning show a tendency to increase the number of layers in neural networks, with new problems arising which negatively affect accuracy. Right at the beginning of their paper He, Zhang, Ren and Sun [8] make it clear that normalization procedures mostly addressed the problem of exploding and vanishing gradients. They further state that with very deep neural networks another problem comes to the fore, namely that of degradation. They explain that accuracy enters saturation and then drops rapidly. He et al. claim that this effect is not due to overfitting, and that merely stacking additional layers on an existing model tends to deteriorate rather than improve results.

They start their argumentation by defining the function $\mathcal{H}(x)$ as a mapping, which should be approximated by the stacked layers in a neural net. They then state that if $\mathcal{H}(x)$ can be approximated, it is also possible to approximate the residual function $\mathcal{H}(x) - x$.

The solution they propose: instead of making layers approximate $\mathcal{H}(x)$, they let them approximate the residual function $\mathcal{F}(x) = \mathcal{H}(x) - x$, thereby defining the original function as $\mathcal{F}(x) + x$.

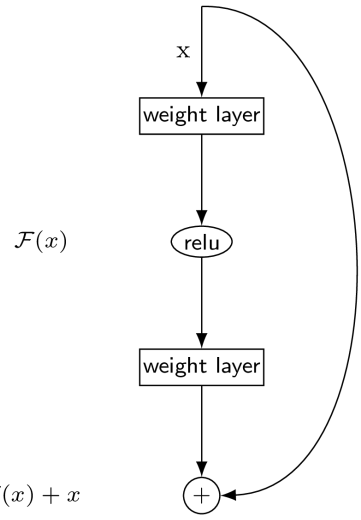
By doing this, He et al. [8] point out, that the error should be no greater in a deeper network than in a shallower one. They state that it is difficult for a neural network to approximate the identity function if multiple non-linear activations are present between the layers. For a residual network to approximate the identity function, it merely has to set all the weights of the respective layers to zero. They explain that while it is rare in practice for the identity function to deliver an optimal result if the optimal solution is close to the identity function, it is easier for the network to learn this mapping.

In their paper He et al. [8] define a building block of a residual network as follows:

$$y = \mathcal{F}(x, \{W_i\}) + x \qquad (1)$$

With x being the input and y the output of the respective layers. $\mathcal{F}$ represents the residual function to be learned and W_i the weight matrices, where i denotes the number of layers, i.e. the number of weight matrices which the residual block comprises. They give a concrete example of a residual block with two layers; the equation and corresponding image are shown on the right and below respectively:

$$\mathcal{F} w_2 \sigma(W_1 x) \qquad (2)$$



Where σ stands for the ReLU activation function.

In case the dimensions of the layers change within the residual block, and thus the output of the last layer no longer matches that of the skip connection, He et al. [8] suggest adjusting the dimensionality of the skip connection by a linear transformation. In concrete terms, this means that the skip connection is multiplied by a matrix, which pads it with zeros to correspond to the dimensionality of the output of the residual block. Another thing to add is that those skip connections can easily be implemented, by adding the value from before the residual block with the output of the last layer within such a residual block.

The concrete equation for this case is given by He et al. [8] as follows:

$$y = \mathcal{F}(x, \{W_i\}) + W_s x \qquad (3)$$

Where W_s is the matrix that performs the aforementioned linear transformation (i.e., padding with zeros).

2.3 LSTM Networks

2.3.1 General Structure

LSTM-networks (long short-term memory) as first proposed by Hochreiter and Schmidhuber [9] are an extension to the general recurrent neural network architecture. They can "… learn to bridge time intervals in excess of 1000 steps even in case of noisy, incompressible input sequences, without loss of short time lag capabilities …". [9]

Those properties are realized by the extension of the traditional recurrent network architecture with an update gate Γ_u; a forget gate Γ_f and an output gate Γ_o. Each of which judges whether or not the internal memory c should be updated, calculating the respective values is only dependent on the activation from the last step and the current input $x^{<t>}$ the corresponding equations are shown below:

$$\Gamma_f = \sigma(W_f[a^{<t-1>}, x^{<t>}] + b_f) \quad (1)$$

$$\Gamma_u = \sigma(W_u[a^{<t-1>}, x^{<t>}] + b_u) \quad (2)$$

$$\Gamma_o = \sigma(W_o[a^{<t-1>}, x^{<t>}] + b_o) \quad (3)$$

At each time step a new potential value for the memory $\tilde{c}^{<t>}$ is computed taking into consideration both the activation of the previous layer $a^{<t-1>}$ as well as the new input $x^{<t>}$:

$$\tilde{c}^{<t>} = tanh(W_c[a^{<t-1>}, x^{<t>}] + b_c) \quad (4)$$

To calculate the new value for the memory cell $c^{<t>}$, Γ_f is applied to the memory cell from the last time-step $c^{<t-1>}$ by computing the element-wise product. In the same manner Γ_u is applied to the new potential value for the memory cell $\tilde{c}^{<t>}$, and thereby determines how significant its influence should be as can be seen in equation (5):

$$c^{<t>} = \Gamma_u \circ \tilde{c}^{<t>} + \Gamma_f \circ c^{<t-1>} \quad (5)$$

Lastly, a nonlinear activation function is applied to the resulting memory cell $c^{<t>}$, and the output gate Γ_o is applied to the result to get the activation $a^{<t>}$.

$$a^{<t>} = \Gamma_o \circ tanh(c^{<t>}) \quad (6)$$

To compute the prediction value y, $a^{<t>}$ is passed through a Softmax layer. A visual representation of the workings of an LSTM cell can be seen in figure 3.

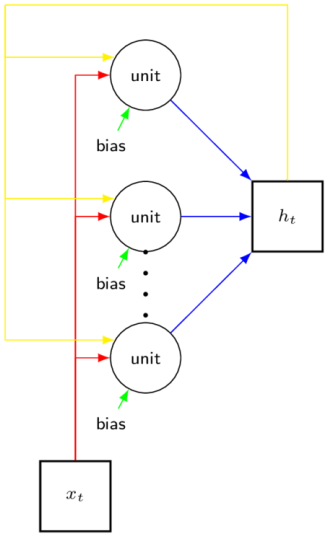


Figure 2- illustration of why the dimensions are the equal

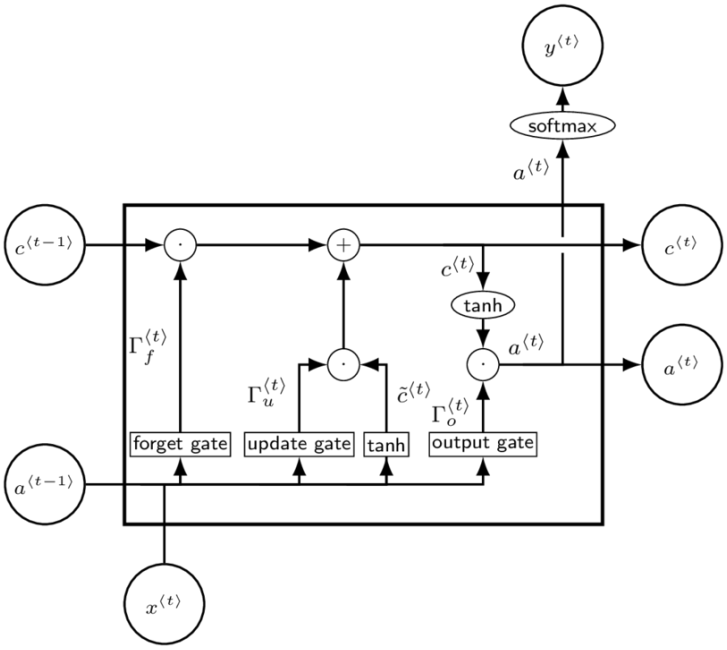


Figure 3 - LSTM Visualization

The dimensions of the matrices of gates Γ as well as the memory cell c are all the same, and in the Keras implementation, only a single LSTM-cell is created. Therefore, the weights of all time-steps are shared. Figure 2 shows this connection.

2.3.2 Many-to-One

LSTM-networks, as well as other recurrent neural networks, are used in different configurations. Since our goal is to obtain a single vector representation of each utterance fed into the LSTM, we would use a many-to-one configuration, where the resulting activation of each time step would be fed to the next time step, and only the resulting activation of the last time-step would be fed to the following dense layer, as shown in figure 4 below.

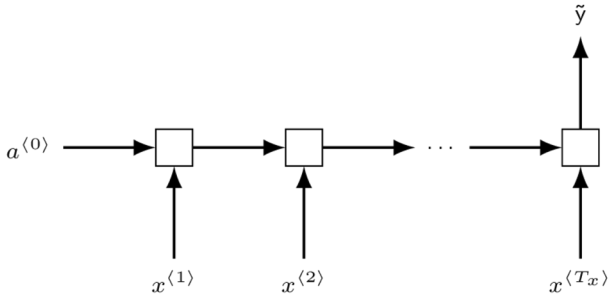


Figure 4 - Many to one

2.4 Combined Kullback Leibler and Hinge Loss

Lukic et al. [2] proposed a combined objective function consisting of Kullback-Leibler divergence and Hinge loss. It is used in a contrastive manner, which means to evaluate the resulting loss, a pair of embeddings would be compared. Depending on the label a different loss would be calculated. When a pair is from the same speakers, the Kullback-Leibler divergence would directly be used as the resulting loss, when a pair is from different speakers, the Kullback-Leibler divergence would be used as a measure for distance, within the Hinge loss with the goal of maximizing their distance, by setting a margin in the Hinge loss. Lukic et al. [2] used a value of 3 for the margin which they found to work better than a value of 2 which was proposed by Hsu and Zsolt [10].

Lukic et al. [2] defined the loss function as follows:

$$KL(p \,||\, q) = \sum_{i}^{c_s} p_i \, log \frac{p_i}{q_i}$$

Where p and q denote the resulting output vector from the last Softmax layer and p_i and q_i are the values in a single cell of the vector. c_s denotes the total number of speakers.

The Hinge loss would then take the calculated Kullback-Leibler divergence and subtract it from the margin, which above was mentioned as having a value of 3:

$$HL(p \,||\, q) \;=\; max(0, margin \;-\; KL(p \,||\, q))$$

Finally, the loss function is combined to return a different value depending on the same/different speaker criteria as already explained. Finally the loss would be symmetrized. I_s: is 1 for same speakers and 0 for different speakers, and conversely for $I_{ds.}$

$$\mathrm{loss(p \,||\, q) = I_s \cdot KL(p \,||\, q) + I_{ds} \cdot HL(p \,||\, q)}$$

$$\mathrm{L(p, q) \;=\; loss(p \,||\, q) + loss(q \,||\, p)}$$

2.5 VoxCeleb and TIMIT

VoxCeleb and TIMIT are two different datasets used in the building of speech systems.

2.5.1 TIMIT Acoustic-Phonetic Continuous Speech Corpus

The TIMIT (Texas Instruments/Massachusetts Institute of Technology) corpus was created by its eponymous organizations to provide a consistent dataset for development of speech recognition systems. Every recording was done in the same room, with the speaker placed in the same position and always using identical audio equipment. The sentences which the speakers pronounce are designed to expose as many distinct linguistic features of their respective dialect (shibboleth sentences). It contains much more male than female speakers and is limited to a few American dialect regions. [11] TIMIT is an old dataset, and much research has been carried out on it.

2.5.2 VoxCeleb - a large-scale speaker identification dataset

The VoxCeleb corpus is extracted from YouTube. The gender ratio is balanced. VoxCeleb contains a great number of non-American speakers but is still biased towards Americans and other anglophones. [4] VoxCeleb is a more recent dataset than TIMIT and relatively few papers dealing with it have been published. In contrast to TIMIT, VoxCeleb provides a more heterogeneous and realistic body of speakers and audio configurations. Therefore, VoxCeleb is a dataset which is not biased towards a certain microphone or, a certain microphone and speaker positioning.

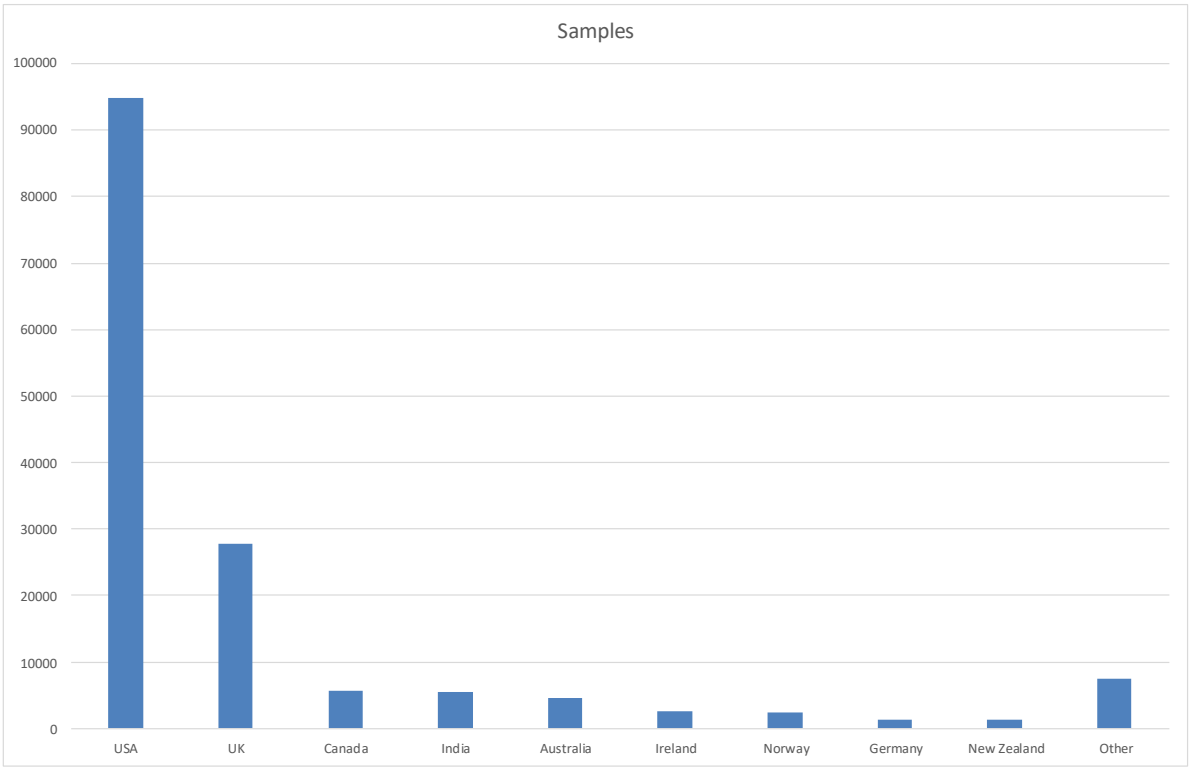


Figure 5 - Breakdown of samples per Nationality in VoxCeleb

2.5.2.1 Arrangement of the Data-Splits

The VoxCeleb data set includes just over 153,000 utterances from 1251 different speakers. Nagrani et al. [4] suggest a different type of subdivision for the task of identification and verification.

In the identification split the dev-set contains 145'256 utterances and in the test set 8'251 utterances. In both cases, the total number of speakers is 1251.

In the verification split, the dev-set contains 148,642 utterances from a total of 1,211 speakers, and the test set contains 4,874 utterances from 40 speakers.

It is important to note that the folder structure of the data set corresponds to the Verification split.

In order to determine the affiliation of the Utterances in the case of identification, one needs the corresponding lists provided by Nagrani et al. [4] on their website.

The list "List of trial pairs for verification" is a pairing of the utterances of the test-set from the verification split proposed by Nagrani et al. in [4] to allow a better comparison of the results between different works. It comprises 37'720 different pairings.

2.5.2.2 Properties of Audio Files:

Nagrani et al. report in [4] to have first merged the audio to a single channel (mono) and then converted it to a sampling frequency of 16'000 Hz. They report having extracted the audio from videos taken from YouTube. There is little information on the quality YouTube provides, and users of the service have different options for uploading audio (with a minimum quality of 128 kbps), but as a Google employee stated: "Right now YouTube Music streams and downloads at 128kbps HE-AAC" [12]. With HE-AAC being a lossy data compression format [13], the conversion to 16'000 Hz is adequate to have a common basis on which to operate, but any further resampling should be avoided since it would possibly only worsen the quality of the respective files.

3 Speaker Embeddings on VoxCeleb

3.1 Related Work

In this section, we consider three reference systems that deal with the VoxCeleb data set in the context of speaker embeddings. They are all concerned with speaker identification and verification.

3.1.1 VoxCeleb - a large-scale speaker identification dataset

In the work of Nagrani et al. [4], the creation of the actual data set is described as well as the implementation of a CNN based system.

3.1.1.1 Network architecture:

Nagrani et al. [4] base the architecture of their network on the VGG-M network as proposed by Chatfield, Simonyan, Vedaldi and Zisserman in [14]. Nagrani et al [4] replace the last fully connected layer with two layers, for the first they use a fully connected layer with dimensions 9 x 1 and for the second one a pooling-layer of dimensions 1 x n, where n is dependent on the length of the segment fed to the network.

To train the network on the task of speaker verification, they use a two-step approach, where they first train a single network on the task of speaker identification and only then continue to train a Siamese network on the task of speaker-verification.

3.1.1.2 Training

3.1.1.2.1 Identification

For the task of identification, they report using an additional Softmax-layer with the dimensionality equal to the number of speakers (1251). The network is trained with categorical cross entropy.

3.1.1.2.2 Verification

For the verification task, they state the use of a Siamese network trained with the contrastive loss as suggested by Chopra et al. [6]. By maintaining the basic network, they can employ transfer learning by using the weights learned during identification. For this task, all layers are frozen except for the last layer which is the only one updated during verification.

3.1.1.3 Embedding Extraction

During embedding extraction, the architecture is kept identical to the one used during training, except for the last Softmax layer, which is adjusted to a dimensionality of 1024, and whose output is then used to generate the embeddings directly.

3.1.1.4 Sampling

They report sampling 3-second crops, where the starting point is determined randomly. Additionally, to identify pairs, they choose half of the negative pairs randomly. The other half is determined by applying hard negative mining, from which only the 10 topmost percent of the most difficult negatives are sampled.

3.1.1.5 Spectrogram

Nagrani et al. [4] report to have used a sliding window process, with a Hamming window of 25 ms, 10 ms steps and 1024-point FFT. They specify to have spectrograms of dimensionality 512 x 300 for 3 seconds of speech. Additionally, they perform mean and variance normalization on every frequency bin of the spectrum. They point out that the normalization step is crucial for the success of their system and state that this leads to an increase of 10% in classification accuracy.

3.1.1.6 Parameter Count

Nagrani et al. [4] report that by replacing the last two layers of the original VGG-M network architecture, they can reduce the number of parameters from 319 million to 67 million, which they explain helps reduce overfitting.

3.1.2 Unified Hypersphere Embedding for Speaker Recognition

3.1.2.1 Network architecture:

Hajibabaei and Dai [15] report that they use the ResNet-20 architecture introduced by He, Zhang, Ren, and Sun in [8] which, similar to that of Nagrani et al. [4], belongs to the category of convolutional neural networks.

3.1.2.2 Identification

For the task of identification, they use an additional Softmax layer on top of the last layer of their basic network, whose dimensionality is equal to the number of speakers in the dataset (1251). For identification, they use categorical cross-entropy among various other loss functions. They report to get the best results by using two specific losses, namely AM-Softmax and logistic margin.

Like Nagrani [4] in a first step, the network of Hajibabaei and Dai is trained with categorical cross entropy. When continuing to train with more contrastive loss functions, such as AM-Softmax or logistic margin, the weights from the training with cross-entropy are adopted, and only the last layer is reinitialized with Xavier.

3.1.2.3 Verification

Although the accuracy for the verification process has been stated, no specific training process is mentioned in this regard.

3.1.2.4 Embedding Extraction

For the extraction of embeddings, 50 randomly selected 3-second pieces are taken from the identification and the verification split and fed to the network. The resulting embeddings are then averaged.

3.1.2.5 Sampling

Particularly noteworthy is the use of data augmentation in their system. They report that a better result can be achieved by repeating an utterance and then randomly setting the starting point for sampling with the additional inclusion of reversed utterances. They emphasize the improved quality of embeddings obtained in this way.

3.1.2.6 Spectrogram

Hajibabaei and Dai's [15] approach to the extraction of spectrograms is very similar to that used by Nagrani. However, they report that instead of 1024 points FFT they used a lower resolution of 512 points. Furthermore, they add the amplitude and the DC component of the signal to generate a short-time Fourier transform. They use audio fragments with a length of 3,015 seconds and thus obtain a spectrogram with dimensions of 300 x 257

3.1.3 Attentive Statistics Pooling for Deep Speaker Embedding

3.1.3.1 Network Architecture

Okabe Koshinaka and Shinoda [16] mention their use of the network architecture as described by Snyder, Garcia Romero, Povey and Khudanpur [17], which consists of a deep neural network which is divided into two parts, one of which operates on the frame level and one which operates on the segment level.

The frame level part is composed of 5 layers, wherein each successive layer frames from different time steps are spliced together.

The first layer of the segment level is referred to as the statistics pooling layer, Snyder et al. [17] explain that it receives the output of the final frame level layer, aggregates over it and computes the mean and standard deviation. They further state that these statistics are then concatenated and passed to two additional hidden layers with dimensions 512 and 300, of which both they are then able to extract embeddings.

In addition to the architecture proposed by Snyder et al. in [17] to implement an additional attention model, Okabe et al. [16] would extend the network in such a way, that for each frame level feature an additional attention score would be calculated and then normalized over all frames. They continue to explain that this normalized score would be used as a weight to calculate a mean vector. By following this approach, they point out that an utterance level feature focuses on important frames and should, therefore, be more speaker discriminative.

The last layer of the network consists of a Softmax layer with categorical cross-entropy as loss function. An overview of the architecture is shown in the image below:

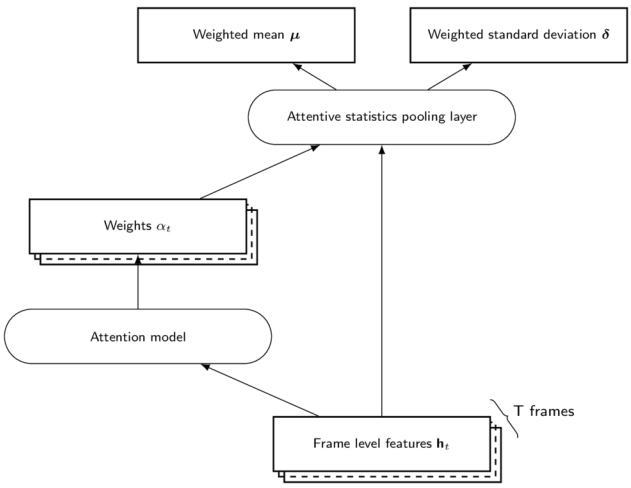


Figure 6- Architecture of the attentive statistics pooling model

3.1.3.2 Training

For the training of their network Okabe et al. [16] would use the splits as proposed by Nagrani et al. [4] for training and verification. From their statements, it can be assumed that the network was only trained on identification and then the resulting embeddings were used directly to examine the particular performance measures.

3.1.3.3 Embedding Extraction

Okabe et al. [16] propose a new pooling mechanism which is distinguished by the fact that the attention model is integrated in the way described above, which they call attentive statistics pooling. In their work, in contrast to Snyder et al. [17], they state to only use the first fully connected layer with a dimensionality of 512 to extract embeddings.

3.1.3.4 Sampling

Okabe et al. [16] report the incorporation of data augmentation in their training process in two specific ways: Firstly, they would mix the utterances with noise samples from the PRISM corpus and secondly, they would add reverberation to the utterances, before training.

3.1.3.5 Spectrogram

In their paper, they describe the usage of 40-dimensional MFCCs for every 10 ms. They explain that similar to Nagrani [4], they also used normalization, but in their case sliding mean normalization with a window of 3 seconds. They were the only ones among the systems considered to use preprocessing in the form of energy-based VAD.

3.1.3.6 Usage of the Dataset

For the training of their system, they used the identification split as proposed by Nagrani et al. [4], where the number of speakers is slightly less than in the original dataset, namely 1206.

3.1.3.7 Parameter Count

Snyder et al. [17] state that their architecture leads to a network with 4.4 million parameters, One can, therefore, assume that the architecture of Okabe et al. [16] has a similar number of parameters since they specify to use the same network, adding only the attention mechanism.

3.1.4 Comparison of the Systems Considered

System	Accuracy	Dimensionality
VGG-M, Nagrani	80.5 %	1024
ResNet-20, Hajibabaei	94.6 %	128
DNN, Okabe	n/a	128

The accuracy refers to the top 1% of the results, and dimensionality refers to the embeddings extracted. It can be observed that Hajibabaei et al. with 128-dimensional embeddings, achieve higher accuracy than Nagrani et al. with 1024. The numbers were taken from [15].

3.2 Overview of our Approach

3.2.1 Preparation of the Data-Set

3.2.1.1 Spectrograms

To generate mel-spectrograms, we used the settings proposed by Lukic et al. in [2] which are listed below:

- Sampling rate: 16'000 Hz
- FFT-window length: 1024
- Hop-length: 160 samples
- Dynamic-range compression in accordance with [18]
- Mel-spectrogram: 128 elements in the frequency direction

For the dimensions of the spectrogram, we used the settings which Lukic et al. [2] found to deliver the best results, namely 400 ms in the time-domain and 128 in the frequency domain. To extract the embeddings, we used the python library librosa. We then created one spectrogram from each utterance contained in VoxCeleb, with a 100 ms offset at the start, to avoid capturing silence or other unpleasant effects such as clicks.

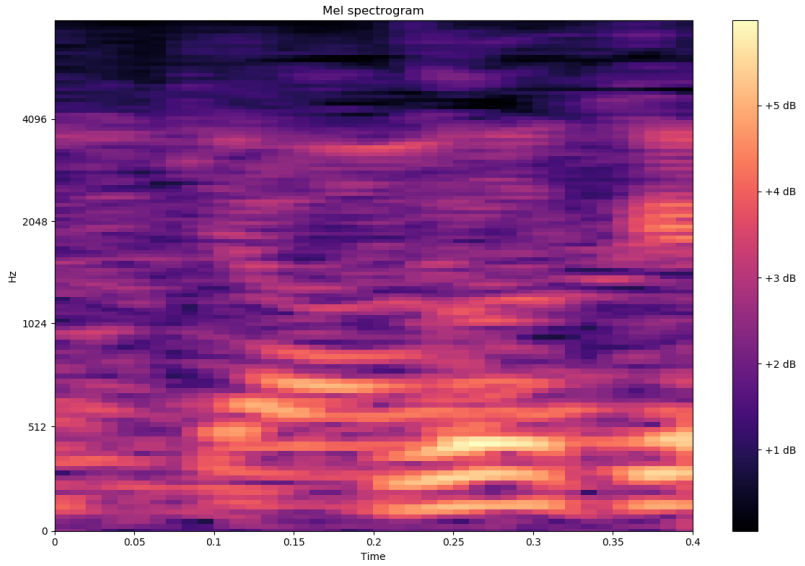


Figure 7 - mel-spectrogram 400ms

The image above shows the mel-spectrogram of a 400ms snippet taken from a single utterance from VoxCeleb captured after applying the same processing chain as would be used on the spectrograms of our system before feeding them to the networks input to generate embeddings. On the x-axis, a single discrete timestep represents a hop of 160 samples, where 40 such hops result in 400 ms given a sampling rate of 16'000 Hz. On the y-axis, a single discrete step represents one of 128 frequency bands. The colors represent the energy of each frequency band at a given timestep x.

3.2.1.2 Data-Splits

In [4] it is defined that for the speaker verification task, all samples from speakers whose names start with the letter 'E' are reserved for testing. The paper states that this procedure provides a good balance of male and female speakers.

To generate training and verification splits, we first calculate a vector where each component corresponds to the number of occurrences of a Gender/Nationality combination in all samples. We then randomly split the speakers in an 80/20 ratio and calculate the same vector for the two resulting split sets. Using those vectors, we calculate a metric;

$$metric = \left|\left|2 \cdot \overrightarrow{occ_{all}} - \overrightarrow{occ_{train}} - \overrightarrow{occ_{dev}}\right|\right|_2$$

which we try to minimize by repeating the process mentioned above.

3.2.1.3 Pair-Generation

To maximize the training effect, we generated pairings which are hard to learn for the Neural Network. In our implementation, a random sample is selected first, followed by a second random sample from the same speaker. Then a negative pairing is created by selecting samples from two different speakers, again randomly, but under the condition that they speak the same language and have the same gender. This should make the two samples as similar as possible while being from two distinct speakers. If there is only one speaker with a particular Gender / Nationality combination, the sample is skipped, and the process repeated.

3.2.2 Network-Architecture

3.2.2.1 Basic Network

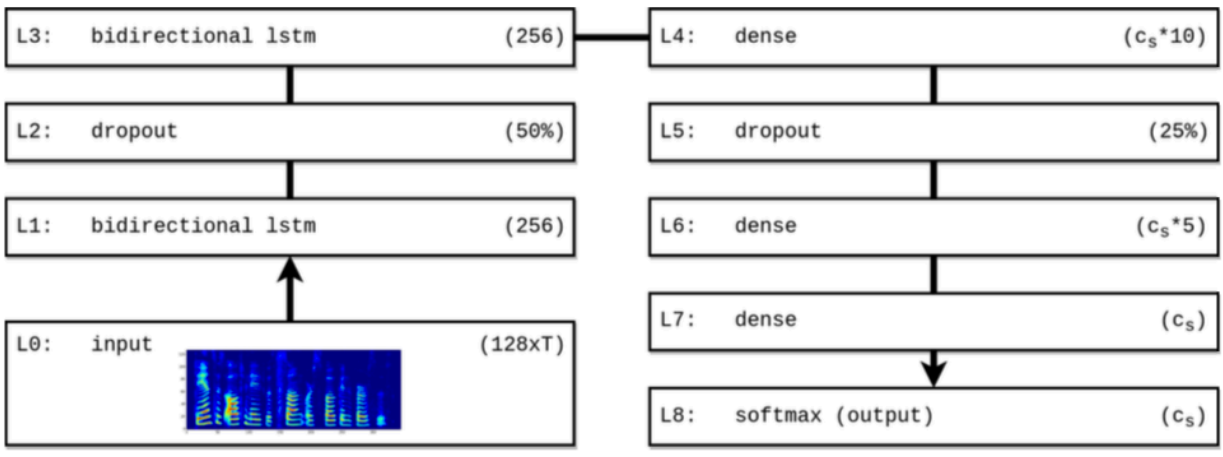


Figure 8 – Visualization of the ANNPR Network Architecture from [3]

The basic network of our system resembles the architecture proposed by Lukic et al. in [2], which consists of 2 Bidirectional layers with dropout between them, followed by 3 dense layers and a Softmax layer, where a second dropout layer is between the first and the second dense layer, as can be seen in figure 8.

The implementation of the second bidirectional LSTM layer by Lukic et al. [2] uses a dimensionality of 512 for the output.

Input is provided at every step of the LSTM layer (40 timesteps), and each input is of dimensionality 128 (mel frequency-bands). The number of units in the corresponding hidden state therefore also amounts to 128. Since in the present case we are dealing with a bidirectional LSTM network, the dimensionality of the resulting output is 256. In the adapted version which we compared to the original, we decided to set the number of units of the second LSTM layer to 128. The resulting output dimensionality in the case of a bidirectional LSTM is 256 as opposed to 512 in the original Version. This decision was motivated by the fact that the number of hidden units and the number of output units should usually match, as described in section 2.3.1 on LSTMs.

In the original version, the dimensionality of the dense layers directly corresponds to the number of speakers present in the dataset, which is denoted as c_s, so the dimensionality of the first layer is 10 * number of speakers. VoxCeleb comprises 1251 speakers, which would have resulted in a dimensionality of 12'510 for the first dense layer. If this approach is applied to all layers, the resulting network has 89'955'039 trainable parameters as can be seen in figure 9, in our experiments this led to prolonged training times and proved to be unfeasible.

Layer (type)	Output Shape	Param #	Connected to
input_a (InputLayer)	(None, 40, 128)	0	
input_b (InputLayer)	(None, 40, 128)	0	
base_network (Sequential)	(None, 1251)	89955039	input_a[0][0] input_b[0][0]
distance (Lambda)	(None, 1)	0	base_network[1][0] base_network[2][0]

Total params: 89,955,039
Trainable params: 89,955,039
Non-trainable params: 0

Figure 9- Number of trainable parameters for original layer settings

We intended to find a compromise that would, on the one hand, maintain the relationships between the individual dimensions as much as possible and, on the other hand, enable effective training.

For this reason, we decided on the following adjusted dimensions for the dense layers:

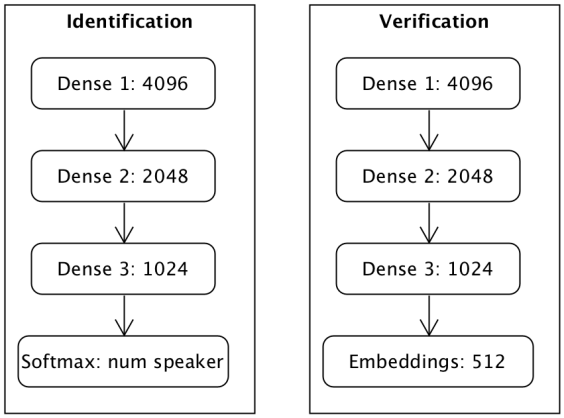
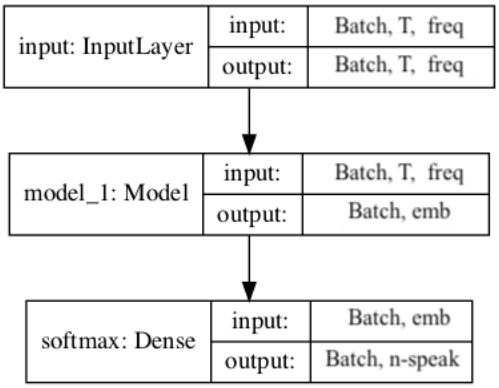


Figure 10 - Adjusted dimensions of the dense layers

This led to a reduction in the number of parameters from over 89,000,000 to 13,193,160.

3.2.2.2 Identification / Pre-Training

Nagrani et al. [4] and Hajibabaei and Dai [15] both use the categorical cross entropy loss function to first train their network on the task of speaker identification, before using more discriminative loss functions for the task of speaker verification. Hajibabaei and Dai state that "Initializing the networks' coefficients before training them with more discriminative loss functions … often results in convergence to sub-optimal solutions or no convergence at all“ [15]. For this reason, and in order to better compare the basic features of our architecture with those of the reference systems, we decided on the following configuration for the pre-training, i.e. the identification step: The base-network, trained with categorical cross entropy and Adam-optimization in accordance with [2] as can be seen in the graphic below:

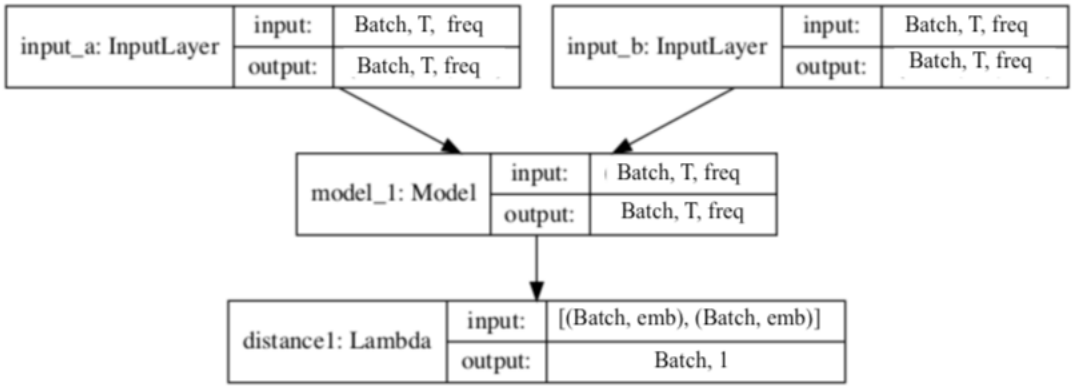


Where Batch denotes batch-size, T the number of timesteps (40), freq the number of frequency-bins (128), emb the dimensionality of the desired embeddings (512 in our case), and n-speak the number of speakers contained in the dataset on which the network is trained.

3.2.2.3 Verification / Training

Nagrani et al. state in [4] that for generating embeddings it is possible to use the network trained on the task of speaker identification but point out that "… it is better to learn an *embedding* by training a Siamese network with a contrastive loss …" [4] such as suggested by Chopra et al. in [6] which is concerned with face verification. We, therefore, modified the original architecture, by using the basic network described above as the two sub-networks of a Siamese network. Each branch would receive a mel-spectrogram. To join the branches, we used a lambda-layer from Keras, which allows applying arbitrary functions to its input-layers, and then used this layer as the final output.

To ensure that the Siamese part of the network shares the weights rather than computing them twice, one has to instantiate the basic network only once, and hand in both branches of the Siamese network as inputs. Team Keras also used this approach in their Siamese network implementation. [19]



With the same naming as in the diagram on pre-training, note the output of distance1 is 1-dimensional, this is the calculated symmetrized Kullback-Leibler divergence.

3.2.2.4 Implementation of the objective function

As can be seen from the formulas in the section on the Loss function of the ANNPR-System, the Kullback-Leibler divergence must only be calculated once and is then also used in the Hinge loss. In consideration of [6] and the Keras implementation by team Keras [19] we decided to calculate the symmetrized Kullback-Leibler divergence as a measure for distance in the lambda layer that connects the two branches of the Siamese network. The output of the lambda layer, in turn, was used to compute the final loss function. The label y of the pair (either 1 or 0) could then be used to determine the applicable loss. Our objective function, therefore, implements the loss defined in equation (4) as follows:

$$Loss(P||Q) = y \cdot KB_{symm}(P||Q) + (1 - y) \cdot HL(KB_{symm})$$

3.2.2.5 Embedding Extraction

To extract the embeddings, one can use an instance of the base-network and feed it the total number of 400 ms spectrograms comprising a complete segment. The embeddings are obtained by capturing the output of the networks last layer, which during the training stage was the input to the lambda layer responsible for computing the Kullback-Leibler divergence. With this approach, we can extract embeddings from the layer that was responsible for generating the values provided to the loss function. Another advantage of this approach is the ability to use a function supplied by Keras called predict_on_batch, that facilitates the transformation of entire segments consisting of any number of spectrograms into a single embedding within a single pass.

4 Experiments

In order to better compare the results of our network with the investigated systems under section 3.1, we chose a two-part approach. In a first step we would examine the properties of the network developed by Lukic et al. [2] concerning the task of identification, Likewise, with the adjusted model, we proceeded in the same way. In a second step, we used the weights of the model that performed better and transferred them to the Siamese model as initialization values, in order to examine its properties concerning verification.

4.1 Identification

4.1.1 Setup

The identification experiment was carried out in two variants. In the first variant, we tried to configure the network as similar as possible to Lukic's description in [2]. However, as already explained in section 3.2.2.1 with adapted dimensions of the layers. In the second variant, we compared the characteristics of the original model with those of the model we had adjusted. In both cases, we used the same parameters except for the addition of a ReLU activation function in the dense layers and an output dimension of 256 in the adjusted model of the second BLSTM layer.

The maximum number of epochs was 500. Early stopping was applied. The abort condition was related to the validation accuracy, and the delta value was 1e-4, hence if the accuracy did not improve by this value, the abort condition is fulfilled. Also, the Patience parameter was set to a value of 10. Thereby ten more epochs were carried out after fulfilling the abort condition. The batch size for all experiments was 256.

4.1.2 Results

4.1.2.1 Original ANNPR Model

4.1.2.1.1 Loss

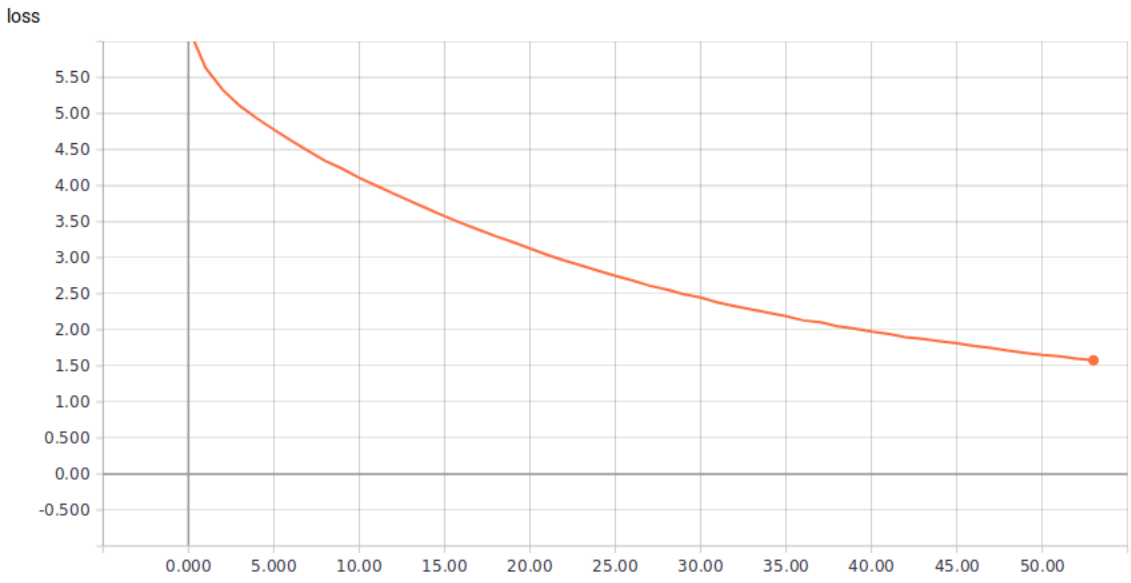


Figure 11- Training loss of the original ANNPR model

As can be seen in figure 11 the loss decreased monotonously during training, with an initial value of just over 6.5 and reached a value of 1.5 after epoch 53. This value would have decreased further, as previous experiments showed, but would have been contrary to the development of the validation loss.

Figure 12-Validation loss of the original ANNPR model

The validation loss shown in figure 12 decreased until epoch 16, where it reached its minimum value of 4.2. After that, the validation loss increased again, with a maximum value of 5.6 at epoch 50. The value after 53 epochs was 5.1.

4.1.2.1.2 Accuracy

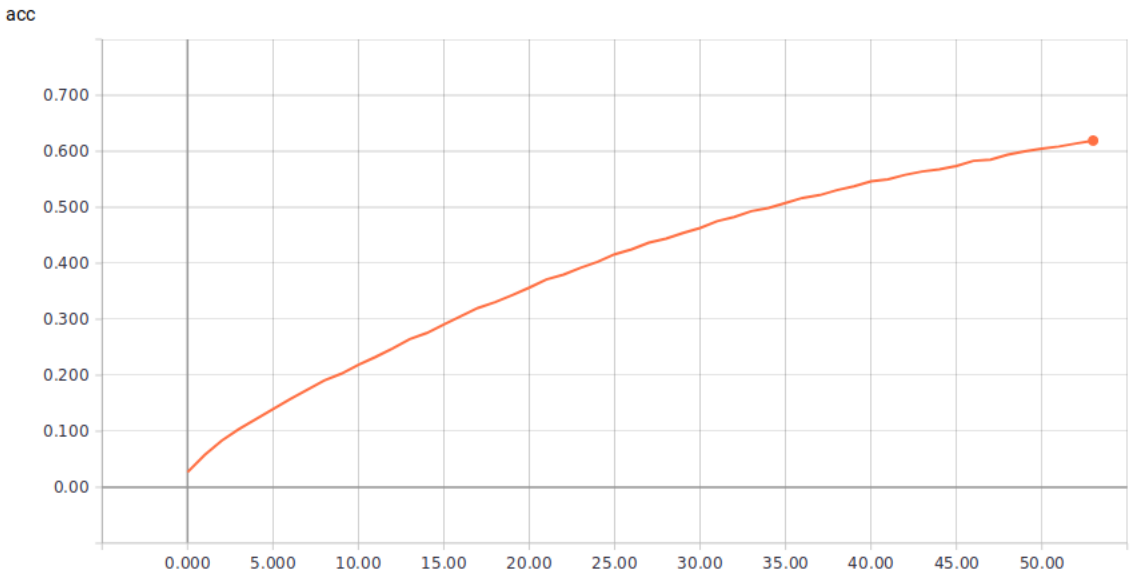


Figure 13- Training accuracy of the original ANNPR model

In figure 13 it can be observed that the accuracy in training increased monotonously, whereby the maximum after 53 epochs amounted to a value of 61 %.

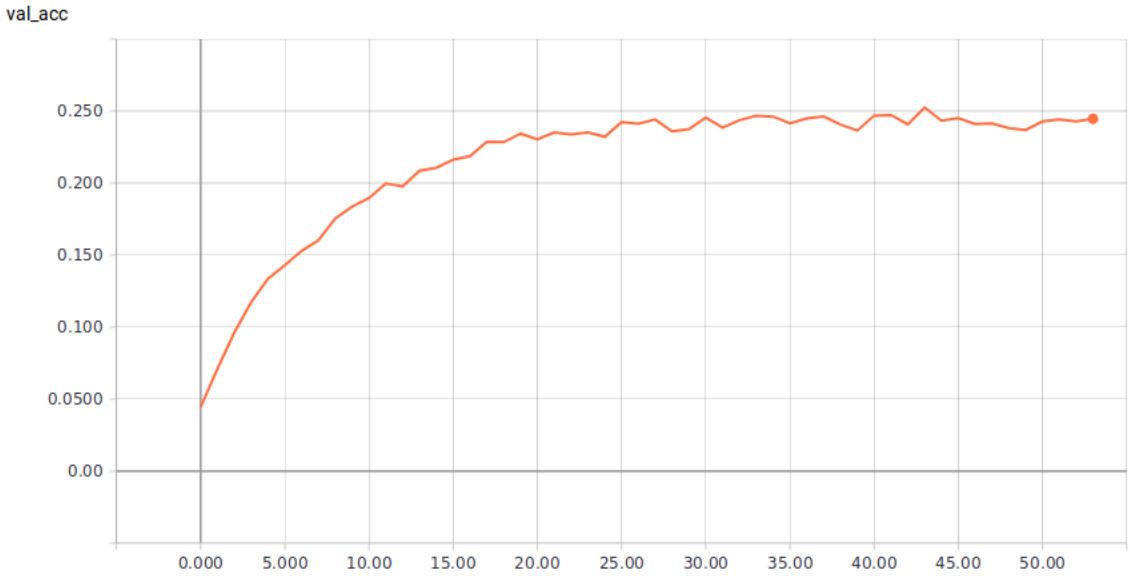


Figure 14- Validation accuracy of the original ANNPR model

The validation accuracy, on the other hand, approached an upper limit of 26 % until epoch 25 and did not increase noticeably afterward, as shown in figure 14. It reached its peak at epoch 43, with a value of 25.25%.

4.1.2.2 Adjusted ANNPR Model

4.1.2.2.1 Loss

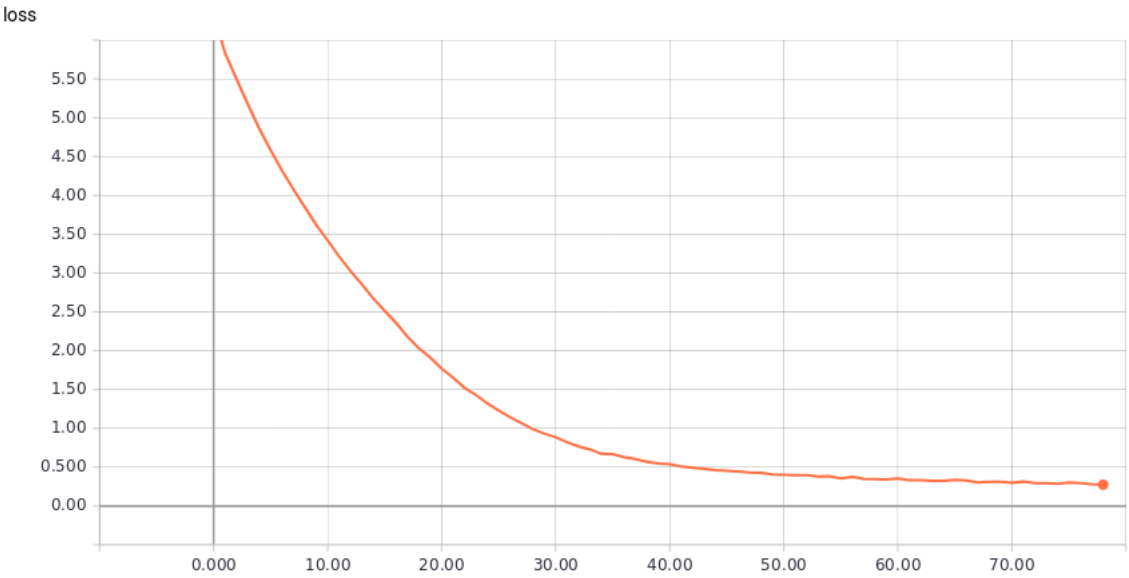


Figure 15 - Training Loss of the adjusted ANNPR model

As with the original model, the loss decreased monotonously, it also began at an initial value of a little over 6.5, but then reached a value of 0.27 at epoch 78. Even taking into account the higher number of trained epochs, the adapted model performed better in this respect, which is shown in figure 15.

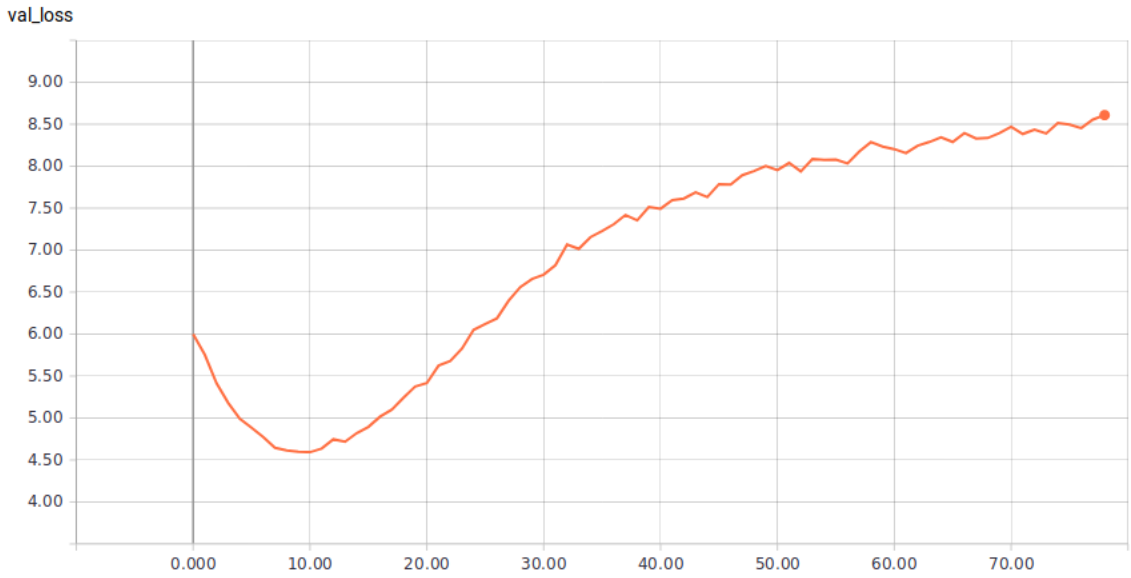


Figure 16- Validation loss of the adjusted ANNPR model

The validation loss, by contrast, only decreased up to epoch 10 reaching a value of 4.5, and then increased continuously, resulting in a value of 8.6 after 78 epochs.

4.1.2.2.2 Accuracy

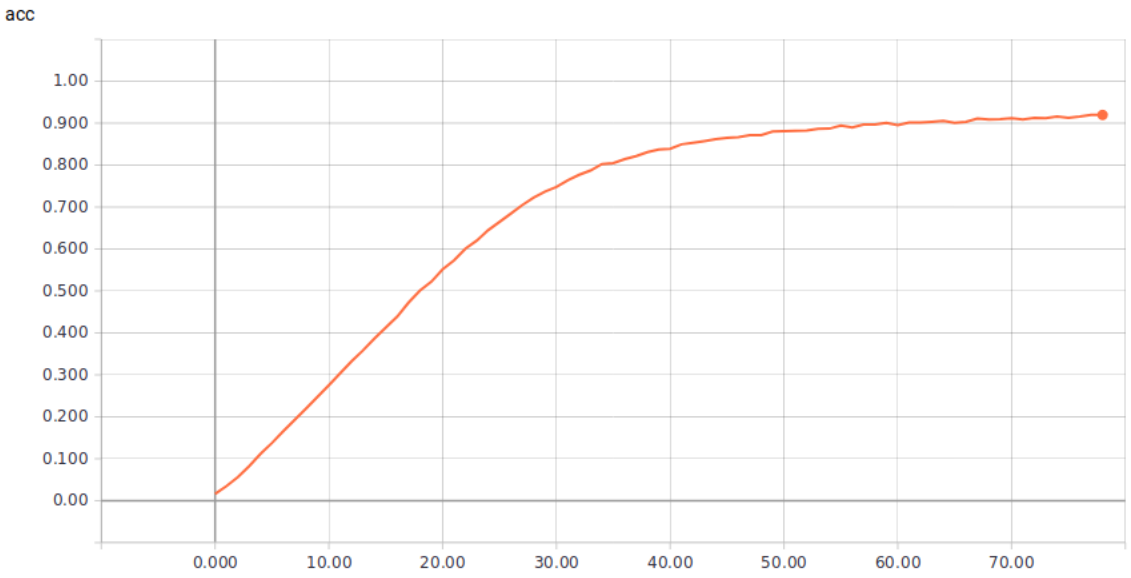


Figure 17 - Training accuracy of the adjusted ANNPR model

In figure 17 the training accuracy of the adapted model is presented. Up to epoch 32, this model has a high slope, which then decreases again, resulting in an accuracy of 91 % after 78 epochs.

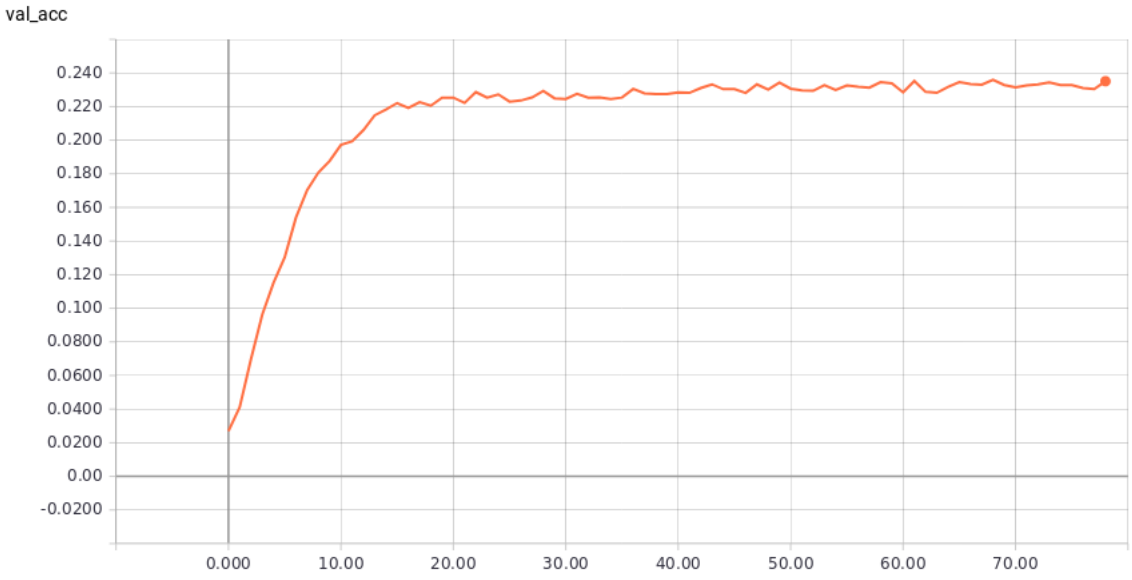


Figure 18- Validation accuracy of the adjusted ANNPR model

Figure 18 shows the validation accuracy of the adjusted model, and its behavior is similar to the original model, the validation accuracy increases monotonously up to a certain point, in this case, epoch 15. After that, however, an upper limit is approached, which is even 2 % lower in the adapted model than in the original model, namely 24 %. The peak was reached at a value of 23.5 during epoch 61.

4.1.3 Verification

The same procedure was used for the verification experiment as for the identification experiment. Only the patience parameter was reduced to a value of 3 since it took about 50 minutes to train on one epoch. The results showed a rapid movement towards a final value, which did not change noticeably afterward. However, the results achieved could not be interpreted meaningfully. We suspect that the implementation of the loss function is faulty. In order to achieve meaningful results, further work in this respect is necessary.

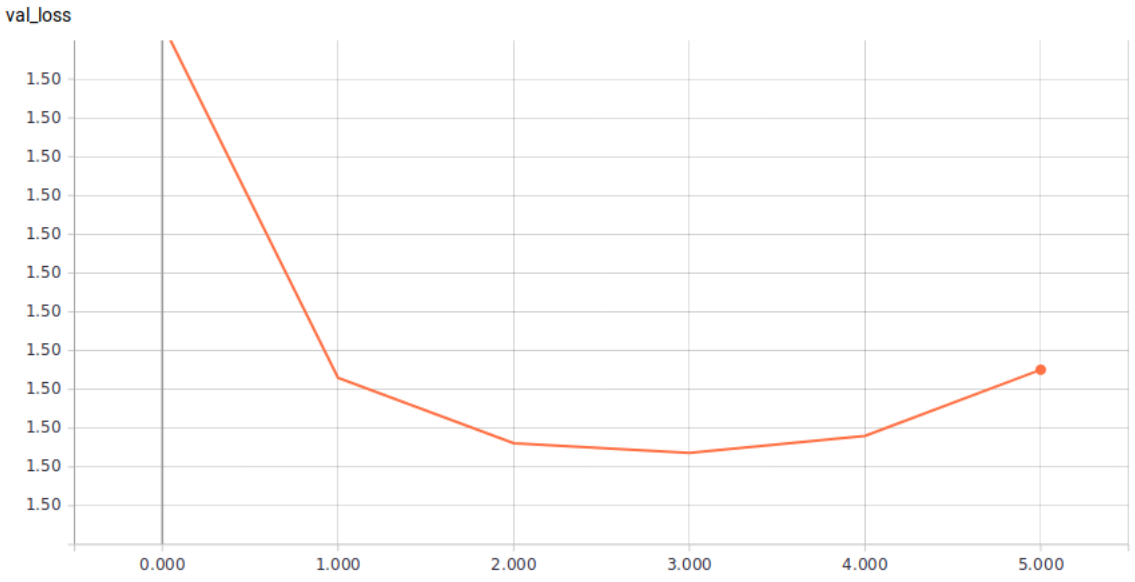


Figure 19-Validation Loss of the Siamese model

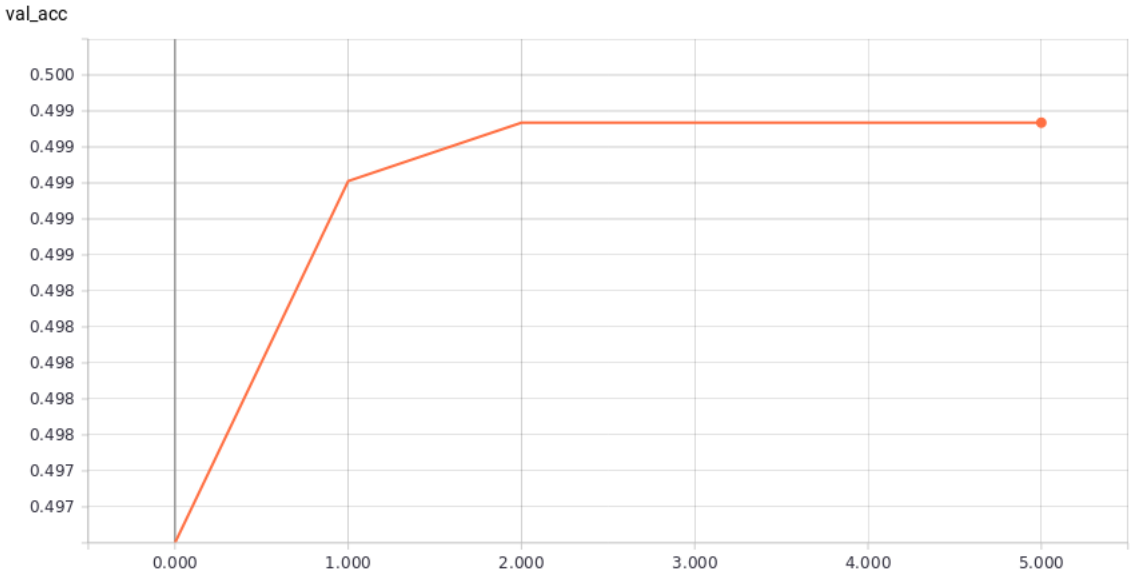


Figure 20 - Validation accuracy of the Siamese model

The diagrams for training and validation are practically identical, except that the loss during training remained completely static after a steep decrease in epoch 1. Therefore, we decided to only present the data of the verification.

4.1.3.1 Summary of the results

The results show that the original model was not able to exceed 62% accuracy in identification during training. Together with the maximum accuracy of 25.25 % in the validation phase, we conclude that this variant has high bias and high variance.

The adjusted model reached an accuracy of 91 % during training. However, the variance compared to the original mode increased all the more since it was even below it, having a value of 23.5 % in the validation phase.

5 Discussion

5.1 Possible improvements of the ANNPR-System

5.1.1 Magic Numbers

Martin suggests in [20] to exchange so-called magic numbers with named constants he points out that "In general it is a bad idea to have raw numbers in your code." [20] In our examination of the existing code we encountered several places, where such numbers would lead to errors, those can usually be identified and corrected, while time-consuming, this does not pose mature harm to the results of the experiments. It is much harder to identify incorrect results, based on values that have been set somewhere in the code, and do not result in errors when they do not match the current setting, and we believe that while not directly related to the system as such, it is worth stressing the importance of this matter.

The reason why we were not able to carry out experiments on the existing codebase is to a certain degree due to issues related to clean code.

5.1.2 Loss Function

In the current System, a contrastive Loss function as described in Combined Kullback-Leibler and Hinge Loss in section 2.4 is used. Lukic et al. [2] would first compute the output of a complete batch and then compare each possible pair. For a batch size of 100 this results in 100^2 comparisons for each batch. Since the compilation of each batch is generated randomly, there is no control over the ratio of same/different-speaker comparisons. Since the success criterion is binary, i.e. only same/different is considered, we believe that the ratio between same and different speakers should be balanced in order to train both cases equally frequently, so as not to distort accuracy in one direction. As a simple example: If a positive pair only occurs in a quarter of the cases, the network could achieve 80% accuracy by merely stating that they are different speakers in each case.

5.1.3 Sampling Rate

The implementation of ANNPR System uses the default sampling rate of librosa which is according to [21] 22'500 Hz, Stadelmann et al. state to use a sampling rate of 16000 Hz in their experiments [3], to which Lukic et al. refer in their work [2].

5.1.4 Rescaling of the LSTM- Output

As explained in the section on LSTMs the shape of all weights within an LSTM block is equal and depends on the dimensions of the input at each given timestep. In the implementation of the ANNPR the dimensionality of the output is rescaled to 512, by setting the unit-parameter, this, in turn, affects the dimensions of the internal weights. As can be seen in figure 2 in section 2.3.

5.1.5 Activation Function

In the dense layers of the ANNPR system, no activation function has been specified, Keras has a default value of None "If you don't specify anything, no activation is applied (i.e. "linear" activation: a(x) = x)" [22].

If $x_1 = Ax_0$, $x_2 = Bx_1$ and $C = BA$ then $x_2 = Cx_0$

Therefore, it might be of interest to add a nonlinear activation function between the dense layers following the bidirectional LSTM.

5.1.6 Manual Embedding Extraction

The extraction of the embeddings in the original implementation was achieved by storing the weights and subsequent manual extraction, which involved reshaping of the dimensions of the obtained weights, this approach might be error prone but more importantly is hard to comprehend for third parties.

5.1.7 Reducing Variance

As the experiments have shown, one of the most urgent problems of the current system is its high variance. To counteract this, we make two proposals:

5.1.7.1.1 Data Augmentation

We found the approach of Hajibabaei in [15] worth mentioning, which is characterized by the fact that an utterance is repeated several times in order to set the starting point differently each time a specific utterance is sampled. Also, he reports that by adding inverted utterances he could achieve better results, which we believe is worth trying.

We find the approach of Okabe et al. [16] should also be considered. It involves applying noise or reverberation to the actual utterances.

Training over significantly more examples might help to reduce the problem of variance.

5.1.7.1.2 Additional 1x1 Convolutional Layer

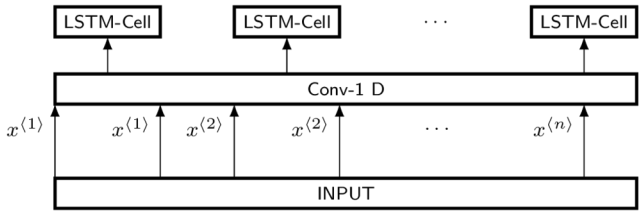


Figure 21- Additional convolutional layer before LSTM

What we noticed during the evaluation of the considered reference systems was that they all used spectrograms of much longer time intervals. We, therefore, suggest adding a 1 x 1 convolutional layer before the first BLSTM layer, which could extract information already before the recurrent part of the network. A possible approach to the implementation is presented in figure 21 where n represents the number of inputs in the time domain. The Conv-1 D layer allows to shrink the dimensionality, and therefore the needed LSTM Blocks. Extending the system in this way would allow covering a longer period without enlarging the computationally intensive recurrent part of the network further.

5.2 Usage of the VoxCeleb Dataset for Future Clustering Experiments

An essential requirement of a clustering experiment is to examine the ability of the system under observation to clusters speakers on whom it has not been trained previously. For this reason, we excluded the identification split offered by Nagrani et al. from the outset.

We propose to use the verification split as a foundation to define a new subdivision, where the test set is reserved for the clustering experiment, and the dev set is used for training and validation respectively.

For reasons of comparability, it is preferable to maintain the subdivision much as possible, but the fact that the test set with 40 speakers is rather small should be considered, in cases where the clustering experiment should comprise more speakers.

To ensure a similar distribution among both, train as well as dev set, we recommend applying stratified sampling with respect to country, as there are the most significant fluctuations in this category while keeping the speakers disjunct.

Since the number of training examples using this procedure is less than 100,000, we think that data augmentation would be of great benefit.

6 Conclusion

We have rewritten the part of the ANNPR codebase concerned with embedding extraction from scratch, focusing on avoiding so-called Magic Numbers, and integrating current versions of the libraries in use. We were able to identify two concrete ways to improve the identification experiments, namely adding non-linear activation functions and avoiding the rescaling of the output of the second BLSTM layer. Furthermore, the results of the experiments showed that the adapted model has a higher accuracy on the training data, but also a higher variance during validation. In our opinion, this indicates that the model needs a more differentiated dataset. To take this into account, we suggested the use of data augmentation techniques, especially those used by Hajibabaei in [15]. Besides, we provided an approach on how to make longer sections of an utterance accessible to the network by

prepending an additional convolutional 1 x 1 layer to the LSTM layer. We developed a new approach to implementation, namely the introduction of a Siamese network. It allows a finer granularity in the selection of the ratio of positive to negative pairs in training. However, we were not able to generate meaningful results using the combined Kullback Leibler Hinge loss function. Further research in this regard is needed.

Additionally, we propose to consider the contrastive loss function, from [6] which is also used by Nagrani for verification in [4], as a possible alternative. Since our architecture already shows a high similarity to the reference implementation of Keras concerning the Siamese part, this should be feasible. We also proposed an approach subdividing VoxCeleb optimized for the clustering experiment, which meets the condition that the system is confronted with speakers in the clustering phase that it has not seen before. These measures should help to increase the quality of the embeddings and ultimately contribute to improving the results of the existing diarization pipeline.

7 Tables

7.1 References

[1] S. B. N. E. C. F. G. F. O. V. Xavier Anguera, "Speaker Diarization: A Review of Recent Research," *IEEE TASLP,* p. , 31 3 2010.

[2] Y. Lukic, C. Vogt, O. Dürr and T. Stadelmann, "Speaker Identification and Clustering Using Convolutional Neural Networks," Zurich University of Applied Sciences, Winterthur, 2016.

[3] G.-H. G. D. Stadelmann, "Capturing Suprasegmental Features of a Voice," 2018.

[4] A. Nagrani, J. S. Chung and A. Zisserman, "VoxCeleb: a large-scale speaker identification dataset," *INTERSPEECH,* 2017.

[5] J. Bromley, I. Guyon and Y. LeCun, "Signature Verification using a "Siamese" Time Delay Neural Network," *International Journal of Pattern Recognition and Artificial Intelligence,* vol. 7, pp. 669-688, 1993.

[6] S. Chopra, R. Hadsell and Y. LeCun, "Learning a Similarity Metric Discriminatively, with Application to Face Verification," Courant Institute of Mathematical Sciences, New York University, New York, 2005.

[7] R. Hadsell, S. Chopra and Y. LeCun, "Dimensionality Reduction by Learning an Invariant Mapping," *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition,* vol. 2, pp. 1735-1742, 2006.

[8] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," *IEEE Conference on Computer Vision and Pattern Recognition,* pp. 770-778, 2016.

[9] S. Hochreiter and J. Schmidhuber, "Long Short-term Memory," *Neural Computation,* vol. 9, no. 8, pp. 1735-1780, 1997.

[10] Y.-C. Hsu and Z. Kira, "Neural Network-Based Clustering Using Pairwise Constraints," Georgia Institute of Technology, Georgia, 2015.

[11] J. S. e. a. Garofolo, "TIMIT Acoustic-Phonetic Continuous Speech Corpus," Linguistic Data Consortium, 1993.

[12] Google, "What's the songs' bitrate?," 20 6 2018. [Online]. Available: https://support.google.com/youtubemusic/thread/167019?msgid=167120. [Accessed 19 12 2018].

[13] Wikipedia, "High Efficiency Advanced Audio Coding," [Online]. Available: https://en.wikipedia.org/wiki/High-Efficiency_Advanced_Audio_Coding. [Accessed 19 12 2018].

[14] K. Chatfield, K. Simonyan, A. Vedaldi and A. Zisserman, "Return of the Devil in the Details: Delving Deep into Convolutional Nets," Proceedings of the British Machine Vision Conference, 2014.

[15] M. Hajibabaei and D. Dai, "Unified Hypersphere Embedding for Speaker Recognition," 2018.

[16] K. Okabe, T. Koshinaka and K. Shinoda, "Attentive Statistics Pooling for Deep Speaker Embedding," in *Interspeech*, 2018.

[17] D. Snyder, D. Garcia-Romero, D. Povey and S. Khudanpur, "Deep Neural Network Embeddings for Text-Independent Speaker Verification," in *Interspeech*, 2017.

[18] S. Dieleman and B. Schrauwen, "End-to-end learning for music audio," *Acoustics, Speech and Signal Processing (ICASSP),* p. 6964–6968, 2014.

[19] Keras, "GitHub/Keras/mnist-siamese," [Online]. Available: https://github.com/keras-team/keras/blob/96c3c195ca805758e094780b59077bea9db99f41/examples/mnist_siamese.py. [Accessed 15 12 2018].

[20] R. C. Martin, Clean Code, New Jersey: Prentice Hall, 2008.

[21] librosa, "librosa.core.load," [Online]. Available: https://librosa.github.io/librosa/generated/librosa.core.load.html. [Accessed 16 12 2018].

[22] Keras, "Keras Layers - Keras Documentation," [Online]. Available: https://keras.io/layers/core/. [Accessed 16 12 2018].

7.2 Figures

8 Appendix

8.1 Original Project Description

School of Engineering

Speaker Recognition & Diarization on Realistic Data

PA18_stdm_2

BetreuerInnen:	Thilo Stadelmann, stdm Mark Cieliebak, ciel
Fachgebiete:	Datenanalyse (DA) Software (SOW)
Studiengang:	IT
Zuordnung:	Institut für angewandte Informationstechnologie (InIT)
Gruppengrösse:	2

Kurzbeschreibung:
Several successful student thesis projects in the last semesters at InIT have created a solid foundation for automatic voice recognition based on deep learning technology (see https://github.com/stdm/ZHAW_deep_voice), leading to several student-lead scientific publications on speaker recognition and clustering at international conferences on machine learning. Last semester, this has been applied for the first time to realistic meeting recordings in order to build a complete pipeline answering the question "who spoke when", given only a multi-minute recording of a multi-speaker discussion.

Goal:

The goal of this project thesis is to build a successfull speaker diarization system based on deep learning voice models. A baseline pipeline for the complete process already exists, but suffers from poor performance when used on meeting recordings while models have been built from studio-quality data. Hence, the focus of this thesis will be to build a voice model (likely a CNN or RNN) on the more realistic VoxCeleb dataset (see link below).

Methodology:

- Read the related scientific work and collect lessons learned from past attempts
- Develop a personal research question together with the supervisors
- Make the baseline pipeline run for you on public meeting data
- Train a deep learning voice model on VoxCeleb to produce embeddings for unknown speakers
- Design a series of experiments to verify your pipeline and evaluate your research hypotheses
- Create a technical report (your thesis) in a scholarly manner (in German or English)

Voraussetzungen:
You can attempt this project in either German or English. No prior knowledge of speaker recognition / diarization or CNNs / RNNs is required. First contacts with AI and machine learning help, but far more important is your fascination for the topic, your skills in quickly engineering a complex processing pipeline (we will likely use Python, TensorFlow, and Docker to run experiments on a GPU cluster), and your eagerness to experiment systematically.

This thesis lends itself perfectly for a continuation in a subsequent Bachelor thesis (and produce a scientific paper about the results as first author, together with the supervisors). It offers a first glance into research and scientific work, and is also a great test run for potentially pursuing Master studies with our team.

Die Arbeit ist vereinbart mit:
Flurin Georg Gishamer (gishaflu)
Philippe Hürlimann (huerlphi)

Weiterführende Informationen:
http://www.robots.ox.ac.uk/~vgg/data/voxceleb/

Thursday 20. December 2018 16:21