POLITECNICO DI TORINO

DEPARTMENT OF CONTROL AND COMPUTER ENGINEERING

Master of Science in Computer Engineering

Master of Science's Degree Thesis

Deep neural networks for language recognition

Supervisor

prof. Sandro CUMANI

Candidate

Axel BARON

Co-supervisor

Salvatore SARNI

March 2023

Summary

This work belongs in the field of Language Recognition, which ultimately aims to determine the language of a given speech sample.

This work mostly focuses on the study of acoustic feature extraction and of Deep neural network technologies in order to solve language recognition problems. In particular, the Log-mel extraction technique and the ECAPA-TDNN model are the main technologies that drew my attention and motivated this work.

This document begins by introducing the concepts and history of Language Recognition. Then I explain what acoustic features are, their purpose and the specific case of log-mel features. To follow there is a report about Neural Networks which slides towards the more complex case of Deep neural networks and the case of ECAPA-TDNN model.

In the end, there is my experimental setup, the decisions I made to treat this subject as well as the analysis of my results.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Language Recognition

Language recognition, also known as language identification, is a field of study that focuses on automatically determining the language of a given text or speech sample. This technology has numerous applications, including spell checkers, machine translation systems, and multi-lingual information retrieval systems. With the growing amount of digital text data being generated every day, language recognition plays a critical role in enabling these systems to effectively process and analyze this data. The goal of language recognition is to accurately identify the language of a text or speech sample with high accuracy, taking into account the complexities of natural language processing.

In this work, I focused on a supervised, closed-set language identification task. In supervised learning, the algorithm is provided with a dataset that includes both the input data (also called features) and the corresponding output (also called labels) for each example. The model then uses this data to learn a mapping function between the input features and the output labels. The goal of the model is to predict the correct output for new, unseen input data. The training process involves adjusting the model's parameters based on the discrepancies between the predicted output and the actual output for each training example. Closed-set means that it is impossible for the model to predict a label out of the given predefined set of labels.

1.2 Motivations

Language recognition's history (Chapter 2) illustrates that it has been less popular than its neighbour speaker verification. However, as speaker verification growth allows us to witness the appearance of new technologies, one may wonder : Can we apply these to improve the language identification task ?

A neural network called Emphasized Channel Attention, Propagation and Aggregation in TDNN(ECAPA-TDNN)[1] emerged and allowed researchers to obtain state-of-the-art performances on speaker verification. Log-mel became a more and more employed technique to produce quality acoustic features.

In this work we analyze the application of speaker-recognition-derived techniques for language identification. The development of huge datasets, such as Voxlingua107[2] that labels utterances with the spoken language, made that attempt of transitioning knowledge through different fields possible.

1.3 Chapters Outline

- Chapter 2 contains a brief description of previous works around language identification.
- Chapter 3 describes the different acoustic features techniques.
- Chapter 4 describes some neural network architectures that can be used in language recognition. It also studies the specific case of ECAPA-TDNN and how it deals with issues ensued from audio data.
- Chapter 5 explains the steps that led to the experimental setup as well as some results.
- Conclusions are drawn in Chapter 6

Chapter 2

Previous work on Language Recognition

Language recognition has made significant progress since the 1960s and 1970s when researchers first began exploring the use of computers for speech recognition. Early efforts in speech recognition relied heavily on hand-engineered features and rule-based methods. However, these early techniques were limited in their ability to handle the complexity and variability of human speech and required significant manual effort to build and maintain systems [3].

In recent years, advancements in deep learning have led to significant improvements in the performance of language recognition systems. Deep learning-based models can automatically learn complex and abstract features from raw speech data, leading to improved accuracy and reduced manual effort in building and maintaining systems [3].

Leena Mary[4] notes that one of the most important factors in language recognition is the extraction of prosodic features. Prosodic features refer to the suprasegmental aspects of speech such as intonation, stress, and rhythm, and they play a critical role in language identification. Mary[4] proposes a method for extracting prosodic features from speech data using a combination of signal processing techniques and machine learning algorithms.

Handbook of Natural Language Processing by Nitin Indurkhya and Fred J. Damerau[5] provides an overview of the various techniques used in natural language processing, including language identification. The book highlights the importance of developing effective algorithms for language identification and emphasizes the need for evaluating these algorithms using appropriate metrics.

K. Sreenivasa Rao and Dipanjan Nandi[6] propose a language identification system that uses excitation source features. The system first extracts excitation source features from speech data and then uses a support vector machine (SVM) to classify the language [6]. The authors show that their system outperforms other state-of-the-art language identification systems.

V. Ramu Reddy, K. Sreenivasa Rao, and Sudhamay Maity[7] propose a language identification system that uses spectral and prosodic features. The system first extracts spectral and prosodic features from speech data and then uses a decision tree to classify the language. The authors show that their system outperforms other state-of-the-art language identification systems.

Wu Chou and Biing-Hwang Juang[8] provide an overview of pattern recognition techniques in speech and language processing. The book covers various aspects of language

recognition, including speech feature extraction, statistical modeling, and machine learning algorithms.

Overall, language recognition has seen significant advancements in recent years, thanks to the development of deep learning-based models and the use of advanced signal processing techniques. These advancements have led to improved accuracy and reduced manual effort in building and maintaining language recognition systems. Prosodic features, such as intonation, stress, and rhythm, play a critical role in language identification, and various techniques have been proposed to extract these features. However, there is still room for improvement, and researchers continue to develop new algorithms and evaluate their performance using appropriate metrics.

Chapter 3

Acoustic features

3.1 Different extraction techniques

3.1.1 Preprocessing

Most of the extraction techniques described below require a preprocessing of the audio information. Given an acoustic signal x, it is necessary to apply changes in order to process it with a machine(cf. Chapter 16 of [3]). The following sections introduce and analyse some of the step that must be applied to an acoustic analog signal to obtain a representation we can work on[9] [10] [11].

Sampling

Sampling refers to the process of reducing a continuous-time signal to a discrete-time signal.

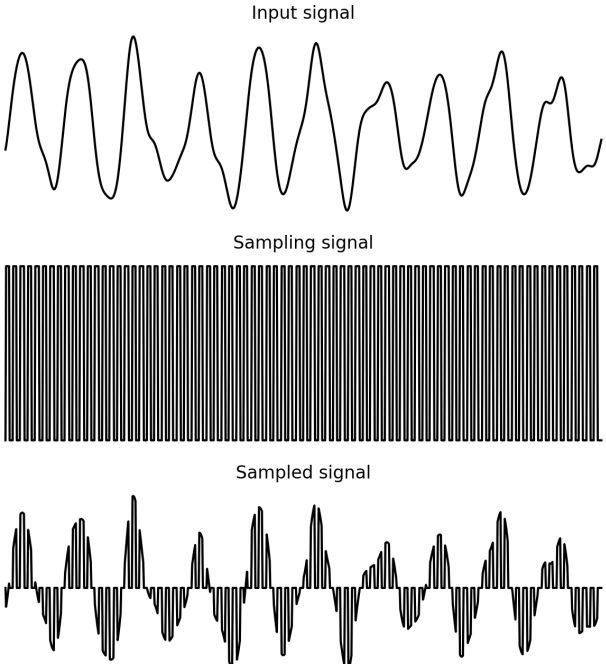


Figure 3.1. Sampling

It is done by performing a multiplication of the input signal x by a sequence of impulses $\sum_k \delta(t-kt_s)$ where t_s is the sampling interval linked to the sample rate f_s by the relation $t_s = \frac{1}{f_s}$.

$$x_s(k) = \sum_k [x(kt_s)\delta(t-kt_s)] \tag{3.1}$$

As it can be seen on Figure 3.1, the output signal after sampling(sampled signal) is riddled with holes. At first it seems like sampling a signal results in a loss of information. However, humans are mainly sensitive to frequencies lower than 4 kHz and the Nyquist theorem states that a signal is reconstructible if the sampling frequency is at least twice the highest frequency of the original signal. Then a sufficient sampling frequency f_s is 8kHz.

Quantization

Quantization transforms continuous values into discrete ones, thus it is always a lossy process.

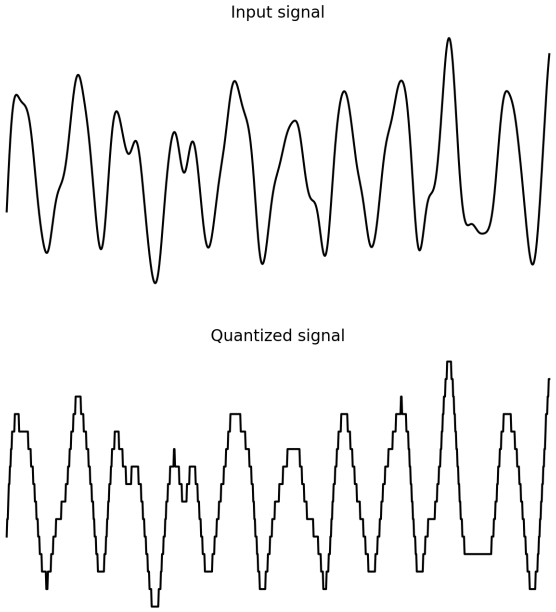


Figure 3.2. Quantization

The most straightforward way to do quantization is to divide the input range evenly and assign each value the index of the related interval (linear quantization). With this method, the quantization error corresponds to one least significant bit and, more importantly, its absolute value is consistent for any given input value. The amplitude distribution of the acoustic signal is highly non-linear. To address this issue, a logarithmic quantization is usually done. That is to say that the logarithm of the acoustic signal is linearly quantized. In this way, the relative quantization error becomes constant. Since the logarithmic function is not defined in zero, slightly different functions are used in practice, such as μ–law (used in American communication nets) or A–law (used in European communication nets) [9] [10].

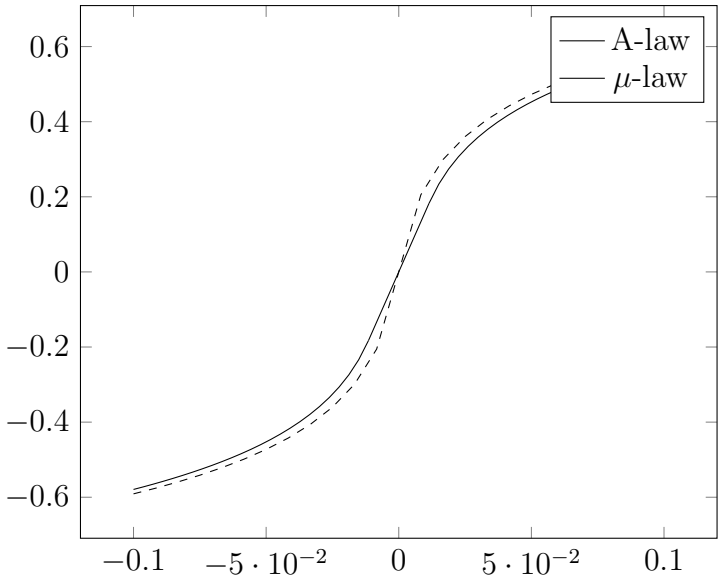


Figure 3.3. A and μ laws

For telephone speech usually values are represented in 8 bits. In practice, logarithmic quantization is done by performing linear quantization using a greater number of bits and then applying one of the laws in Figure 3.3 to map them to 8 bits.

Pre-emphasis

To flatten the signal spectrum in a specific frequency range and protect it against potential errors from finite precision during further processing, the discrete samples undergo filtering with a first-order pre-emphasis filter that has the transfer function

$$H(z) = 1 - az^{-1} \tag{3.2}$$

Where a is $0.9 \leq a \leq 1.0$. Most of the time a's value will be fixed at 0.95. Nevertheless, a's value could be set to adapt through time with a proper adaptation criterion.

Framing

In this step, the speech signal $\bar{s}$ that has undergone pre-emphasis is divided into blocks of N samples, with M samples in between each consecutive block.

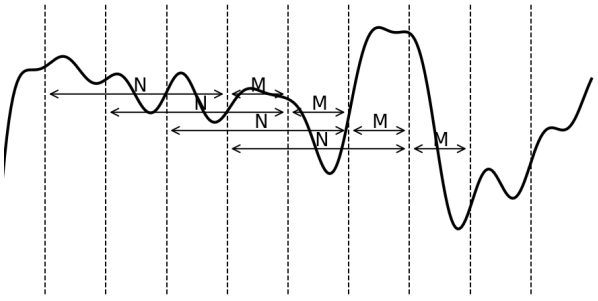


Figure 3.4. Frames overlapping

The process of dividing into blocks is shown in Figure 3.4 for the case when M equals (1/3)N. One frame and its direct following overlap by N - M samples. In the case of M > N, there will be loss of information; it should never happen as it would produce noise.

The resulting representation would be

$$x_l(n) = \bar{s}(Ml + n) \quad 0 \leq n \leq N-1 \quad 0 \leq l \leq L-1 \tag{3.3}$$

where $\begin{cases} \bar{s} \text{ is the signal that has gone through sampling, quantization, pre-emphasis} \\ \text{N is the grouping window size} \\ \text{L is the number of frames in the entire speech signal} \end{cases}$

Windowing

The act of dividing into frames leads to an alteration of the sample's spectrum, a phenomenon known as Gibbs phenomenon. To mitigate the impact of samples near the edges of a frame windowing is carried out. A commonly employed window is the Hamming window(cf. Chapter 16 of [3]) [11].

$$\bar{x}_l(n) = x_l(n)w(n) \quad 0 \leq n \leq N-1 \quad 0 \leq l \leq L-1 \tag{3.4}$$

where

$$w(n) = c + (1-c)\cos(\frac{n\pi}{N-1} - \frac{\pi}{2}) \quad 0 \leq n \leq N-1 \tag{3.5}$$

with $c = \frac{25}{46} \approx 0.54$

3.1.2 Linear Predictive Coding features

The foundation of the Linear Predictive Coding(LPC) model is the concept that a speech sample at time n, $s(n)$, can be estimated through a linear blend of previous p speech samples[10]

$$s(n) = \sum_{i=1}^{p} a_i s(n-i) + Gu(n) \tag{3.6}$$

where $u(n)$ is a normalized excitation, G is the gain of the excitation and a_1,...,a_p are constants over the speech analysis frame.

To obtain the LPC coefficients, the following steps must be applied to each frame of the windowed signal $\bar{x}_l(n)$, with $0 \leq n \leq N-1$.

Autocorrelation Analysis

Autocorrelation can be processed as

$$r(m) = \sum_{n=0}^{N-1-m} \bar{x}_l(n)\bar{x}_l(n+m) \quad m = 0,1,...,p \tag{3.7}$$

where p is the order of the LPC analysis determined by the highest value obtained from the autocorrelation analysis. Usually, p values in the range of 8 to 16 are utilized, with 8 being the most common value applied in various systems. An added advantage of the autocorrelation analysis is that it yields the energy of the l^{th} frame, represented by the zeroth autocorrelation, $R_l(0)$. This frame energy is a crucial parameter for language detection systems.

LPC Analysis

The following step in processing is the LPC analysis, which transforms each frame of $p+1$ autocorrelation values into an "LPC parameter set". This set can be represented by LPC coefficients, reflection coefficients (also known as PARCOR coefficients), log area ratio coefficients, cepstral coefficients, or any desired modification of these sets. The conversion from autocorrelation coefficients to an LPC parameter set is achieved through a technique known as Durbin's method, which can be expressed as an algorithm as follows:

$$E^{(0)} = r(0) \tag{3.8}$$

$$k_i = \frac{r(i) - \sum_{j=1}^{L-1} \alpha_j^{i-1} r(|i-j|)}{E^{i-1}} \quad 1 \leq i \leq p \tag{3.9}$$

$$\alpha_i^{(i)} = k_i \tag{3.10}$$

$$\alpha_j^{(i)} = \alpha_j^{(i-1)} - k_i \alpha_{i-j}^{(i-1)} \tag{3.11}$$

$$E^{(i)} = (1 - k_i^2) E^{(i-1)} \tag{3.12}$$

where the summation in eq. 3.9 is omitted for $i = 1$. These equations can be solved recursively for $i = 1, ..., p$ and the solution is

$$a_m = \text{LPC coefficients} = \alpha_m^{(p)} \quad 1 \leq m \leq p \tag{3.13}$$

$$k_m = \text{PARCOR coefficients} \tag{3.14}$$

$$g_m = \text{log area ratio coefficients} = \log(\frac{1 - k_m}{1 + k_m}) \tag{3.15}$$

One of the crucial LPC parameter sets that can be derived directly from the LPC coefficients is the LPC cepstral coefficients, denoted as $c(m)$. The process used to obtain this set is a recursion

$$c_0 = \ln \sigma^2 \tag{3.16}$$

$$c_m = a_m + \sum_{k=1}^{m-1} \frac{k}{m} c_k a_{m-k} \quad 1 \leq m \leq p \tag{3.17}$$

$$c_m = \sum_{k=1}^{m-1} \frac{k}{m} c_k a_{m-k} \quad m > p \tag{3.18}$$

where σ^2 is the gain term in the LPC model.

The cepstral coefficients, which are the coefficients of the log magnitude spectrum's Fourier transform representation, have been demonstrated[10] to be a more robust and reliable set of features for language recognition compared to the LPC coefficients, PARCOR coefficients, or log area ratio coefficients. Typically, a cepstral representation with $Q > p$ is utilized, where $Q \approx (3/2)p$.

3.1.3 log-Mel features

The following extraction techniques all derive from the log-mel features which itself derives from the Spectral features. Its aim is to provide a representation of the distribution of energy across different frequency bands. In this work, it must be remembered that all the following steps will be applied only to the representation given by the preprocess steps, I do not use raw signals.

Filter bank

This extraction technique is commonly called "Log-mel". It is the simplest way described in this thesis to provide representations that can be used for language identification using the logarithm of the Mel filterbank energies.

To begin with, apply the Fourier Transform to the windowed signal

$$x_f = \mathcal{F}(\bar{x}_l(n))(j) \quad 0 \leq n, j \leq N-1 \quad 0 \leq t, f \leq T-1 \tag{3.19}$$

Then the result will be processed by filters from a filter bank. There are different ways to build this filter bank. For example, there is the uniform filter bank where the i^{th} bandpass filter is defined by

$$f_i = \frac{F_s}{N} i \quad 1 \leq i \leq N_f \tag{3.20}$$

with N the number of uniformly spaced filters to span the frequency range of the speech and N_f the actual number of filters used in the filter bank.

However, filters that are positioned in a linear manner at lower frequencies and a logarithmic manner at higher frequencies have been utilized in speech recognition due to the varying sensitivity of the ear to different frequencies [12]. This allows for the capture of the key phonetic features of speech. Hence the creation of the Mel-scale that mimics the non-linear human ear perception of sound. In the ensuing work, the filter bank used is built by a succession of triangular filters following the Mel-scale[11].

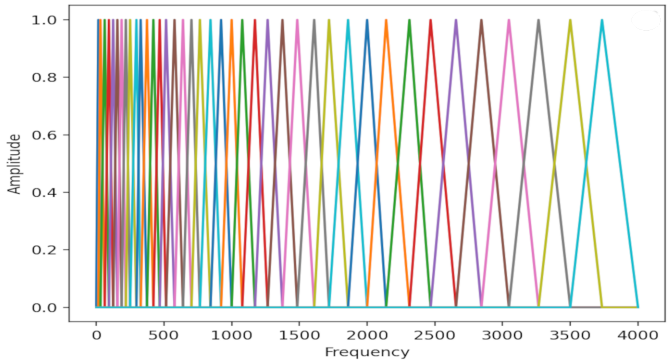


Figure 3.5. Filter bank on a Mel-Scale

For each band the energy of corresponding samples is evaluated as

$$E_i(f) = \sum_{j=L_i}^{H_i} |X_f(j)|^2 \quad 1 \leq i \leq N_f \tag{3.21}$$

where $E_i(f)$ is the energy of the i^{th} band, L_i is the lower bound of the corresponding band, H_i is the higher bound of the corresponding band, N_f the number of filters used in the filter bank.

After this step, the result is called Filter Bank features. It can be used as such for language recognition, or it could be modified further as described in the next steps.

Mel–Frequency Cepstral Coefficients

The filter bank coefficients obtained from the previous step tend to be highly correlated, which can create difficulties for certain machine learning models. To solve this issue, the Discrete Cosine Transform (DCT) can be applied to the filter bank coefficients to produce a condensed and uncorrelated representation of the filter banks

$$C_i(k) = \sum_{j=1}^{N_f} \log(E_j(k)) \cos[i(j - \frac{1}{2})\frac{\pi}{N_f}] \quad 0 \leq i \leq N_f - 1 \tag{3.22}$$

where $C_i(k)$ is the i^{th} MFCC for frame k[10].

Each frame's energy can be computed as

$$E(k) = \sum_{j=1}^{N_f} E_j(k) \tag{3.23}$$

The information contained in $C0(k)$ is already present in $E(k)$, so it is typically disregarded in speech recognition. Additionally, as the indices of the cepstral parameters increase, their variance decreases, meaning that higher index parameters carry less information and can be removed. Typically, between 12 and 24 cepstral parameters are used. To improve the modeling of the acoustic signal, MFCCs are often combined with their differential forms, which approximate the temporal change in MFCCs over a certain number of consecutive frames using the following polynomial expression

$$\Delta\bar{C}_i(k) = G \sum_{j=-N}^{N} j\bar{C}_i(k-j) \quad 1 \leq i \leq p \tag{3.24}$$

The gain factor G is applied in order to ensure that the set of cepstral and depstral parameters have similar variances. N is equal to half the size of the window employed to estimate the derivative. Similarly, it is possible to compute the differential energy as

$$\Delta E(k) = \sum_{j=-N}^{N} jE(k-j) \tag{3.25}$$

Advanced speech recognition systems typically also compute the second-order derivatives of cepstral coefficients in a similar manner. The combination of cepstral parameters, their first derivatives, and their second derivatives form the final feature vector used for analysis

$$O_t = \{\bar{C}_1(t), ..., \bar{C}_p(t), \Delta\bar{C}_1(t), ..., \Delta\bar{C}_p(t), \Delta\Delta\bar{C}_1(t), ..., \Delta\Delta\bar{C}_p(t), E(t), \Delta E(t), \Delta\Delta E(t)\} \tag{3.26}$$

In order to improve the robustness of speech recognition systems, various post-processing methods are frequently applied to the MFCCs, such as feature warping[9][13]. These methods aim to correct short-term distortions caused by noise and differences between the recording and playback channels.

Shifted Delta Coefficients

MFCCs have proven to be effective in language recognition, but incorporating Shifted Delta Cepstral (SDC) features can enhance language identification model performance[14][15]. SDCs add extra temporal information compared to standard MFCCs and are inspired by the success of phonotactic approaches, which rely on features that cover longer time spans than MFCCs. SDCs are defined by four parameters: N, d, P, and k. N represents the number of cepstral coefficients for each frame, d is the size of the delay used in delta computation, k is the number of blocks whose delta coefficients are combined in the final feature vectors, and P is the time shift between blocks[14][15]. The SDC coefficients ($\Delta cj(i,t)$) at time t are calculated as

$$\Delta c_j(i,t) = C_j(t+iP+d) - C_j(t+iP-d) \tag{3.27}$$

where $C_j(t)$ is the j-th MFCC coefficient for time t.

3.1.4 Prosodic features

Prosody is an area of study that focuses on the variations in tone and rhythm in language. These variations are used to convey different meanings and emotions, such as pragmatics, affect, and interaction during a conversation. The study of prosody deals with three main aspects of speech, namely energy, fundamental frequency, and duration. It's important to understand that energy refers to the acoustic quality that determines the loudness of speech, while the fundamental frequency is the frequency of the sound produced and is perceived as the pitch[16] of an utterance.(cf. Chapter 28 of [3])

Speaker recognition using prosodic and lexical features[17] went deeper into this analysis by focusing on different regions of interest : New Extraction Region Features(NERFs). NERF defines a sliding temporal region based on either a fixed window length or boundaries defined by the presence or values of other features. The pause-to-pause regions allow us to extract four types of prosodic features:

1. The duration features are calculated by examining the length of time it takes for phones to be pronounced within a given region. This may involve determining the longest phone or vowel in the region and normalizing it against the average duration for that vowel.

2. The pitch features are obtained by analyzing the pitch track of the audio signal and then using a refined version of the approach in [18] to stylize the pitch contours. Features such as the maximum stylized pitch, the final slope, and the mean pitch within the region are calculated.

3. The energy features are calculated by using a stylized version of the raw energy and creating a linear approximation of the energy contour over each segment of the stylized pitch. Features like the energy range and the first or last slope within the region are determined.

4. The pause features, such as the average pause within the region, the pause before or after the region, or the maximum pause within the region, are calculated.

[19] provides us with N-grams of Syllable based Nonuniform Extraction Region Features(SNERF-grams) that is a new approach to model prosodic information and could be used in language recognition. Figure 3.6 shows the steps to reproduce this model.

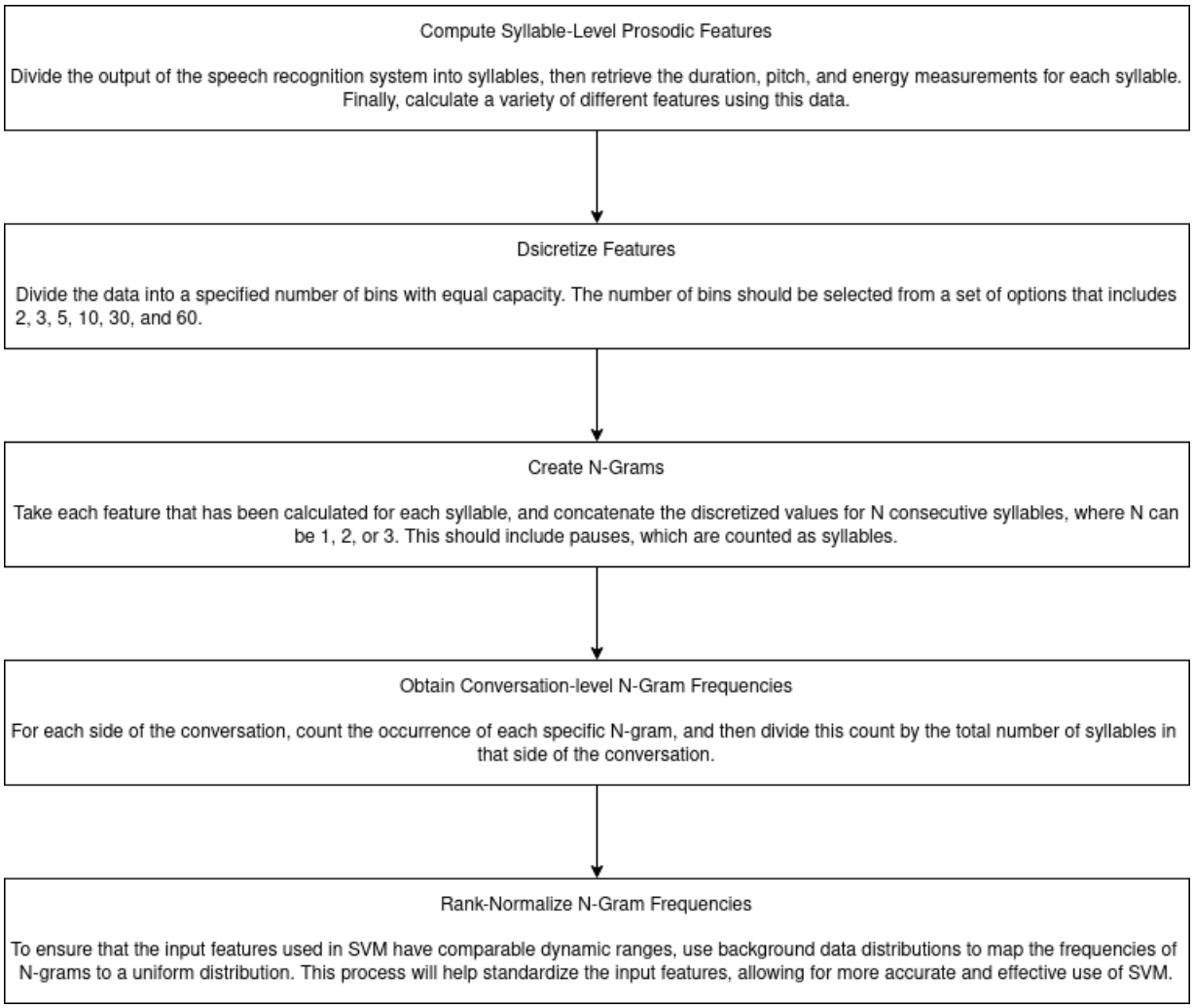


Figure 3.6. SNERF-gram's structure [19]

Syllable-level features

Prosody is an important aspect of language that involves the intonational and rhythmic elements of speech. These elements help to convey meaning beyond the words themselves, and play a role in expressing pragmatic, affective, or conversation-interactional meanings. Energy and fundamental frequency ($F0$) are two of the acoustic quantities that are studied in prosody, with energy referring to the acoustic quality we perceive as loudness and $F0$ referring to the frequency of the sound produced, which we hear as the pitch of an utterance.

To study prosodic features, the authors use a system called N-grams of Syllable-based Nonuniform Extraction Region Features (SNERF-gram). In this system, syllable regions are estimated using the output of a speech recognizer and a syllabification program called "tsylb2" [20] .This program uses human-created rules to determine the best matched dictionary pronunciation for each word in the speech signal.

For each syllable region, phone-level alignment information is obtained from the speech recognizer and a large number of features related to the duration, pitch, and energy values in the syllable are extracted. The duration features are obtained from the recognizer's

alignments, while the pitch is estimated using the "get_f0" function in ESPS/Waves [21] and post-processed using an approach adapted from [18]. This approach involves median filtering the pitch, fitting linear splines, and producing the posterior probability of pitch halving and pitch doubling for each frame using a log-normal tied-mixture model of the pitch. The model also estimates speaker pitch range parameters for normalization.

Energy features are obtained using the root mean square energy values from ESPS/Waves, and post-processed to fit a spline for each segment obtained from the pitch stylization. We note that this approach to energy stylization is suboptimal, but was used for convenience. The authors expect that using a better-fitting algorithm would only yield improved results. After extraction and stylization of these features, the authors of [19] created a number of duration, pitch, and energy features aimed at capturing basic prosodic patterns at the syllable level. They compute features that are highly correlated (differing only in normalization, binning, or N-gram length) because they do not know ahead of time which versions of a feature are best given robustness issues, or how those features interact with other features. Figure 3.7 provides an illustration of the process.

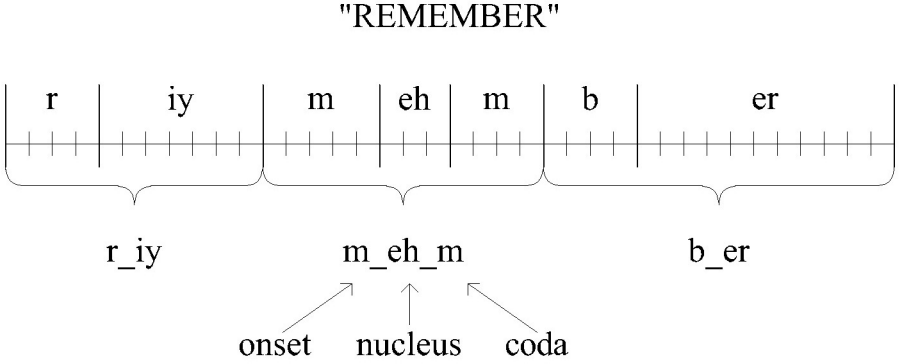


Figure 3.7. The image demonstrates how speech can be separated into syllables using the results of a speech recognition program. The illustration's smallest time segment used is 10 milliseconds, and the phone models being used require a minimum of 3 of these segments. [19]

Duration features

To compute the duration features for the syllables, the authors of [17] examine five different regions within the syllable: the onset, the nucleus, the coda, the combination of onset and nucleus, and the combination of nucleus and coda. The duration of each region is calculated and then normalized using three different techniques. The normalization statistics are determined based on data from speakers in the background model and they are computed using instances of the same sequence of phones appearing in the same syllable position, the same sequence of phones appearing anywhere, and instances of the same triphones anywhere.

The normalization techniques used are: no normalization, division by the distribution mean, Z-score normalization (value mean divided by the standard deviation), and percentile normalization. The illustration of these normalization techniques, along with the regions and measures, is shown in Figure 3.8. This figure, along with Figures 3.9 and 3.10, serves as a visual representation of the features, but not all combinations of regions,

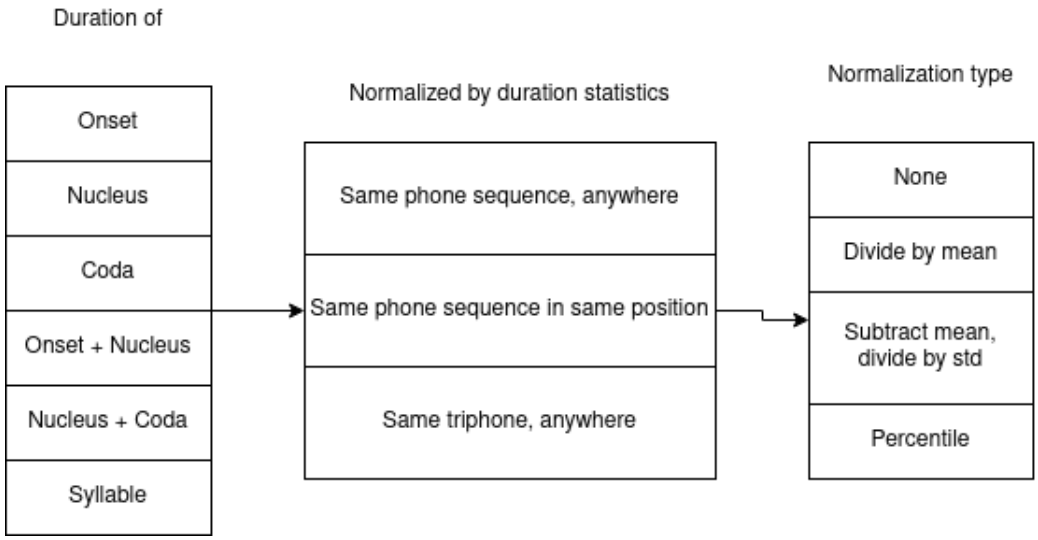
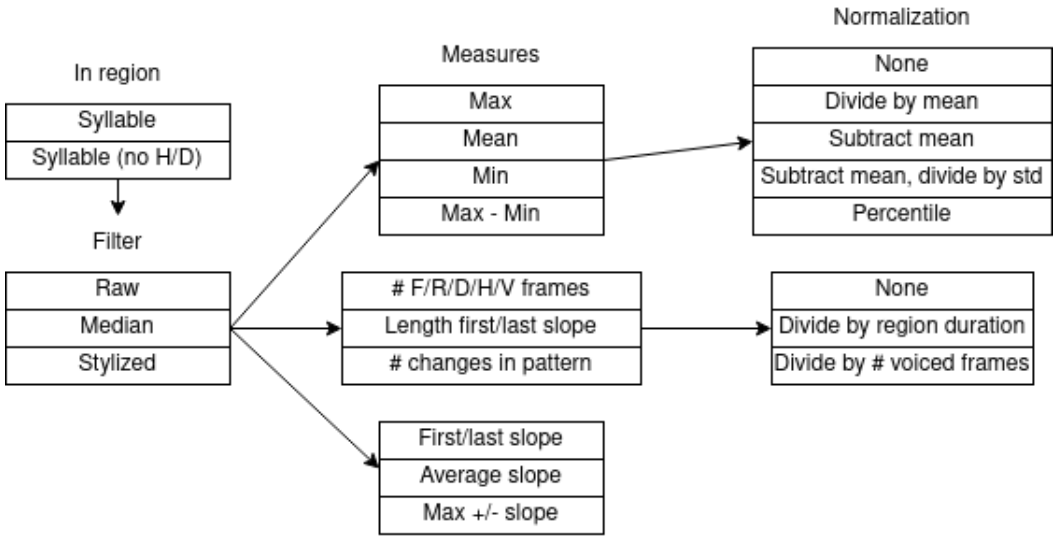


Figure 3.8. Duration features. [19]

measures, and normalization are utilized because some combinations do not make sense or are undefined.

Pitch features

The procedure for calculating pitch features is shown in Figure 3.9.



The symbol "/" indicates different variations of features, while the symbol "#" refers to the number of instances. The letter V denotes a voiced sound, while H/D refers to the estimated pitch being halved or doubled using the LTM model. The letters F/R represent frames that belong to a falling or rising linear spline from the fundamental frequency (F0) fitting. The term "pattern" refers to a series of F/R/H/D frames that have been automatically labeled and combined after merging adjacent frame labels.

Figure 3.9. Pitch features [19]

The process for computing features related to pitch is depicted in Figure 3.9. Two different segments of the syllable are considered - one consisting of only the voiced frames

and the other including all the voiced frames, excluding those marked as halved or doubled by the previously mentioned post-processing step. The pitch values in these segments are then processed in three ways - raw, median-filtered, and stylized using the linear spline method.

For each of these resulting pitch value sequences, several features are computed. These include the maximum and minimum pitch, the range between the maximum and minimum pitch, and the number of frames classified as rising, falling, doubled, halved, or voiced. Additionally, the length of the first and last slopes, the number of changes from falling to rising, the first, last, and average slope, and the maximum positive and negative slope are calculated.

To normalize these features, five different methods are applied to data from the entire conversation, including no normalization, division by the mean, subtraction of the mean, Z-score normalization, and percentile value. Features that are based on the number of frames are normalized by considering the total duration of the region or the duration of the region that only contains voiced frames.

Energy features

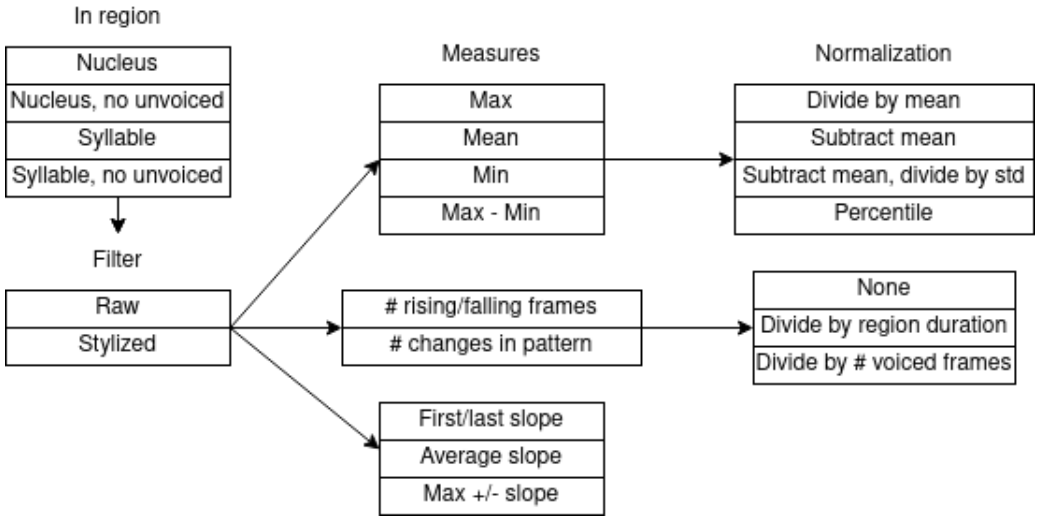


Figure 3.10. Energy features [19]

The energy features were calculated using four separate areas, which are the nucleus, the nucleus excluding any unvoiced frames, the entire syllable, and the complete syllable excluding unvoiced frames. These values were then used to determine features in a way that is comparable to the approach used for computing pitch features, as shown in Figure 3.10. It's important to note that, unlike the calculation of pitch features, raw energy magnitudes were not included as features because they typically reflect the characteristics of the recording channel rather than the speaker's voice.

Syllable-level feature discretization

In order to use the count-based features in the SVM modeling process, it is crucial to categorize the duration, pitch, and energy features that were previously discussed. Figure 3.10 provides an illustration of this process. In order to place these features into specific

categories, it is necessary to discretize them. Given that the ideal placement of these categories is not known beforehand, a small number of total bin counts are tried (2, 3, 5, 10, 30, 60), resulting in various binned versions of each feature. The goal is to create categories with roughly equal amounts of data by evenly discretizing the data based on the distribution of its values.

For features with a small number of different values, a smaller number of categories is used to avoid empty categories. If a particular value is more frequent than the others, it is allowed to have a larger mass and is not split across categories. Missing values, such as pitch features during syllables without any detected voicing, are assigned to a separate category.

Rank-normalization of normalized counts

In [17], to make sure that the N-gram frequencies are consistent across different dimensions and to achieve comparable ranges of values for all features, the authors employ a slightly modified form of a process referred to as rank normalization. The authors keep track of the distribution of feature values for each feature in the background data. During testing, a feature value is substituted with its rank, meaning the number of background data instances that have a value lower than the given feature value in that particular dimension. This rank is then divided by the total number of background instances, which is 1128 in their paper, representing the number of speakers in the background data. This produces a normalized value in the range from 0 to 1. Any zero values, which correspond to instances in which a speaker has no occurrences of a specific prosodic feature sequence, are still mapped to zero, preserving the sparsity of the feature vectors, which is crucial for processing high-dimensional feature vectors efficiently.

To understand rank normalization in a more intuitive way, the difference between any two normalized feature values can be thought of as the percentage of background speakers who fall between these two values. This means that feature value differences are magnified in areas of high population density and reduced in regions of low density, as long as the test samples conform to the distribution of background values. The result of this process is a uniform normalized distribution.

Chapter 4

Neural networks

4.1 Neural networks

A neural network is a type of machine learning model inspired by the structure and function of the human brain, designed to recognize complex patterns and relationships in data. It consists of layers of interconnected nodes (artificial neurons) that process and transmit information, with each layer responsible for extracting increasingly higher-level features from the input data. Neural networks are trained using an optimization algorithm to adjust the strength of connections between neurons, allowing the network to learn how to map inputs to outputs. Once trained, a neural network can be used for a variety of tasks, including classification, regression, and generation of new data. In this work, it will only be used for classification, more specifically for supervised training.

4.1.1 Perceptron

The history of neural networks dates back to the 1940s, when Warren McCulloch and Walter Pitts proposed the concept of artificial neurons as a mathematical model for the computation of certain binary functions[3]. However, it wasn't until the mid-1980s, with the introduction of backpropagation, that neural networks began to gain popularity as a tool for solving complex problems, including speech recognition[22].

Neural networks are inspired by the structure and function of the human brain, consisting of interconnected processing elements known as artificial neurons. These neurons are organized into layers, with the input layer receiving input data and the output layer producing the final result[3].

From a mathematical point of view, each artificial neuron receives inputs, performs a weighted sum of these inputs, and applies a nonlinear activation function to produce its output(Figure 4.1). In this work, the activation function will be either ReLU function (eq. 4.1) or sigmoid function (eq. 4.2)(cf. Chapter 6 of [23]).

$$\varphi(x) = max(0, x) \tag{4.1}$$

$$\varphi(x) = \frac{1}{1 + e^{-x}} \tag{4.2}$$

The resulting output is then passed to the next layer of neurons in the network in the case of the most common neural networks, the feed-forward neural network. The weights of the connections between neurons are adjusted during the training process to minimize the error between the predicted output and its corresponding label[22].

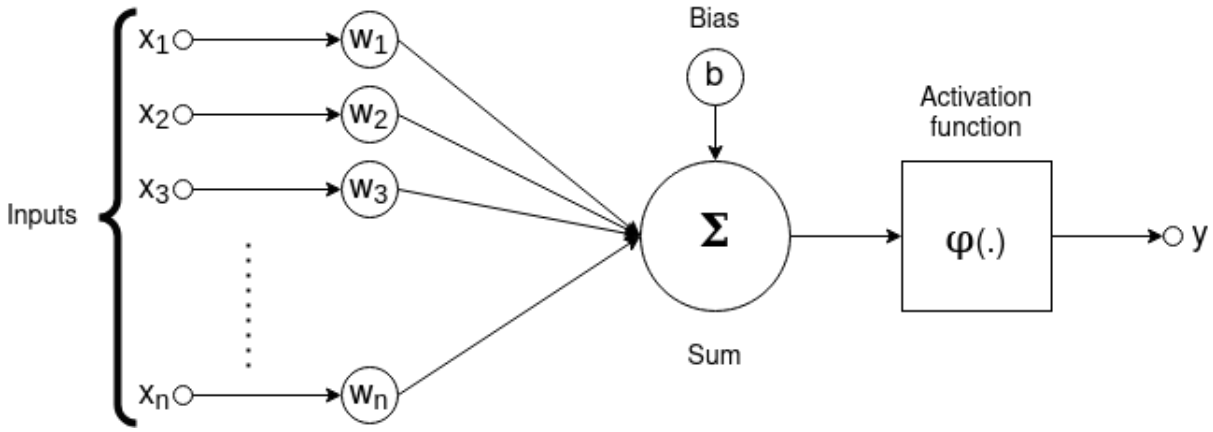


Figure 4.1. Artificial Neuron Model

4.1.2 Feed-forward Neural Network

The feed-forward neural network is the most common type of artificial neural network devised. It is composed of layers of neurons, namely the input layer, one or few(or none) hidden layers and the output layer. Its peculiarity relies on the fact that the information only navigates through from the beginning to the end of the model(Figure 4.2).

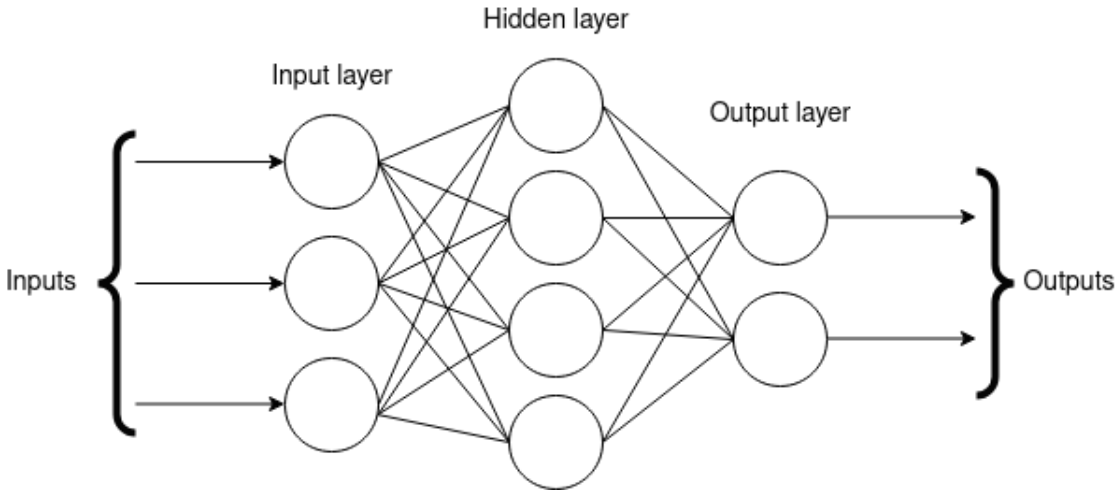


Figure 4.2. Feed-forward Neural Network

4.1.3 Training strategies

Supervised learning

Supervised learning involves learning a mapping function between input data and output labels by using labeled training data. The goal of supervised learning is to learn the underlying relationship between input features and output labels, so that the model can accurately predict output labels for new, unseen input data.

In supervised learning, the input data consists of a set of features or attributes that are used to predict a target output variable. The labeled training data contains pairs

of input-output data points, and the goal is to learn a function that can map the input features to the output labels. The performance of the model is evaluated using a test dataset, which contains input features and corresponding output labels that the model has not seen before.

Supervised learning can be used for various applications, such as classification, regression, and sequence prediction. In classification, the goal is to predict a categorical output variable based on a set of input variables, while in regression, the goal is to predict a continuous output variable. Sequence prediction involves predicting a sequence of output variables based on a sequence of input variables, such as in natural language processing or speech recognition.

Unsupervised learning

Unsupervised learning, on the other hand, involves learning patterns in data without explicit guidance or labels. In unsupervised learning, the model is trained on a set of input data without any corresponding output labels, and the goal is to identify underlying patterns or structure in the data.

In unsupervised learning, the input data consists of a set of features or attributes, but the data points are not labeled with any corresponding output labels.

Unsupervised learning can be used for various applications, such as data compression, anomaly detection, and exploratory data analysis. In data compression, the goal is to reduce the dimensionality of the input data by identifying underlying patterns, while in anomaly detection, the goal is to identify data points that deviate from the expected patterns in the data.

Transfer learning

Transfer learning is a machine learning technique where a pre-trained model is used as a starting point for training a new model on a related task. The pre-trained model has already learned to extract useful features from a large dataset, and this knowledge can be transferred to a new, related task. The goal of transfer learning is to improve the performance of the new model by using the pre-trained model's knowledge as a starting point.

In transfer learning, the input data consists of the data from the new task, and the output is the predicted output variable. The pre-trained model is used to extract features from the input data, and these features are then used as input to a new model that is trained on the new task. The performance of the model is evaluated using a metric specific to the task, such as accuracy or precision.

Transfer learning can be used for various applications, such as image classification, natural language processing, and speech recognition. In image classification, the goal is to classify images into categories such as animals, buildings, or vehicles. In natural language processing, the goal is to process and understand human language, such as sentiment analysis or machine translation. In our peculiar case of speech recognition, the goal is to transcribe spoken words into text.

4.1.4 Training

For this work, I focused on supervised training over a closed set of languages. It means that for each input there is an expected output, a ground truth(also called a target).

Closed-set refers to a situation where the model is restricted to only predicting labels that belong to a predefined set of labels.

The loss function is a mathematical function that measures the difference between the predicted output of the network and the target for a given input. The loss function is used to evaluate how well the network is performing on a given task, and it provides feedback on how to adjust the network's weights and biases to improve its accuracy.

The training process is based on the optimizer, which iteratively adjusts the weights to minimize the error provided by the loss function. Backpropagation is used to calculate the gradient of the error with respect to the weights, allowing the algorithm to determine which weights need to be adjusted and by how much. This process continues until the error is sufficiently small, at which point the network is considered to be converged to an optimum[24].

Optimizer

In machine learning, an optimizer is an algorithm that adjusts the parameters of a model in order to minimize the difference between its predicted output and the actual output of the training data.

The optimizer is responsible for finding the optimal set of parameters that minimize the loss function. This is usually done by iteratively adjusting the parameters in the direction of steepest descent of the loss function. The most common optimization algorithm used in machine learning is gradient descent, which works by computing the gradient of the loss function with respect to the parameters of the model and adjusting the parameters in the direction of the negative gradient. The learning rate is a hyperparameter that determines the step size of the gradient descent algorithm.

There are many variations of gradient descent, such as batch gradient descent, which computes the gradient using the entire training dataset at once, and stochastic gradient descent (SGD), which computes the gradient using a single example at a time. Mini-batch gradient descent is another popular variation that computes the gradient using a small batch of examples at a time.

In addition to gradient descent, there are several other optimization algorithms used in machine learning, such as AdaGrad, RMSprop, and Adam. These algorithms use more sophisticated techniques to adapt the learning rate based on the history of gradients, resulting in faster convergence and improved performance.

The choice of optimizer can have a significant impact on the performance of a model, and it is often necessary to experiment with different optimizers to find the one that works best for a given task. Some optimizers may work better for certain types of models or datasets, while others may be more robust to noisy or sparse data.

4.2 Deep Neural Network architectures

For this work, I will focus on the following architectures to build my models. These models can be used for tasks such as image classification, speech recognition, natural language processing, and decision making. I chose to keep only the most known choice for language identification which is ResNet architecture and to compare it to a promising architecture ECAPA-TDNN which derives from the TDNN architecture.

4.2.1 Deep Neural Network

The fundamental difference between a Feed-forward Neural Network and a Deep Neural Network is the number of layers. Deep Neural Network has more hidden layers, thus it leads to superior performance(Figure 4.3). Deep neural networks have been shown to have the ability to learn hierarchical representations, where higher levels of the hierarchy are composed of lower levels, and can learn complex and abstract concepts.

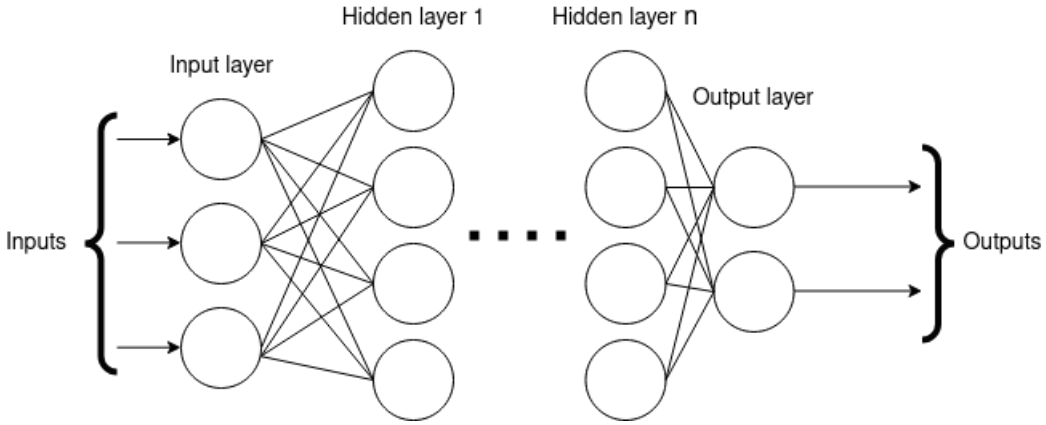


Figure 4.3. Deep Neural Network

Deep learning methods have been used to achieve state-of-the-art performance in a wide range of tasks such as image recognition, natural language processing, speech recognition, and video analysis. The success of deep learning is largely due to the availability of large amounts of labeled data and powerful hardware such as GPUs, which makes it possible to train large and complex neural network models.

Batch normalization

Batch normalization[25] is a technique used in deep neural networks to improve the performance and stability of the model during training. It works by normalizing the input to each neuron in a layer based on the statistics of the inputs in a batch of data.

During training, the distribution of inputs to each layer of a deep neural network can change as the weights are updated. This can lead to the problem of covariate shift where the distribution of inputs to a layer changes over time, making it difficult for the model to learn. Batch normalization helps to address this problem by normalizing the inputs to each layer so that they have zero mean and unit variance, which makes the network more robust to changes in the input distribution.

In addition to improving the stability of the model during training, batch normalization can also improve the generalization performance of the model, leading to better performance on new, unseen data. It has become a standard technique in deep learning, and is used in a wide variety of applications.

Embeddings

I-vectors An i-vector is a fixed-length vector representation of an utterance that captures both speaker and channel characteristics[26]. It is obtained by passing the Mel frequency cepstral coefficients (MFCCs) extracted from an utterance through a factor

analysis model. The resulting low-dimensional vector representation is referred to as the i-vector.

The i-vector can be used to model speaker and channel variability in automatic speech recognition systems. Specifically, it is commonly used as an input feature to speaker verification systems, where the goal is to determine whether a speaker's voice matches a previously enrolled speaker model.

X-vectors Similar to i-vectors, x-vectors[27] are typically obtained by passing a sequence of frame-level features, such as MFCCs or filterbank energies, through a deep neural network (DNN) trained to classify speaker identities. The output of the DNN is a fixed-dimensional vector representation of the input sequence, which is referred to as the x-vector.

Compared to i-vectors, x-vectors have been shown to be more effective in capturing long-term speaker characteristics, and are therefore better suited for speaker verification tasks. Additionally, x-vectors can be trained end-to-end using large amounts of labeled speaker data, which allows them to leverage the power of deep learning techniques for feature extraction and modeling.

4.2.2 Time-delay neural networks

The work [28] discusses the limitations of most neural network architectures in dealing with the dynamic nature of speech, which can lead to mixed performance results. One key challenge is the need for a network to represent temporal relationships between acoustic events, while also providing for invariance under translation in time. For example, the specific movement of a formant in time is an important cue for determining the identity of a voiced stop, but it is irrelevant whether the same events occur slightly sooner or later in the course of time. Without translation invariance, a neural net requires precise segmentation to align the input pattern properly, which is not always possible in practice. As a result, learned features can get blurred and performance can deteriorate. Shift invariance is a critically important property for connectionist systems, and a number of promising models, including time delay neural networks, have been proposed for speech and other domains to address this limitation.

The basic neuron(or perceptron) from Section 4.1.1 is replaced by the one in Figure 4.4. In the modified version of the perceptron, delays (D1 through Dn) are introduced, which affect the inputs to the neuron. This means that instead of simply multiplying the input by a weight, the input is first delayed by a certain amount of time (D1 through Dn) and then multiplied by a weight specific to that delay. There is also a weight for the undelayed input.

The result of this modification is that the perceptron is able to process input data that varies over time. By introducing delays, the perceptron can take into account the temporal structure of the input data, allowing it to better classify or predict future data points.

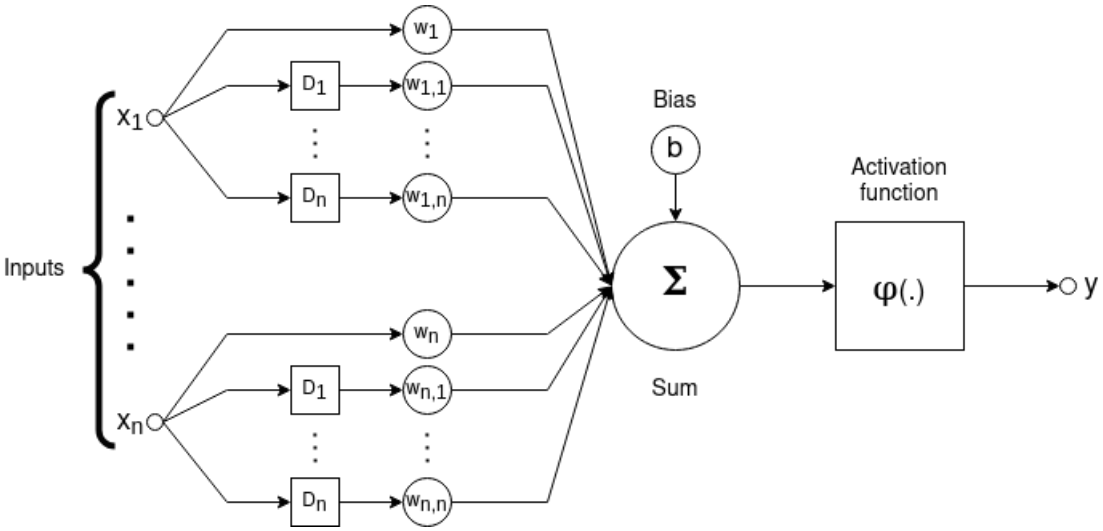


Figure 4.4. TDNN perceptron[28]

4.2.3 Residual Neural Network

A Residual Neural Network (ResNet) is a type of deep neural network architecture that is designed to address the problem of vanishing gradients in very deep networks[29].

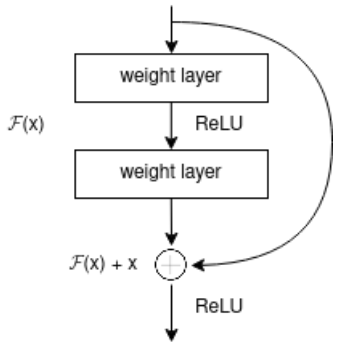


Figure 4.5. Single ResBlock [29]

The main idea behind the ResNet architecture is to gather residual blocks(Figure 4.5) and use residual connections (also known as skip connections) between layers to enable the network to learn the residual function instead of the full mapping.

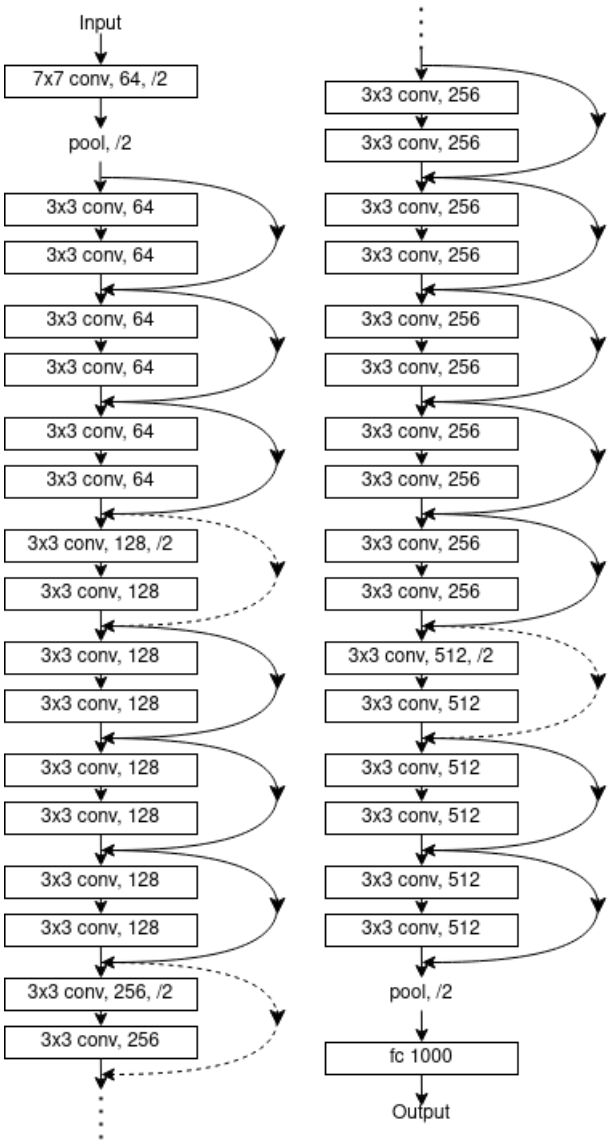


Figure 4.6. The example of Resnet34 [29]

To understand how a ResNet block works, let's first consider a basic neural network without any skip connections. In such a network, the input passes through a series of layers, with each layer applying a non-linear function to the output of the previous layer. The goal of the network is to learn a mapping from the input to the output by adjusting the weights of the layers during training.

However, as the number of layers in the network increases, it becomes more difficult for the network to learn the desired mapping. This is due to the vanishing gradient problem, which occurs when the gradient of the loss function with respect to the weights becomes very small, making it difficult to update the weights using gradient descent.

The ResNet block solves this problem by adding a skip connection, which allows the input to bypass one or more layers and be added to the output of a later layer. This creates a "shortcut" for the gradient to flow through the network, making it easier for the network to learn the desired mapping.

More specifically, a ResNet block consists of two paths: a "main" path and a "shortcut" path. The main path is a sequence of convolutional layers followed by batch normalization

and a non-linear activation function. The shortcut path in most cases is simply the identity function, which passes the input directly to the output of the block(Figure 4.6).

The output of the main path and the shortcut path are then added together element-wise, creating the final output of the ResNet block. This output is then passed on to the next ResNet block or output layer.

The addition of the shortcut connection allows the network to learn residuals, which are the difference between the input and the output of the block. By learning residuals, the network can focus on the "difficult" part of the mapping, which is the part that cannot be easily learned by a few layers, and can better optimize the weight updates during training.

4.2.4 Specific case of ECAPA-TDNN

Neural networks are a powerful tool for processing audio data and have been used to achieve great performance on a variety of speech processing tasks, such as speech recognition, speaker verification, and language identification. Emphasized Channel Attention, Propagation, and Aggregation in TDNN (ECAPA-TDNN) is a known deep neural network that has achieved good results in speaker verification[1].

In this section, I will review the specific issues when working with audio data. I will also describe the structure of the ECAPA-TDNN network and how it is a state-of-the-art technology to deal with these issues[1].

ECAPA-TDNN architecture

The following sections contain the architecture of Emphasized Channel Attention, Propagation and Aggregation in TDNN (ECAPA-TDNN) as described in [1]. The log-mel features provide with multi-channel features where each channel is a stream of audio data and corresponds to a different frequency band.

Channel and context-dependent statistics pooling The article describes the implementation of a channel-dependent self-attention mechanism in an x-vector architecture for speaker recognition. The self-attention mechanism is based on the soft self-attention method described in reference [30] and has been adapted to be channel-dependent.

$$e_{t,c} = v_c^T f(Wh_t + b) + k_c \tag{4.3}$$

where c is the index of the current channel. The activations of the last frame layer (h_t) are transformed into a smaller R-dimensional representation through a linear layer (W and b) followed by a non-linearity ($f()$). This representation is then transformed into a channel-dependent self-attention score ($e_{t,c}$) through a linear layer (v_c and k_c) and is normalized over all frames by applying the softmax function channel-wise across time ($\alpha_{t,c}$).

$$\alpha_{t,c} = \frac{\exp(e_{t,c})}{\sum_{\tau}^{T} \exp(e_{\tau,c})} \tag{4.4}$$

The self-attention score is then used to calculate the weighted mean ($\tilde{\mu}_c$) and weighted standard deviation ($\tilde{\sigma}_c$) of each channel.

$$\tilde{\mu}_c = \sum_t^T \alpha_{t,c} h_{t,c} \tag{4.5}$$

$$\tilde{\sigma}_c = \sqrt{\sum_t^T \alpha_{t,c} h_{t,c}^2 - \tilde{\mu}_c^2} \tag{4.6}$$

The final output of the pooling layer is given by concatenating the vectors of the weighted mean and weighted standard deviation. The article also mentions expanding the temporal context of the pooling layer by allowing the self-attention to look at global properties of the utterance by concatenating the local input with the global non-weighted mean and standard deviation of *ht* across the time domain.

1-Dimensional Squeeze-Excitation Res2Blocks The article discusses the integration of 1-dimensional Squeeze-Excitation (SE) blocks in the x-vector system for speaker recognition. The original x-vector system[27] has a limited temporal context of 15 frames and the authors argue that a wider temporal context could improve performance. The SE-block is introduced to rescale the frame-level features based on the global properties of the recording. The SE-block consists of a squeeze operation, which generates a descriptor for each channel by calculating the mean vector(eq. 4.7) across the time domain, and an excitation operation(eq. 4.8), which calculates a weight for each channel using the descriptors.

$$z = \frac{1}{T} \sum_t^T h_t \tag{4.7}$$

$$s = \sigma(W_2 f(W_1 z + b_1) + b_2) \tag{4.8}$$

where $\sigma(.)$ is the sigmoid function, f$(\cdot)$ a non-linearity, $W_1 \in \mathbb{R}^{R \times C}$ and $W_2 \in \mathbb{R}^{C \times R}$.

The resulting vector is used to multiply the original input channel-wise(eq. 4.9).

$$\tilde{h_c} = s_c h_c \tag{4.9}$$

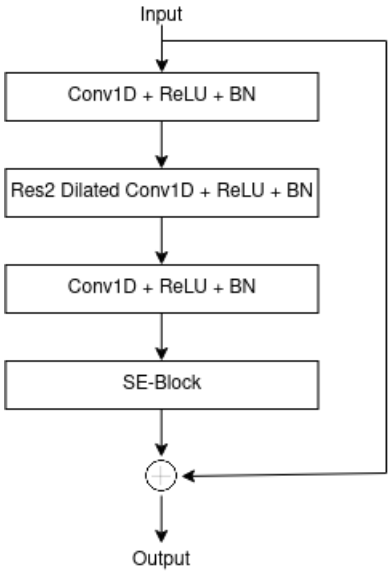


Figure 4.7. The ECAPA-TDNN architecture features an SE-Res2Block shown in this Figure. The Conv1D layers within this block have a kernel size of 1. The central Res2Net [31] Conv1D, which has a scale dimension s = 8, expands the temporal context by using a kernel size k and dilation spacing d. This helps to improve the architecture's ability to process and analyze data over time. [1]

The SE-Res2Block is shown in Figure 4.7, which is a combination of dilated convolutions, dense layers, and the SE-block with a skip connection. The authors mention that the integration of recent advancements in computer vision architecture, such as the Res2Net module, can further improve performance while reducing the number of model parameters.

Multi-layer feature aggregation and summation The original x-vector system only uses the feature map of the last frame-layer to calculate the pooled statistics. The proposed system takes into account the contribution of shallow feature maps by concatenating the output feature maps of all the SE-Res2Blocks, performing a Multi-layer Feature Aggregation (MFA) and processing the concatenated information with a dense layer to generate the features for the attentive statistics pooling. Another way to exploit multi-layer information is to use the output of all preceding SE-Res2Blocks and initial convolutional layer as input for each frame layer block. This is implemented by defining the residual connection in each SE-Res2Block as the sum of the outputs of all previous blocks. The final architecture without the summed residual connections is depicted in Figure 4.8.

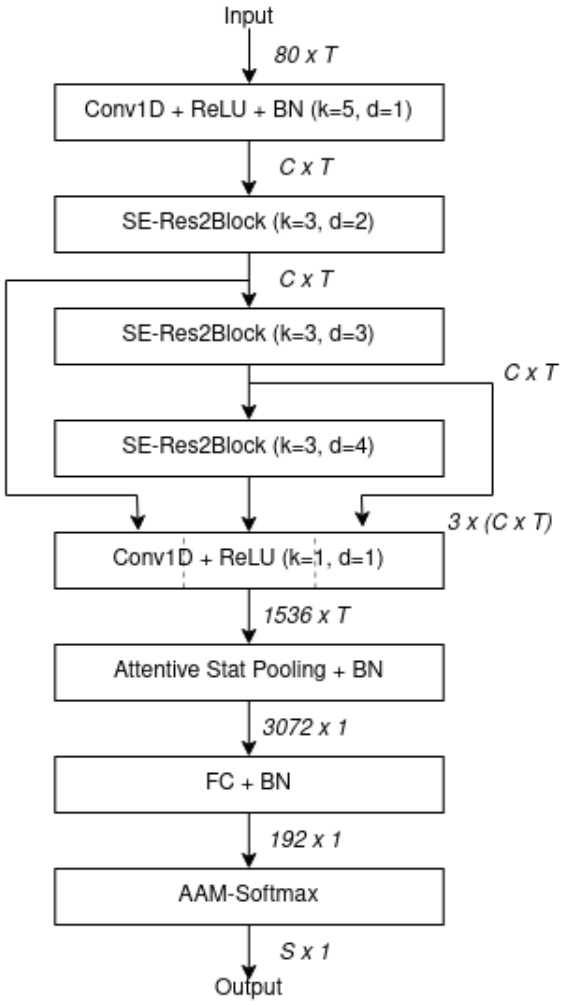


Figure 4.8. The network topology of the ECAPA-TDNN. It is characterized by the use of Conv1D layers and SE-Res2Blocks, with kernel size denoted by k and dilation spacing denoted by d. The intermediate feature-maps have channel and temporal dimensions represented by C and T respectively. Additionally, the number of training speakers is denoted by S. [1]

Issues related to audio data

The ECAPA-TDNN network is a neural network that is designed to address some of the issues that arise when working with audio data. The network is based on a time-delay neural network (TDNN) and it uses an emphasized channel attention mechanism to extract relevant information from the input audio, and a propagation and aggregation mechanism to enhance the speaker-specific information. The channel attention mechanism assigns different weights to the channels based on their relevance, and then combines them to form a single representation of the input audio. The propagation and aggregation mechanism propagates the speaker-specific information through the network and aggregates it to form a single representation of the speaker.

In the following subsection I treat in detail the issues that, I think, may appear and how I think they may be fixed by the structure of ECAPA-TDNN.

Variable duration of the audio recordings When working with audio data, one of the main issues is the variable duration of the audio recordings. Audio recordings can have different lengths. Moreover, it can be difficult to train neural networks that can handle input of varying lengths. To address this issue, neural networks can be designed with a summarization layer that can condense the audio data into a fixed-length representation. This summarization layer can be based on techniques such as pooling or self-attention that can aggregate the information from the audio data in a way that is robust to the variable duration of the audio recordings. ECAPA-TDNN successfully address this issue through both pooling and self-attention.

Temporal context Another issue when working with audio data is the need to account for context. Audio data is a time-series data and the meaning of the audio can change depending on the context in which it is spoken. Neural networks can be designed with a temporal context layer that can model the dependencies between adjacent audio frames. In the case of ECAPA-TDNN, the introduction of 1-dimensional Squeeze-Excitation blocks should improve the TDNN solution.

Variability in speaker characteristics Audio samples can be spoken by different speakers with different characteristics, such as accent, dialect, and speaking style. This variability can make it difficult for a neural network to generalize to different speakers. The ECAPA network may solve this issue by using the propagation and aggregation mechanism, which can capture the speaker-specific information and enhance it. This can allow the network to focus on the speaker-specific information and ignore the variability in speaker characteristics.

Chapter 5

Experiment

5.1 Experimental setup

5.1.1 Dataset

To train the model I used the VoxLingua107 dataset[2].

The VoxLingua107 dataset includes speech data for 107 different languages, with a total training set size of 6628 hours. On average, each language in the dataset has 62 hours of training data. However, the amount of training data for each language varies widely. The dataset also includes a separate development set of 1609 speech segments from 33 languages, which have been verified by at least two volunteers to ensure that they truly represent the specified language.

5.1.2 Training

Data extraction

I accessed the dataset[2] with the class WebDataset from the library webdataset[32]. It is a Pytorch's[33] IterableDataset providing efficient access to datasets stored in POSIX tar archives(each tar archive will be called a "shard"). However since the dataset is substantial, it is impossible to work on it while it is hosted online or it generates errors. Hence, I downloaded all the shards of the training and the development sets. Then after loading the webdataset, it can be shuffled and decoded with wds.torchaudio.

In order to produce an iterable from a webdataset, the class DataLoader from torch.utils.data was used. I implemented a custom *collate* function to sort the data format as wanted. A *collate* function is function that is given as a parameter to the DataLoader instance to map the data contained in the shards(tar files) to the data format we would like to use during training.

In the original dataset, audio signals were associated with labels. I mapped these labels with integer from 0 to 106 on the whole training set in order to feed them to the loss function. In this *collate* function, as performed in the speechbrain Voxlingua107 recipe[34], instead of using raw sample length I extracted a random segment of 3 seconds from each sample.

Then the resulting signals are :

1. sampled (sample frequency of 16000)
2. batched

3. the Hamming window is applied

4. the short time Fourier transform is applied

5. a set of filters from the filter bank is applied(n_mel = 40) as in Section Fbank Features 3.1.3.

The result is normalized at batch level :

$$normalized = (features - mean_{batch})/std_{batch} \tag{5.1}$$

where mean is the mean of the entire batch, std the variance of the batch.

Finally, the resulting features can be fed to the model and the Model Training occurs.

Model

ECAPA-TDNN The model is the one described in the ECAPA-TDNN architecture section 4.2.4. Whenever it was not specified in the paper[1] I used as default kernel_size=1, stride=1, padding=0, dilation=1. The parameters used are 512 for the in between channels, 192 for the embedding dimension at the end of the model and before the loss function application. Finally, the input dimension of the network was chosen according to the different kinds of features that I used(40 for Fbank only, 120 for Fbank with first and second derivatives added, ...).

ResNet Only the 2D version has a library on pytorch[33]. Hence, I wrote this model from scratch. It always begins with the first layer Conv1d+BatchNorm1d+ReLU and ends with nn.Linear which is a fully connected layer that produces the scores. In between numerous blocks are added with the structure of the first layer repeated but with different sizes of in between channels(64, 128, 256, 512). After adding the residual connections as described in 4.2.3, it produces a ResNet.

ResNet18 A ResNet18 is a Residual Network that has in total 18 layers :
The first and last layer, 2 blocks with 64 in between channels, 2 blocks with 128 in between channels, 2 blocks with 256 in between channels and 2 blocks with 512 in between channels. As each block contains 2 layers, it makes a total of 18 layers.

ResNet34 A ResNet34 is a Residual Network that has in total 34 layers :
The first and last layer, 3 blocks with 64 in between channels, 4 blocks with 128 in between channels, 6 blocks with 256 in between channels and 3 blocks with 512 in between channels. As each block contains 2 layers, it makes a total of 34 layers.

Loss function and classifier

Softmax function The softmax function is a mathematical function that is often used as an activation function in the output layer of neural networks. The softmax function takes a vector of real-valued numbers as input and transforms it into a vector of probabilities, where each element of the output vector represents the probability that the input belongs to a specific class.

The standard softmax function would compute the output of the network as :

$$o_i = \frac{e^{n_i}}{\sum_{j=1}^{K} e^{n_j}} \tag{5.2}$$

where $[n_1, ..., n_K]$ are the embeddings at the end of the model, K the number of classes and

$$\sum_{i=1}^{K} o_i = 1 \tag{5.3}$$

as the result produced by the softmax function are probabilities.

As specified in the paper of ECAPA-TDNN[1], the final layer of the model is a softmax function called Additive Angular Margin Softmax[35]. Margins refer to the angular distances between the input feature vector and the decision boundaries of the classifier. By combining three margins, the Additive Angular Margin Softmax increases the robustness of the classifier to small variations in the input data and improves its generalization performance.

Additive Angular Margin Softmax function is calculated combining the three margins m_1, m_2 and m_3 as following :

$$L_{AAM} = -\frac{1}{N} \sum_{i=1}^{N} \log \frac{e^{s(\cos(m_1 \theta_{y_i} + m_2) - m_3)}}{e^{s(\cos(m_1 \theta_{y_i} + m_2) - m_3)} + \sum_{j=1, j \neq y_i}^{n} e^{s \cos \theta_j}} \tag{5.4}$$

where N is the batch size, n the class number.

Loss function Then I apply the cross entropy loss and obtain the loss that will be backpropagated :

$$L_{CE} = -\sum_{i=1}^{n} y_i \log p_i \tag{5.5}$$

where y_i is the truth label, p_i is the probability for the i^{th} class calculated by eq. 5.4.

In the case of ResNet18 or ResNet34, the output dimension of the model is selected to be equal to the number of classes(107) and the standard softmax function is applied. The highest probability's index in this final vector determines the predicted label. Then the cross entropy loss is calculated as well and the weights are updated by the optimizer.

Optimizer

The loss calculated in 5.1.2 is backpropagated to update each parameter of the model as described in 4.1.4. To do so, I use a stochastic gradient descent(SGD) optimizer that performs the following parameter update for each training sample $x^{(i)}$:

$$\theta = \theta - \eta \nabla_\theta J(\theta, x^{(i)}, y^{(i)}) \tag{5.6}$$

where θ is the parameter which is being updated according to the stochastic gradient descent algorithm.

As specified in the ECAPA-TDNN document[1], to prevent overfitting I apply a weight decay of 2e-4 on AAM-Softmax and 2e-5 on the model.

Scheduler

I used a scheduler that seemed to perform well with SGD. I used the scheduler ReduceLROnPlateau from the torch.optim library [33].

Model training

For training the model, I used the training dataset of Voxlingua107. This dataset is composed of 508 shards(from 000000 to 000507) that I divided into a train dataset(from 000000 to 000338) and a test dataset(from 000339 to 000507). These datasets will be then loaded as webdatasets and read with DataLoader in the test and train part of the code.

The batch size is set to 128 for training and 32 for testing. The number of epochs for training is 40, which has been extended in some cases to see if it led to an improvement of performance.

The features are computed, normalized and fed to the model. The embeddings produced by the model(ECAPA-TDNN) are fed in the AAM-softmax classifier (m = 0.2, s = 30). The predictions are compared to the target value and nn.CrossEntropyLoss() computes the loss that is backpropagated to update the parameters of the model.

Then the validation loss is computed with the rest of the data that remained untouched and fed to the scheduler that adapts the learning rate value for the next epoch.

In between epochs, the train and test datasets are shuffled and the process starts again in this new epoch.

5.1.3 Experiments

Development dataset

At first, I tried working exclusively with the development dataset of Voxlingua107[2]. I downloaded the files, parsed the folder and wrote some metadata containing : the name of the file, the path of the file, the label of the file and the file duration(which can be obtained by $\frac{number\_of\_frames}{frame\_rate}$). Then for each label I calculated the total duration of speech available in the dataset. I wanted to produce a train dataset that contained around two thirds of the global duration of each label and a test dataset that would be composed of the rest. I multiplied the dictionary of total duration per speech by two thirds and took the floor value. Then I parsed the whole dataset once again and associated a sample to train dataset if the sum of the previous samples' duration of the same label was below the threshold contained in the dictionary else I associated it to the test dataset.

I started training my networks on the resulting dataset. However, this dataset contained in total 1609 speech segments and it was not enough to produce decent results.

In the end, I wanted to use the development dataset to evaluate the performance of the training of my model on the training dataset of Vowlingua107. To do so I had to map the 33 labels used in the development dataset on the same value that I mapped the corresponding keys(labels) of the training dataset.

Features

For each sample, a feature extraction is performed. I tried doing 4 cases on the ECAPA-TDNN model :

- Fbank :

  - Fbank features
  - no first and second derivatives added
  - Number of Mel filters = 40
  - Length (in ms) of the sliding window used to compute the STFT = 25
  - filter shape = triangular
  - left frames = 5
  - right frames = 5

- Fbank derivatives :

  - Fbank features
  - first and second derivatives added
  - Number of Mel filters = 40
  - Length (in ms) of the sliding window used to compute the STFT = 25
  - filter shape = triangular
  - left frames = 5
  - right frames = 5

- MFCC :

  - MFCC features
  - no first and second derivatives added
  - Number of Mel filters = 40
  - Length (in ms) of the sliding window used to compute the STFT = 25
  - filter shape = triangular
  - left frames = 5
  - right frames = 5

- MFCC derivatives :

  - MFCC features
  - first and second derivatives added
  - Number of Mel filters = 40
  - Length (in ms) of the sliding window used to compute the STFT = 25
  - filter shape = triangular
  - left frames = 5
  - right frames = 5

To have a comparison, I trained both ResNet18 and ResNet34 models with Fbank :

- Fbank features

- no first and second derivatives added

- Number of Mel filters = 40
- Length (in ms) of the sliding window used to compute the STFT = 25
- filter shape = triangular
- left frames = 5
- right frames = 5

5.1.4 Cluster

Given the size of the dataset and the complexity of my training, I used the HPC cluster. More specifically, I used one node and 2 GPUs simultaneously to complete every job using nn.DataParallel().

Technical specification	
CPU Model	2x Intel Xeon Scalable Processors Gold 6130 2.10 GHz 16 cores
GPU Node	24x nVidia Tesla V100 SXM2 - 32 GB - 5120 cuda cores
OS	CentOS 7.6 - OpenHPC 1.3.8.1

Table 5.1. Hardware used

Computational resources were provided by hpc@polito, which is a project of Academic Computing within the Department of Control and Computer Engineering at the Politecnico di Torino (http://www.hpc.polito.it)

5.2 Experimental results

For all the following subsections, the training dataset are the shards from 000000 to 000338 of the Voxlingua107 training dataset. If the samples length is higher than 3 seconds, a random 3 seconds segment will be taken from it to represent this sample during the training phase.

5.2.1 Training set

First, some results for the ResNet models, using Fbank features and 40 epochs.

Model	Accuracy
ResNet18(Fbank)	40.1%
ResNet34(Fbank)	42.6%

Table 5.2. Experiments Voxlingua107 training dataset, ResNet models, 40 epochs

These results will be a source of comparison for the ECAPA-TDNN model since I did not know what to expect working on a language recognition problem with 107 possible labels.

Features	Accuracy
Fbank	49.1%
Fbank with derivatives	48.5%
MFCC	47.7%
MFCC with derivatives	48.7%

Table 5.3. Experiments Voxlingua107 training dataset, ECAPA-TDNN model, 40 epochs

As expected, with the same feature extraction (Fbank) and the same number of epochs, the ECAPA-TDNN model outperforms the ResNet18 and the Resnet34 models. I expected the features with added derivatives to perform better than the classic feature extraction since it contained more information.

By curiosity, I took the ECAPA-TDNN models I saved at 40 epochs of training, loaded them and pursued the process with the same settings for 20 more epochs. Here are the results I got :

Features	Accuracy
Fbank	53.9%
Fbank with derivatives	53.3%
MFCC	52.3%

Table 5.4. Experiments Voxlingua107 training dataset, ECAPA-TDNN model, 60 epochs

The increase of accuracy in my results by going 20 epochs further notified me that there is still a lot of room for improvement in my work. Since it takes more than 9 days to perform one training in parallel on 2GPUs, I do not think it is wise to add more epochs. The solution may lie in a better use of the scheduler and optimizer. In fact, my scheduler decreases learning rate when it detects a “plateau” in the loss validation calculations on the test dataset.

5.2.2 Development set

To perform the following experiment, I loaded the Fbank 40epochs ECAPA-TDNN model trained with shards 000000 to 000338 of the Voxlingua107 training dataset. Then I evaluated the model using the whole development dataset from Voxlingua107 and computed the accuracy.

I reproduced this experiment with two other features extractions, here is what I got from it :

Features	Accuracy
Fbank	21.0%
Fbank with derivatives	22.0%
MFCC	20.2%

Table 5.5. Experiments development set

This experiment has been led by the speechbrain Recipe of Voxlingua107[34] with an accuracy of 93.3% for the Fbank features. There are numerous differences between my implementation and what is done on the speechbrain recipe. I will cite some of them :

- They used the standard softmax while I used AAM-softmax as a classifier.
- They used n_mels = 60 while I used n_mels = 40.
- They used noise and time modifying augmentations, which I didn't implement by lack of time.
- They used Adam optimizer while I used SGD optimizer.
- Finally they used a Linear scheduler while I used ReduceLROnPlateau scheduler.

Chapter 6

Conclusions

To conclude, as assumed the fields of speaker recognition and language recognition are closely related. It is then possible to reuse models like ECAPA-TDNN[1] initially built for speaker recognition to solve language recognition problems. It has been done with state-of-the-art performances by speechbrain recipes[34].

I tried to apply what I learnt resulting in lower performances, but showing that on the application of language recognition on the Voxlingua107 training dataset the ECAPA-TDNN model performed better than the ResNet18 and ResNet34 models. Also using the same dataset and always the ECAPA-TDNN model, what was working the best with my settings was the Fbank feature extraction technique with no additional derivatives insertion.

There are already state-of-the-art algorithms working with ECAPA-TDNN as the Voxlingua107 speechbrain recipe[34]. For future work, we can always think of combining existing technologies to obtain more precise results. If I had to improve my work, I would implement Shifted Delta Coefficients, augmentation features and focus on the optimizer and the scheduler to make sure my networks do not need more than 40 epochs to learn.

Bibliography

[1] Jenthe Thienpondt Brecht Desplanques and Kris Demuynck. "ECAPA-TDNN: Emphasized Channel Attention, Propagation and Aggregation in TDNN Based Speaker Verification". In: *INTERSPEECH 2020* (October 25–29, 2020).

[2] Jörgen Valk and Tanel Alumäe. "VoxLingua107: a Dataset for Spoken Language Recognition". In: *Proc. IEEE SLT Workshop*. 2021.

[3] Daniel Jurafsky and James H. Martin. *Speech and Language Processing*. Prentice Hall, Draft of January 7, 2023.

[4] Leena Mary. *Extraction of Prosody for Automatic Speaker, Language, Emotion and Speech Recognition*. Springer, 2019.

[5] Nitin Indurkhya and Fred J. Damerau. *Handbook of Natural Language Processing, Second Edition*. Chapman and Hall/CRC, 2010.

[6] K. Sreenivasa Rao and Dipanjan Nandi. *Language Identification Using Excitation Source Features*. Springer, 2015.

[7] V. Ramu Reddy K. Sreenivasa Rao and Sudhamay Maity. *Language Identification Using Spectral and Prosodic Features*. Springer, 2015.

[8] Wu Chou and Biing-Hwang Juang. *Pattern Recognition in Speech and Language Processing*. CRC PRESS, 2003.

[9] S. Cumani. "Speaker and language recognition techniques". PhD thesis. Politecnico di Torino, 2012.

[10] L. R. Rabiner and B.-H. Juang. *Fundamentals of Speech Recognition*. Englewood Cliffs, NJ: Prentice Hall, 1993.

[11] Haytham M. Fayek. *Speech Processing for Machine Learning: Filter banks, Mel-Frequency Cepstral Coefficients (MFCCs) and What's In-Between*. 2016. URL: https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html.

[12] E. F. Evans T. H. Bullock and Dahlem Konferenzen. "Recognition of complex acoustic signals". In: (1977).

[13] J. Pelecanos and S. Sridharan. "Feature Warping for Robust Speaker Verification". In: *Proceedings of Odyssey* (2001), pp. 213–218.

[14] E. S. M. A. Kohler P. A. Torres-Carrasquillo D. A. Reynolds, R. J. Greene, and J. J. R. Deller. "Approaches to language identification using Gaussian Mixture Models and shifted delta cepstral features". In: *Proceedings of ICSLP 2002* (2002), pp. 89–92.

[15] P. A. Torres-Carrasquillo W. Campbell and D. Reynolds. "Approaches to language identification using Gaussian Mixture Models and shifted delta cepstral features". In: *Proceedings of Odyssey: The Speaker and Language Recognition Workshop* (2004), pp. 41–44.

[16] Kornel Laskowski and Qin Jin. "Modeling Prosody for Speaker Recognition:Why Estimating Pitch May Be a Red Herring". In: *Odyssey 2010: The Speaker and Language Recognition Workshop* (2010).

[17] S. Kajarekar, L. Ferrer, A. Venkataraman, K. Sonmez, E. Shriberg, A. Stolcke, H. Bratt, and R.R. Gadde. "Speaker recognition using prosodic and lexical features". In: *2003 IEEE Workshop on Automatic Speech Recognition and Understanding (IEEE Cat. No.03EX721)*. 2003, pp. 19–24.

[18] L. Heck K. Sonmez E. Shriberg and M. Weintraub. "Modeling Dynamic Prosodic Variation for Speaker Verification". In: *5th International Conference on Spoken Language Processing* (1998), pp. 3189–3192.

[19] E. Shriberg, L. Ferrer, S. Kajarekar, A. Venkataraman, and A. Stolcke. "Modeling prosodic feature sequences for speaker recognition". In: *Speech Communication* 46.3 (2005), pp. 455–472.

[20] W. Fisher. *The tsylb2 Program: Algorithm Description*. NIST, 1995.

[21] Entropic. *ESPS Version 5.0 Programs Manual*. Entropic Research Laboratory, Washington, D.C, 1993.

[22] Dong Yu and Li Deng. *Automatic Speech Recognition: A Deep Learning Approach*. Springer, 2015.

[23] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[24] Nelson Morgan Ben Gold and Dan Ellis. *Speech and Audio Signal Processing: Processing and Perception of Speech and Music*. Wiley, 2011.

[25] Sergey Ioffe and Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. ICML, 2015.

[26] R. Dehak P. Dumouchel N. Dehak P. Kenny and P. Ouellet. "Front-end factor analysis for speaker verification". In: *IEEE Transactions on Audio, Speech, and Language Processing* 19.4 (2011), pp. 788–798.

[27] G. Sell D. Povey D. Snyder D. Garcia-Romero and S. Khudanpur. "X-vectors: Robust DNN embeddings for speaker recognition". In: *Proc. ICASSP* (2018), pp. 5329–5333.

[28] G. E. Hinton K. Shikano A. H. Waibel T. Hanazawa and K. J. Lang. "Phoneme recognition using time-delay neural networks". In: *IEEE Trans. Acoust. Speech Signal Process.* 47.3 (1989), pp. 328–339.

[29] S. Ren K. He X. Zhang and J. Sun. "Deep residual learning for image recognition". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), pp. 770–778.

[30] T. Koshinaka K. Okabe and K. Shinoda. "Attentive statistics pooling for deep speaker embedding". In: *Proc. Interspeech* (2018), pp. 2252–2256.

[31] K. Zhao X. Zhang M.-H. Yang S. Gao M.-M. Cheng and P. H. S. Torr. "Res2Net: A new multi-scale backbone architecture". In: *IEEE TPAMI* (2019).

[32] *Webdataset*. URL: https://github.com/webdataset/webdataset.

[33] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[34] Mirco Ravanelli et al. *SpeechBrain: A General-Purpose Speech Toolkit*. arXiv:2106.04624. 2021. arXiv: 2106.04624 [eess.AS].

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