university of groningen

faculty of science and engineering

A MACHINE LEARNING APPROACH TO AUTOMATIC LANGUAGE IDENTIFICATION OF VOCALS IN MUSIC

HERMAN GROENBROEK

Supervisors:

DR. M.A. WIERING
Artificial Intelligence, University of Groningen

ARRYON TIJSMA
Machine Learning Engineer, Slimmer AI

Master's Thesis Artificial Intelligence
University of Groningen

April 7, 2021

ABSTRACT

Audio classification is an important field within data science. Having an automated system that is able to appropriately and quickly respond to incoming audio can ultimately save lives, for instance in an emergency call centre. An important first step of audio classification is Automatic Language Identification (LID). Unfortunately, LID of vocals in music has not made the same advancements as that of speech. At this time, there exists no publicly accessible system that is able to accurately classify the language that music is sung in, nor a labelled dataset to train one.

In this thesis, a novel music dataset with language labels is described: the 6L5K Music Corpus. A vocal fragment dataset is obtained by taking 3-second audio fragments from the 6L5K Music Corpus classified by a pretrained vocal detector to contain vocals. Two neural network architectures are implemented: a feedforward Deep Neural Network (DNN), which works well for LID on speech data, and VGGish, a powerful architecture for general audio classification tasks. For the input features, mel spectrograms and MFCCs are computed from the vocal fragment data, and the networks are trained and optimized for classifying the sung language.

The results in this thesis indicate that the task of LID of sung music is non-trivial. The DNN with various setups performs better than chance, obtaining at best 35% accuracy with six languages. VGGish shows more promising results on the vocal fragment data, obtaining 41% accuracy on the same six-class dataset. When using these systems on unseen test data however, the DNN drops to 18.1% accuracy, whereas VGGish drops to a more respectable 35.2%. We finally implement an Ensemble by combining the two models, but the results are no better than an average of the two. Although VGGish performs significantly better than the DNN, it is not accurate enough to be used reliably in any modern-day system. These results, combined with the fact that little research is done on LID of sung music, indicate that this subset of audio classification has plenty of potential still for novel research.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my external supervisor at Slimmer AI, Arryon Tijsma, for always helping out, for instance whenever I was stuck with programming or running code on an external machine. His insights into the problems at hand and weekly video calls during the COVID-19 pandemic managed to keep me motivated during these bizarre times. I am very grateful to have had a supervisor who could help solve problems ranging from trivial to complex, often within a day or two. A supervisor who managed to always keep a positive outlook, and whom I could rely on whenever I needed assistance.

I would also like to thank my internal supervisor, Dr. M.A. Wiering, for more broadly guiding this thesis in the right direction, helping to set deadlines, and supplying proper feedback in this scientific field where needed.

I am very grateful to all my friends and family who have directly or indirectly helped me. Specifically, to Rachelle Bouwens for always supporting me; to Henry Maathuis for sharing thoughts and advice on-topic; to Adiëlle Westercappel for sharing her linguistics expertise; to Hidde Ozinga for clearing my mind on walks, and to my family for being patient.

Finally, I would like to thank my colleagues at Slimmer AI who supported me through a tough time. My sincerest thanks to everyone who checked in with me to see if I was doing alright, and for allowing me to continue my research at my own pace.

I dedicate this thesis to my father, who was unable to witness its completion.

CONTENTS

1 INTRODUCTION

1.1 LANGUAGE & MUSIC

1.1.1 *Language*

Language is a tool for communication thought to be exclusive for humans [7]. A language is often expressed in a written form or as speech, but alternative encodings of language exist, such as signing, whistling, and braille. In all cases, a language contains elements of meaning (words), and rules on how to use these elements. A large variety of languages exist across the globe, and languages continue to vary over time, adding elements, adjusting rules, or even being created in the first place. These modifications are often related to the culture in which languages are used. As a common example, it is said that, whereas the English language only has a single word for "snow", the Inuit language has multiple [31]. Regardless of the debate whether this is actually true, one can see how in one culture, commonly referenced objects and ideas evolve into new, more specific language elements. In this thesis, we look at language to the extent of spoken or sung words.

1.1.2 *Music*

Music has been an integral part of human society for as long as we can remember. Music consists of various sounds that are created by tapping, slapping, plucking, or blowing objects manufactured to produce pleasing sounds. For thousands of years now, people have been making musical instruments with the purpose of creating pleasing, harmonic sounds. The earliest known musical instruments – flutes made of bone and ivory – are said to be more than 35,000 years old [9]. Although it is still not perfectly clear why we tend to enjoy various harmonic sounds [21], it is clear that this pleasure is shared across humans all over the world, regardless of cultural background. Besides manufacturing physical instruments, humans have also learned to use their own vocal chords to produce rhythmic and harmonic sounds. Although we tend to enjoy purely instrumental music as well, the majority of modern-day pop music is accompanied by vocals. These vocals are usually expressed as poetry-like texts – lyrics – that are commonly sung, rapped, or otherwise rhythmically spoken, with the texts often intended to either express the artist's feelings, or for listeners to relate to.

1.1.3 *Cultural Differences*

Music has been used as a cultural means for a long time. Various cultures have adopted certain instruments and play styles that yields a sound signature that is distinct for that culture. For instance, a Spanish guitar is more commonly found in Spanish music, and for typical oriental music, the erhu delivers sounds that make it highly recognizable as such. Although globalisation has caused popular international hits from various countries to sound somewhat similar in terms of instrument usage, some musicians are keen on using instruments that go back to the roots of their culture, which can sometimes be found in these international hits as well. A more direct cultural difference in music would be the vocals. Many countries have a language of their own, and regions within these countries tend to have varied dialects. These languages and dialects of individuals all make their way into the music they produce, and as such, a set of music tracks sung in one language may still have a large variation in vocal sounds.

1.1.4 *Language Classification*

In this thesis, the focus is put on classification of language in music. Although language is a broad concept, our focus is specifically on the vocalized lyrics of a song. This means that, given a song, the goal is to find in which language the song was sung. Interestingly, given the differences between cultures, for instance with regard to instrument usage as described in Section 1.1.3, being able to identify the instruments that are used may help in the classification of the vocals. That is, if a song where the erhu can be heard is often sung in Mandarin, then being able to detect the erhu in a track can help suggest that the vocal language is Mandarin as well. However, is is also possible that various instruments uncommon for a culture matching some language are used for an artistic purpose, and as such they may not match the expected language of vocals.

More certainty of vocal language classification can be obtained when only considering the vocals themselves, without any instrumental additions that are technically irrelevant to the vocal language. Spoken and sung languages are distinct in their phonotactic restrictions, meaning that various languages do not allow some phonemes in specific locations of a word, as well as languages featuring certain phonemes more than others. If the phonemes in an audio fragment can be detected accurately, it is possible to reasonably classify the language of this audio fragment as well [35].

1.2 RESEARCH QUESTIONS

The purpose of this research is to discover how automatic language identification (LID) of sung music benefits from training neural network architectures for classification. For this, it is necessary to look deep into the training data, to find which types of data work best for obtaining an accurate neural network classifier. The main research question can thus be defined as:

***Research Question**: How can we train a neural network to perform best on the task of Automatic Language Identification in vocal music?*

As this is quite a broad research question, we pose three questions that – when answered – suggest an answer to the main research question:

***Q1**: What type of **data** is best for training neural networks to be able to classify the language music is sung in?*

***Q2**: Which neural network **architecture** works best for Automatic Language Identification of sung music?*

***Q3**: How do we determine which system performs **better** on the task of Automatic Language Identification?*

These three questions will be answered throughout this thesis, and to conclude, the main research question will be answered in Section 8.2.2.

1.3 SOCIETAL IMPACT

In the current state of technology, companies like Google use automatic language identification (LID) of speech for their smart speakers to allow multilingual use thereof. LID is often applied in real-time to speech with the purpose of better understanding what is said, and acting accordingly. In some cases, for instance in the understanding of emergency calls, having an accurate LID system can make the difference between life or death [34].

LID of music is a related field, but is less often used. Identifying the language that music is sung in can be used for better recognizing which song is being played however. This is already done in real-time on modern smartphones nowadays, being able to show users on their lock screen what the artist and title of the track is that is currently playing in the background, even when the device is in low-power mode [16]. First detecting the language that music is sung in can help subset the potential music matches, thus improving the efficiency in music recognition by having to compare fewer track fingerprints. LID

in music can also be used to automatically add a language label to music videos uploaded to video sharing websites. These websites can then better cater video suggestions to visitors by knowing in which language music in the video is being sung.

Whereas research on Automatic Language Identification of speech has been a well covered topic [2, 8, 17, 25, 27, 29, 32, 34, 35, 53], LID of music is much less researched [6, 33, 41, 49]. This thesis serves as a starting point for the application of deep neural networks on the problem of LID of music. This thesis explains the current state of technology regarding this topic, and applies various neural network approaches in order to see what type of data is best trained on, and which architectures of neural networks function best with these types of data.

1.4 SLIMMER AI

This thesis is written as a result of a Master's Project offered by Slimmer AI, an artificial intelligence company in Groningen, The Netherlands. Slimmer AI, at the time known as Target Holding, were curious to see whether language as a feature of music tracks could help predict which tracks would become popular (a 'hit'), and which would not (a 'flop'). Being able to predict which tracks can become a hit or a flop will not only inform the industry ahead of time how likely it is for tracks to become popular, it would also be a great tool during music creation to make alterations such that the system would be more likely to predict the track becoming a hit.

The hypothesis for using language as a feature to predict hits and flops is that tracks sung in a certain language have better odds of becoming a hit in countries that generally prefer the sung language. For instance, since the Dutch like to visit nearby warm countries such as Spain for summer vacation, tracks sung in Spanish may have higher odds of becoming a hit around the summer. On the other hand, when some country has a very poor relationship with another country, songs sung in that language may be less likely to become popular there.

Part I

THEORETICAL BACKGROUND

2 AUTOMATIC LANGUAGE IDENTIFICATION

2.1 THE LITERATURE

As we have noted in Section 1.3, Automatic Language Identification (LID) has been widely studied for application on speech data, but not so much for music data. Here, we will give an overview of previous research separated by whether it has been done on speech or music data, given the inherent differences between the two. It remains to be seen whether knowledge from existing literature of LID on speech can help design a system that can classify sung languages in music.

2.1.1 *Identifying Speech*

LID of speech has been a topic of interest for quite some time. In 1980, Li et al. researched statistical models for identifying spoken languages [27]. Furthermore, in 1989, Cole et al. [8] already studied the feasibility of using neural networks by classifying patterns using the distribution of phonetic categories. In older work, language identification is often done with telephone recordings. Muthusamy et al. reviewed various LID methods that could be considered state-of-the-art at the time, ranging from acoustic and language modeling to speaker identification [34, 35]. Zissman continued their work by comparing four techniques for LID of telephone speech, containing Gaussian Mixture Models (GMMs) and language modeling methods [53].

More recent work includes that of Brümmer et al. where Joint Factor Analysis (JFA) is used to obtain state-of-the-art results at the time [5]. Shortly thereafter, Martinez et al. (2011) outperformed their JFA methods by using iVectors to represent the relevant speaker data, and building various classifiers on this [32]. This work formed the basis of research done at Google by Lopez, Gonzales, et al. where a novel speech dataset was created ('Google 5M') and was used with Deep Neural Networks (DNNs) to classify the spoken language [17, 29]. Continuing this, Bartz et al. (2017) utilized Deep Convolutional Recurrent Neural Networks (DCRNNs) which were able to handle noise well, and was also easily extendable to new languages [2].

2.1.2 *Identifying Music*

As described, LID of music is a field that is much less researched than LID of speech. As one of the first attempts at researching this

field, Schwenninger et al. (2006) evaluated how well existing state-of-the-art LID systems transfer to sung music [41]. They concluded that an existing system used to distinguish Mandarin and English did not transfer well to sung music in these languages. Tsai et al. (2007) continued their work and included Japanese as a third language [49]. However, their results remained insufficient, arguing that one of the problems was a lack of available data. In 2011, Mehrabani and Hansen evaluated a successful system for LID on singing speech [33]. However, in the same year, Chandrasekhar et al. came up with various new approaches to LID in music [6]. Working for Google, they had access to a large database of music videos, of which 1000 music videos were obtained for each of 25 selected languages. Training a set of Support Vector Machines (SVMs), one for each language, their goal was to see whether visual information of music videos would aid language classification of music. Given the fact that 25 languages are used and thus the guessing probability would be 4%, their results of audio-only features yielding 44.7% accuracy, and audio + video features yielding 47.8% accuracy, it becomes clear that this method of training multiple SVMs is a feasible method of distinguishing vocals to some extent, although there is plenty of room for improvement still.

2.2 AUDIO FEATURES

Digital audio is a representation of sound waves, having been transformed into samples with a specific sample depth. In general, CD-quality audio can be seen as a standard for digital audio, which contains 44,100 samples per second (the sampling rate), with each sample having 16-bit depth. Higher bit depth standards exist, such as DVD and Blu-ray quality, which support up to 24-bit depth. Often, audio is recorded in stereo, featuring two audio tracks per file. The lower the bit-depth, the smaller the file sizes, but this is paired with a lower audio quality.

Monophonic digital audio in itself can be seen as a one-dimensional array of data points. A 3-second audio clip for instance with a sampling rate of 44,100 can be seen as a feature vector of 132,300 dimensions. For pattern recognition, which is at the basis of classification tasks, it is key to reduce the dimensionality of the data while retaining most relevant data. As an example, the last second of a 3-second audio clip may be completely silent; a third of the feature vector would therefore not contain any more information than the single fact that no audio is present. Classification tasks are often solved through mathematical computations on the feature vectors. A higher dimensionality may result in exponentially longer computation times. As such, there is a large variety of methods to reduce the dimensionality of specific

types of data while retaining the important information, also known as *feature extraction* [42].

One of the most important discoveries for feature extraction of audio signals is the Fourier transform, which transforms a time-series into a frequency-series [4]. Simply put, an audio signal can be transformed into the frequencies that it consists of. For feature extraction, the short-time Fourier transform (STFT) is often used, which is a windowed method of computing the Fourier transform per time frame [18]. With the STFT, essentially a frequency summary per timestep can be taken. This can be visualized in a spectrogram, with the x-axis containing the time frames, and on the y-axis the frequency. For feature extraction of speech, the mel scale comes into play: this is a scale with which the frequency scale can be transformed into a subjective magnitude scale, which deals with the fact that the human perception of differences in frequencies is not linear [46]. Given this mel scale, a mel spectrogram of any audio signal can be computed, which results in a spectrogram that is 'corrected' for human perception. For many speech-related classification tasks such as speaker identification or automatic language identification, these mel spectrograms provide a useful representation of the audio data [30, 39]. Moreover, a mel spectrogram is often converted to mel-frequency cepstral coefficients (MFCCs) by taking the discrete cosine transform [11]. This yields a set of coefficients that can be more compact than mel spectrograms, while still representing the power spectrum of an audio signal.

2.3 ARTIFICIAL NEURAL NETWORKS

2.3.1 *Multilayer Perceptron*

For the purpose of classification tasks, countless statistical and self-learning methods exist. In recent years, many different architectures of artificial neural networks (ANNs) have been applied to classification tasks, yielding unprecedented accuracy scores [17, 24, 26, 47]. Although these ANNs largely differ in architecture, they are all based on the concept of a multilayer perceptron.

The first model that we are implementing is a multilayer perceptron (MLP). An MLP is a feedforward neural network, meaning that there is no cycle or loop in the connections between nodes. It is defined as having one input layer with at least one input node, one or more hidden layers each with at least one hidden node, and one output layer with at least one output node. Although an MLP with a single hidden layer is not often referred to as a Deep Neural Network (DNN), one with multiple hidden layers *can* be described as such. For each node in a layer of the network, there is a connection to all

nodes in the following layer. Given input data, an MLP can be trained for either supervised, unsupervised, or reinforcement learning. Here, supervised refers to data which comes with labels, where the system learns to predict these labels. With unsupervised learning, there are no explicit labels to predict, and a network is generally trained to recognize patterns, to cluster, or as an autoencoder. Lastly, with reinforcement learning, the network learns to optimize some reward, which may increase upon taking good actions and decrease upon bad ones. Regardless of the learning type, an MLP is able to learn by updating each layer's set of weights. These weights are what each of the values from the input layer towards the output layer are multiplied with. The key is to update these weights in such a way that the current output given an input sample more closely resembles the label: the values that the output should have been, in the ideal scenario.

With an initially randomly initialized network (i.e. randomly initialized weights), the initial prediction for an input sample is likely to be very different from the label of the sample. From the differences between output prediction and true label, an error can be computed. The function for this is referred to as the *cost* or *loss function*, and is decided on by the neural network architect depending on how severe the differences are deemed (the mean squared error is often used). Given the error between prediction and true label, the goal is to figure out which weights to change how much, in order for the same input to predict an output closer to the true label. For this, Rumelhart et al. (1986) introduced the backpropagation algorithm [40]. With this algorithm, the gradients of the loss function can be computed. Given the gradients, the weights can be adjusted in the right direction with the use of an optimizer. Changing the magnitude of weight adjustments can be done with varying the learning rate, but keep in mind that a learning rate that is too low or too high may not let the network converge at all.

2.3.2 *Convolutional Neural Network*

The second model that we are implementing is VGGish, a Convolutional Neural Network (CNN) [19]. A CNN is based on the concept of an MLP, but with a key difference that allows it to work better on image-like data. Whereas each node in a hidden layer of an MLP receives input from all nodes in the previous layer, a CNN contains kernels that reach only a subset of nodes in the previous layer, also known as the *receptive field*. This takes place in a convolution layer, where kernels with weights slide across the entire input, each learning a specific feature that appears relevant in the input data. After a convolution layer, the dimensionality is often reduced with a pooling layer, downsampling by taking the maximum or the average of multiple

values. After the convolution and/or pooling layers, of which multiple may be used, a CNN often contains fully connected (FC) layers at the end, which are also found in MLPs. The penultimate layer of a CNN, when using FC layers, can be seen as an embedding layer, as this is a vector representation of all relevant information found in the input data for the task that a CNN has been trained for. Although an MLP or DNN can also be used on the same image-like data, generally speaking CNNs learn faster and perform better on image classification tasks, as they are better suited for find relevant patterns in the input data due to the sliding window method.

2.3.3 *Regularization*

Neural networks are not trivial to train. Despite the promise of MLPs being a universal approximator [10, 20], these networks are often very computationally complex, require expensive hardware to train, and a lot of data to be able to generalize well. A supervised neural network only learns to predict correct labels for the types of data that it has seen during training. Using a well-designed neural network architecture does not guarantee success on any given problem. In order to make sure that neural networks are able to generalize to unseen data, which does not appear in the training dataset, there are a number of regularization methods that are often applied.

Validation Data

First off, for training a supervised neural network, a large set of relevant, labelled training data needs to be available. In almost all cases, the training data will not cover all potential input and output (label) combinations. As such, the network will have to learn an approximation to the function mapping the input to the correct output. When only the accuracy on a training dataset is taken into account, running the network on unseen data may drastically lower the test accuracy. It is good practice to split the training dataset into a validation portion, which is never used during training, but will be tested on after each epoch of training. This way, you can see during training whether the network is able to generalize what it learns from the training data, to the unseen (validation) data. The expectation is that the accuracy on the validation data more closely resembles the accuracy on future test data, given that the validation data is also representative of the data the network will be run on in the future.

Dropout

Dropout is a technique that regularizes a neural network by randomly dropping units and their connections between the input layer and the penultimate layer during training with a specified probability p [45].

After training, the units are present but their weights are multiplied by this p. This results in a significant reduction of overfitting. Although this is a popular technique for fully connected layers in a feedforward neural network, the use of dropout in convolution and pooling layers of a CNN is still debated [51].

Batch Normalization

Generalization can also be improved by using batch normalization [22]. Here, normalization is applied per mini-batch. This is a regularization technique that helps train a network faster, allows for stable training with a higher learning rate, and may eliminate the need for dropout in some cases. Similarly to dropout, batch normalization may also not work well with a CNN as it does with a feedforward DNN. It is possible to restructure batch normalization in order to work well with CNN architectures, however [23].

Cyclical Learning Rate

When training a neural network, the learning rate is one of the main hyperparameters that needs to be decided on, which can drastically influence the course of the training session. With a default learning rate, the idea is to decrease the learning rate as the training progresses, in order to slowly converge to an optimal set of network weights. With an initial learning rate that is too low or too high for the machine learning problem at hand, the network may converge very slowly, or it may not converge at all. Smith (2017) came up with a cyclical learning rate, where the learning rate varies between epochs of the training process, meaning that the learning rate in the short term varies within reasonable bounds, while as a whole, the learning rate is still ever decreasing [44]. This is shown to help a network converge faster, and to allow it to better move beyond local optima. In their paper, Smith argues that this reduces how much experimental fine-tuning the learning rate needs in order to obtain sufficient results.

3 DATASETS

As we saw in Section 2.3, neural networks act as a function approximator. Let us look at Automatic Language Identification (LID) as a problem of finding a function that takes as input a music track, and outputs the vocal language that the track contains. If we wish to train a model using data with the purpose of approximating the function of LID, the dataset that is trained on needs to be representative of the music that the model will be used for in the future. As such, given the fact that an LID-trained model needs to function well for all possible variations of vocal music, it is necessary for the dataset to represent as many genres and variations in music as possible. Otherwise, the model may not be able to generalize well, and even a state-of-the-art model will not be able to perform well in this case.

As we now understand the importance of finding a dataset representative of the complete set of possible vocal music, let us dive into exactly how the dataset may look. At the time of writing, there exists no publicly available music dataset containing sufficient language labels. As such, we are left with a few options. For starters, there exist multiple speech datasets that are labelled with the language that is spoken, for the purpose of LID of speech. It may be possible to augment this speech data to make it resemble music, for instance changing the pitch and adding background instrumentals. Whether this is an accurate representation of actual music remains to be seen. A second option is to look at music datasets and add a language label indirectly. That is, by means of song lyrics and other metadata. However, this may be prone to labelling errors depending on the quality of the metadata. Alternatively, relevant metadata may be largely missing. As a third option, one could exclusively look at correctly labelled music data, and make do with the amount of data that can be directly found. In this chapter we will describe which method works best.

3.1 SPEECH DATA

TopCoder Biblical texts

TopCoder Inc. have released a speech dataset containing 66,176 .mp3 audio files spoken in one of 176 possible languages. The files contain audio from spoken Biblical texts [48]. Unfortunately, within the 176 languages, we do not find the more common Western languages such as English, French, Spanish or Italian. Instead, the languages are rather uncommon, including languages such as Ojibwa Northwest-

ern, Quechua Margos-Yarowilca-Lauricocha, and Nahuatl Highland Puebla. Although the dataset is interestingly large, given the rarity of the languages that are contained, this dataset will not be very useful for automatic language identification in music.

CSS10: Collection of 10 Single Speakers

Since there aren't many publicly available multi-language speech datasets, Park and Mulc have published a freely available 10-language dataset containing speech from LibriVox audiobooks produced by a single speaker per language, with the purpose of applying machine learning techniques to for instance text-to-speech [37]. The languages include Dutch, Finnish, German, Russian, and Spanish. Since the dataset contains a large amount of speech data for various common languages, this could be used for language identification. However, as every language is spoken by only a single speaker, it is easy to forget that a model trained for language identification on this dataset may act more like a speaker identifier, as this is generally an easier function to approximate in the context of machine learning.

Google 5M LID

Lopez-Moreno et al. and Gonzalez-Dominguez et al. have successfully applied deep neural networks to the problem of automatic language identification (LID) of speech. For this, they created a dataset they refer to as the *Google 5M LID Corpus* [17, 29]. Unfortunately, they have not made this dataset publicly accessible. According to Gonzalez-Dominguez et al., the dataset contains *"[...] anonymized queries from several Google speech recognition services such as Voice Search or the Speech Android API"*. The dataset is worth mentioning given the sheer size and the fact that LID has been successfully applied using this dataset. However, it will not be of further use for this thesis, given its private nature.

3.2 MUSIC DATA

3.2.1 *FMA: Free Music Archive*

The Free Music Archive (FMA) is a publicly available dataset containing 106,574 music tracks with metadata [12]. Just over 15,000 tracks contain the label of which language the track is sung in. Unfortunately, 14,819 tracks contain the language label 'English'. The second most occurring language label in the FMA dataset is Spanish, which occurs for only 205 tracks. For the purpose of language identification, this music dataset is simply not varied enough.

3.2.2 *Google Audio Set*

Researchers at Google have made a huge human-labelled audio dataset available to the public [14]. This dataset contains 1,011,305 YouTube videos labelled to contain music. Although the quantity and quality is high, it does not contain any language labels directly. Since the dataset refers to YouTube videos, it is possible to look into the video description language and the video uploader's country of residence for a potential language label. On the other hand, this would require large assumptions to be made, such as the fact that music language would always match the video description language, which is sometimes, but far from always, the case. As such, this dataset will not be of much use to our final LID dataset.

3.2.3 *Slimmer AI: HitFinder Charts*

For a few years now, Slimmer AI's HitFinder system has been tracking which songs are popular on Spotify in which country. Using this dataset of popular tracks for various countries, it is possible to create a dataset of labelled music tracks. For a subset of languages that was manually picked (Dutch, English, French, German, Portuguese, and Spanish), we know which country the track was popular in, and we have the track's artist and title. Given the following two assumptions, we can turn this into a language-labelled dataset:

(1) The language of a song title matches the language that the song is sung in.

(2) A song sung in some language is more likely to be popular in the country where this language is spoken natively, than in another, with the exception of world languages such as English.

There are methods to classify the language of written text. Given such a method, for instance `langdetect`[1] in Python, it is possible to classify the language of a track under assumption (1). Let us take the Dutch song: "Ronnie Flex - In Mijn Bed". Given the fact that this track was found in the charts for The Netherlands, and the track title language is Dutch, we may want to assume that this track is in fact sung in Dutch. The effectiveness of this assumption largely relies on the quality of the language detector. The issue is that many song titles contain only one or two words, making the language classification either tough, ambiguous, or impossible altogether. A solution is to look into song lyrics, and use the language detector on this instead of the title. This requires one to find the lyrics for every track and run the

1 A Python port of Google's *language-detection* library: https://pypi.org/project/langdetect/

language detector on this. Although theoretically possible, this may be a computationally expensive operation.

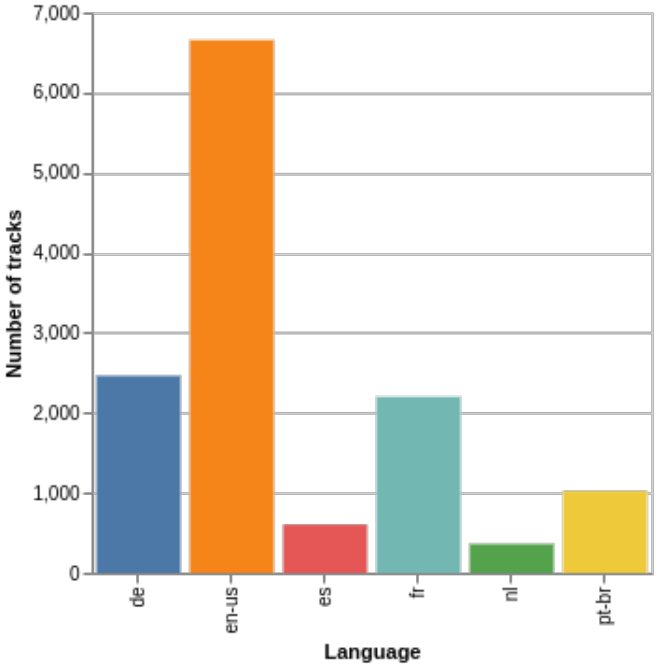


Figure 3.1: The number of tracks per language of six hand-picked languages in the HitFinder Charts dataset, given a labelling method: if a track's title language matches the country that the track is popular in, or if the track's title is English, then classify the track's language as such.

Excluding the fetching of track lyrics, we have given this method a try to classify tracks' sung language. In Figure 3.1 it can be observed that the dataset is not exactly balanced in terms of language frequencies. Furthermore, manual inspection has shown that a sizeable number of track titles' languages are classified poorly by the language detector. As such, this dataset requires quite a bit of cleaning up before use.

3.2.4 *Spotify API & 'spotdl'*

An alternative method for obtaining a relevant, balanced dataset takes advantage of the free Spotify Web API that can be used to query Spotify.[2] With this API, the Spotify search results can be queried for tracks and playlists, and a track's metadata can be requested. Unfortunately, the sung language is not present in a track's metadata at this time. However, we can take advantage of user generated playlists to get language-labelled music after all. This takes inspiration from Chandrasekhar, Sargin, and Ross, who created a dataset of YouTube

2 A (free) Spotify account is required for access to the Spotify Web API: https://developer.spotify.com/documentation/web-api/

music videos (1,000 videos for each of 25 different languages) by querying YouTube for "[...] 'English songs', 'Arabic songs', etc." [6]. Querying the Spotify API similarly, a dataset of track titles, artists and Spotify URLs can be created. Although these tracks cannot be directly downloaded using the free Spotify Web API, having the Spotify track URL, or artist and title, is sufficient for a command-line tool such as '*spotdl*'[3] to process similarly named tracks off YouTube. Note that this method makes various assumptions that may affect the quality of the dataset:

(1) The Spotify API search results are representative of the search query.

(2) The tracks inside user-generated playlists on Spotify are representative of the playlists' title (i.e. there is only music sung in Dutch inside a playlist called "Dutch music").

(3) The command-line tool `spotdl` *is able to match a track title and artist found through Spotify with audio of a YouTube video that matches the track, if such video exists.*

Note that (2) does not always hold. Sometimes 'language playlists' contain piano music; sometimes a playlist for "English music" contains, in fact, Spanish music, and sometimes English music sung by a Dutch artist is considered "Dutch music". That being said, (3) is the main cause for a sub-optimal quality of the dataset. In the ideal case, downloading tracks directly off Spotify would ensure the highest quality possible, and there would be little to no mismatches with track data and the track's audio itself. Downloading the audio of a YouTube video similarly titled to the track's title and artist is prone to mismatches. For instance, some downloaded tracks are karaoke versions of the original track, meaning that there are no vocals present. If these errors are only present in a limited quantity, we may make a fourth assumption:

(4) The errors caused by assumptions (1), (2), and (3) do not affect the quality of the dataset enough to significantly deteriorate the performance of an LID model that is trained on this dataset.

3 Spotify-Downloader: https://pypi.org/project/spotdl/

Part II

METHODOLOGY

4 USED METHODS

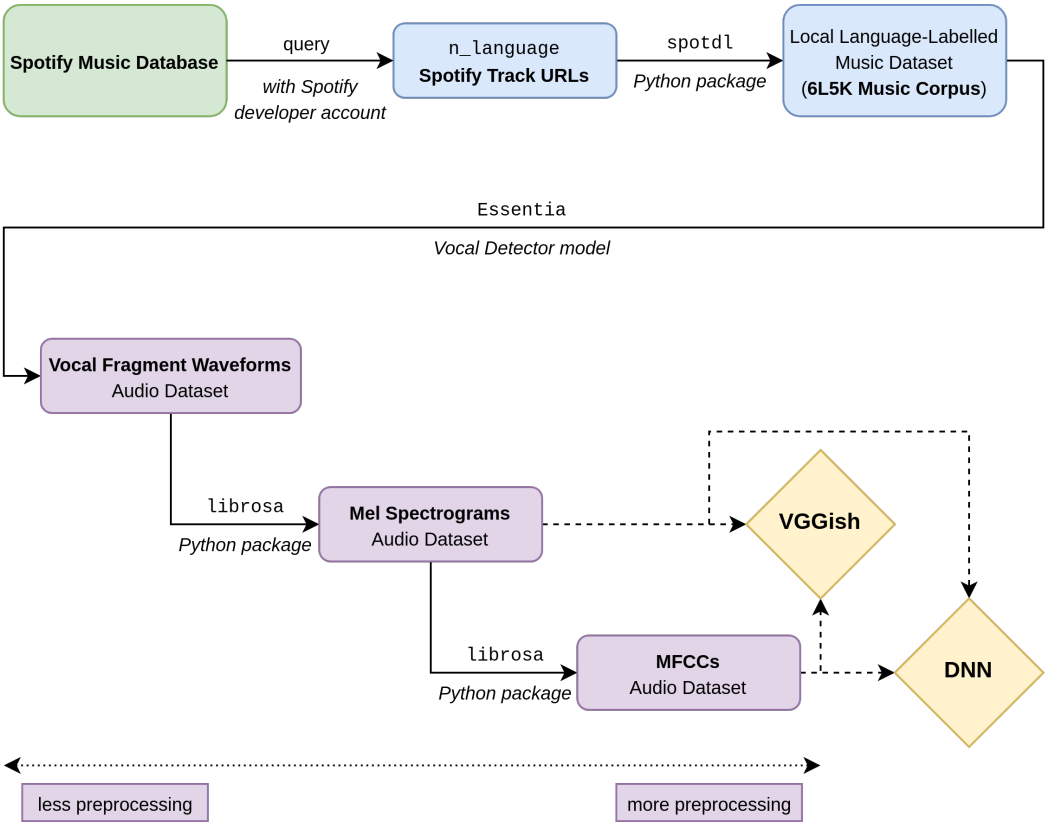


Figure 4.1: Diagram showing an overview of the methodology used in this thesis. In green: Spotify is queried for track URLs for the six languages we are interested in. In blue: for each URL, matching audio is downloaded from YouTube (this is our 6L5K Music Corpus) and processed into 3-second fragments. 'Essentia' makes sure only 3-second fragments containing vocals are kept. In purple: the 3-second waveforms are further processed into mel spectrograms and MFCCs, as input for our systems.

4.1 THE 6L5K MUSIC CORPUS

In Section 3.2 we described existing music datasets. As can be remarked, there is no single publicly available dataset that is large, balanced, varied, and has the sung language as a label. Naturally, without a publicly available, labelled dataset, a music dataset for Automatic Language Identification (LID) needs to be hand-crafted. In Section 3.2.4, we described a method using the Spotify API, downloading music for each language label off YouTube, which gives the

most promising results of the methods described for language-labelled music. As such, we opted for the method described in Section 3.2.4. Here, we start building the music dataset from music, as opposed to starting from speech, which results in the dataset better representing the complete set of music. Moreover, a similar technique has been successfully used by Chandrasekhar, Sargin, and Ross for obtaining a language-labelled sung music dataset [6].

A music dataset for the purpose of LID needs to be carefully crafted. There are a number of requirements for crafting a dataset in order to make it representative of music sung in various languages for use with LID:

1. The data needs to be available: there needs to be a source for obtaining a sufficient amount of music sung in various languages.
2. The data needs to be varied: all tracks must be unique; many different genres must be represented for each language; artists must not be over-represented; the ratio male-to-female of singers must be near equal; and for each language the music must represent the various accents available in that language.
3. The data needs to be high-quality: the quality of the audio tracks at the source must be consistently good, meaning that there should be little to no clipping; all tracks must have a similar volume; and tracks should be near CD-quality.

Based on these three requirements, we have managed to collect audio tracks sung in six languages: English, Dutch, German, French, Spanish, and Portuguese. We call this new language-labelled music dataset the '6L5K Music Corpus'. The following sections go more in-depth into how this music dataset was obtained.

4.1.1 *Spotify Track URLs*

Generating a language-labelled music corpus is not a trivial task. Most music is not widely available for download. Spotify – the popular music streaming service – requires a paid subscription for downloading music. On the other hand, Spotify has an API that can be accessed with a free Spotify account, with which music metadata may be collected. In order to obtain a dataset containing audio, we use a Python library `spotdl`, which is able to download audio from YouTube given a track URL on Spotify. It also uses the metadata found on Spotify as the metadata for the downloaded track.

Language	Abbrev.	Query
English	`en`	*english music*
Dutch	`nl`	*nederlandse muziek*
German	`de`	*deutsche musik*
French	`fr`	*musique française*
Spanish	`es`	*música española*
Portuguese	`pt`	*música portuguesa*

Table 4.1: Queries for Spotify for obtaining music in each of the languages in the 6L5K Music Corpus.

For Requirement 1 of hand-crafting a music dataset, YouTube suffices as a source for audio, given its popularity for music videos. This can be downloaded as described with `spotdl`. What remains is the question of how to get the language labels right. For this, we use a similar method to Chandrasekhar, Sargin, and Ross (2011) [6]. With the Spotify API, we can query Spotify for track URLs in certain playlists. We can also query Spotify for playlists with certain names. Our method of obtaining language-labelled music tracks is by querying Spotify for playlists that are named after each language: see Table 4.1 for the exact queries used. This makes the assumption that playlists found by using this type of query contain music sung in the same language. Since the majority of playlists on Spotify are made by its users, we cannot claim for the language labels to be perfectly accurate. However, given that Spotify had 320 million monthly active users in September 30, 2020[1], and given that there is no clear reason for users to put music in playlists that does not represent the playlist's title, this assumption should be safe to make.

4.1.2 *Downloading Audio With SpotDL*

For the 6L5K Music Corpus, Spotify was queried for 5,000 track URLs and metadata, and audio was downloaded from YouTube on August 1, 2019. The queries that were used for each of the six languages can be seen in Table 4.1. In order to best adhere to Requirement 2, we made sure during the collection of Spotify track data to only take unseen tracks into account, so that all tracks are unique. For each artist, a maximum of five tracks was allowed in the dataset, to keep artist variation high as well. With the queries we used, we also left out genre descriptions. It has to be noted that a query such as "english music" might yield more tracks of a genre that is more typical of the language and its countries, as opposed to yielding generic music sung in that language. We hope that this does not affect how well the data represents the overall set of music sung in that language too much.

1 Spotify company info: https://newsroom.spotify.com/company-info/

Furthermore, it is tough to analyse the male-to-female singer ratio and accent diversity, given that there is no direct metadata available for this.

Having obtained a dataset of Spotify track URLs from the queries, audio can be downloaded from YouTube. Specifically, for a given Spotify track URL, it looks at the metadata on Spotify for the artist and title, then queries YouTube with the following format:

```
{artist} - {title} lyrics
```

Spotdl selects the first video YouTube suggests for the query. This best matching video is then used to download the audio from. Note that this leaves room for mismatching errors, since YouTube videos are often uploaded by consumers rather than professionals. However, for popular music tracks, record labels often upload high-quality music videos. Furthermore, by design of many modern search engines, the more popular a search result is, the higher it appears in the searched list. This applies to YouTube as well as Spotify, meaning that querying either will result in popular, thus likely high quality content. As such, track mismatching between Spotify and YouTube should be kept to a minimum.
It has to be noted however, that the further we dive into the language-labelled playlists on Spotify, the fewer popular results we find, thus the higher the odds that mismatched or low-quality audio makes its way into the 6L5K Music Corpus. Figure 4.2 indicates that hundreds of Spotify track URLs did not have matching audio on YouTube. This means that none of the languages have 5,000 tracks in the 6L5K Music Corpus. Luckily, it is apparent that each of the languages have an approximately equal number of tracks missing, and so this dataset can still be considered 'balanced' in terms of the number of tracks per language. Finally, apart from the filters we put in place for obtaining the Spotify track URLs (querying only language-labelled playlists; allowing only unique tracks; one artist occurring at most 5 times), we did not filter out any tracks afterwards, as this would put additional bias into the music dataset regarding what we would filter out.

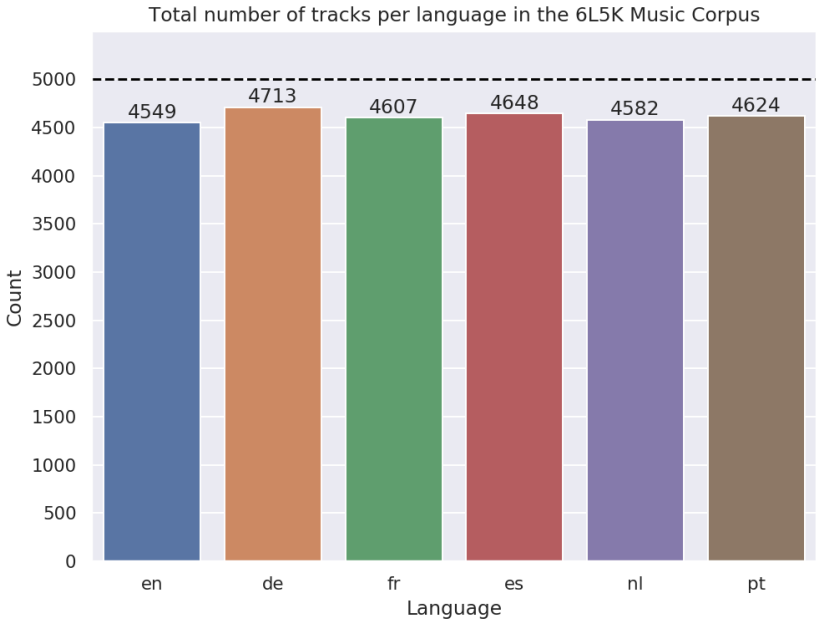


Figure 4.2: Diagram showing the total number of tracks in the 6L5K Music Corpus for each of the six hand-picked languages. Note that none of the counts reach 5,000 – this is because some Spotify URLs do not have matching audio tracks on YouTube.

4.1.3 *Inspecting The Dataset*

For the 6L5K Music Corpus, it is important that the data is representative of the complete set of music, since it will be used to build a generalized classifier with. For this, we may look into the diversity of track release year, track duration, and genres.

Figure 4.3 shows the distribution of release years of all tracks in the 6L5K Music Corpus. From this, we can tell that most tracks have been released at most 20 years from now. This means that the 6L5K Music Corpus is representative of modern music; much less of music throughout history. As such, any classifier trained on this music dataset is expected to work better on music released after the year 2,000 rather than before, although this improvement may be negligible if there is no clear difference between various periods of music production.

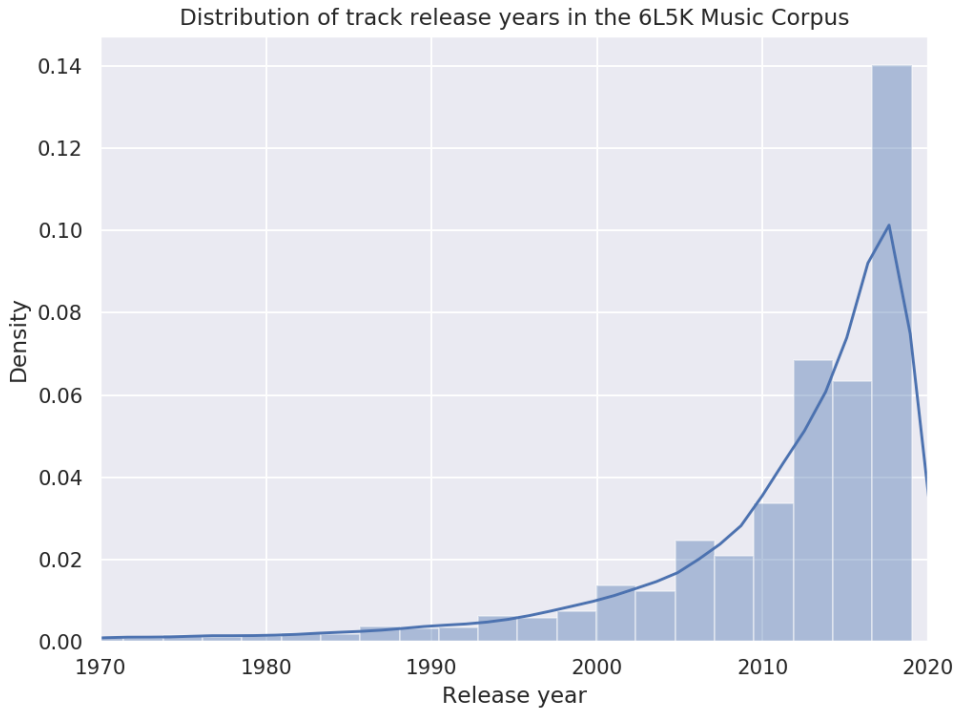


Figure 4.3: Distribution of the release years of tracks in the 6L5K Music Corpus, plotted from 1970 through 2019.

In Figure 4.4, we see how track durations differ per language label. Most importantly, we can see that for each of the languages, a majority of tracks have a duration of approximately three to four minutes, with variations ranging from two to six minutes – there is no clear difference between the languages in this aspect. This means that, given the fact that each of the languages also contain an approximate equal amount of tracks, the total amount of music data for each of the languages is approximately equal as well. This suggests that the 6L5K Music Corpus is balanced in the amount of training data per label. If we look more closely however, we notice that the languages have some outliers in terms of track durations as well. Most notably, the music corpus contains a few English and German tracks which take over 14 minutes. Of these outliers, two are classical tracks without vocals, and two are spoken fairy tales.

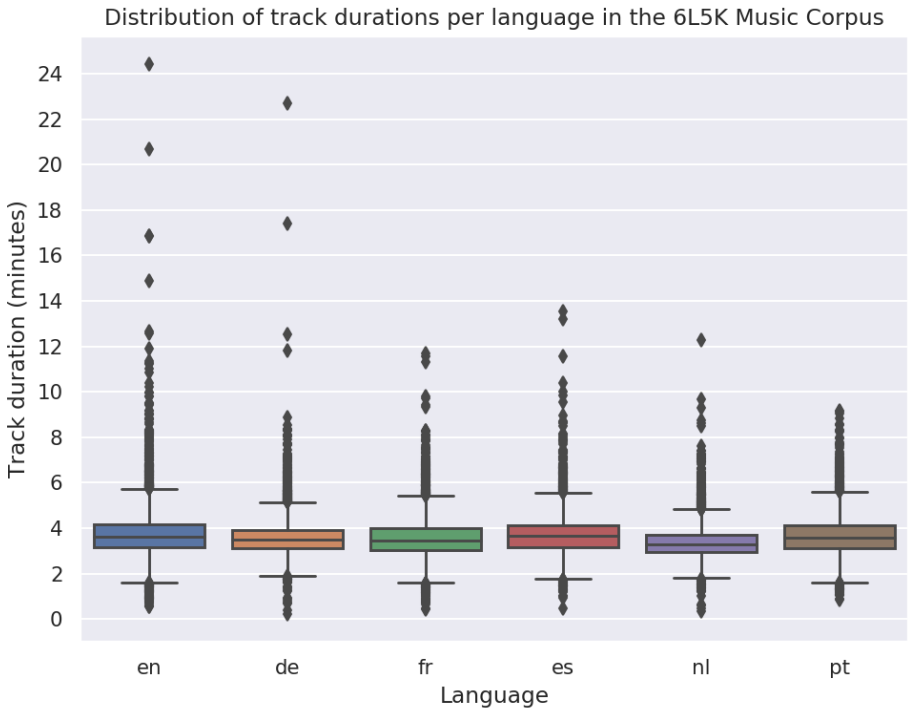


Figure 4.4: Box plot showing the variation in track durations per language in the 6L5K Music Corpus. Although some languages have outlier tracks with a duration of more than 10 minutes, it can be seen that the majority of tracks are between 3 and 4 minutes long, regardless of sung language.

In order to tell whether the 6L5K Music Corpus is a balanced dataset, it can be useful to look at music genres within the dataset – if the most frequent genres correspond with popular genres, then the dataset shows resemblance to the complete set of modern music. Furthermore, genres that appear in all languages indicate that the music between languages does not show too much variation, which is ideal for any model trained on the data in order to not overfit on these specific genres and the sounds that are typical for it. Figure 4.5 shows the 10 most frequent genres in the 6L5K Music Corpus, and how often each genre occurs per language label. Two notable observations can be made. On the one hand, it appears that most languages feature popular genres such as Hip Hop, Pop, Indie, and Rock ('genre balance'). On the other hand, in the 10 most frequent genres, there are a few that are apparently specific to only one of the languages ('genre uniqueness'): Chanson, Cantautor, Carnaval, and Francoton. What this means is that, for each language, we find overlap in the more popular genres, but we also find genres that are language-specific. This can be argued to be a good trait of a music dataset: on the one hand, with genre balance, there will be overlap between languages in terms of how the music

sounds, which means that a self-learning classifier needs to properly learn to identify the sung language in order to classify accurately. On the other hand, with genre uniqueness, there are language-specific genres which are much easier for such a classifier to learn, meaning that the classifier can have a high accuracy on tracks that are more typical for a specific language.

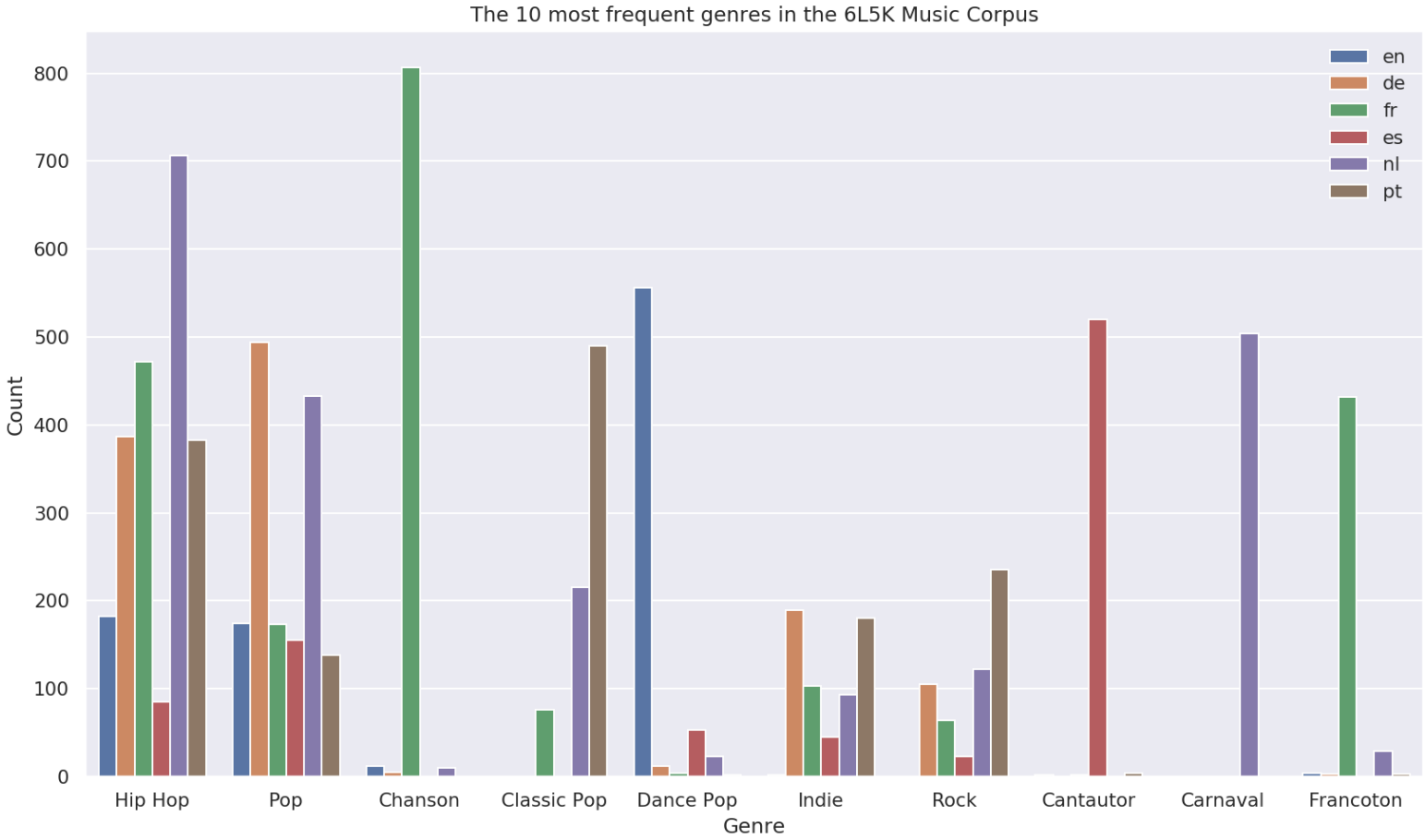


Figure 4.5: The 10 most frequent genres in the 6L5K Music Corpus, and the number of times they occur per language. Note that these genres are derived from the track metadata on Spotify. It can be observed that four genres occur almost exclusively in a single language (i.e. Chanson, Cantautor, Carnaval, and Francoton).

Figure 4.6 illustrates differences in audio quality (specifically the amplitude) of tracks in the 6L5K Music Corpus. Most tracks that were downloaded via YouTube are high-quality, and have waveforms similar to the upper waveform. However, some music videos on YouTube – especially those with lyrics – contain low-quality audio, with the lower waveform being one of which. What you can see in the lower waveform is a very low maximum amplitude, meaning that the music sounds rather quiet. This suggests that the audio may have been tampered with between the initial recording in a studio and uploading to YouTube, as most tracks are released with a higher dynamic range. As such, some audio found on YouTube may be of lower quality than how the music was originally recorded. Fortunately in general, the higher the audio quality of a music video, the more likely it is to become popular as it is then more pleasant to listen to. As we have explained in Section 4.1.2, the 6L5K Music Corpus is expected to feature mostly popular tracks, hence this is not likely to pose much of a problem. Therefore, Requirement 3 holds as well for our dataset.

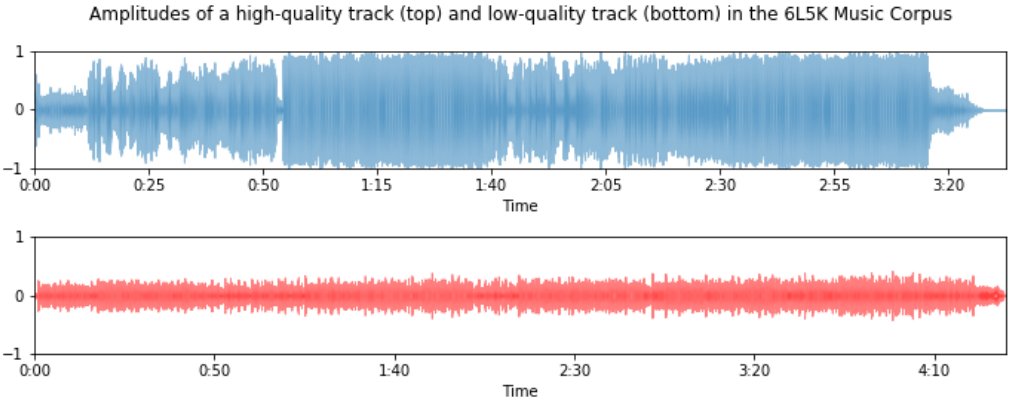


Figure 4.6: Differences in quality of tracks in the 6L5K Music Corpus, specifically in terms of amplitudes (i.e. track volume). The top waveform belongs to the track "Alan Walker - Faded", whereas the bottom is a low-quality version of "Coldplay - Amazing Day".

4.2 VOCAL FRAGMENTATION

4.2.1 *Fragmentation*

Given the audio in the 6L5K Music Corpus, we would like to be able to use this as input to our neural networks. In earlier attempts at LID but on speech, researchers have been able to obtain state-of-the-art classification accuracies using only 3-second audio fragments to learn from, as opposed to using the full track [17]. Using fragments of 3 seconds also increases the number of samples in the dataset by a large margin: given that the average track duration is approximately 3.5 minutes (see Figure 4.4), splitting tracks into non-overlapping 3-second

fragments yields approximately 70× more samples than when a single track is seen as one sample.

For our fragmented data, we take each of the tracks, we trim potential silence at the start and at the end, and we split each track into non-overlapping 3-second fragments and discard any audio less than 3 seconds that may remain at the end of a track. We decided against overlap due to the fact that the 6L5K Music Corpus is a large dataset by itself, and 3 seconds of unseen audio is always preferred over overlapping audio for less overfitting on the training data. If the fragmented dataset ends up too small for its purpose, fragment overlap may be reconsidered. We also keep the information on which artist and title belongs to these fragments, such that fragments from one track do not appear in both the training and the validation dataset, which could otherwise give the validation dataset an unfair advantage given the similarity in sounds within one track.

The next step for the fragmented dataset is to try and put a focus on the vocals in each track. This is a preprocessing step that is not strictly necessary, as we plan to use mel spectrograms and MFCCs which already put a focus on human speech, and neural networks that will be trained to classify the language and should thus learn to distinguish the vocals themselves. However, preprocessing to put a focus on the vocals may drastically help out the training of neural networks if they are otherwise unable to distinguish the language based on the vocals. We describe two methods to separate the vocals from the instrumental and background audio: *direct* and *indirect* vocal separation.

4.2.2 *Direct Vocal Separation*

With direct vocal separation, the portion of the audio that makes up vocals is separated completely from the background or instrumental portion of the audio. As neural networks can be used for general purpose classification tasks, they can also be used to classify which frequencies in a Fourier transform correspond with the vocal portion, and which correspond with the background audio. Zhou (2018) built and trained a singing voice separator using a Recurrent Neural Network (RNN) [52]. It first computes the short-time Fourier transform (STFT) of an audio track using Python package `librosa`, which becomes the input to the RNN, and outputs the STFT ideally containing only the vocals. Using the inverse STFT, the audio can be reconstructed. We found, however, that the quality of the reconstructed audio did not represent a high enough quality vocal separation to be used as a preprocessing step for all audio fragments. The reconstructed audio sounded like various frequency filters were applied, and background

noises as well as instrumentals were still audible to some extent. As such, we opted for a more indirect approach to vocal separation.

4.2.3 *Indirect Vocal Separation*

With indirect vocal separation, we do not separate the vocals from the background audio by modification, but by selection instead. A vocal detector can be run on all 3-second audio fragments to classify which fragments contain vocals. The fragments that do not (sufficiently) contain vocals can be discarded. This way, we can be certain that the audio fragments that are used all contain vocals, and thus the fragments with vocals are separated from those without. Keep in mind however that this does not necessarily reduce the amount of instrumental or background audio in the fragments with vocals. The assumption is that neural networks will learn to focus on the parts of the audio that make up the vocals, as this is ultimately what defines the sung language of a track, and thus these trained networks will learn to disregard background audio.

Essentia's Vocal Classifier

Essentia is an open-source library for audio analysis, which comes with many implementations of audio-related algorithms and pre-trained classifier models [3][2]. One of these classifiers is the voice-instrumental classifier, a Support Vector Machine (SVM) which has been trained to classify an audio sample as either containing voice, or containing purely instrumentals. Although it is unclear exactly how this SVM has been trained, Essentia specifies the accuracy of this pre-trained SVM, shown in Table 4.2. Further specification indicates that a set of high-level descriptors are used as the feature vector for the SVM, including the spectral, time-domain, and tonal descriptors.

	Predicted (%)		
Actual (%)		instrumental	voice
	instrumental	94.20	5.80
	voice	6.60	93.40

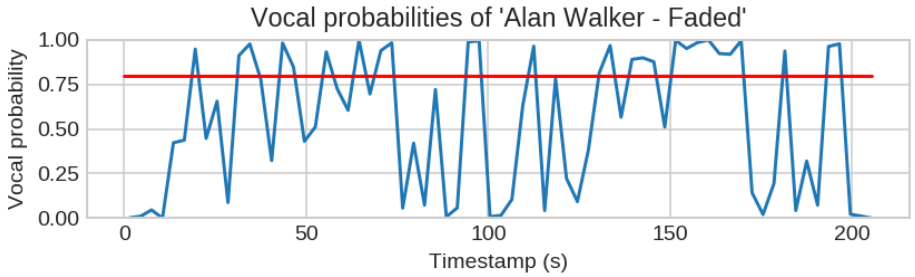
Table 4.2: Accuracy of the voice-instrumental Support Vector Machine by Essentia [3]. The overall accuracy is 93.80%.

Obtaining Vocal Fragments

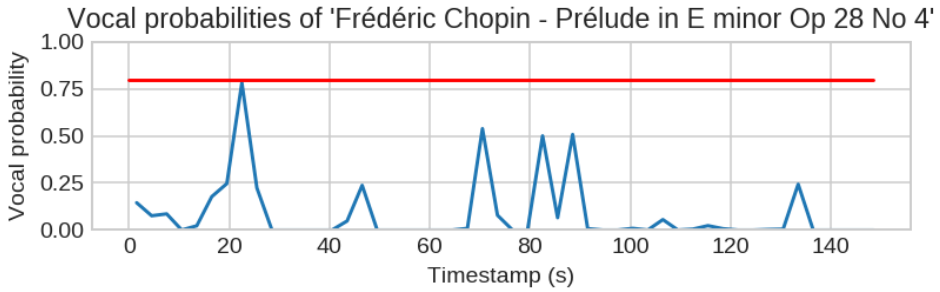
After Section 4.2.1, we were left with a dataset of 3-second fragments in all six languages. Using Essentia, for each fragment, we compute the available high-level features, and then run the voice-instrumental

2 We used version `Essentia 2.1_beta1`

classifier on these features. The classifier output indicates whether voice has been detected in the audio fragment with probability p, or whether instrumental or background audio has been detected with probability $1 - p$. Given probability p, we would like to separate fragments containing little to no voice or vocals, in order to keep the remaining fragments as *vocal fragments*. This can be done by setting a threshold θ for p, above which the fragments are considered to contain vocals (and thus, to 'be vocal'). A default threshold of $\theta = 0.5$ was considered, but we found that too many fragments were kept that did not contain clear vocals. Increasing the threshold yields fewer vocal fragments for the final dataset, but it improves the clarity of vocals in these fragments. Considering this trade-off, we found that $\theta = 0.8$ allows for a large dataset of vocal fragments, while making sure that most fragments clearly contain vocals. Figure 4.7 shows how the vocal fragmentation differs between pop and classical music. With $\theta = 0.8$, the pop-music track still yields 27 vocal fragments out of 71 fragments in total, whereas the classical piece yields zero vocal fragments, as would be expected given that no singing or speaking is present throughout the track. Note that the classification on this classical piece shows the inconsistency of the voice-instrumental classifier – as with any imperfect classifier, variations in output classification probabilities are present, despite the fact that only one piano can be heard throughout the track. It is also not clear where the spike 21 seconds into the track stems from; the audio simply sounds like piano, not even a cough can be heard.



(a) Voice classification probability per 3-second fragment in the pop-music track 'Alan Walker - Faded'.



(b) Voice classification probability per 3-second fragment in the classical piece 'Frédéric Chopin - Prélude in E minor Op 28 No 4'.

Figure 4.7: Voice classification probabilities for each of the 3-second fragments (each of which is plotted at its centre) in a pop-music track and a classical piece. The red line indicates threshold $\theta = 0.8$, above which the 3-second fragments are considered to contain sufficiently clear vocals: 27 fragments for the pop-music track, and 0 fragments for the classical piece.

With $\theta = 0.8$, the vocal fragments have been separated from the non-vocal ones. The size of the resulting dataset can be seen in Figure 4.8. Unfortunately, the balance in terms of the number of tracks per language that was present in the 6L5K Music Corpus does not transfer to the vocal fragmented dataset. Here we can see that Dutch and German tracks have more voice detected in the 3-second fragments. For Dutch this can be explained due to the fact that Carnaval is one of the major genres present in the Music Corpus (see Figure 4.5), and Carnaval hits often sound more speech-like than the average pop song. It remains to be seen whether this language count imbalance leads to any under- or overfitting of a trained neural network for any of the languages.

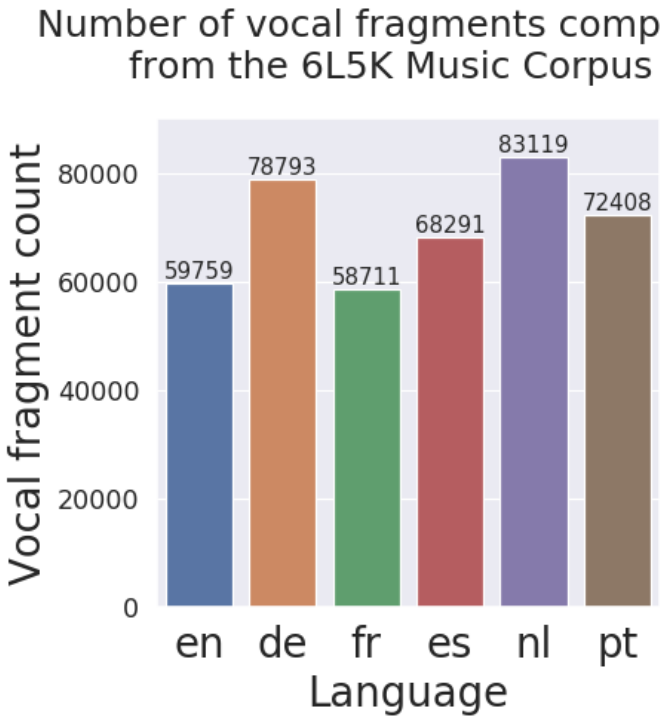


Figure 4.8: The number of vocal fragments per language, computed from the tracks in the 6L5K Music Corpus. Although the 6L5K Music Corpus is balanced in terms of the amount of tracks per language, the vocal separation does not keep this balance.

4.3 MEL SPECTROGRAMS & MFCCS

4.3.1 *Mel Spectrograms*

In Section 4.2 we obtained a dataset of 3-second music fragments which likely contain vocals. Although these vocal fragments can be used directly as input to a 1-dimensional Convolutional Neural Network (1D CNN) [43], we opt to process the data further in order to put more emphasis on the vocal aspects of the audio. As described in Section 2.3, the mel spectrogram uses a mel scale which allows more emphasis on human perception of the audio, as it scales linearly with how it is perceived by humans. Using a spectrogram as input instead of the raw waveform, we are able to use neural networks that have been optimized for pattern recognition in image data. The mel spectrograms are computed with `librosa`, a popular Python library for audio manipulation and analysis[3]. The default parameters are used except that the sampling rate is 16,000. As such, the input is

3 For documentation on the computation of mel spectrograms, see: https://librosa.org/doc/main/generated/librosa.feature.melspectrogram.html

sampled with windows of size 2048 using the Hann window function, with a hop length of 512. A total of 128 mels are used, which results in mel spectrograms of 130×128 pixels. For the first model that we are implementing, a Deep Neural Network (DNN), this mel spectrogram could be used as the input by flattening it to a feature vector of $130 \times 128 = 16640$ input nodes. However, for a DNN this is a rather large feature vector. DNNs generally perform best with a much smaller feature vector; even a smaller image of $64 \times 64 = 4096$ input nodes is seen as rather large for a DNN, and may pose a problem for convergence when training the network. On the other hand, our second model – VGGish – is a convolutional neural network (CNN) which generally performs better on image-related tasks than a DNN, meaning that for VGGish, an image of $64x64$ pixels is perfectly usable. VGGish, which will be further described in Section 4.6, comes with pretrained weights, which have been trained on mel spectrograms of 96×64 pixels however. As such, we are using mel spectrograms of the same size despite the fact that $96 \times 64 = 6144$ input nodes makes the DNN rather complex. Our computed mel spectrograms need to be downscaled from 130×128 to 96×64. Doing this through linear interpolation, the mel spectrograms are downscaled but the features remain similar, which is important for pattern recognition within the spectrograms. A downscaled sample is shown in Figure 4.9. Note that the values of the mel spectrograms are not normalized, as VGGish uses the full range of spectrogram values as the input by default.

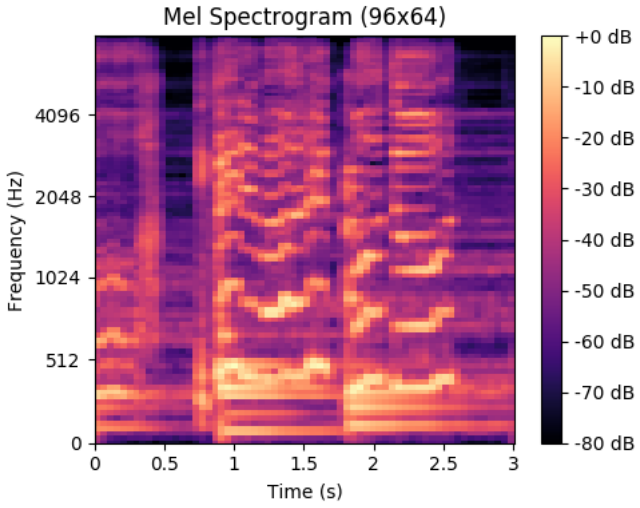


Figure 4.9: A sample mel spectrogram of a fragment of music containing vocals, downscaled to 96×64 features.

4.3.2 *Mel-Frequency Cepstral Coefficients*

Although VGGish should train well with mel spectrograms as input, DNNs have been more successful for Automatic Language Identifica-

tion when using Mel-Frequency Cepstral Coefficients (MFCCs) [28], or MFCC-like features such as PLP [29]. We focus on MFCCs instead of PLP since there is little improvement in performance when using either of the two [38], and MFCCs are better documented. The MFCCs are calculated with `librosa` as well[4]. MFCCs are a more compressed, but also more decorrelated representation of what is captured with mel spectrograms. The computation relies heavily on the computation of mel spectrograms, as it can be seen as processing the mel spectrograms further. Specifically, the MFCCs can be computed by calculating the Discrete Cosine Transform (DCT) of the mel spectrogram, essentially creating a spectrogram of a spectrogram – this is referred to not as a spectrum-of-a-spectrum but a *cepstrum* (notice the first four letters reversed). As such, MFCCs are the cepstral coefficients given the previously computed frequencies on the mel scale. Given that Lopez-Moreno et al. (2014) calculated 39 PLP features, we decided to let our MFCCs capture 39 features as well (`n_mfcc = 39`). However, whereas they are using 21 frames for each of the PLP features, we decided to keep the 64 frames from the previously computed mel spectrograms for our MFCCs, as the data can always be downscaled when needed. In fact, we tested both the DNN and VGGish on downscaled mel spectrograms and MFCCs (mel spectrograms of 48×64 and 48×32; MFCCs of 39×32 and 20×32), but we saw no noticeable improvements of using downscaled feature vectors. As such, we will keep using mel spectrograms of 96×64 and MFCCs of 39×64 pixels. A sample of MFCCs is shown in Figure 4.10. Again, no normalization is applied to the MFCCs; this is because VGGish uses non-normalized spectrograms as input, therefore we also feed it similar MFCC data.

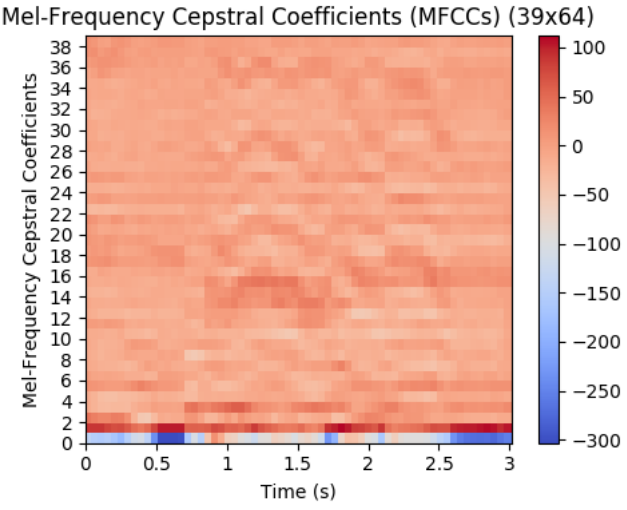


Figure 4.10: Sample MFCCs computed from the mel spectrogram in Figure 4.9 with 39×64 features.

4 For documentation on the computation of MFCCs, see: https://librosa.org/doc/main/generated/librosa.feature.mfcc.html

4.4 INPUT DATA

4.4.1 *Training & Validation Data*

Given the dataset of vocal fragments, we have now computed mel spectrograms and MFCCs for each of the fragments. With this and the original language label of a fragment, the data that serves as the input to our neural networks is complete. The set of vocal fragments, each now consisting of a mel spectrogram, the MFCCs, and the language label, is shuffled. In order to check during training how well a model performs on unseen data, a subset of the data is taken as a separate validation dataset. 90% of the vocal fragments are used for training; the remaining 10% is used for validation. Each of the vocal fragments contains the name of the track it originates from, and we make sure that there is no overlap in track fragments between the training and validation data. That is, if any vocal fragment of some Track A appears in the validation data, all vocal fragments of Track A are used for the validation data. After all, having very similar samples in both the training and the validation data will give the validation data an unfair advantage, as these samples should be unseen data. It can be argued that the same is true to some extent not just for various fragments of one track, but also for various tracks of one artist. We consider different tracks by a single artist to be different enough to be allowed in both the training and the validation data, and assume that this does not give the validation data an unfair advantage over truly unseen test data.

4.4.2 *Testing Data*

The testing data is obtained completely separately from the training data. In fact, none of the tracks in the 6L5K Music Corpus are used. In a similar fashion however, new tracks are downloaded off YouTube via the Spotify API (see Section 4.1.2). Instead of the 5,000 tracks per language that were downloaded on August 1, 2019, for the testing dataset we downloaded metadata for another 5,000 tracks from Spotify on 17 December, 2020. However, not all tracks are used, since it is very likely to overlap with the training and validation data. We take a subset of the tracks, containing only tracks that appear in the 2020 metadata but not in the 2019 metadata. These unique tracks are download, and considered to be the final unseen test dataset. The number of tracks is shown in Table 4.3. Similar to the training and validation data, 3-second fragments are taken from each of the tracks, and we separate these fragments into vocal and non-vocal (instrumental) using the vocal detector from Essentia (see Section 4.2.3). The distribution of fragments which contain vocals per language is shown in Figure 4.11. Here we can see that there is not much difference between vocal

fragments computed for tracks of each of the languages. On average, a track in the test dataset appears to yield roughly 4 vocal fragments, and most tracks yield between 0 and 10 vocal fragments.

Language	Track count
English	2440
German	2222
French	1886
Spanish	2015
Dutch	1456
Portuguese	1914

Table 4.3: Number of tracks per language, unique to the testing dataset.

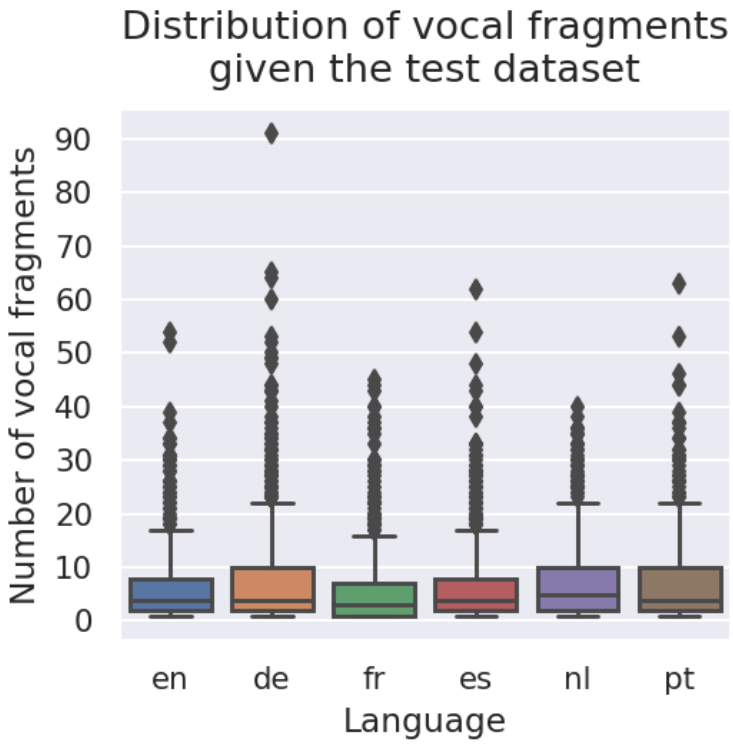


Figure 4.11: Distribution of vocal fragments per track, computed from the tracks in the test dataset. Most tracks regardless of language have fewer than 10 computed 3-second audio fragments containing vocals.

4.5 DEEP NEURAL NETWORK

For the Deep Neural Network (DNN) architecture, we use a similar method to the model trained by Lopez-Moreno et al. (2014) for

Automatic Language Identification of speech data [29]. Our default parameters can be seen in Table 4.4, and the architecture is visualized in Figure 4.12. Interestingly, Lopez-Moreno et al. do not indicate whether dropout was used. We use 20% dropout for the input layer and 50% dropout for the hidden layers, as this is more commonly used [1]. Furthermore, since the mel spectrograms and MFCCs are not normalized, we make use of Batch Normalization in the DNN to effectively feed normalized values to the DNN anyway (refer to Section 2.3.3). Lastly, we make use of early stopping, which means that when the validation loss did not improve over the last n epochs, then training is halted to prevent overfitting on the training data when no improvements on the validation data are seen.

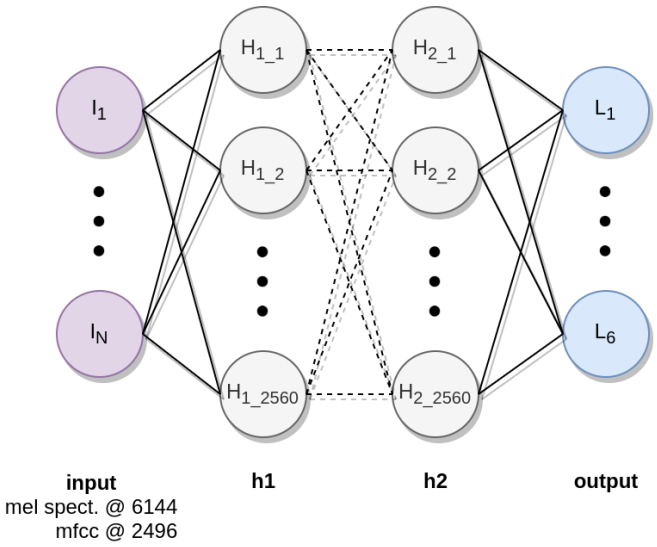


Figure 4.12: Visualization of our Deep Neural Network (DNN) architecture. The number of input nodes depends on whether mel spectrograms or MFCCs are used. Both hidden layers contain 2560 nodes each, and the output layer contains 6 nodes corresponding with the number of languages to be classified.

Parameter	Value
Mel architecture:	6144] – [2560 – 2560] – [6
MFCC architecture:	2496] – [2560 – 2560] – [6
Dropout:	20%] – [50% – 50%] – [0%
Activation functions:	linear] – [relu – relu] – [softmax
Loss function:	Categorical Cross-Entropy
Regularization:	Batch Normalization
Epochs:	50
Early stopping:	10
Batch size:	200
Optimizer:	Adam (β_1 = 0.9; β_2 = 0.999; ϵ = 1e-7)
Learning rate:	0.001
Weight initialization:	Truncated Normal

Table 4.4: Default parameters for the Deep Neural Network (DNN). For the network architectures and activation functions, the open blocks indicate the input and output layer sizes; the closed block refers to the hidden layers.

4.6 VGGISH

VGGish is a model built on a Convolutional Neural Network (CNN), shown to obtain very promising results in the field of audio classification [19]. Not only is the implementation of VGGish open-source, but a model pretrained on mel spectrograms from the Audio Set corpus is available as well [15]. Given that VGGish has a predefined architecture, we base our version of VGGish on this same architecture[5]. Our exact default parameters for VGGish can be found in Table 4.5, and an overview of the architecture is shown in Figure 4.13. Two pretrained models are available: one with the top fully connected (FC) layers, and one without. We will use the model that includes the FC layers. For any modifications made to to the architecture that make it so that the pretrained model weights cannot be used (such as resizing an FC layer), those specific weights will be randomly initialized by means of the Glorot Uniform initialization while the rest keeps the pretrained weights. Furthermore, when using MFCCs instead of mel spectrograms, the pretrained weights can still be used in all layers except the input layer. Even though mel spectrograms and MFCCs are very different data types, the patterns may look similar, and the pretrained weights may still be a better initialization than a random one. Before the hyperparameters are formally optimized in Section 6, a

5 Specifically, the `Keras` version at https://github.com/DTaoo/VGGish which is in turn based on the original `TensorFlow` version at https://github.com/tensorflow/models/tree/master/research/audioset/vggish

number of informal runs are made in order to find parameters that let the network converge to some extent. This is why a cyclical learning rate of 0.00002 is used, as it appeared to converge well enough to be used as a good starting point.

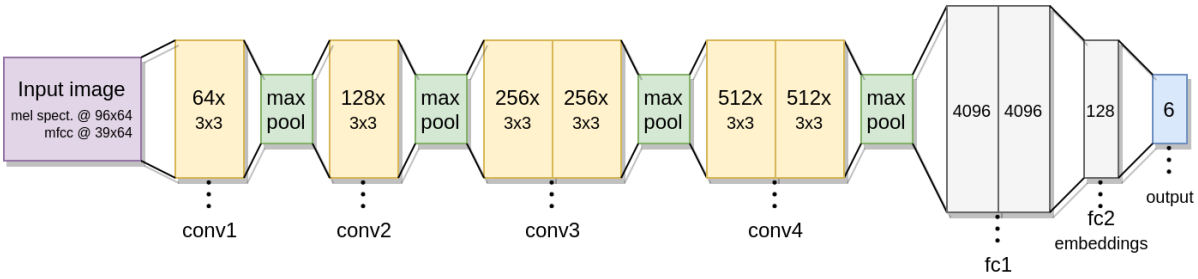


Figure 4.13: Visualization of our VGGish architecture. The yellow blocks indicate convolutional layers, the green blocks indicate the 2×2 max-pooling layers, and the grey blocks indicate the fully connected (FC) layers (twice 4096 nodes, then once 128 nodes), much like the hidden layers in the Deep Neural Network in Figure 4.12. The FC output layer is shown in blue, which contains 6 nodes corresponding with the number of languages to be classified.

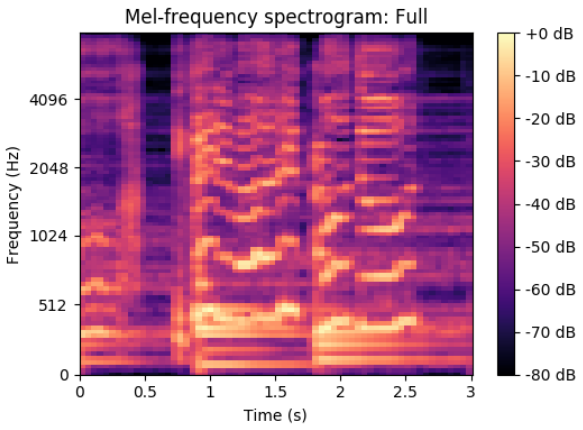
Parameter	**Value**
Mel spectrogram architecture:	6144] – [{FMs} – {FCs} – Embeddings] – [6
MFCC architecture:	2496] – [{FMs} – {FCs} – Embeddings] – [6
FC layers:	[4096 – 4096]
Embeddings:	[128]
Activation functions:	linear] – [{relu} – {relu} – relu] – [softmax
Loss function:	Categorical Cross-Entropy
Regularization:	No Dropout
Epochs:	100
Early stopping:	10
Batch size:	200
Optimizer:	Adam (β_1 = 0.9; β_2 = 0.999; ϵ = 1e-7)
Learning rate:	0.00002, cyclical
Weight initialization:	Glorot Uniform

Table 4.5: Default parameters of our version of VGGish, with the same architecture base as the original. In the architectures described above, the open blocks indicate the input and output layer sizes; the closed block refers to the hidden layers. Furthermore, curly brackets indicate a multi-layer structure, where 'FMs' refer to VGGish' default Feature Maps and 'FCs' refer to the Fully Connected feedforward layers. 'Embeddings' refer to the single, penultimate layer.

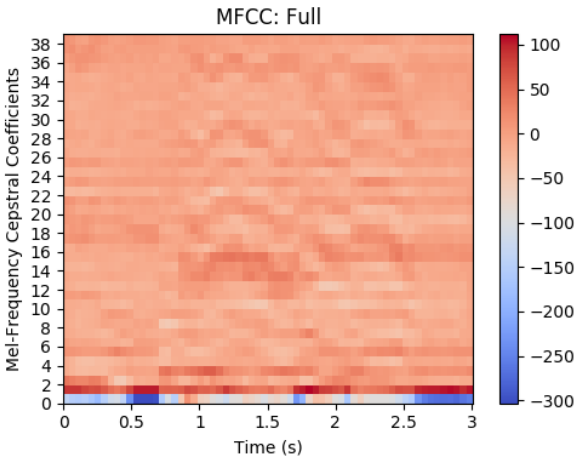
5 DNN OPTIMIZATION

5.1 DNN: BAND REMOVAL

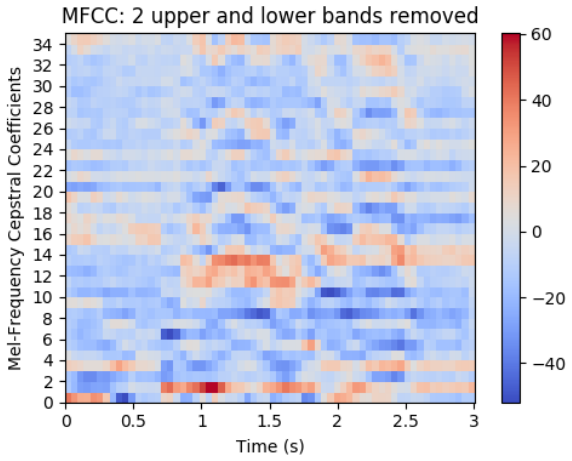
For the Deep Neural Network (DNN), we use the architecture as described in Table 4.4 as the default. Figure 5.1 shows a sample 3-second vocal fragment sung in Dutch. More precisely, Figure 5.1(a) shows a mel spectrogram that is computed from the audio fragment, and Figure 5.1(b) shows the MFCCs that are computed from the mel spectrogram. The mel spectrogram shows distinct patterns of power, whereas such pattern in the MFCCs is less apparent, instead showing the most distinct pattern in the lower frequency bands (this pattern can be found in effectively all vocal fragment MFCCs). As such, we hypothesized that the initial training may profit from more contrast in the MFCCs, meaning more distinct patterns. By removing a number of frequency bands, Figure 5.1(c) and (d) were obtained. For use with the DNN, all image arrays are flattened. This way, the mel spectrograms of 96x64 yield 6144 input values; the default MFCCs of 39x64 yield 2496 values, and the MFCCs with four bands removed, at 35x64, yield 2240 values. We will see whether removing the lower frequency bands and optionally the upper bands as well improves the initial performance of the DNN.



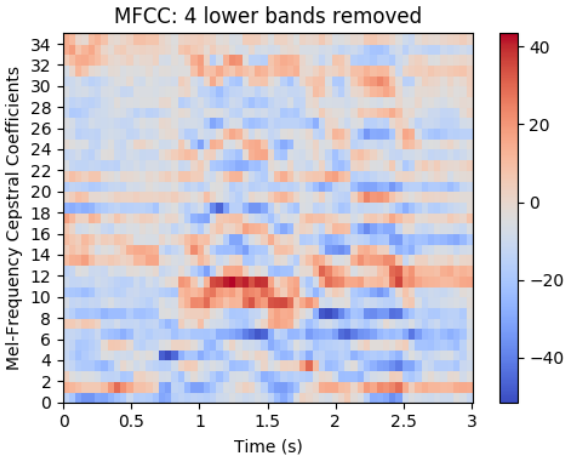
(a) Mel-frequency spectrogram of 96x64 pixels.



(b) MFCCs of 39x64 pixels.



(c) MFCCs of 35x64 pixels;
2 upper and 2 lower bands removed from (b).



(d) MFCCs of 35x64 pixels;
4 lower bands removed from (b).

Figure 5.1: A 3-second vocal fragment visualized by its mel spectrogram and its MFCCs. Figures (c) and (d) are a variation on (b) only with specific frequency bands removed.

Figure 5.2 shows the results of the DNN trained with our default parameters, on the four data types shown in Figure 5.1. First off, we can see that the DNN converges more slowly on the mel spectrograms than on MFCCs, and both the training and validation accuracy remain lower than when MFCCs are used. Furthermore, there does not seem to be a large effect of the removal of MFCC bands, except that training and validation accuracies stay slightly lower when more of the lower bands are removed. It appears that the best results are obtained when keeping the MFCCs as-is. As such, we will put the focus on the full MFCCs as input data, visualized in Figure 5.1(b).

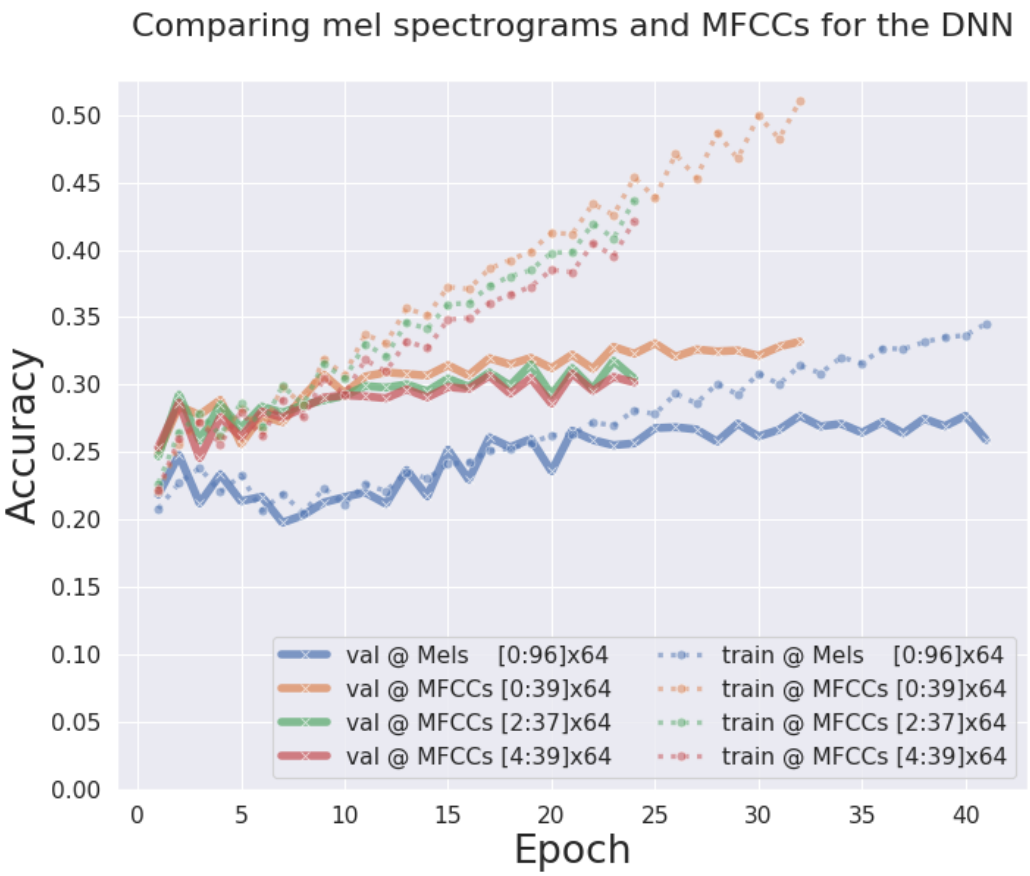


Figure 5.2: Results of the Deep Neural Network (DNN) with default architecture being trained on mel spectrograms and MFCCs; both training and validation accuracies (resp. 'train', 'val') are shown. For some of the MFCCs, four bands have been removed, indicated by array indexing (e.g. `[2:37]` discarding two of the lower and upper bands).

5.2 DNN: HIDDEN LAYERS

Now that the focus remains on training from MFCC data, let us experiment with the network architecture. The number and size of the hidden layers determine the capacity of the network, which needs to be large enough to be able to approximate the function of Automatic Language Identification (LID) given the input data, yet small enough to converge quickly enough and to prevent overfitting. Figure 5.3 shows the performance of the DNN with various hidden layers. As described in Table 4.4, the default hidden layer contains `2560-2560` nodes, which is based on the findings by Lopez-Moreno et al. (2014). Interestingly, although not many differences can be observed in this Figure, it is clear that the highest validation accuracy at the end of the training sessions is obtained with the same setup for the hidden layer

as worked best in the literature for LID of speech, namely `2560-2560`. When using a fewer nodes for the hidden layers, thus having a less complex network, the validation accuracy starts higher but never reaches much above this score. With three large hidden layers, that is `4096-4096-4096`, the network does not progress beyond 19.80% validation accuracy, which seems to be obtained when consistently predicting only the most frequent class in the training data. Since the network needs to have the capacity to approximate the complex function of LID, and the literature also indicates that `2560-2560` works well for LID given PLP (MFCC-like) features, it is a safe choice to keep this hidden layer architecture.

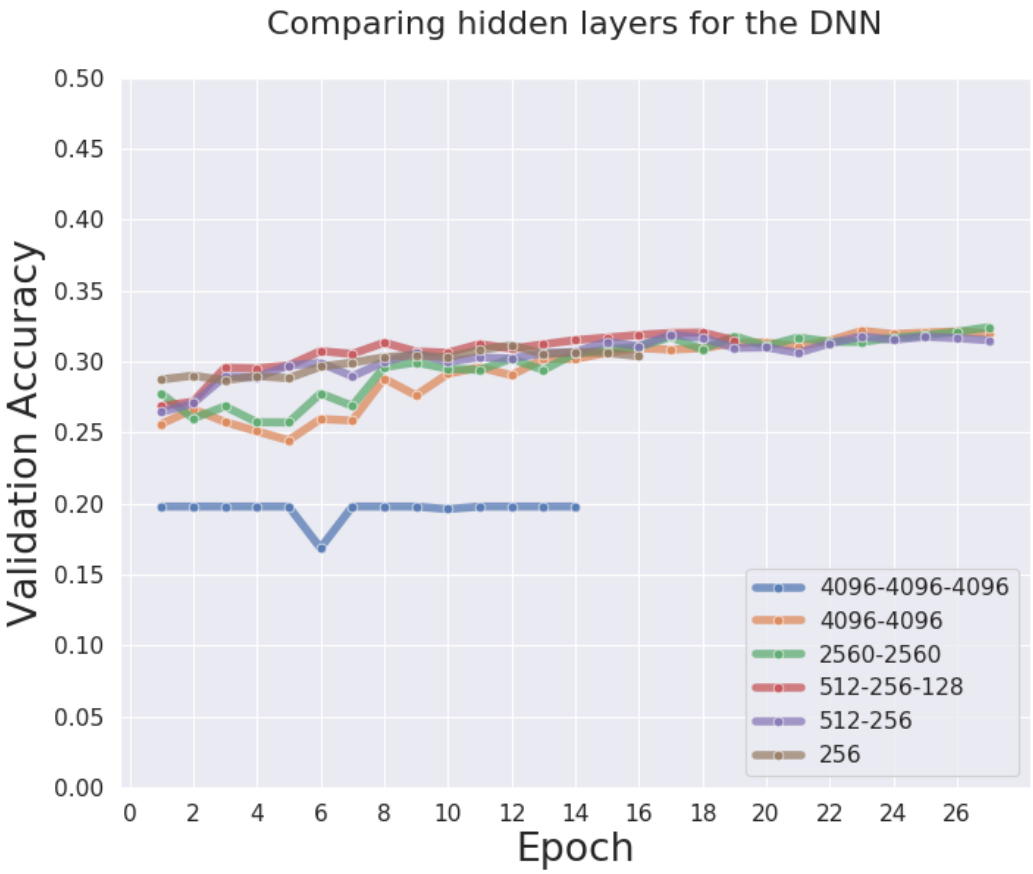


Figure 5.3: Results of the Deep Neural Network (DNN) being trained on default MFCCs, comparing various hidden layer architectures of the network.

5.3 DNN: LEARNING RATE

So far, none of the potential improvements to the training process have been shown to be beneficial. Looking at various learning rates on the other hand, we have found a welcome change for improved

performance. Not only did we test different learning rates, we also implemented a cyclical learning rate [44], which effectively varies the learning rate within a training session. Figure 5.4 shows the effect of various cyclical and non-cyclical learning rates on the validation accuracy. The learning curves between various learning rates are similar, but it becomes clear that the use of a cyclical learning rate helps to converge faster and train for longer. The highest validation accuracy (35%) is found when using a cyclical learning rate of 0.00002, which is also where the model trains the longest given the implementation of *early stopping* with 10 epochs.

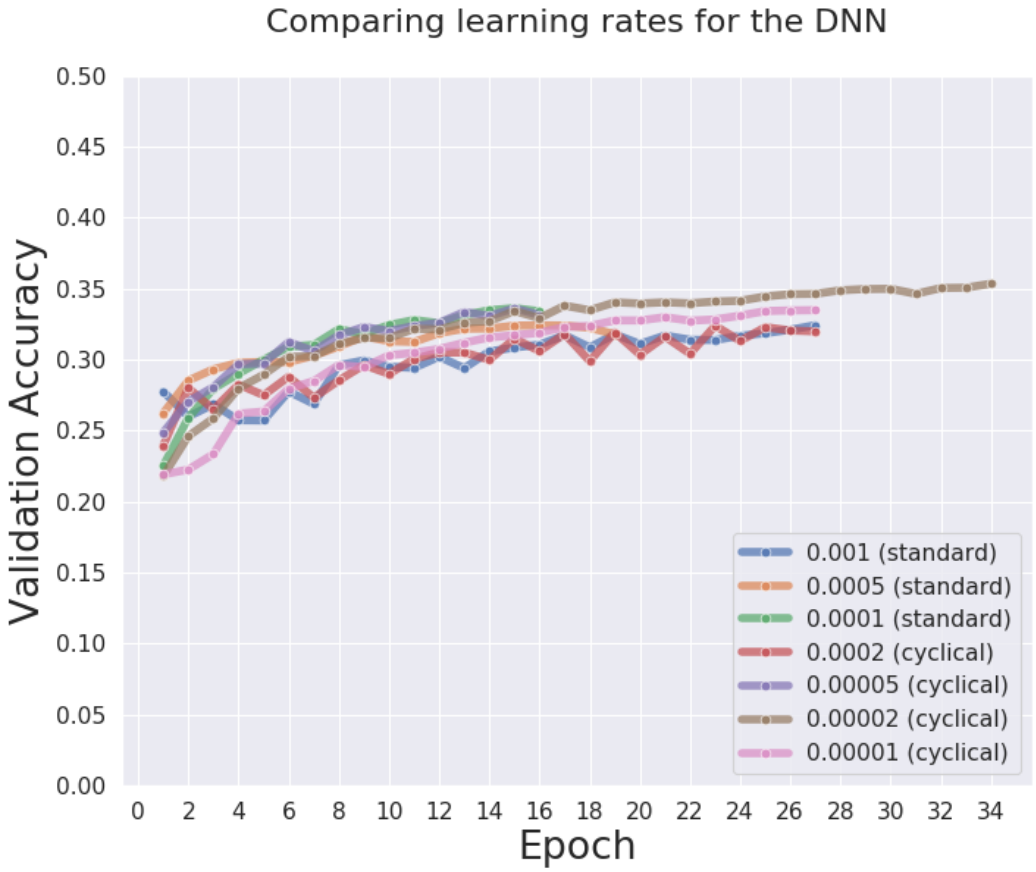


Figure 5.4: Results of the Deep Neural Network (DNN) being trained with various learning rates; cyclical and non-cyclical.

5.4 DNN: FINE-TUNED PARAMETERS

In the previous sections, we described our method of optimizing the hyperparameters of the DNN for use with the vocal fragments computed from the 6L5K Music Corpus. We have hereby shown that the hyperparameters in Table 5.1 lead to the best found model for the Deep Neural Network (DNN) when training on the vocal fragment

data. The predictive power of this setup can be seen in Chapter 7, where the network is examined with the test dataset.

Parameter	Value
MFCC architecture:	2496] – [2560 – 2560] – [6
Dropout:	20%] – [50% – 50%] – [0%
Activation functions:	linear] – [relu – relu] – [softmax
Loss function:	Categorical Cross-Entropy
Regularization:	Batch Normalization
Epochs:	50
Early stopping:	10
Batch size:	200
Optimizer:	Adam (β_1 = 0.9; β_2 = 0.999; ϵ = 1e-7)
Learning rate:	**0.00002, cyclical**
Weight initialization:	Truncated Normal

Table 5.1: Optimal parameters for the Deep Neural Network (DNN) by empirical evidence. Bold indicates alterations from the default parameters in Table 4.4. For the network architecture and activation functions, the open blocks indicate the input and output layer sizes; the closed block refers to the hidden layers.

6 VGGISH OPTIMIZATION

6.1 VGGISH: WEIGHT INITIALIZATIONS

In Table 4.5, we described the network architecture behind VGGish, as well as the hyperparameters for training VGGish on the vocal fragment data computed from the 6L5K Music Corpus. Although VGGish was designed for training on mel spectrograms, we have also trained VGGish on MFCCs, making for a slightly different input size at 39x64 pixels as opposed to 96x64. Figure 6.1 shows the validation accuracy of VGGish being trained on mel spectrograms and MFCCs. Each input type is either trained with randomly initialized weights; weights from an existing, pretrained model that was trained on mel spectrograms for speech classification, or the same pretrained weights but with all convolutional layers being frozen (i.e. only the input layer and fully connected layers change weights during training). In the case of pretrained weights, all weights that cannot be loaded are still randomly initialized.

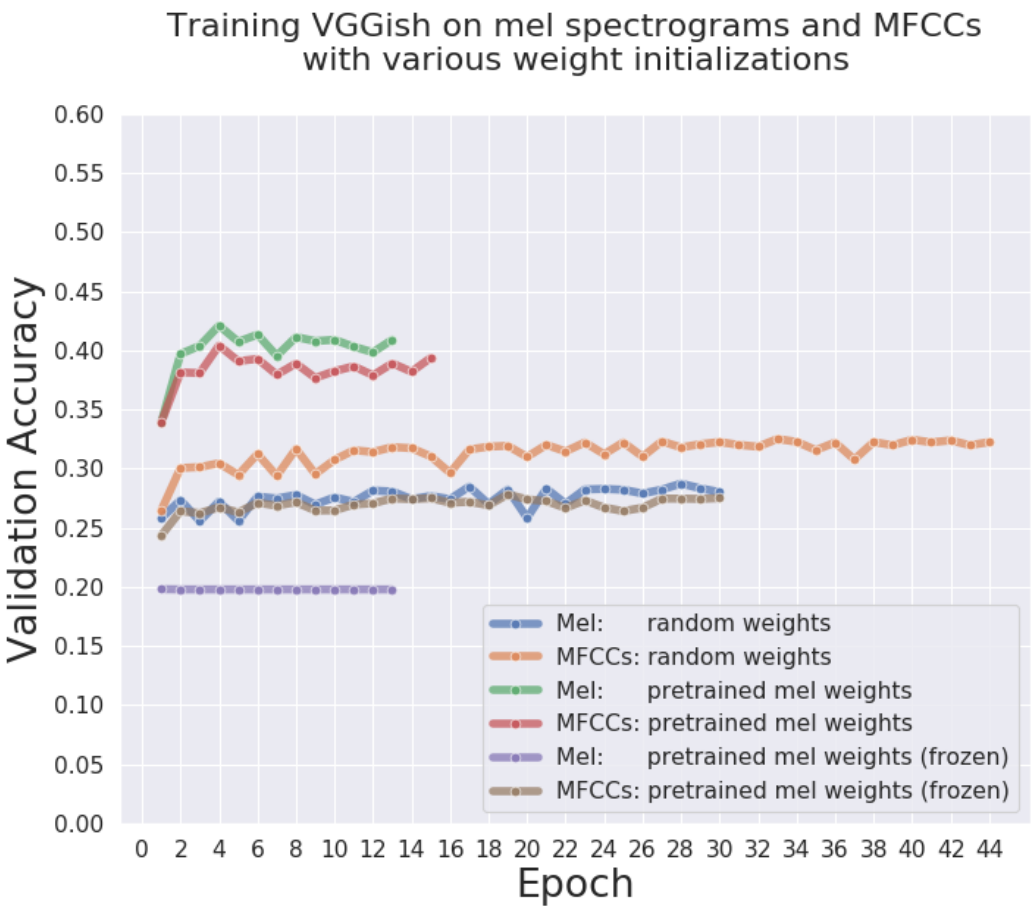


Figure 6.1: Results of VGGish being trained on mel spectrograms and MFCCs with various weight initializations. The pretrained mel weights refer to an existing model that was trained on mel spectrograms for speech classification.

As is shown, VGGish converges most quickly with the pretrained weights, and with all layers being trainable (not frozen). It appears that the pretrained model's weights are a better initialization for learning a similar task (Automatic Language Identification (LID) of music as opposed to speech) than randomly initialized weights (Glorot Uniform). Interestingly, this is regardless of whether the same input type as the pretrained model is used (mel spectrograms), or a different type of input (MFCCs), although similar patterns between the two data types can be seen (refer to Figure 5.1(a) and 5.1(b)). Still, mel spectrograms with pretrained weights yields the best results, which is in contrast to the results of the Deep Neural Network (DNN) in Section 5.1 where MFCCs allowed the network to score a higher validation accuracy than mel spectrograms, albeit with no pretrained weights available.

6.2 VGGISH: DROPOUT

Our model of VGGish can be seen to overfit on the training data in Figure 6.2. Here, we apply dropout to the Fully Connected (FC) layers, and not in the convolutional or pooling layers [51], to try and alleviate the overfitting. The results indicate that a higher amount of dropout helps marginally reduce overfitting on the training data, with the training accuracy lowering as the amount of dropout increases. However, the validation accuracy stays at roughly the same level. In order to decide on which amount of dropout works best, we can look at the training sessions that ran the longest, indicating that the validation loss was improving for longer due to `early stopping`. For this, the runs at 30% and 50% dropout stand out slightly. Since the differences seem negligible when it comes to validation accuracy, we prefer to go for the middle ground, using 30% dropout for the FC layers in future runs, which also happens to end with a slightly higher validation accuracy at the final epoch here.

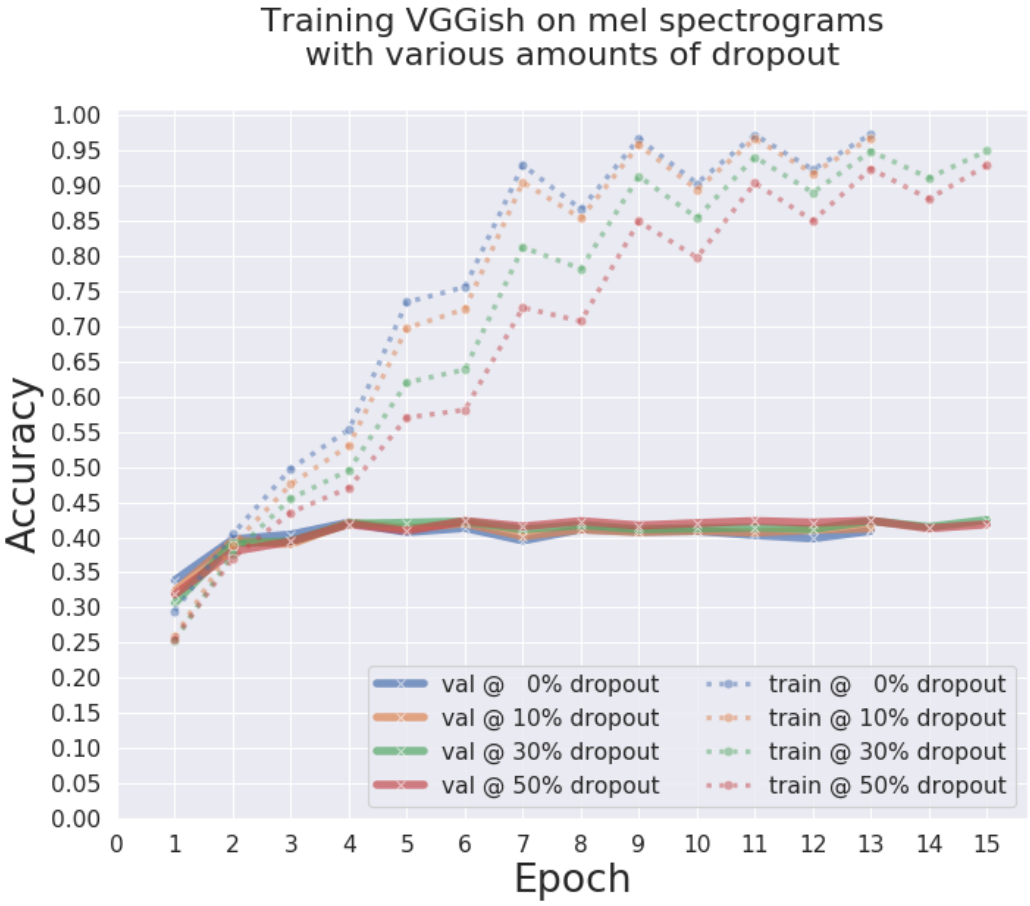


Figure 6.2: Training and validation accuracies for various amounts of dropout during the training of VGGish on mel spectrograms.

6.3 VGGISH: HIDDEN LAYERS

Continuing improving the hyperparameters of the VGGish network, we will take a closer look at the architecture of the Fully Connected (FC) layers. Originally, after the Feature Maps (FMs) in VGGish, there are two FC layers each of 4096 nodes, after which there is an embedding layer (Embeddings) of 128 nodes, also fully connected. We added a final softmax layer after the Embeddings for classifying each of the six languages. For hyperparameter testing, we vary the FC layers to see how much it affects the training and validation accuracy. See Figure 6.3. Here, we can see that the validation accuracy is not affected much by a change in the FC architecture. On the other hand, smaller FC layers (i.e. having 512 nodes per layer as opposed to 4096) cause less overfitting on the training data. Although not much of a difference can be seen in the validation accuracy, the architecture where there is only a single FC layer of 4096 nodes does yield a higher validation accuracy than the other architectures for most epochs. Despite the fact that this architecture makes VGGish largely overfit on the training data, obtaining almost 95% training accuracy after 15 epochs, we still consider this the better architecture given the validation accuracy scores.

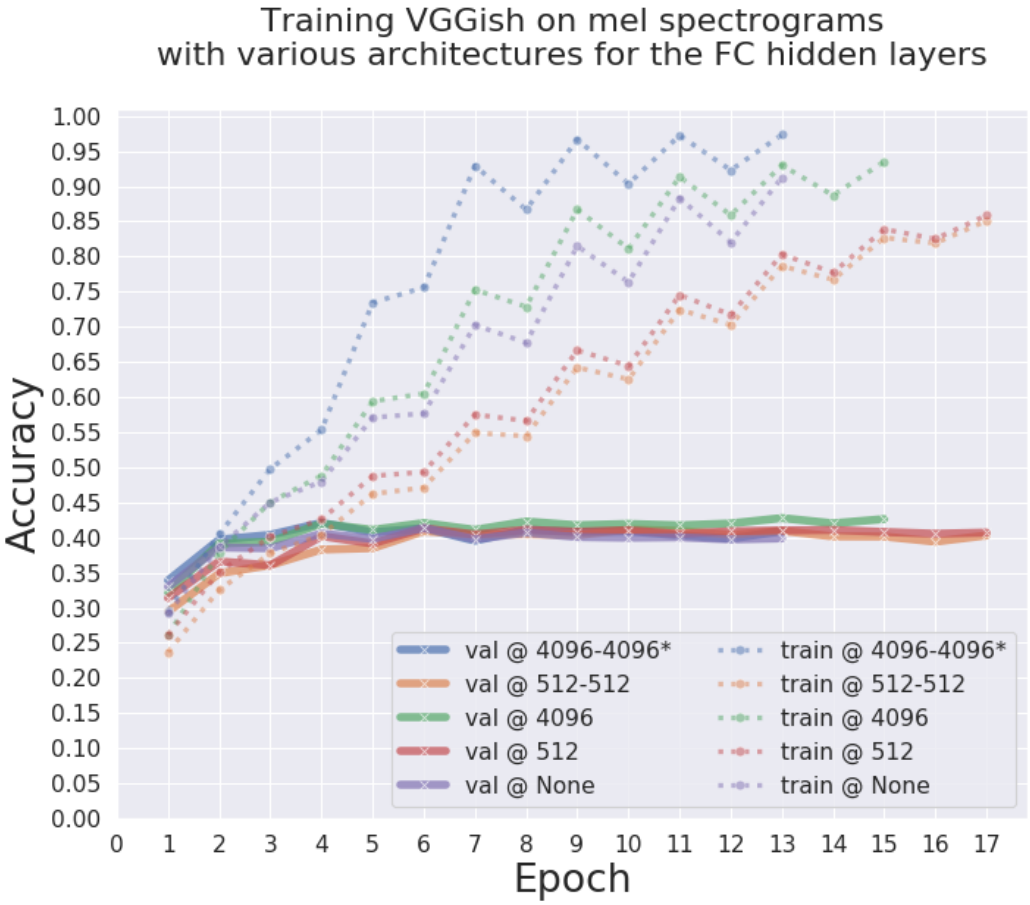


Figure 6.3: Training and validation accuracies for various amounts of nodes in the Fully Connected (FC) layers of VGGish, training on mel spectrograms. A dash indicates multiple layers (i.e. '512-512' indicates two layers with 512 nodes each). Note that for the FC architecture with 4096-4096* nodes, pretrained weights are available, and are used.

6.4 VGGISH: LEARNING RATE

Now that we have a decent architecture for training VGGish on our mel spectrograms, let us look at differences in training when using a variety of learning rates. Figure 6.4 shows numerous differences in the results between various learning rates. Commonly, a standard learning rate of 0.001 is used initially to see if the model is able to train on the training data. We can see that this yields the lowest training and validation accuracies for all tested learning rates at the end of the training run. We can also see a distinct pattern of using a cyclical learning rate: standard learning rates yield smooth accuracy curves, whereas cyclical learning rates yield large variations between epochs. This is because with cyclical learning rates, internally the learning rate differs per epoch in the range of $1 - 6\times$ the supplied learning rate.

This also means that the initial learning rate when using a cyclical learning rate should be lower than when using a standard learning rate, to get a similar average learning rate. Still, it is the internal variation of the learning rate when using a cyclical one that makes it converge more quickly than a standard one. In this figure, the highest validation accuracy (41%) is found with a cyclical learning rate of 0.0001, which also yields the highest training accuracy of 97%.

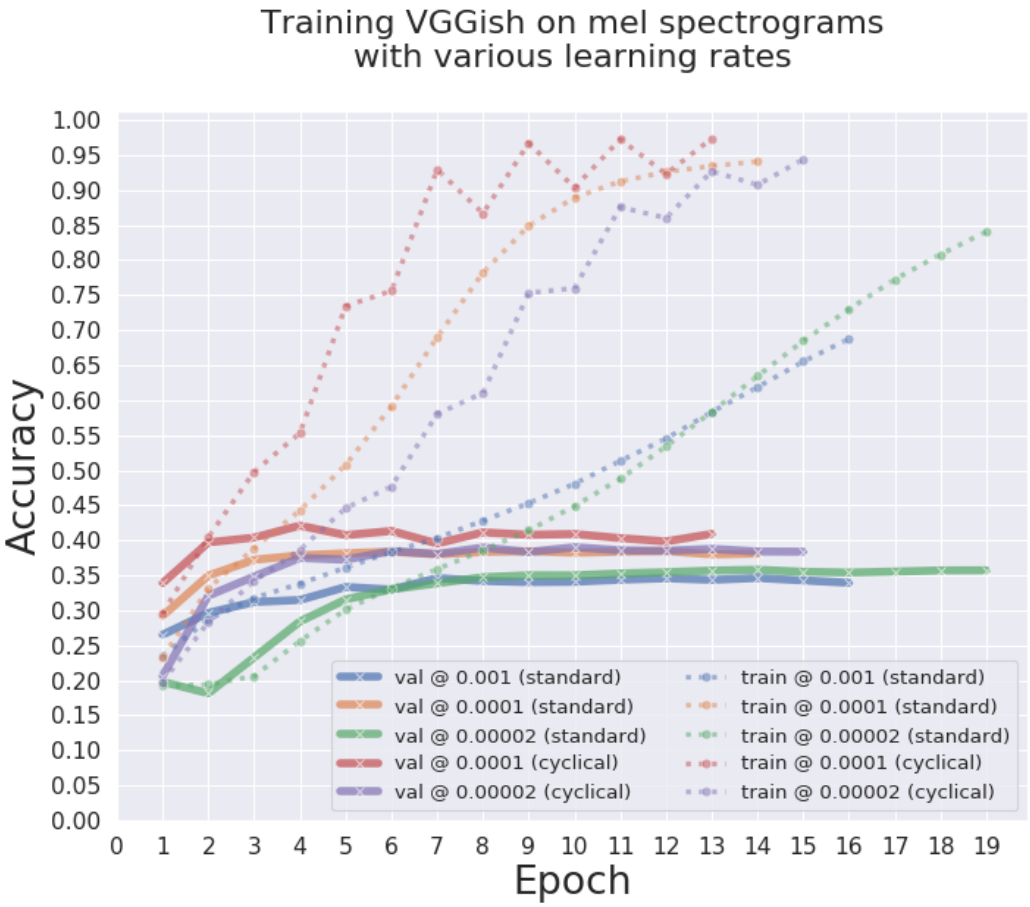


Figure 6.4: Training and validation accuracies for various learning rates, cyclical and non-cyclical (standard), training VGGish on mel spectrograms.

6.5 VGGISH: FINE-TUNED PARAMETERS

Starting with our default parameters for VGGish in Table 4.5, we found a number of improvements through hyperparameter optimization for training VGGish on the mel spectrograms computed from vocal fragments in the 6L5K Music Corpus. The best found model of VGGish can be trained using the fine-tuned parameters shown in Table 6.1. Chapter 7 examines the predictive power of our VGGish setup on the test dataset.

Parameter	Value
Mel spectrogram architecture:	6144] – [{FMs} – {FCs} – Embeddings] – [6
FC layers:	**[4096]**
Embeddings:	[128]
Activation functions:	linear] – [{relu} – {relu} – relu] – [softmax
Loss function:	Categorical Cross-Entropy
Regularization:	**30% FC Dropout**
Epochs:	100
Early stopping:	10
Batch size:	200
Optimizer:	Adam (β_1 = 0.9; β_2 = 0.999; ϵ = 1e-7)
Learning rate:	**0.0001, cyclical**
Weight initialization:	**Pretrained on mel spectrograms for Speech LID**

Table 6.1: Optimal parameters for VGGish by empirical evidence. Bold indicates alterations from the default parameters in Table 4.5. In the architectures described above, the open blocks indicate the input and output layer sizes; the closed block refers to the hidden layers. Furthermore, curly brackets indicate a multi-layer structure, where 'FMs' refer to VGGish' default Feature Maps and 'FCs' refer to the Fully Connected feedforward layers. 'Embeddings' refer to the single, penultimate layer.

7 TEST RESULTS

7.1 MODELS

In Chapter 5, we experimented and found which parameters work best for training the Deep Neural Network (DNN) on our vocal fragment training data. Furthermore, in Chapter 6 we found the optimal parameters for VGGish as well, given the data. Both architectures have been trained for more epochs, with `early stopping` set to 20 epochs instead of 10 which was used during hyperparameter optimization. The prolonged training runs are visualized in Figure 7.1.

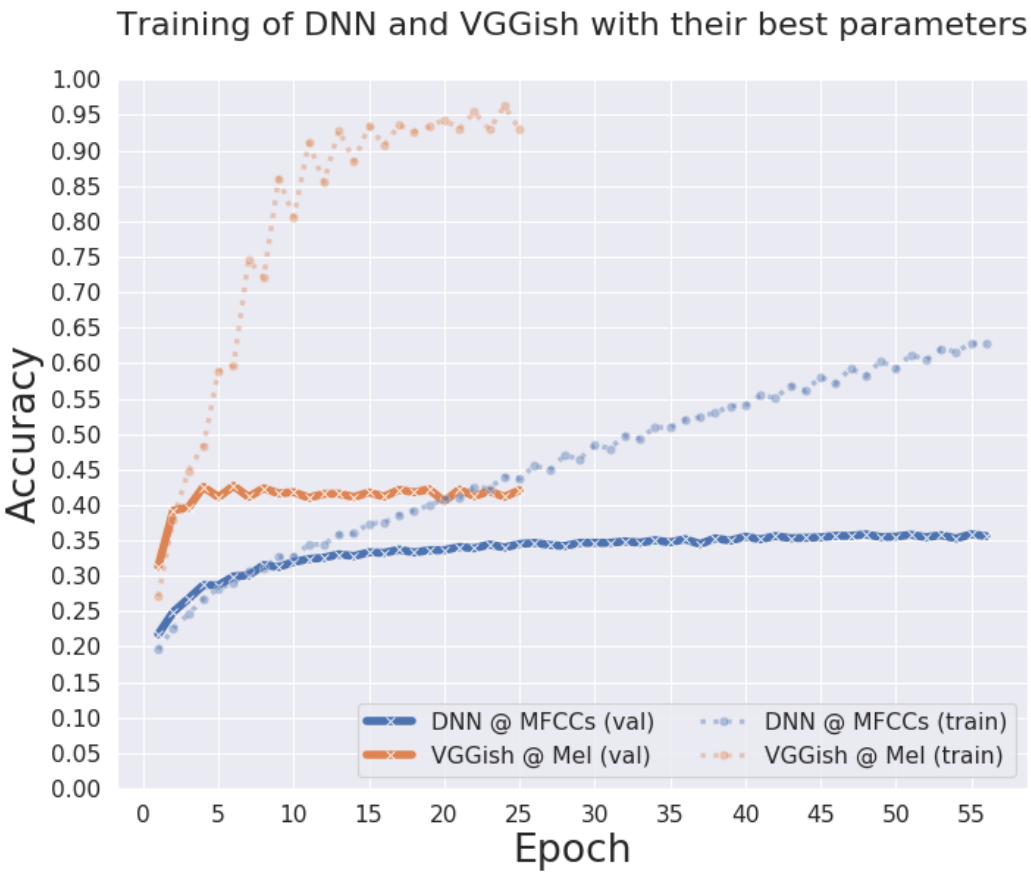


Figure 7.1: Training and validation accuracy during the training of the DNN and VGGish, each with the best tuned hyperparameters.

For both the DNN and VGGish, we can see the model overfitting on the training data. In both cases, the model has converged well before

the final epoch is reached, indicating that training for more epochs would likely not improve the model by much. The final results show that the DNN reaches a validation accuracy of 36% being trained on MFCCs, whereas VGGish reaches 42% being trained on mel spectrograms. The trained weights for our prediction models for the DNN and VGGish are taken at the very end of the prolonged training runs.

7.2 ENSEMBLE

Having two separately trained neural networks, the Deep Neural Network (DNN) and VGGish, we may introduce a third system: an Ensemble based on the two. For each vocal fragment in the test dataset, the computed MFCCs are classified by the DNN, and the computed mel spectrograms are classified by VGGish. This Ensemble relies on the concept of *"the whole is greater than the sum of its parts"*, as its prediction is based on the sum of the output classification probabilities of the two models which is then normalized. The idea of how this might work can be described with an example:

Suppose we have a Dutch vocal fragment, which the DNN classifies as English first ($p = 0.6$), as Dutch second ($p = 0.5$), and as Spanish third ($p = 0.1$). The same vocal fragment is classified by VGGish as Spanish first ($p = 0.7$), as Dutch second ($p = 0.6$), and as English third ($p = 0.2$). The DNN will label the vocal fragment incorrectly as English, whereas VGGish will classify the vocal fragment incorrectly as Spanish. The ensemble will classify the vocal fragment as follows: English: $p = \frac{0.6+0.2}{2} = 0.4$; Spanish: $p = \frac{0.7+0.1}{2} = 0.4$, Dutch: $p = \frac{0.5+0.6}{2} = 0.55$. As such, the Ensemble correctly classifies the example as Dutch, whereas each model that it consists of classifies the example incorrectly.

Whereas the Ensemble may sound promising, it may also reject a correct classification from one of the models in favour of a close second of both models. However, the Ensemble can never reject a correct classification made by both models on a vocal fragment, as the sum of this class' probability will inevitably also be the highest. There is no guarantee that this method improves performance over the existing models.

7.3 PREDICTIONS

Taking each of the trained models for the Deep Neural Network (DNN) and VGGish as described in Section 7.1, and the Ensemble explained in Section 7.2, we run these models on a separate test dataset from which we have computed the vocal fragment mel spectrograms and MFCCs (see Section 4.4.2). For each model and for each language, we

compute how often the class with the highest predicted classification probability matches the true language label of a vocal fragment, which is then converted to a percentage. The results are shown in Figure 7.2. For a more specific visualization of the predictions on vocal fragments of a single arbitrary track per language, refer to Appendix A.

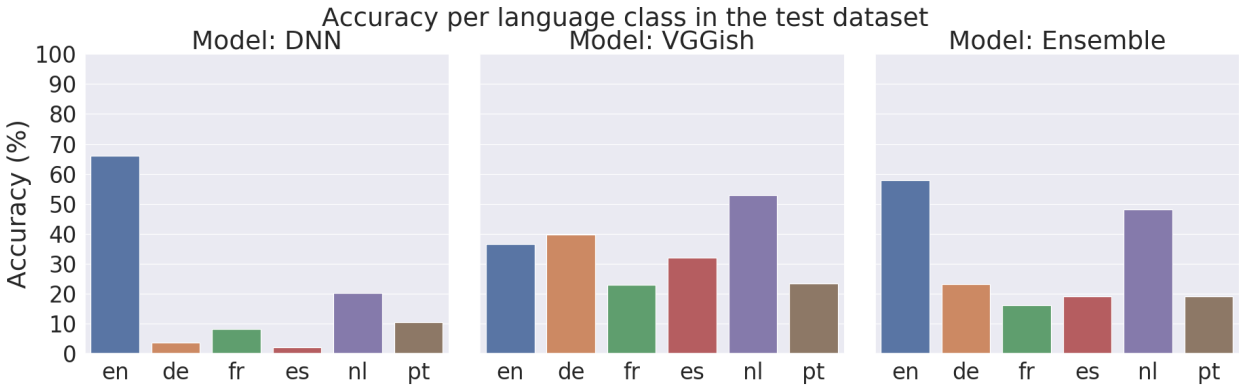


Figure 7.2: Accuracies on predicting the correct language class from the set of 3-second vocal fragments computed from the test set.

Looking at the accuracies per language, we see large differences between the DNN and VGGish. Whereas VGGish' classifications of the test set are somewhat similar between languages, the DNN mostly classifies English well, and all of the other languages obtain a rather low accuracy. The Ensemble method yields accuracies that appear to be roughly the average of the DNN and VGGish results combined, with a rather high accuracy for English, but with lower accuracies for the other languages than VGGish. All in all, the results for VGGish are comparable to the results on the validation dataset, whereas the DNN behaves somewhat unexpectedly, classifying mostly English. If the DNN were to classify most inputs as English, then it makes sense for a single class to obtain such a high accuracy. However, this would make it no better than random guessing. A more detailed comparison between the models can be made with confusion matrices, where one can compare which false languages have been confused for each of the true languages.

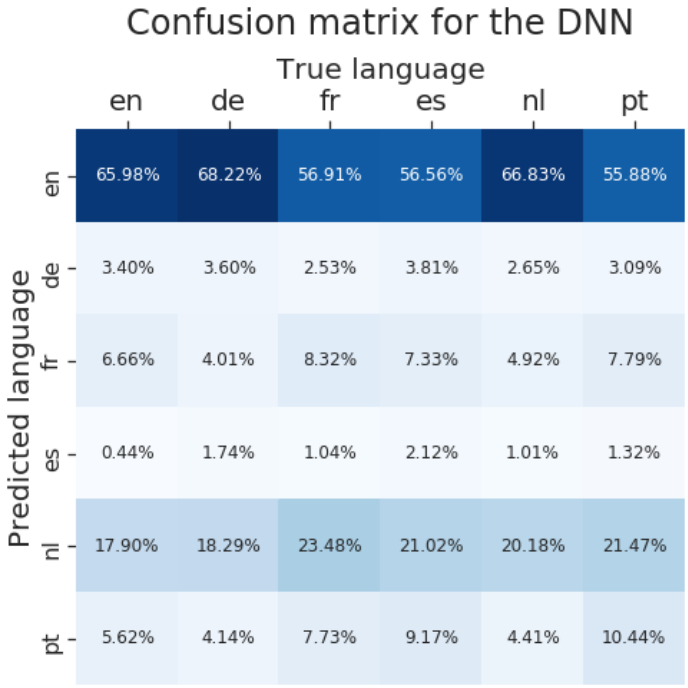


Figure 7.3: Confusion matrix showing how often the correct class in the test dataset was predicted by the DNN (in the diagonal), and how often each of the other classes was erroneously predicted instead.

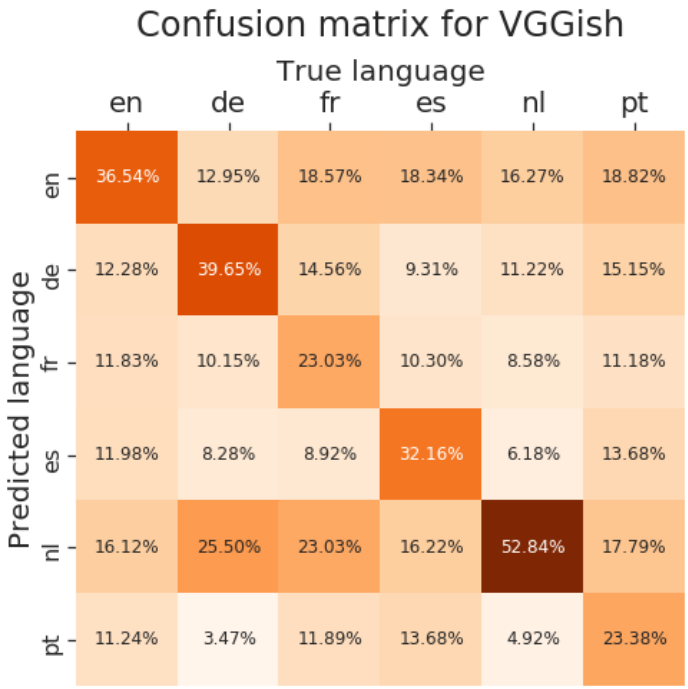


Figure 7.4: Confusion matrix showing how often the correct class in the test dataset was predicted by VGGish (in the diagonal), and how often each of the other classes was erroneously predicted instead.

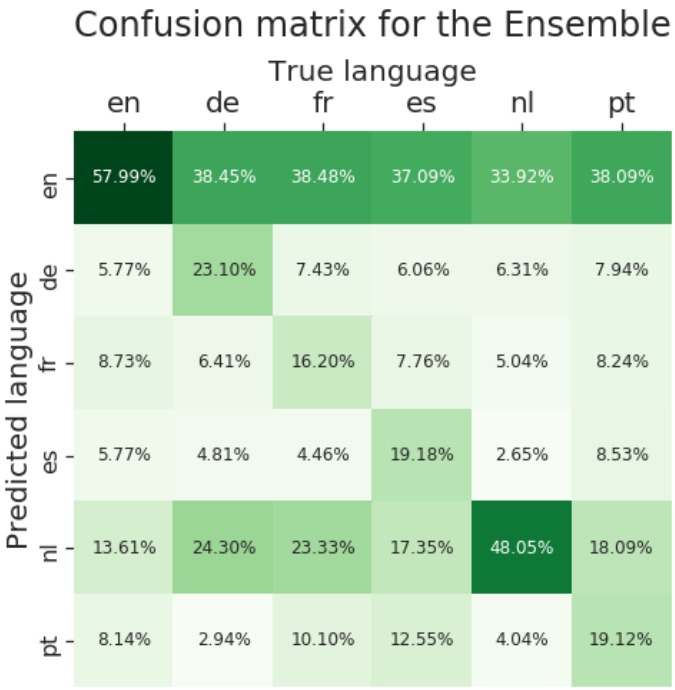


Figure 7.5: Confusion matrix showing how often the correct class in the test dataset was predicted by the Ensemble (in the diagonal), and how often each of the other classes was erroneously predicted instead.

In Figures 7.3, 7.4, and 7.5, the confusion matrices for each of the models are shown. In these confusion matrices, the top-left to bottom-right diagonal indicates the correct classification accuracies for each of the languages. These accuracies correspond with the accuracies found in Figure 7.2. For each column, the non-diagonal values indicate how often the column language was mistaken for another one.

As we can see for the DNN in Figure 7.3, there is no clear pattern to be found in the diagonal, which means that the model was not able to learn to distinguish the classes in the test dataset. Rather, each of the columns appear similar in values, indicating that the predictions of the DNN did not depend on which class the input data was from. It can be seen that mostly true Dutch and true German vocal fragments have been classified as English, more than the other languages, and the rest of the values for these two true languages also appear similar. If the DNN has learnt anything, it could be that Dutch and German songs sound somewhat similar. However, since it could not distinguish any of the languages from one another, and since Dutch and German songs are not particularly more confused by one another, it is unclear whether Dutch and German songs are actually similar, or whether this is found by chance.

The confusion matrix for VGGish in Figure 7.4 clearly shows a pattern of higher accuracies on the diagonal. This indicates that for each of the languages in the test dataset, VGGish has been able to more or less distinguish these tracks from other languages. The highest accuracy is obtained for Dutch vocal fragments, at 52.8%. Moreover, there are two cases outside of the diagonal where VGGish erroneously classified one language as another for over 20% of the cases. In both cases, the languages were erroneously classified as Dutch. This suggests that VGGish may have been overfitting on Dutch vocal fragments, given that it also has the highest accuracy of all languages. In one case, German vocal fragments are confused for Dutch, which may make sense as the languages can be seen as closely related, and the countries where these languages are spoken are adjacent. Furthermore, in the other case French vocal fragments are mistaken for Dutch, which could be explained by Belgium. In this country, both Dutch and French is spoken and sung, and Belgian music sung in Dutch may contain French accents or influences.

Lastly, the confusion matrix of Figure 7.5 is remarkable: it appears as roughly the mean of Figure 7.3 and 7.4. Although we had hoped that the Ensemble would lead to more interesting results, it appears as though the combination of the DNN and VGGish does not lead to any noteworthy improvements.

7.4 END RESULTS

In Section 1.2 we posed the question of which neural network architecture works best for Automatic Language Identification (LID) of sung music. We have fully tested two models – the Deep Neural Network (DNN) and VGGish – and created a third based on these two: the Ensemble. Table 7.1 shows the total number of correct predictions made per language, by each of the models. In accordance with Section 7.3, the DNN has the highest number of correct predictions for the English language, whereas VGGish scores better for all of the other languages. The Ensemble does worse than either of the two models for all languages.

Language	DNN	VGGish	Ensemble	*Total*
en	**446**	247	392	*1085*
de	27	**297**	173	*497*
fr	56	**155**	109	*320*
es	15	**228**	136	*379*
nl	160	**419**	381	*960*
pt	71	**159**	130	*360*
Total	*775*	*1505*	*1321*	*3601*

Table 7.1: The number of correct predictions for each language and for each of the systems. Bold indicates which system performs best on a language.

The overall accuracy on the test dataset for the DNN, VGGish, and the Ensemble can be seen in Figure 7.6. The DNN scores barely higher than chance, and the best model – VGGish – scores 35.2%. Although VGGish can distinguish each of the languages as shown in Section 7.3, a final accuracy score of 35.2% on an unseen dataset is not enough to put to use in a system that needs to be reliable. Comparing this to other attempts at LID in the literature, our results are not bad. For one, Schwenninger (2006) obtained 64% classifying only two languages: Mandarin and English [41]. Tsai et al. (2007) obtained 65% accuracy distinguishing three classes: Mandarin, English, and Japanese [49]. More notably however, Chandrasekhar et al. (2011) obtained 44.7% accuracy having trained a set of linear Support Vector Machines (SVMs) on 25 classes on a combination of audio features (19.6% accuracy on spectrograms; 26.1% on MFCCs) [6]. In comparison, our best results of VGGish seem to fit in with the existing literature; LID of vocals in music is a complex problem.

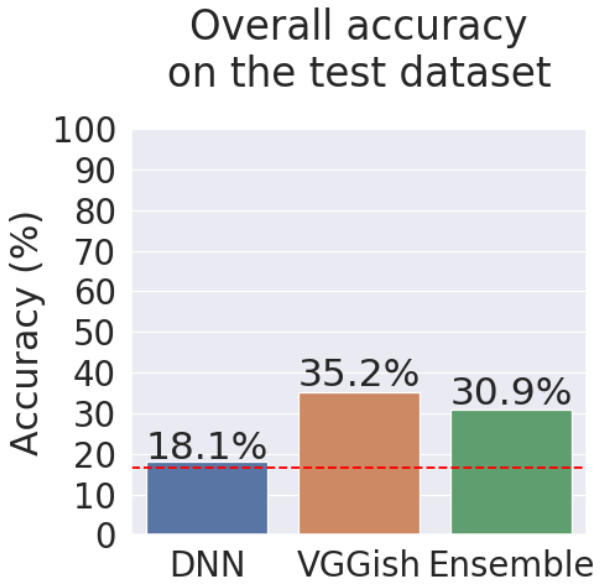


Figure 7.6: The overall accuracy per system on the test dataset. The red line indicates an accuracy no better than chance, at 16.7% given the six classes.

In Section 1.2 we asked ourselves how we can determine which model performs best on the task of LID. For this, we will focus again on the two neural networks and disregard the Ensemble, which does not appear to perform better than the averaged sum of its parts. As the DNN scores marginally better than chance, and VGGish is able to distinguish sung languages, we want to show whether VGGish is in fact able to predict the sung language significantly better than the DNN. For this, we need to perform a statistical test on the test results. Using Fisher's Exact Test, we can tell whether the proportions of one variable are significantly different from the proportions of another variable [13]. We first compute the contingency table for the DNN and VGGish with counts for the number of correct and incorrect predictions, the results of which can be seen in Table 7.3.

	Correct	**Incorrect**	**Total**
DNN	775	3505	4280
VGGish	1505	2775	4280
Total	2280	6280	8560

Table 7.2: Contingency table showing the total number of correct and incorrect classifications for the DNN and VGGish on the test dataset.

Fisher's Exact Test is calculated as:

$$p = \frac{(a+b)!\,(c+d)!\,(a+c)!\,(b+d)!}{a!\,b!\,c!\,d!\,n!} \tag{7.1}$$

with the variables for Fisher's Exact Test being defined as:

	Correct	**Incorrect**	**Total**
DNN	a	b	$a+b$
VGGish	c	d	$c+d$
Total	$a+c$	$b+d$	n

Table 7.3: Indication of variables for Fisher's Exact Test.

Calculating Fisher's Exact Test given the equation in 7.1 and the data in Table 7.3, we get:

$$p = \frac{4280!\,4280!\,2280!\,6280!}{775!\,3505!\,1505!\,2775!\,8560!} = 1.55 \times 10^{-72} \tag{7.2}$$

From the calculation in Equation 7.2, we can conclude that there is a significant difference in proportions between the number of correct predictions made by the DNN, and those made by VGGish ($p < 0.0001$). As such, VGGish scores significantly better than the DNN, which in turn is marginally better than chance.

8 DISCUSSION & CONCLUSION

8.1 DISCUSSION

This research has not fully gone according to plan. A large portion of work went into obtaining the 6L5K Music Corpus, and determining a representation of this data that can be used as input to various neural networks. Still, we argue that the quality of the dataset and the representation of the data are the main areas for improvement. Furthermore, we only tested two types of neural networks and added a third combination of the two. There are a number of improvements that can be attempted, unfortunately outside the scope of this research.

8.1.1 *The 6L5K Music Corpus*

Although a large portion of time went into obtaining the language-labelled dataset we have named the '6L5K Music Corpus', it is inherently flawed, to some extent. The playlist queries for obtaining language-labelled tracks off Spotify look fine at first glance (refer to Table 4.1), but given the fact that it is an automated method for obtaining labelled data which solely relies on Spotify's search engine and users creating the playlists, it is prone to causing errors. We have discussed this in Section 4.1: there are audio mismatches, causing tracks sung in a completely different language to appear at each label. Not only that, but Spotify's search engine also yields unexpected results. For instance, Chinese, Japanese, and even orchestra music would be obtained with the language playlist queries. Given that it is highly unlikely that tracks in one of our languages has a title containing *kanji*, we filtered out tracks containing these types of symbols. However, there is no trivial automatic filtering method to remove orchestra tracks, or more importantly, tracks sung in a different language. It is unclear exactly how many tracks have an incorrect label in the training data. If we knew, we would be able to filter these out. Unfortunately, manual filtering is outside the scope of this research, but it would be very beneficial for future work to ensure that the labels are correct. The danger in having incorrect language labels can be seen when tracks that belong to one language class have a label of another existing class. This makes it so that self-learning systems have a harder time distinguishing these two languages.

Besides label mismatching and overlap, there is more bias put into the selection of the data. Although we checked genre differences between

the language classes, we did not check for gender differences. If one of the classes contains significantly more music sung at a higher pitch, then this becomes an easy factor for a self-learning system to overfit on. Balancing a music dataset to eliminate most of the variations so that a system will only learn the sung music is a complex task. In fact, one might argue that it is okay for such a system to overfit on, for instance, specific instrumental sounds mainly produced in specific languages.

Moreover, the computed vocal fragments may also contain some bias. Each 3-second audio fragment is determined to be selected or omitted based on whether Essentia's Vocal Detector classifies the fragment as being 'vocal' (see Section 4.2). As such, if the Vocal Detector has any bias in the sense that it detects vocals more for certain languages or instruments, then this bias translates directly to our vocal fragment dataset. For this reason however, we decided to look at the total number of vocal fragments computed per language class in the training dataset (see Figure 4.8). We found that the counts differ, with the Dutch class containing most vocal fragments at 83,119 vocal fragments, and English the second to lowest amount, at 59,759 vocal fragments. Comparing this to the trained neural network predictions in Section 7.3, we would expect a bias towards Dutch given the quantity, which can actually be seen in the results of VGGish. Furthermore, the second highest amount of vocal fragments is for German, which also has the second highest accuracy for VGGish. It looks like VGGish has been overfitting on the sheer number of vocal fragments, making Dutch and German more likely predictions than the other classes. However, it does not explain why the DNN overfit almost entirely on English, having nearly the least amount of vocal fragments. What we can learn from this is that it is important to balance the final dataset that is used as the input to self-learning systems in terms of total counts. Although the number of tracks per language is balanced in terms of counts (see Figure 4.2), the transformation to vocal fragments is not, and the effects can be seen in the best trained model of VGGish.

8.1.2 *The Algorithms*

In this thesis, not many different algorithms have been tested. In the literature, various non-neural-network approaches to Automatic Language Identification (LID) have obtained decent results on speech, for instance Hidden Markov Models [34] and Logistic Regression [32]. Moreover, there are other types of complex neural network architectures that may yield better results as well, such as a Long Short-Term Memory (LSTM) network, a 1-dimensional Convolutional Neural Network (1D-CNN), or even Google's WaveNet altered as a discriminator [36]. In other words, there are a number of existing algorithms and

network architectures that are worth researching for the purpose of Automatic Language Identification of vocals.

As we discussed in Section 8.1.1, the input data may have been flawed as well. Rather than looking at the flaws in the method of obtaining the training data, we may also wonder if the representation of the data is good enough to use as input for neural networks. Typically, Deep Neural Networks (DNNs) do not handle image-like data very well, and the MFCCs that were used, although flattened to a 1-dimensional vector, can be seen as an image (see Figure 5.1). For this purpose, CNN architectures better handle the complexity of image data as these architectures are better suited for finding patterns in the images. Our results of VGGish, performing significantly better than the DNN (see Section 7.4), add to this argument. Mel spectrograms and MFCCs are widely used for audio classification tasks. However, converting vocal music to 3-second fragments, determining whether a fragment contains vocals, and turning the vocal fragments directly into mel spectrograms and MFCCs may not have been the best representation for learning to classify sung languages. Although we do try to make sure that the systems obtain music fragments that feature vocals, some of these fragments will only contain instrumentals (given that Essentia's Vocal Detector is imperfect), and all fragments will contain instrumentals and background noise overlapping with the vocals. Rather than directly computing mel spectrograms and MFCCs, reducing the instrumentals and background noise first, such that the opposite of a karaoke-version of a track is given, the audio and its mel spectrogram and MFCC translations become much more clearly focused on the vocals in music. Our systems had to find patterns in the instrumentals and vocals together, in order to classify in which language is sung, which makes it much harder to determine whether these systems really learnt which language the vocals are sung in, or whether they simply learned how likely certain instruments appear in each of the languages, although it has to be noted that the transformation to MFCCs already emphasizes the human-perceptible range and thus also vocals. We believe that improving the preprocessing steps may drastically increase the performance of the systems.

Finally, VGGish comes with pretrained weights for speech embeddings, which can be used for transfer learning. It was interesting to see whether weights that have been trained on speech would benefit classification of vocals as well. Since speech and vocals are produced by the same organ, the frequency range is similar, though vocals do tend to be higher and lower pitched than speech, essentially increasing the expected range. Interestingly, the pretrained weights for speech embeddings resulted in faster convergence of the network when used as a starting point for training further (see Figure 6.1). However, the

pretrained weights had been trained on mel spectrograms, but MFCC input also benefited from this weight initialization. Because of this, it appears that the pretrained weights for speech embeddings are simply a good starting point for VGGish in any case, rather than randomly initializing the weights. After all, these pretrained weights give the network non-random patterns to start from, which may not differ much between speech and vocal mel spectrograms, and even MFCCs.

8.2 CONCLUSION

8.2.1 *Findings*

Automatic Language Identification (LID) of vocals in music is not a trivial task. Existing literature mostly focuses on LID of speech, and the literature on LID of music is few and far between. In the literature, the accuracy scores of LID of music also shows that there is plenty of room for improvement (64% on two languages [41]; 65% on three languages [49]; 44.7% on 25 languages using audio-only features [6]). In this thesis, we explore the field of LID, create and describe a novel language-labelled music dataset – the 6L5K Music Corpus, compute 3-second fragments containing vocals, compute mel spectrograms and MFCCs given these vocal fragments, and train neural networks to classify the language thereof.

Initially, a Deep Neural Network (DNN) is designed and tuned for the task of LID on the vocal fragments. We find that the DNN works best with MFCC input, however it is shown to mainly overfit on a single language: English. Whereas the accuracy on the validation data is over 35%, the accuracy on a separate test dataset, which we transformed into vocal fragments similar to the training data, is no higher than 18.1%. Next, for a CNN architecture, we use VGGish on mel spectrogram input, using pretrained weights for speech embeddings. Training on the vocal fragment training dataset, we optimize VGGish' parameters and come to a validation accuracy of 41%. As we use this model on the unseen test data, the accuracy lowers to 35.2%. The confusion matrices show that the DNN is unable to distinguish the languages of the test data, whereas VGGish is able to distinguish the languages to some extent. We also merge the two models into an Ensemble, where the output is based on the combined output probabilities of both the DNN and VGGish. Although it is possible for this Ensemble technique to yield better results, in the end the Ensemble appeared to be no better than an average of the DNN and VGGish, obtaining an accuracy of 30.9% on the test data. The DNN performance on the test set is marginally better than chance, and using a Fisher's Exact Test we show that VGGish performs significantly better than the DNN for LID

of vocals in music.

8.2.2 *Research Questions*

In Section 1.2, we defined the main goal of this research. Three questions were posed, with which the main research question can be answered. We will go through these questions in the same order.

***Q1**: What type of **data** is best for training neural networks to be able to classify the language music is sung in?*

For starters, the more the data focuses on the vocals in music, the better the data can be used for classifying the sung language. Since no publicly available language-labelled dataset existed at the time of researching, we created the 6L5K Music Corpus from language-labelled music playlists. With this music data, we put emphasis on vocals by splitting each music track into 3-second fragments, and using a Vocal Detector to omit audio fragments where no vocals are detected, such that all 3-second fragments should contain vocals. Next, instead of using the raw audio waveform as input to our networks, we reduce the dimensionality by computing mel spectrograms, which more clearly show patterns for the frequencies that are used in the fragment. Lastly, for more emphasis on human-perceptible frequencies, we also compute the MFCCs, for a more compact representation of the vocal fragments. These are the methods that we used, but there are improvements to be made by for instance preprocessing the data such that only vocals are present in the audio, as our vocal fragments still contain background audio that may be irrelevant to the language classification.

***Q2**: Which neural network **architecture** works best for Automatic Language Identification of sung music?*

Given image-like data such as mel spectrograms and MFCCs, Deep Neural Networks (DNNs) tend to not be able to handle the complexity of finding patterns in this data. On the other hand, a CNN architecture is able to find patterns in this type of data. The best neural network architecture for Automatic Language Identification (LID) of sung music largely depends on the type of input data. In our findings, VGGish – a CNN architecture with Fully Connected layers – is able to distinguish six classes of vocal music based on mel spectrograms thereof. However, there may be novel state-of-the-art discriminators that can distinguish languages of vocals better, such as attention models, or Transformers [50].

***Q3**: How do we determine which system performs **better** on the task of Automatic Language Identification?*

In order to determine which system performs better, a statistical test is necessary. For neural network performance, a contingency table can be made with the number of correct and incorrect classifications per model, and Fisher's Exact Test can be applied to find whether there is a statistically significant difference in proportions between the models. Note however that if more than two models are compared, it is important to adjust the *p-value* for multiple tests, for instance using the Bonferroni correction. We find that our best trained VGGish model in fact performs significantly better on the test dataset than the DNN ($p < 0.0001$; see Section 7.4).

***Research Question**: How can we train a neural network to perform best on the task of Automatic Language Identification in vocal music?*

In this thesis, we have described our methods with which an accuracy of 35.2% on the task of LID of vocals in unseen music can be obtained. All of our methodology, results, and discussions indicate how a neural network can, and cannot, be trained to perform well on the task of Automatic Language Identification in vocal music. This thesis is also intended as a starting point for future research on this problem, as a quick-start in order to attempt novel approaches. Although our best model – VGGish – is able to distinguish six languages to some extent, it will likely not transfer well to the problem when more than six languages are present. As such, the overall problem of LID of vocal music remains open for further research, as there is room for improvement.

8.2.3 *Limitations & Future Work*

This work poses numerous insights for a very narrow field: Automatic Language Identification (LID) of vocals in music. The results are limited by relatively low accuracies on unseen data, as well as time constraints. The best trained network, VGGish, has been trained to distinguish six languages to some extent, but in real-world systems, this model should be no more than an informative addition. An accuracy of 35.2% can unfortunately not be reliably used. However, none of the LID systems of vocals in the literature can be reliably used, as either very few languages are used [41, 49], or the accuracy remains under 50% when more than six languages are used [6]. As such, there is a lot of potential for future work in this narrow field of LID to improve over the current findings. Specifically, taking care that preprocessing steps are put in place that emphasize the vocals in music and decrease the presence of instrumentals and background noise, perhaps researching

a better representation of vocals than mel spectrograms or MFCCs, and using different state-of-the-art discriminators [50], should make way for vastly improved performance with regard to LID of vocals in music.

BIBLIOGRAPHY

[1] Pierre Baldi and Peter Sadowski. "The dropout learning algorithm." In: *Artificial intelligence* 210 (2014), pp. 78–122.

[2] Christian Bartz, Tom Herold, Haojin Yang, and Christoph Meinel. "Language identification using deep convolutional recurrent neural networks." In: *International conference on neural information processing*. Springer. 2017, pp. 880–889.

[3] Dmitry Bogdanov, Nicolas Wack, Emilia Gómez Gutiérrez, Sankalp Gulati, Herrera Boyer, Oscar Mayor, Gerard Roma Trepat, Justin Salamon, José Ricardo Zapata González, Xavier Serra, et al. "Essentia: An audio analysis library for music information retrieval." In: *Britto A, Gouyon F, Dixon S, editors. 14th Conference of the International Society for Music Information Retrieval (ISMIR); 2013 Nov 4-8; Curitiba, Brazil.[place unknown]: ISMIR; 2013. p. 493-8.* International Society for Music Information Retrieval (ISMIR). 2013.

[4] Ronald Newbold Bracewell and Ronald N Bracewell. *The Fourier transform and its applications*. Vol. 31999. McGraw-Hill New York, 1986.

[5] Niko Brümmer, Albert Strasheim, Valiantsina Hubeika, Pavel Matějka, Lukáš Burget, and Ondřej Glembek. "Discriminative acoustic language recognition via channel-compensated GMM statistics." In: *Tenth Annual Conference of the International Speech Communication Association*. 2009.

[6] Vijay Chandrasekhar, Mehmet Emre Sargin, and David A Ross. "Automatic language identification in music videos with low level audio and visual features." In: *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2011, pp. 5724–5727.

[7] Dorothy L Cheney and Robert M Seyfarth. "Why animals don't have language." In: *Tanner lectures on human values* 19 (1998), pp. 173–210.

[8] Ronald A Cole, Jon WT Inouye, Yeshwant K Muthusamy, and Murali Gopalakrishnan. "Language identification with neural networks: a feasibility study." In: *Conference Proceeding IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*. IEEE. 1989, pp. 525–529.

[9] Nicholas J Conard, Maria Malina, and Susanne C Münzel. "New flutes document the earliest musical tradition in southwestern Germany." In: *Nature* 460.7256 (2009), pp. 737–740.

[10] George Cybenko. "Approximation by superpositions of a sigmoidal function." In: *Mathematics of control, signals and systems* 2.4 (1989), pp. 303–314.

[11] Steven Davis and Paul Mermelstein. "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences." In: *IEEE transactions on acoustics, speech, and signal processing* 28.4 (1980), pp. 357–366.

[12] Michaël Defferrard, Kirell Benzi, Pierre Vandergheynst, and Xavier Bresson. "FMA: A Dataset for Music Analysis." In: *18th International Society for Music Information Retrieval Conference*. 2017. URL: https://arxiv.org/abs/1612.01840.

[13] Ronald A Fisher. "On the interpretation of χ 2 from contingency tables, and the calculation of P." In: *Journal of the Royal Statistical Society* 85.1 (1922), pp. 87–94.

[14] Jort F. Gemmeke, Daniel P. W. Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R. Channing Moore, Manoj Plakal, and Marvin Ritter. "Audio Set: An ontology and human-labeled dataset for audio events." In: *Proc. IEEE ICASSP 2017*. New Orleans, LA, 2017.

[15] Jort F. Gemmeke, Daniel P. W. Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R. Channing Moore, Manoj Plakal, and Marvin Ritter. "Audio Set: An ontology and human-labeled dataset for audio events." In: *Proc. IEEE ICASSP 2017*. New Orleans, LA, 2017.

[16] Beat Gfeller, Ruiqi Guo, Kevin Kilgour, Sanjiv Kumar, James Lyon, Julian Odell, Marvin Ritter, Dominik Roblek, Matthew Sharifi, Mihajlo Velimirović, et al. "Now Playing: Continuous low-power music recognition." In: *arXiv preprint arXiv:1711.10958* (2017).

[17] Javier Gonzalez-Dominguez, Ignacio Lopez-Moreno, Pedro J Moreno, and Joaquin Gonzalez-Rodriguez. "Frame-by-frame language identification in short utterances using deep neural networks." In: *Neural Networks* 64 (2015), pp. 49–58.

[18] Karlheinz Gröchenig. *Foundations of time-frequency analysis*. Springer Science & Business Media, 2001.

[19] Shawn Hershey et al. "CNN Architectures for Large-Scale Audio Classification." In: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2017. URL: https://arxiv.org/abs/1609.09430.

[20] Kurt Hornik. "Approximation capabilities of multilayer feedforward networks." In: *Neural networks* 4.2 (1991), pp. 251–257.

[21] David Huron. "Is music an evolutionary adaptation?" In: *Annals of the New York Academy of sciences* 930.1 (2001), pp. 43–61.

[22] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." In: *arXiv preprint arXiv:1502.03167* (2015).

[23] Wonkyung Jung, Daejin Jung, Sunjung Lee, Wonjong Rhee, Jung Ho Ahn, et al. "Restructuring batch normalization to accelerate CNN training." In: *arXiv preprint arXiv:1807.01702* (2018).

[24] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks." In: *Advances in neural information processing systems* 25 (2012), pp. 1097–1105.

[25] Lori F Lamel and Jean-Luc Gauvain. "Language identification using phone-based acoustic likelihoods." In: *Proceedings of ICASSP'94. IEEE International Conference on Acoustics, Speech and Signal Processing*. Vol. 1. IEEE. 1994, pp. I–293.

[26] Quoc V Le. "Building high-level features using large scale unsupervised learning." In: *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE. 2013, pp. 8595–8598.

[27] K Li and T Edwards. "Statistical models for automatic language identification." In: *ICASSP'80. IEEE International Conference on Acoustics, Speech, and Signal Processing*. Vol. 5. IEEE. 1980, pp. 884–887.

[28] Beth Logan et al. "Mel frequency cepstral coefficients for music modeling." In: *Ismir*. Vol. 270. Citeseer. 2000, pp. 1–11.

[29] Ignacio Lopez-Moreno, Javier Gonzalez-Dominguez, Oldrich Plchot, David Martinez, Joaquin Gonzalez-Rodriguez, and Pedro Moreno. "Automatic language identification using deep neural networks." In: *2014 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE. 2014, pp. 5337–5341.

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

[31] Lauras Martin. ""Eskimo words for snow": A case study in the genesis and decay of an anthropological example." In: *American anthropologist* 88.2 (1986), pp. 418–423.

[32] David Martinez, Oldřich Plchot, Lukáš Burget, Ondřej Glembek, and Pavel Matějka. "Language recognition in ivectors space." In: *Twelfth annual conference of the international speech communication association*. 2011.

[33] Mahnoosh Mehrabani and John HL Hansen. "Language identification for singing." In: *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2011, pp. 4408–4411.

[34] Yeshwant K Muthusamy, Etienne Barnard, and Ronald A Cole. "Reviewing automatic language identification." In: *IEEE Signal Processing Magazine* 11.4 (1994), pp. 33–41.

[35] Yeshwant Muthusamy, Kay Berkling, Takayuki Arai, Ronald Cole, and Etienne Barnard. "A comparison of approaches to automatic language identification using telephone speech." In: *Third European Conference on Speech Communication and Technology*. 1993.

[36] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. "Wavenet: A generative model for raw audio." In: *arXiv preprint arXiv:1609.03499* (2016).

[37] Kyubyong Park and Thomas Mulc. "CSS10: A Collection of Single Speaker Speech Datasets for 10 Languages." In: *arXiv preprint arXiv:1903.11269* (2019).

[38] Josef Psutka, Ludek Müller, and Josef V Psutka. "Comparison of MFCC and PLP parameterizations in the speaker independent continuous speech recognition task." In: *Seventh European Conference on Speech Communication and Technology*. 2001.

[39] Shauna Revay and Matthew Teschke. "Multiclass language identification using deep learning on spectral images of audio signals." In: *arXiv preprint arXiv:1905.04348* (2019).

[40] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. "Learning representations by back-propagating errors." In: *nature* 323.6088 (1986), pp. 533–536.

[41] Jochen Schwenninger, Raymond Brueckner, Daniel Willett, and Marcus E Hennecke. "Language Identification in Vocal Music." In: *ISMIR*. 2006, pp. 377–379.

[42] Garima Sharma, Kartikeyan Umapathy, and Sridhar Krishnan. "Trends in audio signal feature extraction methods." In: *Applied Acoustics* 158 (2020), p. 107020.

[43] Shikhar Shukla, Govind Mittal, et al. "Spoken language identification using convnets." In: *European Conference on Ambient Intelligence*. Springer. 2019, pp. 252–265.

[44] Leslie N Smith. "Cyclical learning rates for training neural networks." In: *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE. 2017, pp. 464–472.

[45] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Dropout: a simple way to prevent neural networks from overfitting." In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[46] Stanley Smith Stevens, John Volkmann, and Edwin Broomell Newman. "A scale for the measurement of the psychological magnitude pitch." In: *The journal of the acoustical society of america* 8.3 (1937), pp. 185–190.

[47] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. "Deepface: Closing the gap to human-level performance in face verification." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 1701–1708.

[48] Inc. TopCoder. *Problem: SpokenLanguages2*. 2010. URL: https://community.topcoder.com/longcontest/?module=ViewProblemStatement\&rd=16555\&compid=49304.

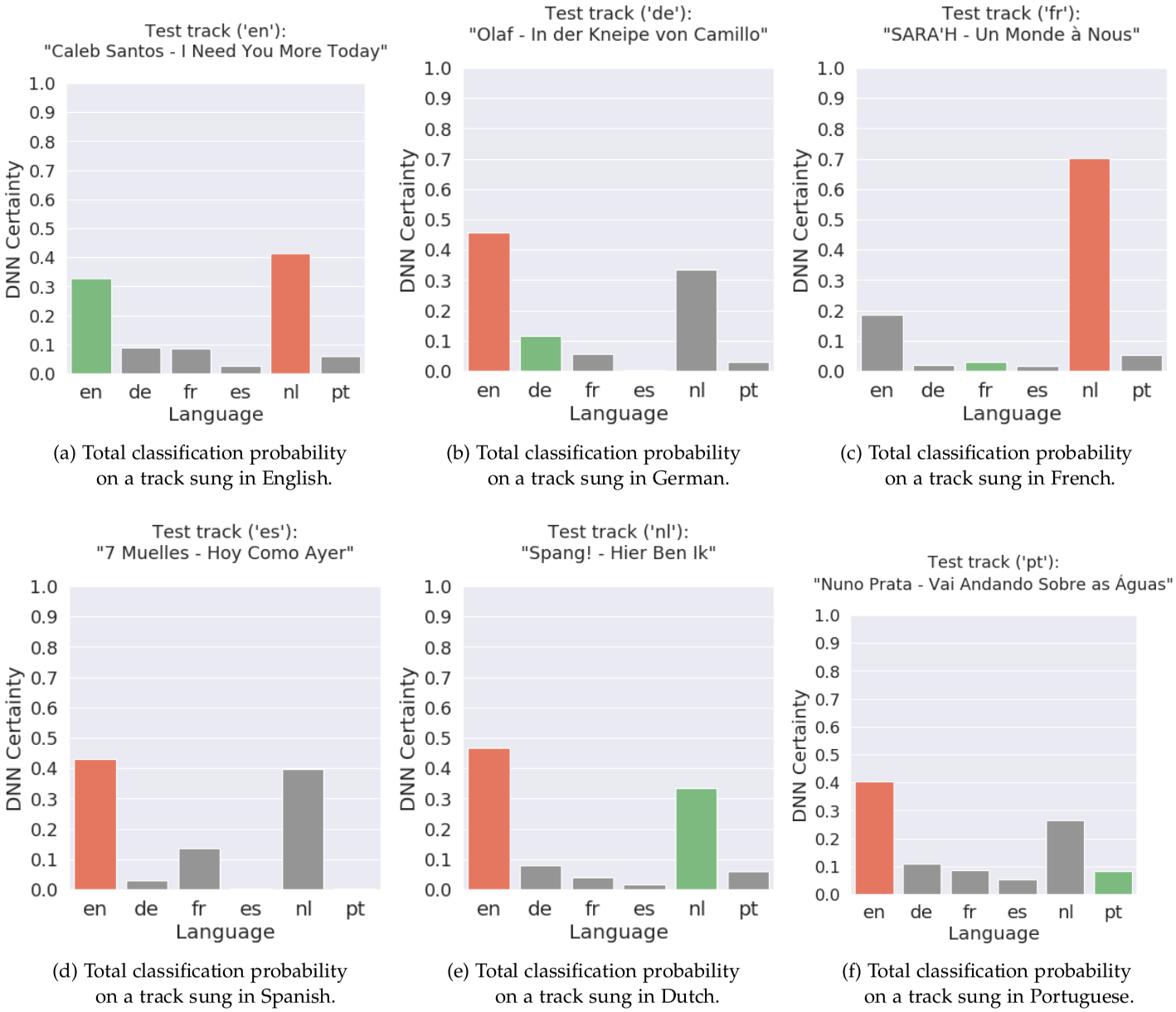
[49] Wei-Ho Tsai and Hsin-Min Wang. "Automatic identification of the sung language in popular music recordings." In: *Journal of New Music Research* 36.2 (2007), pp. 105–114.

[50] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. "Attention is all you need." In: *arXiv preprint arXiv:1706.03762* (2017).

[51] Haibing Wu and Xiaodong Gu. "Towards dropout training for convolutional neural networks." In: *Neural Networks* 71 (2015), pp. 1–10.

[52] Yuxiao Zhou. *Singing Voice Separation*. https://github.com/CalciferZh/Singing-Voice-Separation. 2018.

[53] Marc A Zissman. "Comparison of four approaches to automatic language identification of telephone speech." In: *IEEE Transactions on speech and audio processing* 4.1 (1996), p. 31.



(a) Total classification probability on a track sung in English.

(b) Total classification probability on a track sung in German.

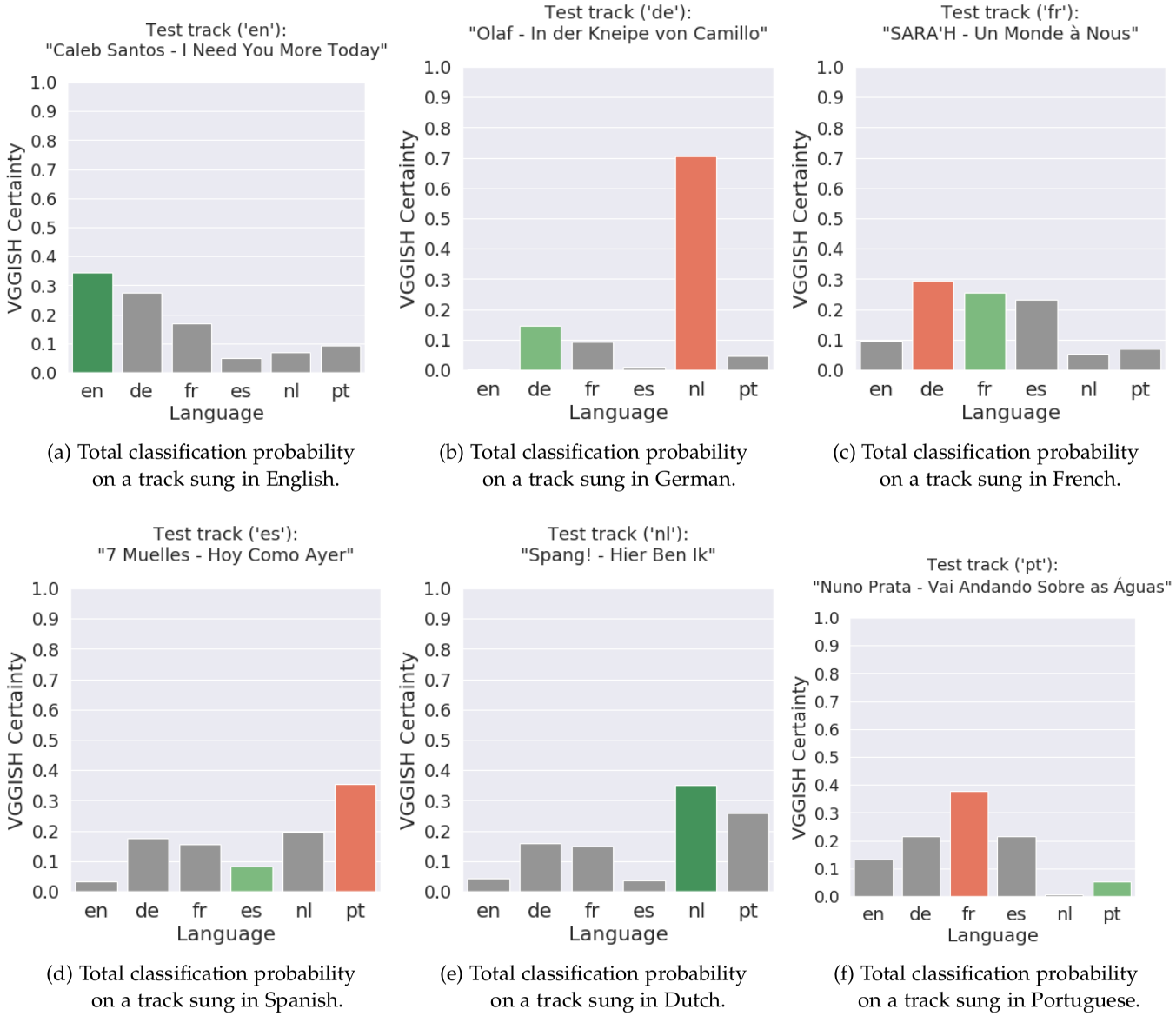
(c) Total classification probability on a track sung in French.

(d) Total classification probability on a track sung in Spanish.

(e) Total classification probability on a track sung in Dutch.

(f) Total classification probability on a track sung in Portuguese.

Figure A.1: Example predictions produced by our best trained Deep Neural Network (DNN) model on an arbitrary music track per language in the test dataset. All vocal fragments (see Section 4.2; Table A.1) of the track were used as input, and the output predictions were summed and normalized. Each figure shows the summed probabilities. Red indicates an erroneous prediction; green indicates the true language of the track.



(a) Total classification probability on a track sung in English.

(b) Total classification probability on a track sung in German.

(c) Total classification probability on a track sung in French.

(d) Total classification probability on a track sung in Spanish.

(e) Total classification probability on a track sung in Dutch.

(f) Total classification probability on a track sung in Portuguese.

Figure A.2: Example predictions produced by our best trained VGGish model on an arbitrary music track per language in the test dataset. All vocal fragments (see Section 4.2; Table A.1) of the track were used as input, and the output predictions were summed and normalized. Each figure shows the summed probabilities. Red indicates an erroneous prediction; green indicates the true language of the track.

Language	Artist - Title	# F	# VF
en	Caleb Santos - I Need You More Today	78	**11**
de	Olaf - In Der Kneipe Von Camillo	65	**20**
fr	SARA'H - Un Monde à Nous	74	**21**
es	7 Muelles - Hoy Como Ayer	92	**18**
nl	Spang! - Hier Ben Ik	69	**33**
pt	Nuno Prata - Vai Andando Sobre as Águas	65	**5**

Table A.1: An arbitrarily-picked track per language, the total number of 3-second fragments in the track ('# F'), and the number of fragments that contain vocals ('# VF') with which the results of Figure A.1 and Figure A.2 were obtained.