Ulm University | 89069 Ulm | Germany

Faculty of Engineering, Computer Science and Psychology
Neural Information Processing

Deep CNNs With Self-Attention for Speaker Identification

Project Report at Ulm University

Submitted by:
Ethan Swistak
Jiayi Liu
Yile Bai
ethan.swistak@uni-ulm.de
jiayi.liu@uni-ulm.de
yile.bai@uni-ulm.de

Reviewer:
Friedhelm Schwenker

Supervisor:
Friedhelm Schwenker

2020

Version from December 7, 2020

Abstract

Speaker identification has gained wide traction as an application area for artificial neural networks over recent years. However, the majority of implementations have utilized i-Vector methods. An emerging novel approach to speaker identification has been the application of convolutional neural networks.

==

ChangeLog:

2018-10-10: Small changes in structure
2018-10-09: Faculty and department name adjusted

Acknowledgment

We would like to show our deepest gratitude to all the people who offer help to this project.

Contents

1 Introduction

This project will investigate the marginal cost and methods of adding new speakers into a convolutional neural network trained for speaker identification. The goal is to identify an approach that will allow additional speakers to be added to the network and additionally be identified when they issue a new, previously unheard utterance. The project will investigate different approaches to adding additional speakers to a trained neural network and attempt to quantify the additional loss incurred, what, if any, effect on accuracy this has, and the training time such an approach would require. Ideally, this could result in systems that could learn the identities of their users and have applications in both human-computer interraction and security areas.

1.1 Problem statement

Given a pre-trained convolutional neural network that is able to identify a speaker given a 3-second sample utterance or a series of 3-second sample utterances add additional speakers to the neural networks corpus incrementally such that the network learns to identify new speakers.

1.2 Contributions and results

In the process of adding speakers we want to ensure that both the additional training time required is minimized and that the accuracy of the network remains high.

1.3 Structure of the thesis

This report will first go over the architecture and pre-processing pipeline for the neural network and give a brief overview of the data which the network was trained on. We will additionally summarize the results of previous experiments conducted by the researchers and finally, give an overview of what changes we intend to make to the network in order to allow for it to be trained incrementally.

2 Methods

The dataset used in this project is the VoxCelebA dataset, which consists of a corpus of .wav files generated from youtube with labeled celebrity speaker identities. The speakers have been hand labeled along with their gender and nationality. There are, in total 148,642 utterances divided among 1,211 speakers in the training dataset. The dataset contains a good separation of male and female speakers, although the majority of the speakers hail from english speaking countries, which may lead to some bias.

The convolutional neural network used in this experiment is a state of the art design which consists of 3 components, a pre-training layer which, in addition to generating 3-second windowed samples from the longer .wav files applies some pre-processing steps to make the resultant data more amenable to training the neural network. A ResNet layer which learns a representation of the resultant utterance and outputs a feature vector, and a self-attention/decision layer that applies a weighting to the resultant feature vector and applies a probability to the speaker's identity.

2.1 Pre-processing

Given that the audio samples are of a variable length in the training set, it is necessary to cut them down into 3-second chunks for further processing. There are also some mathematical conversions applied to the audio signal in order to better emphasize features relative to training and limit the influence of approximation errors.

Pre-emphasis

Signal energies in speech identification tasks tend to drop off at higher frequencies. The exact rate of decrease tends to vary among speakers and environmental conditions but a good rule of thumb is a dropoff of 2 dB/kHz. This presents a problem for finite discretization of a signal during a fourier transform since floating point values tend to lose precision given large variations in min and max values. To compensate for this fact, a linear pre-emphasis filter can be applied to the speech signal in the time domain to boost the signal energies in higher spectral ranges. The formula is:

$$\forall X \in X, x[n] = x[n] - \alpha x[n-1]$$

where α is a pre-determined constant.

Framing

Since the speech signals occur in files of varying length, the speech signals must be windowed into 3-second chunks to be fed into the neural network. The effect of doing so is essentially a convolution with a function equal to one in the sample range and zero elsewhere. This can lead to some undesirable distortions of the spectrum whent he signal is discretized. Specifically, since the fourier transform doesn't know about the remaining signal outside of the window it may attempt to compensate for very low frequencies that the window does not capture by skewing other higher frequencies to compensate for that part of the signal. This leads to a noisy signal with too much energy in the higher frequency bands that is not "real" signal energy.

Hamming Window

The hamming window is an attempt to suppress the noise create by attempting to sample an infinite signal in a finite window. It essentially attempts to generate some "counter-noise" to cancel out the noise resulting from the windowing function. A variety of window functions have been proposed in an attempt to counteract this high frequency noise but one of the most widely used is the hamming window:

$$H(\theta) = 0.54 + 0.46\cos[\frac{2\pi}{N}n]$$

While a mathematical proof of this formula is outside the scope of this paper, suffice it to say that this results in a signal that is a more pure approximation the frequencies actually contained in the window and reduces the impact of frequencies that are too low to be detected in such a finite window.

Fourier Transform

The fourier transform is used to obtain the frequency components and signal energies associated with these frequency components given a signal in the time domain. It is widely used in a variety of signal processing tasks. Most computers use an algorithm to compute the fourier transform known as the fast fourier transform that takes advantage of the orthogonality of the transform in different dimensions. This brings the time complexity of the transformation down from $O(n^2) to O(n \log n)$.

Log-Mel Features

The final step in the pre-processing pipeline is to apply emphasis to the signal to better approximate how it would be perceived by a human listener. Since, in this case, the network is being trained to identify human speech patterns it makes sense to modify the signal such that it is more in line with human auditory perception. The mel-scale is based on research into the human auditory system and applies a transformation to the input signal such that sounds of equal distance from eachother also "sound" as if they are equal distance, ie, if one tone is twice as far away from some base tone as another tone then the tone would sound twice as high. The formula is:

$$m[f] = 2595 \log 10(1 + \frac{f}{7000})$$

2.2 ResNet Architecture

The resnet, short for Residual Network, is a state of the art convolutional neural network that attempts to partially solve the vanishing gradient problem by introducing short connections between different layers of the network, thereby lower level information to flow directly to a higher level activation function. The network is able to consider information inputs from two layers simultaneously. This also provides a shorter path during backpropagation that mitigates some of the effects of the vanishing gradient. While ResNets have been primarily applied as feature detectors in visual computing applications, they are also finding increasing usage in auditory processing of spectral graphs. Although speech patterns do not contain the same types of local structure as images, the patterns can be identified in much the same way for both types of information.

The Resnet inroduces the concept of an "identity shortcut connection" that skips one or more layers. This partially solves the vanishing gradient problem since it provides a shorter alternate path for gradients to propagate without impacting the overall network performance. The argument of the origional authors was that stacking residual layers should not reduce network performance since the network, in the worst case, could use an identity mapping for all the shortcuts and simply achieve the same performance as before. In this case, this would indicate that a deeper network should not produce more error than it's shallower counterpart. Much like a Long-Term Short-Term network whose "forget gate" controls how much information is transfered from one time-step to the next, the ResNet architecture learns how to control how much information from lower layers is passed to higher layers in the network.

2.3 Self-Attention Layer

Self-Attention layers have gained increasing attention since Vaswani et al wrote the seminal paper "Attention is All You Need" which advanced the state of the art in sequence transduction models using solely attention cells as the building blocks and achieved outstanding results. However, attention layers have been used for a long time previously in RNN and CNN architectures to provide an efficient mechanism to relate distant dependencies. In speech comprehension, this would take the form of relating different terms in a long sentence. However, in our case, we are attempting to extract sections of the feature vector which are most relevant to the task of speaker classification.

Self-attention, sometimes called intra-attention, is an attention mechanism relating different positions of a single sequence in order to compute a representation of the sequence. An attention function can be seen as mapping a set of key-value pairs to an output, where query, keys, values, and outputs are all vectors. The output is a weighted sum of the input vectors, where each value is computed by a probability function to the key. The self-attention layer used in our network is a Scaled Dot-Product Attention layer, our key in this case is simply a learned series of weights but we add a second layer of keys to further segregate the input feature vector.

The self-attention layer produces, what is in essence, a weighting to apply to different features so that only the most relevant features of the speech sample are considered in classifying the speaker. Since applying too much weight to many different features would defeat the purpose of having an attention layer, a factor is added to the loss function which encourages the network to limit the number of attention hops it performs, keeping the features considered limited.

3
Training

3.1 Loss Function

By calculating the loss between the predicted value and the true value, relying on the back-propagation algorithm, the loss is fed back-ward layer by layer from the last layer. The parameters of each layer are updated, and the parameters are updated again to feed forward, and so on, until the network model Convergence, so as to achieve the purpose of training the model. Generally, Convolution Neural Network is a hierarchical model which uses convolution and other operations as the basic units on raw data (x^1 in the following formula) layer by layer, such as RGB image, raw audio data, etc ending with the calculation of the loss function and each layer of data as a 3-dimensional tensor.

$$x^1 \longrightarrow \omega^1 \longrightarrow x^2 \longrightarrow \omega^2 \cdots \longrightarrow x^{L-1} \longrightarrow \omega^{L-1} \longrightarrow \omega^L \longrightarrow z$$

If y is the ground truth corresponding to input x^1, then the loss function is expressed as the following formula, and the parameter in the function L(·) is ω^L.:

$$z = L(x^L, y)$$

In practical applications for different tasks, the form of the loss function also changes. Taking the regression problem as an example, the commonly used $l2$ loss function can be used as the objective function of the convolution network:

$$z = L_{regression}(x^L, y) = \frac{1}{2}\left\|x^L - y\right\|^2$$

For classification problems, the objective function of the network often uses the cross-entropy loss function:

$$z = L_{classification}(x^L, y) = -\sum_i y_i \log(p_i)$$

$$p_i = \frac{e^{x_i^L}}{\sum_{j=1}^{C} e^{x_j^L}} (i = 1, 2, ..., C)$$

C: classification task function There are two types of objective functions in the classical prediction tasks, which are classification and regression.

3.1.1 Objective Function of Classification Task

For classification assignments, suppose a classification task has N training samples, the input feature for the i-th sample of the final classification layer of the network is xi, and its corresponding true label is $y_i \in 1, 2, ..., C$, and $h = (h_1, h_2, ..., h_C)^T$ is the final output of the network, that is, the prediction result of sample i, where C is the number of classification tasks. The true label of the sample corresponds to a one hot vector. The cross entropy loss function is also called the softmax loss function.

$$L_{cross\ entropy\ loss} = L_{softmax\ loss} = -\frac{1}{N} \sum_{i=1}^{N} \log(\frac{e^{h_{y_i}}}{\sum_{j=1}^{C} e^{h_j}})$$

The hinge loss function is widely used in vector machines:

$$L_{hinge\ loss} = \frac{1}{N} \sum_{i=1}^{N} \max\{0, 1 - h_{y_i}\}$$

The central loss function also puts some attention on reducing inter-class differences while considering the distance between classes.

$$L_{center\ loss} = \frac{1}{2} \sum_{i=1}^{N} \frac{1}{2} \|x_i - C_{y_i}\|_2^2$$

Where C_{y_i} is the mean ("center") of all depth features of the y_i class, hence the name "central loss function". Intuitively, 9.11 forces all samples belonging to the y_i class not to be too far away from their center, otherwise the penalty will be increased. In actual use, since the central loss function itself considers the difference within the class, the central loss function should be used in conjunction with other loss functions that mainly consider the distance between classes, such as the cross-entropy loss function. The final objective function of the network is as follows:

$$L_{final} = L_{cross\ entropy\ loss} + \lambda L_{center\ loss}\ (h, y_i) = -\frac{1}{N}\sum_{i=1}^{N} \log(\frac{e^{h_{y_i}}}{\sum_{j=1}^{C} e^{h_j}}) + \frac{\lambda}{2}\sum_{i=1}^{N}\frac{1}{2}\left\| x_i - C_{y_i}\right\|_2^2$$

3.1.2 Objective Function of Regression Task

The true label of the sample also corresponds to one vector, but the difference is that each dimension of the true label is a real number instead of a binary value (0 or 1). In the regression task, the result of the regression is some integers or real numbers and there is no prior probability density distribution. The commonly used losses are l1 loss and l2 loss.

The prediction error is used to measure how close the predicted value of the model is to the true label. Assuming that the true label corresponding to the i-th input feature xi in the regression problem is $y_i = (y_1, y_2, ..., y_M)$, and M is the total dimension of the label vector, then lit represents the network regression predicted value on sample$i$ (y^i). The prediction error (also known as the residual) of the t-th dimension from its true label:

$$l_t^i = y_t^i - \hat{y}_t^i$$

Mean absolute loss (MAE) is also called l1 Loss, which uses the absolute error as the distance:

$$L_{l_1\ loss} = \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{M}\left|l_t^i\right|$$

Due to the sparseness of l1 loss, in order to penalize larger values, it is often added as a regular term to other losses as a constraint. The biggest problem of l1 loss is that the gradient is not smooth at the zero point, which causes the minimum value to be skipped. Mean Squared Loss/ Quadratic Loss (MSE loss) is also called l2 loss, or Euclidean distance, which uses the sum of squared errors as the distance:

$$L_{l_2\ loss} = \frac{1}{N}\textstyle\sum_{i=1}^{N}\sum_{t=1}^{M}(l_t^i)^2$$

L2 loss is also often used as a regular term. When the predicted value is very different from the target value, the gradient is easy to explode because the gradient contains x-t.

In this paper, the proposed network is asked to classify speakers using a multi-class cross entropy objective function, which is commonly used for image object detection and speaker recognition in neural networks. Previous experiments were trained to predict speaker label from frames while in the paper predicting speakers from variable-length segments. Given a dataset with N training examples from K speakers, with a speech segment consisting of T input frames$x_1^(n),x_2^(n),\ldots\ ,x_T^(n)$. The prediction probability of the deep network model for the k-th speaker is defined as$p(y_k|x_{1:T}^{(n)})$. The cross-entropy objective function is defined as below:

$$l = -\textstyle\sum_{i=1}^{N}\sum_{k=1}^{K}d_{n_k}(\log(p(y_k|x_{1:T}^{(n)}))$$

where the quantity d_{n_k} is 1 if the speaker label for the n-th training segment is k; otherwise, it is 0.

3.2 Hyper parameter Training

3.2.1 Setting Hyper parameter

Before building the entire network architecture, specifying various hyper parameters related to the network structure is a must: input image pixels, number of convolution layers, convolution kernel related parameters, etc. When using convolutional neural networks to process image problems, different input images are output with the same specifications, and at the same time, it is convenient for GPU devices to parallelize, and the images will be compressed to 2^n size uniformly. The hyper parameters of the convolution layer mainly include the size of the convolution kernel, the step size of the convolution operation, and the number of convolution kernels. Similar to the size of the convolution kernel, the kernel size of the convergence layer is generally set to a smaller value, which plays a role of "down sampling". In the paper, there are several hyper parameters defined in the self-attention mechanism. The hyper-parameter n_k corresponds to the number of attention hops.

3.2.2 Training

When training a convolutional neural network, although the training data is fixed, due to the mini-batch processing training mechanism, we can train the data set before each epoch of the model is trained Randomly shuffled to ensure that the data "sees" in the same batch of different epochs of the model is different. Such processing will not only increase the rate of model convergence, but at the same time, compared to a model trained in a fixed order, this operation will slightly improve the prediction result of the model on the test set. Another key setting during model training is the model learning rate. An ideal learning rate will promote model convergence, while an undesirable learning rate will even directly lead to the loss of the model's direct objective function. Regarding the optimization algorithm selection of model parameters. The stochastic gradient descent method is currently the most commonly used network training method.

3.3 Grid Search

Grid search is a model hyperparameter optimization technology, which is often used to optimize three or less hyperparameters. It is essentially an exhaustive method. For each hyperparameter, the user chooses a smaller finite set to explore. Then, the Cartesian product of these hyperparameters obtains several sets of hyperparameters. Grid search uses each group of hyperparameters to train the model, and selects the hyperparameter with the smallest validation set error as the best hyperparameter.

Grid search uses an exhaustive approach, and its computational complexity will increase exponentially with the size of the hyperparameters that need to be optimized. Therefore, this method is only suitable for small-scale hyperparameter optimization problems. Improvement: When the hyperparameter scale is large, random search will be a more efficient hyperparameter optimization method. At the same time, because the pure grid search is relatively extensive for the optimization of hyperparameters, the more fine-grained restriction and processing of the grid search strategy and scope may be useful for improving the efficiency of the grid search. In this paper, grid search is used to determine a suite of hyper-parameters such as weight decay, the size of the embedding layer, the dropout probability, and the maximum gradient norm. All experiments in the paper are implemented by the widely used deep learning tool TensorFlow. The batch size is set as 128 and train neural networks on one NVIDIA GTX 1080 Ti GPU. In the paper, the Adam optimization algorithm is used. The Adam optimization algorithm is an extension to stochastic gradient descent that has recently seen broader adoption for deep learning applications in computer vision and natural language processing. The attractive benefits of using Adam on non-convex optimization problems are being straightforward to implement, being computationally efficient, little memory requirements, being invariant to diagonal rescale of the gradients, and being appropriate for non-stationary objectives. Adam is different to classical stochastic gradient descent. Stochastic gradient descent maintains a single learning rate (termed alpha) for all weight updates and the learning rate does not change during training. That is to say, learning rate is maintained for each network weight (parameter) and separately adapted as learning unfolds. The parameters of the Adam optimizer are alpha (learning rate or step size, the proportion that weights are updated), beta1(the exponential decay rate for the first moment estimates), beta2(the exponential decay rate for the second-moment estimates), epsilon(a small number to prevent any division by zero in the implementation) The parameter of the Adam optimizer is used with $\beta 1 = 0.9$, $\beta 2 = 0.98$, $\epsilon = 10^(- 9)$. Then adopting the warm-up process for varying the learning rate increases the learning rate linearly for the first predefined training steps and then decreases it proportionally.

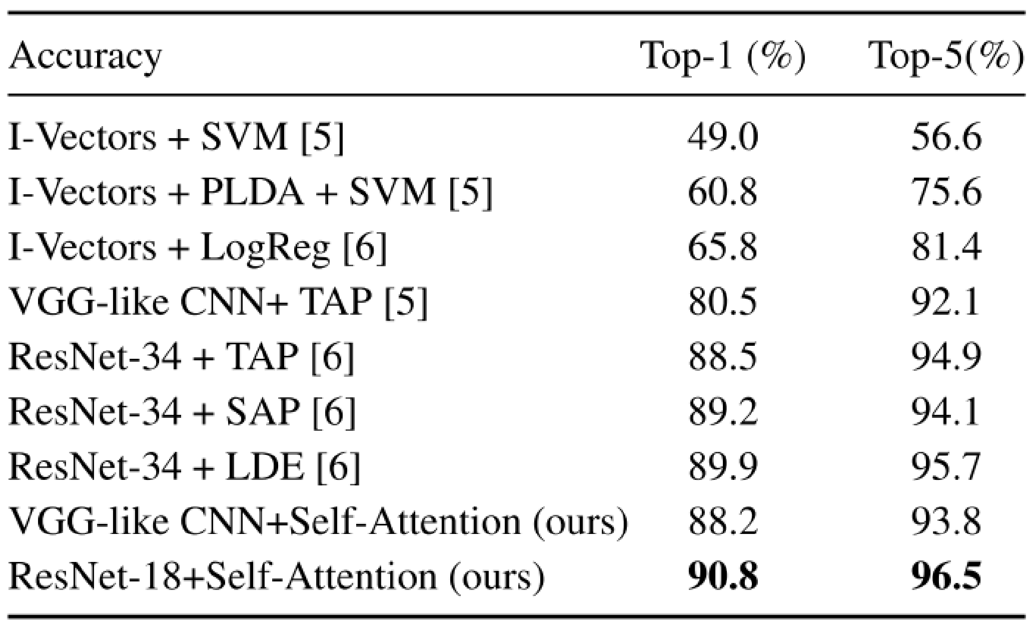
4 Previous Results

In the essay, researchers conduct experiments focusing on various elements of the speaker identification system. They quantify the testing result in the form of top-1 and top-5 accuracy, which suggests the probability of the right speaker existing in the most or 5 most possible labels. The experimental results show the superiority of self-attention mechanism, and lead us to focus on FBank acoustic feature and average pooling layer.

4.1 Results for speaker identification

From the perspective of methods, researchers compare CNN with self-attention mechanism, other CNN baselines and traditional i-vector-based methods on VoxCeleb database. While i-vector-based methods can only achieve a maximum top-1 accuracy of 65.8%, VGG-like CNN can get a 80.6% accuracy and ResNet-34 get a 89.9%. Our proposed systems reach 88.2% and 90.8% of the top-1 accuracy respectively and 93.8% and 96.5% of the top-5 accuracy. Generally speaking, traditional i-vector-based methods have the worst performance and CNN with self-attention mechanism has the best.

It's obvious that ResNets outperforms VGG in speaker identification task, thus suits our future experiments better. ResNet-18 with self-attention mechanism outperforms others significantly, reaching 90.8% top-1 accuracy. The outstanding performance also suggests the importance of self-attention mechanism in improving identification accuracy.



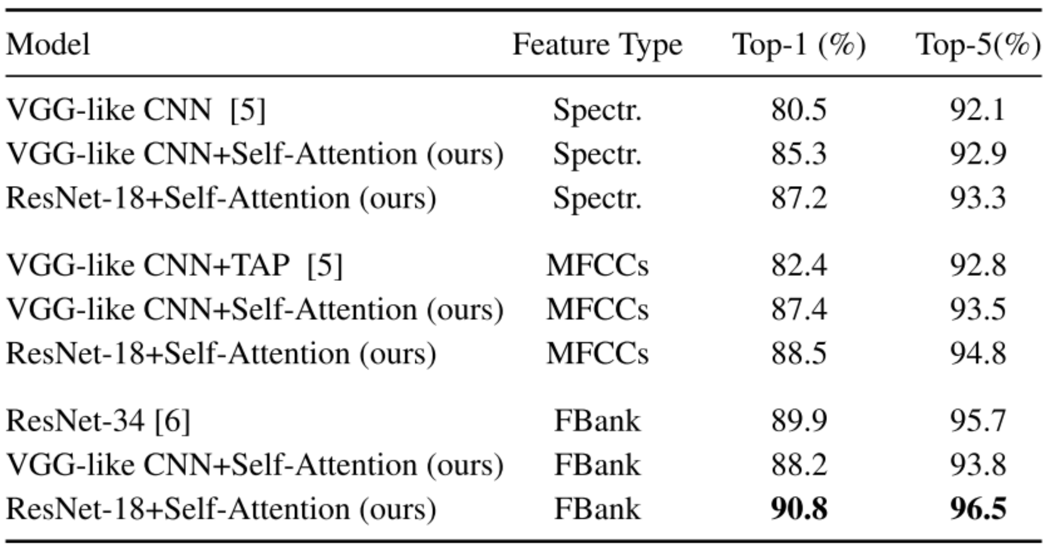
Accuracy	Top-1 (%)	Top-5(%)
I-Vectors + SVM [5]	49.0	56.6
I-Vectors + PLDA + SVM [5]	60.8	75.6
I-Vectors + LogReg [6]	65.8	81.4
VGG-like CNN+ TAP [5]	80.5	92.1
ResNet-34 + TAP [6]	88.5	94.9
ResNet-34 + SAP [6]	89.2	94.1
ResNet-34 + LDE [6]	89.9	95.7
VGG-like CNN+Self-Attention (ours)	88.2	93.8
ResNet-18+Self-Attention (ours)	**90.8**	**96.5**

Figure 4.1: The results for speaker identification on VoxCeleb.

4.2 Results for different acoustic features

Compared with spectrograms and MFCCs, FBank features lead to the highest accuracies. Using the FBank features, our proposed architecture reaches a top-1 accuracy as high as 90.8% and top-5 accuracy as high as 96.5%. Among all the three feature types, self-attention mechanism improves the performance obviously, especially for the top-1 accuracy.

These findings prompt us to adopt FBank features in the following experiments. FBank features can be generated after the pre-processing stage.



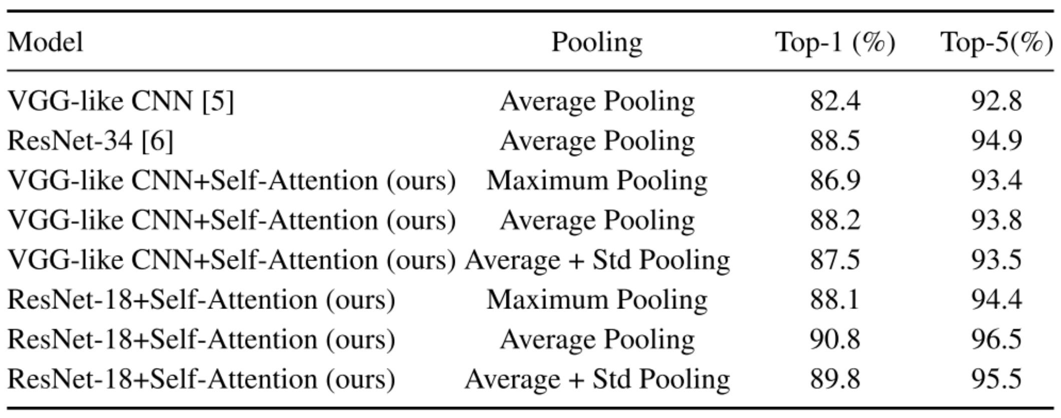
Model	Feature Type	Top-1 (%)	Top-5(%)
VGG-like CNN [5]	Spectr.	80.5	92.1
VGG-like CNN+Self-Attention (ours)	Spectr.	85.3	92.9
ResNet-18+Self-Attention (ours)	Spectr.	87.2	93.3
VGG-like CNN+TAP [5]	MFCCs	82.4	92.8
VGG-like CNN+Self-Attention (ours)	MFCCs	87.4	93.5
ResNet-18+Self-Attention (ours)	MFCCs	88.5	94.8
ResNet-34 [6]	FBank	89.9	95.7
VGG-like CNN+Self-Attention (ours)	FBank	88.2	93.8
ResNet-18+Self-Attention (ours)	FBank	**90.8**	**96.5**

Figure 4.2: The results of the three most frequently used acoustic features for speaker identification on VoxCeleb.

4.3 Results for temporal pooling layers

In our proposed architecture, a temporal pooling layer is inserted between the self-attention layer and the last fully connected layer. There are three common types of temporal pooling layers, i.e., average pooling, maximum pooling and combination of average and standard deviation pooling. Average pooling always gets the best result regardless of the CNN type, reaching 90.8% top-1 accuracy and 96.5% top-5 accuracy for ResNet-18 variant.

These findings prompt us to adopt average pooling in the following experiments.



Model	Pooling	Top-1 (%)	Top-5(%)
VGG-like CNN [5]	Average Pooling	82.4	92.8
ResNet-34 [6]	Average Pooling	88.5	94.9
VGG-like CNN+Self-Attention (ours)	Maximum Pooling	86.9	93.4
VGG-like CNN+Self-Attention (ours)	Average Pooling	88.2	93.8
VGG-like CNN+Self-Attention (ours)	Average + Std Pooling	87.5	93.5
ResNet-18+Self-Attention (ours)	Maximum Pooling	88.1	94.4
ResNet-18+Self-Attention (ours)	Average Pooling	90.8	96.5
ResNet-18+Self-Attention (ours)	Average + Std Pooling	89.8	95.5

Figure 4.3: Results of the speaker identification experiment on VoxCeleb for different common temporal pooling layers when the FBank features are extracted for the experiments.

5 Research question

Our preliminary research question is adding new speakers to the trained network for identification. This project aims at not only the training and the identification on basic dataset, but also on additional speaker utterance. We will try to add additional speakers to the trained CNN model, adjusting parameters accordingly to achieve the lowest loss and highest identification accuracy. To be more specific, we would like to quantify the loss incurred by the newly added data and the effect on final result. If all goes well, we can compare the performance of the proposed CNN variants with other CNN baselines and traditional i-vector-based methods.

The ability to learn and identify new utterance will be critical in practical applications like smart homes and smart vehicles. That's why we try to focus our project on it.

It is worth noting that it is possible to cluster an unknown speaker through the activations of an upper (dense or softmax) layer of a pre-trained identification CNN as a feature vector. Randomly chosen unknown speakers from VCTK database can produce a clear separation for the resulting vectors of last layers. This provides the basis for our research question.

Bibliography

[1] Liu, W., Wen, Y., Yu, Z., Yang, M.: Large-margin softmax loss for convolutional neural networks. In: ICML. Volume 2. (2016) 7

[2] Wu, J.: Introduction to convolutional neural networks. National Key Lab for Novel Software Technology. Nanjing University. China **5** (2017) 23

[3] An, N.N., Thanh, N.Q., Liu, Y.: Deep cnns with self-attention for speaker identification. IEEE Access **7** (2019) 85327–85337

[4] Nagrani, A., Chung, J.S., Zisserman, A.: Voxceleb: a large-scale speaker identification dataset. arXiv preprint arXiv:1706.08612 (2017)

[1] [2] [3] [4]

A
Sources

Appendix contains important source code snippets.

```java
public class Hello {
    public static void main(String[] args) {
        System.out.println("I love machine learning");
    }
}
```

Listing A.1: Lines of code

List of Figures

Name: Ethan Swistak
Jiayi Liu
Yile Bai Matriculation number: 1052792
1064804
1064812

Honesty disclaimer

I hereby affirm that I wrote this thesis independently and that I did not use any other sources or tools than the ones specified.

Ulm, ...

Ethan Swistak

Jiayi Liu

Yile Bai