Ville Jokinen

Few-shot learning for speaker recognition

Master's Thesis in Information Technology

June 2, 2021

University of Jyväskylä

Faculty of Information Technology

Author: Ville Jokinen

Contact information: `ville.o.jokinen@student.jyu.fi`

Supervisors: Tommi Kärkkäinen, and Joonas Hämäläinen

Title: Few-shot learning for speaker recognition

Työn nimi: Puhujan tunnistaminen FSL-menetelmillä

Project: Master's Thesis

Study line: Software and Telecommunication Technology

Page count: 63+6

Abstract: This thesis sets out to compare recent methods in speaker recognition, from a small amount of data. Speaker recognition aims to distinguish speakers from within audio data containing speech, the use cases include for example speaker diarization and voice biometric authentication. The scope is limited to identification, two samples from one or two distinct previously unknown speakers are provided. With the aim being to identify whether the two samples are spoken by the same speaker. Additionally, the accuracy of networks trained on English speech on Finnish speech is also measured. For which a new dataset, suitable for benchmarking speaker recognition, consisting of Finnish speech was developed from an existing speech recognition dataset.

The results show that the latest methods perform very well. However, to achieve the best results it is apparent that more training data is required, than what was used in this thesis. The methods generalized to Finnish speech, despite being trained with English speech. Additionally, interesting observations are made regarding the parameters chosen for training. In addition to comparing different methods, the effects of different number of speakers used for training, various sample lengths and data augmentation are also compared.

Keywords: machine learning, speaker recognition, speaker identification, few-shot learning

Suomenkielinen tiivistelmä: Tutkielman tavoitteena on vertailla uusimpia koneoppimiseen pohjautuvia menetelmiä puhujan tunnistamiseen vähäisellä datan määrällä. Puhujan tunnistamisessa tavoitteena on tunnistaa eri puhujat äänidatasta, sen käyttötarkoituksiin sisältyy mm. puhujan diarioiminen ja biometrinen tunnistus äänen avulla. Tutkielma rajoittuu puhujan tapaukseen, jossa käytettävissä on kaksi lyhyttä nauhoitetta, joko yhdeltä tai kahdelta, ennestään tuntemattomalta puhujalta. Joiden pohjalta pyritään tunnistamaan, sisältävätkö nauhoitteet puhetta samalta puhujalta. Lisäksi tutkielmassa tutkitaan Englanninkielisellä puheella koulutettujen neuroverkkojen tarkkuutta Suomenkieliseen puheeseen sovellettuna. Johon kehitetään sopiva datasetti Suomenkielisen puhekorpuksen pohjalta.

Tutkielman tulokset osoittavat uusimpien menetelmien suoriutuvan erinomaisesti. Vaikkakin parhaiden tuloksien saavuttaminen osoittautui vaativan enemmän koulutusdataa kuin mitä tutkielmassa käytetään. Menetelmät yleistyvät hyvin myös suomenkieliselle puheelle sitä huolimatta, että koulutuksessa käytettiin vain englanninkielistä puhetta. Lisäksi tuloksien pohjalta tehdään mielenkiintoisia huomioita vertailuun valittujen muuttujien osalta, joita käytetään neuroverkkojen koulutuksessa. Vertailussa oli menetelmien lisäksi koulutusdatan puhujien määrä, puhe esimerkkin pituus ja äänidatan augmentointi.

Avainsanat: koneoppiminen, puhujan tunnistaminen

Glossary

$\textcircled{\Sigma}$	Summation over the 3rd columns of a tensor.
$\boxed{\text{FC}}$	Fully-connected layer.
$\boxed{\text{conv}, N}$	2D convolution with kernel size N.
$\odot$	Hadamard (element-wise) product.
ASP	Attentive Statistics Pooling, a network layer used for aggregating temporal frames.
BCE	Binary cross-entropy, a loss function.
CNN	Convolutional neural network, a type of neural network.
DCF	Detection cost function.
DCT	Discrete cosine transform.
DET	Detection error trade-off.
DNN	Deep neural network, an artificial neural network consisting of multiple layers.
EER	Equal error rate, an evaluation metric.
FNR	False negative rate.
FPR	False positive rate.
FRR	False rejection rate.
FSL	Few-shot learning.
GPU	Graphics processing unit.
MAML	Model-Agnostic Meta-Learning.
MFCC	Mel-frequency cepstral coefficients.
minDCF	Minimum detection cost function.
ReLU	Rectified Linear Units, a hidden unit in a neural network or an activation function.
ResNet	Residual network, a type of deep convolutional neural network.
ResNetSE34	34-layer Residual network with squeeze and excitation modules.
RIR	Room impulse response.
ROC	Receiver operating characteristics.

SAP	Self-Attentive Pooling.
SNN	Siamese neural network.
SNR	Signal to noise ratio.
STFT	Short-time Fourier transform.
TPR	True positive rate.
VLAD	Vector of Locally Aggregated Descriptors.

List of Figures

List of Tables

Contents

1 Introduction

The intent of this thesis is to investigate recent methods for speaker recognition, particularly focusing on methods for differentiating previously unknown speakers using single speech sample. Additional focus of the research is, to measure how these methods perform when the models are trained using both limited computational capabilities and limited data. The accuracy of the models trained with English speech is also measured for Finnish speech. For this a new dataset, suitable for benchmarking speaker recognition, consisting of Finnish speech was developed ad-hoc in this thesis.

The evaluation of the methods is performed using three distinct datasets. The results indicated that while latest methods achieve excellent results, the methods do appear to require large amount of training data and time to truly generalize well. The networks trained using metric learning with angular prototypical loss functions (Chung et al. 2020) showed better results across the board, than the end-to-end siamese neural network. Though the best methods trained for this thesis achieved good results with the test sets based on the same dataset as the training data, the methods struggled to perform with the other datasets. Interestingly there not much difference was observed in the prediction performance between Finnish and English speech. A surprising discovery was made regarding the optimal sample length, as models trained using 2 second samples appeared to perform better than the popular 4 second sample length.

Speaker recognition is complementary to speech recognition, as the focus is on answering the question *who* is speaking rather than *what* is spoken. It is also an essential component in voice biometric authentication and speaker diarization, where an audio stream is partitioned into segments by speaker (Beigi 2011). Recent research findings have shown good results with deep convolutional neural networks trained using large datasets (Kwon et al. 2020; Heo et al. 2020; Chung et al. 2020). Because these results were achieved using a dataset consisting of tens of gigabytes of audio data with hundreds of hours of training using multiple graphics processing units (GPU), replicating the results would likely also require large amount of data and computational resources. Thus it was chosen to investigate how these methods fare with less data and computation time.

The thesis begins with overview on machine learning in Chapter 2, detailing the relevant background on machine learning. Chapter 3 describes the problem space and covers signal representation and the evaluation of speaker recognition methods, while also including a light literature review of recent articles related to speaker recognition. In Chapter 4 the implemented methods and experiments are explained in detail. In Chapter 5 the results of the experiments are presented and analysed. Finally, Chapter 6 presents the conclusions drawn from the results.

2 Machine learning

This chapter covers machine learning topics relevant to this thesis, and is largely based on the *Deep Learning* textbook by Goodfellow, Bengio, and Courville (2016) with the exception of the portions covering ResNet and few-shot learning. According to the authors of the textbook, machine learning (ML) algorithms to can be divided into three key parts: class of tasks, performance measure and experience. Tasks can include for example classification, regression, anomaly detection, synthesis and sampling. The choice of a performance measure is generally task specific. For example for classification we could measure accuracy (proportion of correct outputs) or error rate.

ML algorithms experience a dataset, which is a collection of examples (or data points). Goodfellow, Bengio, and Courville (2016) categorize ML algorithms into two classes: supervised or unsupervised. In contrast to unsupervised learning, supervised learning dataset contains labels (or targets) that annotate the expected result for each example. Traditionally classification and regression have been considered as supervised learning problems. Regression is a type of task where the goal is to predict a numerical value from a given input. In this type of task the learning algorithm should output a function $f : \mathbb{R}^n \to \mathbb{R}$. A simple example would be predicting future values of stock prices based on past values.

In classification, the task is to specify which k categories the input belongs to, which means the learning algorithms task is to produce a function $f : \mathbb{R} \to \{1, \dots, k\}$. A simple example of this is object recognition, where the task is to identify an object in an image from a list of known objects. For example we could have a model trained to recognize whether the input image is a cat, a dog or a chair. A more advanced example of *binary classification*, or classification with only two classes, would be *open-set speaker identification*. In this task, the input is two audio samples, one featuring the target speaker and the other the test speaker. The output is an estimate on whether the two are the same. Where the output of f is a probability distribution over the 2 classes.

2.1 Deep feedforward networks

Goodfellow, Bengio, and Courville (2016) describes *feedforward neural networks* as having the goal of approximating some function $f*$. A feedforward network $y = f(x;\theta)$ maps input x to y, learning the value of the parameters (or weights) θ that result in the best approximation of $f*$. The term feedforward indicates the direction the information flows through the evaluated function from x to the output y, and that there are no feedback connections which would feed the output of the model back into itself.

Feedforward networks are typically represented by composing multiple functions together, forming *deep feedforward networks* where the depth comes from the multiple layers of functions. How the functions are composed together is described by a *directed acyclic graph*, in other words the network forms a linear chain of layers. The intermediate layers between the input layer and the output layer are called hidden layers. Fixed nonlinear functions, called activation functions, are used to compute the hidden layer values (for example sigmoid 2.5 and ReLU 2.7).

According to Goodfellow, Bengio, and Courville (2016), training a deep feedforward network can be done by the means of iterative gradient-based optimizers, that aim to minimize the cost function (or loss function). Iterative optimization is chosen, because the use of non-linear functions prevents the use of closed-form optimization. The authors say, that popular iterative optimizer choices include *stochastic gradient descent* (SGD), which is an extension of gradient descent, and *Adam*, an adaptive learning rate optimization algorithm. Also according to the authors, the choice of an optimizer appears to still be a matter of taste, as no consensus on optimal optimization algorithm choice has been made. In this thesis we use *AdamW* (Loshchilov and Hutter 2019), a recent variant of Adam which decouples the weight decay from the loss function.

According to Goodfellow, Bengio, and Courville (2016), *Negative log-likelihood*, or equivalently the *cross-entropy* (eq. 2.2) between the training data and the model distribution, is commonly used as the cost function when training neural networks using gradient-based optimization. For discrete variables, this is given by

$$\mathbb{E}_{x \tilde{} P}\big(f(x)\big) = \sum_{x} P(x) f(x) a \tag{2.1}$$

$$\begin{aligned} J(\theta) &= -\mathbb{E}_{x,y \tilde{} \tilde{p}_{data}} \log p_{\text{model}}(y|x) \\ &= -\sum_{x} \tilde{p}_{data}(y|x) \log p_{\text{model}}(y|x) \end{aligned} \tag{2.2}$$

where $\mathbb{E}_{x,y \tilde{} \tilde{p}_{data}}$ is the expected value (eq. 2.1) of the model with respect to the probability distribution of the training data $\tilde{p}_{data}$.

In this thesis, the following form of cross-entropy is used

$$H(x) = -\frac{1}{N} \sum_{i=1}^{N} \log x_i, \tag{2.3}$$

where N is the number of elements in the input vector x and the probability of encountering each element is presumed to be $\frac{1}{N}$.

Output unit is the final layer of the network, the output of which is then passed to the cost function. Depending on the implementation this can be a part of the cost function, part of the network or split between both. A *fully connected layer* (FC, or linear unit) is a simple output unit without nonlinearity

$$\text{FC}(x) = W^T x + b, \tag{2.4}$$

where W^T are the weights of the FC, x is the output of the network and b is a bias constant (Goodfellow, Bengio, and Courville 2016).

Sigmoid output unit is used when predicting values of a binary variable. The logistic sigmoid function is defined as

$$\sigma(x) = \frac{1}{1+e^{-x}}, \tag{2.5}$$

and the sigmoid output unit is $\sigma(W^T x + b)$. This gives us an *Bernoulli distribution*, and the network needs to predict only $P(y = 1|x)$ (Goodfellow, Bengio, and Courville 2016).

The *softmax* output unit can be used to represents a probability distribution over a discrete variable with N possible values, which makes it a common output for classifiers. This can be seen as a generalization of the sigmoid function. The softmax (or softargmax) function is defined as

$$\text{softmax}(x)_i = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}}, \tag{2.6}$$

where N is the number of elements in x. Like with sigmoid, the softmax output unit follows a linear unit (eq. 2.4) $\text{softmax}(W^T x + b)_i$. Which gives us a *categorical distribution* (also known as *multinoulli distribution*) (Goodfellow, Bengio, and Courville 2016).

Feedforward neural networks use hidden units for the output of hidden layers. All of the previously described output units are also suitable for use as hidden units. *Rectified Linear Units* (ReLU) are a common type of hidden unit. The ReLU activation function is defined as

$$\delta(z) = \max(0, z), \tag{2.7}$$

which is typically combined on top of a linear unit (eq. 2.4) resulting in $\delta(W^T x + b)$.

2.2 Convolutional neural networks

Goodfellow, Bengio, and Courville (2016) describe *convolutional neural networks* (CNN) as neural networks that use convolution in place of general matrix multiplication in at least one of their layers. Additionally the authors describe a typical convolutional layer consisting of three stages: convolution, detector and pooling. According to the authors, in machine learning libraries the convolution operation is generally implemented using a related function called *cross-correlation*. A look into the documentation[1] of the PyTorch library used in this thesis confirms this claim, thus only the relevant definitions of cross-correlation are detailed here.

The discrete cross-correlation for one-dimensional input $I : \mathbb{N} \to \mathbb{R}^n$ and kernel $K : \mathbb{N} \to \mathbb{R}^n$ is defined as

$$(I * K)(i) = \sum_m I(i+m)K(m), \tag{2.8}$$

where I and K are discrete functions defined on integer values of i. When the input $I : \mathbb{N}^2 \to \mathbb{R}^n$ and kernel $K : \mathbb{N}^2 \to \mathbb{R}^n$ are two-dimensional, the definition is

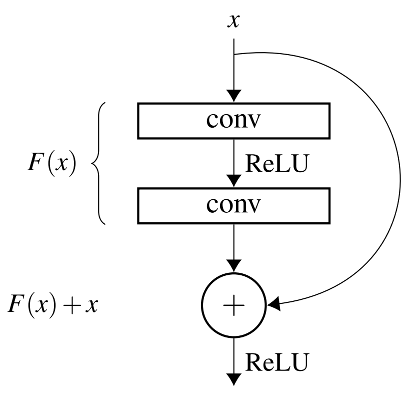
$$(I * K)(i, j) = \sum_m \sum_n I(i+m, j+n)K(m, n). \tag{2.9}$$

[1] https://pytorch.org/docs/1.7.1/generated/torch.nn.Conv2d.html

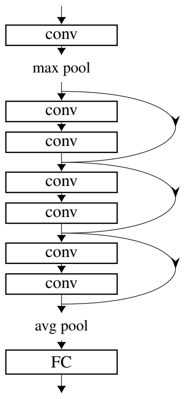
The second stage of a CNN, that Goodfellow, Bengio, and Courville (2016) called the detector stage, simply consists of passing each linear activation function through a nonlinear activation function such as ReLU. In the third stage, a pooling function is used. Pooling refers to the act of reducing output of the network using a summary statistic of nearby values. Generally, this means we take either the maximum (referred to as *max pooling*), or the average of the nearby values. What nearby values mean here varies depending on the method. For example it could mean using a kernel of some size to pool maximum values within the kernel, or taking a weighted average of an entire dimension of the output tensor.

Recently, convolutional networks have been shown to be performant at feature extraction in speaker recognition tasks, outperforming previous approaches using *vector quantization*, *gaussian mixture model-universal background model* (GMM-UBM), *i- and x-vector-based* methods (Nagrani, Chung, Xie, et al. 2020). Palanisamy, Singhania, and Yao (2020) even show, that *transfer learning* using CNN models, such as ResNet and DenseNet pre-trained for image classification, also perform well on spectrogram based audio classification tasks. Further Ding et al. (2020) propose an architecture search approach for automated search of suitable CNN architectures for speaker recognition tasks. Thus presently, CNNs appear to be a safe choice for implementing speaker recognition. Though other approaches have also been tried as an alternative to a CNN, as there have also been attempts at using capsule networks at different stages for speaker recognition (Anand et al. 2019; Hajavi and Etemad 2020).

Building upon CNN, He et al. (2016) present a residual learning framework and describe a type of deep convolutional network that they name *residual networks* (ResNet). Residual networks have proven to be performant, and have since become largely ubiquitous in recent publications utilizing convolutional networks (Xie et al. 2019; Nagrani, Chung, Xie, et al. 2020; Heo et al. 2020). He et al. (2016) construct residual networks by chaining multiple residual network blocks back to back, while gradually increasing the dimension of the convolution layer output. An example of the overall ResNet architecture can be seen in Figure 1b. The key idea introduced by the authors is the shortcut connection (see Figure 1a), that is used to add the original input x into the output of the layers inside the block $F(x)$.



(a) Residual network block.



(b) Minimal example of a 6-layer residual network.

Figure 1: General overview on the residual neural network architecture. Each residual network block has convolutional two layers, thus the example 1b has three blocks.

Residual networks have been further extended with the *squeeze-and-excitation* (SE) blocks proposed by Hu et al. (2020). This is shown by the authors to improve ResNet performance, and it has been adopted by recent ResNet implementations. The standard SE block $S(x)$, as described in Figure 2, is placed within the regular ResNet block as an additional step before the addition. The SE block introduces an tunable hyperparameter for reduction ratio r, which allows for tuning of the capacity and computational cost of the SE blocks. The output of function composition $(S \circ F)(x)$ is combined with the output of $F(x)$ using element-wise multiplication (the Hadamard product $\odot$), before adding the residual to original input x.

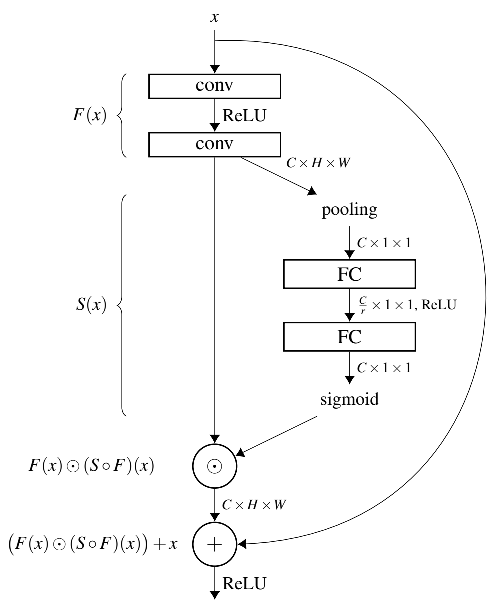


Figure 2: Residual network block with the squeeze and excitation extension. Where $C \times H \times W$ is a 3-D tensor with dimensions representing height H, width W and channels C.

Further variants of ResNet have been developed, in this thesis we chose two 34-layer variants commonly used in recent publications on speaker recognition (Xie et al. 2019; Nagrani, Chung, Xie, et al. 2020; Chung et al. 2020; Heo et al. 2020). Of these, the *thin-ResNet-34* uses reduced amount of channels in the residual blocks to reduce computational cost, and has been shown to perform better than larger ResNets. Additionally the *fast-ResNet-34* based on the thin variant was considered. In fast-ResNet-34, the input dimensions are smaller and the convolution strides have been tweaked, resulting even further reduction in computational cost (Chung et al. 2020).

2.3 Few-shot learning

In *few-shot learning* (FSL), the aim is to learn to generalize from only a few examples. It is common to refer to specific FSL problems using the term *k-way n-shot learning*, where k is the number of classes and n the number of examples in each class (see Figure 3). For example, *one-shot learning* can be written as k-way 1-shot learning. (Yaqing Wang et al. 2020).

According to Hospedales et al. (2020), the aim of *meta learning* is often simply described

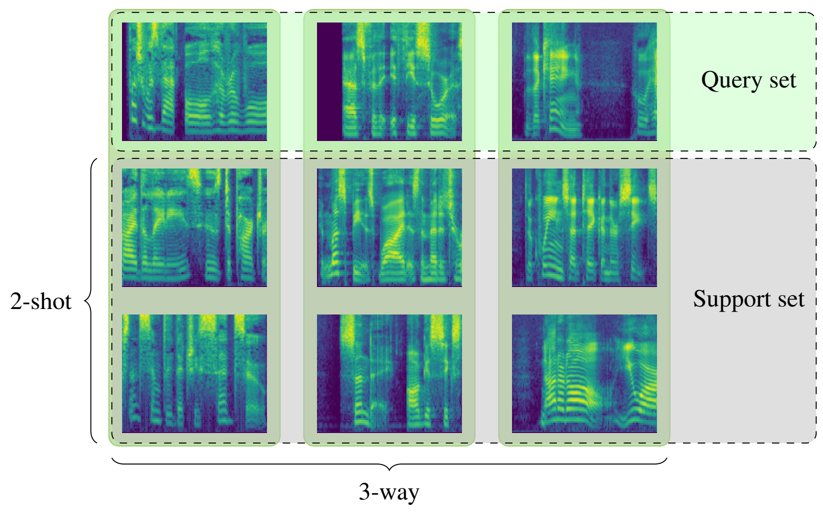


Figure 3: Example of input data for a single 3-way 2-shot learning step, illustrating the terminology used. In this figure *ways* correspond to speakers and *shots* to speech samples.

as *learning to learn*. However, for the needs of this thesis it is simply useful to know, that meta-learning is a larger paradigm that can also be applied for few-shot learning problems. Li, Sun, et al. (2020) reviews recent few-shot meta learning methods.

Siamese neural network (SNN), or *twin neural network*, is one the earliest neural network architectures applied to FSL. Originally, Bromley et al. (1993) developed the SNN for signature verification. Conceptually, an SNN consists of two sub-networks sharing weights, with the intent of extracting the features of the input (see Figure 4). The extracted features are then compared and a distance measure is obtained. When using the model to classify inputs there is no strict requirement for running each input through the network at the same time, thus it is possible to pre-compute the outputs of the sub-networks. Adding additional twins is also possible by adjusting the distance measurement to handle the additional inputs.

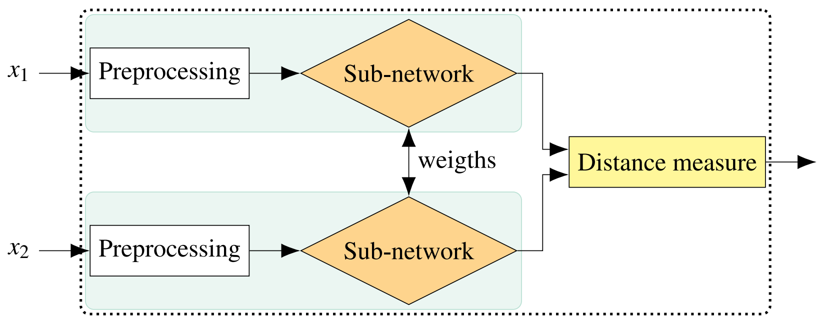


Figure 4: Overview of the siamese neural network architecture.

Distance measure generally consists of reducing the outputs of the sub-networks into one similarity metric, which is further reduced into single probability measure (ex. by adding a fully connected layer). Popular distance metrics include cosine similarity (eq. 2.10) and Euclidean distance (eq. 2.11). Alternatively, the distance measure could also itself be an *deep neural network* (DNN), as shown by Hajavi and Etemad (2020) in their paper applying a capsule networks (CapsNet) for learning the distance measure. In this thesis, we use a basic learning distance measure with a fully connected layer (eq. 2.12).

$$\cos(a,b) = \frac{a \cdot b}{||a||\,||b||} \tag{2.10}$$

$$d(a,b) = ||a-b||_2 = \sqrt{(a-b)^2} \tag{2.11}$$

$$\text{fcdist}(a,b) = FC(|a-b|) \tag{2.12}$$

Prototypical networks proposed by Snell, Swersky, and Zemel (2017) are an example of a more recent method for FSL, like SNN they are also metric based and utilize a distance measure for classification. The key idea being, that each class can be represented by the mean of its examples. The authors note, that training prototypical networks using higher number of classes than during test-time is not only possible but beneficial.

3 Speaker recognition

Speaker recognition focuses on answering the question *who* is speaking. Common applications of speaker recognition include voice authentication and speaker diarization. Beigi (2011) divides speaker recognition tasks into *identification*, *verification* (or *authentication*), *classification*, *segmentation*, *tracking* and *detection*. Speaker identification is further divided into either *closed-set* or *open-set* problems. Additionally speaker recognition, particularly speaker verification, can be dependent on the content of the speech. In this case, it is called *text-dependent*. In this thesis, the focus is on *text-independent open-set speaker identification*, an overview of which is show in Figure 5. This chapter is mostly based on *Fundamentals of Speaker Recognition* by Beigi (2011).

In open-set speaker identification, the test speaker is verified against one or more target speakers, if verification fails the test speaker is rejected. Whereas closed-set identification matches a test speaker to the closest match among known speakers. Closed-set identification would require a machine learning model to be specifically trained for a known set of speakers. Whereas open-set identification requires a model trained to measure the similarity of arbitrary speakers, which is a few-shot learning problem.

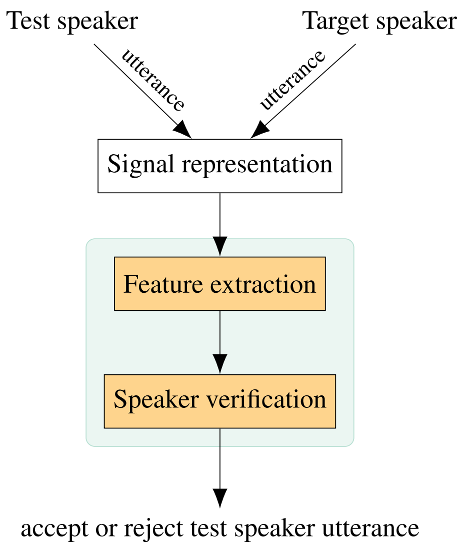


Figure 5: General overview on the open-set speaker identification process.

Speaker verification (or speaker authentication) refers to matching a test speaker to a known speaker, and can be considered a special case of open-set identification where there is only one target speaker. The key difference to open-set identification is that a speaker verification model could be trained to specifically recognize a single known speaker.

Speaker classification focuses classifying speakers to different classes such as gender or age. Speaker segmentation is concerned on identifying the sections of an audio stream in which each distinct speaker is speaking. Speaker detection refers to detecting one or more speakers in an audio stream. Speaker tracking is an extension of speaker detection, where the segments of each distinct speaker are also identified.

3.1 Signal representation

For using speech signal as an input to a neural network, it is first necessary to choose the representation of the audio signal. Beigi (2011) describes the signal representation of speech from first principles, but here we start directly with the first common representation. The speech waveform, obtained by sampling the amplitude of the source audio signal at a chosen frequency, is a discrete two-dimensional representation of the audio signal amplitude with respect to time (as seen in Figure 6a). Ravanelli and Bengio (2018) and Jung et al. (2020) show, that even raw waveform can successfully be applied as the input of a neural network.

The waveform can be pre-processed using various signal processing techniques. However, we only do *pre-emphasis* (see Figure 6b) in this thesis. Pre-emphasis is commonly used to improve speech feature extraction, by amplifying the magnitude of higher frequencies. Equation 3.1 shows pre-emphasis as implemented by Heo et al. (2020) and Zeghidour et al. (2018) using a first-order high-pass filter. Which is implemented as a convolution layer with one-dimensional cross-correlation (eq. 2.8) on input $I : \mathbb{N} \to \mathbb{R}^n$ with stride of 1. Where the kernel $K : \mathbb{N} \to \mathbb{R}^n$ of size 2 is initialized with weights $[-0.97, 1]$.

$$\begin{aligned}(I * K)(i) &= \sum_m I(i+m)K(m) \\ &= I(i)K(0) + I(i+1)K(1) \\ &= I(i+1) - 0.97I(i). \end{aligned} \tag{3.1}$$

Spectrogram is a three dimensional signal representation, exposing the frequency and the

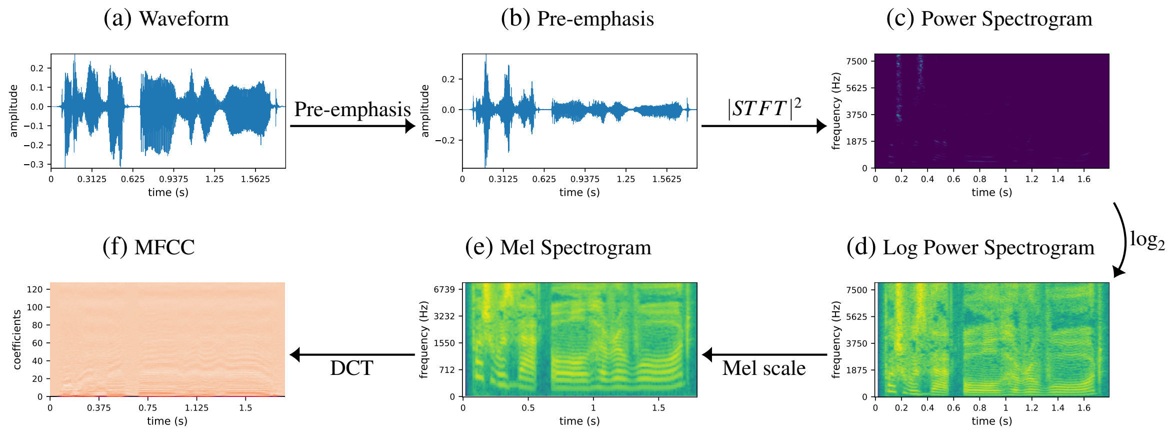


Figure 6: Various signal representations of the same audio sample, and coincidentally the process of computing the MFCC.

amplitude with respect to time. A power spectrogram can be obtained by taking the square of the *short-time Fourier transform* (STFT) of the waveform (as seen in Figure 6c and Figure 6d). The STFT is computed by sampling the waveform using a moving window function (such as Hann or Hamming window).

The *mel spectrogram* (as seen in Figure 6e), is a spectrogram where the frequencies are scaled to the *mel scale*, which is a perceptual scale developed in the late 1930s that has stuck around to this day. *Mel-frequency cepstral coefficients* (MFCC, see Figure 6f) is another commonly used representation, that can be obtained by applying the *discrete cosine transform* (DCT) on the mel spectrogram.

3.2 Evaluation

The performance of a speaker recognition system is commonly reported using *equal error rate* (EER), this practice originates from biometrics applications. EER is the intersection point of *false negative rate* (FNR, or FRR) and *false positive rate* (FPR, or FAR), in other words it is the point where the chance of chance of false positives and false negatives is equal. TPR, FPR and FNR can be inferred once the amount of real positive and negative labels (P and N) is known, and the number of true positive and false negative predictions

(TP and FN) have been measured.

$$TPR = \frac{TP}{P}$$
$$FPR = \frac{FN}{N}$$
$$FNR = 1 - TPR.$$

The Receiver Operating Characteristics curve (ROC curve), as seen in Figure 7, can be obtained by plotting FPR and *true positive rate* (TPR, or TAR) while varying the threshold at which a prediction is classified as positive or negative. (Beigi 2011; Mohri, Rostamizadeh, and Talwalkar 2018).

In the experiments of this thesis, an approximation of the EER (eq. 3.3) is computed by finding the index of the threshold θ nearest to the intersection point of FNR and FPR, then taking the arithmetic mean of FNR and FPR as follows

$$\theta = \arg\min |FNR - FPR| \tag{3.2}$$
$$EER = \frac{FNR(\theta) + FPR(\theta)}{2}. \tag{3.3}$$

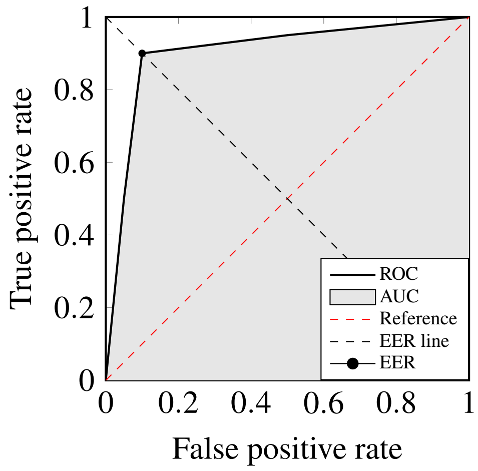


Figure 7: Example ROC curve, where EER is 10%. The reference line signifies where the ROC curve would land on in the case of random guess.

The *minimum detection cost function* (minDCF) was chosen as an additional evaluation metric, as it has been preferred over EER in speaker recognition challenges since 1997. It is based on the Detection Error Trade-Off (DET) curve, constructed from the same data as the ROC curve. (Beigi 2011). Sadjadi et al. (2020) defines the minDCF metric for the NIST 2020 CTS Speaker Recognition Challenge (NIST SRC), similar metric has also been used in earlier challenges. Nagrani, Chung, Huh, et al. (2020) also use this metric for VoxCeleb Speaker Recognition Challenge (VoxSRC) 2020. The minDCF (eq. 3.4) metric used is calculated for each trial using one test dataset. Thus it differs from the metric used in the NIST challenge, where evaluation is further divided into multiple trials and partitions.

The method used for computing the minDCF in this thesis is defined as follows

$$
\begin{aligned}
C_{Det}(\theta) &= C_{miss}P_{target}P_{miss}(\theta) + C_{fa}(1-P_{target})P_{fa}(\theta) \\
C_{Default} &= \min \begin{cases} C_{miss}P_{target} \\ C_{fa}(1-P_{target}) \end{cases} \\
\text{minDCF} &= \frac{C_{Det}(\arg\min C_{Det})}{C_{Default}},
\end{aligned} \tag{3.4}
$$

where, P_{miss} represents the FNR points on the ROC curve, and P_{fa} represents the FPR points. The DCF control parameters C_{miss}, C_{fa} and P_{target} can be used to tune the measurement for different scenarios. C_{miss} is the cost of a missed detection (false rejection), C_{fa} is the cost of a false-alarm (false positive). While P_{target} is the *a priori* probability that the test speaker is the target speaker. The values used for the DCF control parameters in this thesis are $C_{miss} = 1$, $C_{fa} = 1$ and $P_{target} = 0.05$, matching the values used by both the NIST SRC and VoxSRC. The parameters can also be adjusted to measure performance in different use cases, for example access control for which values such as $C_{fa} = 10$ and $P_{target} = 0.99$ could be used. (Vestman 2020; Sadjadi et al. 2020).

3.3 Recent work in speaker recognition

A light literature review was performed to gain an overview on the state of research on speaker recognition, mainly by searching for articles published between 2019–2020. This was done by searching with various keywords such as “few shot learning”, “one shot learning”, “speaker recognition” and “siamese neural network” from the Google Scholar search engine. Initially, 82 articles were found and sorted by subject, of which relevant articles were selected. This process was also repeated over the course of this thesis, ending with 260 articles.

Most of the articles found focused on utilizing image and video data. Common topics among these were classification, object detection, object tracking and segmentation. A small amount of articles on natural language processing utilizing FSL and SNN were also found, but these were not considered relevant to this thesis. As enough articles covering specifically audio data were found, the articles utilizing other types of data were not inspected more closely. The review uncovered 70 articles focused on audio data, of which 18 were specifically on speaker verification.

From the articles found, Shimada, Koyama, and Inoue (2020) and Yu Wang et al. (2020) utilized FSL methods for sound event detection. While Zhang and Duan (2017), Zhang, Pardo, and Duan (2019), Manocha et al. (2018), and Keren et al. (2018) made use of SNN and FSL methods for audio search. For more unique use cases, Mimilakis, Bryan, and Smaragdis (2020) utilized SNN for style transfer and Chen, Smith, and Yang (2020) utilized SNN for assessing loop compatibility in music production. Finally, Rozenberg, Aronowitz, and Hoory (2020), Li, Jiang, Li, et al. (2020), Chung et al. (2020), Ravanelli and Bengio (2018), Hajavi and Etemad (2020), Snyder et al. (2019), and Okabe, Koshinaka, and Shinoda (2018) did speaker recognition using SNN and FSL methods.

The following datasets were found to be used in the articles:

- VoxCeleb: a large-scale speaker identification dataset Nagrani, Chung, and Zisserman (2017). Used by Mishra (2020), Okabe, Koshinaka, and Shinoda (2018), Snyder et al. (2019), Chung et al. (2020), and Hajavi and Etemad (2020).
- TUT 2016: TUT database for acoustic scene classification and sound event detection Mesaros, Heittola, and Virtanen (2016) and Stowell et al. (2015). Used by Manocha et al. (2018), Pons, Serra, and Serra (2019), and Shimada, Koyama, and Inoue (2020).
- Librispeech: an ASR corpus based on public domain audio books Panayotov et al. (2015). Used by Keren et al. (2018), Ravanelli and Bengio (2018), and Li, Jiang, Li, et al. (2020).

- VoxCeleb2: Deep speaker recognition Chung, Nagrani, and Zisserman (2018). Used by Rozenberg, Aronowitz, and Hoory (2020) and Snyder et al. (2019).
- Vocalsketch: Vocally imitating audio concepts Cartwright and Pardo (2015). Used by Zhang and Duan (2017) and Zhang, Pardo, and Duan (2019).
- RAVDESS: Ryerson Audio-Visual Database of Emotional Speech and Song Livingstone and Russo (2018). Used by Feng and Chaspari (2020).
- eNTERFACE05: audio-visual emotion database Martin et al. (2006). Used by Feng and Chaspari (2020).
- CREMA-D: Crowd-sourced emotional multimodal actors dataset Cao et al. (2014). Used by Feng and Chaspari (2020).
- DAPS: Device and Produced Speech dataset Mysore (2015). Used by Mimilakis, Bryan, and Smaragdis (2020).
- Spoken Wikipedia Corpora Köhn, Stegen, and Baumann (2016). Used by Yu Wang et al. (2020).
- VOICES: Voices Obscured in Complex Environmental Settings Richey et al. (2018).

From the datasets encountered in the articles, a few potential datasets (see 1) that could be utilized in this thesis were chosen. From these, LibriSpeech and VoxCeleb datasets were chosen, in particular LibriSpeech (Panayotov et al. 2015) was chosen for training the models implemented in this thesis. Due to the authors having made available subsets of the dataset, with more reasonable size of 100 hours of speech from 251 speakers, weighing only 6.3 gigabytes on disk. VoxCeleb1 (Nagrani, Chung, and Zisserman 2017) was chosen for use in testing, as it was commonly used in articles on speaker recognition and would enable comparing the results to other research.

The neural networks implemented in the articles on speaker recognition were also lightly analysed, by comparing the key sections of the networks. Table 2 shows the structure some of the SNN implementations found. While Table 3 shows some of the non-SNN networks found. Finally, Table 4 lists some of the non-FSL networks found in the articles. In the tables, question mark signifies that information was not found in the article, while a dash signifies that the method was not used in the network.

	# of Speakers	Samples	Description
VoxCeleb1	1251	145265	utterances from YouTube
VoxCeleb2	6112	1128246	utterances from YouTube
LibriSpeech	2484	148688	audiobooks from LibriVox
VOiCES	200	999168	acoustically challenging conditions
RAVDESS	24	7356	emotional, 2 statements 1 song
CREMA-D	91	7442	emotional, 12 sentences

Table 1: Datasets that were identified to be potentially relevant for this thesis.

	Audio repr.	Network	Distance measure	Optimizer	Aggregation	Loss	Dataset
Hajavi and Etemad (2020)	?	thin-ResNet34	Cosine similarity	Adam	GhostVLAD	Triplet loss (variation)	VoxCeleb1
Rozenberg, Aronowitz, and Hoory (2020)	?	X-vector DNN	Cosine similarity	?	?	?	VoxCeleb2
Nagrani, Chung, Xie, et al. (2020)	Spectrogram	thin-ResNet34	Cosine similarity	Adam	GhostVLAD	Softmax + Contrastive loss	VoxCeleb1/2

Table 2: Prior-art on SNNs in speaker recognition

	Audio repr.	Network	Distance measure	Optimizer	Aggregation	Loss	Dataset
Li, Jiang, Li, et al. (2020)	MFCC	2D-CNN	Euclidean distance	SGD	?	binary cross entropy	LibriSpeech
Kye et al. (2020)	MFB	ResNet34	Cosine similarity	SGD	TAP	Normalized softmax	VoxCeleb1/2
Anand et al. (2019)	Spectrogram	CapsNet	?	?	–	Prototypical loss	VoxCeleb1/VCTK

Table 3: Prior-art non-SNN networks in speaker recognition

	Audio repr.	Network	Optimizer	Aggregation	Loss	Dataset
Xie et al. (2019)	STFT	Thin-ResNet34	Adam	GhostVLAD	softmax?	VoxCeleb2
Li, Jiang, Wu, et al. (2020)	MFCC	GMM-based MDN	?	?	?	LibriSpeech
Snyder et al. (2019)	MFCC	X-vector DNN	?	?	?	VoxCeleb1/2
Ravanelli and Bengio (2018)	Raw audio	SincNet	RMSprop	–	softmax?	LibriSpeech

Table 4: Prior-art non-FSL in speaker recognition

4 Implementation and experiments

The experiments performed in this thesis, consisted of training multiple speaker recognition models and evaluating their performance. Initially, the experiments were set to compare:

- overall architectures, such as Siamese networks (see Figure 4), Model-Agnostic Meta-Learning (MAML) (Finn, Abbeel, and Levine 2017) or prototypical networks.
- trunk architectures (thin-ResNet (see Section 4.1), fast-ResNet or other).
- amount of training data (half of LibriSpeech-100, LibriSpeech-100, LibriSpeech-360, LibriSpeech-500, VoxCeleb1).
- augmented training data with different methods (Gaussian noise, MUSAN corpus (Snyder, Chen, and Povey 2015), room impulse response (RIR) noise, SpecAugment (Park et al. 2019)).
- various audio representations (see Figure 6) with different parameters.

However, this narrowed down to comparing SNN and prototypical networks using thin-ResNet trained on LibriSpeech-100 with and without augmentation with mel spectrogram input. Additionally, a subset of The Parliament of Finland (2017) was constructed ad hoc as a test set to measure performance of the models on speech in the Finnish language. The test set is further detailed in Section 4.3.1. The code written for implementing the experiments in this thesis has been made available as a public Git repository[1], including the scripts used for constructing the Finnish dataset.

It should be noted, that the experiments were not entirely rigorous, as there were some changes to the training setup changed over time. Not only were most models trained only once, the development of the implementation was continued during experimentation. Changes include change in impulse response data used for augmentation, as it was found to be incorrect. Also the 1cycle maximum learning rate parameter was increased as an attempt to shake out not-a-number (NaN) errors during training. There were also likely other changes that went unnoticed.

For implementing the experiments, PyTorch (Paszke et al. 2019) was chosen for it's apparent

[1] https://github.com/vjoki/fsl-experi

popularity among other researches in the field of speaker recognition. To further simplify the code and speed up development, PyTorch Lightning [2] framework was selected. Other alternatives were also considered, but PyTorch Lightning was chosen due to various advanced features being available out of the box.

To increase robustness against noise, random data augmentation is applied to the data during training. For this purpose, the Audiomentations[3] Python library was utilized to implement the following augmentations

- gaussian noise with random *signal-to-noise ratio* (SNR) from range of 0.2–0.5, with probability of 0.5,
- convolve the audio with random impulse responses from *RIRs noise set* (Ko et al. 2017), with probability of 0.3,
- adding random background noise from the RIRs noise set, with probability of 0.3,
- adding random short noises from the RIRs noise set using maximum SNR of 80 decibels, with probability of 0.3.

The implementations for *ResNet*, *AngularProto* and *SoftmaxProto* were borrowed, with minor modifications, from the unofficial Git repository[4] for the articles by Chung et al. (2020) and Heo et al. (2020). Additionally, a third party[5] implementation for *CapsNet* was borrowed, to implement a network based on the work of Hajavi and Etemad (2020), but in the end was not included in the experiments.

4.1 ResNetSE34

All of the models trained for this thesis share the same implementation of the 34-layer residual network with squeeze and excitation blocks (ResNetSE34) as the encoder layer, and the preprocessing layer for transforming the waveform into a spectrogram.

The preprocessing layer also used by Chung et al. (2020), with small additions, was used for

[2]https://github.com/PyTorchLightning/pytorch-lightning

[3]https://github.com/iver56/audiomentations

[4]https://github.com/clovaai/voxceleb_trainer/

[5]https://github.com/adambielski/CapsNet-pytorch

the implementation. The layer consists of

- applying pre-emphasis to the input signal (eq. 3.1), with the gradient calculation of the learning pre-emphasis layer disabled,
- transforming the signal into a mel spectrogram with 512 fast Fourier transform (FFT) frequency bins, using Hamming windowing with window length of 400 samples and hop length of 160 samples,
- transforming the spectrogram into logarithmic scale,
- applying 1-D instance normalization to the spectrogram.

Additionally, plain spectrogram, MFCC transforms and SpecAugment (Park et al. 2019) were implemented, but ultimately not explored in the experiments due to time constraints.

For the experiments, ResNetSE34V2 (or thin ResNet-34) (Heo et al. 2020) was used, but the implementation of ResNetSE34L (or fast ResNet-34) (Chung et al. 2020) was also provided. The ResNet implementations use the SE-ResNet module proposed by Hu et al. (2020).

ResNetSE34V2 consists of 8.0 million parameters. It differs from the original ResNet-34, in addition to the SE blocks, in that the channel counts in the residual blocks are halved and the first convolutional layer has stride 1. (Heo et al. 2020). The convolutional layers are initialized using the method described by He et al. (2015), for which uniform distribution is used for 1D layers and normal distribution for the 2D layers. Batch normalization layers have their weights initialized to 1, and bias to 0.

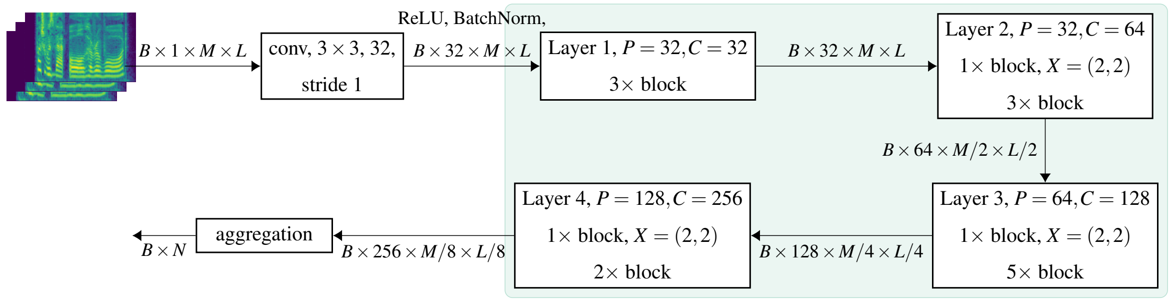
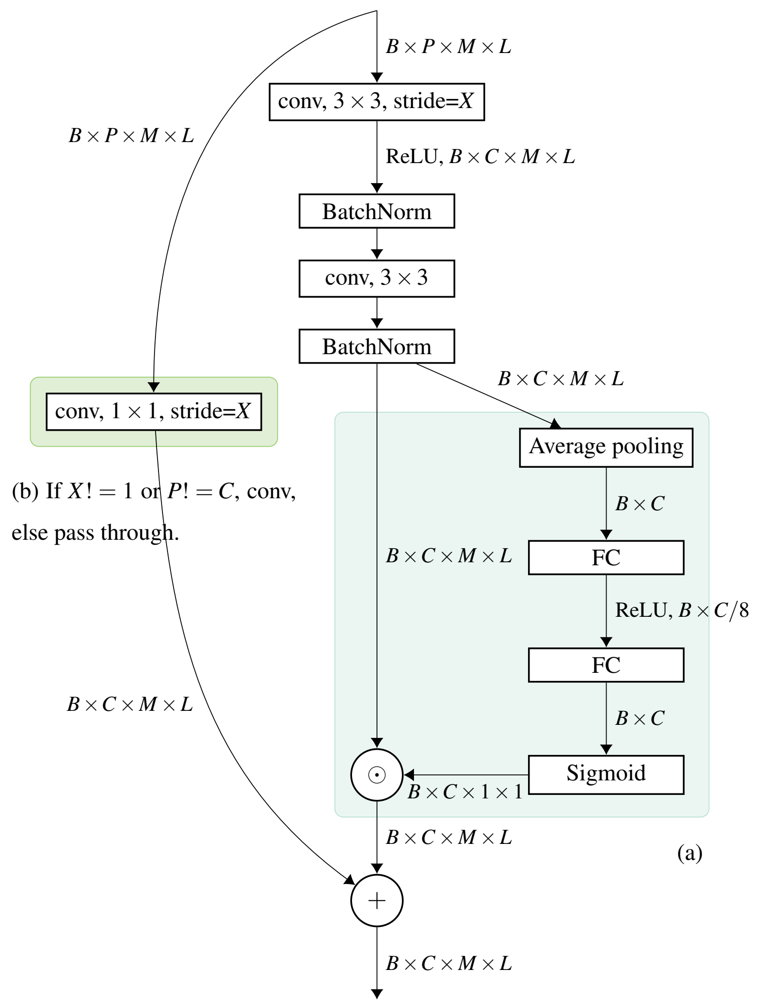

$B \times 1 \times M \times L$
conv, 3×3, 32, stride 1
ReLU, BatchNorm,
$B \times 32 \times M \times L$
Layer 1, $P = 32, C = 32$ $3\times$ block
$B \times 32 \times M \times L$
Layer 2, $P = 32, C = 64$ $1\times$ block, $X = (2,2)$ $3\times$ block
$B \times 64 \times M/2 \times L/2$
Layer 3, $P = 64, C = 128$ $1\times$ block, $X = (2,2)$ $5\times$ block
$B \times 128 \times M/4 \times L/4$
Layer 4, $P = 128, C = 256$ $1\times$ block, $X = (2,2)$ $2\times$ block
$B \times 256 \times M/8 \times L/8$
aggregation
$B \times N$

Figure 8: ResNetSE34V2 overview, parameter output features N. M is the mels count, L is the length of the input sequence and B is the batch size. The value of Y is dependent on the aggregation used. The structure of a block is detailed in Figure 9, while aggregation methods are shown in Figure 10.



$B \times P \times M \times L$
conv, 3×3, stride=X
ReLU, $B \times C \times M \times L$
BatchNorm
conv, 3×3
BatchNorm
$B \times C \times M \times L$
$B \times P \times M \times L$
conv, 1×1, stride=X
(b) If $X! = 1$ or $P! = C$, conv, else pass through.
Average pooling
$B \times C$
FC
ReLU, $B \times C/8$
FC
$B \times C$
Sigmoid
$B \times C \times 1 \times 1$
$B \times C \times M \times L$
$B \times C \times M \times L$
$B \times C \times M \times L$
(a)
$B \times C \times M \times L$

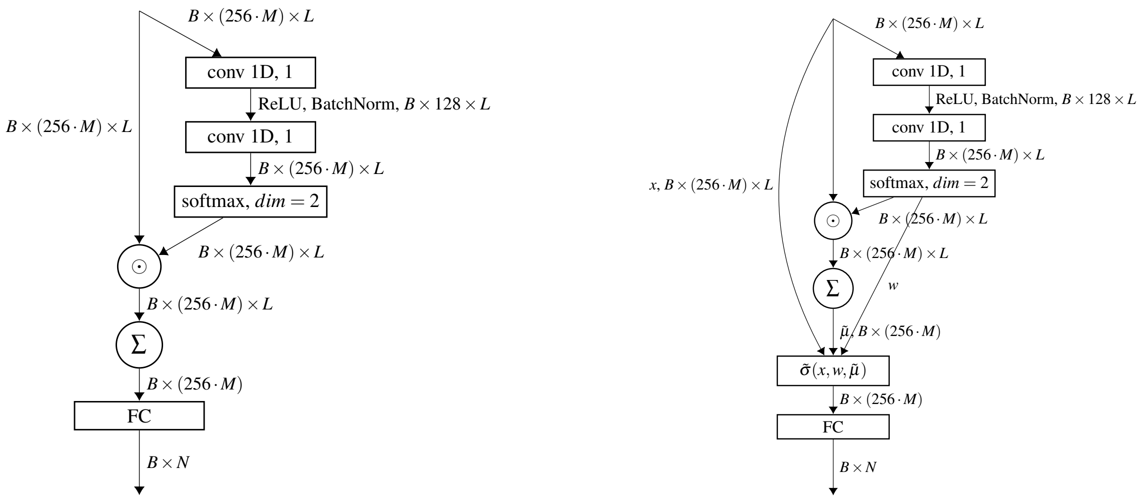
Figure 9: ResNetSE34 block, the input parameters are input channels P, output channels C and stride X. If no stride X specified $X = 1$. (a) is the squeeze and excitation block with reduction rate $r = 8$. (b) Convolution is applied if X is not 1 or the amount of input channels P is not the same as the amount of output channels C.

Self-Attentive Pooling (SAP) (Cai, Chen, and Li 2018) and Attentive Statistics Pooling (ASP) (Okabe, Koshinaka, and Shinoda 2018) are used to aggregate (or pool) the temporal frame-level features into a speaker embedding. These could be considered as more advanced extensions to the ideas of pooling described in Chapter 2.2. ASP differs from SAP, by adding an extra step to the process after the attention model as follows

$$\tilde{\sigma}(x, w, \tilde{\mu}) = \sqrt{\left(\sum x^2 \odot w\right) - \tilde{\mu}^2} \tag{4.1}$$

where $\sum$ signifies column-wise summation along the 3rd column. Figure 10 contains more detailed overview of these methods, which appear curiously similar to the ResNet block in Figure 9.

Additionally, NetVLAD (Arandjelovic et al. 2018), inspired by *Vector of Locally Aggregated Descriptors* (VLAD) image representation and GhostVLAD (Zhong, Arandjelovi, and Zisserman 2019), which builds upon NetVLAD, pooling layers were implemented. With the help of multiple third party implementations found online[6,7,8], but these were not thoroughly experimented with. Other methods were also noted but not implemented, such as Cross Attentive Pooling (CAP) (Kye, Kwon, and Chung 2021), and the method proposed by Kye, Chung, and Kim (2021).



(a) Self-Attentive Pooling (SAP). (b) Attentive Statistics Pooling (ASP).

Figure 10: Description of SAP and ASP aggregation layers from Figure 8.

4.2 Training

Training of the models was performed using the free notebook instances on Google Colaboratory (or Colab), with either a Nvidia K80 GPU with 12GB of VRAM or a Nvidia T4 GPU with 16GB of VRAM. This posed some significant limits on the experiments, due to the

[6]https://github.com/lyakaap/NetVLAD-pytorch/
[7]https://github.com/sitzikbs/netVLAD/
[8]https://github.com/Nanne/pytorch-NetVlad/

variable runtime limits and difficulty of reliably obtaining a graphics processing unit (GPU) instance. The claimed maximum lifetime of a free Colab runtime is 12 hours, but in practice the lifetimes were found to be quite variable.

For training the models, AdamW optimizer (Loshchilov and Hutter 2019) was used with learning rate scheduling according to the 1cycle learning rate policy (Smith and Topin 2018) based on the validation loss.

The `train-clean-100` subset, containing approximately 100 hours of audio, of LibriSpeech dataset was used for training the models. The `train-clean-100` set consists of 25 minutes of audio from each of the 251 speakers, of which 125 are female and 126 are male (Panayotov et al. 2015). The 100 hour subset was chosen over the larger subsets, as it was small enough to be used on a free Colab instance, and because the one of the initial focuses of this thesis was to investigate learning from low amount of data. The `dev-clean` set included in the LibriSpeech corpus was used for cross-validation.

As the PyTorch Lightning library was chosen as the framework for the implementation, only the training, validation and testing steps of the training process had to be implemented. This included feeding of the data to the model, handling of the loss function and the reshaping of the data being passed. An overview of the training algorithm can be seen in Algorithm 1.

To speed up the training of the models and to allow for larger batch sizes, the PyTorch implementation of automatic mixed precision (AMP) training based on the work of Micikevicius et al. (2018) was utilized. AMP makes use of half-precision floats (FP16) when appropriate, instead of simply using single-precision format (FP32) everywhere. This is done by casting the operations to mixed precision for forward propagation, and scaling the loss gradients. The optimizer is skipped, if gradients contain `inf` or `NaN` values, otherwise gradients are unscaled before they are passed to the optimizer.

Algorithm 1 Basic overview of the training algorithm.

```
for $epoch \leftarrow 0, epoch <$ max epoch do
    for $batch \leftarrow$ training data do
        $data, labels \leftarrow batch$
        $output \leftarrow$ model.forward($data$)                ▷ Forward propagation
        $loss \leftarrow L(output, labels)$
        $\nabla loss \leftarrow loss$.backward(model)              ▷ Backpropagation
        optimizer($\nabla loss$, model)                ▷ Update model parameters
    end for
    validation()                                       ▷ Cross-validation
end for
```

The training algorithm runs until a specific count of epochs has been reached, for most of the experiments this count was set to 50. During each epoch, the entire training dataset was iterated over in randomized batches. The batch size chosen for the experiments generally amounted to 64 samples per batch. As the size of the batched data varied based on the loss function and the sample size used, the actual batch size used was different. The choice of maximum epoch count was made based on the limits on available computation time, likewise batch size was chosen based on the amount of VRAM available on the GPU.

Each batch starts with forward propagation, where the training data in the batch is passed through the model. The resulting output is then further passed to the loss function, along with the labels containing the expected values. Backpropagation is performed using the loss scalar, computing the gradients with respect to the model parameters. The gradients are then utilized for updating the model parameters, with the chosen optimization algorithm. After each batch cross-validation is performed to monitor how well the model generalizes.

4.2.1 Loss functions

The PyTorch implementation[9] of *binary cross entropy* (BCE) with a sigmoid (eq. 4.3) layer is used in all the validation and test phases, but it is also utilized as the training loss of the simple SNN model. The rationale for including sigmoid (eq. 2.5) in the loss function is explained to be due to increased numerical stability.

$$
\begin{aligned}
\mathrm{BCE}(x,y) &= H\big(y_m \log x_m + (1-y_m)\log(1-x_m)\big) \\
&= -\frac{1}{M}\sum_{m=1}^{M}\big(y_m \log x_m + (1-y_m)\log(1-x_m)\big) \qquad (4.2)\\
L_{\mathrm{BCE}} &= \mathrm{BCE}(\sigma(x),y) \\
&= \mathrm{BCE}\left(\frac{1}{1+e^{-x}}, y\right), \qquad (4.3)
\end{aligned}
$$

where x represents the output of the model for a given training example, y the associated label value, M is the amount of training examples.

The implementation of *angular prototypical loss* (AngularProto) by Chung et al. (2020) was adapted in the implementation used in this thesis. Cosine similarity (eq. 2.10) is used as the similarity metric. The angular prototypical loss is almost identical to prototypical loss proposed by Snell, Swersky, and Zemel (2017), the key difference being the use of cosine similarity instead of Euclidean distance.

$$
c_i = \frac{1}{M-1}\sum_{m=2}^{M} x_{i,m} \qquad (4.4)
$$

$$
\begin{aligned}
S_{j,k} &= w \cdot \cos(x_{j,1}, c_k) + b \\
&= w\frac{x_{j,1}\cdot c_k}{\|x_{j,1}\|\,\|c_k\|} + b \qquad (4.5)\\
L_{\mathrm{AP}} &= H(\mathrm{softmax}(S_j)) \\
&= -\frac{1}{N}\sum_{j=1}^{N}\log \mathrm{softmax}(S_j)_j \\
&= -\frac{1}{N}\sum_{j=1}^{N}\log\frac{e^{S_{j,j}}}{\sum_{k=1}^{N} e^{S_{j,k}}}, \qquad (4.6)
\end{aligned}
$$

[9] https://pytorch.org/docs/1.7.1/generated/torch.nn.BCEWithLogitsLoss.html

where N is the number of speakers (or ways) in the batch, M is the number of utterances (or shots) for each speaker, x is an N by M matrix of model outputs. 32-way 2-shot (thus $N = 32$, $M = 2$) batches were used for training all the models using AngularProto in the experiments. (Chung et al. 2020; Heo et al. 2020).

Heo et al. (2020) combine AngularProto with *softmax loss*, which is used in the experiments under the name SoftmaxProto. The combination is achieved by obtaining both softmax loss (eq. 4.7) and angular prototypical loss (eq. 4.6), and adding the resulting losses together. Additionally, before passing the model output to the cross-entropy loss function, it is passed through a fully connected layer as shown in Figure 11. Softmax loss is also often referred to as *cross-entropy loss*, which is somewhat misleading as it is actually the combination of softmax function (eq. 2.6) and cross-entropy (eq. 2.3).

$$\begin{aligned} L_{\text{softmax}} &= H(\text{softmax}(x)) \\ &= -\frac{1}{N}\sum_{i=1}^{N} \log \text{softmax}(x)_i \\ &= -\frac{1}{N}\sum_{i=1}^{N} \log \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}}, \end{aligned} \tag{4.7}$$

where N is the number of speakers in the training set.

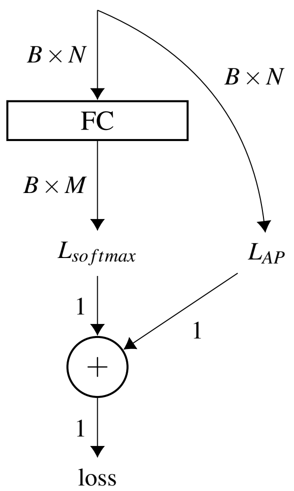


Figure 11: The combination of cross-entropy loss and AngularProto. B is the batch size, N is the number of features per utterance, and M is the number of speakers in the training set.

4.3 Evaluation

The evaluation of the trained models consisted of running speaker verification tasks on the test data, and measuring the performance of the models using the metrics described in Section 3.2. The test data consists of lists of labeled pairs of audio samples from a single speaker, where the label signified whether the speakers in the two samples were the same. The speakers in the test datasets should not appear in any of the training or validation datasets.

For evaluating the model performance, the same method us used for prediction as in validation during training. Which means doing forward propagation on batches from the test (or validation) dataset. However, depending on the model, evaluation differs slightly from training. For example, when measuring target and test speaker distance using a model trained BCE with sigmoid (eq.4.3), one needs to remember to apply sigmoid to the model output. Whereas measuring speaker distance with a model trained using AngularProto and SoftmaxProto requires passing each of the two inputs through the model. Then normalizing the output vectors x and y by taking the Euclidean norm along dimension 1. Finally, obtaining the result by taking the Euclidean distance (see eq. 2.11) of the normalized outputs.

The test data lists for LibriSpeech and Eduskunta sets were generated by randomly selecting test pairs from the dataset. The lists used were also made available in the same Git repository[10] as the implementation. For the VoxCeleb1 dataset, the tests were done using the test list published by the authors. The same generated lists were used for each test. During testing, 3 positive and 3 negative pairs were removed from the Eduskunta set test list, as the samples used were preventing test runs using full sample length. More details on the test sets used can be seen in Table 5.

Test set	Total utterances	# of speakers	# of test pairs	Positive/negative split
LibriSpeech test-other (Panayotov et al. 2015)	3029	33	2883	1673/1210
Librispeech test-clean (Panayotov et al. 2015)	2707	40	2611	1523/1088
VoxCeleb1 test (Nagrani, Chung, and Zisserman 2017)	4874	40	37720	18860/18860
Eduskunta (Section 4.3.1)	13680	171	13640	6837/6797

Table 5: Description of the datasets and test pair lists used.

[10] https://github.com/vjoki/fsl-experi/

4.3.1 Plenary Sessions of the Parliament of Finland dataset

To compare how speaker recognition models trained on English speech perform on other languages, a dataset consisting of Finnish speech suitable for speaker verification was constructed ad hoc. For this purpose, audio data in the waveform audio file format (WAVE, or WAV) and annotation files in the ELAN annotation format (EAF) from the dataset by The Parliament of Finland (2017) were utilized. Specifically, the test set was constructed using the audio of the plenary sessions of 2016.

As the original data is chunked in large WAV files, with per word aligned transcriptions. It was necessary to extract suitably long utterances, by grouping the annotations. The grouping was performed by terminating on the end of the sentence, if utterance duration was greater than 3 seconds or when utterance reached duration greater than 20, but less than 25 seconds. This process was achieved by first programmatically editing the annotation files, and then splitting the WAV file based on the timestamps associated with the annotations.

To further reduce the size of the test set, 80 samples were selected from each speaker. Speakers with less samples were excluded from the test set. Additionally, it was required to manually tweak the results as some speakers ended up with multiple speaker ids, but this could be easily fixed in further iterations. The resulting dataset consists of speech from 171 speakers and 13680 utterances.

Further caveats include the dataset not being verified in any way for overlapping speech, misaligned timestamps, incorrect grouping of speakers. Additionally, there may be some Finland Swedish speech in some utterances. The full process was documented and scripts published in the same public Git repository[11] as the implementation of the experiments.

[11] https://github.com/vjoki/fsl-experi/

5 Results

In this chapter, the results of the evaluation described in Section 4.3 are presented. Section 5.1 covers the reference models used and their results, while Section 5.2 focuses on the models trained for this thesis. Complete lists of all the results can be found in the Appendix A.

The results show that, for speaker recognition, an SNN trained using BCE loss does not generalize as well as a ResNet trained using 32-way 2-shot metric learning using prototypical loss. Additionally, the SNN did not benefit as much from data augmentation. Also it was discovered that training using 2 second sample length resulted in better performance than training with 4 second samples. Surprisingly the models performed as well on the Eduskunta test set consisting of Finnish speech, as they did on the English speech of the VoxCeleb1 test set.

Though same labeled lists of sample pairs were used for the tests, it is possible that adverse effects from randomness to each test run was not completely eliminated, as the 4 or 2 second section of an audio sample was chosen randomly each time. However, the same random number generator seeds were used each time to mitigate this, and no noticeable variance between test runs was observed in the EER (eq. 3.3) and minDCF (eq. 3.4) metrics.

Accuracy measures how close to the expected values the model predictions are, and it is one of the possible evaluation metrics. While the accuracy of the model was not used for evaluation, it was observed that it was necessary to use the EER threshold (eq. 3.2) for computing the accuracy. One possible cause for this could be the class imbalance in the training (and validation) data, the results of the trained models are slightly shifted from the expected output range of $[0,1]$. Thus, to get a usable measure of the model accuracy, it was necessary to find a usable threshold value for each model. Using the EER threshold resulted in approximately $accuracy + EER = 1$.

5.1 Reference benchmarks

Unofficial pre-trained models[1], based on fast ResNet-34 (Chung et al. 2020), and *H / ASP* (Heo et al. 2020), based on the thin ResNet-34, were tested for reference. The models have been trained on the VoxCeleb2 training set consisting of 1092009 utterances from 5994 distinct speakers, with sample length of 2 seconds. However, the mel scale used to construct the mel spectrogram for the models differs, for fast ResNet-34 40 mels were used while H / ASP model was trained with 64 mels. The fast ResNet-34 model by Chung et al. (2020) uses SAP encoder (see Figure 10a), and was trained using the angular prototypical loss function. While the H / ASP model by Heo et al. (2020) uses ASP encoder (see Figure 10b), and was trained using the softmax prototypical loss function with online data augmentation.

Table 6 shows the results of benchmarks performed in this thesis using the pre-trained models, giving a reference point for comparison to the models specifically implemented and trained for this thesis. Here one can see, that H / ASP shows excellent results on the LibriSpeech and VoxCeleb1 datasets, with the fast ResNet-34 model slightly behind. Further, the models are tested on the Finnish Eduskunta dataset, showing some increase in the error rate, but still quite good results.

	Fast ResNet-34		H / ASP	
Test set	EER	minDCF	EER	minDCF
LibriSpeech test-other	2.40%	0.1288	0.83%	0.0317
LibriSpeech test-clean	2.85%	0.0814	0.85%	0.0404
VoxCeleb1 test	2.18%	0.1690	1.18%	0.0865
Eduskunta test	6.36%	0.2356	5.86%	0.1214

Table 6: Reference benchmark results.

[1] https://github.com/clovaai/voxceleb_trainer/

5.2 Benchmarks

The results of the benchmarks performed on the models trained for this thesis are described in this section. Tables 7 and 8 list the parameters used for training each model, which are described further in Section 4.2. The models are primarily grouped by loss function used in training (see Section 4.2.1).

Test model				Training		
Encoder	Training loss	Speakers	Epoch	Aug.	Epochs	Steps
GhostVLAD	BCE	251	8			
ASP	BCE	251	43		49	21810
SAP	BCE	251	34		39	17360
SAP	BCE	251	47		49	21810
SAP	BCE	100	18		49	15222
SAP	BCE	50	31		49	13040
SAP	BCE	251	22	✓	29	12910
SAP	BCE	251	24	✓	29	12910
SAP	AngularProto	251	15		49	20290
SAP	AngularProto	251	21		49	20140
SAP	AngularProto	251	29	✓	29	12010
ASP	AngularProto	251	30		49	20430
ASP	AngularProto	100	126		144	21600
ASP	AngularProto	50	120		499	28940
ASP	AngularProto	251	47	✓	49	19850
GhostVLAD	AngularProto	251	36		49	20340
SAP	SoftmaxProto	251	49		49	20090
ASP	SoftmaxProto	251	40		49	20290
ASP	SoftmaxProto	100	124		124	19840
ASP	SoftmaxProto	50	105		349	20240
ASP	SoftmaxProto	251	49	✓	49	20430

Table 7: Training parameters for models using mel spectrogram of 64 mels with sample length of 4 secs.

Test model					Training			
Encoder	Training loss	Mels	Speakers	Epoch	Sample	Aug.	Epochs	Steps
SAP	BCE	64	251	47	2s		49	21810
SAP	BCE	64	251	33	2s	✓[1]	49	10880
SAP	BCE	64	251	27	2s	✓[1]	49	10930
ASP[2]	AngularProto	64	251	43	2s	✓	49	9997
ASP	AngularProto	64	251	49	3s	✓	49	20140
ASP[3]	AngularProto	64	251	41	2s	✓	49	19990
ASP	AngularProto	64	251	48	2s	✓	49	20140
ASP	SoftmaxProto	64	251	44	2s	✓	49	20040

Table 8: Training parameters for models using different sample lengths. 1. Model using GPU optimized augmentations. 2. 32-way 4-shot. 3. Using the fast-ResNet34 network.

5.2.1 Binary cross-entropy

This section contains the results for the SNN model trained using binary cross-entropy loss. This model was not able to effectively generalize, showing significantly worse performance in the VoxCeleb1 and Eduskunta tests.

The results of the variant using SAP in the ResNet trained using 251, 100 and 50 speakers are shown in Table 11. Contrary to expectations, the model trained on 251 speakers performed significantly worse with the full sample than the other models, with no clear indication as to why this happened. An additional model was trained using the same parameters, but the behaviour did not change significantly. It may be possible to achieve better results, with larger number of speakers in the training set. The SAP network was also trained using 251 speakers with data augmentation (see Table 12). While the expectation was that augmentation would improve the performance across the board, Figure 12 shows that this was not the case. Performance increased noticeably only for the test-other and VoxCeleb1 sets, but for test-clean set the performance decreased. The cause for this is not clear, but may be due to test-clean being the most similar to the training set.

The results of the variant using ASP in the ResNet are shown in Table 9. The ASP model shows similar performance to the nonaugmented SAP model, without issues with the full length samples. This network was also trained with 2 second sample length and GPU accelerated data augmentation, resulting in better performance in all tests (shown in Table 10). Also, unlike the nonaugmented model, the best results were achieved in the test-clean set instead of the weaker test-other set.

	Results (4s)		Results (2s)		Results (full)	
Test set	EER	minDCF	EER	minDCF	EER	minDCF
LibriSpeech test-other	17.10%	1.0	17.10%	1.0	*14.39%*	*0.9178*
LibriSpeech test-clean	21.14%	1.0	21.14%	1.0	16.35%	1.0
VoxCeleb1 test	32.87%	1.0	32.87%	1.0	30.17%	1.0
Eduskunta test	32.13%	1.0	32.69%	1.0	28.04%	1.0

Table 9: Tests results for SNN ASP BCE 64 mels, trained on 251 speakers with sample length 4 seconds.

	Results (4s)		Results (2s)		Results (full)	
Test set	EER	minDCF	EER	minDCF	EER	minDCF
LibriSpeech test-other	12.97%	0.9444	12.14%	0.9137	10.09%	0.7964
LibriSpeech test-clean	13.87%	0.6811	14.36%	0.8085	*9.65%*	*0.4637*
VoxCeleb1 test	28.64%	0.9999	30.74%	0.9998	27.88%	0.9996
Eduskunta test	27.57%	1.0	29.72%	1.0	24.69%	0.9999

Table 10: Tests results for SNN ASP BCE 64 mels using GPU accelerated data augmentation with 251 speakers with sample length of 2 seconds.

		Results (4s)		Results (2s)		Results (full)	
Test set	Speakers	EER	minDCF	EER	minDCF	EER	minDCF
LibriSpeech test-other	251	17.10%	0.9851	17.10%	0.9851	44.18%	1.0
	100	23.65%	1.0	23.55%	1.0	21.82%	1.0
	50	30.25%	1.0	29.65%	1.0	26.78%	1.0
LibriSpeech test-clean	251	*16.28%*	*0.9509*	*16.28%*	*0.9509*	44.03%	1.0
	100	33.28%	1.0	31.17%	1.0	28.84%	1.0
	50	36.95%	1.0	36.74%	1.0	38.42%	1.0
VoxCeleb1 test	251	32.44%	0.9999	32.44%	0.9999	46.44%	1.0
	100	36.00%	1.0	36.69%	1.0	33.10%	1.0
	50	39.60%	1.0	39.85%	1.0	37.84%	1.0
Eduskunta test	251	28.39%	0.9994	28.63%	0.9997	45.15%	1.0
	100	37.68%	0.9999	37.87%	1.0	33.94%	1.0
	50	41.75%	0.9999	41.84%	1.0	40.55%	1.0

Table 11: Comparison of test results for SNN SAP BCE 64 mels models trained using reduced number of speakers with sample length 4 seconds.

	Results (4s)		Results (2s)		Results (full)	
Test set	EER	minDCF	EER	minDCF	EER	minDCF
LibriSpeech test-other	13.39%	0.9916	14.29%	0.9824	*9.43%*	*0.8530*
LibriSpeech test-clean	22.33%	0.9810	22.41%	0.9934	17.54%	0.9866
VoxCeleb1 test	29.13%	0.9998	29.38%	0.9995	25.32%	0.9994
Eduskunta test	29.77%	0.9990	30.63%	0.9997	25.17%	0.9991

Table 12: Tests results for SNN SAP BCE 64 mels using data augmentation with 251 speakers, sample length 4 seconds.

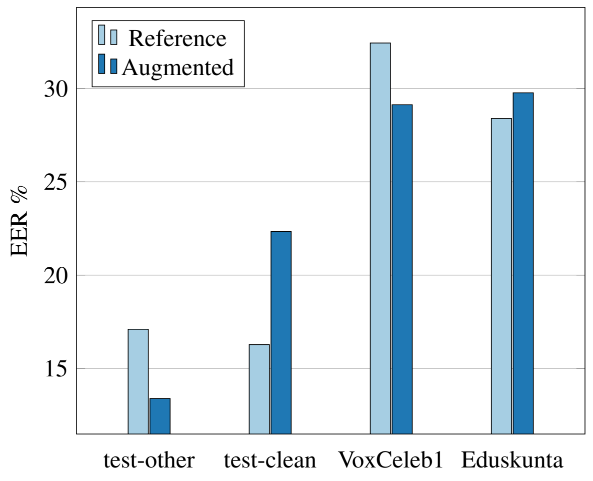


Figure 12: Comparison of the effect of augmentation on SNN SAP model with 4 seconds samples.

5.2.2 AngularProto

This section details the results for the angular prototypical loss, also used in the Fast ResNet-34 reference model. The results of the variant using ASP in the ResNet trained using 251, 100 and 50 speakers are shown in Table 13. Interestingly, the model trained with 100 speakers shows performance on par to the model trained with 251 speakers for the LibriSpeech sets, in fact outperforming it on the test-clean set. Performance on the harder VoxCeleb1 and Eduskunta tests is poor, and comparable to the augmented BCE ASP model. The SAP variant trained with 251 speakers performed worse than the ASP model in the hard tests, and slightly better on the easier LibriSpeech tests (see Table 14).

Table 15 lists the results for SAP and ASP models trained with 251 speakers with data augmentation and sample length of 4 seconds. Figure 13 shows augmentation significantly increases performance on the hard VoxCeleb1 and Eduskunta sets with all the results going under EER of 23%, indicating that generalization of the AngularProto model benefits noticeably from data augmentation. This is further supported by how the performance does not depend on augmentation for the test-clean set, that has been split from the training set.

It was found that training using 2 second sample length gives better results (see Figure 14). Sample lengths of 1 and 3 seconds were also attempted, but training with 1 second samples did not result in a usable model.

		Results (4s)		Results (2s)		Results (full)	
Test set	Speakers	EER	minDCF	EER	minDCF	EER	minDCF
	251	9.09%	0.4452	8.36%	0.4658	6.52%	*0.3870*
LibriSpeech test-other	100	8.84%	0.5753	8.67%	0.5832	6.94%	0.4803
	50	14.78%	0.7937	13.80%	0.7669	11.58%	0.6602
	251	9.27%	0.6999	8.92%	0.7652	6.78%	0.5898
LibriSpeech test-clean	100	8.38%	0.7395	8.81%	0.7280	*4.87%*	0.6419
	50	14.06%	0.8641	14.44%	0.8622	9.38%	0.6107
	251	30.78%	0.9967	30.95%	0.9970	28.36%	0.9876
VoxCeleb1 test	100	32.66%	0.9945	32.43%	0.9968	29.01%	0.9850
	50	32.93%	0.9984	33.09%	0.9990	29.05%	0.9871
	251	26.01%	1.0	26.01%	0.9991	21.29%	0.9999
Eduskunta test	100	28.01%	0.9993	27.82%	0.9987	22.36%	0.9999
	50	28.99%	0.9996	28.84%	0.9996	24.89%	0.9985

Table 13: Comparison of test results for ASP AngularProto 64 mels models trained using reduced number of speakers with sample length of 4 seconds.

	Results (4s)		Results (2s)		Results (full)	
Test set	EER	minDCF	EER	minDCF	EER	minDCF
LibriSpeech test-other	6.69%	0.5297	7.01%	0.5425	*5.27%*	*0.3977*
LibriSpeech test-clean	8.73%	0.7632	7.81%	0.7092	5.52%	0.5623
VoxCeleb1 test	32.37%	0.9944	31.98%	0.9983	30.83%	0.9983
Eduskunta test	28.13%	0.9997	27.83%	1.0	22.44%	0.9999

Table 14: Tests results for SAP AngularProto 64 mels with sample length of 4 seconds.

		Results (4s)		Results (2s)		Results (full)	
Test set	Encoder	EER	minDCF	EER	minDCF	EER	minDCF
LibriSpeech test-other	SAP	5.38%	0.3485	4.62%	0.2671	2.57%	*0.1601*
	ASP	5.03%	0.3551	5.27%	0.3284	*2.39%*	0.1846
LibriSpeech test-clean	SAP	9.92%	0.6915	10.22%	0.7253	6.51%	0.4975
	ASP	9.11%	0.7202	9.38%	0.6853	6.24%	0.5347
VoxCeleb1 test	SAP	22.75%	0.9598	22.90%	0.9496	16.98%	0.8310
	ASP	22.25%	0.9559	22.55%	0.9544	16.18%	0.7938
Eduskunta test	SAP	21.92%	1.0	21.91%	0.9990	17.02%	0.9999
	ASP	21.85%	0.9895	21.94%	0.9988	16.95%	0.9877

Table 15: Tests results for AngularProto 64 mels using data augmentation with sample length of 4 seconds.

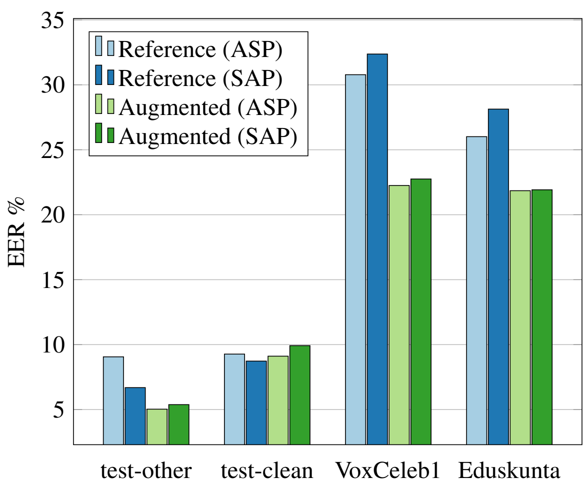


Figure 13: Comparison of the effect of augmentation on AngularProto model with 4 second sample length.

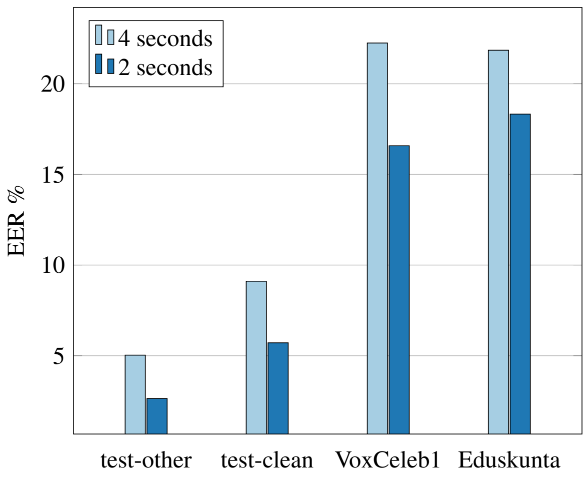


Figure 14: Comparison of the effect of sample length on AngularProto ASP model using data augmentation with the full sample.

5.2.3 SoftmaxProto

This section details the results for the softmax prototypical loss, a direct extension of AngularProto also used in the H / ASP reference model. During training the of the models, it was observed that SoftmaxProto is noticeably more sensitive to higher learning rates, requiring extra care to avoid NaNs.

Table 16 shows the results for the ASP and SAP variants trained with 251 speakers, using sample length of 4 seconds. The results on the hard VoxCeleb1 and Eduskunta sets are slightly better here than the equivalent for AngularProto.

The results of the variation using ASP in the ResNet trained with 251, 100 and 50 speakers are shown in Table 17. The results are more or less as could be expected, with the full sample length and more speakers giving better results linearly. Performance is slightly better than AngularProto.

Table 18 lists the results for the ASP variant trained with data augmentation. As with AngularProto, data augmentation increases performance significantly (see Figure 15). However, for test-clean set augmentation seemed to decrease performance by 31%. Like before, the results are slightly better than AngularProto.

		Results (4s)		Results (2s)		Results (full)	
Test set	Encoder	EER	minDCF	EER	minDCF	EER	minDCF
LibriSpeech test-other	SAP	8.50%	0.4917	8.60%	0.4933	7.18%	0.5094
	ASP	7.18%	0.5414	6.21%	0.5745	5.06%	0.4923
LibriSpeech test-clean	SAP	7.62%	0.6139	7.62%	0.6525	*4.87%*	0.5292
	ASP	6.32%	0.5350	6.32%	0.5466	5.32%	*0.3443*
VoxCeleb1 test	SAP	29.78%	0.9968	29.80%	0.9986	26.16%	0.9888
	ASP	29.64%	0.9948	29.55%	0.9941	26.86%	0.9696
Eduskunta test	SAP	25.13%	0.9999	25.28%	1.0	18.71%	1.0
	ASP	25.34%	0.9960	24.94%	0.9994	21.21%	0.9996

Table 16: Tests results for SoftmaxProto with sample length of 4 seconds.

		Results (4s)		Results (2s)		Results (full)	
Test set	Speakers	EER	minDCF	EER	minDCF	EER	minDCF
	251	7.18%	0.5414	6.21%	0.5745	*5.06%*	0.4923
LibriSpeech test-other	100	10.82%	0.6650	10.41%	0.6240	8.67%	0.5413
	50	14.39%	0.7929	13.80%	0.8060	10.75%	0.5559
	251	6.32%	0.5350	6.32%	0.5466	5.32%	*0.3443*
LibriSpeech test-clean	100	9.84%	0.5625	9.27%	0.5063	6.36%	0.3703
	50	13.60%	0.8493	13.33%	0.8361	7.92%	0.5594
	251	29.64%	0.9948	29.55%	0.9941	26.86%	0.9696
VoxCeleb1 test	100	31.43%	0.9958	31.36%	0.9963	27.83%	0.9827
	50	31.23%	0.9993	31.26%	0.9998	26.76%	0.9545
	251	25.34%	0.9960	24.94%	0.9994	21.21%	0.9996
Eduskunta test	100	26.78%	0.9997	26.41%	0.9988	21.97%	0.9982
	50	29.16%	0.9991	29.41%	0.9988	23.93%	0.9913

Table 17: Comparison of test results for ASP SoftmaxProto 64 mels models trained using reduced number of speakers with sample length of 4 seconds.

		Results (4s)		Results (2s)		Results (full)	
Test set	Encoder	EER	minDCF	EER	minDCF	EER	minDCF
LibriSpeech test-other	ASP	4.62%	0.2861	3.88%	0.3094	*1.84%*	*0.1591*
LibriSpeech test-clean	ASP	9.19%	0.6982	10.00%	0.7317	6.62%	0.6406
VoxCeleb1 test	ASP	22.20%	0.9217	21.89%	0.9201	15.96%	0.8001
Eduskunta test	ASP	21.43%	0.9990	21.42%	0.9999	16.51%	0.9996

Table 18: Tests results for SoftmaxProto using 64 mels and data augmentation with sample length of 4 seconds.

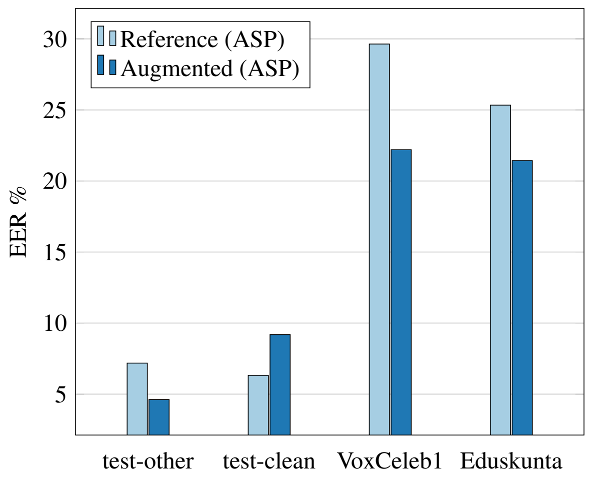


Figure 15: Comparison of the effect of augmentation on SoftmaxProto model.

5.2.4 Comparison on different number of speakers

Figures 16 and 17 show that the error rate did not always decrease linearly with respect to number of speakers included in the training set. The linear decrease of SNN model and ASP SoftmaxProto on the LibriSpeech seems to indicate, that further increasing the number of speaker would likely significantly improve the performance on the LibriSpeech test set. While ASP AngularProto seems to reach best performance at 100 speakers, the ASP SoftmaxProto graph for VoxCeleb1 set seems to contradict this conclusion. Thus, it seems like more data points would be necessary to make predictions on behaviour with larger amount of speakers.

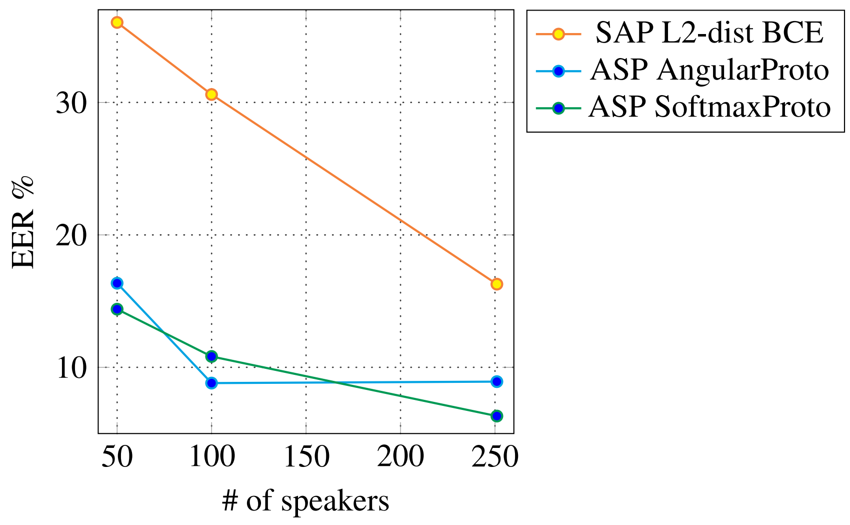


Figure 16: Comparison of EER on LibriSpeech test set for SNNs trained using different number of speakers from the training set.

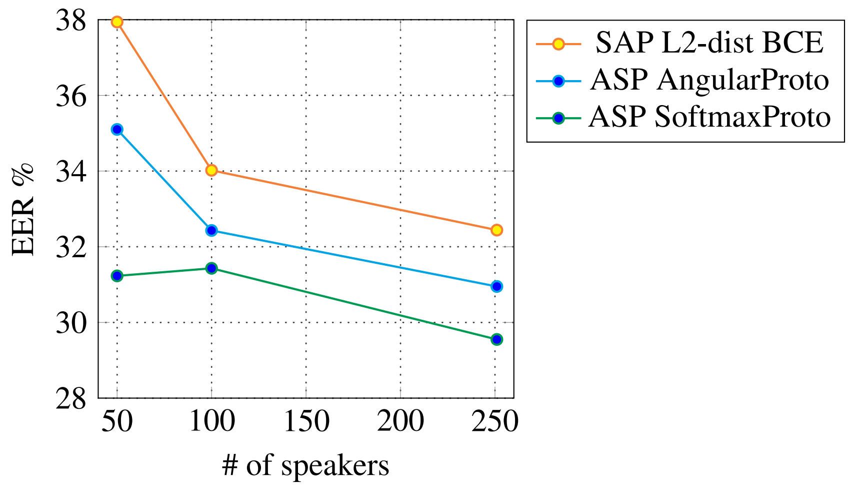


Figure 17: Comparison of EER on VoxCeleb1 test set for SNNs trained using different number of speakers from the training set.

5.2.5 Summary of the results

The results show, that the examined speaker recognition models trained with English speech perform nearly as well when tested with Finnish speech. With the Eduskunta test set performing on par with the VoxCeleb1 test set, when tested with the models trained for this thesis. The best results for each being EER of 16.02% for Eduskunta test set, and EER of

15.32% for VoxCeleb1 test set. In the reference benchmark, we could see Eduskunta test performing noticeably worse but still acceptably, achieving only EER of 5.86% as opposed to VoxCeleb1 test reaching EER of 1.18%.

Additionally, it was clearly visible that the models using AngularProto and SoftmaxProto loss performed significantly better, than the naive end-to-end SNN models. Further, it can be hypothesized that achieving better generalization requires either or both longer training time or more training data, as was shown by the reference benchmark. However, achieving decent performance with similar data could be achieved quite fast with little training data, as the best models reached EER of 4.67% on the LibriSpeech test-clean set. Performance, when training with less data from fewer speakers, did not behave linearly for AngularProto loss. Only very minor difference was observed when using training data with 100 or 251 speakers with performance dropping only at 50 speakers. Towards the end of the experiments, it was noticed that training on 2 second sample length achieved better results on nearly all benchmarks, than the 4 seconds used for most of the experiments. The cause for this is not clear, but one possible explanation could be that it is due to the parameters chosen for transforming the signal to a spectrogram. Further, the sample length used when testing, appeared to not be tied by the length used in training the model. As using the full sample length in the benchmarks showed uniformly better results.

6 Conclusion

In this thesis, machine learning was applied to implement, train and test of multiple neural networks for speaker recognition task. As ground work for the thesis, a light literature review was performed and the results were collected in Chapter 3.3. This helped to guide the decisions made for the implementation and the experiments. Additionally, it was found that the Plenary Sessions of the Parliament of Finland (The Parliament of Finland 2017) dataset could easily be transformed into a Finnish language speaker recognition dataset in Section 4.3.1. Which was then used for testing the models, that were trained for this thesis using the LibriSpeech (Panayotov et al. 2015) dataset consisting of English speech. For training the networks, an additional restriction was in place, in that only small amount of data and relatively limited computational resources could be used. In this light, the results were promising and included some interesting observations regarding generalizing on multiple languages and optimal sample length. Also while each experiment was not performed a statistically significant amount of times, due to the large scope, the networks that were trained multiple times with same parameters did indicate that reproducing each result within a reasonable margin of error ought to be possible.

The experiments documented in Chapter 4 were implemented using Python with the PyTorch ML library. The implemented networks were trained using free Google Colaboratory notebook instances, and the resulting models were benchmarked locally on the authors personal hardware. The training on Colab instances was found to be somewhat inconvenient due to the time limitations, that were found to vary from time to time.

The results in Chapter 5 show, that while the models trained in this thesis understandably did not generalize as well as the reference models, they did achieve decent performance on similar audio data. It is also shown, that the models trained with English speech do appear to generalize to Finnish speech as well. Other interesting observations made based on the results also include the sample length of 2 seconds clearly outperforming the other choices.

Future work could further investigate how models trained on Finnish perform on English data, or how these methods generalize over larger variety of languages. Also it could be worthwhile to investigate why 2 second samples appeared to perform best. There is also likely plenty to discover by simply trying additional neural network architectures, or for example investigating the effects of tuning the parameters of the various possible audio representations.

Bibliography

Anand, Prashant, Ajeet Kumar Singh, Siddharth Srivastava, and Brejesh Lall. 2019. “Few Shot Speaker Recognition Using Deep Neural Networks.” April 17. arXiv: 1904.08775 [cs, eess].

Arandjelovic, Relja, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. 2018. “NetVLAD: CNN Architecture for Weakly Supervised Place Recognition.” *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40 (6): 1437–1451. ISSN: 0162-8828, 2160-9292. doi:10.1109/TPAMI.2017.2711011.

Beigi, Homayoon. 2011. *Fundamentals of Speaker Recognition.* Boston, MA: Springer US. ISBN: 978-0-387-77592-0. doi:10.1007/978-0-387-77592-0.

Bromley, Jane, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. 1993. “Signature Verification Using a "Siamese" Time Delay Neural Network.” *Advances in neural information processing systems* 6:737–744. doi:10.1142/9789812797926_0003.

Cai, Weicheng, Jinkun Chen, and Ming Li. 2018. “Exploring the Encoding Layer and Loss Function in End-to-End Speaker and Language Recognition System.” In *Odyssey 2018 The Speaker and Language Recognition Workshop,* 74–81. ISCA, June 26. doi:10.21437/Odyssey.2018-11.

Cao, Houwei, David G. Cooper, Michael K. Keutmann, Ruben C. Gur, Ani Nenkova, and Ragini Verma. 2014. “CREMA-D: Crowd-Sourced Emotional Multimodal Actors Dataset.” *IEEE Transactions on Affective Computing* 5 (4): 377–390. ISSN: 1949-3045. doi:10.1109/TAFFC.2014.2336244.

Cartwright, Mark, and Bryan Pardo. 2015. “VocalSketch: Vocally Imitating Audio Concepts.” In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems - CHI ’15,* 43–46. Seoul, Republic of Korea: ACM Press. ISBN: 978-1-4503-3145-6. doi:10.1145/2702123.2702387.

Chen, Bo-Yu, Jordan B. L. Smith, and Yi-Hsuan Yang. 2020. “Neural Loop Combiner: Neural Network Models for Assessing the Compatibility of Loops.” August 5. arXiv: 2008.02011 [cs, eess].

Chung, Joon Son, Jaesung Huh, Seongkyu Mun, Minjae Lee, Hee Soo Heo, Soyeon Choe, Chiheon Ham, Sunghwan Jung, Bong-Jin Lee, and Icksang Han. 2020. “In Defence of Metric Learning for Speaker Recognition.” In *Proc. Interspeech 2020,* 2977–2981. doi:10.21437/Interspeech.2020-1064. arXiv: 2003.11982 [cs, eess].

Chung, Joon Son, Arsha Nagrani, and Andrew Zisserman. 2018. “VoxCeleb2: Deep Speaker Recognition.” In *Proc. Interspeech 2018,* 1086–1090. doi:10.21437/Interspeech.2018-1929. arXiv: 1806.05622 [cs, eess].

Ding, Shaojin, Tianlong Chen, Xinyu Gong, Weiwei Zha, and Zhangyang Wang. 2020. “AutoSpeech: Neural Architecture Search for Speaker Recognition.” In *Proc. Interspeech 2020,* 916–920. doi:10.21437/Interspeech.2020-1258. arXiv: 2005.03215 [cs, eess].

Feng, Kexin, and Theodora Chaspari. 2020. “A Siamese Neural Network with Modified Distance Loss For Transfer Learning in Speech Emotion Recognition.” June 4. arXiv: 2006.03001 [cs].

Finn, Chelsea, Pieter Abbeel, and Sergey Levine. 2017. “Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks”: 10.

Goodfellow, Ian, Joshua Bengio, and Aaron Courville. 2016. *Deep Learning.* MIT Press. ISBN: 9780262035613, accessed October 12, 2020. https://www.deeplearningbook.org/.

Hajavi, Amirhossein, and Ali Etemad. 2020. “Siamese Capsule Network for End-to-End Speaker Recognition In The Wild.” September 28. arXiv: 2009.13480 [cs, eess].

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. “Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification.” In *2015 IEEE International Conference on Computer Vision (ICCV),* 1026–1034. December. ISBN: 978-1-4673-8391-2. doi:10.1109/ICCV.2015.123.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. "Deep Residual Learning for Image Recognition." In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* 770–778. doi:10.1109/CVPR.2016.90. arXiv: 1512.03385 [cs].

Heo, Hee Soo, Bong-Jin Lee, Jaesung Huh, and Joon Son Chung. 2020. "Clova Baseline System for the VoxCeleb Speaker Recognition Challenge 2020." September 29. arXiv: 2009.14153 [cs, eess].

Hospedales, Timothy, Antreas Antoniou, Paul Micaelli, and Amos Storkey. 2020. "Meta-Learning in Neural Networks: A Survey." April 11. arXiv: 2004.05439 [cs, stat].

Hu, Jie, Li Shen, Samuel Albanie, Gang Sun, and Enhua Wu. 2020. "Squeeze-and-Excitation Networks." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42 (8): 2011–2023. doi:10.1109/TPAMI.2019.2913372. arXiv: 1709.01507 [cs].

Jung, Jee-weon, Seung-bin Kim, Hye-jin Shim, Ju-ho Kim, and Ha-Jin Yu. 2020. "Improved RawNet with Feature Map Scaling for Text-Independent Speaker Verification Using Raw Waveforms." In *Proc. Interspeech 2020,* 1496–1500. doi:10.21437/Interspeech.2020-1011. arXiv: 2004.00526 [cs, eess].

Keren, Gil, Maximilian Schmitt, Thomas Kehrenberg, and Björn Schuller. 2018. "Weakly Supervised One-Shot Detection with Attention Similarity Networks." June 27. arXiv: 1801.03329 [cs, stat].

Ko, Tom, Vijayaditya Peddinti, Daniel Povey, Michael L. Seltzer, and Sanjeev Khudanpur. 2017. "A Study on Data Augmentation of Reverberant Speech for Robust Speech Recognition." In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP),* 5220–5224. March. ISBN: 978-1-5090-4117-6. doi:10.1109/ICASSP.2017.7953152.

Köhn, Arne, Florian Stegen, and Timo Baumann. 2016. "Mining the Spoken Wikipedia for Speech Data and Beyond": 4644–4647. https://www.aclweb.org/anthology/L16-1735.

Kwon, Yoohwan, Hee-Soo Heo, Bong-Jin Lee, and Joon Son Chung. 2020. "The Ins and Outs of Speaker Recognition: Lessons from VoxSRC 2020." October 29. arXiv: 2010.15809 [cs, eess].

Kye, Seong Min, Joon Son Chung, and Hoirin Kim. 2021. "Supervised Attention for Speaker Recognition." In *2021 IEEE Spoken Language Technology Workshop (SLT),* 286–293. doi:10.1109/SLT48900.2021.9383579. arXiv: 2011.05189 [cs, eess].

Kye, Seong Min, Youngmoon Jung, Hae Beom Lee, Sung Ju Hwang, and Hoirin Kim. 2020. "Meta-Learning for Short Utterance Speaker Recognition with Imbalance Length Pairs." In *Proc. Interspeech 2020,* 2982–2986. doi:10.21437/Interspeech.2020-1283. arXiv: 2004.02863 [cs, eess, stat].

Kye, Seong Min, Yoohwan Kwon, and Joon Son Chung. 2021. "Cross Attentive Pooling for Speaker Verification." In *2021 IEEE Spoken Language Technology Workshop (SLT),* 294–300. doi:10.1109/SLT48900.2021.9383565. arXiv: 2008.05983 [cs, eess].

Li, Ruirui, Jyun-Yu Jiang, Jiahao Liu Li, Chu-Cheng Hsieh, and Wei Wang. 2020. "Automatic Speaker Recognition with Limited Data." In *Proceedings of the 13th International Conference on Web Search and Data Mining,* 340–348. Houston TX USA: ACM, January 20. ISBN: 978-1-4503-6822-3. doi:10.1145/3336191.3371802.

Li, Ruirui, Jyun-Yu Jiang, Xian Wu, Hongda Mao, Chu-Cheng Hsieh, and Wei Wang. 2020. "Bridging Mixture Density Networks with Meta-Learning for Automatic Speaker Identification." In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP),* 3522–3526. Barcelona, Spain: IEEE, May. ISBN: 978-1-5090-6631-5. doi:10.1109/ICASSP40776.2020.9054111.

Li, Xiaoxu, Zhuo Sun, Jing-Hao Xue, and Zhanyu Ma. 2020. "A Concise Review of Recent Few-Shot Meta-Learning Methods." *Neurocomputing.* ISSN: 0925-2312. doi:10.1016/j.neucom.2020.05.114. arXiv: 2005.10953 [cs, stat].

Livingstone, Steven R., and Frank A. Russo. 2018. "The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS): A Dynamic, Multimodal Set of Facial and Vocal Expressions in North American English." Edited by Joseph Najbauer. *PLOS ONE* 13 (5): e0196391. ISSN: 1932-6203. doi:10.1371/journal.pone.0196391.

Loshchilov, Ilya, and Frank Hutter. 2019. "Decoupled Weight Decay Regularization." January 4. arXiv: 1711.05101 [cs, math].

Manocha, Pranay, Rohan Badlani, Anurag Kumar, Ankit Shah, Benjamin Elizalde, and Bhiksha Raj. 2018. "Content-Based Representations of Audio Using Siamese Neural Networks." In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP),* 3136–3140. Calgary, AB: IEEE, April. ISBN: 978-1-5386-4658-8. doi:10.1109/ICASSP.2018.8461524.

Martin, O., I. Kotsia, B. Macq, and I. Pitas. 2006. "The eNTERFACE'05 Audio-Visual Emotion Database." In *22nd International Conference on Data Engineering Workshops (ICDEW'06),* 8–8. Atlanta, GA, USA: IEEE. ISBN: 978-0-7695-2571-6. doi:10.1109/ICDEW.2006.145.

Mesaros, Annamaria, Toni Heittola, and Tuomas Virtanen. 2016. "TUT Database for Acoustic Scene Classification and Sound Event Detection." In *2016 24th European Signal Processing Conference (EUSIPCO),* 1128–1132. Budapest, Hungary: IEEE, August. ISBN: 978-0-9928626-5-7. doi:10.1109/EUSIPCO.2016.7760424.

Micikevicius, Paulius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, et al. 2018. "Mixed Precision Training." February 15. arXiv: 1710.03740 [cs, stat].

Mimilakis, Stylianos I., Nicholas J. Bryan, and Paris Smaragdis. 2020. "One-Shot Parametric Audio Production Style Transfer with Application to Frequency Equalization." In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP),* 256–260. Barcelona, Spain: IEEE, May. ISBN: 978-1-5090-6631-5. doi:10.1109/ICASSP40776.2020.9054108.

Mishra, Prateek. 2020. "Few Shot Text - Independent Speaker Verification Using 3D - CNN." arXiv: 2008.11088 [eess.AS].

Mohri, Mehryar, Afshin Rostamizadeh, and Ameet Talwalkar. 2018. *Foundations of Machine Learning 2nd edition.* MIT Press. ISBN: 9780262039406, accessed October 12, 2020. https://cs.nyu.edu/~mohri/mlbook/.

Mysore, Gautham J. 2015. "Can We Automatically Transform Speech Recorded on Common Consumer Devices in Real-World Environments into Professional Production Quality Speech? A Dataset, Insights, and Challenges." *IEEE Signal Processing Letters* 22 (8): 1006–1010. ISSN: 1070-9908, 1558-2361. doi:10.1109/LSP.2014.2379648.

Nagrani, Arsha, Joon Son Chung, Jaesung Huh, Andrew Brown, Ernesto Coto, Weidi Xie, Mitchell McLaren, Douglas A. Reynolds, and Andrew Zisserman. 2020. "VoxSRC 2020: The Second VoxCeleb Speaker Recognition Challenge." December 12. arXiv: 2012.06867 [cs, eess].

Nagrani, Arsha, Joon Son Chung, Weidi Xie, and Andrew Zisserman. 2020. "Voxceleb: Large-Scale Speaker Verification in the Wild." *Computer Speech & Language* 60:101027. ISSN: 08852308. doi:10.1016/j.csl.2019.101027.

Nagrani, Arsha, Joon Son Chung, and Andrew Zisserman. 2017. "VoxCeleb: A Large-Scale Speaker Identification Dataset." In *Proc. Interspeech 2017,* 2616–2620. doi:10.21437/Interspeech.2017-950. arXiv: 1706.08612 [cs].

Okabe, Koji, Takafumi Koshinaka, and Koichi Shinoda. 2018. "Attentive Statistics Pooling for Deep Speaker Embedding." In *Interspeech 2018,* 2252–2256. ISCA, September 2. doi:10.21437/Interspeech.2018-993.

Palanisamy, Kamalesh, Dipika Singhania, and Angela Yao. 2020. "Rethinking CNN Models for Audio Classification." November 13. arXiv: 2007.11154 [cs, eess].

Panayotov, Vassil, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. 2015. "Librispeech: An ASR Corpus Based on Public Domain Audio Books." In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP),* 5206–5210. South Brisbane, Queensland, Australia: IEEE, April. ISBN: 978-1-4673-6997-8. doi:10.1109/ICASSP.2015.7178964.

Park, Daniel S., William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D. Cubuk, and Quoc V. Le. 2019. "SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition." *Interspeech 2019:* 2613–2617. doi:10.21437/Interspeech.2019-2680. arXiv: 1904.08779.

Paszke, Adam, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, et al. 2019. "PyTorch: An Imperative Style, High-Performance Deep Learning Library." In *Advances in Neural Information Processing Systems 32,* edited by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, 8024–8035. Curran Associates, Inc. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Pons, Jordi, Joan Serra, and Xavier Serra. 2019. "Training Neural Audio Classifiers with Few Data." In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP),* 16–20. Brighton, United Kingdom: IEEE, May. ISBN: 978-1-4799-8131-1. doi:10.1109/ICASSP.2019.8682591.

Ravanelli, Mirco, and Yoshua Bengio. 2018. "Speaker Recognition from Raw Waveform with SincNet." In *2018 IEEE Spoken Language Technology Workshop (SLT),* 1021–1028. Athens, Greece: IEEE, December. ISBN: 978-1-5386-4334-1. doi:10.1109/SLT.2018.8639585.

Richey, Colleen, Maria A. Barrios, Zeb Armstrong, Chris Bartels, Horacio Franco, Martin Graciarena, Aaron Lawson, et al. 2018. "Voices Obscured in Complex Environmental Settings (VOiCES) Corpus." In *Interspeech 2018,* 1566–1570. ISCA, September 2. doi:10.21437/Interspeech.2018-1454.

Rozenberg, Shai, Hagai Aronowitz, and Ron Hoory. 2020. "Siamese X-Vector Reconstruction for Domain Adapted Speaker Recognition": 1526–1529. doi:10.21437/Interspeech.2020-1742.

Sadjadi, Seyed Omid, Craig S. Greenberg, Elliot Singer, Douglas A. Reynolds, and Lisa Mason. 2020. "NIST 2020 CTS Speaker Recognition Challenge Evaluation Plan."

Shimada, Kazuki, Yuichiro Koyama, and Akira Inoue. 2020. "Metric Learning with Background Noise Class for Few-Shot Detection of Rare Sound Events." In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP),* 616–620. Barcelona, Spain: IEEE, May. ISBN: 978-1-5090-6631-5. doi:10.1109/ICASSP40776.2020.9054712.

Smith, Leslie N., and Nicholay Topin. 2018. “Super-Convergence: Very Fast Training of Neural Networks Using Large Learning Rates.” May 17. arXiv: 1708.07120 [cs, stat].

Snell, Jake, Kevin Swersky, and Richard Zemel. 2017. “Prototypical Networks for Few-Shot Learning.” In *Proceedings of the 31st International Conference on Neural Information Processing Systems,* 4080–4090. NIPS’17. Long Beach, California, USA: Curran Associates Inc. ISBN: 9781510860964. arXiv: 1703.05175 [cs.LG].

Snyder, David, Guoguo Chen, and Daniel Povey. 2015. “MUSAN: A Music, Speech, and Noise Corpus.” October 28. arXiv: 1510.08484 [cs].

Snyder, David, Daniel Garcia-Romero, Gregory Sell, Alan McCree, Daniel Povey, and Sanjeev Khudanpur. 2019. “Speaker Recognition for Multi-Speaker Conversations Using X-Vectors.” In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP),* 5796–5800. Brighton, United Kingdom: IEEE, May. ISBN: 978-1-4799-8131-1. doi:10.1109/ICASSP.2019.8683760.

Stowell, Dan, Dimitrios Giannoulis, Emmanouil Benetos, Mathieu Lagrange, and Mark D. Plumbley. 2015. “Detection and Classification of Acoustic Scenes and Events.” *IEEE Transactions on Multimedia* 17 (10): 1733–1746. ISSN: 1520-9210, 1941-0077. doi:10.1109/TMM.2015.2428998.

The Parliament of Finland. 2017. *Plenary Sessions of the Parliament of Finland, downloadable version 1.* speech corpus. http://urn.fi/urn:nbn:fi:lb-2017030901.

Vestman, Ville. 2020. “Methods for fast, robust, and secure speaker recognition.” PhD diss., University of Eastern Finland. http://urn.fi/URN:ISBN:978-952-61-3484-0.

Wang, Yaqing, Quanming Yao, James Kwok, and Lionel M. Ni. 2020. “Generalizing from a Few Examples: A Survey on Few-Shot Learning.” *ACM Computing Surveys* (New York, NY, USA) 53, no. 3 (June). ISSN: 0360-0300. doi:10.1145/3386252. arXiv: 1904.05046 [cs].

Wang, Yu, Justin Salamon, Nicholas J. Bryan, and Juan Pablo Bello. 2020. “Few-Shot Sound Event Detection.” In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP),* 81–85. Barcelona, Spain: IEEE, May. ISBN: 978-1-5090-6631-5. doi:10.1109/ICASSP40776.2020.9054708.

Xie, Weidi, Arsha Nagrani, Joon Son Chung, and Andrew Zisserman. 2019. “Utterance-Level Aggregation for Speaker Recognition in the Wild.” In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP),* 5791–5795. Brighton, United Kingdom: IEEE, May. ISBN: 978-1-4799-8131-1. doi:10.1109/ICASSP.2019.8683120.

Zeghidour, Neil, Nicolas Usunier, Gabriel Synnaeve, Ronan Collobert, and Emmanuel Dupoux. 2018. “End-to-End Speech Recognition From the Raw Waveform.” In *Proc. Interspeech 2018,* 781–785. doi:10.21437/Interspeech.2018-2414. arXiv: 1806.07098 [cs, eess].

Zhang, Yichi, and Zhiyao Duan. 2017. “IMINET: Convolutional Semi-Siamese Networks for Sound Search by Vocal Imitation.” In *2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA),* 304–308. New Paltz, NY: IEEE, October. ISBN: 978-1-5386-1632-1. doi:10.1109/WASPAA.2017.8170044.

Zhang, Yichi, Bryan Pardo, and Zhiyao Duan. 2019. “Siamese Style Convolutional Neural Networks for Sound Search by Vocal Imitation.” *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 27, no. 2 (February): 429–441. ISSN: 2329-9290, 2329-9304. doi:10.1109/TASLP.2018.2868428.

Zhong, Yujie, Relja Arandjelovi, and Andrew Zisserman. 2019. “GhostVLAD for Set-Based Face Recognition.” In *Computer Vision – ACCV 2018,* 35–50. Cham: Springer International Publishing. ISBN: 978-3-030-20890-5. doi:10.1007/978-3-030-20890-5_3. arXiv: 1810.09951 [cs].

Appendices

A Benchmark results

		Training parameters			Results (4s)		Results (2s)		Results (full)	
Model	Encoder	Training loss	Speakers	Aug.	EER	minDCF	EER	minDCF	EER	minDCF
Fast ResNet-34	SAP	AngularProto	5994		2.40%	0.1288	2.09%	0.1036	2.65%	0.1055
H / ASP	ASP	SoftmaxProto	5994	✓	0.83%	0.0317	0.66%	0.0179	0.96%	0.0335
	GhostVLAD	BCE	251		17.37%	0.9948	17.62%	0.9952	13.15%	0.7830
	ASP	BCE	251		17.10%	1.0	17.10%	1.0	14.39%	0.9178
	SAP	BCE	251		16.61%	0.9976	18.03%	1.0	40.81%	1.0
	SAP	BCE	251		17.10%	0.9851	17.10%	0.9851	44.18%	1.0
	SAP	BCE	100		23.65%	1.0	23.55%	1.0	21.82%	1.0
	SAP	BCE	50		30.25%	1.0	29.65%	1.0	26.78%	1.0
	SAP	BCE	251	✓	13.39%	0.9916	14.29%	0.9824	9.43%	0.8530
	SAP	BCE	251	✓	10.41%	0.8186	13.80%	0.9386	9.02%	0.7806
	SAP	AngularProto	251		8.11%	0.5125	8.92%	0.5321	6.80%	0.4312
	SAP	AngularProto	251		6.69%	0.5297	7.01%	0.5425	5.27%	0.3977
	SAP	AngularProto	251	✓	5.38%	0.3485	4.62%	*0.2671*	2.57%	0.1601
	ASP	AngularProto	251		9.09%	0.4452	8.36%	0.4658	6.52%	0.3870
	ASP	AngularProto	100		8.84%	0.5753	8.67%	0.5832	6.94%	0.4803
	ASP	AngularProto	50		14.78%	0.7937	13.80%	0.7669	11.58%	0.6602
	ASP	AngularProto	251	✓	5.03%	0.3551	5.27%	0.3284	2.39%	0.1846
	GhostVLAD	AngularProto	251		13.88%	0.7846	14.39%	0.7474	12.73%	0.6276
	SAP	SoftmaxProto	251		8.50%	0.4917	8.60%	0.4933	7.18%	0.5094
	ASP	SoftmaxProto	251		7.18%	0.5414	6.21%	0.5745	5.06%	0.4923
	ASP	SoftmaxProto	100		10.82%	0.6650	10.41%	0.6240	8.67%	0.5413
	ASP	SoftmaxProto	50		14.39%	0.7929	13.80%	0.8060	10.75%	0.5559
	ASP	SoftmaxProto	251	✓	*4.62%*	*0.2861*	*3.88%*	0.3094	*1.84%*	*0.1591*

Table 19: Test results on the LibriSpeech test-other set (2883 pairs, 33 speakers), with sample lengths of 4, 2 seconds and the full sample. The best overall result of models trained for this thesis is bolded. Fast ResNet-34 (Chung et al. 2020) and H / ASP (Heo et al. 2020) are pretrained reference models.

		Training parameters			Results (4s)		Results (2s)		Results (full)	
Model	Encoder	Training loss	Speakers	Aug.	EER	minDCF	EER	minDCF	EER	minDCF
Fast ResNet-34	SAP	AngularProto	5994		2.85%	0.0814	2.39%	0.0821	2.76%	0.0978
H / ASP	ASP	SoftmaxProto	5994	✓	0.85%	0.0404	0.92%	0.0243	1.19%	0.0657
	GhostVLAD	BCE	251		20.33%	0.9781	21.68%	0.9630	15.63%	0.8327
	ASP	BCE	251		21.14%	1.0	21.14%	1.0	16.35%	1.0
	SAP	BCE	251		18.84%	0.9732	21.95%	0.9993	40.89%	1.0
	SAP	BCE	251		16.28%	0.9509	16.28%	0.9509	44.03%	1.0
	SAP	BCE	100		33.28%	1.0	31.17%	1.0	28.84%	1.0
	SAP	BCE	50		36.95%	1.0	36.74%	1.0	38.42%	1.0
	SAP	BCE	251	✓	22.33%	0.9810	22.41%	0.9934	17.54%	0.9866
	SAP	BCE	251	✓	16.28%	0.9418	19.76%	0.9941	13.60%	0.8718
	SAP	AngularProto	251		7.16%	*0.5301*	7.54%	0.5298	**4.67**%	0.3642
	SAP	AngularProto	251		8.73%	0.7632	7.81%	0.7092	5.52%	0.5623
	SAP	AngularProto	251	✓	9.92%	0.6915	10.22%	0.7253	6.51%	0.4975
	ASP	AngularProto	251		9.27%	0.6999	8.92%	0.7652	6.78%	0.5898
	ASP	AngularProto	100		8.38%	0.7395	8.81%	0.7280	4.87%	0.6419
	ASP	AngularProto	50		14.06%	0.8641	14.44%	0.8622	9.38%	0.6107
	ASP	AngularProto	251	✓	9.11%	0.7202	9.38%	0.6853	6.24%	0.5347
	GhostVLAD	AngularProto	251		12.30%	0.7987	11.41%	0.7993	9.65%	0.6993
	SAP	SoftmaxProto	251		7.62%	0.6139	7.62%	0.6525	4.87%	0.5292
	ASP	SoftmaxProto	251		**6.32**%	0.5350	**6.32**%	0.5466	5.32%	**0.3443**
	ASP	SoftmaxProto	100		9.84%	0.5625	9.27%	**0.5063**	6.36%	0.3703
	ASP	SoftmaxProto	50		13.60%	0.8493	13.33%	0.8361	7.92%	0.5594
	ASP	SoftmaxProto	251	✓	9.19%	0.6982	10.00%	0.7317	6.62%	0.6406

Table 20: Test results on the LibriSpeech test-clean set (2611 pairs, 40 speakers), with sample lengths of 4, 2 seconds and the full sample. The best overall result of models trained for this thesis is bolded. Fast ResNet-34 (Chung et al. 2020) and H / ASP (Heo et al. 2020) are pretrained reference models.

		Training parameters			Results (4s)		Results (2s)		Results (full)	
Model	Encoder	Training loss	Speakers	Aug.	EER	minDCF	EER	minDCF	EER	minDCF
Fast ResNet-34	SAP	AngularProto	5994		2.18%	0.1690	2.35%	0.1822	2.39%	0.1699
H / ASP	ASP	SoftmaxProto	5994	✓	1.18%	0.0865	1.35%	0.0933	1.30%	0.0981
	GhostVLAD	BCE	251		33.15%	0.9996	33.38%	0.9998	30.22%	0.9988
	ASP	BCE	251		32.87%	1.0	32.87%	1.0	30.17%	1.0
	SAP	BCE	251		29.18%	0.9997	32.15%	1.0	45.99%	1.0
	SAP	BCE	251		32.44%	0.9999	32.44%	0.9999	46.44%	1.0
	SAP	BCE	100		36.00%	1.0	36.69%	1.0	33.10%	1.0
	SAP	BCE	50		39.60%	1.0	39.85%	1.0	37.84%	1.0
	SAP	BCE	251	✓	29.13%	0.9998	29.38%	0.9995	25.32%	0.9994
	SAP	BCE	251	✓	26.41%	0.9997	29.48%	1.0	25.24%	0.9995
	SAP	AngularProto	251		31.43%	0.9999	31.37%	1.0	28.94%	0.9943
	SAP	AngularProto	251		32.37%	0.9944	31.98%	0.9983	30.83%	0.9983
	SAP	AngularProto	251	✓	22.75%	0.9598	22.90%	0.9496	16.98%	0.8310
	ASP	AngularProto	251		30.78%	0.9967	30.95%	0.9970	28.36%	0.9876
	ASP	AngularProto	100		32.66%	0.9945	32.43%	0.9968	29.01%	0.9850
	ASP	AngularProto	50		32.93%	0.9984	33.09%	0.9990	29.05%	0.9871
	ASP	AngularProto	251	✓	22.25%	0.9559	22.55%	0.9544	16.18%	*0.7938*
	GhostVLAD	AngularProto	251		32.07%	0.9998	31.99%	0.9995	30.20%	0.9986
	SAP	SoftmaxProto	251		29.78%	0.9968	29.80%	0.9986	26.16%	0.9888
	ASP	SoftmaxProto	251		29.64%	0.9948	29.55%	0.9941	26.86%	0.9696
	ASP	SoftmaxProto	100		31.43%	0.9958	31.36%	0.9963	27.83%	0.9827
	ASP	SoftmaxProto	50		31.23%	0.9993	31.26%	0.9998	26.76%	0.9545
	ASP	SoftmaxProto	251	✓	*22.20%*	*0.9217*	*21.89%*	*0.9201*	*15.96%*	0.8001

Table 21: Test results on the VoxCeleb1 test set (37720 pairs, 40 speakers), with sample lengths of 4, 2 seconds and the full sample. The best overall result of models trained for this thesis is bolded. Fast ResNet-34 (Chung et al. 2020) and H / ASP (Heo et al. 2020) are pretrained reference models.

		Training parameters			Results (4s)		Results (2s)		Results (full)	
Model	Encoder	Training loss	Speakers	Aug.	EER	minDCF	EER	minDCF	EER	minDCF
Fast ResNet-34	SAP	AngularProto	5994		6.36%	0.2356	7.17%	0.2565	5.78%	0.2360
H / ASP	ASP	SoftmaxProto	5994	✓	5.86%	0.1214	6.45%	0.1470	5.43%	0.1271
	GhostVLAD	BCE	251		31.56%	0.9999	31.74%	0.9999	27.41%	0.9995
	ASP	BCE	251		32.13%	1.0	32.69%	1.0	28.04%	1.0
	SAP	BCE	251		27.97%	1.0	31.65%	1.0	47.67%	1.0
	SAP	BCE	251		28.39%	0.9994	28.63%	0.9997	45.15%	1.0
	SAP	BCE	100		37.68%	0.9999	37.87%	1.0	33.94%	1.0
	SAP	BCE	50		41.75%	0.9999	41.84%	1.0	40.55%	1.0
	SAP	BCE	251	✓	29.77%	0.9990	30.63%	0.9997	25.17%	0.9991
	SAP	BCE	251	✓	25.29%	0.9999	29.03%	0.9990	23.18%	1.0
	SAP	AngularProto	251		26.49%	1.0	25.73%	1.0	20.60%	1.0
	SAP	AngularProto	251		28.13%	0.9997	27.83%	1.0	22.44%	0.9999
	SAP	AngularProto	251	✓	21.92%	1.0	21.91%	0.9990	17.02%	0.9999
	ASP	AngularProto	251		26.01%	1.0	26.01%	0.9991	21.29%	0.9999
	ASP	AngularProto	100		28.01%	0.9993	27.82%	*0.9987*	22.36%	0.9999
	ASP	AngularProto	50		28.99%	0.9996	28.84%	0.9996	24.89%	0.9985
	ASP	AngularProto	251	✓	21.85%	*0.9895*	21.94%	0.9988	16.95%	*0.9877*
	GhostVLAD	AngularProto	251		31.56%	0.9990	31.58%	0.9997	27.44%	0.9972
	SAP	SoftmaxProto	251		25.13%	0.9999	25.28%	1.0	18.71%	1.0
	ASP	SoftmaxProto	251		25.34%	0.9960	24.94%	0.9994	21.21%	0.9996
	ASP	SoftmaxProto	100		26.78%	0.9997	26.41%	0.9988	21.97%	0.9982
	ASP	SoftmaxProto	50		29.16%	0.9991	29.41%	0.9988	23.93%	0.9913
	ASP	SoftmaxProto	251	✓	*21.43%*	0.9990	*21.42%*	0.9999	*16.51%*	0.9996

Table 22: Test results on the Eduskunta test set (13640 pairs, 171 speakers), with sample lengths of 4, 2 seconds and the full sample. The best overall result of models trained for this thesis is bolded. Fast ResNet-34 (Chung et al. 2020) and H / ASP (Heo et al. 2020) are pretrained reference models. Fast ResNet-34 was trained using 40 mel input spectrogram, all of the other models were trained on 64 mels.

	Training parameters			Results (4s)		Results (2s)		Results (full)	
Encoder	Training loss	Speakers	Aug.	EER	minDCF	EER	minDCF	EER	minDCF
SAP	BCE	251		12.14%	0.8702	12.83%	0.9636	10.92%	0.7582
SAP	BCE	251	✓[1]	14.64%	0.9886	14.71%	0.9994	11.83%	0.9940
ASP	BCE	251	✓[1]	12.97%	0.9444	12.14%	0.9137	10.09%	0.7964
ASP[2]	AngularProto	251	✓	3.47%	0.2382	4.20%	0.2812	2.88%	0.1903
ASP[3]	AngularProto	251	✓	2.08%	**0.1409**	3.81%	0.2211	2.39%	0.1388
ASP[4]	AngularProto	251	✓	4.23%	0.2806	5.03%	0.2865	3.54%	0.2111
ASP	AngularProto	251	✓	2.71%	0.1471	3.88%	**0.1894**	2.64%	0.1557
ASP	SoftmaxProto	251	✓	**1.98%**	0.1436	**2.64%**	0.2062	**1.66%**	**0.1058**

Table 23: Test results on the LibriSpeech test-other set (2883 pairs, 33 speakers), with sample lengths of 4, 2 seconds and the full sample. The best overall result of models trained for this thesis is bolded. 1. Model using GPU optimized augmentations. 2. 32-way 4-shot. 3. Trained on 3 second samples. 4. Uses the fast-ResNet34 network.

	Training parameters			Results (4s)		Results (2s)		Results (full)	
Encoder	Training loss	Speakers	Aug.	EER	minDCF	EER	minDCF	EER	minDCF
SAP	BCE	251		16.74%	0.9974	16.09%	0.9915	12.49%	0.9554
SAP	BCE	251	✓[1]	19.22%	0.9947	19.22%	0.9888	14.79%	0.9019
ASP	BCE	251	✓[1]	13.87%	0.6811	14.36%	0.8085	9.65%	*0.4637*
ASP[2]	AngularProto	251	✓	7.54%	0.6555	8.73%	0.7475	6.36%	0.6739
ASP[3]	AngularProto	251	✓	**5.90%**	0.5774	8.08%	0.6121	*5.25%*	0.5621
ASP[4]	AngularProto	251	✓	8.81%	**0.5029**	9.92%	*0.5510*	7.08%	0.4790
ASP	AngularProto	251	✓	6.32%	0.5059	*7.16%*	0.5525	5.71%	0.5187
ASP	SoftmaxProto	251	✓	7.16%	0.6139	8.00%	0.6086	5.59%	0.6670

Table 24: Test results on the LibriSpeech test-clean set (2611 pairs, 40 speakers), with sample lengths of 4, 2 seconds and the full sample. The best overall result of models trained for this thesis is bolded. 1. Model using GPU optimized augmentations. 2. 32-way 4-shot. 3. Trained on 3 second samples. 4. Uses the fast-ResNet34 network.

	Training parameters			Results (4s)		Results (2s)		Results (full)	
Encoder	Training loss	Speakers	Aug.	EER	minDCF	EER	minDCF	EER	minDCF
SAP	BCE	251		27.68%	1.0	29.41%	0.9997	27.53%	1.0
SAP	BCE	251	✓[1]	30.46%	1.0	31.96%	1.0	30.37%	0.9996
ASP	BCE	251	✓[1]	28.64%	0.9999	30.74%	0.9998	27.88%	0.9996
ASP[2]	AngularProto	251	✓	18.36%	0.8699	22.05%	0.9176	17.49%	0.8499
ASP[3]	AngularProto	251	✓	16.95%	0.8109	20.91%	0.8986	15.51%	0.7724
ASP[4]	AngularProto	251	✓	20.12%	0.7881	22.72%	0.8450	18.98%	0.7838
ASP	AngularProto	251	✓	17.50%	0.7961	20.51%	0.8921	16.58%	0.7624
ASP	SoftmaxProto	251	✓	**16.59%**	**0.7242**	**19.50%**	**0.8344**	**15.32%**	**0.6866**

Table 25: Test results on the VoxCeleb1 test set (37720 pairs, 40 speakers), with sample lengths of 4, 2 seconds and the full sample. The best overall result of models trained for this thesis is bolded. 1. Model using GPU optimized augmentations. 2. 32-way 4-shot. 3. Trained on 3 second samples. 4. Uses the fast-ResNet34 network.

	Training parameters			Results (4s)		Results (2s)		Results (full)	
Encoder	Training loss	Speakers	Aug.	EER	minDCF	EER	minDCF	EER	minDCF
SAP	BCE	251		28.05%	1.0	29.97%	1.0	27.28%	1.0
SAP	BCE	251	✓[1]	33.17%	1.0	33.26%	1.0	31.13%	1.0
ASP	BCE	251	✓[1]	27.57%	1.0	29.72%	1.0	24.69%	0.9999
ASP[2]	AngularProto	251	✓	**17.42%**	0.9999	21.84%	0.9991	17.42%	0.9999
ASP[3]	AngularProto	251	✓	17.93%	**0.9808**	21.09%	0.9973	16.58%	0.9875
ASP[4]	AngularProto	251	✓	17.51%	0.9993	**20.48%**	1.0	**16.02%**	0.9999
ASP	AngularProto	251	✓	19.15%	0.9873	21.62%	0.9955	18.33%	0.9807
ASP	SoftmaxProto	251	✓	17.98%	0.9850	20.62%	**0.9873**	16.85%	**0.9874**

Table 26: Test results on the Eduskunta test set (13640 pairs, 171 speakers), with sample lengths of 4, 2 seconds and the full sample. The best overall result of models trained for this thesis is bolded. 1. Model using GPU optimized augmentations. 2. 32-way 4-shot. 3. Trained on 3 second samples. 4. Uses the fast-ResNet34 network.