Master's Thesis

Computational Linguistics

Acoustic Dialect Recognition Using Deep Learning

Author:
Ryan Callihan

Supervisors:
Çağrı Çöltekin
Kurt Eberle
Fabian Tomaschek

Seminar für Sprachwissenschaft
Eberhard-Karls-Universität Tübingen

September 2018

Hiermit versichere ich, dass ich die Arbeit selbständig verfasst, keine anderen als die angegebenen Hilfsmittel und Quellen benutzt, alle wörtlich oder sinngemäß aus anderen Werken übernommenen Aussagen als solche gekennzeichnet habe und dass die Arbeit weder vollständig noch in wesentlichen Teilen Gegenstand eines anderen Prüfungsverfahrens gewesen ist und dass die Arbeit weder vollständig noch in wesentlichen Teilen bereits veröffentlicht wurde sowie dass das in Dateiform eingereichte Exemplar mit den eingereichten gebundenen Exemplaren übereinstimmt.

I hereby declare that this paper is the result of my own independent scholarly work. I have acknowledged all the other authors' ideas and referenced direct quotations from their work (in the form of books, articles, essays, dissertations, and on the internet). No material other than that listed has been used.

Tübingen, September 26, 2018

Ryan Callihan

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Philosophische Fakultät

Name:

Vorname:

Matrikel-Nummer:

Adresse:

Hiermit versichere ich, die Arbeit mit dem Titel:

_____

im Rahmen der Lehrveranstaltung _____

im Sommer-/Wintersemester _____ bei _____

selbständig und nur mit den in der Arbeit angegebenen Hilfsmitteln verfasst zu haben.

Mir ist bekannt, dass ich alle schriftlichen Arbeiten, die ich im Verlauf meines Studiums als Studien- oder Prüfungsleistung einreiche, selbständig verfassen muss. Zitate sowie der Gebrauch von fremden Quellen und Hilfsmitteln müssen nach den Regeln wissenschaftlicher Dokumentation von mir eindeutig gekennzeichnet werden. Ich darf fremde Texte oder Textpassagen (auch aus dem Internet) nicht als meine eigenen ausgeben.

Ein Verstoß gegen diese Grundregeln wissenschaftlichen Arbeitens gilt als Täuschungs- bzw. Betrugsversuch und zieht entsprechende Konsequenzen nach sich. In jedem Fall wird die Leistung mit **„nicht ausreichend“ (5,0)** bewertet. In schwerwiegenden Fällen kann der Prüfungsausschuss den Kandidaten/die Kandidatin von der Erbringung weiterer Prüfungsleistungen ausschließen; vgl. hierzu die Prüfungsordnungen für die Bachelor-, Master-, Lehramts- bzw. Magisterstudiengänge.

Datum: _____ Unterschrift: _____

Contents

Abstract

Dialect Identification (DID) is a well defined task with text data and models using both text and audio data have been shown to work especially well. However, the task of DID when using only audio signals is a bit more complicated and, as such, is much more difficult to achieve good and reliable results. In this thesis, the task of DID using only audio signals was investigated and different methods of feature extraction and machine learning were experimented with. The audio feature extraction methods explored were: Mel-spectrograms, Mel-frequency cepstral coefficients (MFCC) and Delta-Spectral Cepstral Coefficients (ΔSCC). Each method is widely used in the field of speech recognition but their effects on dialect recognition were a bit different. The machine learning techniques used were Recurrent Neural Networks (RNN), Convolutional Neural Networks (CNN), and a hybrid combination of the two. Two corpora were used: one for Arabic, and one for English. The methods listed above were tried on both corpora and the results were analysed.

List of Figures

List of Tables

List of Abbreviations

Δ	Delta Cepstral Coefficient
ΔΔ	Delta Delta Cepstral Coefficient
ΔSCC	Delta-Spectrum Cepstral Coefficient
ASR	Automatic Speech Recognition
CNN	Convolutional Neural Network
DCT	Discrete Cosine Transform
DID	Dialect Identification
DSL	Discriminating Similar Languages
EGY	Egyptian Dialect
FFNN	Feed-Forward Neural Network
FFT	Fast Fourier Transform
FRED-S	Freiburg Corpus of English Dialects - Sampler
GLF	Gulf Dialect
GMM	Gaussian Mixture Model
GRU	Gated Recurrent Unit
LAV	Lavantine Dialect
LID	Language Identification
LSTM	Long Short-Term Memory
MFCC	Mel-Frequency Cepstral Coefficient
MGB-3	Multi Genre Broadcasting 3 Corpus
MSA	Modern Standard Arabic
NOR	Northern African Dialect
QCRI	Qatar Computing Research Institute
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
WER	Word Error Rate

1 Introduction

Automatic speech recognition (ASR) has recently increased in popularity due to it's prevalence in all aspects of technology. Integration of speech recognition is found in everything from mobile devices, customer service platforms, or even everyday kitchen appliances. But even with it's popularity, it leaves something to be desired. In pristine testing situations it may work quite well, but when background noise, idiolects, and dialects are thrown into the mix, the efficacy decreases. It does not matter the level of negative effect, every time speech recognition fails in some way, it is frustrating. Therefore, taking into account the way people speech and the situations they find themselves in is the natural progression of speech technology.

In the past several years, *Language Identification* (LID) has become increasingly more important due to things like an increase unstructured data or ASR. Text based language models already work exceptionally well. Especially with the expansion of machine learning capabilities and the wide variety of linguistic features.

Audio based acoustic models, on the other hand, are usually a bit more tricky to deal with. Especially when languages are extremely similar, like Czech and Slovak for example. Unlike text based models, there are far fewer features which audio based methods can use like part-of-speech or syntax and there is usually a lot of correlation in the data. Recently, the usage of *i-vectors* and deep neural networks has made this task a bit more manageable (Dehak et al., 2011).

Dialect Identification (DID), a subset of LID, is an even more difficult task. There is a large acoustic diversity between different languages which makes LID a relatively manageable task, whereas DID deals with variations of the same language. As a result, the variations are usually much smaller. The distance between dialects varies greatly across languages and regions and many are entirely mutually distinguishable such as Standard American English and Received Pronunciation in England or Mexican and Puerto Rican Spanish. However, many dialects are not universally understood as is the case with many Arabic or German dialects.

This variation can quickly become a problem for computational systems. The acoustic differences between dialects have been shown to have negative effects on ASR systems. These dialectal differences can sometimes be humorous, but are more often frustrating when interacting with technology. These frustrations dissuade many from even attempting to use speech technology.

Due to it's large effect on many systems, it might be surprising to note that the task of DID from audio data is relatively unexplored. There are several reasons why this may be the case such as the lack of dialectal datasets, the

more broad focus on LID, or the fact that it is just quite challenging.

In this thesis, I will be testing methods of DID. I will use deep learning architectures to do so, namely *Recurrent Neural Networks*, *Convolutional Neural Networks*, as well as hybrid of the two. Furthermore, I will be testing different methods of acoustic data processing techniques. Finally, I will be analysing the differences between the models and acoustic processing techniques and the effect the dialects have on them.

The structure of this thesis is as follows: in Section 2, I will describe the two datasets used in this study. Section 3 will outline the methodology of this study including audio feature extraction in Section 3.1 and machine learning methods in Section 3.2. Section 4 will show the results and outcome of the models created. And finally, in Section 6 I will discuss the results and findings of this study.

2 Dataset

For the task of DID, it was crucial to obtain enough dialectal recordings. Due to the narrow nature of this field, this was not the easiest task.

First, the data had to be publicly available for research, so using general recordings from radio, film, or television were not possible.

Second, there had to be enough data to work with for deep learning. A dataset of a couple thousand items can work well for traditional machine learning approaches but not so well for deep learning.

Third, the data had to be methodologically sound and verified. Using crowd sourced data can be a great tool, but unless it has been backed by theory and tested for accuracy, it would not fit my purposes.

Fourth, at least one dataset should be able to serve as a baseline. And lastly, although not a requirement, it should be a clean and clear recording.

There were two corpora I found which fit most of these requirements: The *Multi Genre Broadcasting 3* (MGB-3) corpus of Arabic dialects (Ali et al., 2016) and the *Freiburg Corpus of English Dialects - Sampler* (FRED-S) of British English dialects (Szmrecsanyi and Hernández, 2007). In the next two sections, I will describe these corpora, explain what makes them suitable for the task of DID, and the expectations for each.

2.1 Multi Genre Broadcasting - 3 Corpus

The MGB-3 corpus was put together by the *Qatar Computing Research Institute* (QCRI) and consists of broadcast news in multiple Arabic dialects, mainly collected from *Aljazeera*. It was sampled at 16kHz and segmented into

a range of durations in order to avoid speaker overlap, mixtures of dialects, background noise, and music (Bahari et al., 2014)

The MGB-3 corpus contains Arabic dialects: *Egyptian* (EGY), *Lavantine* (LAV), *North African* or *Maghrebi* (NOR), and *Gulf* or *Arabian Peninsula* (GLF). Additionally, it contains *Modern Standard Arabic* (MSA). The number of hours represented for each dialect can be seen in Table 2.1.

Data	EGY	GLF	LAV	NOR	MSA
Hours	13	9.5	11	9	10

Table 2.1: Number of hours of speech for each dialect in MGB-3

Expectations for MGB-3 The MGB-3 is an excellent resource and is very well put together. It is theoretically backed and easy to use. And not only is the audio data publicly available, there are a variety of annotations and transcriptions available, as well as *i-vectors* [1].

However, as Ali et al. (2016) points out, Arabic dialects are historically related, but not necessarily synchronically and are not mutually intelligible. Therefore, this corpus may but regarded as ideal for the task of *Discriminating Similar Languages* (DSL). In Table 2.2, an example taken from Ali et al. (2016) can be seen of the difference between dialects for two simple phrases.

EGY	GLF	LAV	NOR	MSA	Translation
AzAYk	A$lwnk	kyfk / A$lwnk	kyf HAlk	wA$ rAk	How are you?
Ant fyn	wynk	wynk	Ayn Ant	wyn rAk	Where are you?

Table 2.2: Arabic lexical dialectal examples in Buckwalter format

However, even though the lexical variation between dialects can be quite high, this paper will only be dealing with the acoustic features. Therefore, this corpus can still be used in the task of DID.

[1] The MGB-3 can be found here: `http://alt.qcri.org/resources/ArabicDialectIDCorpus/`

2.2 Freiburg Corpus of English Dialects

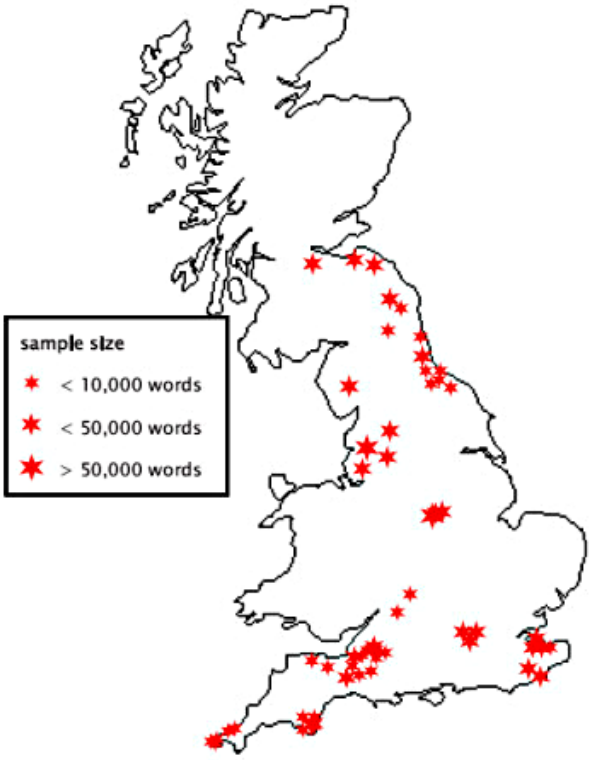


Figure 2.1: Dialect dispersal of FRED-S

FRED-S was compiled by the research group 'English Dialect Syntax from a Typological Perspective' in the English department of Universität Freiburg and is a subset of the much larger FRED corpus, which is not subject to copyright restrictions (Szmrecsanyi and Hernández, 2007). The full version is only available to researchers located at Universität Freiburg. Even though the FRED-S is a subset, it still is a sizeable collection of recordings of the 'traditional' varieties of British English spoken in the British Isles in the second half of the 20th century.

It includes 123 hours of recorded speech, 121 interviews, 144 dialect speakers, 57 different locations, 18 different counties, and 5 major dialect areas which include: the Southwest of England (SW), the Southeast of England (SE), the Midlands (MID), the North of England (N), and the Scottish Lowlands (SCL). In Table 2.3, you can see the length of recordings for each major dialect area.

Most of the 144 interviewees were born before 1905 and were 60 years old or older when interviewed. The interviewers were all native speakers, and conducted the interviews in the interviewee's homes to make them feel comfortable as well as to keep them in an environment where their dialect would be uninhibited. The interviewees were located in a number of counties

Dialect	Code	Hours
Southwest	SW	22.75
Southeast	SE	28.5
Midlands	MID	18
North	N	29
Scottish Lowlands	SCL	8.5

Table 2.3: Length of recording data for counties in FRED-S

around England and Scotland as can be seen in Figure 2.1. In Table 2.4, the amount of hours of recordings from each county can be seen.

County	Dialect	Hours
Cornwall	SW	3.5
Devon	SW	1
Oxfordshire	SW	2.5
Somerset	SW	7.5
Wiltshire	SW	8.25
Kent	SE	17.5
London	SE	7.5
Middlesex	SE	3.5
Leicestershire	MID	0.5
Nottinghamshire	MID	17.5
Durham	N	3.75
Lancashire	N	11.5
Northumberland	N	4.5
Westmorland	N	3.25
Yorkshire	N	6
East Lothian	SCL	3.5
Midlothian	SCL	2.5
West Lothian	SCL	2.5

Table 2.4: Length of recording data for counties in FRED-S

Fred-S was designed to be a tool for morphological analysis and is not suitable for discourse research due to the leading questions from interviewers. Also, because it was not created with acoustic dialect analysis in mind, there are some potential problems, which will be addressed in the following section.

Expectations for FRED-S Due to the nature of the corpus, there are some potential pitfalls to using FRED-S for acoustic dialect research.

Firstly, although the interviewers are native English speakers, they do not necessarily have the same dialect as the interviewee. In this study, I have chosen not to remove the interviewers due to time constraints but also because the introduction of external factors into the data may make the model more robust, simulating a 'real world' situation. I expect that the model trained on the MGB-3 will be much more accurate than a model trained on FRED-S, but I think that this could serve as a good metric of the strength of the acoustic dialect recognition model.

Secondly, because the age of the recordings ranges from between 50-20 years old, the quality is not always extremely clear. Therefore, there will

not only be quite a lot of background noise in the data but the recordings themselves will be quite unclear at times.

And lastly, there is an imbalance of data in regards to the dialects from the Scottish Lowlands, as can be seen in Table 2.3. This will need to be taken into account. It is not as nicely tailored as the MGB-3

But, even with the problems, I expect this to be a very interesting dataset to work with for dialect research. In comparison with the MGB-3, which does not include mutually understandable dialects, this one does.

3 Methodology

In this section, the methods used in this story are listed and explained. First, the audio processing techniques and pipeline will be explained. Then, there will be an introduction into the deep learning techniques used. And finally, the descriptions of the models used will be explained.

3.1 Audio Processing

To understand how to use machine learning for speech recognition or dialect or native language identification, it is important to understand the structure of audio data at a base level. In this section, I will outline the steps taken to process the audio data in preparation for statistical models. The following overview in this first section has been adapted from the Audacity Development Manual [2]

On a physical level, sound is varying pressure which travels through air. These physical pressure waves can be represented in a tangible form, and is the case for vinyl records, for instance. However, representing varying pressure sound waves in digital format requires a different approach.

One of the most important factors for audio data is time. Audio signals are time bound, and information must be represented on a continuum. It is computationally infeasible to have an unbroken continuous audio sample and thus a predefined *sampling rate* is used so that less data needs to be recorded.

Sample rates are a measurement of how many audio samples are recorded every second, and are represented in *hertz*. The higher the sampling rate, the higher the frequencies which can be represented. To represent audio waves in a digital format, the sample rate must be at least double the highest frequency recorded in order to reconstruct the original audio (how this is done will be spoken of later). Anything less will result in distortions and poor quality

[2] Audacity® software is copyright © 1999-2018 Audacity Team. Web site: `https://audacityteam.org/`. It is free software distributed under the terms of the GNU General Public License. The name Audacity® is a registered trademark of Dominic Mazzoni.

recordings. The highest sampling rate to be recorded is called the *Nyquist frequency*.

Because the human ear can only register sounds between around 20 Hz to 20,000 Hz, a standard sampling rate used for most audio CDs and MP3s which fulfils the minimum Nyquist frequency is 44,100 Hz. Being slightly more than double the frequency detectable by humans makes it both effective and efficient. Higher sampling rate will not necessarily always produce recognisably higher quality sound and will require more memory.

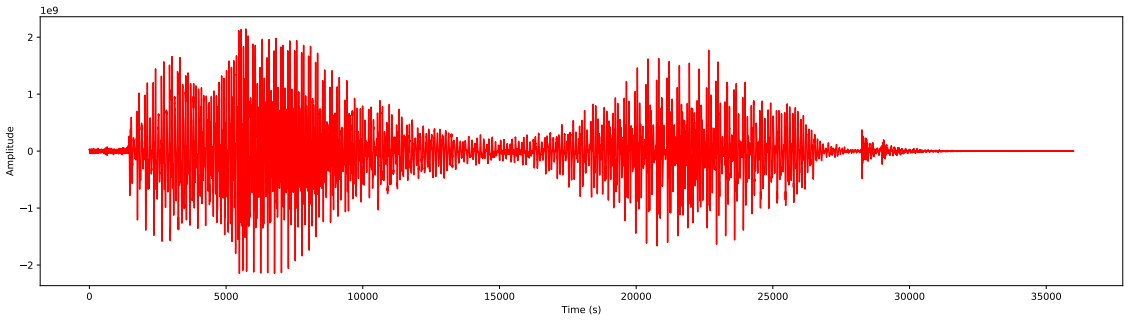


Figure 3.1: Periodogram of me saying "Hello, world!" without emphasis

In Figure 3.1, a plotted signal can be seen. The Wav file consists of 36,000 samples, which can be seen on the *x-axis*, and amplitude is represented on the *y-axis*. This is also called a *periodogram*. Nowadays, this alone can be enough for deep learning but only if there is enough data available. As I do not have near limitless data, further processing is required.

3.1.1 Preparing signal for statistical models

The following processing steps are modelled after how the human ear processes sound. The cochlea's surface is covered with fine hairs. Each region of the cochlea corresponds to a different frequency which in turn is what our brains interpret as sound. The periodogram in Figure 3.1 is only one step, but our ears do not solely respond to the amplitude of a signal in this linear fashion. Therefore, we must extract the frequency and amplitude of frames of the signal to simulate this. Two of the most common processing techniques are extracting a spectrogram and the *Mel-frequency cepstral coefficients* (MFCC).

In the next sections, I will outline the steps taken to convert the signal to spectrograms and MFCCs for use with statistical models. These steps have been adapted from Huang et al. (2001) and Sahidullah and Saha (2012).

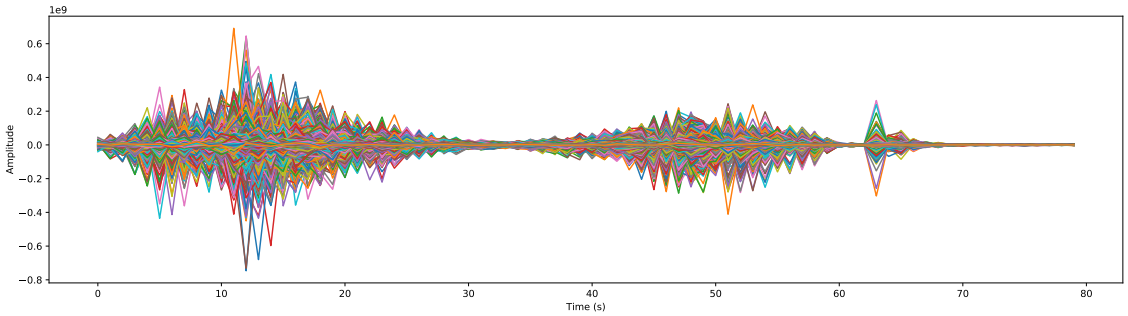


Figure 3.2: Signal divided into 80 frames in 25ms windows with 10 ms steps

Framing To extract features from the text for use in statistical models, the text must be divided into frames. A common approach is to make frames in the range of 20-40 milliseconds with 10-15 millisecond overlapping steps. In Figure 3.2, we can see the same signal as in Figure 3.1 but with many smaller, overlapping lines. Frames 25 milliseconds long with 10 millisecond overlap were used.

Hamming window

$$w[n] = \alpha - \beta cos(\frac{2\pi n}{N-1}) \tag{1}$$

In order to prepare each frame for *Fourier Transformation*, a *Hamming Window*, defined in Equation 1, must be applied to cancel the largest side lobe to avoid discontinuity in the signal (Harris, 1978). In my audio processing, $\alpha = 0.54; \beta = 0.46; n = frame, N = frame_length$.

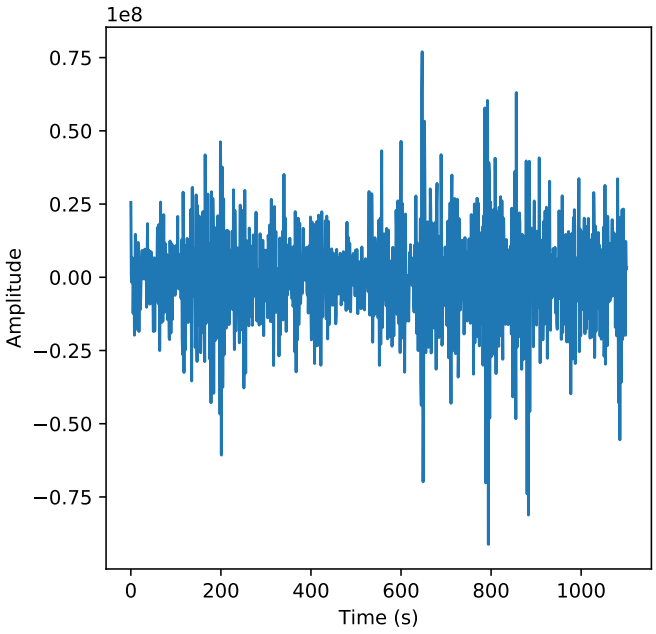


Figure 3.3: Frame of signal without hamming applied

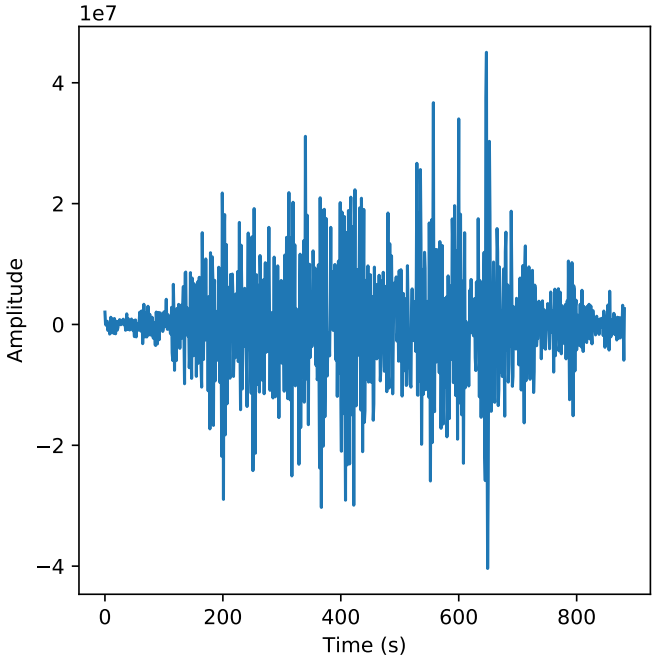


Figure 3.4: Frame of signal with hamming applied

Figure 3.3 displays one frame from the audio signal without a hamming window applied. In contrast, Figure 3.4 displays the same frame with it applied. It is clear that the left and right lobes are balanced in the former, making the signal smoother to work with.

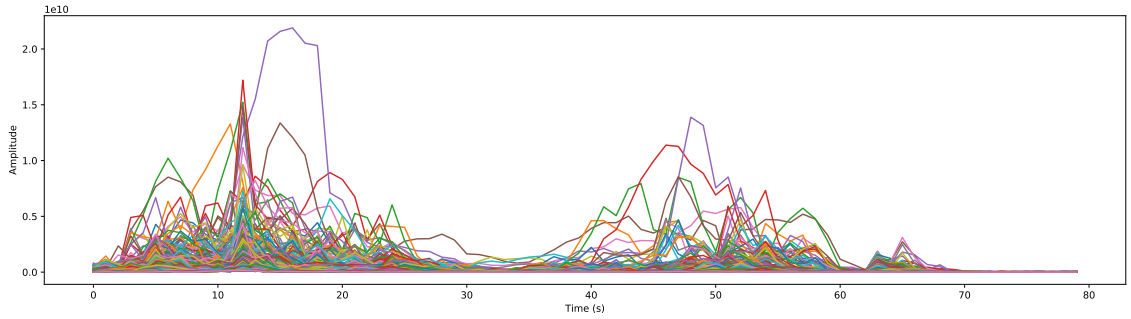


Figure 3.5: Framed signal with Fast Fourier Transformation for *Hello, world!*

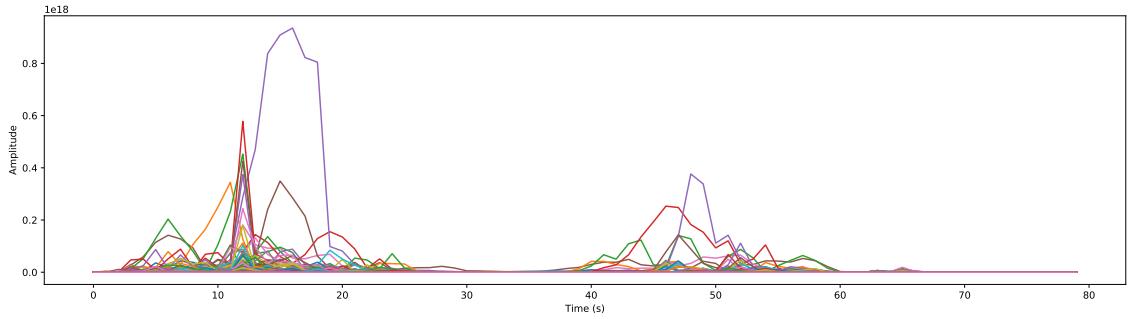


Figure 3.6: Power spectrum for *Hello, world!*

Fast Fourier transform and power spectrum Next, a *Fast Fourier Transform* (FFT) is performed (Numpy (Oliphant, 2006) was used in this study). This is an algorithm which extracts the spectral components of a signal in order to pass on frequency information. Subsequently, the power spectrum was computed on the signal, removing white noise. Figure 3.5 shows the spectral components of the signal after undergoing FFT. The signal is no longer a fluid wave, but rather a measure of amplitude. Figure 3.6 is the power spectrum which reduces the noise.

Filter banks As mentioned earlier, the human ear not only processes amplitude, but also frequency. The *Mel scale* is a scale of pitches perceived to be equal in distance from each other (Stevens et al., 1937).

$$m = 2595 log_10(1 + \frac{f}{700}) \tag{2}$$

$$f = 700(10\frac{m}{2595} - 1) \tag{3}$$

The Mel-frequency can be obtained with with the formula in Equation 2 and then reversed with Equation 3.

$$H(k,m) = \begin{cases} 0 & f(k < f(m-1)) \\ \frac{k - f(m-1)}{f(m) - f(m-1)} & f(m-1) <= k < f(m) \\ 1 & k == f(m) \\ \frac{f(m+1) - k}{f(m+1) - f(m)} & f(m) < k <= f(m+1) \\ 0 & k > f(m-1) \end{cases} \tag{4}$$

After obtaining the Mel-frequencies, a *filter bank* must be created using Equation 4 (Sigurdsson et al., 2006), where k is the frequency, M is the number of filters, $m = 1, 2, ..., M$. This creates a an identity matrix of overlapping Mel-frequencies of increasing size. Figure 3.7 shows the result of Equation 4 with 40 filters.

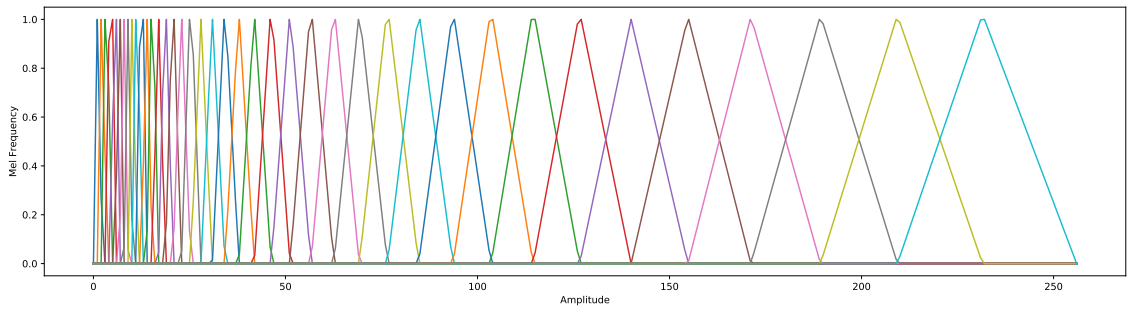


Figure 3.7: Mel filter banks with 40 filters

To obtain a Mel-spectrum filter bank, one must simply multiply each frame of the power spectrum with the periodogram then take the logarithm of them.

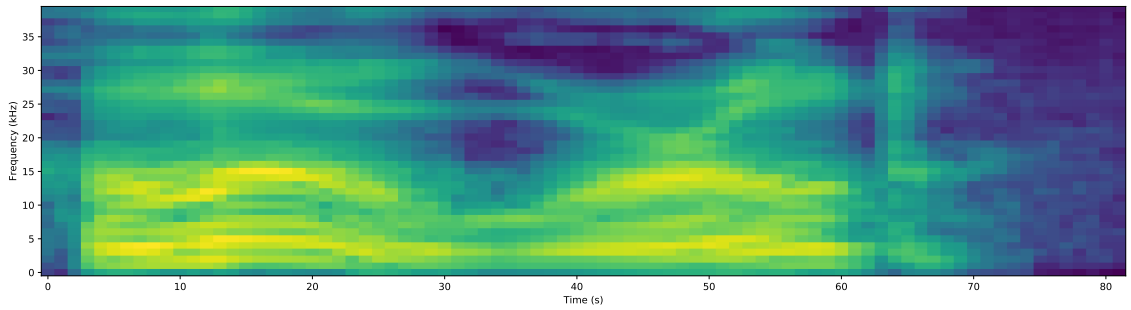


Figure 3.8: Log Filter Bank for *"Hello, world!"*

The log filter bank for "Hello, world!" can be seen in Figure 3.8

Spectrograms Spectrograms are a common three dimensional representation of an audio signal. The typical representation has time on the *x-axis*, frequency on the *y-axis*, and amplitude on the *z-axis*. They can be represented in two dimensions with amplitude being represented as various shades on the frequency axis. The spectrogram of me saying "Hello, world!" is in Figure 3.9

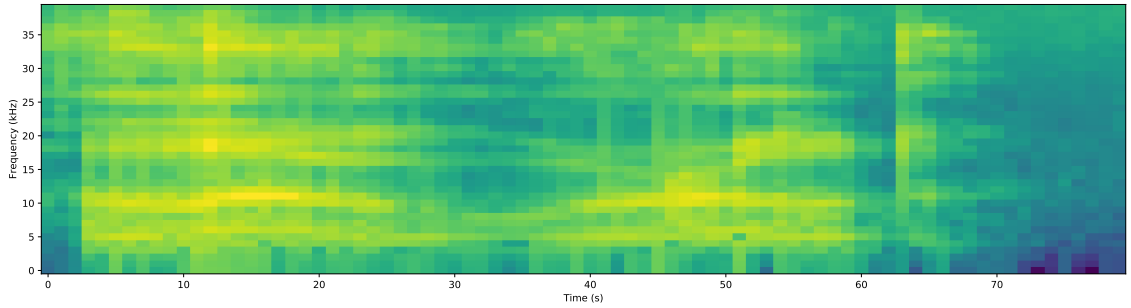


Figure 3.9: Spectrogram for *"Hello, world!"*

To generalize the data in statistical models, it is also common to normalize the spectrograms. Figure 3.10 shows the same spectrogram but with *mean normalization*.

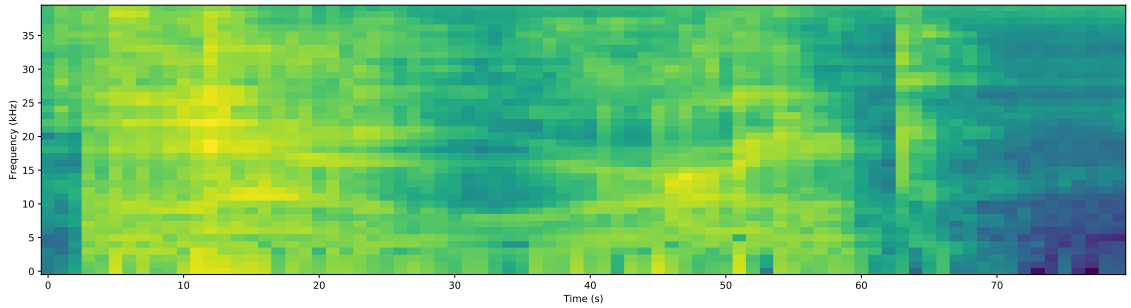


Figure 3.10: Normalized Spectrogram for *"Hello world!"*

Because the filter banks used for processing spectrograms are overlapping, the values are highly correlated. This poses a problem for machine learning

because finding discriminable patterns in highly correlated features in difficult and slow going. Therefore, further processing is usually done.

Mel-Frequency Cepstral Coefficients Mel-frequency cepstral coefficients are a compressed version of a spectrogram which are used extremely often for ASR.

To generate MFCCs, the log spectrogram undergoes a *Discrete Cosine Transform* (DCT) (Ahmed et al., 1974) to decorrelate the features. This makes the data smaller which is computationally more efficient. For tasks such as ASR, only 12 cepstral coefficients are usually retained. The MFCCs can be seen in Figure 3.11.

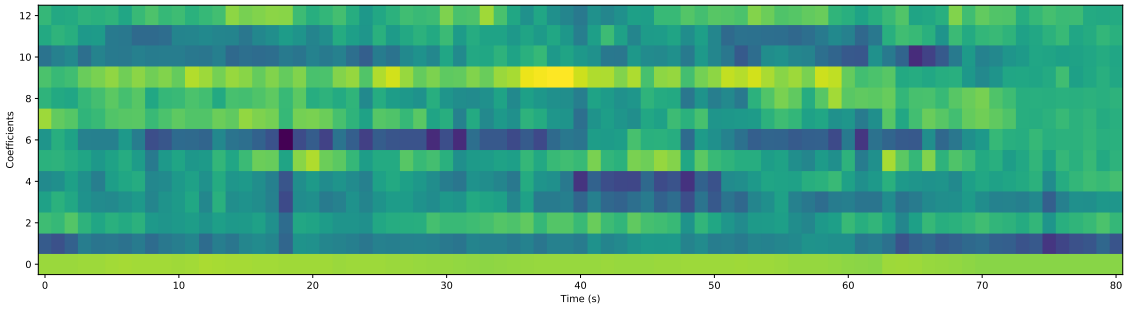


Figure 3.11: Mel-Frequency Cepstral Coefficients of signal for *"Hello world!"*

Although it looks much different from a spectrogram, there are similar patterns which can be seen.

Delta and Delta-Delta Cepstral Coefficients MFCCs present acoustic information very well, but they are static and do not include dynamic features. A solution to this is to use *Delta Cepstral Coefficients* (Δ) and *Delta-Delta Cepstral Coefficients* ($\Delta\Delta$) in addition to MFCCs (Furui, 1986). They are calculated using Equation 5. It is computed for frame f where $N = 2$.

$$\Delta_t = \frac{\sum_{n=1}^{N} n(c_{t+n} - c_{t-n})}{2\sum_{n=1}^{N} n^2} \tag{5}$$

Δ, which include the differential information, are created by computing each frame of the MFCCs (Figure 3.12).

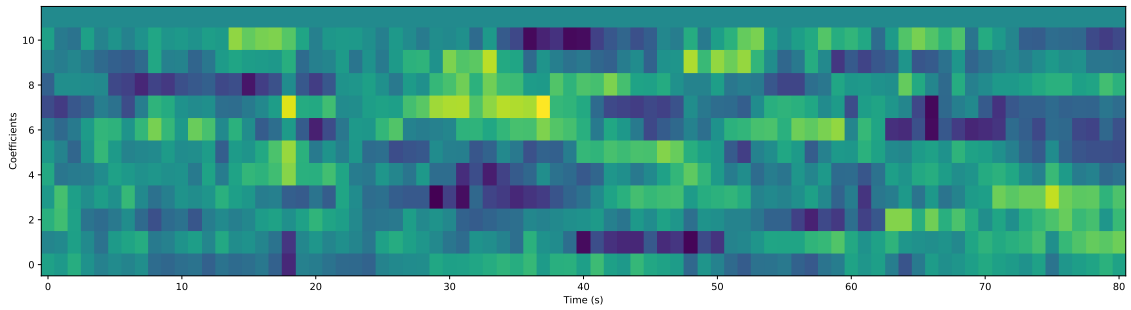


Figure 3.12: Δ coefficients of MFCCs for *"Hello world!"*

ΔΔ (Figure 3.13), which include the acceleration information, are created by computing each frame of Δ.

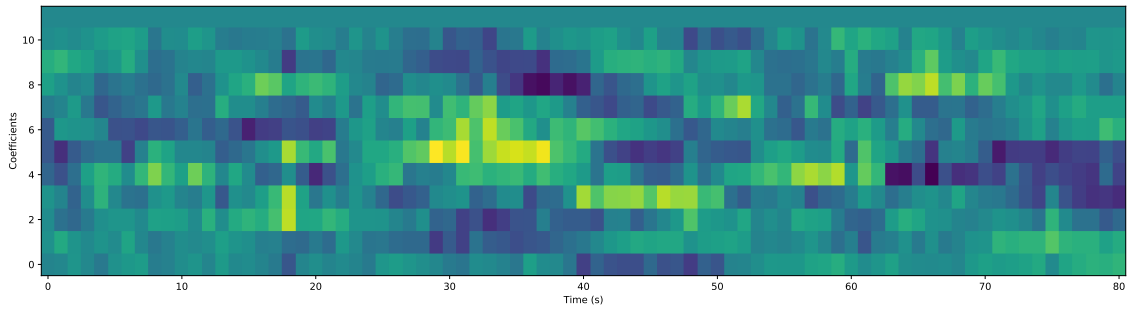


Figure 3.13: ΔΔ coefficients of MFCCs for *"Hello world!"*

It has been reported that using Δ and ΔΔ in speech recognition tasks has increased performance of models by 20% (Kumar et al., 2011a).

Delta-Spectral Cepstral Coefficients Although, Δ and ΔΔ provide good dynamic information, they are susceptible to noise and reverberation. To counteract this, Kumar et al. (2011b) have proposed *Delta-Spectral Cepstral Coefficients* (ΔSCC) which capture the contextual information of Δ with more robustness to additive noise.

To obtain the ΔSCC, first Δ is obtained using spectral rather than cepstral features using the delta-operation in Equation 5. Then, *histogram normalization* is performed in order to give the data a *Gaussian* distribution followed by a DCT (Figure 3.14).

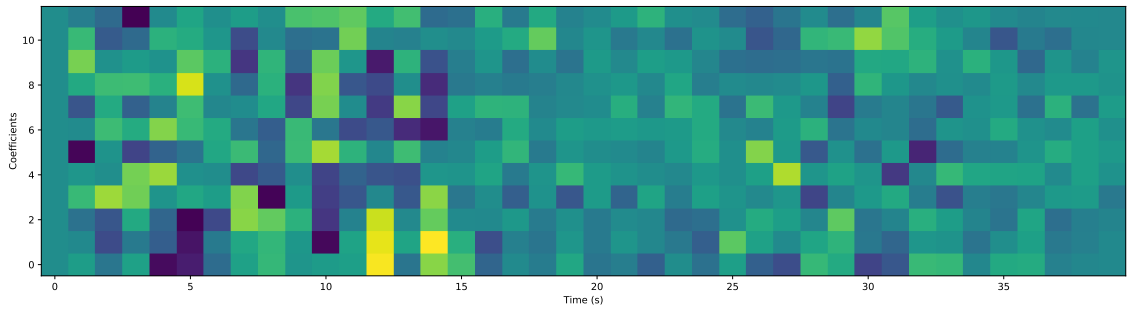


Figure 3.14: Δ on spectral features for *"Hello world!"*

These features then undergo a second delta-operation is then extracted from the Δ-spectral features from Figure 3.14 and the results are the ΔSCC.

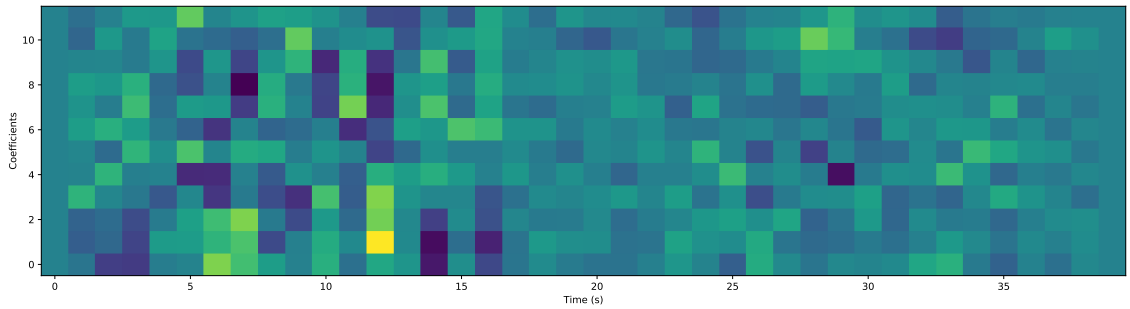


Figure 3.15: ΔSCC for *"Hello world!"*

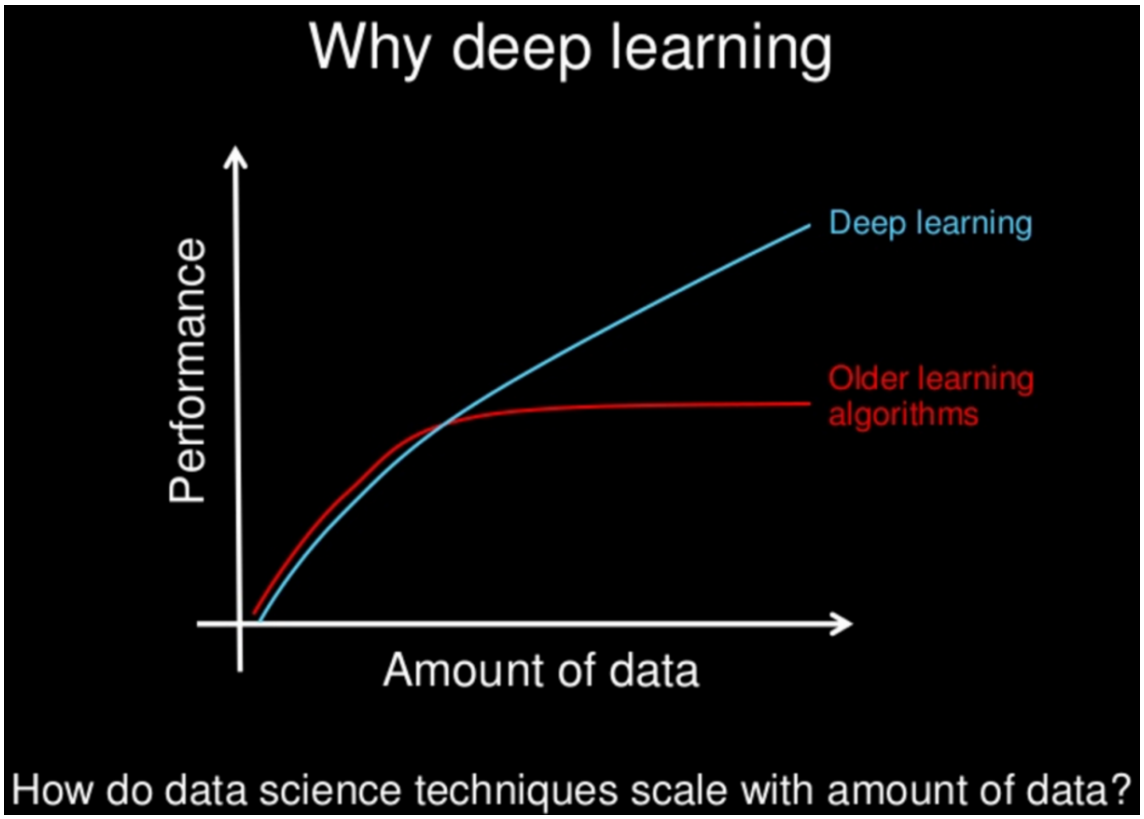
Using ΔSCC has proven to reduce *Word Error Rate* (WER) in speech recognition tasks by 20-30% (Kumar et al., 2011b). In currently unpublished ASR results, I have also found them to greatly boost accuracy. This success and robustness warrants attention, and could be very beneficial to acoustic related tasks.

Being a relatively new technique, the only implementation available was made available by the authors for Matlab. I have created a python implementation which can be found here:

3.2 Deep Learning Methods

Deep learning, a subset of machine learning as a whole, has been around for several decades already but it has only recently exploded in popularity due to advances in computational resources and the increasing availability of large datasets. Although more traditional machine learning approaches such as *Support Vector Machines* or *AdaBoost* work quite well for many classification tasks and can be computationally efficient, they can be quite limited. They rely heavily on feature engineering and data selection which requires a lot of attention paid to input details. From a research perspective, this is usually

fine, but in a production setting, finding engineers and specialists to work with the data can be time consuming.



Credit: *Andrew Ng* Source :https://www.slideshare.net/ExtractConf/andrew-ng-chief-scientist-at-baidu

Figure 3.16: Deep learning algorithms consistently outperform older approaches with more data

Furthermore, it has been shown that there can be a ceiling to how effective traditional machine learning approaches can be for many tasks which cannot be solved with feature engineering or more data.

However, given enough data, deep learning algorithms consistently outperform traditional machine learning approaches as can be seen in Figure 3.16.

Another advantage to Deep Learning is that the lack of a need for feature engineering. Using highly engineered features in Deep Learning models could even be detrimental and cause overfitting. The trade-off is that neural models will need much more data.

Here is an example: although most neural network based speech recognition systems do use some of the features outlined in Section 3.1, many of the most recent models use nothing but the raw *WAV* signal. These newer models are beginning to perform better than previous engineered models, but the amount of data and processing time they need are much more than most causal programmers and researchers can facilitate.

So, as this study will be conducted with limited resources, a balance between feature engineering and deep learning will be experimented with. In

the following sections, a brief overview of *Neural Networks* will be given along with the specific hyperparameters used. Because there will be multiple neural architectures, a separate description will be given for *Feed Forward Neural Networks*, *Convolutional Neural Networks*, and *Recurrent Neural Networks*.

3.2.1 Feed Forward Neural Network

A Feed Forward Neural Network (FFNN) is the base architecture of Deep Learning. In this section, I will describe the foundation of FFNNs which will concurrently describe the base concepts for the architectures outlined in Sections 3.2.2 and 3.2.3.

A *Neural Network* is composed of the following: An *input layer*. A series of one or more *hidden layers* which are composed of *neurons*. And finally, depending on the type of prediction desired, a *sigmoid*, *linear*, or *softmax* layer for output predictions.

In Section 3.2.1, I will describe the process in which features are extracted from the input and predictions are made and in Section 3.2.1 I will describe how the model is adapted during training.

Forward Propagation To extract features, *Forward Propagation* is computed on the input (x) of a Neural Network. Each hidden layer (l) has it's own *bias* (b) and is composed of a series of neurons. Each neuron has it's own set of weights (w) which are randomly set during initialization and an *activation function*.

$$z^l(x) = w^l x + b^l \tag{6}$$

All inputs x from *timestep* t are multiplied with the weights w of each neuron and then the bias b is added. This is formally represented in Equation 6.

Because each set of weights for each neuron are different, multiplying them with the input emphasises varying patterns and features. In this way, a neural network is able to discern the nuances in data much more than older machine learning approaches.

Batch Normalization An optional step at this point, would be to use *Batch Normalization*. Batch normalization is a technique created by Ioffe and Szegedy (2015) and has almost become ubiquitous in deep learning. It has been proven to act as a regularizer, allow the use of much higher learning rates, and even be used as a substitute for dropout. The authors claim that it can make training up to 14 times faster.

$$BN = \frac{\gamma(x - \mu)}{\sigma} + \beta \tag{7}$$

Seen in Equation 7, it is computed separately for each hidden layer where μ is the mean of a tensor, σ is it's variance, and γ and β are learnable hyper-parameters for the scale and offset respectively.

Batch normalization was used for some models in this study to speed up training as well as prevent overfitting.

Activation Functions Once Z is computed for a layer of the network, a *non-linearity* must be introduced to the output of Equation 6. The result of Z is linear in nature and if left inactivated, the network would become an outrageously expensive linear model and would not learn any of the fine-grained differences it should. There are a variety of different activation functions out there such as the $sigmoid(x)$ or $tanh(x)$ functions. These functions produce excellent results, however, they can become computationally taxing and can suffer from the problem of *vanishing gradients*.

The most used activation function at the moment, and the activation I chose to use for all hidden layers in my models, is the *Rectified Linear Unit* (ReLU) (Nair and Hinton, 2010). ReLUs have two main advantages over other activation functions. First, they were found to accelerate the convergence of gradient descent by a factor for 6 in comparison to *sigmoid* or *tanh* (Krizhevsky et al., 2012). Secondly, it is much faster than other.

$$A(x) = max(0, x) \tag{8}$$

As can be seen in the formal definition of ReLUs in Equation 8, the threshold is either 0, or simply x, or in other words, the results of z. Therefore, the formal definition for a hidden layer of a FFNN is seen in Equation 9

$$A^l(x) = A^l(Z^l(x)) = A^l(W^l x^{l-1} + b^l) \tag{9}$$

Softmax After the input is propagated through all hidden layers, it must be normalized and prepared for use in classification. a of the final layer will not be a normalized as probabilities for classification. This is where the *Softmax* function in comes in handy for multi-class classification problems.

$$softmax(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \tag{10}$$

The softmax function, as seen in Equation 10, creates a *squashed* vector of arbitrary values from the last hidden layer where K is the number of classes to predict and $j = 1, \ldots, K$. The result will be a probability distribution where all values are within the range of $(0, 1)$ and all add up to 1. Often, the outputs of the softmax layer are referred to as *logits*.

From here, either predictions can be made, or the loss can be computed in

order to common-sense *Back Propagation* and train the network.

Back Propagation In order to train a neural network, the parameters such as the weights and biases must be updated and adapted according to the data they encounter. To do this, back propagation must be computed.

Cross Entropy Loss After one batch of forward propagation, the *loss* of the predictions is computed to see how close it is to the target predictions. If the predictions are accurate, the loss will be low, whereas if the predictions are wildly inaccurate, the loss will be high.

There are several possible loss functions, but the one used in this study is *Cross Entropy*, so this is the example I will use.

$$\mathcal{L}(\hat{y}, y) = -\sum_{i} y_i \log_2(\hat{y}_i) \tag{11}$$

In Equation 11, $\hat{y}$ is the gold standard labels and y are the predicted logits from the softmax function.

$$\mathcal{L}_{L2} = \mathcal{L} + \beta||W||^2 \tag{12}$$

An optional step at this stage is to add L2 regularization, which has been shown to reduce overfitting. The L2 norm is computed for all layer weights, summed, and then added to the loss. This is shown in Figure 12 where $\mathcal{L}$ is the loss computed in Equation 11 and β is a small arbitrary value, which increases regularization the larger it is.

Optimization Once, the loss is computed, the network must be adjusted and optimized accordingly. The optimization method used in this paper is the *Adaptive Moment Estimation* or simply *Adam* Optimizer (Kingma and Ba, 2014). Adam has consistently been shown to be both extremely effective as well as more computationally efficient, thus speeding up training, in comparison to traditional approaches such as gradient descent.

Using Adam, the learnable parameters of the model are updated according to the loss. After this, the process is then reiterated from the beginning and will continue to do so until a well performing and robust model is created.

This concludes the brief overview on FFNN. In the following sections, I will explain CNNs and RNNs. I will explain what makes them different from FFNN and point out the attributes they share.

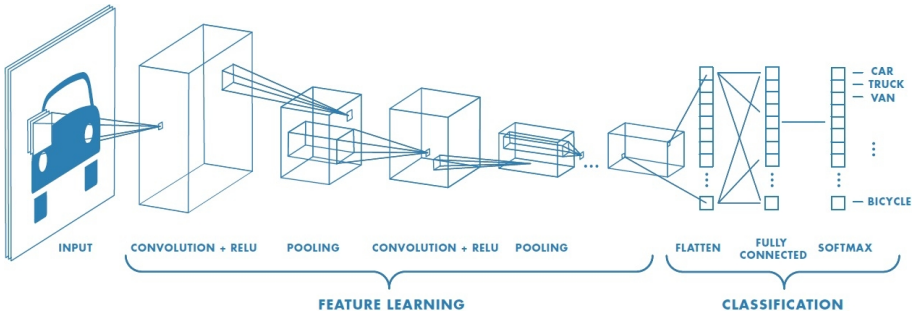
3.2.2 Convolutional Neural Network

CNNs are extremely effective for data with spacial relations such as images. As seen in Section 3.1, audio data can be represented as images in a multi-dimensional space with spacial dependencies.

In the next section, I will describe how a CNN functions and what makes it different from the FFNN described in Section 3.2.1.

Convolution Layers The basic steps for a convolution layer are as follows:

- Convolution
- Activation
- Pooling (optional)



Credit: *MathWorks* Source: `https://it.mathworks.com/solutions/deep-learning/convolutional-neural-network.html`

Figure 3.17: Convolutional neural network

Convolution Operation First, the *Convolution* operation is both what makes a CNN different from a FFNN as well as makes it crucial for image recognition. A 3-dimensional feature $X \in R^{c \times b \times f}$ where c is the number of channels, b is the frequency bandwidth, and f is the time length is given to a CNN as input

Then, a *Kernel* with preinitialized weights $W \in R^{c \times m \times n}$ where c is the number of channels, m and n are a predefined hight and width convolved X.

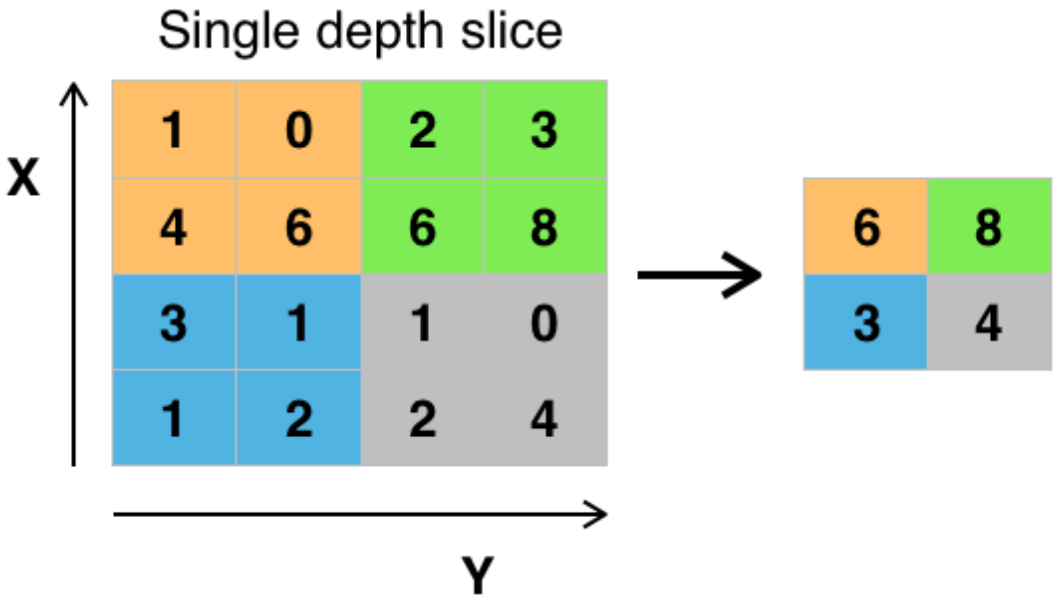
Given a step length, each element of the kernel is multiplied with the corresponding element on the section of the input tensor it is on. The resulting values are summed and and then passed on to a new tensor. This is repeated until the entire input tensor has been convolved and the resulting tensor will be smaller than the original unless padding is used.

The number of kernels used can be set, which will result in a corresponding number of filters. The result will be a 3-dimensional *feature map* which can then undergo the chosen activation function. In this study, only the ReLU activation was used.

$$H_i = W_i * X + b_i, i = 1, \ldots, k \tag{13}$$

This entire convolution is shown in Equation 13, where $*$ denotes the convolution operation and k is the number of filters.

Pooling In order to speed up a CNN, the data can be reduced using *Pooling*. At initialization, a pooling window's size and stride can be defined. For *Max Pooling*, the largest value within the window at each step is taken and transferred to a smaller resulting matrix, as can be seen in Figure 3.18.



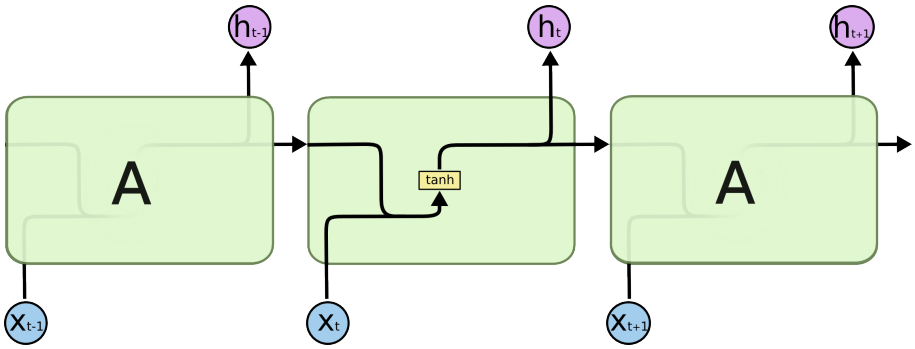
Credit: *Aphex34* Source: `https://commons.wikimedia.org/wiki/File:Max_pooling.png`

Figure 3.18: Max pooling example

Average Pooling can also be used. Instead of taking the max value as in max pooling, the average of the values in the window is used instead.

3.2.3 Recurrent Neural Network

One thing that humans are much better at than machines at the moment, is using a multitude of past experiences to influence a current decision. Our past experiences may not be directly related to what we are doing at the moment, but we can still make inferences from them.



Credit: *Christopher Olah* Source: `http://colah.github.io/`

Figure 3.19: RNN visualized

That is something which FFNNs and CNNs lack. This has been attempted to be solved by *Recurrent Neural Networks* (RNN) with some measure of success. They look almost exactly like a FFNN, but with a major difference. The computations at each hidden layer look similar to those in Equation 6, but it includes the weights from the previous timestep at the current timestep.

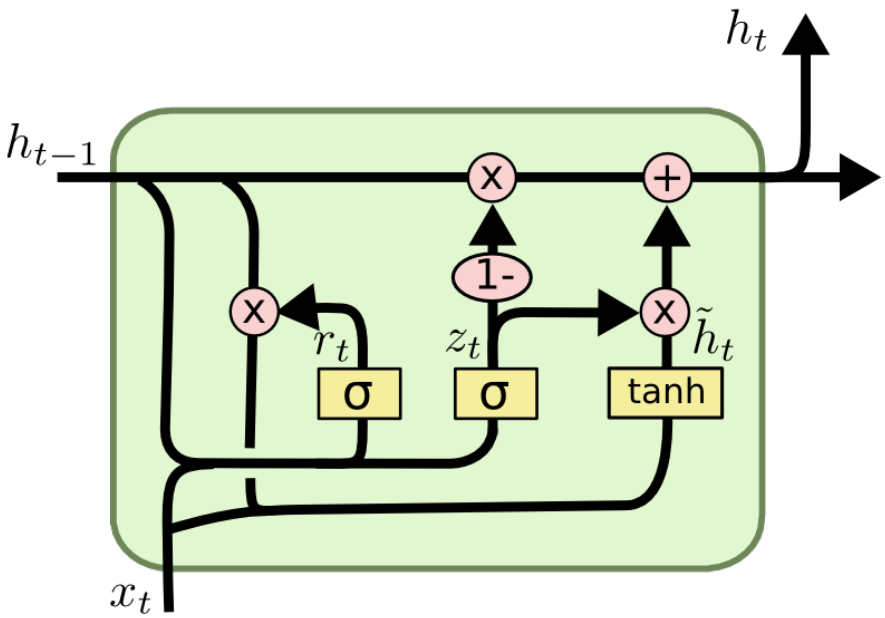
$$z^t(x) = W_{aa}a^{t-1} + W_{ax}x^t + b_a \tag{14}$$

This can be seen in Figure 3.19 or in Equation 14 where W_{aa} is a weight matrix pasted through every timestep and a^{t-1} is the output of the hidden layer at the last timestep. In so doing, "memories" of what happened at previous timesteps are passed on. This has been shown to work wonderfully for NLP tasks such as text generation, machine translation, and ASR. Therefore, it is my belief that this can also work very well for dialect recognition.

However, a problem of generic RNNs is that they pass on long term dependencies indiscriminately. This has been remedied to a degree with the use of RNN cells which enact a bit of discretion. Primarily, with the use of *Gated Recurrent Units* (GRU) (Cho et al., 2014) and *Long Short-Term Memory* (LSTM) (Hochreiter and Schmidhuber, 1997). Although different, GRU and LSTM architectures are very similar and Chung et al. (2014) found the differences in performance negligible. There is the slight expectation that LSTMs can hold slightly longer dependencies than a GRU, whereas a GRU is much more computationally efficient.

In the next section, I will describe the technical side of GRUs and what makes them different from typical RNNs.

Gated Recurrent Unit The gates in GRUs, as can be seen in Figure 3.20, are what make them so effective. For a vanilla RNN, it simply passes on former weights indiscriminately whereas a GRU has it's own set of activations and learnable weights which choose which information to pass on or not.



Source: https://mc.ai/

Figure 3.20: GRU visualized

A GRU has two gates: a reset gate and an update gate. First, the reset gate in Equation 15 is activated. It looks very similar to the normal RNN Equation 14 but with a *sigmoid* activation which outputs a value between 0 and 1. This determines how the new input x should be combined with the memory.

$$r_t = \sigma(W_r[h_{t-1}, x_t]) \tag{15}$$

Next, the update gate in Equation 16, which is identical to the reset gate function, is used to determine how much of the memory is used.

$$z_t = \sigma(W_z[h_{t-1}, x_t]) \tag{16}$$

The output of the reset gate r_t is combined again with the memory from the previous timestep h_{t-1} and then multiplied with a new set of weights. Then the input x_t is multiplied with its own set of weights. A *tanh* activation is then used on this intermediate memory level $\tilde{h}_t$ as can be seen in Equation 17.

$$\tilde{h}_t = \tanh(W_h[r_t h_{t-1}, x_t]) \tag{17}$$

Finally, the update gate z_t is subtracted from 1 and multiplied with the memory h_{t-1}. The previous memory h_{t-1} is multiplied with the current memory h_{t-1} in order to get the final output h_t.

$$h_t = (1 - z_t)h_{t-1} + z_t\tilde{h}_t \tag{18}$$

3.3 Models

Three different model architectures were tested on each dataset: A RNN sequence model, a one-dimensional CNN, and a hybrid one-dimensional CNN-RNN.

I tried to keep the models as similar as I could between datasets, but some adjustments needed to be made. In the next section, I will describe the model architectures and the parameters used for each. Parameters which differ between datasets are in bold

Each model/feature combination was trained on 30 epochs, used Adam optimizer, and ReLU activations.

3.3.1 RNN Approach

For my RNN model, I chose to use a sequence classification based approach, also known as a many-to-one classification.

In this approach, an audio segment is treated as a sequence of frames, where each frame is a timestep. Only the final frame, or timestep, is used as the output from a RNN cell. This has the advantage of having the previous memory weights from the entire audio sequence for the last timestep, but will not have the weights from other audio segments.

GRU cells were used over LSTMs due to their more efficient nature. LSTMs might be better for longer dependencies, but I do not think they are necessary for such short sequences.

Parameter	Configuration	
	MGB-3	FRED-S
GRU sizes	$(256 - 256 - 64)$	$(256 - 256 - 64)$
Batch size	64	64
Dropout	0.2	0.2
Activation	ReLU	ReLU
Optimization	Adam	Adam
Starting learning rate	0.001	0.001
L2 Regularization	True	True

Table 3.1: Parameters for RNN model

Model Parameters For this model, I worked with segments 512 frames second long (circa 5 seconds). Sound clips shorter than 5 seconds were padded at the beginning of the sequence.

The parameters for both datasets was the same. 3 GRU hidden layers were used $(256 - 256 - 64)$ with dropout of 0.2, a batch size of 64 and L2 regularization.

The learning rate was set at 0.001 and learning rate decay use to reduce the learning rate after every epoch.

I tested 2 feature types: Mel-Spectrograms with 40 frequency bands and MFCC. I did not use ΔSCC because of a problem with loss calculations.

3.3.2 CNN Approach

The model used is based on Shon et al. (2018). They used the MGB3 corpus, augmented the data for speed and volume, and randomly chunked it into segments ranging from 1-10 seconds. The Mel-scale filter bank energies (FBANK) were used as features.

They used a one-dimensional CNN with 4 convolution layers with $(500 - 500 - 500 - 3000)$ filters, strides of $(1 - 2 - 1 - 1)$, kernel sizes of $(40 \times 5 - 500 \times 7 - 500 \times 1 - 3000 \times 1)$, ReLU activation, batch normalization, no pooling layers and no dropout. They used *Global Average Pooling* for dimensionality reduction. There were then two fully connected layers of size $(1500 - 600)$. They used *Gradient Descent* for optimization.

They achieved 73.39% accuracy on the MGB3 using this model and only acoustic features. This is the most impressive acoustic based DID attempt I have found.

I have therefore chosen to adopt this model as a base for my model. But, because of computational limitations, I was forced to make some alterations to the model, which may have effected the final results.

Parameter	Configuration	
	MGB-3	FRED-s
Num. filters	**(64-128-256)**	**(128-256-512)**
Kernel sizes	$(5 - 7 - 2)$	$(5 - 7 - 2)$
Strides	$(1 - 2 - 1)$	$(1 - 2 - 1)$
Pooling sizes	$(2 - 2 - 2)$	$(2 - 2 - 2)$
Pooling strides	$(1 - 1 - 1)$	$(1 - 1 - 1)$
FC layer size	$(128 - 64)$	$(128 - 64)$
Batch size	64	64
Dropout	**0.2**	**0.5**
Activation	ReLU	ReLU
Optimization	Adam	Adam
Starting learning rate	0.0001	0.0001
Batch Normalisation	True	True

Table 3.2: Parameters for CNN model

Model Parameters For my CNN model, I worked with segments 512 frames second long (circa 5 seconds). Sound clips shorter than 5 seconds were padded at the beginning of the sequence. Due to computational limitations, I did not use as many trainable parameters as Shon et al. (2018). My model has 3 convolution layers of $(64 - 128 - 256)$ (MGB-3) and $(128 - 256 - 512)$

(FRED-S) filters, strides of $(1-2-1)$ and kernel sizes of $(5-7-1)$. Max pooling was used for data reduction. A pooling window of 2 and a stride of 1 was used for each convolution layer. Batch normalization and a dropout rate of 0.2 (MGB-3) and 0.5 (FRED-S) was used for regularization.

The output of the convolution layers was flattened and I used 2 fully connected layers of $(128-64)$ nodes with batch normalization and dropout.

The learning rate was set at 0.0001 and learning rate decay use to reduce the learning rate after every epoch.

I tested 4: Mel-Spectrograms with 40 frequency bands, MFCC, Mel-Spectrograms + ΔSCC, and MFCC + ΔSCC. ΔSCC features had 12 cepstrals apiece.

The model differences between the two datasets were due to the FRED-s corpus needing stronger regularisation measures and prevent drastic overfitting.

3.3.3 Fusion Model Approach

Because sound not only has spacial characteristics, but a time element, I wanted to experiment with a model which combines the feature selection of a CNN with the time bound information from an RNN.

It is very similar to the previous two models, but with some key differences.

Parameter	Configuration	
	MGB-3	FRED-S
Num. filters	**(64-128-256)**	**(128-256-512)**
Kernel sizes	$(5-7-2)$	$(5-7-2)$
Strides	$(1-2-1)$	$(1-2-1)$
Pooling sizes	$(2-2-2)$	$(2-2-2)$
Pooling strides	$(1-1-1)$	$(1-1-1)$
GRU layer size	$(128-64)$	$(128-64)$
Batch size	64	64
Dropout	**0.2**	**0.5**
Activation	ReLU	ReLU
Optimization	Adam	Adam
Starting learning rate	0.0001	0.0001
Batch Normalisation	True	True

Table 3.3: Parameters for fusion model

Model Parameters For this model, I worked with segments 512 frames second long (circa 5 seconds). Sound clips shorter than 5 seconds were padded.

The parameters for the convolution layers in the fusion model were the same as the one-dimensional CNN architecture in the previous section.

However, instead of flattening the output, which would have removed the time sequence information, the direct output after max pooling was used as input to two GRU cells. Dropout was used on the GRU cells, but no normalisation was used.

4 Results

There were some confirmed theories and some surprises in the final results of this study. First, I will start with the confirmed theories.

I had expected a hybrid CNN and RNN model to work the best for this task. This was confirmed as every feature produced it's best output with this model. I had also expected that the MGB-3 corpus would be much easier to work with, and this was also true.

However, there were several things which I did not expect. I had expected MFCC features would work the best for DID and had also expected adding ΔSCC features to greatly improve results, as it has shown to do for speech recognition. But the opposite of this was in fact true. Mel-Spectrogram features alone consistently proved to be the best and ΔSCC actually decreased model results in every case. The potential reasons for this will be discussed in Section 6.

The results for the MGB-3 corpus were quite good overall. During training, there was a steady increase in classification accuracy but with some variations in the loss. The top model and feature combination achieved an accuracy of 82% on the testing data.

I was a bit surprised by the results from the FRED-S corpus. I had very low expectations for classification accuracy due to the poor quality of the data and the extreme noisiness of it. I tried a few models with data that had had the ambient noise reduced and dead space removed. However, the best models were those with the unedited data.

Training the FRED-S models proved to be much more challenging than it was with the much cleaner MGB-3 data. As can be seen in the graphs below, the testing accuracy and loss during training had quite a bit of variance. It did, however, work and the best model and feature combination achieved a classification accuracy of 81%.

In this rest of this section, the final results of the models will be analysed. For each model, corpus, and feature type, the top accuracy and loss on the testing set within 30 epochs is given as well as a graph of each after every epoch.

4.1 RNN Approach

The RNN approach performed much better than I thought it would, overall. However, it did not perform well with MFCC and presented some difficulties with ΔSCC. ΔSCC created NaN loss during training, which made back propagation impossible and were, therefore, not used.

Feature Type	MGB-3		FRED-S	
	Accuracy	Loss	Accuracy	Loss
MFCC	0.30	10.62	0.28	11.0
Mel-Spec	**0.65**	1.05	**0.71**	1.21

Table 4.1: Results for RNN Model

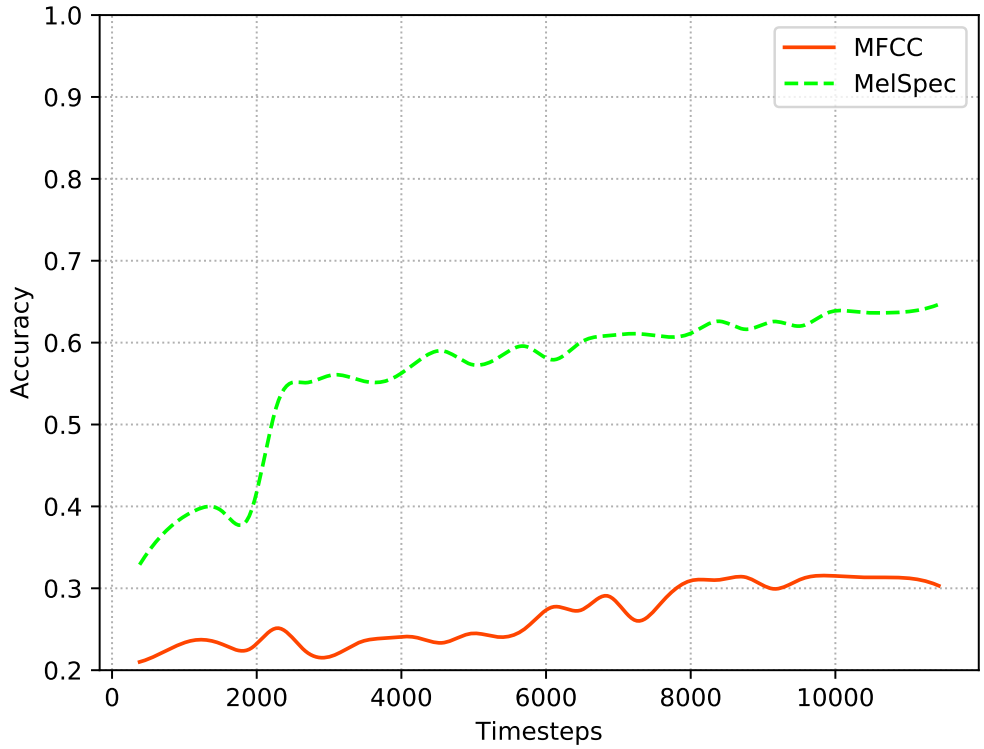


Figure 4.1: RNN development accuracy for MGB-3

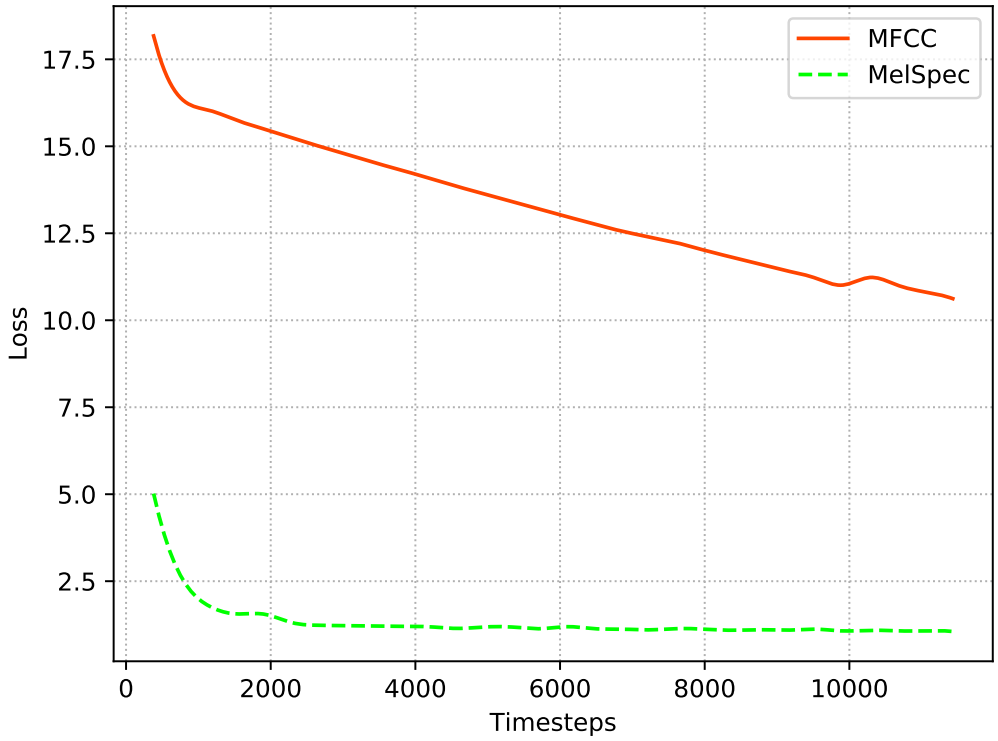


Figure 4.2: RNN development loss for MGB-3

MGB-3 Results The RNN approach did have some degree of success with Mel-Spectrograms as features, achieving a final accuracy of 65% and a loss of 1.05. Loss was also quite good in this, however it plateaued fairly early during training as can be seen in Figure 4.1.

However, MFCC features proved to be completely insufficient for the RNN model. With a final classification accuracy of 30%, the results were almost completely random. This accuracy in conjunction with a terrible final loss value as seen in Figure 4.2 makes this approach useless.

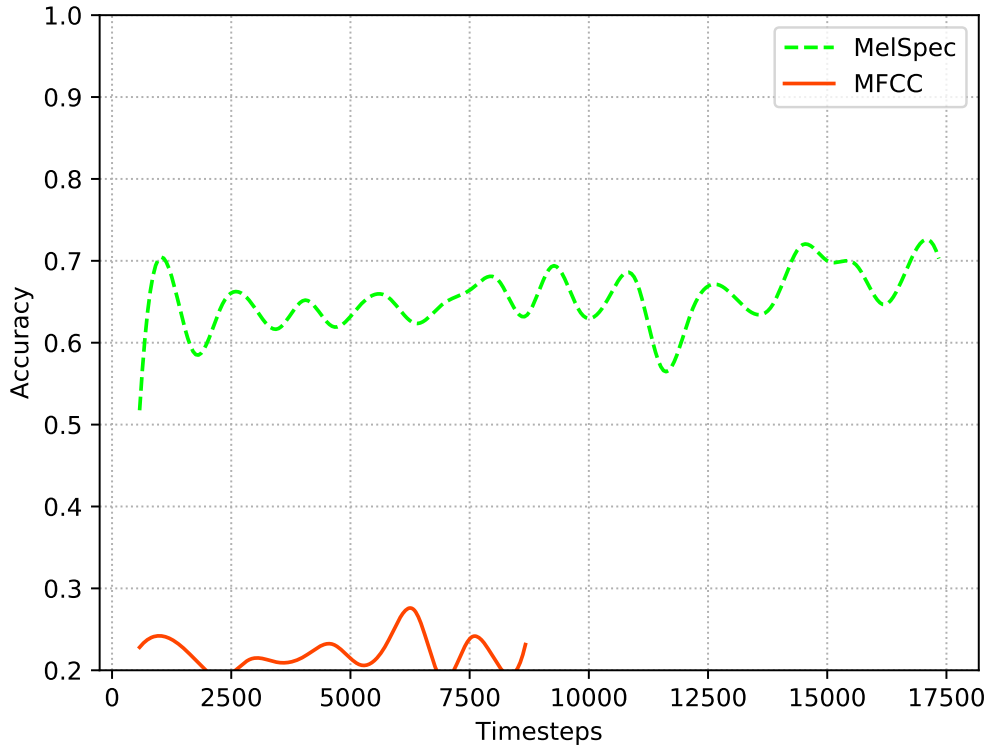


Figure 4.3: RNN development accuracy for FRED-S

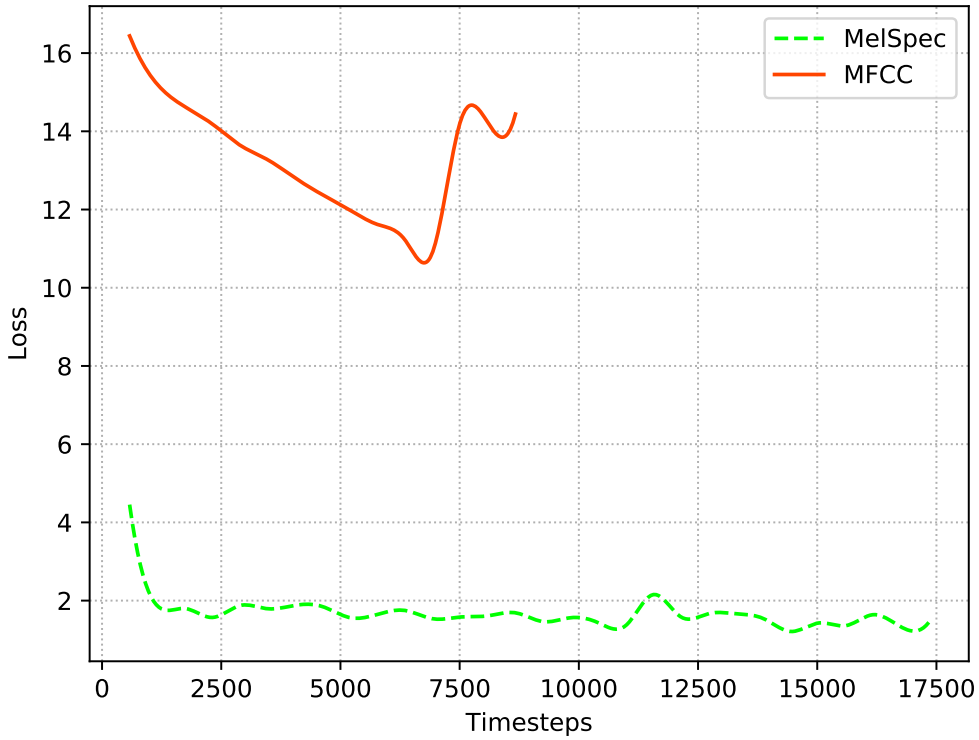


Figure 4.4: RNN development loss for FRED-S

FRED-S Results The RNN model actually worked very well with the FRED-S corpus. As will become evident in the next two models, the English data was a bit chaotic to train. However, the RNN model was probably the least chaotic model, even if it did not get the highest classification accuracy.

As can be seen in Figures 4.3 and 4.4, the steady improvement over training is clear when using Mel-Spectrograms. The development accuracy has a steady increase to 71% with a loss of 1.21. It has more variation than the same model with the Arabic data, but far less than the English data with the next two models.

However, MFCCs performed even worse on this dataset, strengthening the observation that this combination failure was not just due to chance. As can be seen in Figures 4.3 and 4.4, the training cuts out around 80000 timesteps. Eventually, loss was NaN and optimisation was not possible, just like when using ΔSCC.

4.2 CNN Approach

The one-dimensional CNN approach is generally quite good. However, it is prone to overfitting. How this played out is outlined below.

Feature Type	MGB-3		FRED-S	
	Accuracy	Loss	Accuracy	Loss
MFCC	0.64	1.62	0.64	2.17
Mel-Spec	**0.66**	1.60	**0.70**	1.55
MFCC + ΔSCC	0.47	1.57	0.61	2.17
Mel-Spec + ΔSCC	0.47	2.32	0.69	1.04

Table 4.2: Results for CNN Model

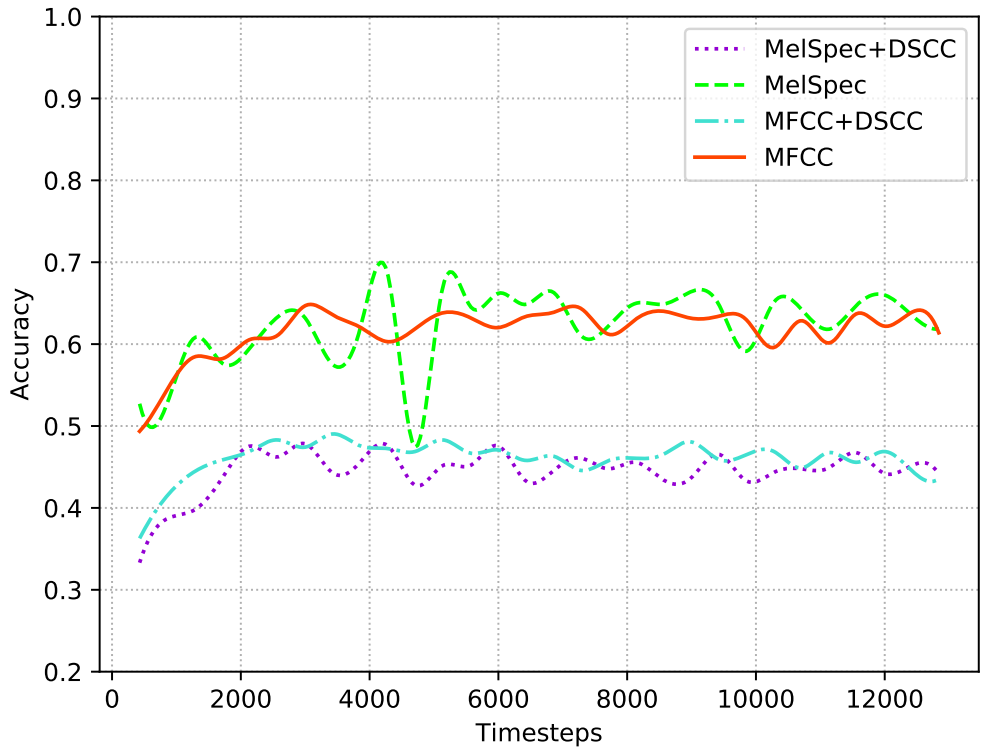


Figure 4.5: CNN development accuracy for MGB-3

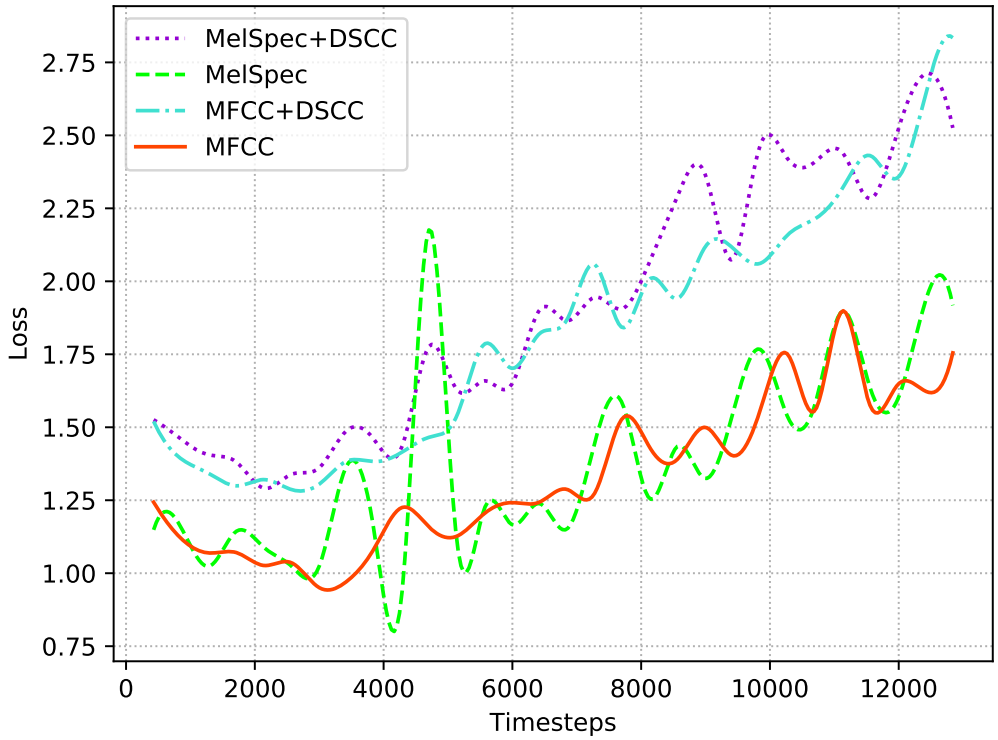


Figure 4.6: CNN development loss for MGB-3

MGB-3 Results The results from the CNN model proved to be far more reliable than from the RNN model as far as accuracy is concerned. As can be seen in Figure 4.5, the development accuracy seems to plateau for all feature types, but with Mel-Spectrograms coming out on top.

However, the loss as seen in Figure 4.6 ceases to decrease after around 3000 steps and actually steadily increases, suggesting that a 1-dimensional CNN model may either not be ideal for the task or that heavier regularisation measure must be taken due to overfitting.

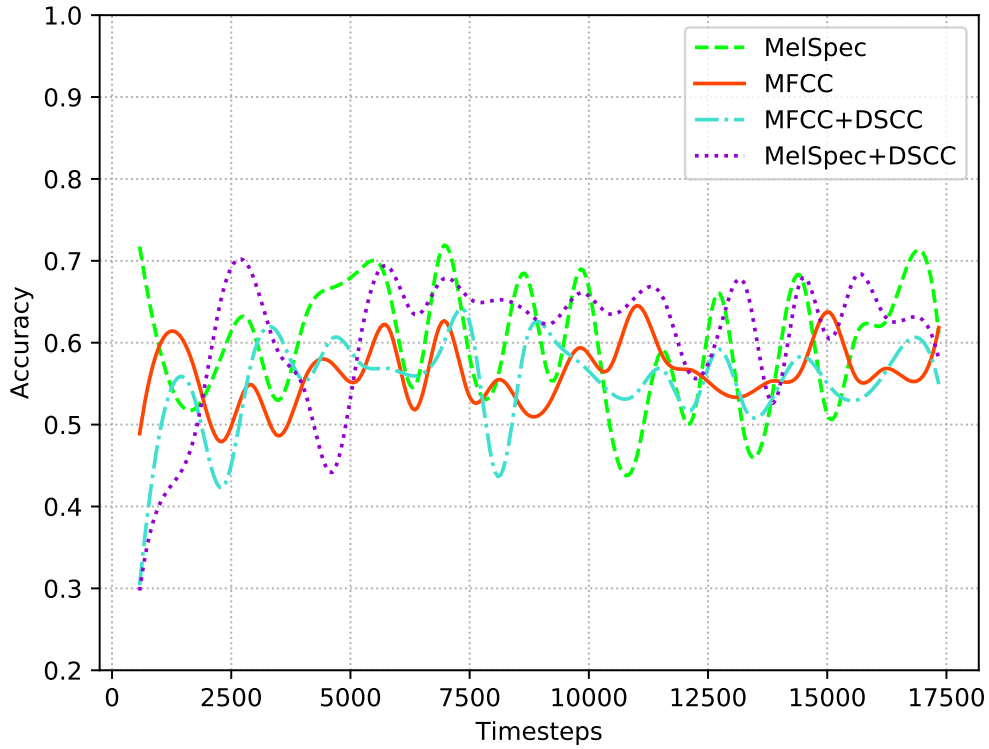


Figure 4.7: CNN development accuracy for FRED-S

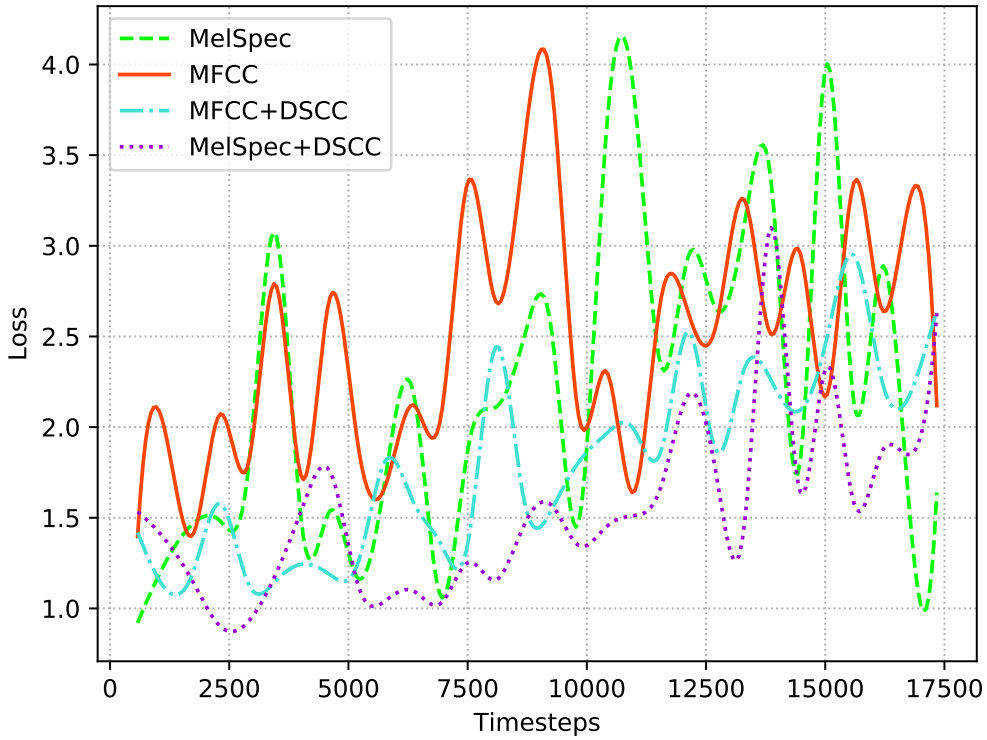


Figure 4.8: CNN development loss for FRED-S

FRED-S Results The results of the model when used on the FRED-S corpus are quite different than from the MGB-3 corpus. Development accuracy does not seem to increase much for either model over the course of training. Some classification accuracy clearly works, but there is quite a lot of variance during training.

Mel-Spectrograms are also better than MFCC, but only slightly.

Figure 4.8 shows that there is a slight upward trend in the loss as well, but with quite a lot of variance. This is further evidence that the CNN model is susceptible to overfitting. The actual final values of the loss, however, are not great, just as with the MGB-3 models.

4.3 Fusion Approach

The fusion model is the definite winner out of the three tested. Having both the feature selection of the CNN and the time information from the RNN clearly was beneficial. It was still susceptible to overfitting, but not nearly as much as the previous model.

Feature Type	MGB-3		FRED-S	
	Accuracy	Loss	Accuracy	Loss
MFCC	0.79	0.76	0.66	1.90
Mel-Spec	**0.82**	0.66	**0.81**	0.81
MFCC + ΔSCC	0.67	1.04	0.67	2.45
Mel-Spec + ΔSCC	0.56	1.25	0.71	1.58

Table 4.3: Results for Fusion Model

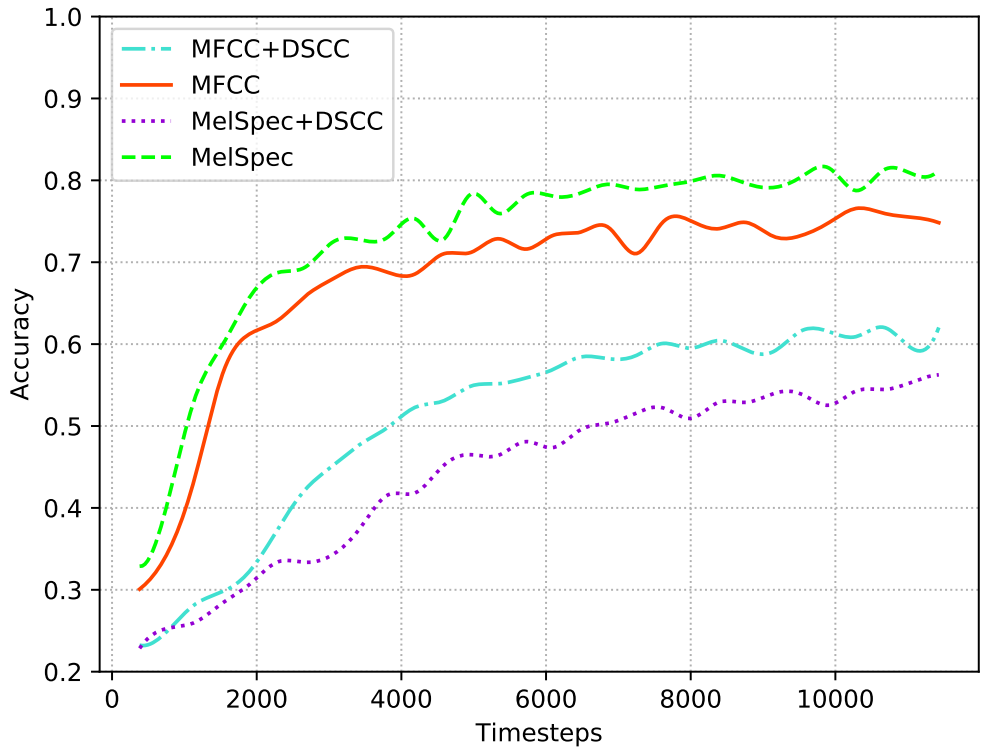


Figure 4.9: CNN+RNN development accuracy for MGB-3

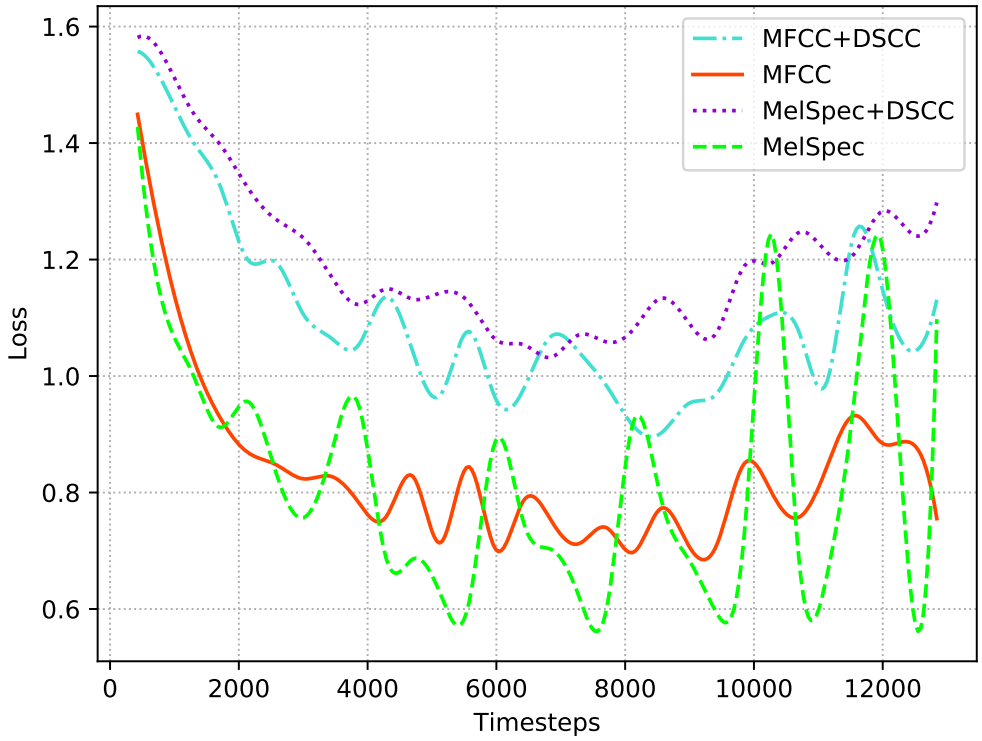


Figure 4.10: CNN+RNN development loss for MGB-3

MGB-3 Results The fusion model with Mel-Spectrogram features was the best for the MGB-3 dataset. In Figure 4.9, a steady upward trend can be seen in the development accuracy during training for all features. As was seen in the CNN model, Mel-Spectrograms and MFCC alone prove to be better than the models with ΔSCC added.

Figure 4.10 also shows a steady downward trend in the loss during training. There does seem to be some overfitting in the model around 8000 timesteps, but I believe this could be fixed with a larger network and more regularization.

This, in combination with the high accuracy scores prove the fusion model to be the best for the task.

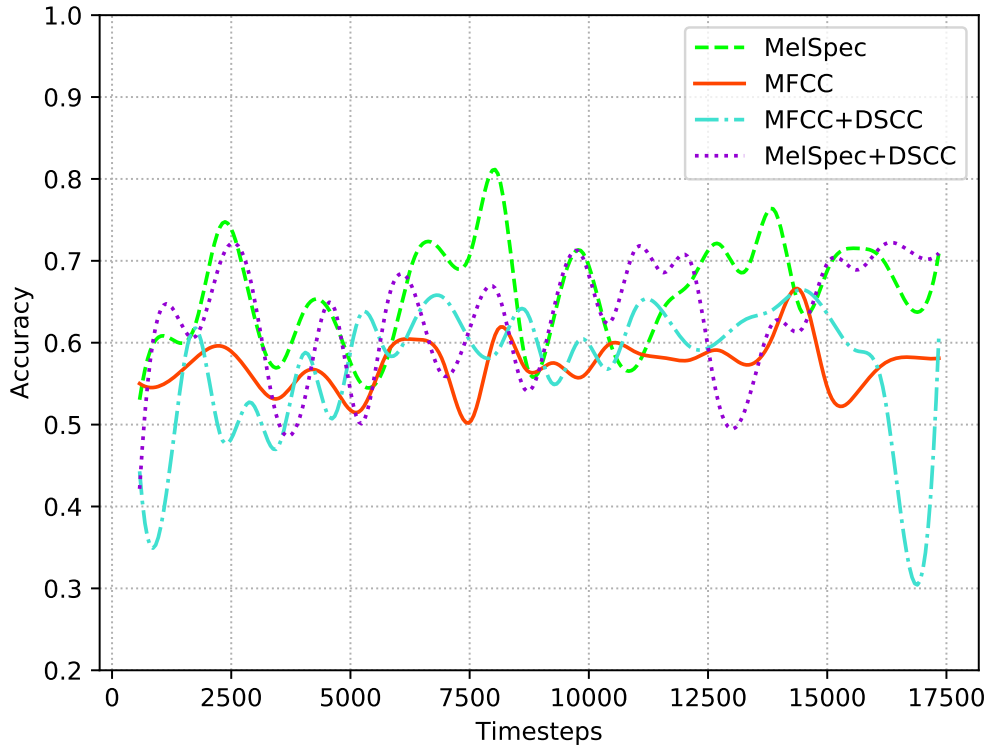


Figure 4.11: CNN+RNN development accuracy for FRED-S

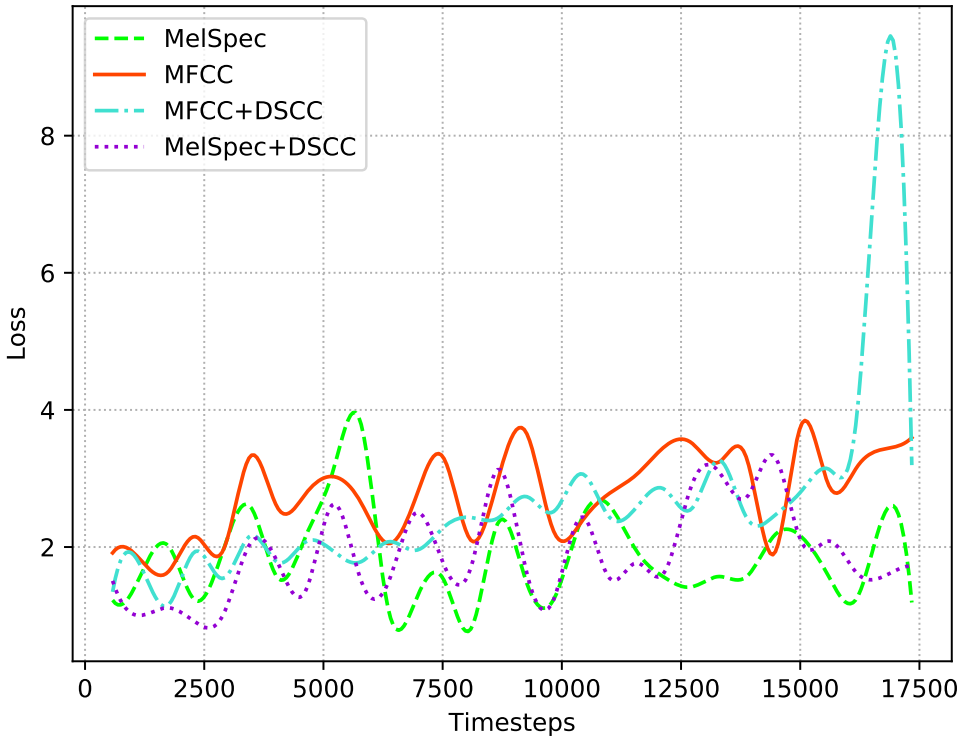


Figure 4.12: CNN+RNN development loss for FRED-S

FRED-S Results Although the results for the fusion FRED-S models are just as messy as for the CNN models, the accuracy in Figure 4.11 and loss in Figure 4.12 are clearly superior. There is even a more noticeable upward trend in development accuracy when using Mel-Spectrograms.

The same pattern as for the MGB-3 dataset follows here as well, but with one difference. MFCC performs the worst in this model. The performance of the hybrid model on both datasets supports the assumption that this architecture is the best of the three.

5 Discussion

I will now discuss the study process and results, sharing observations and final thoughts and conclusions. In this final section, the surprising and expected results will be detailed for the different models and the acoustic features. Then, the dialects will be compared and contrasted using misclassification as a metric.

5.1 Models

I am rather pleased with the result of this study overall. The final classification accuracy was much better for both corpora than I had expected.

During this study, 3 different model architectures were tested to see which would be a better fit for the task of DID: A RNN, a CNN, and a hybrid between the two.

At the beginning, I had both expected and hoped that a hybrid CNN and RNN would perform the best at the DID task, and it did. Logically, it makes sense that a CNN would be able to distil distinguishing features from the acoustic data, just as it has been shown to work for image recognition.

However, unlike image recognition, the acoustic features are bound and affected by time. Therefore, using a one-dimensional CNN preserves this relation to a degree, along with being computationally more efficient. These features can then be used with a RNN to process the acoustics as a time-bound sequence and classify them as belonging to one dialect or another.

This combination uses more features than just a CNN or RNN alone, and the effects of that are seen in the results. And, from the graphs in Figures 4.9 and 4.10, I am confident that, with a larger network to prevent overfitting and more time, an even robuster model could be trained.

I had expected the CNN model to perform better than it did. Perhaps more regularization measures would have prevented overfitting and produced a better model which could have been more competitive with the hybrid model. This is supported by the results from Shon et al. (2018).

The RNN model, however, was surprising to me. I did not expect this model alone to produce competitive results. Using the time information and the raw Mel-Spectrogram features produced results which were on par with the CNN model, if not better and less affected by overfitting. Although the loss for the CNN model was able to get lower, perhaps more training for the RNN would have produced better results. Also, further investigation is needed to see why MFCC did not work well with the RNN. I suspect that the compression of information, which would work well for speech recognition, does not allow the model to find the fine grained dialectal differences.

What was also really unexpected about the RNN model was how well it worked for the English data. Because of the chaotic training for the CNN and Hybrid models, I almost did not train an RNN model on the FRED-S corpus to save time. However, even though it did not get the top accuracy, the model had the least chaotic training. By least chaotic I mean that there was minimal variation, and a steady upward trend in training, at least when Mel-Spectrograms were used.

This suggests that an RNN might be much better and more efficient when working with very messy and noisy audio data. The CNN is quite powerful

in picking up and identifying different features, however it reduces and compresses the data a bit. The acoustic variations in dialects might be too subtle for a small CNN. Even the RNN model for the MGB-3 data performed well in this regard which strengthens the idea that it is a good tool for DID.

In the end, I believe that a RNN and a CNN are each able to pick up different features which are useful in the DID task, but combining both of them is far more advantageous.

5.2 Acoustic features

4 different feature types were used as input into the model for this study: MFCC, Mel-Spectrograms, MFCC + ΔSCC, and Mel-Spec + ΔSCC.

For most speech recognition tasks, MFCC features have been proven to be superior to Mel-Spectrograms for a number of reasons. First, the data is more compressed, which leads to faster and more efficient training. Second, because the features are so distilled, it can generalize as to which phoneme is being uttered, and is thus less sensitive to noise and other influences such as accent and idiosyncratic differences.

Because of the overwhelming success and prevalence of MFCC features in speech recognition, I had expected them to be ideal for acoustic DID as well. However, the evidence clearly shows that this is not the case. MFCC features do work, however, the more generalized and less processed Mel-Spectrograms work much better. My theory is that they work well for the exact opposite reason that MFCCs work well for speech recognition: they contain more information about things like accent and inflection. Being more general and having more correlated information (40 bands as opposed to 13), the models could pick up the dialectal differences a bit better.

Furthermore, I had expect that the addition of ΔSCC to the model would improve classification, but the opposite was true. For speech recognition, the addition of this dynamic information greatly improves model accuracy. However, in my DID models, it had a detrimental effect, which was disappointing.

Mel-Spectrogram features alone were proven during this study to be the ideal acoustic features for DID.

5.3 Dialect Analysis

A side benefit of machine learning classifiers is that they can sometimes show the similarity and correlation between different classes. When one class is mislabelled as another class enough, it is usually because the two classes are closer and more similar that the others.

In the case of DID, if a dialect is mislabelled as another dialect enough, it

can be assumed that they have a lot in common. In our case, a lot of acoustic similarities.

The misclassifications can be represented with a confusion matrix. In the following two confusion matrices, the actual dialect of a file is on the y axis and the predicted dialect is on the x axis. The percentage of actual dialect per row was used rather than just the number to normalize the data and make it more readable. Percentages were rounded up. For example, in Figure 5.1, Egyptian Arabic is correctly labelled 81% of the time, whereas it is misclassified as Modern Standard Arabic 10% of the time.

The diagonal line which crosses from the top left down to the bottom right shows the percentage of a dialect which was classified correctly by the model.

The hybrid model using Mel-Spectrogram features was used to create the confusion matrices for both corpora as it was the best model. With each having only around 18% classification error, there is not an extremely strong leaning of one dialect towards another, but some correlations appear nonetheless.

It must be noted that, although this is a very useful and interesting tool, it is not definitive proof that two dialects are similar or dissimilar. A separate and independent study would be needed to prove this. In addition to that, I am not a speaker of Arabic and am not greatly familiar with English dialects as I am from the USA.

5.3.1 MGB-3 classification

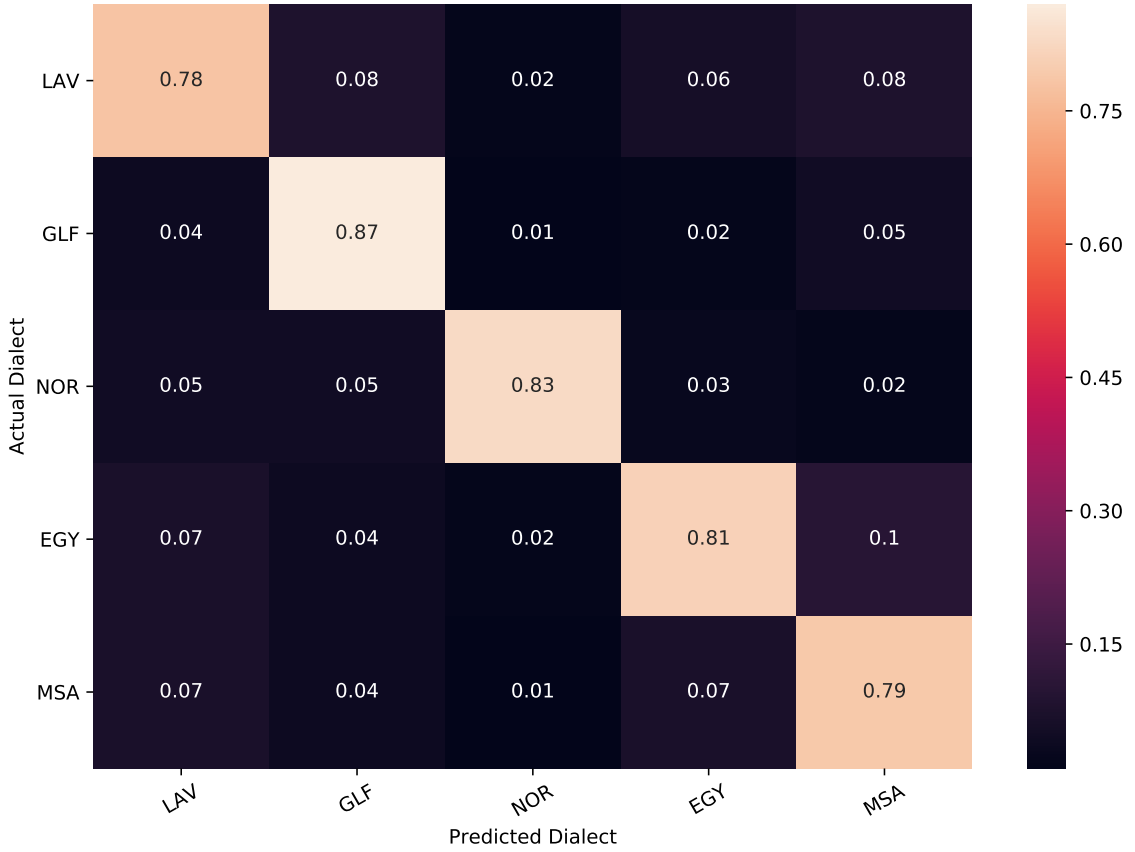


Figure 5.1: Confusion matrix for Arabic predictions with the hybrid model and Mel-Spectrogram features

For the MGB-3 dataset, the first thing to pop out from the confusion matrix in Figure 5.1 is that the Egyptian dialect has been mislabelled as Modern Standard Arabic quite often, and vice versa. As was stated above in the example, Each is missclassified as the other between 7-10% of the time. This makes sense to me because Egyptian Arabic is used in a large majority of Arabic film, music, news, and television, which might have an effect on the standard varieties.

The Levant dialect was misclassified often as well and had the lowest classification accuracy out of all the dialects, which may mean that there are not as many unique features. The only dialect it was not misclassified a significant amount was the Northern African dialect.

The Northern African and Gulf dialects had the highest classification accuracy and this is seen in the matrix, indicating that they might be the most

unique. Modern Standard Arabic, Lavant, and Egyptian dialects, on the other hand, seem to have many acoustic similarities in comparison.

5.3.2 FRED-S classification

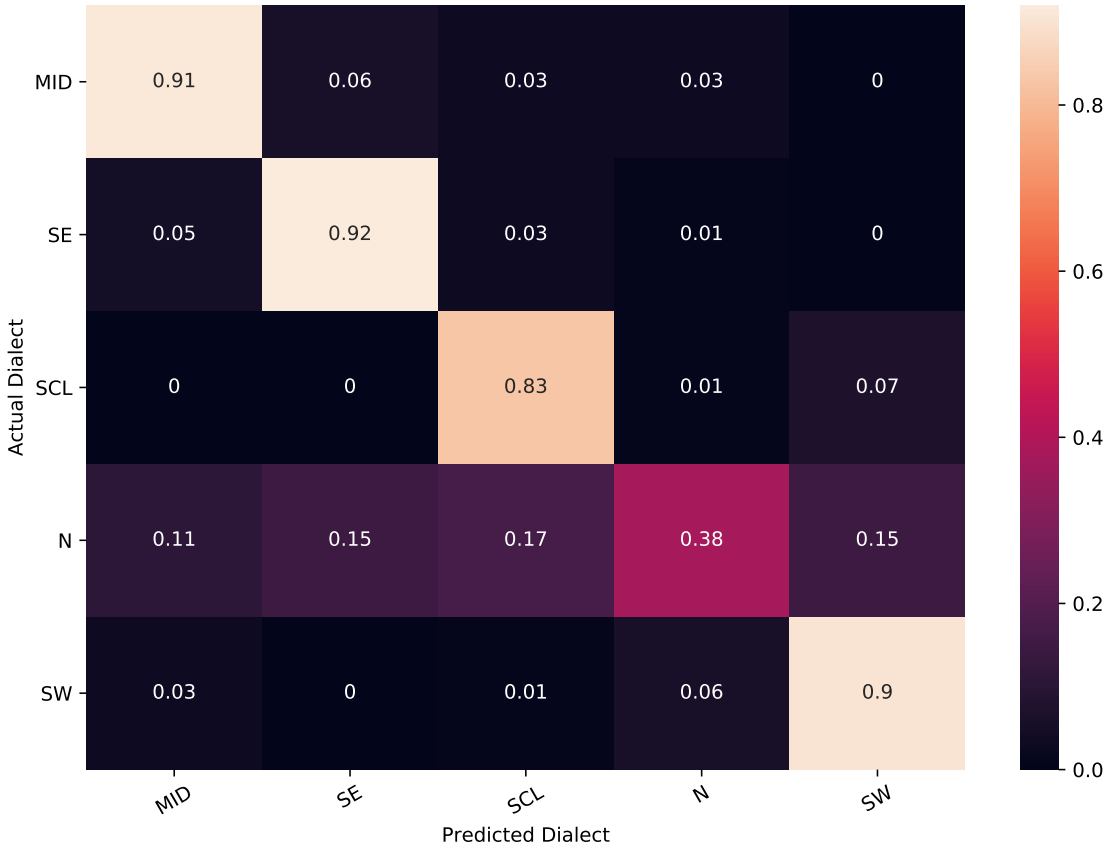


Figure 5.2: Confusion matrix for English predictions with the hybrid model and Mel-Spectrogram features

The first and very evident thing to jump out of the confusion matrix in Figure 5.2 is that the model had a very difficult time with the Northern dialect. I believe that this may be because the Northern dialect covers quite a large space and several counties. This might also be due to the fact that it is in the middle of other dialectal regions, so there might be some overlap.

It is not surprising that the highest percent of Northern misclassifications (17%) were as the Scottish Lowland dialect, given it's close proximity.

I had actually expected the Scottish Lowland dialect to have the highest classification accuracy because Scottish is usually hard to outsiders to understand, but this was not so. However, with significantly less data than the

other dialects, it makes a bit more sense. With more data, I believe that the classification accuracy for this one would have been much more interesting.

6 Conclusion

In this study, different methods for the task of dialect identification were tested. First, 3 different neural network architectures were used: a Recurrent Neural Network, a Convolutional Network, and a hybrid between the two. Also, 4 different audio processing techniques were used: MFCC, Mel-Spectrograms, MFCC + ΔSCC, and Mel-Spec + ΔSCC. Two datasets and languages were used as well: the MGB-3 corpus of Arabic dialects and the FRED-S corpus of English dialects.

A hybrid CNN+RNN model with Mel-Spectrogram features was shown to work the best for the task. A simple RNN was also a simple and far less computationally intensive method as well. A CNN alone was more prone to overfitting.

Adding ΔSCC features to the model had a negative effect on classification, which is not true for speech recognition.

The study ended with a qualitative comparison of the dialects using misclassification as a metric.

7 Appendix

All code used in this study can be found here: `https://github.com/ryancallihan/thesis_project`

8 Acknowledgements

First, I would like to thank Kiki. Being able to move to Tübingen to take part in this program would not have been possible without her. She put in just as much work as I did.

Thanks to Mai Mayeg for answering my Arabic questions multiple times.

I would also like to thank my advisers for their help and direction, answering all my questions and addressing my concerns, even when swamped with work themselves.

9 References

Bibliography

Ahmed, N., Natarajan, T., and Rao, K. R. (1974). Discrete cosine transform. *IEEE Transactions on Computers*, C-23(1):90–93.

Ali, A., Dehak, N., Cardinal, P., Khurana, S., Yella, S. H., Glass, J., Bell, P., and Renals, S. (2016). Automatic dialect detection in arabic broadcast speech.

Bahari, M. H., Dehak, N., Van Hamme, H., Burget, L., Ali, A. M., and Glass, J. (2014). Non-negative factor analysis of gaussian mixture model weight adaptation for language and dialect recognition. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 22(7):1117–1129.

Cho, K., van Merrienboer, B., Gülçehre, Ç., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078.

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Dehak, N., Kenny, P. J., Dehak, R., Dumouchel, P., and Ouellet, P. (2011). Front-end factor analysis for speaker verification. *Trans. Audio, Speech and Lang. Proc.*, 19(4):788–798.

Furui, S. (1986). Speaker-independent isolated word recognition based on emphasized spectral dynamics. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'86.*, volume 11, pages 1991–1994. IEEE.

Glembek, O., Burget, L., Matejka, P., Karafiát, M., and Kenny, P. (2011). Simplification and optimization of i-vector extraction. pages 4516 – 4519.

Harris, F. J. (1978). On the use of windows for harmonic analysis with the discrete fourier transform. *Proceedings of the IEEE*, 66(1):51–83.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Huang, X., Acero, A., and Hon, H.-W. (2001). *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pages 1097–1105, USA. Curran Associates Inc.

Kumar, K., Kim, C., and Stern, R. (2011a). Delta-spectral cepstral coefficients for robust speech recognition. pages 4784–4787.

Kumar, K., Kim, C., and Stern, R. M. (2011b). Delta-spectral cepstral coefficients for robust speech recognition. In *ICASSP*, pages 4784–4787. IEEE.

Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pages 807–814, USA. Omnipress.

Oliphant, T. E. (2006). *Guide to NumPy*. Provo, UT.

Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Hannemann, M., Motlicek, P., Qian, Y., Schwarz, P., Silovsky, J., Stemmer, G., and Vesely, K. (2011). The kaldi speech recognition toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society. IEEE Catalog No.: CFP11SRW-USB.

Sahidullah, M. and Saha, G. (2012). Design, analysis and experimental evaluation of block based transformation in mfcc computation for speaker recognition. *Speech Communication*, 54(4):543 – 565.

Shon, S., Ali, A., and Glass, J. R. (2018). Convolutional neural networks and language embeddings for end-to-end dialect recognition. *CoRR*, abs/1803.04567.

Sigurdsson, S., Petersen, K. B., and Lehn-Schiøler, T. (2006). Mel frequency cepstral coefficients: An evaluation of robustness of mp3 encoded music.

Stevens, S., Volkmann, J., and Newman, E. (1937). *A scale for the measurement of the psychological magnitude pitch.*

Szmrecsanyi, B. and Hernández, N. (2007). Manual of information to accompany the freiburg corpus of english dialects sampler fred-s.

Wray, S. and Ali, A. (2015). Crowdsource a little to label a lot: Labeling a speech corpus of dialectal arabic. In *Sixteenth Annual Conference of the International Speech Communication Association.*