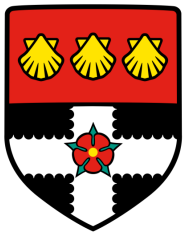


University of Reading

Department of Computer Science

Machine Learning for Speech (English Language Accent) Classification

Jagjeevan Singh Shergill

25011925

Supervisor: Dr. Varun Ojha

A report submitted in partial fulfilment of the requirements of
the University of Reading for the degree of
Bachelor of Science in *Computer Science*

May 11, 2020

Declaration

I, Jagjeevan Shergill, of the Department of Computer Science, University of Reading, confirm that all the sentences, figures, tables, equations, code snippets, artworks, and illustrations in this report are original and have not been taken from any other person's work, except where the works of others have been explicitly acknowledged, quoted, and referenced. I understand that if failing to do so will be considered a case of plagiarism. Plagiarism is a form of academic misconduct and will be penalised accordingly.

Jagjeevan Shergill
May 11, 2020

Abstract

During speech classification, machine learning techniques are applied to classify and predict user input. Advanced voice recognition is the technology of tomorrow. Many businesses rely on data to drive business growth. Thus far, although voice recognition is common, it is rare to classify metadata about the audio. So, the question derives, *what is being done about accent recognition, if anything?* This project is concerned with taking user input in the form of speech data, to classify and then predict which region of the United Kingdom the user is from, and also what gender they are. Research was conducted on regional accents, machine learning models, data pre-processing, Fourier Transforms and data visualisation techniques. It became evident that there is a lack of industrial-sized datasets for this type of project, so a dataset was created from scratch (12 regions with 1:1 gender split). Python was used to develop this project in Google Colab. In this paper, the final project solution is proposed with a discussion of how it is possible to use machine learning to classify human voice by accent and gender, and then exposing this information graphically. This project has the potential to enhance current voice recognition systems, by adding functionality to categorise speakers' accent and gender, in turn creating a more immersive user experience. This metadata allows for extra dimensions within business intelligence as it is potentially valuable data that can be leveraged by, for example, marketing and recommendation-based systems.

Keywords: Regional Accents, Speech Classification, Machine Learning

Report's total word count: 16,456.

Acknowledgements

Starting the project in the summer before final year, I had high ambitions to implement a unique and interesting project. Meeting with my supervisor, Dr. Varun Ojha, multiple ideas were noted down which best suited my interests and strengths. The investigation I decided to follow up on was that of *Machine Learning for Speech (English Language Accent) Classification*. Being made aware of the difficulty of implementing such a task, I was warned that it would demand a lot of perseverance and thorough research to achieve a final solution.

Having said this, I would like to express my gratitude to Varun for his continuous support and attention to detail, ensuring I remained on the right track and offered excellent advice when I ran into obstacles. In addition to this, during the unexpected closure of The University due to COVID-19, Varun was able to adapt instantly to ensure that his students were not at a disadvantage and moved his assistance onto a virtual platform.

I would also like to thank my family, for their unconditional support during my time at the University of Reading.

Contents

List of Figures

List of Tables

Listings

List of Abbreviations

ANN	Artificial Neural Network
API	Application Programming Interface
CNN	Convolutional Neural Network
DFT	Discreet Fourier Transfrom
EE	East of England
EM	East Midlands
FFT	Fast Fourier Transfrom
GL	Greater London
GPU	Graphics Processing Unit
HMM	Hidden Markov Model
IDE	Integrated Development Environment
LSTM	Long Short Term Memory
ML	Machine Learning
MLP	Multi Layer Perceptron
NE	North East
NI	Northern Ireland
NN	Neural Network
NW	North West
RNN	Recurrent Neural Network
SC	Scotland
SE	South East
SGD	Stochastic Gradient Descent
SMART	Specific, Measurable, Achievable, Realistic, and Timely
STFT	Short-Time Fourier Transform
SW	South West
UK	United Kingdom
WA	Wales
WM	West Midlands
YH	Yorkshire and the Humber

Chapter 1

Introduction

In the information age, voice recognition is becoming increasingly more important. Users are rapidly adopting voice recognition systems, whether it be within their handheld devices or home assistants such as Amazon Alexa. The high volume of metadata created through each interaction with a voice recognition system is incredible, and there is so much potential to leverage this data and do more with it, other than just simply recognising words and commands. *Machine Learning for Speech (English Language Accent) Classification* is a classification problem, and this project investigates if machine learning can expose this speech metadata and thus allowing a new generation of software to exploit this.

1.1 Background

In 1911, the first voice recognition system was released; a toy named "Radio Rex" (FZ Blogs, 2011). Although a very basic system, this paved the way for research that would go onto change how one interacts with technology on a day-to-day basis. 100 years later, Apple released Siri (part of the iPhone 4S release), illustrating how the complexity and potential of voice recognition had evolved. This release captured the imagination of the public, and also showed what could be achieved using this technology (The Economic Times India, 2011). Since then, multiple voice recognition technologies have been incorporated into many systems, most notably home assistants such as Google Home and Amazon Alexa. Many companies have released different forms of voice recognition systems, trying to establish themselves in this competitive space. In reality, this is only the start for voice recognition systems and what can be achieved with speech data, and in turn how it can provide a more immersed experience for the user.

These new additions to voice recognition systems are fascinating and shows what can be done using speech data. Although these developments to voice recognition technologies benefit the user, more could be done with the metadata that can be retrieved from speech, to give the user a more personalised experience. This project aims to leverage machine learning techniques to be able to predict the regional accent (of the United Kingdom) and gender of a speaker. The introduction of these two types of classification from speech could be tied into systems to mine valuable data or to customise a personal user experience. Examples of this include recommendation engines that make use of this technology, such as being able to advise online users on possible communities to interact with based on their accents and gender, perhaps in conjunction with other data, such as hobbies and interests. There is also scope to feed this classification into business decisions, such as marketing.

1.2 Aims and Objectives

The aims and objectives of this project are presented below.

1.2.1 Aims

- To train an appropriate machine learning model for speech classification of the United Kingdom, namely regional accent and gender of the speaker.
- To present the results using data visualisation techniques.

1.2.2 Objectives

- To collect and investigate speech data to formulate a dataset with a 1:1 ratio of male and female from each region of the United Kingdom
- To apply data pre-processing techniques to the dataset
- To investigate and implement a suitable set of machine learning algorithms for speech data
- To train and test the implemented neural network model and assess the network's performance
- To visualise the neural network's outputs using data visualisation tools

A high-level flow chart of the aims and objectives is presented in Figure 1.1 (Pentathlon GB, n.d.).

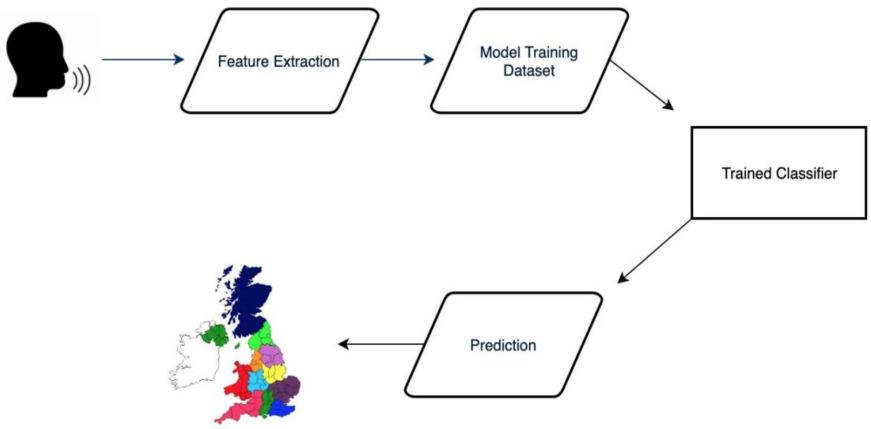


Figure 1.1: Flowchart of Aims and Objectives

1.3 Problem Statement

The purpose of this study is to establish if one can take speech data from a user and predict which regional accent that user belongs to within the United Kingdom, and also what gender they are. Mathematically, for a given input speech signal $\mathbf{x}_i = (x_0, x_1, \ldots, x_t)$ paired with a target class y_i for $i = 1, 2, 3, \ldots, n$ training examples, where $y = \{\text{Male Voice}, \text{Female Voice}\}$

for speech to gender classification or $y = \{a_1, a_2, \ldots, a_{12}\}$ for 12 regional accent classifications, the aim is to minimise a classification error rate $\mathcal{L}$ to train a classifier $f(X, \mathbf{w})$ as per the following:

$$\mathcal{L}(X, f(X, \mathbf{w})) = \sum_{i=0}^{n} (f(x_i, \mathbf{w}) \neq y_i) \tag{1.1}$$

where $X = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]$, $Y = [y_1, y_2, \ldots, y_n]$, and $\mathbf{w}$ is the classes parameters. It is hoped that a trained classifier $f(X, \mathbf{w})$ with low error rate $\mathcal{L}$ would correctly classify a users accent and gender.

1.4 Solution Approach

In order to reach the final solution, this problem was broken down into constituent parts, such that each part could be investigated and completed individually in order to build up to the final complete solution. Each subtask at a high level is discussed below:

1. Literature review
    - Research was conducted on topics that surrounded the subject matter. The goal was to build a solid foundation of the project prior to implementation, in terms of researching and comprehending the theoretical concepts that underpin the machine learning behind voice recognition. This is a complex area of expertise and thus required thorough analysis.
2. Obtain or create a dataset
    - Extensive research was carried out into finding an appropriate industry-sized dataset to be used for training and prediction purposes. Unfortunately, there are none available which would work for this type of project. To solve this issue, a dataset containing audio files of individuals from the UK regions will be collected, in an even ratio of male and female to ensure that there is a balance in data that does not skew the predicted output during development.
3. Decide on an appropriate learning algorithm
    - It is essential to find the optimal neural network model that can be trained to classify audio into the required classes. Audio is feature rich, and any algorithm must be able to recognise and separate these features. It is felt that a simple neural network such as MLP lacks the ability to accurately classify an audio dataset. This project will start with a simple convolution neural network, and add layers as needed to this basic starting point. In addition, the model will need to be tuned and verified.
4. Decide on how to visualise the results
    - In general, data science can produce complex numerical outputs that can be difficult to digest, but there are APIs available to visualise the same results using graphics to aid comprehension. In keeping with that philosophy, this project will aim to visualise the accent and gender prediction by colour encoding a map of the UK regions.
5. Decide on an appropriate implementation language and software development tools

- For this project, it was important to identify both an integrated development environment and development language that has built in and easy to use support for building machine learning models. Google Colab was chosen as the Integrated Development Environment (IDE); this has built-in support for machine learning algorithms, it uses remote virtual machines with Graphics Processing Units (GPU) allowing fast development without using local resources and allows sharing of the project code with interested parties (Medium, 2018*a*). The programming is carried out in Python and imports libraries readily available in Google Colab such as TensorFlow and Keras.

The above approach allowed for clear aims and objectives to be outlined. Chapter 3 will provide further details of this solution approach.

1.5 Summary of Contributions and Achievements

1. Create a purpose-made dataset of audio files.
    - Research all regions and how their accents distinguish themselves from other regions.
    - Find individuals from each region.
    - Download audio samples, sourced from public YouTube videos. The data consisted of the 12 regions within the United Kingdom and a 1:1 ratio of male and female per region.
    - Upload data to Google Drive where each file could be pre-processed in readiness for input to learning model, such as was trimming length or resampling for consistency across samples.
2. Transform data into frequency domain.
    - Raw representation of the audio does not lend itself well to analysis by speech recognition machine learning models, it does not illustrate any information of frequencies present. By taking a Fourier Transform of the raw audio samples, it allows the audio to be transformed to a format that is amenable to a machine learning model. The audio is converted from the time domain into individual spectral components, providing a frequency breakdown of the signal, exposing information about magnitudes and phases of different frequencies.
3. Build a machine learning model.
    - Build and define a machine learning model, using Keras and TensorFlow (with a train and validate model using a 80:20 split of the dataset).
4. Test with real-time user audio.
    - Obtain user audio sample via the microphone of the local computer.
    - Predict accent and gender of the speaker using the learning model.
    - Visualise the accent and gender prediction by a colour-encoded UK map.

The in-depth implementation details will be discussed further in Chapter 3.

1.6 Organisation of the Report

This report is organised into seven chapters. Chapter 1 provides an introduction, detailing the overall aims and objectives of the project along with the approach taken to the solution. Chapter 2 details the literature review of this project, providing the necessary theoretical background to lay the foundation for the implementation. The methodology discussed in Chapter 3 details the algorithms used, implementation of all aspects of the project, testing and verification and also the experiments carried out. Chapter 4 details the results of these experiments, followed by discussion of the results in Chapter 5. The conclusion and suggestions of future work are given in Chapter 6.

Chapter 2

Literature Review

This section discusses all aspects of the project. A description of UK accents and gender specific voice characteristics is given, discussing what they have in common, but also the differences that make each unique. A history of voice recognition is provided, including the competitors that exist in today's market. The different types of machine learning algorithm are presented, along with the possibilities of development environments for implementing these algorithms. Dataset sources are discussed along with analysis of suitability of these.

2.1 Regions and Accents Within the United Kingdom

To implement this project, it must first be understood how the United Kingdom is divided into regions and how each region differentiates itself from the next. Below, the findings of the UK regions will be discussed. These findings were established through extensive research on each region, using various online resources, but is well overviewed by the following sources: (YouTube, 2014) and (Netherlands Worldwide, n.d.). The reason for which this is important is because it is a key component of the implementation phase where the audio files are pre-processed and transformed into a form that can characterise speech traits, for training and prediction purposes.

The UK is incredibly diverse in accents. In total, there are 12 regions. These regions are:

1. Northern Ireland – 1 region of the UK, with 5 cities.
    - *Traits:* Focussed mainly on the pronunciation of vowels (*a, e, i, o, u*). This distinguishes their accent from other regions in the UK. In addition, those from this country also tend to shorten the length of their vowels.
2. Scotland – 1 region of the UK, with 7 cities.
    - *Traits:* Typically, Scottish accents are very strong and thick in comparison to the rest of the UK. They tend to cut off words mid-pronunciation, hence having shorter sentences. When pronouncing certain words, there is an emphasis on the 'R', done by slightly rolling the tongue. The letter 'G' is also dropped from a lot of words, an example being 'mornin' rather than 'morning'.
3. Wales – 1 region of the UK, with 6 cities.
    - *Traits:* The Welsh have their own currently practiced language, hence the uniqueness of the pronunciation in certain words where they tend to prolong the length of them. In addition, many Welsh people turn words with an 'I' into an 'E' and emphasise the 'R' by rolling their tongues on pronunciation.

4. England – 9 regions within this country, holding 51 cities.

   (a) North East

   Traits: The accents from the North East is very recognisable – in particular those from Tyne and Wear. Some consonant sounds such as an 'R' is similar to the way the French pronounce it. An example of this being 'Laurgh' instead of 'Laugh'. In addition to this, the vowel sounds are what tends to separate the North East from other regions in England. Examples of this being 'Broon' instead of 'Brown'. Finally, the way in which certain pronouns are said has been adapted overtime by those in this region. Examples being 'yous' rather than 'you'.

   (b) North West

   Traits: The North West have distinctive accents and it is often easy to tell if someone is from there. The influence of these accents comes from the arrival of mass Welsh and Irish residents. Those from this region join the 'ng' together (i.e. king, ping, ring) and pronounce words ending in a 'G' with a hard 'G'.

   (c) Yorkshire and the Humber

   Traits: Those from this region use predominant sounds like 'York' and cut off words mid-pronunciation, therefore making sentences shorter.

   (d) East Midlands

   Traits: The East of Midlands does not hold one distinct accent. Their dialect changes across the region, depending on which part (north, east, south, west) the individual is from. Based on the location, the accent will begin to sound slightly similar to its neighbouring region. The East Anglian English of Cambridge (East of England region) has an influence on the east part of this region. Lincolnshire (in East Midlands) is a great example as it can often sound very similar to East Anglia. The north part of East Midlands has influence from the Yorkshire accent. Just as the south part of East Midlands can sound like a South East accent, such as Aylesbury, Buckinghamshire. The west side of East Midlands (Leicestershire and the North West of Derbyshire) can sometimes be confused with West Midlands. In conclusion, linguistic distinctiveness has hugely decreased due to the influence other regions have on the East of Midlands.

   (e) West Midlands

   Traits: Most people from the West of Midlands hold a very distinctive accent, which is recognised throughout the UK. It is also home to the Black Country, a name gained due to the smoke from the thousands of ironworking foundries, forges and coal seams. In this region, the accent comes across as fairly monotone, where they also tend to use a sharp 'A' in words, creating a phonic sound. An example of this is 'A-sk', other known as 'Ask'. However, the more south of the West Midlands one goes, the more southern the accent becomes.

   (f) East of England

   Traits: Those from the East of England have a unique accent which can come across as Australian-like. Just as Australians don't drop their 'H' when speaking, neither do many from this region. Words such as 'David' is pronounced 'Dayvuhd' rather than 'Dayvidd'.

(g) Greater London

Traits: The Capital of England. Accents within London vary from Cockney, Estuary English and Urban English. Cockney originated in the East of London by the working-class. However, today it is much more widespread and is now spoken in the South East as well as London. Some letters are not pronounced, such as the 'T' and 'H'. An example of this being 'Ard' instead of saying the word 'Hard'. In addition, the vowels can also sound different. Estuary English is mainly spoken in areas towards the River Thames. This accent includes received pronunciation and some features of cockney. Received pronunciation (also referred to as BBC English) is commonly found in the South East, spoken by residents, actors and news anchors, as it is thought of as 'well-spoken'. Although, received pronunciation does not have a specific region and it can be found anywhere in the UK. Urban English is spoken mostly by the younger generations, featuring a lot of slang and an extremely upbeat voice.

(h) South East

Traits: The perception of the South East is that most speak in Cockney Estuary English. However, this can change depending on where you are from in the region, as it can reflect the bordering region also. An example of the way those from the South East pronounce words is 'Luva' rather than 'Lover'. In addition, the phonic sound of 'F', or 'th' is spoken with emphasis. An example of this is the word 'Father'.

(i) South West

Traits: This is home to the West Country. Many from this region pronounce the letter 'R' with a heavy voice, almost stereo typically farmer-like.

There is a distinct difference between those from Northern Ireland, Scotland, Wales and England - unless their accents have been diluted through living in multiple locations. In this case, they may tend to pronounce certain words in a mixed accent (Independent, 2015). Some examples of mixed accents are those who are politicians or celebrities.

Stephen Fry, a well renown English actor, comedian and writer was born in London. His family moved to Norfolk, East of England, where he was then raised. Fry later moved to London and now splits his time between both regions (East of England and Greater London) (Britannica, 2020). Through listening to his interviews, one can see that his accent is diluted between both regions mentioned. His Estuary English is noticeable, but on some words his East of England accent and emphasis of 'H' can be noted (YouTube, 2015*b*).

As one moves into England and explores its regions, it can sometimes be tricky to distinguish accents to specific regions. This is due to the overlapping of accents, and influence of its neighbouring regions. The first example of this is the East Midlands. Depending on which part of the region an individual is from, a similar accent can be found in its neighbouring region. Secondly, many individuals easily mistake London and the South East accents to be the same. This is due to the overlap of Cockney and Estuary English, something that is also spoken within the South East region (Altendorf, 2003).

In conclusion, dialects have changed massively over recent years. Accents such as heightened and standard received pronunciation are slowly becoming less common, classic regional accents are fast disappearing and the new form of dialects are fast appearing (The Guardian, 2018). This is due to multiple factors; some of which include how the new generation speak and accessibility to travel, meaning there is no geographical isolation anymore. This, therefore dilutes accents over a period of time.

2.2 Gender Traits Within the United Kingdom

Section 2.1 discusses the wide range of variation in accents within the UK. In contrast, with regards to gender classification, there are only two classes, and the variation between the genders is generally thought of as a difference in pitch. The pitch of a male voice is generally lower than that of a female voice. However, it should be noted that this is not universally true, and that in some cases, a male may have a high pitch and a female may have a low pitch, or some set of male and female voice may lie within a narrow range of pitch (Meena et al., 2013).

The frequency of the pitch is related to anatomy, and the "high" and "low" pitch of the genders may lie with a Gaussian distribution around a mean, such as 110Hz for male speech and 200Hz for female speech. Children and young adults may have a variable pitch as anatomy changes over time (Sigmund, 2008).

2.3 Voice Recognition

Voice recognition is software that is concerned with using voice to identify an input and convert that speech to on-screen text/computer command. Using the ADC (analog-to-digital converter), the wave of the speech is translated into digital data which can be understood by the computer (HowStuffWorks, n.d.).

The history of voice recognition is presented in Figure 2.1 and expanded upon below.

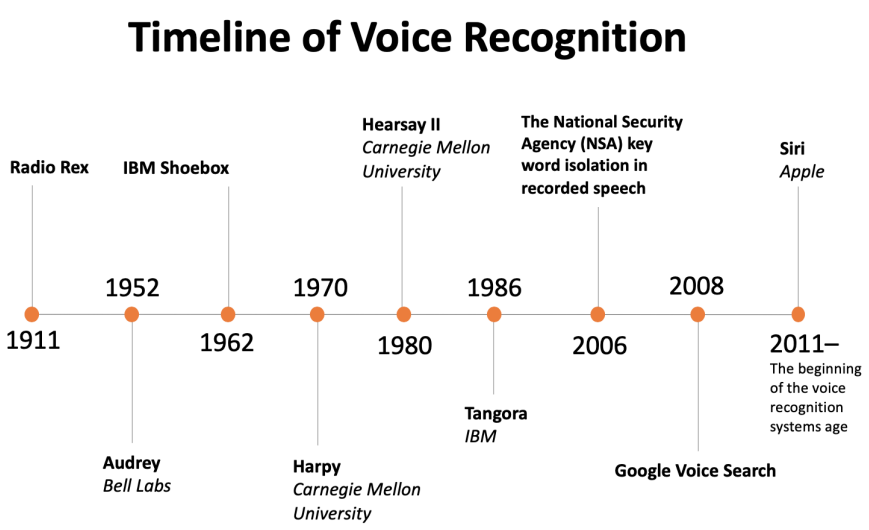


Figure 2.1: History of Voice Recognition

The history of voice recognition systems began in 1911 when a voice activated toy named "Radio Rex" was released. The toy was very simple. The dog "Rex" must be in the house and via voice input of "Rex" it will pop out of its house (FZ Blogs, 2011). However, through research, it has been noted that most words ending in "ex" will still be accepted as a valid input (YouTube, 2016).

In 1952, Audrey was released by Bell Labs. The system was able to understand digits 0-9 with an accuracy of 70-80%. However, when spoken to by the inventor, the system recorded 90% accuracy (BBC, 2017). The system worked by following 3 steps (Asta Speaks, 2014):

1. The speaker spoke their chosen digits into a telephone with a 0.35 second gap between each digit

2. The system then tries to understand the input from the user and sorts the speech sound/wave frequency into electrical classes that correlate with the correct reference patterns held within an analog memory

3. Once understood, Audrey would flash the appropriate light to show competition

Looking at the variable success rate of input to the system being understood correctly, it shows that due to individuals voice tones, accents and spoken language, it can cause a system to be inconsistent unless it is well tested with data. Nonetheless, this was ground-breaking technology when released.

A more complex voice recognition machine "capable of operating an adding machine" was presented to the public in 1962; the IBM Shoebox. It consisted of 10 spoken digits (0-9) and six command words *(plus, minus, total, subtotal, false, off)*. The 'adding machine' better known today as a calculator was able to successfully perform the arithmetic operations input into the 'Shoebox' (IBM, n.d.).

8 years later, in 1970, a tool named "Harpy" was developed at the Carnegie Mellon University (CMU) (Lowerre, 1976). Harpy was able to recognise entire sentences, mastering around 1000 words, approximately that of the vocabulary of a 3-year-old. This project was funded by the US Department of Defense's research agency, DARPA. Alexander Waibel, a Computer Science professor at CMU who worked on the project, stated, "We don't want to look things up in dictionaries – so I wanted to build a machine to translate speech, so that when you speak in one language, it would convert what you say into text, then do machine translation to synthesise the text, all in one" (BBC, 2017). This was only the start for voice recognition systems, turning voice input in form of sentences into text.

In 1980, Hearsay II was developed by the same group that created Harpy. However, this tool could analyse entire word sequences, made possible by a Hidden Markov Model (HMM) (Erman et al., 1980). A HMM 'is a statistical Markov model in which the system being modelled is assumed to be a Markov process with unobservable (i.e. *hidden*) states' (Medium, 2018*b*). Put simply, this allows prediction of the most likely phonemes to follow a given phoneme. Further, Hearsay II was also the first blackboard system (Nii, 1986). A blackboard system is an artificial intelligence-based approach using the blackboard architectural model (Corkill, 1991). Figure 2.2 illustrates a HMM (YouTube, 2015*a*).

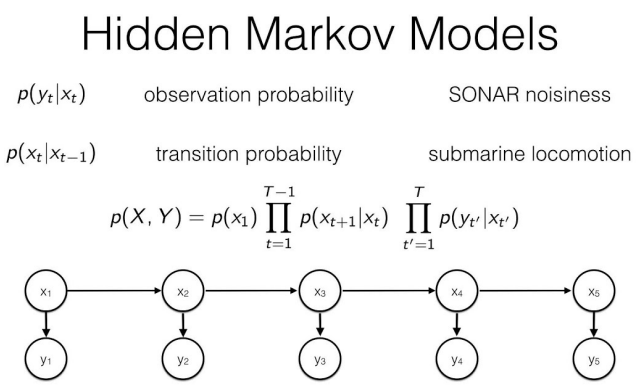


Figure 2.2: Hidden Markov Models

A few years later, in 1986, voice recognition systems evolved further. IBM developed the first real-time large vocabulary (20,000 words) typewriter, using voice-activation (Das and

Picheny, 1996). This was named "Tangora". Just as the Hearsay II, IBM's approach was based on the Hidden Markov Model (Bahl et al., 1988).

The National Security Agency developed a keyword spotting algorithm in 2006. The algorithm was designed to analyse and extract the content of conversations, flagging conversations that will be of interest (The Intercept, 2015). This was implemented by searching through mass volumes of recorded audio and monitoring when an unexpected utterance occurred (The Intercept, 2018). The audio recordings could be saved and analysed at a later time (ProPublica, 2013).

Google Voice Search is a product which allows users to use Google Search through voice command (Schalkwyk et al., 2010). Via large volumes of training data, the voice recognition software was able to show greater accuracy in delivering valid output, in comparison to other voice recognition software (Medium, 2018*c*).

Fast forward to 2011, the complexity of voice recognition had increased. Apple released a voice recognition software under the name of "Siri", on a release of a new iPhone (4S) (Apple, 2011). Siri Inc. was acquired by Apple in 2010 from SRI International (a research institute), who created Siri as a spin-off company in 2007. Just as Harpy was financed by the US Department of Defense's research agency, DARPA in 1970, so was SRI's work on the 'CALO' project, where Siri's technology was developed (Tech Target, 2018). It was this voice recognition software that captured the imagination of the general public, and what could be done using voice recognition.

Since the rise of voice recognition and its possibilities, the market has been full of different types of software, allowing all industries to benefit from them in one way or another. Home assistants can now learn and recognise international language and personalise preferences and settings for the user (Business Insider, 2018). It seems that there are almost daily new additions of features for these sorts of systems, just as this project, concerned with regional classification, could be an addition to such systems. In addition, the speech data opens up marketing opportunities for companies, based on the user audio (Venture Beat, 2019). Speech data processing is now much faster, due to powerful computer processing and an increased number of algorithms exposed by publicly available libraries (Forbes, 2018). Computational power will arguably keep increasing in line with Moore's Law, and thus opens up the possibility of increasingly complex machine learning voice recognition systems becoming widely available (Schaller, 1997). Current state of the art technologies such as Amazon's Alexa and Apple's HomePod (Siri) have been a real driving force in today's technologies of voice recognition, taking the industry possibilities even further. An excellent example of recent innovation is Google's Translatotron; there is some synergy there with this project, specifically the notion of basing learning on spectrograms and also the ability to retain the source voice characteristics in the translated output. This is somewhat analogous to recognising the character of an accent (Google AI Blog, 2019).

The recently proposed approach to machine translation, called Neural Machine Translation aims to address possible performance problems in the current paradigm of building a translation model from many sub-components, each of which is tuned separately, by building a single large neural network that can be jointly tuned to maximise the translation performance (Bahdanau et al., 2014).

The future is bright for voice recognition. The explosion of constant voice recognition software being released into the market, shows how the world is changing and how relevant speech input is becoming, rather than having to input data physically.

2.4 Machine Learning and Speech Classification

Voice recognition systems are incredibly common in today's devices and their primary purpose is to recognise words. Voice recognition and classification is made by a form of machine learning whereby large amounts of training data is used by software to recognise patterns in that data for specified classifications (Call Rail, 2017). This is an example of supervised learning where a learning model makes predictions on the classification of input data and is corrected if the prediction is incorrect. This process of learning creates some function, that predicts an output, Y, based on some inputs, X. At its basic form, this can be represented as:

$$Y = f(X) \tag{2.1}$$

Techniques of self-correction are built into the learning, and the training continues in an iterative manner until the accuracy of the prediction is at an acceptable level, or the accuracy ceases to increase. By contrast, unsupervised learning is a model whereby the input data is not labelled with a classification; this type of learning can still be used to group by similarity, such as for clustering similar inputs together into separate groups, but does not translate an input to exactly one distinct classification (Machine Learning Mastery, 2019*b*).

Supervised learning models may be implemented with a range of different underlying machine learning algorithms. Artificial neural networks (ANN) are algorithms based on human biological neural networks, whereby a decision is made at a neuron and the neuron either fires or stays dormant for some given input(s) and logic at the neuron, and some feedback mechanism used to change the decision as to when a neuron should fire (Data Science Central, 2017). Examples of algorithms in this class are:

- Perceptron
- Multilayer Perceptron (MLP)
- Back-Propagation

Modern computing power allows the building of much larger ANNs to be able to classify richer datasets such as images and audio. Popular examples of these deep learning algorithms include:

- Convolutional Neural Network (CNN)
- Recurrent Neural Network (RNN)
- Long Short-Term Memory Networks (LSTMs)

Multilayer Perceptrons, or MLPs for short, are very suitable for classification problems such as word or image recognition. However, an MLP with many hidden layers may not scale well as the size of the input increases (Machine Learning Mastery, 2016). Convolutional Neural Networks (CNN) are also used for classification and have additional benefit of being able to downsample large input data without losing key classification information, and so scale better than simple MLPs.

Recurrent Neural Networks are also extremely popular for ML applications. RNNs are better suited to sequence prediction problems. Long Short-Term Memory is a specific type of RNN and is widely used for natural language processing, whereby sequences of words, as in sentences or paragraphs, are a key objective (Towards Data Science, 2018*c*).

This project aims to recognise characteristics from words, rather than recognise sequences of words, and seeks to exploit features of audio, and so CNN to effectively downsample, followed by MLPs is the chosen initial machine learning model.

2.5 Software to aid Machine Learning

Machine learning algorithms are extremely complex mathematical models and include many variations and tuneable parameters. Such complexity means it would not be feasible to hand craft these from scratch for every machine learning application. For such reasons, these algorithms are available as open source, allowing both beginners and experts to build learning models.

For example, TensorFlow is a widely used framework that provides both high level and low level APIs to build and train deep learning models. It is a free to use open source library developed by Google to be used for building complex deep learning models (InfoWorld, 2019).

As mentioned, TensorFlow provides a high level API. One well-used example is Keras. This is very user-friendly and allows simple configuration of building blocks required to construct a learning model. In addition, the Keras API can be called using the Python programming language (Keras, n.d.*b*). Several possibilities are possible for the IDE, such as local development using Anaconda's Jupyter Notebook or PyCharm. Google Colab provides an IDE that allows cloud-based development, removing the need to invest in expensive hardware and local software maintenance (Medium, 2018*a*). In contrast, PyTorch provides low level APIs that provide greater flexibility but at the cost of increased complexity; this is ideally suited to academic research (KETKAR, 2017).

Since Keras is fully integrated into the TensorFlow framework and is intuitive to use even for a novice, this combination is favoured for this project.

Programming in Python is the natural choice, as the language is simple to use and fits perfectly with using the TensorFlow framework. In terms of a development environment, programming in Jupyter Notebooks is very powerful since data visualisation is performed inline with each cell of code. This is also possible with Google Colab, with the added benefits that libraries are pre-installed, the notebook and any data can be saved to Google Drive and there is no requirement to use local computing resources. These advantages of using cloud computing make Google Colab a compelling choice as the development environment for this project.

2.6 Data Collection

In order to train any machine learning model, a suitable dataset is required. For the purpose of this project domain, a dataset that encompasses audio samples from each region of the UK for both male and female voices is required.

An industrial sized dataset is ideal, to give the best possible chance of a high prediction rate. Many datasets can be found online that provide accents from across the UK; however, these were not deemed suitable and so were rejected. Some of these datasets that were considered are shown below.

- International Dialects of English Archive (International Dialects of English Archive, n.d.)
- The Speech Accent Archive (George Mason University, 2020)
- British Library Accents and Dialects (British Library, n.d.)

- 1.5 TB of Labelled Audio Datasets (Towards Data Science, 2018*b*)
- Common Voice Dataset (Common Voice, n.d.)
- English (British Neutral) Accents (Voquent, n.d.)

As mentioned, none of the datasets presented met the specifications of this project. Therefore, a purpose built dataset was created from scratch that addressed the needs of this project. In Chapter 3, the solution to this problem will be expanded upon in greater detail, explaining how the audio was sourced and how it was then pre-processed, such that it could be used as an input training dataset to a machine learning model.

2.7 Critique of the Review

Findings on the topics concerning this project have been presented in this section. Researching into how the UK is divided into regions gave background to the differences in accents and how they can change from region to region. The history of voice recognition and today's technology is shown. The significance of this history shows that the complexity of today's state-of-the-art technology has its foundations set in 1911 when Radio Rex was released. Due to this, there are now endless possibilities using speech data. Machine learning and speech classification is a major part of voice recognition. Currently, there are many techniques available that allows one to successfully implement a final solution to a problem. One such enhancement introduced in recent times is the different libraries available for use in the machine learning domain. Through research and comparison, the most appropriate framework, ML libraries, neural network, development language and IDE have been selected. Further, through understanding the current datasets readily available, it was established that a dataset will need to be created from scratch to meet the requirements of the project. Conducting thorough research and therefore collecting informative data, gives the potential to make well-educated decisions to implement a successful project; to enable the development of a project which takes speech input and classifies and predicts an output for the user. Without a well-detailed literature review, it is not possible to quantify the currently available software in the relevant space, and therefore, not possible to fully define the project domain. The bulk of the research is centered around the study of UK regional accents and independent of this, machine learning algorithms for audio classification problems. In addition, it is important to understand the history of voice recognition, and what is currently happening in the industry now, in both the classification domain and also recognition domain. These topics were addressed in relevant sub sections.

In conclusion, the literature reviewed has given the right basis to go and find a solution to the problem in hand and complete a complex machine learning problem.

2.8 Summary

The first part of the review shows research conducted on UK regional accents. In addition, the use of various sources, such as (Netherlands Worldwide, n.d.), allowed for the understanding of each region within the UK and how they differ. This was an important step in the process of conducting a literature review. It is vital to understand the locations of each region, history of accents, and how these accents are changing over time. As touched on, diluted accents are becoming common within the UK. *Could this have an effect on classification and prediction success rate?* The differences between voice traits of different genders are then discussed.

After this, the field of voice recognition is explored, presenting a timeline that illustrates the history of voice recognition and developments within this field. Developments for this technology, and how it has contributed to pioneering the next generation voice recognition software is then discussed, on how it has shaped our day-to-day lives. Towards the end of 2011, it is read that voice recognition systems changed, and that speech data is becoming more and more relevant. The topic of machine learning and speech classification was then discussed, and how it can be incorporated within the voice recognition field, looking into the different types of classifications and ML techniques available for use in different project domains. A speech classification problem can be approached in different ways, and it was established that a CNN based network was the best approach in implementing such a project, to classify speech data from a given input. The CNN learning model fits best with classification of features of audio features. Libraries available for machine learning were then discussed, whereby the use of TensorFlow and Keras was established to use for this project. TensorFlow framework with its integrated Keras support provides the flexibility and abstraction needed for implementation. In terms of development, using Python as the programming language in the Google Colab development environment is best suited. Data collection and the availability of any existing data was then explored, with the conclusion that relevant and qualifying data is not available. As a result, it was decided that a purpose made dataset was to be created.

Chapter 3

Methodology

The methodology chapter presents in-depth research into the algorithms background required prior to embarking on coding the project. The implementation is then discussed with extensive code snippets to demonstrate the full flow of implementation, along with steps taken to test and verify the code. Finally, the experiments undertaken to validate the implementation are listed.

3.1 Algorithm Descriptions

3.1.1 Machine Learning and Neural Networks

Machine learning (ML) is a subset of artificial intelligence. ML algorithms are based on neural networks (NN) and datasets (pre-existing data that will be used by a NN for training), from which they are able to make decisions/predictions from.

3.1.1.1 Artificial Neural Networks

This project uses a supervised learning system to classify sounds into a set of classes. The learning system is a deep learning neural network, an extension of an artificial neural network (ANN). In order to be able to explain the algorithms used, a background is provided here (Schmidhuber, 2015).

At its simplest, a neural network is a layer of inputs connected to smaller layer of output(s) via an intermediate layer. The intermediate layer is called the hidden layer since the inputs and outputs are externalised, but the inner workings between these layers cannot be seen. Figure 3.1 shows an example of a simple artificial neural network.

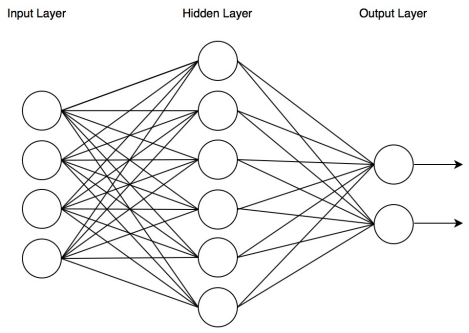


Figure 3.1: Simple Artificial Neural Network

Each element in the network is called a neuron; a neuron can make simple decisions and feed those outputs into other neurons. The input layer neurons (first layer) accept inputs to the model and the output neurons (last layer) output the classification of the input. Figure 3.1 depicts a fully connected network, since every neuron is connected to all neurons in neighbouring layers. The process is called supervised because the input is a training data set against which the output classification can be verified for accuracy of classification; essentially, there is a well-defined mapping between the inputs and outputs and the learning algorithm must approximate this mapping to a required level of accuracy, based on probability.

This simple network is called a shallow network as there is only a single hidden layer. Adding more hidden layers enables the network to improve its learning accuracy, and such a network is referred to as a deep network. Modern neural networks of today use a deep network between 3-10 hidden layers. Figure 3.2 shows a deep neural network with three hidden layers.

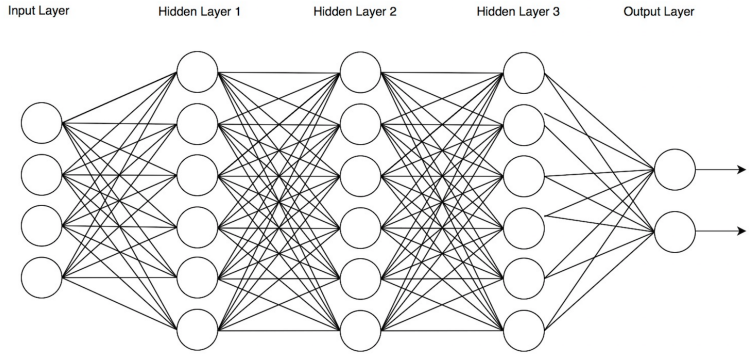


Figure 3.2: Deep Artificial Neural Network

The details of the mathematical operations at each hidden layer neuron are key to understanding how the neural network is able to classify into a set of outputs.

The maths at each neuron can explored by looking at the perceptron model, as shown in Figure 3.3 (Jantzen, 1998) (Towards Data Science, 2019). At set of inputs x_1 to x_n is shown, each with it's own importance, or weighting w_1 to w_n.

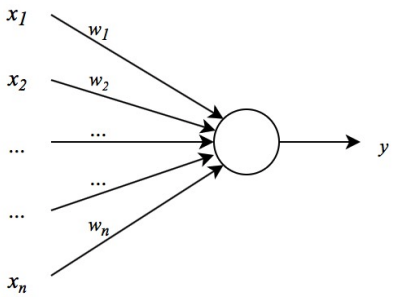


Figure 3.3: Perceptron Model

The output y can be thought of as the sum of the product of all inputs and their weightings, a binary value can be assigned to the output, depending on the value of this sum being smaller or larger than some threshold, θ, as shown in Eq. (3.1)

$$y = 1 \; if \sum_{i=1}^{N} x_i.w_i \geq \theta$$
$$y = 0 \; if \sum_{i=1}^{N} x_i.w_i < \theta \tag{3.1}$$

This can be re-written with θ on the left-hand side of the inequality, as seen in Eq. (3.2)

$$y = 1 \; if \sum_{i=1}^{N} x_i.w_i - \theta \geq 0$$
$$y = 0 \; if \sum_{i=1}^{N} x_i.w_i - \theta < 0 \tag{3.2}$$

The problem with these equations is that the neuron is not self-describing, since an externally provided hard coded value of θ is required. This can be avoided by making the θ an input to the neuron as shown in Figure 3.4

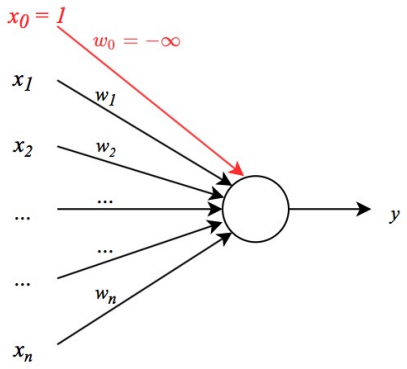


Figure 3.4: Perceptron Model with w_0 Input

Note that a new input x_0=1 is added with a weighting of w_0=-θ, and the product reflects the θ in the right-hand side of Eq. 3.1.

This equation can now be re-written as shown in Eq. (3.3).

$$y = 1 \; if \sum_{i=0}^{N} x_i.w_i \geq 0$$
$$y = 0 \; if \sum_{i=0}^{N} x_i.w_i < 0 \tag{3.3}$$
$$where, \; x_0 = 1 \; and \; w_0 = -\theta$$

w_0 is referred to as the bias to the neuron, and just like all other weights, it can be tuned by the model as required, in order to best fit the outputs compared to the training data.

Breaking down the sum of all values such that $w_0.x_0$ is separated, then the above can rewritten again as Eq. (3.4).

$$y = 1 \; if \sum_{i=1}^{N} x_i.w_i + x_0 w_0 \geq 0$$
$$y = 1 \; if \; z + w_0 \geq 0 \tag{3.4}$$
$$if \; z \geq -w_0 \; , where \; z = \sum_{i=1}^{N} x_i.w_i$$

If this is represented as a graph, then a stepwise change in the value of y is seen at a specific value of w_0, as shown in Figure 3.5.

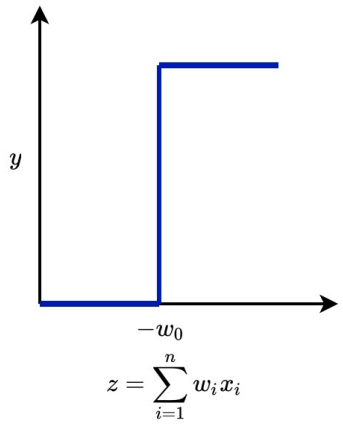


Figure 3.5: Step Function

This is not desirable in most applications of learning. A smoother transition is preferred and there are many mathematical equations where *f(x)* is a smoother curve for any given input *x*, and these can be generalised as having a graph similar to Figure 3.6.

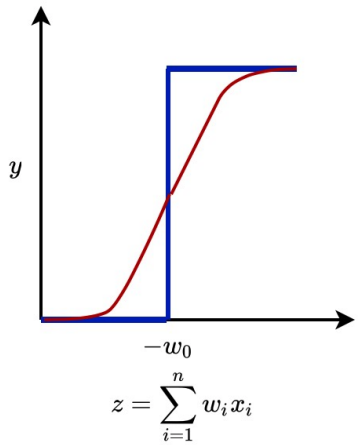


Figure 3.6: Smoother Step Function

Note how the simple binary output is now a smooth continuous line that is akin to providing the probability of an output of 'on' or 'off'.

This mathematical function effectively decides when the neuron output is 'on' or in other words, when the neuron is activated, and is therefore called the activation function. Activation functions are discussed further later in Section 3.1.1.5.

The set of weights discussed above are not fixed in value; these change as the model is trained, as part of a process call backpropagation.

3.1.1.2 Backpropagation

The discussion of bias and weightings explained that these can change as the model learns. This process of providing feedback from the output back to the input neurons is known as backpropagation. This can be broken down into the following series of actions (Buscema, 1998):

- The training data is fed into the network, the weights are initialised, and the model generates the initial set of output predictions. This is known as the forward pass.
- The model compares the output against the known class value and computes an error function (also known as loss function) based on the difference between the actual and predicted values. The loss function may be as simple as the sum of the square of differences between actual and predicted outputs. The goal is to have the loss function to be computed to as close as possible to 0.
- The loss function is used to feedback into the model such that weights can be changed in such a way that the loss function would then decrease in value. For performance, it is important to establish the rate at which the error changes relative to the changes to the weight. For this, the derivative of the loss function is computed. The derivative function output dictates if the weight change should be made. The objective is to identify the minimal set of weights that could be changed in order to minimise the loss function.
- The weights could be updated after every single iteration of input to output classification and loss function computation. However, this is could be very slow for large datasets. In general, the weights are updated after a number of iterations, and this is referred to as the batch size. A very simple way of updating the weights is to adjust by a small fraction of the derivative function output and slowly bring the weight closer to the nominal value. Many other methods exist, each with their own pros and cons. These methods are known as optimisers. The two optimisers considered for this project are Adaptive Moment Estimation (Adam) and Stochastic Gradient (SGD), as these are two of the most popular optimisers for classification problems (Wilson et al., 2017) (SALu, 2017).

  SGD is an efficient and effective optimisation method that has been central to many advances in deep learning. SGD behaviour can be further tuned by adjusting hyper parameters such as Nesterov Momentum (Towards Data Science, 2017*b*). Adam is a stochastic optimiser that is simple to employ with acceptable default values for hyper parameters, is efficient in terms of computation and has low memory requirement (Kingma and Ba, 2014). The performance of a model will be measured using cross-entropy. This a loss function for classification problems, and is a measure of the difference between two probability distributions for the given dataset (Keras, n.d.*d*).

Simple neural networks are the basic building block for learning. However, they do not scale well for large input datasets. For example, for image classification of an image with 512*128 pixels would result in 512*128=65636 inputs to every neuron in the next layer. If

the next layer has say, 256 neurons, that would mean 65636*256 (over 16 million) weights to consider for training. As further hidden layers are added, the complexity increases further.

One solution is to devise a method to decrease the size of the input to a fully connected neural network, whilst still retaining all of the important information. This can be achieved by a Convolutional Neural Network.

3.1.1.3 Convolutional Neural Network

A Convolutional Neural Network is a commonly used neural network for classification of complex data such as images. CNNs provides multiple building blocks, or layers, to progressively simplify complex input data without losing summary information that defines the data set, such that this can become an input to a fully connected layer for final classification. This approach greatly speeds up performance and accuracy of the classification.

CNNs have been used extensively for image recognition, where an image has a shape of a 2D matrix of pixels. For audio samples, it is possible to transform the audio into an "image" by executing a STFT of the audio. Once this is done, a STFT image recognition model can be used.

A CNN is made of the following layers (Krizhevsky et al., 2012) (Towards Data Science, 2018*a*):

1. Convolution
2. Activation Function
3. Pooling
4. Normalisation
5. Flatten
6. Dense

These layers will be discussed below.

3.1.1.4 Convolution

The convolution stage is analogous to scanning the input image one small part at a time, and at each sample, produce a small summary that describes the piece being inspected.

Once an image has passed through a convolutional layer, it has been abstracted to a feature map. A feature map is a set of probabilities of a feature belonging to the input class.

The implementation of this involves sliding a small viewing window, or kernel, across an image to extract features from the image.

The kernel is itself is a small 2D matrix of numbers that slides over a part of the image, known as the tensor. At each overlap, a product of each element of the kernel with corresponding position in the image is made; the sum of these products is assigned to the corresponding position in the output tensor. The output tensor is the feature map.

For example, assume there is an input image, I, with dimensions (5, 5) and a kernel, K, with dimensions (3,3), as shown in Figure 3.7.

A (3,3) window for the kernel is created and placed on the top left corner of the image. An element-by-element multiplication is then performed and summed together, then assigned to the output as shown in Figures 3.8 and 3.9.

0	1	1	1	2
2	0	0	1	1
2	1	0	2	1
1	0	1	1	2
2	0	0	1	0

1	0	1
1	0	0
0	1	1

Figure 3.7: Image and Kernel

0	1	1	1	2
2	0	0	1	1
2	1	0	2	1
1	0	1	1	2
2	0	0	1	0

Figure 3.8: Initial Kernel Placement

The output for the feature map is calculated as

$$\begin{aligned} F(0,0) = I(0,0) * K(0,0) + I(0,1) * K(0,1) + I(0,2) * K(0,2) + \\ I(1,0) * K(1,0) + I(1,1) * K(1,1) + I(1,2) * K(1,2) + \\ I(2,0) * K(2,0) + I(2,1) * K(2,1) + I(2,2) * K(2,2) \end{aligned}$$

$$\begin{aligned} = 0 * 1 + 1 * 0 + 1 * 1 + \\ 2 * 1 + 0 * 0 + 0 * 0 + \\ 1 * 0 + 1 * 1 + 0 * 1 \end{aligned}$$

$$= 0 + 0 + 1 + 2 + 0 + 0 + 0 + 1 + 0$$

$$= 4$$

As the kernel slides over the image, the resultant feature map will have dimensions (3,3), as shown below with the value at (0, 0) filled in

4		

Figure 3.9: Kernel at Starting Point Over Image

Continue sliding the kernel over the image until it arrives at the final position as shown in Figure 3.10.

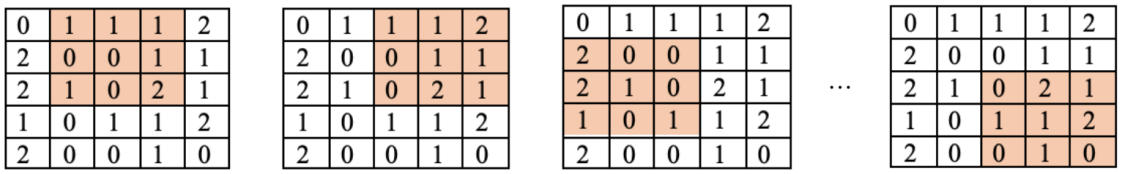


Figure 3.10: Kernel Sliding Over Image

The last element to be filled in the feature map is calculated as

$$F(2,2) = I(2,2) * K(0,0) + I(2,3) * K(0,1) + I(2,4) * K(0,2) + I(3,2) * K(1,0) + I(3,3) * K(1,1) + I(3,4) * K(1,2) + I(4,2) * K(2,0) + I(4,3) * K(2,1) + I(4,4) * K(2,2)$$

In general for an input, *I*, with of dimension *(i, j)*, a kernel, *K*, and a feature output of dimension *(m, n)*, then an arbitrary position in the feature map is defined as

$$f(m,n) = \sum_i \sum_j K[i,j] I[m-i, n-j] \tag{3.5}$$

In the example given, the final feature map is smaller in dimension than the input image, or tensor. This may not be desirable if the feature map is to be used as an input to another convolution layer. This can be avoided by padding the input tensor such that the eventual feature map dimensions are the same as the input tensor. Extending the earlier example, a padded input tensor is shown in Figure 3.11, where the increased dimensions are filled with zeroes.

0	0	0	0	0	0	0
0	0	1	1	1	2	0
0	2	0	0	1	1	0
0	2	1	0	2	1	0
0	1	0	1	1	2	0
0	2	0	0	1	0	0
0	0	0	0	0	0	0

Figure 3.11: Convolutional Layer Padding

With padding, the first kernel overlap is shown in Figure 3.12.

0	0	0	0	0	0	0
0	0	1	1	1	2	0
0	2	0	0	1	1	0
0	2	1	0	2	1	0
0	1	0	1	1	2	0
0	2	0	0	1	0	0
0	0	0	0	0	0	0

Figure 3.12: Sliding Kernel Over Image with Padding

Sliding this kernel across the input image will yield a feature map with dimension (5,5), just the original image dimension, resulting in the same size input for the next layer as was for this layer.

Note that the examples show the kernel window sliding by 1 position to the right; this is referred to as the stride. Stride positions greater than 1 will result in down sampling of the feature map; that is, the feature map will be smaller than the input and could also introduce instability into the model. Any required down sampling is deferred to the pooling layer.

The size of the kernel is specified in the model setup. However, the kernel element values are not user specified. These are automatically learned during training of the model.

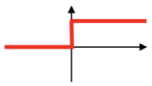
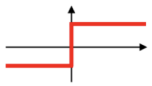
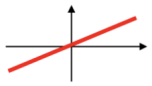
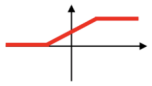
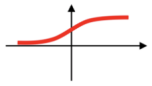
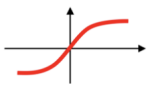
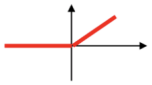
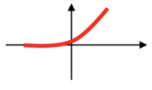
3.1.1.5 Activation Function

As discussed previously in Section 3.1.1.1, an activation function is a mathematical equation that defines when a neuron is activated, based on input values and depending on the relevance of the neuron in the prediction model. In addition, an activation function normalises the output of a neuron to be within a well-defined range (the actual range depends on the range). Since the activation function is attached to every neuron, it is important that the function is computationally efficient.

Since the activation function is attached to every neuron, it is important that the function is computationally efficient. The activation function is applied after performing the convolution.

Table 3.1 lists examples of common activation functions (Medium, 2019).

Table 3.1: Common Activation Functions

Activation function	Equation	Example	1D Graph
Unit step (Heaviside)	$\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Sign (Signum)	$\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Linear	$\phi(z) = z$	Adaline, linear regression	
Piece-wise linear	$\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}, \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multi-layer NN	
Hyperbolic tangent	$\phi(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$	Multi-layer Neural Networks	
Rectifier, ReLU (Rectified Linear Unit)	$\phi(z) = max(0, z)$	Multi-layer Neural Networks	
Rectifier, softplus	$\phi(z) = \ln(1 + e^{z})$	Multi-layer Neural Networks	

3.1.1.6 Pooling

The feature map summarises features present in the input. The summary is strongly coupled to the precise position of a feature in an input. As a result of this, any small change in the input feature can produce a different feature map. This is not desirable as small variations within an accent should not alter the description, or feature, of that accent.

The objective of the pooling layer is to take the feature map as an input and provide a pooled output that produces a set of summaries of small neighbourhoods of the feature map into smaller summary areas, such that any small changes in the input does not result in large variances in the pooled output, ensuring there is resistance to overfitting and increasing stability of the model. The decreased output also results in faster calculations. There are different types of pooling; maximum, minimum and average pooling for example. The pooled output is smaller than the feature map input, whilst still retaining the specifics of the information, and is often referred to as down sampling. Note that using larger strides in the convolution layer can also cause down sampling, but a separate layer makes this a more deterministic process. As with the convolutional layer, pooling can also vary the kernel (filter) and stride sizes.

Figure 3.13 shows examples of minimum, average and maximum pooling, where a set of neighbourhood values are summarised in the relevant entry in the pooled output.

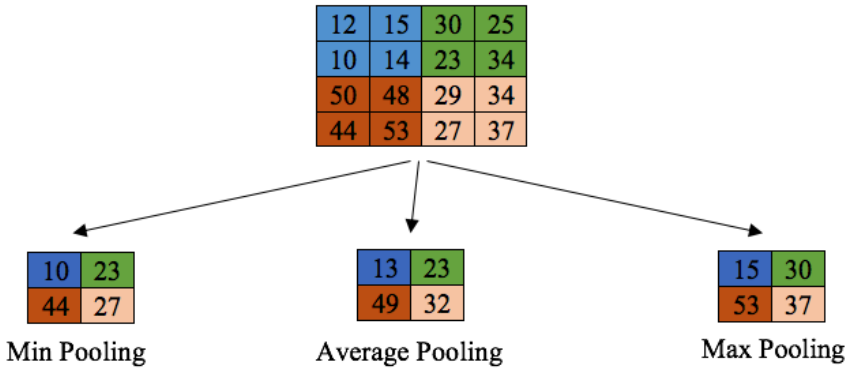


Figure 3.13: Pooling Examples

3.1.1.7 Normalisation

After pooling, the next stage is normalisation. The objective is to increase training speed by normalising values such that features spanning a wide range of values, with perhaps very few values in between the extremes, are adjusted and scaled such that the result is a smaller range of values. A side-effect of this is possible increase in learning accuracy since extreme values that did not work well with some activation functions can now be used in training.

3.1.1.8 Flatten

After potentially several rounds of convolution, pooling and normalisation, the next stage is to flatten the output of the previous layer into a vector, so that it is in the required shape to connect to all neurons in the fully connected layer that follows.

3.1.1.9 Dense

The dense layer is the fully connected layer, where every input is connected to every output by a weight, and the end result is passed through a non-linear activation function such as the ReLU or Sigmoid functions.

3.1.2 Short-Term Fourier Transform

An audio file is often represented in the time domain. This is a "raw" amplitude vs. time mapping and can be intuitively understood by the reader, since audible sound corresponds to peaks in the graph and silence to a relatively flat line around an amplitude of 0. If a human listens to an audio file and visually inspects the raw amplitude plot against time, then the correlation between the audio being listened to and the visual plot can be made intuitively.

Although this form is human-friendly, it does not lend itself well to classification algorithms. Such algorithms work best for audio represented in the frequency domain rather than the time domain. To transform audio from the time domain to the frequency domain, the audio is broken into millisecond samples, and for each sample a Short-Time Fourier Transform is computed (Science Direct, 2006). The STFT for that sample is plotted as a coloured vertical line. The aggregated set of STFT plots for all the millisecond samples that constitute the audio samples is known as a spectrogram. An STFT is a set of Discrete Fourier Transforms (DFT), where each DFT is taken of a short window of the overall audio. Note that the DFT may be a Fast Fourier Transform (FFT), an efficient method of calculating a DFT.

An STFT helps to isolate the occurrence of frequencies to a short time interval. This technique allows decomposition of the characteristics of an audio sample. This representation of the audio as a set of frequencies characterises the accent and gender of the speaker in a way that the raw sound cannot, and consequently, this form is better suited as an input to machine learning algorithms. The resulting spectrogram conveys three-dimensional data:

- The horizontal axis represents time.
- The vertical axis represents the frequency.
- A colour encoded representation of amplitudes (corresponding to the vertical axis in the raw audio plot), where the colour represents the amplitude of a specific frequency at a specific time.

In effect, a sound wave is converted into a picture, as shown in the raw plot of amplitude against time in Figure 3.17 compared to the STFT of the audio in Figure 3.18.

3.1.2.1 Example of a Multiple Layer Network

Figure 3.14 shows an example of a multi layer network, showing multiple convolution layers with pooling and fully connected layers before a final prediction output layer (Towards Data Science, 2018*a*).

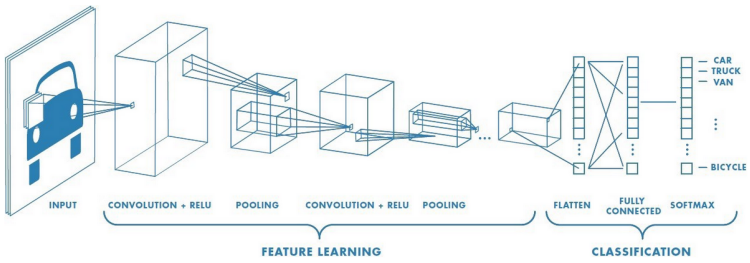


Figure 3.14: Example of a Multi Layer Network

3.2 Implementation

3.2.1 Data Collection

As mentioned in Chapter 2, no suitable dataset was identified to meet the scope of this project. Therefore, a dataset had to be formulated from scratch. Several steps were taken to achieve a solution to this problem, which eventually led to an appropriate dataset. This was to have audio data from each region within the UK, with an evenly split ratio of male to female. The steps taken are as follows.

1. Explore the possible options as to how a dataset can be formed to meet the scope of this project.
    - Concluded that audio would be sourced directly from YouTube. This allowed for flexibility by giving many different options of audio to select from, as well as the length of that audio. All of the sourced audio is referenced in Appendix A.
2. Study and understand the regions within the UK.
    - It was important to refer to Chapter 2, which gives full background knowledge of how the UK is divided and how their languages distinguish from each other.
3. Find individuals from each region that held a distinct accent.
    - To find individuals with distinct accents, it helps for consistency and better test and prediction when training the model. Furthermore, it was essential to find the same amount of audio for both male and female. By using multiple online sources for identification, appropriate individuals were selected for the dataset. As the process of selecting individuals went on, it was noted that those who are celebrities, politicians, and who had moved around the country a lot, tended to have a more diluted accent than a distinct regional accent. Therefore, the consideration of diluted accents was taken into account when choosing audio files.
4. Using YouTube, find appropriate audio of the individuals.
    - It was essential to find clear and uninterrupted audio with minimal background noise. This benefits the quality of the dataset, along with having minimal impurities within the audio file as the focus is on the accent.
5. Download audio files.
    - Each file was converted from .mp4 file to a .wav file. This allowed us to use LibROSA library to load in Python and subsequently do pre-processing on the files.

6. Organise and label the dataset.

    - *Train and Test:* For training and test purposes, 720 .wav files (360 male, 360 female) were labelled and organised into its appropriate regions. Per region, there are 60 .wav files, with 1:1 split of male and female across the 12 regions.
    - *Unseen:* For unseen data, 72 .wav files (36 male, 36 female) were also labelled and organised into appropriate regions. This meant there were 6 .wav files per region, where there was again, a 1:1 split between male and female. The use of unseen data was to test the model with data not used in the initial training and testing set.

7. Pre-process the dataset.

    - The next stage is to apply pre-processing techniques on the dataset. This will be expanded upon in greater detail in the following subsection.

3.2.2 Data Cleaning and Pre-Processing

Raw audio files need to be pre-processed prior to being used as training data or prediction. For consistency, the audio samples should be the same length and also be sampled at the same frequency, and also transformed to the frequency domain, since this form is amenable to machine learning as it decomposes an audio into it constituent frequencies. Much of this pre-processing work is carried out using the Python package for audio manipulation and analysis, called librosa.

3.2.2.1 Data Length

The initial audio files sourced from YouTube are approximately thirty seconds long. These audio files are split into three pieces of ten seconds each, with the small remainder discarded. After the split, any audio files less than ten seconds are rejected, although this is not possible if the original audio is greater than thirty seconds long. For audio taken from a user via the local microphone, the audio is trimmed to exactly ten seconds.

3.2.2.2 Data Sampling

An audio file is an analogue signal and needs to be transformed into a digital sound; this is done by a technique called sampling, whereby the amplitude is measured, or sampled, many times a second. The number of samples per second is called the sampling rate, measured in Hz. All audio files are resampled at 8kHz because it is important have a common rate for all files, so as to standardise all audio input. In addition, resampling at 8kHz still retains most of the frequencies related to speech, whilst removing any unwanted noise outside of the human hearing frequency range.

The combination of truncating at ten seconds and resampling at a common rate ensures all audio samples have the same "shape", which is a requirement for the learning model.

3.2.2.3 Short-Time Fourier Transform

All of the raw audio files are converted from the time domain to the frequency domain, since the latter is amenable to machine learning as it decomposes an audio into it constituent frequencies; this can be achieved by taking a Fourier Transform of multiple windows in the raw audio file, as discussed in Section 3.1.2.

3.2.3 Implementation of the Learning Model

This section shows code snippets from the Colab Notebook implementation and discussion of the outputs.

3.2.3.1 Splitting Original Audio Data

The audio collected from YouTube was originally over 30 seconds long, and placed into the relevant regional accent folder as a long clip. These files were split into three samples, each of 10 seconds duration prior to uploading to Google Drive, as shown in Listing 3.1.

```
from pydub import AudioSegment

DIRIN = '/Users/jshergil/Documents/Dataset/trainval_30s'
DIROUT = '/Users/jshergil/Documents/Dataset/trainval_10s'

LEN=10  #length per sample
SEGS=30/LEN # number of segments

from glob import glob
import os

# all wav files
wav_files = glob(os.path.join(DIRIN, '*/*wav'))

# Loop over files to get samples
data = []
for i, wav_file in enumerate(wav_files):
    frsplit = wav_file.rsplit('/',2)
    print(frsplit[0], frsplit[1], frsplit[2])
    label = frsplit[1] #NI_F, NI_M, SC_F etc
    name = frsplit[2]  #the wav file name
    frsplit = name.rsplit('.',2) #0 is the name, 1 is the .wav extension
    t1=0
    t2=LEN*1000
    print(wav_file)
    for j in range(SEGS):
        print(t1, t2)
        newAudio = AudioSegment.from_wav(wav_file)
        newAudio = newAudio[t1:t2]
        t1=t2 #start next clip at t2
        t2=t2+(LEN*1000)#end next clip LEN (eg,10) secs later
        newfile=frsplit[0] + '_' + str(j)
        path = DIROUT + '/' + label

        if not (os.path.exists(path)):
            os.mkdir(path);
        newfilefull = path + '/' + newfile + '.wav'
        print(newfilefull)
        newAudio.export(newfilefull, format="wav")
```

Listing 3.1: Splitting Original Audio Samples

3.2.3.2 Accessing the Training Data

All of the training data is uploaded to Google Drive. Listing 3.2 shows how the Google Drive is mounted at /content/gdrive and lists the data directories, as shown in Figure 3.15. The listing of the directory shows that training data is divided into the 12 regions of the UK, with

separate directories for Male and Female samples per region. This is also the case for the unseen data directory.

```
# mount the google drive to access the wav files
from google.colab import drive
drive.mount('/content/gdrive')

# list the subdirectories of the 10 second wav file directory
# the subdirectory names correspond to labels in the dataset
print('train and validation data under directory trainval_10s')
!ls "/content/gdrive/My Drive/Project/trainval_10s"

print('train and validation data under directory UNSEEN_10s')
!ls "/content/gdrive/My Drive/Project/UNSEEN_10s"
```

Listing 3.2: Regional Accent Data on Google Drive

```
train and validation data under directory trainval_10s
EE_F  EM_F  GL_F  NE_F  NI_F  NW_F  SC_F  SE_F  SW_F  WA_F  WM_F  YH_F
EE_M  EM_M  GL_M  NE_M  NI_M  NW_M  SC_M  SE_M  SW_M  WA_M  WM_M  YH_M
train and validation data under directory UNSEEN_10s
EE_F  EM_F  GL_F  NE_F  NI_F  NW_F  SC_F  SE_F  SW_F  WA_F  WM_F  YH_F
EE_M  EM_M  GL_M  NE_M  NI_M  NW_M  SC_M  SE_M  SW_M  WA_M  WM_M  YH_M
```

Figure 3.15: Google Drive Project Data Directory Listing

Listing 3.3 shows how the subdirectories containing the audio files are loaded as the labels, or classes for subsequent learning.

```
# every subdirectory under the top level directory is a label name
# ignore hidden operating system files (these start with a .)
subdirs = [file for file in
           os.listdir(WAV_DIR) if not file.startswith('.')]

#present in sorted alphabetical order for easy of reading output, and call
    them labels
LABELS = sorted(subdirs)
print(LABELS)

# number of labels, or classes
NUM_CLASSES = len(LABELS)
```

Listing 3.3: Creating The Label Set

3.2.3.3 Visualise the Audio in the Time Domain

To show an example of pre-processing of the data, an example audio wave is sampled using the library librosa. Listing 3.4 and Figure 3.16 show that after sampling, the audio file is represented as set of values in a 1D array, and in this case the sampling rate is approximately 22kHz.

```
# load the example wav file with lobrosa
wav_file, sample_rate = librosa.load(WAV_DIR + EXAMPLE_WAV)
 #an array of discrete values (samples) of the continuous sound
print(wav_file)

print(sample_rate)
```

Listing 3.4: Sampled Data as 1D Array

The code in Listing 3.5 plots the sampled data in the time domain to show how the amplitude varies with time, as shown in Figure 3.17.

```
[ 0.          0.          0.         ... -0.00039724 -0.00051321
 -0.0007516 ]
22050
```

Figure 3.16: Output Showing Sampled Data as 1D Array

```
# print interesting information
print('len of the wavfile', len(wav_file), 'sample rate', sample_rate)
print('shape of the wav file', wav_file.shape, 'or as an array',
      np.array(wav_file.shape))

# plot the amplitude/time graph
fig = plt.figure(figsize=(20, 10))
subplot = fig.add_subplot(211)
subplot.set_title('Amplitude/Time Graph of ' + WAV_DIR + EXAMPLE_WAV)
subplot.set_xlabel('Time')
subplot.set_ylabel('Amplitude')

# the number of sample points on the x-axis
num_time_samples = np.linspace(0, sample_rate/len(wav_file),
                               sample_rate*WAV_LEN)

# plot the wav file
subplot.plot(num_time_samples, wav_file)
```

Listing 3.5: Audio File in Time Domain

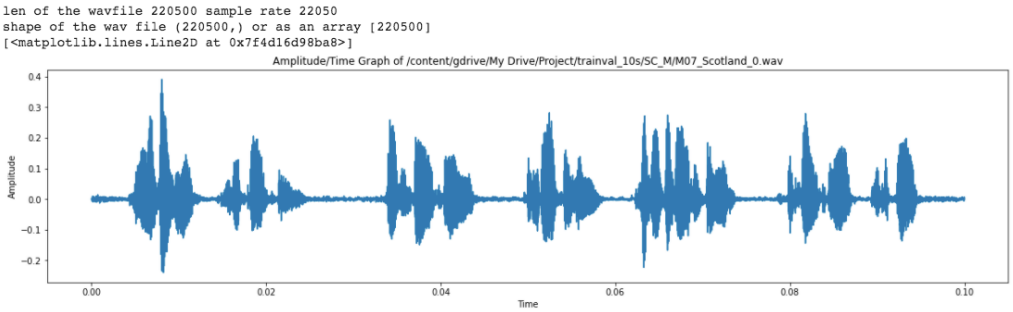


Figure 3.17: Time Domain Audio Representation

3.2.3.4 Visualise the Audio in the Frequency Domain

The code in Listing 3.9 transforms the raw audio into the frequency domain by a executing a STFT. Figure 3.18 shows the resulting spectogram.

```
# get STFT (Fourier Transform) of the wav file
stft_o = librosa.stft(wav_file)

# get the absolute values of the frequency magnitude at a given time
stft_o = np.abs(stft_o)

# print interesting info
print('shape of the spectogram is', stft_o.shape)

# convert amplitude spectrogram to dB-scaled spectrogram
db_spec = librosa.amplitude_to_db(stft_o, ref=np.max)

# display the spectogram
librosa.display.specshow(db_spec, y_axis='log', x_axis='time')
plt.title('Short-time Fourier Transform dB-scale Spectogram')

# add a colour map
plt.colorbar(format='%+2.0f dB')

plt.tight_layout()
plt.show()
```

Listing 3.6: Audio File in Frequency Domain

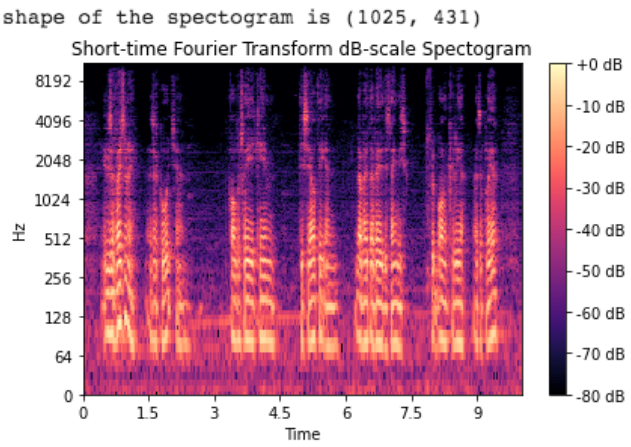


Figure 3.18: Frequency Domain Audio Representation

3.2.3.5 Resampling Audio Files

In Listing 3.9, the original sampling rate is obtained to prior to resampling, and displays a button to play back the audio, as shown in Figure 3.19.

```
print('original sample rate ', sample_rate,
      '. wave file shape ', wav_file.shape)

ipd.Audio(wav_file, rate=sample_rate)
```

Listing 3.7: Show Original Sampling Rate

The audio is then resampled as shown in Listing 3.8; Figure 3.20 shows that the new sampling rate is 8kHz.

```
wav_file = librosa.resample(wav_file, sample_rate, RESAMPLE_RATE)
print('resample sample rate', RESAMPLE_RATE,
```

Figure 3.19: Original Sampling Rate

```
      ', wave file shape ', wav_file.shape)
ipd.Audio(wav_file, rate=RESAMPLE_RATE) # play the audio
```

Listing 3.8: Resample Audio

Figure 3.20: New Sampling Rate

3.2.3.6 Number of Samples Per Audio Class

The code in Listing 3.9 visualises the number of audio samples available per class, as shown by the barchart in Figure 3.21.

```
# print the number of samples per label
total_samples = 0  # total number of samples
num_samples = []   # array of sample sizes

# loop through every label (subdirectory), counting all .wav files
for label in LABELS:
    waves = glob(os.path.join(WAV_DIR + '/' + label, '*wav'))
    num_samples.append(len(waves))
    total_samples += len(waves)

print('Total number of samples:', total_samples)

# plot number of samples per label
plt.figure(figsize=(20,5))
y_pos = np.arange(len(LABELS))
plt.bar(y_pos, num_samples)
plt.xticks(y_pos, LABELS, rotation=90)   #rotate label name on x-axis
plt.ylabel('Num Samples')
plt.xlabel('Class Label')
plt.title('Number of Samples per Label')

plt.show()
```

Listing 3.9: Number of Samples per Class

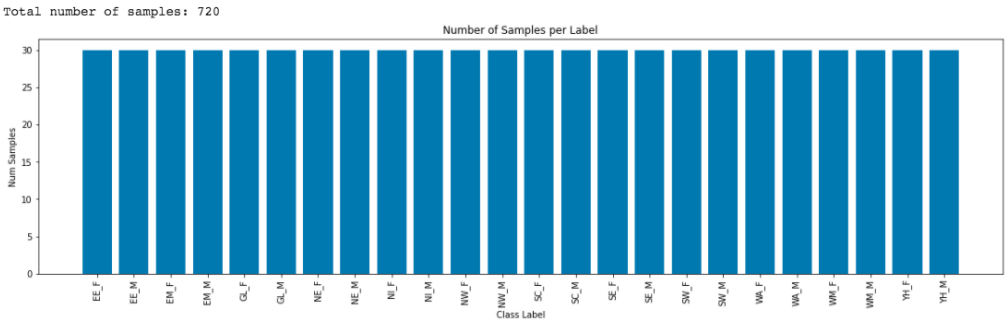


Figure 3.21: Number of Samples per Class

3.2.3.7 Pre-processing Training Dataset

The example of taking a single audio file and resampling at 8kHZ, and then representing in either the time or frequency domains must be carried out for all audio samples in both the training dataset and then later for the unseen dataset. Listing 3.10 defines a method called 'loadfft()' to abstract these details.

```
# define a function called loaddfft that does the following:
#   load all wav files then:
#     pad it to be the correct length if required
#     resample the wav file
#     execute an FFT on the resampled wav file
#     return the FFTs as an array plus corresponding label
def loadfft(wavdir):
  # FFT representation of wav files
  all_stft = []
  # the label id for each of the wavefiles
  all_stft_label = []
  # all_stft_label[i] represents the label identifier for the
  # FFT at all_stft[i]

  # number of good samples
  num_good_samples = 0
  for label in LABELS:
    print('processing label', label)
    # names of all wav files for this label
    wavefiles = [file for file in
                 os.listdir(wavdir + '/'+ label) if file.endswith('.wav')]

    # for each wav file in the list
    for wavefile in wavefiles:
      # load the wav file
      wav_sample, wav_sample_rate = librosa.load(wavdir + '/' + label +
                                                 '/' + wavefile)
      # pad with blanks if the wav file is too short
      if(len(wav_sample) != wav_sample_rate*WAV_LEN):
        diff=wav_sample_rate*WAV_LEN-len(wav_sample)
        ind=np.random.randint(0, diff)
        wav_sample = np.pad(wav_sample,
                            (ind, wav_sample_rate*WAV_LEN-
                                  (ind+len(wav_sample))),
                            "edge")
      # resample the wav file
      wav_sample = librosa.resample(wav_sample,
                                    wav_sample_rate, RESAMPLE_RATE)

      # if the length is exactly WAV_LEN seconds, then get a STFT
      # this means any files that are not the correct length will
      # be discarded
      if(len(wav_sample) == RESAMPLE_RATE*WAV_LEN):
        # get the STFT of this sample
        stft_o = np.abs(librosa.stft(wav_sample))

        # store the FFT of the wav file, and the label id for the file
        all_stft.append(stft_o)
        all_stft_label.append(label)
        num_good_samples += 1
      else:
        # these are wav files that are discarded, if any
        print(wavefile)
        print(len(wav_sample))
  return all_stft, all_stft_label, num_good_samples;

# call the function with all of the wav files in our base directory
all_stft, all_stft_label, num_good_samples = loadfft(WAV_DIR)
```

Listing 3.10: Function to Abstract Data Pre-processing

This function returns an array of images in the frequency domain, and also the label corresponding to each of these images. These arrays are used for training and validation of the model.

It is useful to visualise how different or similar each audio file is in the frequency domain, since this is the input to the learning model and helps define the classifications. Some spectograms are shown in Figure 3.22, for casual comparison, in order to demonstrate this point.

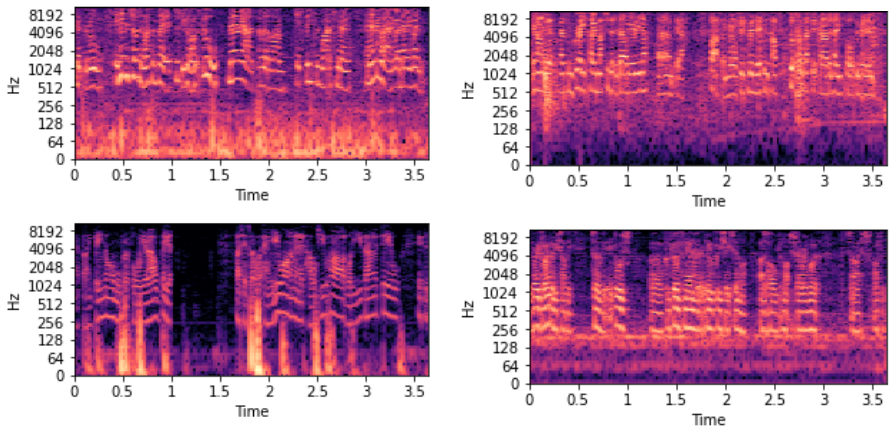


Figure 3.22: Example Spectograms

3.2.3.8 Preparing for the Model

The code in Listing 3.12 encodes the character strings for the label names to numerical values, for later input into the model. Each audio file in a given directory is encoded with the same numerical value.

```
# all of the output labels are character strings. These need to be encoded
# into numerical values to be input into the model
from sklearn.preprocessing import LabelEncoder

labenc = LabelEncoder()

# do the encoding
encoded_values = labenc.fit_transform(all_stft_label)

# remember the class names
class_names= list(labenc.classes_)
print(class_names)
print(encoded_values)
```

Listing 3.11: Integer Encode Output Labels

In Listing 3.12, the encoded values are converted to a binary class matrix; this is used with categorical_crossentropy for model compile below.

```
from keras.utils import np_utils
encoded_binary_class_matrix=np_utils.to_categorical(encoded_values,
    num_classes=len(LABELS))

# the output binary class matrix is abbreviated to y
y=encoded_binary_class_matrix
print(y)
```

Listing 3.12: Binary Class Encoding

The dataset is split into a training:test ratio of 80:20, using the sklearn method train_test_split(), as shown in Listing Referenceslist:DataSplit.

```
# now split the data based on train/validate sizes, resulting in
#   x_train, y_train        - the training set
#   x_validate, y_validate  - the validation set
from sklearn.model_selection import train_test_split
x_train, x_validate, y_train, y_validate =
    train_test_split(np.array(all_stft), np.array(y), stratify=y,
                     test_size = val_pct, random_state=888, shuffle=True)
```

Listing 3.13: Split Data into Training and Test

The parameters to the train_test_split() are explained below:

- np.array(all_stft). Array of all STFTs
- np.array(y). Array of all labels
- test_size = val_pct. Proportion of whole dataset to use for training
- shuffle=True. Shuffle the data before splitting.
- stratify=y. Do stratified sampling
- random_state=888. Seed for random number generator

3.2.3.9 Define and Build The Model

The multi-layer model is created as shown in Listing 3.15. For context, the preamble for this code is shown in Listing 3.14.

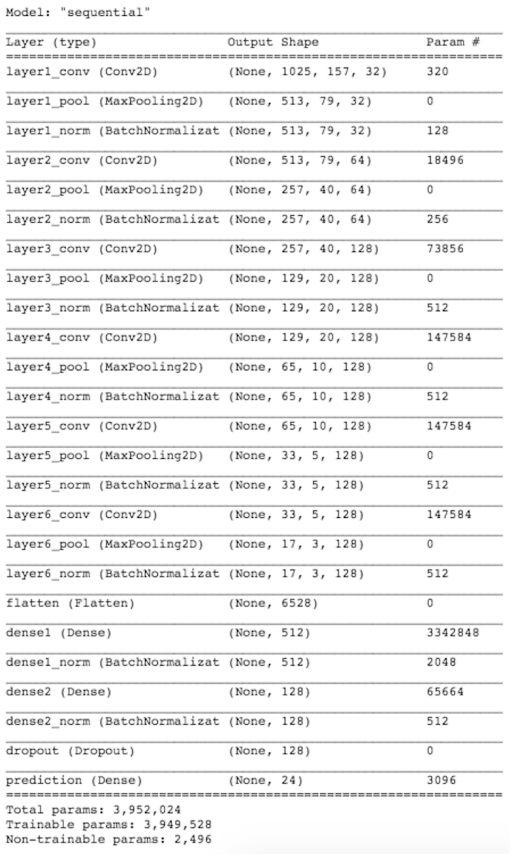
```
from tensorflow.keras.layers import Conv2D, MaxPooling2D
from tensorflow.keras.layers import BatchNormalization
from tensorflow.keras.layers import Flatten, Dense, Dropout
from tensorflow.keras.models import Model
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
from tensorflow.keras import backend as K
from tensorflow.keras import Sequential

# initialise
K.clear_session()

#the wave dimensions dictate the input shape
shape=(wave_dim_1, wave_dim_2, 1)

# activation function is Recitfied Linear Unit (relu)
act='relu'

# build sequential model
model = Sequential()
```

Listing 3.14: Convolutional Network Initialisation

```
# Layer 1 Convolutional, Maxpooling, Normalization
model.add(Conv2D(32, (3, 3), activation=act, padding='same', strides=1,
          name='layer1_conv', input_shape=shape))
model.add(MaxPooling2D((3, 3), strides=(2,2), padding='same',
          name='layer1_pool'))
model.add(BatchNormalization(name='layer1_norm'))

# Layer 2 Convolutional, Maxpooling, Normalization
model.add(Conv2D(64, (3, 3), activation=act, padding='same', strides=1,
          name='layer2_conv'))
model.add(MaxPooling2D((3, 3), strides=(2,2), padding='same',
          name='layer2_pool'))
model.add(BatchNormalization(name='layer2_norm'))

# Layer 3 Convolutional, Maxpooling, Normalization
model.add(Conv2D(128, (3, 3), activation=act, padding='same', strides=1,
          name='layer3_conv'))
model.add(MaxPooling2D((3, 3), strides=(2,2), padding='same',
          name='layer3_pool'))
model.add(BatchNormalization(name='layer3_norm'))

# Layer 4 Convolutional, Maxpooling, Normalization
model.add(Conv2D(128, (3, 3), activation=act, padding='same', strides=1,
          name='layer4_conv'))
model.add(MaxPooling2D((3, 3), strides=(2,2), padding='same',
          name='layer4_pool'))
model.add(BatchNormalization(name='layer4_norm'))

# Layer 5 Convolutional, Maxpooling, Normalization
model.add(Conv2D(128, (3, 3), activation=act, padding='same', strides=1,
          name='layer5_conv'))
model.add(MaxPooling2D((3, 3), strides=(2,2), padding='same',
          name='layer5_pool'))
model.add(BatchNormalization(name='layer5_norm'))

# Layer 6 Convolutional, Maxpooling, Normalization
model.add(Conv2D(128, (3, 3), activation=act, padding='same', strides=1,
          name='layer6_conv'))
model.add(MaxPooling2D((3, 3), strides=(2,2), padding='same',
          name='layer6_pool'))
model.add(BatchNormalization(name='layer6_norm'))

# Flatten
model.add(Flatten(name='flatten'))

# Dense layers
model.add(Dense(512, activation=act, name='dense1'))
model.add(BatchNormalization(name='dense1_norm'))
model.add(Dense(128, activation=act, name='dense2'))
model.add(BatchNormalization(name='dense2_norm'))

# Dropout
model.add(Dropout(0.2, name='dropout'))

# Predictions
model.add(Dense(NUM_CLASSES, activation='softmax', name='prediction'))

# print the model summary
model.summary()
```

Listing 3.15: Convolutional Network Layers

The output model summary is shown in Figure 3.23



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
layer1_conv (Conv2D)         (None, 1025, 157, 32)     320
_________________________________________________________________
layer1_pool (MaxPooling2D)   (None, 513, 79, 32)       0
_________________________________________________________________
layer1_norm (BatchNormalizat (None, 513, 79, 32)       128
_________________________________________________________________
layer2_conv (Conv2D)         (None, 513, 79, 64)       18496
_________________________________________________________________
layer2_pool (MaxPooling2D)   (None, 257, 40, 64)       0
_________________________________________________________________
layer2_norm (BatchNormalizat (None, 257, 40, 64)       256
_________________________________________________________________
layer3_conv (Conv2D)         (None, 257, 40, 128)      73856
_________________________________________________________________
layer3_pool (MaxPooling2D)   (None, 129, 20, 128)      0
_________________________________________________________________
layer3_norm (BatchNormalizat (None, 129, 20, 128)      512
_________________________________________________________________
layer4_conv (Conv2D)         (None, 129, 20, 128)      147584
_________________________________________________________________
layer4_pool (MaxPooling2D)   (None, 65, 10, 128)       0
_________________________________________________________________
layer4_norm (BatchNormalizat (None, 65, 10, 128)       512
_________________________________________________________________
layer5_conv (Conv2D)         (None, 65, 10, 128)       147584
_________________________________________________________________
layer5_pool (MaxPooling2D)   (None, 33, 5, 128)        0
_________________________________________________________________
layer5_norm (BatchNormalizat (None, 33, 5, 128)        512
_________________________________________________________________
layer6_conv (Conv2D)         (None, 33, 5, 128)        147584
_________________________________________________________________
layer6_pool (MaxPooling2D)   (None, 17, 3, 128)        0
_________________________________________________________________
layer6_norm (BatchNormalizat (None, 17, 3, 128)        512
_________________________________________________________________
flatten (Flatten)            (None, 6528)              0
_________________________________________________________________
dense1 (Dense)               (None, 512)               3342848
_________________________________________________________________
dense1_norm (BatchNormalizat (None, 512)               2048
_________________________________________________________________
dense2 (Dense)               (None, 128)               65664
_________________________________________________________________
dense2_norm (BatchNormalizat (None, 128)               512
_________________________________________________________________
dropout (Dropout)            (None, 128)               0
_________________________________________________________________
prediction (Dense)           (None, 24)                3096
=================================================================
Total params: 3,952,024
Trainable params: 3,949,528
Non-trainable params: 2,496
```

Figure 3.23: Learning Model Summary

The model is compiled as shown in Listing 3.16 (Keras, n.d.*c*).

```
# compile the Keras model
#  use categorical_crossentropy as the loss function - this is the
#  measure of exactness between and output and target sensor, and
#  defines the loss of the model

#  the sgd optimizer
from tensorflow.keras import optimizers

sgd = optimizers.SGD(learning_rate=0.01, decay=1e-6,
                     momentum=0.9, nesterov=True)

model.compile(loss='categorical_crossentropy', optimizer=sgd,
              metrics=['acc'])
```

Listing 3.16: Model Compilation

The parameters for model compilation are explained below (Towards Data Science, 2017*a*):

1. optimizers.SGD. Instantiate the Stochastic Gradient Descent optimiser, with non-default parameters. The parameters used are
   - learning_rate=0.01. The learning rate.
   - decay=1e-6. Parameter that accelerates SGD in the relevant direction and dampens oscillations.

- momentum=0.9. Momentum for faster convergence.
- nesterov=True. Nesterov momentum to be applied.

2. model.compile. Compile and configure the model for training. The parameters used are

   - loss='categorical_crossentropy'. The name of the objective function, or Loss instance.
   - optimizer=sgd. Use the instantiated SGD model.
   - metrics=['acc']. Evaluate the accuracy metric during model training and testing.

Listing 3.17 shows how the early stopping and model checkpoints callback functions are setup for model training.

```
# Define the model callbacks during training
# define when the model should stop training, rather than try all epochs
#   patience: stop early if the accuracy does not improve after this many
    runs
EarlyStopCbk = EarlyStopping(monitor='val_loss', mode='min',
                             verbose=1, patience=10, min_delta=0.0001)

# save the best output model so far
ModelCkptCbk = ModelCheckpoint('best_model.hdf5', monitor='val_acc',
                               verbose=1, save_best_only=True,
                               mode='max')
```

Listing 3.17: Model Callbacks

The callback functions parameters are given below:

1. EarlyStopping. This function stops the training when a monitored quantity has stopped improving. The parameters used are:

   - Monitor='val_loss'. Monitor changes to the validation loss.
   - Mode='min'. Stop when the validation loss stops decreasing.
   - Verbose=1. This is just the verbosity of the output.
   - Patience=10. Training stopped after this number of epochs without improvement.
   - Min_delta=0.0001. The minimum change in the monitored quantity to qualify as an improvement.

2. ModelCheckpoint. This function saves the model after every epoch. The parameters used are:

   - Filepath. The name of the saved model file.
   - Monitor='val_acc'. Monitor changes to validation accuracy.
   - Verbose=1. This is just the verbosity of the output.
   - Save_best_only=True. Only save the best model found so far.
   - Mode='max'. Save the model with the maximum, or best, validation accuracy.

The code in Listing 3.18 trains the model using the fit() method, with the following parameters:

1. x = x_train. This is the input training data.
2. y = y_train. This is the target (classification) of the training data.
3. epochs = 100. This is the number of epochs (iterations) to train the model. A single epoch is an iteration of the entire x_train and y_train training data.
4. batch_size = 16. The number of samples to run before making a gradient update for back propagation. The batch_size number of samples are used to train the network.
5. Callbacks= [EarlyStopCbk,ModelCkptCbk]. These are the EarlyStopping and ModelCheckpoint callbacks defined earlier.
6. Validation_data=(x_validate, y_validate). This tuple is the validation data against which the model is evaluated at the end of each epoch. This data is used for validation only, and not for training.

```
history=model.fit(x_train, y_train, epochs=100,
                  callbacks=[EarlyStopCbk,ModelCkptCbk], batch_size=16,
                  validation_data=(x_validate,y_validate))
```

Listing 3.18: Training the Model

The partial output of the fit() method is shown in Figure 3.24. In general, the best model (the model with highest validation accuracy) is found at around Epoch 40. In this run, a very slight improvement is seen in a later epoch.

```
Epoch 39/100
18/18 [==============================] - ETA: 0s - loss: 0.0043 - acc: 1.0000
Epoch 00039: val_acc did not improve from 0.61806
18/18 [==============================] - 4s 205ms/step - loss: 0.0043 - acc: 1.0000 - val_loss: 1.4615 - val_acc: 0.6042
Epoch 40/100
18/18 [==============================] - ETA: 0s - loss: 0.0038 - acc: 1.0000
Epoch 00040: val_acc did not improve from 0.61806
18/18 [==============================] - 4s 206ms/step - loss: 0.0038 - acc: 1.0000 - val_loss: 1.4553 - val_acc: 0.6181
Epoch 41/100
18/18 [==============================] - ETA: 0s - loss: 0.0043 - acc: 1.0000
Epoch 00041: val_acc improved from 0.61806 to 0.62500, saving model to best_model.hdf5


Epoch 49/100
18/18 [==============================] - ETA: 0s - loss: 0.0043 - acc: 1.0000
Epoch 00049: val_acc improved from 0.62500 to 0.63194, saving model to best_model.hdf5


Epoch 00057: val_acc did not improve from 0.63194
18/18 [==============================] - 4s 207ms/step - loss: 0.0026 - acc: 1.0000 - val_loss: 1.5140 - val_acc: 0.6042
Epoch 00057: early stopping
```

Figure 3.24: Best Model Epoch

3.2.3.10 Check Model Performance

The code in listings 3.19 and 3.20 measures the model performance by comparing the training data loss versus testing data loss, or alternatively by comparing the training data accuracy versus the test data accuracy, as the number of epochs increases. Figures 3.25 and 3.27 show the corresponding training of the model.

```
from matplotlib import pyplot
pyplot.plot(history.history['loss'], label='train')
pyplot.plot(history.history['val_loss'], label='test')
loss = history.history['loss']
val_loss = history.history['val_loss']
plt.title('Training vs Validation Loss')
plt.ylabel('loss')
plt.xlabel('epoch')
# limit y axis to show difference between train and test
x1,x2,y1,y2 = plt.axis()
plt.axis((x1, x2, 0, 10))
pyplot.legend()
pyplot.show()
```

Listing 3.19: Training Loss

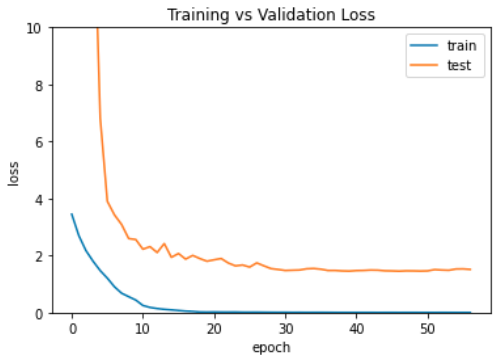


Figure 3.25: Learning Model Loss

```
from matplotlib import pyplot
pyplot.plot(history.history['acc'], label='train')
pyplot.plot(history.history['val_acc'], label='test')
acc = history.history['acc']
val_acc = history.history['val_acc']
plt.title('Training vs Validation Accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
pyplot.legend()
pyplot.show()
```

Listing 3.20: Training Accuracy

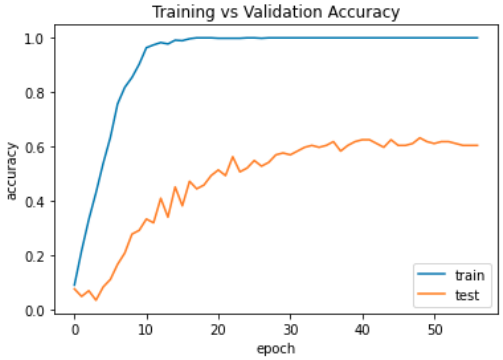


Figure 3.26: Learning Model Accuracy

3.2.3.11 Predictions

The code in Listing 3.21 loads the best model that was saved during training.

```
from tensorflow.keras.models import load_model
model=load_model('best_model.hdf5')
```

Listing 3.21: Loading The Best Model

Listing 3.22 shows a user method called predict() that accepts an audio file and retrieves a predicted class output. The model.predict() method returns an array of probabilities, and the largest probability is chosen as the classification.

```
# This function takes audio as an input parameter, and returns the
# predicted output classification
def predict(audio):
  # predict returns probability of each classification label
  prob=model.predict(audio.reshape(1, wave_dim_1, wave_dim_2, 1),
                     batch_size=32)
  # get the index of the largest probability
  index=np.argmax(prob[0])
  # use the index to get the name of the class
  return class_names[index]
```

Listing 3.22: Audio Prediction

As a way of verifying that the prediction has a success rate similar to model accuracy during training, a random set of 100 audio files from the validation set are passed to the predictor, and the prediction is checked against the actual known class. This is repeated 5 times, as shown in Listing 3.23.

```
y_true = []
y_pred = []

num_samples = 100
num_runs = 5
total_good = 0
total_good_mf = 0
for j in range(num_runs):
    num_good = 0    # number of good predictions
    num_bad = 0     # number of bad predictions
    good_mf = 0     # number of good gender predictions
    for i in range(num_samples):
        index=random.randint(0,len(x_validate)-1) # random index
        samples=x_validate[index].ravel()         # random sample
        actual_class = class_names[np.argmax(y_validate[index])]
        y_true.append(actual_class)
        predicted_class = predict(samples)
        y_pred.append(predicted_class)
        if (actual_class == predicted_class):
            num_good += 1
        else:
            num_bad += 1
        # the gender character is at index -1 (last character)
        if (actual_class[-1] == predicted_class[-1]):
          good_mf += 1
    total_good += num_good
    total_good_mf += good_mf
```

Listing 3.23: Random Data Predictions

The calculated percentages of successful predictions from the code in Listing 3.23 are shown in Figure 3.27

```
correct  62 : incorrect  38 : success rate  62.0 percent
correct gender classification  99.0 percent
correct  64 : incorrect  36 : success rate  64.0 percent
correct gender classification  97.0 percent
correct  61 : incorrect  39 : success rate  61.0 percent
correct gender classification  98.0 percent
correct  67 : incorrect  33 : success rate  67.0 percent
correct gender classification  97.0 percent
correct  60 : incorrect  40 : success rate  60.0 percent
correct gender classification  95.0 percent
overall success rate  62.8 percent
overall gender success rate  97.2 percent
```

Figure 3.27: Random Predictions

The code in Listing 3.24 creates a confusion matrix from these results, as shown in Figure 3.28.

```
from sklearn.metrics import confusion_matrix
import seaborn as sns

def plot_confusion_matrix(y_true, y_pred):
  con_matrix = confusion_matrix(y_true, y_pred)
  con_matrix_df = pd.DataFrame(con_matrix, index = LABELS, columns =
   LABELS)
  figure = plt.figure(figsize=(10, 10))
  sns.heatmap(con_matrix_df, annot=True, cmap=plt.cm.Blues)
  plt.tight_layout()
  plt.ylabel('True label')
  plt.xlabel('Predicted label')
  plt.show()

plot_confusion_matrix(y_true, y_pred)
```

Listing 3.24: Random Data Confusion Matrix

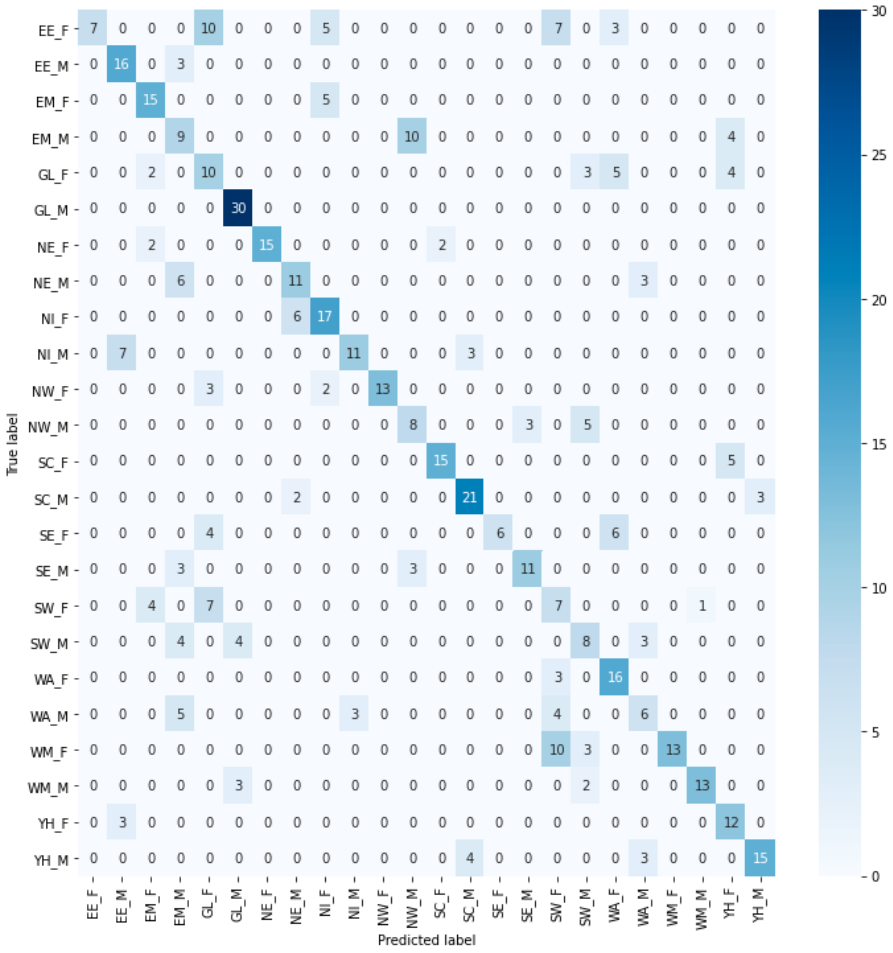


Figure 3.28: Random Data Confusion Matrix

3.2.3.12 Unseen Data

A dataset of unseen data (data that is not in either training or test datasets) was used to test the classification prediction of the model. For each region, the unseen data consists of three clips of 10 seconds for both male and female voices.

As for the training set, the audio files must be pre-processed to check the length is correct, resampled and finally transformed into the frequency domain by an STFT. To do this, call the loadfft() method, providing the unseen data directory as input, as shown in Listing 3.25.

```
1 UNSEEN_WAV_DIR = '/content/gdrive/My Drive/Project/UNSEEN_10s'
2 all_wave_stft_s, all_label_s, num_good_samples_s = loadfft(UNSEEN_WAV_DIR)
```

Listing 3.25: Pre-processing Unseen Audio Data

Every unseen audio file is then passed to the predict() method and the returned prediction is compared against the actual classification. In addition, the predicted gender is also checked against actual gender of the speaker, as shown in Listing 3.26.

```
y_true = []
y_pred = []

num_samples = len(all_wave_stft_s)
num_good = 0       # number of good predictions
num_bad = 0        # number of bad predictions
num_good_mf = 0    # number of good gender predictions

for i in range(num_samples):
      wave_stft=all_wave_stft_s[i]
      label = all_label_s[i]      #actual label
      y_true.append(label)
      plabel = predict(wave_stft)  #predicted label
      y_pred.append(plabel)
      label_mf = label[-1]        #label gender
      plabel_mf = plabel[-1]      #predicted gender
      if (all_label_s[i] == predict(wave_stft)):
          num_good += 1
      else:
          num_bad += 1
      if (label_mf == plabel_mf):
        num_good_mf += 1
print('correct ', num_good, ': incorrect ', num_bad, ':
      success rate ', (num_good/num_samples)*100, 'percent')
print('genders correct', (num_good_mf/num_samples)*100,
      'percent')
```

Listing 3.26: Classification of Unseen Data

The results of the unseen data classification are shown in Figure 3.29, with the associated confusion matrix in Figure 3.30.

```
correct  11 : incorrect  61 : success rate  15.277777777777779 percent
genders correct 84.72222222222221 percent
```

Figure 3.29: Unseen Data Prediction

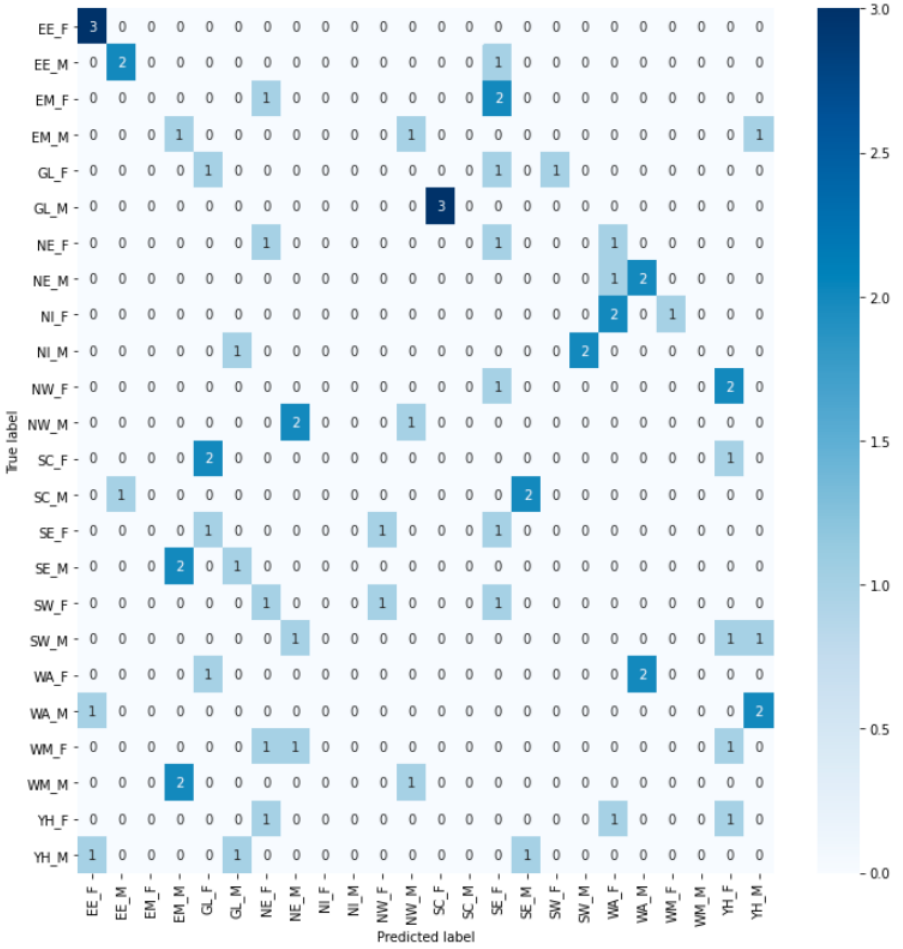


Figure 3.30: Random Data Confusion Matrix

3.2.3.13 Microphone Setup and Realtime Audio Collection

As discussed, the development of the project is made on Google Colab, which means the host running the code is a virtual machine on the Google Cloud. Consequently, it is not possible to use the microphone on such a machine to collect user audio. As such, the microphone on the local machine needs to be "connected" to the virtual machine. The FFmpeg pipe protocol is used to capture the audio from the local microphone and pipe to the server machine running the code notebook. HTML and Javascript code is used to display a "record" button that is pressed to begin recording. The same button is pressed again to stop recording. If the captured audio length is less than ten seconds, the user is prompted to try again. The audio is returned as a numpy array, and used as an input to the existing predict function, after being trimmed to exactly ten seconds long, resampled and transformed into the frequency domain by STFT.

The top level driving code to get the user audio is shown in Listing 3.27.

```
# keep looping until get an audio file
# at least 10 seconds long

user_audio_len = 0;

while user_audio_len < WAV_LEN:
  # get the audio
  user_audio, user_audio_sr = get_audio()

  # check the audio length
  user_audio_len = len(user_audio)/user_audio_sr

  # retry if the recording is not long enough
  if (user_audio_len < WAV_LEN):
    print ("len < ", WAV_LEN, " Please try again")
```

Listing 3.27: Capturing User Audio

Listing 3.28 shows part of the code for the get_audio() method called in Listing 3.27, in order to show how the ffmpeg() method is used to setup the input and output pipes for audio capture from the local microphone.

```
import io
import scipy.io.wavfile as wavfile


def get_audio():
  # display the HTML page to control the record button
  display(HTML(AUDIO_HTML))
  data = eval_js("data")
  binary = b64decode(data.split(',')[1])

  # use FFMpeg to connect the input pipe to the output pipe, in
  # order to run and stream the input to an output
  process = (ffmpeg
    .input('pipe:0')
    .output('pipe:1', format='wav')
    .run_async(pipe_stdin=True, pipe_stdout=True, pipe_stderr=True, quiet=
   True, overwrite_output=True)
  )
  output, err = process.communicate(input=binary)

  <...code removed from here for brevity...>

  # read the wav file
  sr, audio = wavfile.read(io.BytesIO(riff))

  return audio, sr
```

Listing 3.28: Using FFMpeg Pipe Protocol

The Javascript code that is executed by the *display(HTML(AUDIO_HTML))* call in get_audio() is written in Javascript, as shown in Listing 3.29 (GitHub, 2019).

```
from IPython.display import HTML, Audio
import io
import ffmpeg

from IPython.display import HTML
from google.colab.output import eval_js
from base64 import b64decode

AUDIO_HTML = """
<script>
    <...Javascript code removed for brevity...>
</script>
"""
```

Listing 3.29: Using Javascript to Manage Record Button

3.2.3.14 Calling Prediction Function With User Audio

Once the user audio has been captured and pre-processed (trimmed to the correct length and has been resampled), it then undergoes a STFT and is input to the predict() function, as shown in Listing 3.30.

```
# get STFT (Fourier Transform) of the user wav file
stft_u = librosa.stft(wav_file)

# get the absolute values of the frequency magnitude at a given time
stft_u = np.abs(stft_u)

# pass the STFT through the predict function
predicted_audio_region = predict(stft_u)

# print the predicted region
print(predicted_audio_region)
```

Listing 3.30: User Audio Prediction

3.2.3.15 Geographical Mapping

The predicted UK region is visualised on a UK map, with a different colour scheme for a male or female voice. To do this, the geopandas library is used on map datasets sourced from (UK Data Service, n.d.).

Two map files are used; one for the countries of the UK (Scotland, Wales and Northern Ireland are regions of the UK) and one for the regions within England. The regions of England are used to replace the country of England in the map of the countries of the UK.

Listing 3.31 shows the environment required for geographical mapping.

```
# install the environment for Geo Mapping
!pip install geopandas
!pip install descartes
%matplotlib inline
import pandas as pd
import matplotlib.pyplot as plt
import geopandas as gpd
```

Listing 3.31: Geomapping Environment

The mapfiles of interest are loaded as dataframes using geopandas, as shown in Listing 3.32.

```
# list the Geodata directory contents
!ls "/content/gdrive/My Drive/Project/GEODATA/infuse_ctry_2011"
!ls "/content/gdrive/My Drive/Project/GEODATA/infuse_rgn_2011"
# mapfile of all regions in England
fp = "/content/gdrive/My Drive/Project/GEODATA/infuse_rgn_2011/
    infuse_rgn_2011.shp"
map_england_df = gpd.read_file(fp)

# mapfile of all the four contries in the UK
fp = "/content/gdrive/My Drive/Project/GEODATA/infuse_ctry_2011/
    infuse_ctry_2011.shp"
map_countries_df = gpd.read_file(fp)
```

Listing 3.32: Creating DataFrames From Map Data

The dataframe for the regions in England is explored in Listing 3.33 to better understand the data that describes each region.

```
# view dataframe of England regions
print('number of regions in the UK =', len(map_england_df))
map_england_df
```

Listing 3.33: Exploring the England Regions Dataframe

The output is shown in Figure 3.31; note that the geo_label column names the region, and this will be used later to map against the region predicted by the machine learning model.

```
number of regions in the UK = 45
```

	geo_code	geo_label	label	name	geometry
0	E12000006	East of England	E12000006	East of England	POLYGON ((617106.688 343482.308, 617111.010 34...
1	E12000003	Yorkshire and The Humber	E12000003	Yorkshire and The Humber	POLYGON ((510758.595 482644.305, 510766.754 48...
2	E12000008	South East	E12000008	South East	MULTIPOLYGON (((508236.589 171797.716, 508240....
3	E12000004	East Midlands	E12000004	East Midlands	POLYGON ((454996.828 276726.552, 454974.133 27...
4	E12000007	London	E12000007	London	POLYGON ((534527.083 200056.467, 534528.375 20...
5	E12000009	South West	E12000009	South West	MULTIPOLYGON (((409873.806 239325.091, 409867....
6	E12000005	West Midlands	E12000005	West Midlands	POLYGON ((431443.073 299111.230, 431449.304 29...
7	E12000002	North West	E12000002	North West	POLYGON ((380424.183 522381.432, 380424.209 52...
8	E12000001	North East	E12000001	North East	MULTIPOLYGON (((450648.313 535677.376, 450686....

Figure 3.31: England Regions Dataframe

The code in Listing 3.34 visualises the dataframe as a map, as shown in Figure 3.32.

```
# visualise the dataframe as a map
map_england_df.plot()
```

Listing 3.34: Visualise the England Regions Dataframe

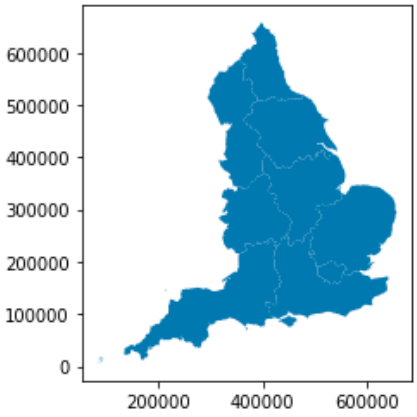


Figure 3.32: England Regions Map

In a similar way to this, Listing 3.35 explores the dataframe of the UK countries, with output shown in Figure 3.33.

	geo_code	geo_label	geo_labelw	label	name	geometry
0	N92000002	Northern Ireland	Gogledd Iwerddon	N92000002	Northern Ireland	MULTIPOLYGON (((177146.270 514561.089, 177151....
1	S92000003	Scotland	Yr Alban	S92000003	Scotland	MULTIPOLYGON (((319837.500 678417.500, 319847....
2	E92000001	England	Lloegr	E92000001	England	MULTIPOLYGON (((525567.186 412502.310, 525583....
3	W92000004	Wales	Cymru	W92000004	Wales	MULTIPOLYGON (((322909.309 299295.807, 322907....

Figure 3.33: UK Regions Dataframe

Listing 3.35 plots the dataframe map, as shown in Figure 3.34.

```
# view dataframe of UK countries
map_countries_df.head()
# visualise the dataframe as a map
map_england_df.plot()
```

Listing 3.35: Visualise the UK Regions Dataframe

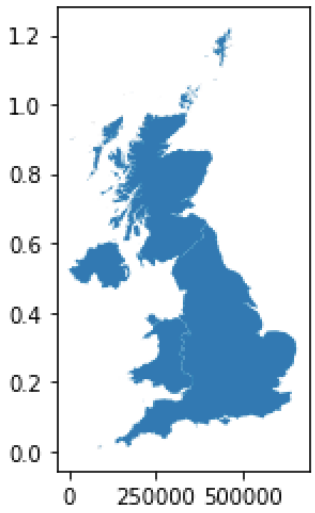


Figure 3.34: UK Regions Map

The aim is to end up with separate regions from the England; Listing 3.36 removes the England region from the UK countries map, as shown in Figure 3.35.

```
# remove England as a country
map_countries_df =
    map_countries_df[map_countries_df.geo_label != 'England']
# visualise countries without England
map_countries_df.plot()
```

Listing 3.36: UK Regions without England

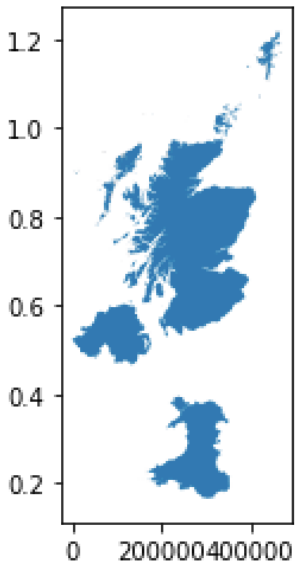


Figure 3.35: UK Regions without England

Figure 3.35 demonstrates how England is "missing" from the UK map. The next step is to re-introduce England, but this time as a collection of regions. The concat() methods concatenates the country dataframe (with England) with the England regions dataframe, as shown in Listing 3.37.

```
# add England as set of regions, replacing England as a single country
map_df = pd.concat([map_countries_df, map_england_df], sort=False)
map_df.reindex()
```

Listing 3.37: Visualise All UK Regions

The map_df dataframe now includes all of the UK regions. The next step is to be able to show a particular region in a different colour to other regions. Listing 3.39 demonstrates this concept by plotting a unique column, which will automatically assign a different colour to each unique value of the column, as shown in Figure 3.36.

```
# visualise with colours, with a different colour for each accent
map_df.plot(column='geo_label')
```

Listing 3.38: UK Regions With Colour Column

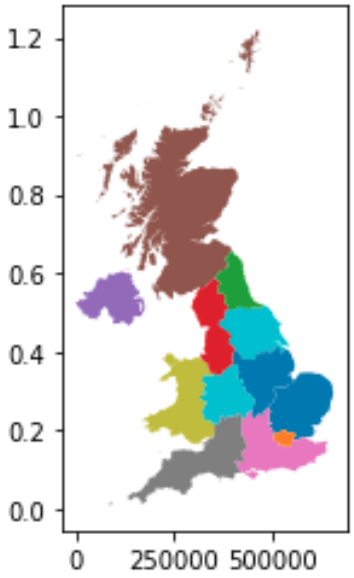


Figure 3.36: England Regions with Colour

Note that the geo_label attribute values from the map_df dataframe are not the same values as the class label outputs from the learning model. Therefore, a lookup table, in the form of a csv was created, and read as a dataframe, as shown in Listing 3.39.

```
# Read csv file to map each label to a geo label in the map file
df_csv = pd.read_csv("/content/gdrive/My Drive/Project/map.csv")

# show the csv dataframe
df_csv
```

Listing 3.39: UK Regions With Colour

Figure 3.37 shows the mapping between geo_label and class names from this dataframe.

	geo_label	audio_label	gender
0	East of England	EE_F	Female
1	East of England	EE_M	Male
2	East Midlands	EM_F	Female
3	East Midlands	EM_M	Male
4	London	GL_F	Female
5	London	GL_M	Male
6	North East	NE_F	Female
7	North East	NE_M	Male
8	Northern Ireland	NI_F	Female
9	Northern Ireland	NI_M	Male
10	North West	NW_F	Female
11	North West	NW_M	Male
12	Scotland	SC_F	Female
13	Scotland	SC_M	Male
14	South East	SE_F	Female
15	South East	SE_M	Male
16	South West	SW_F	Female
17	South West	SW_M	Male
18	Wales	WA_F	Female
19	Wales	WA_M	Male
20	West Midlands	WM_F	Female
21	West Midlands	WM_M	Male
22	Yorkshire and The Humber	YH_F	Female
23	Yorkshire and The Humber	YH_M	Male

Figure 3.37: Geo Label to Class Label Mapping Dataframe

In order to colour only the predicted region of the user audio, a new column is added, called colour, to the map_df dataframe. The new column has a value of 0 for all rows by default. This means that none of the regions are selected. The predicted region row will set the color column to 1, and when plotted, that region will be coloured separately to the remainder of the map, as shown in Listing 3.40.

```
# add a column for colour to the map dataframe, with default value 0
map_df['colour'] = 0
```

Listing 3.40: Initialise Colour Column

The last stage of the process is to take the predicted class label and lookup the corresponding UK region, which should then set the 'colour' attribute to 1 for that corresponding row in df_maps.

In addition, the gender should be extracted from the class label, and a colour scheme is chosen based on gender, in order to visualise the predicted output.

The code in Listing 3.41 locates the predicted row in the map dataframe and sets the colour attribute for that row. The colouring of the UK region map is based on one of two colour schemes, one for each gender.

```
# predicted_audio_region
print(predicted_audio_region)

# locate the predicted audio region in the lookup table, returning the geo
    label
geo_label1 = df_csv.loc[df_csv['audio_label'] == predicted_audio_region,
                        'geo_label']

print(geo_label1.iloc[0])

geo_label1 = geo_label1.iloc[0]

# locate the row in the map dataframe for this geo label
row = map_df['geo_label']==geo_label1

# for this row, set the colour to 1. All other colours are 0
map_df.at[row, 'colour']=1

# get the gender of the speaker
gender_label = df_csv.loc[df_csv['audio_label'] == predicted_audio_region,
                          'gender']

# pick a colour map scheme for the map
if (gender_label.iloc[0] == 'Female'):
  #females use a pink->yellow colour map
  gender_cmap = 'spring'
else:
  # males use a blue colour map
  gender_cmap = 'winter'
# visualise the map df, highlighting the colour column
map_df.plot(column='colour', figsize=(10,15), cmap=gender_cmap)
```

Listing 3.41: Colour Encoding UK Map

The output of the code in Listing 3.41 is shown in Figure 3.38 as the visualisation of the predicted UK regional accent and gender.

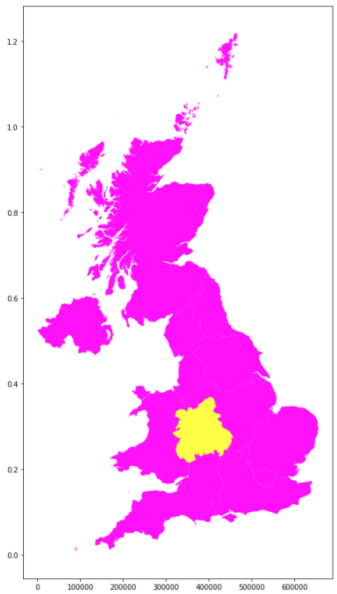


Figure 3.38: Visualisation of Prediction

3.2.4 Code Testing and Verification

The approach taken was to build in testing into the implementation, such that a test was added to verify the behaviour of code cells in the notebook as soon as practically possible. Adding such unit tests for code cells enables verification of the code behaviour at every stage. The tests added in this way are discussed in Table 3.2.

ID	Description	Test Details	Output	Pass
1	Verify Google Drive Mounted	Add a "ls" command to list the mount point contents	The label directories correctly displayed	Y
2	Verify that class labels are correct	Read all sub directories, sort and display	All of the labels are correctly displayed	Y
3	Verify that a single audio file can be read correctly by librosa	Print the wav file as a 1D array, plot a raw amplitude-time graph	1D array correctly shows amplitudes, the amplitude/graph is displayed	Y
4	Wav file can be transformed to frequency domain using librosa STFT	Plot the STFT with a db-scale spectogram	Plot shows variations of decibel level in the frequency domain	Y
5	Audio is successfully resampled to 8kHz	Play the audio before and after sampling, print out the original and resampled frequencies	The sample rate returned by librosa illustrates the change	Y
6	Verify that all audio files can be read from all source label directories	Read files and plot a barchart of files per label	Barchart correctly shows 30 samples for 24 labelled classes	Y
7	Verify the loadfft() method processes all wav files in the training dataset	Print number of samples processed, length of array of STFTs, length of array of labels	Lengths of arrays show 720 samples, random plotting from stft array shows varying STFTs	Y
8	Verify the sklearn train_test_split API correctly splits training dataset	Check the returned x and y train and validate array sizes	Array sizes correctly show lengths of 576 and 144	Y
9	Verify the Keras model shows progressively smaller convolution layer outputs, and also for Dense layers	Print the model summary	Summary shows each group of layers has increasing smaller number of outputs	Y

ID	Description	Test Details	Output	Pass
10	Verify that the validation accuracy improves during model training	Start with an approximate model, then modify and tune the model by changing the type and number of layers, change hyper parameters	Section 3.3.2 details the tuning of the model, and the associated improvement in validation accuracy	Y
11	Verify the predict() function works for a known label input	Call predict() with a known input and match the predicted output against the known label	The predicted output matched the known label for the same rate as the training accuracy	Y
12	Verify that the user audio is captured correctly	After capture, play audio file, plot a raw amplitude graph	Audio playback sound is correct, plot is valid	Y
13	Verify user audio is checked to be at least 10 seconds	Record for less than 10 seconds	Recordings of under 10 seconds not accepted, user prompted for new recording	Y
14	Verify the UK region map shapefile is displayed correctly	Manipulate the UK map files to create the England regions plus remaining countries dataframe	The displayed map correctly shows all of the English regions	Y
15	Verify a colour encoded map can be displayed	Assign a unique colour to each region and plot the map dataframe	The plot shows every region has a different colour	Y
16	Verify that a predicted region and gender is colour coded in the map visualisation	Plot the dataframe with colour encoding based on prediction	The plot shows the correct colour combination for the predicted gender and UK region	Y

Table 3.2: Unit Tests for Code Cells

3.3 Experiments Design

3.3.1 Pre-processing Data

After sourcing from YouTube, the data was uploaded to Google Drive. The original data was pre-processed prior to being used as input to the machine learning model. Table 3.3 details the experiments undertaken to check model prediction accuracy with various options for data pre-processing; these results were used to make decisions about obtaining STFTs of ten second audio clips resampled at 8kHz.

Change	Description	Result
1	Check accuracy for 5, 10, 30 second audio samples	10 seconds gives best prediction accuracy
2	Check speed of learning for 5, 10, 30 second audio samples	30 second samples too very long time to complete learning
4	Check accuracy for raw and STFT input	STFT input yields higher prediction accuracy
5	Check accuracy for 16kHz and 8kHz resampled audio	8kHz sampled audio provides marginally higher prediction accuracy

Table 3.3: Pre-processing Data Experiments

3.3.2 Tuning the Learning Model

The aim of the learning model is to produce acceptable predication rates, and ideally to do so in an efficient manner. There are many variables in producing a learning model. At the outset it may not be clear how many layers should be in the model, or what types these may be (for example, convolutional, the use of max pooling or normalisation, the number of fully connected layers etc). Further, the Keras APIs have many hyperparameters, for which the default values may not be optimal. An added complication is that a model that produces good results for one dataset may not do so for another. Given this, the approach to building a model was to first start with a basic framework, and then take an iterative approach to tuning the model by either adding or configuring layers of the model.

A starting model was built as shown in Figure 3.39

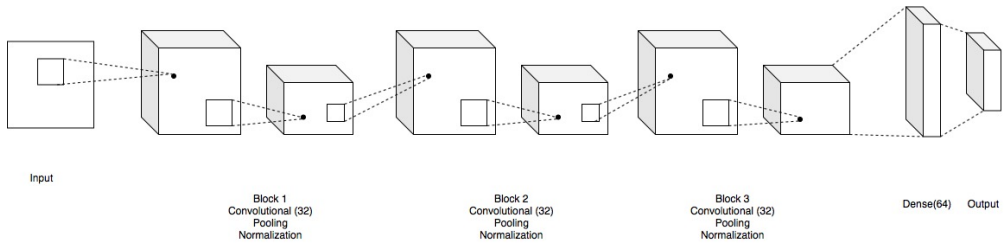


Figure 3.39: Initial Model Built

Table 3.4 shows the activation shape and size of this model at each layer.

Block	Layer	Configuration	Activation Shape	Activation Size
1	Convolution	filters=32 kernel=3 strides=1	1025.157.32	5,149,600
1	Maxpooling	pool_size=2 strides=2	513.79.32	1,296,864
1	Normalisation	filters=32 kernel=3 strides=1	513.79.32	1,296,864
2	Convolution	filters=32 kernel=3 strides=1	513.79.32	1,296,864
2	Maxpooling	pool_size=2 strides=2	257.40.32	328,960
2	Normalisation		257.40.32	328,960
3	Convolution	filters=32 kernel=3 strides=1	257.40.32	328,960
3	Maxpooling	pool_size=2 strides=2	129.20.32	82,560
3	Normalisation		129.20.32	82,560
4	Flatten		82560	82,560
5	Dense	units=64	64	64
5	Normalisation		64	64
5	Dropout	rate=0.2	64	64
6	Prediction	units=24	24	24

Table 3.4: Initial Model Activation Sizes

The validation accuracy of this model was 0.25. Modifications were made to the model, and the validation accuracy measured at each stage. If the accuracy improved, the modified model became the current best model. If the accuracy decreased, then the modification was reverted and discarded. This was repeated for many modifications, increasing the accuracy at each stage. These decisions are depicted as a decision tree, where each node shows the accuracy of the model for a given configuration and the transition from a node is the modification attempted.

This decision tree is shown in Figure 3.40, where the nodes with a bold outline show the path taken from the initial configuration to the one with the best training accuracy.

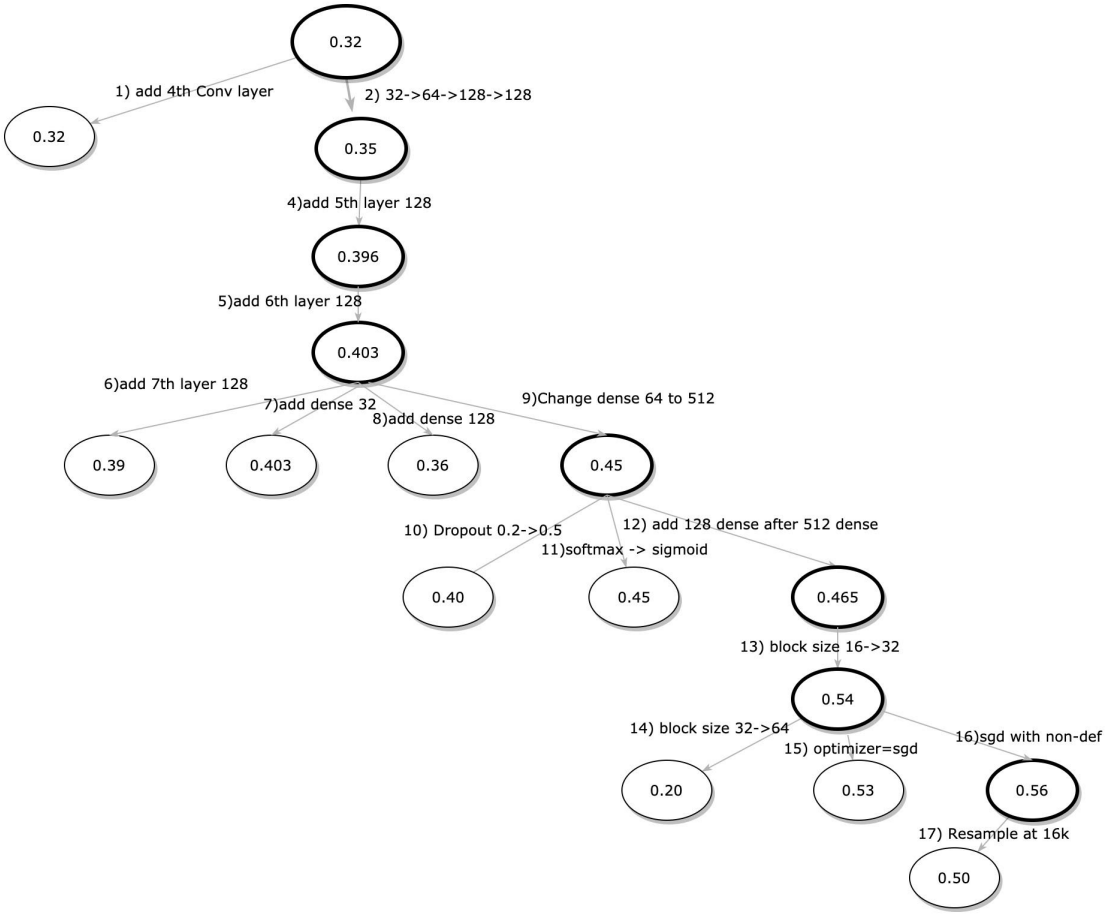


Figure 3.40: Experiment Decisions

The nodes and actions at each node of the tree are explained in Table 3.5, where each change is described in greater detail, along with the change in accuracy seen during training of the model.

Change	Description	Prev	New	Keep
1	Add 4th convolution/maxpooling layer, keeping filter=32	0.32	0.32	N
2	Create 4 convolution layers, with filter sizes (32,64,128,128)	0.32	0.35	Y
3	Add 5th convolution layer, with filter=128	0.35	0.396	Y
4	Add 6th convolution layer, with filter=128	0.396	0.403	Y
5	Add 7th convolution layer, with filter=128	0.403	0.39	N
6	Add an extra Dense layer, with units=32	0.403	0.403	N
7	Add an extra Dense layer, with units=128	0.403	0.36	N
8	Change existing Dense layer, with units change from 64 to 512	0.403	0.45	Y
9	Change existing Dropout layer, with rate change from 0.2 to 0.5	0.45	0.40	N
10	Change existing output Dense layer, with activation function change from softmax to sigmoid	0.45	0.45	N
11	Add an extra Dense layer, with units=512	0.45	0.465	N
12	Change batch size for model learning from 16 to 32	0.465	0.54	Y
13	Change batch size for model learning from 32 to 64	0.54	0.20	N
14	Change the compile model from Adam to SGD, with default SGD parameters	0.54	0.53	N
15	Change the SGD parameters from default to custom	0.54	0.56	Y
14	Change the audio resampling rate from 8kHz to 16kHz	0.56	0.50	N

Table 3.5: Learning Model Tuning Iterations

Figure 3.41 shows the final model, with some intermediate blocks of convolutional and maxpooling layers removed for brevity. Table 3.6 shows the activation shape and size for all layers in this model.

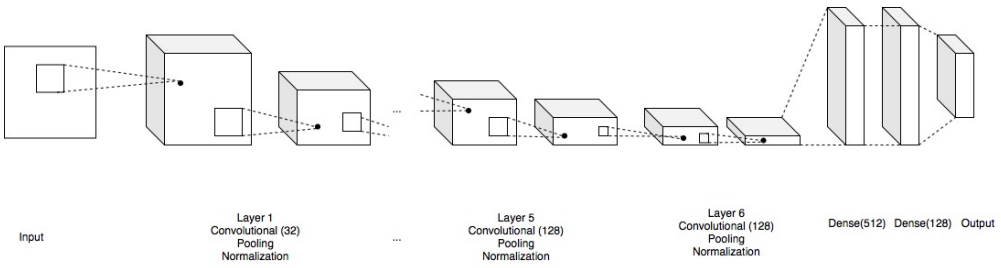


Figure 3.41: Final Model Built

Block	Layer	Configuration	Activation Shape	Activation Size
1	Convolution	filters=32 kernel=3 strides=1	1025.157.32	5,149,600
1	Maxpooling	pool_size=2 strides=2	513.79.32	1,296,864
1	Normalisation	filters=32 kernel=3 strides=1	513.79.32	1,296,864
2	Convolution	filters=64 kernel=3 strides=1	513.79.64	2,593,728
2	Maxpooling	pool_size=2 strides=2	257.40.64	657,920
2	Normalisation		257.40.64	657,920
3	Convolution	filters=128 kernel=3 strides=1	257.40.128	1,315,840
3	Maxpooling	pool_size=2 strides=2	129.20.128	330,240
3	Normalisation		129.20.128	330,240
4	Convolution	filters=128 kernel=3 strides=1	129.20.128	330,240
4	Maxpooling	pool_size=2 strides=2	65.10.128	83,200
4	Normalisation		65.10.128	83,200
5	Convolution	filters=128 kernel=3 strides=1	65.10.128	83,200
5	Maxpooling	pool_size=2 strides=2	33.5.128	21,120
5	Normalisation		33.5.128	21,120
6	Convolution	filters=128 kernel=3 strides=1	33.5.128	21,120
6	Maxpooling	pool_size=2 strides=2	17.3.128	6,528
6	Normalisation		17.3.128	6,528
7	Flatten		6528	6528
8	Dense	units=512	512	512
8	Normalisation		64	64
9	Dense	units=128	128	128
9	Normalisation		128	128
9	Dropout	rate=0.2	128	128
10	Prediction	units=24	24	24

Table 3.6: Final Model Activation Sizes

Figures 3.42 and 3.43 compare the training and validation loss and validation rates, clearly showing the improvement seen by tuning the model.

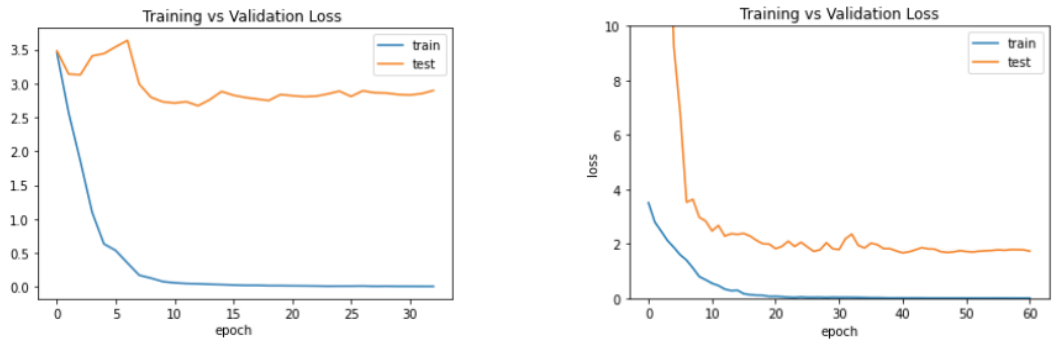


Figure 3.42: Original and Final Model Loss

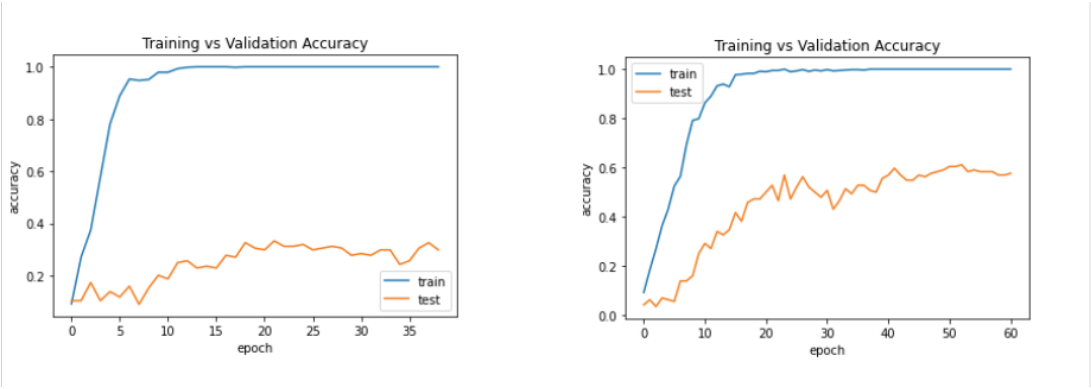


Figure 3.43: Original and Final Model Accuracy

3.3.3 Model Validation with Unrelated Dataset

The configuration and validation of the learning model has been based solely on the training dataset for this project. Given that the dataset is custom made and is relatively small in terms of samples per classification label, it would be useful to verify the model using an independent, larger dataset. The dataset chosen is the Google Speech Commands Dataset; this is a collection of 65,000 one-second long samples of 30 short words (Google AI Blog, 2017). For the test, 500 samples of the words for each digit from zero to nine were chosen as the training dataset. With this dataset, the model showed approximately 75% validation accuracy during training of the model. The unseen data produced a successful prediction rate of 90%. These accuracy rates are greater than the corresponding accuracy rates obtained from the project dataset, and suggest the learning model is well tuned.

This approach is useful in the context of creating a general learning model, but caution should be exercised in terms of the overall aims of the project, since there are major differences between the two datasets used in the experiment.

3.4 Summary

This chapter provides an in-depth discussion of the algorithms that are used for machine learning and neural networks, and how those algorithms are used to solve classification problems. The implementation of the learning model is demonstrated in great detail, with an explanation of the code employed, followed by the steps taken to test and verify the code. This is followed by an explanation of the iterative experiments carried out to tune and validate the learning model.

Chapter 4

Results

The goal of the project is to be able to predict the gender and UK regional accent of the speaker in an audio sample provided by a random user of the software. The achievement of this goal is broken down into the results of the constituent parts of the implementation.

4.1 Training Dataset

1. Deciding on appropriate pre-processing.
    - The dataset initially comprised of audio clips over 30 seconds long. This data was pre-processed prior to being used as an input to the learning model. The experiments of Section 3.3.1 show that the data should be split into 10 second segments, resampled at 8kHz and transformed into the frequency domain with STFT, as shown in Table 3.3.
2. Splitting audio into 10 second segments.
    - Section 3.2.3.1 shows that the audio can be split easily into segments and organised into class folders with appropriate naming prefixes.
3. Resampling the audio.
    - Figures 3.19 and 3.20 show that the Python package librosa was used to resample the audio to 8kHz.
4. Transforming to frequency domain.
    - Figures 3.17 and 3.18 demonstrate the use of librosa to convert the audio from the time to frequency domain by using STFTs.
5. Applying pre-processing to the whole dataset.
    - Section 3.2.3.7 shows the method used to abstract the details of applying the full set of pre-processing techniques to the whole of the dataset. Figure 3.22 shows an a sample set of the processed files visualised using spectograms.

4.2 Learning Model Training

1. The experimental design for building and tuning the learning model.

- This is discussed in Section 3.3.2.
- The initial model shown in Figure 3.39 provided an accuracy rate that varied between 0.25 and 0.32.
- The decision tree shown in Figure 3.40 with the explanation in Table 3.5 was used to iteratively build and test the model, with the objective of increasing the prediction accuracy of the learning model.
- Figures 3.42 and 3.43 show the result of the model tuning by comparing the initial model with the final model in terms of loss and accuracy. This shows that an accuracy of over 60% was attained after tuning, approximately twice the accuracy achieved by the initial model.
- Table 3.6 shows that the activation shape has a slower decrease towards the final number of classes than the initial model shown in Table 3.4.

2. Convergence to best model.
    - The model learning is set to run for 100 epochs, with early stopping if the change to validation loss is less than a delta of 0.0001 for 10 epochs. It was found that the lowest validation loss, and correspondingly, the highest validation accuracy was seen after approximately 40 epochs. Figure 3.24 shows the output of the model.fit() method, where epoch 41 yields an accuracy of 62.5%, with a small increase to 63.2% after 49 epochs. In general, the accuracy did not increase after around 40 epochs.
    - The model performance by comparison of training and validation is shown in Figure 3.25 for loss and Figure 3.27 for accuracy.
3. The model was validated by learning from an independent dataset of one second audio samples for the digits zero to nine.
    - The same model without any modification produces 75% accuracy during training using a subset of the publicly available Google Speech Commands Dataset. This validated the approach to refining the model as an audio classifier.

4.3 Random Multi-Sample Prediction

In order to self-check accuracy of the model, random samples from all of the dataset were taken.

- 100 of the audio samples from the dataset randomly chosen and fed into prediction function. This is repeated five times.
- Figure 3.27 shows that the average regional classification accuracy was approximately 63%, exactly matching the validation accuracy during training.
- Figure 3.27 also shows that the gender was also checked against the input, and this showed 97% success rate in predictions.
- The gender prediction rate is significantly higher than the regional accent prediction rate.
- The confusion matrix in Figure 3.28 shows the diagonal running from top left to bottom right is highlighted in darker colours, demonstrating the overall effectiveness in prediction.

4.4 Unseen Data Prediction

An unseen data comprising of three samples per region of each gender was used as source data for prediction of region and gender.

- Figure 3.29 shows that the region prediction was around 15%, which is lower than the prediction accuracy seen during the model learning.
- Figure 3.29 also shows that gender prediction was over 80%.
- The gender prediction rate is significantly higher than the regional accent prediction rate.
- The confusion matrix in Figure 3.30 shows a scattering of darker coloured squares, rather than highlighting a diagonal across the matrix. This is in keeping with the lower prediction accuracy for the unseen data.

4.5 Real Time User Audio Capture

Section 3.2.3.13 shows the local machine microphone was able to "connect" to the Google Colab environment, and that the user audio was captured. The audio is pre-processed just as the training and unseen pre-recorded data, and can be passed to the predict() function for classification. Playback of the audio does not suggest any loss of information to the human ear; the audio sounds as clear as the samples obtained from YouTube.

4.6 Geographical Mapping Visualisation

Geopandas can be used to load a map as a dataframe, and then the dataframe can be manipulated to tailor the map as needed.

- Figure 3.31 shows the dataframe for the regions of England, and this is visualised by the plot in Figure 3.32.
- Figure 3.33 and 3.34 show the equivalent for the UK regions; these break down as the four countries in the UK.
- Figures 3.35 and 3.36 show how these dataframes can be combined to encompass all regions in the UK as one dataframe, and also illustrate colour encoding of each region for visualisation purposes.
- The lookup table of Figure 3.37 shows the mapping between a class label and a geographical identifier in the maps dataframe. This lookup table is used to produce the final prediction visualisation seen in Figure 3.38.
- The visualisation is based on the prediction from the model. The test result of Section 3.2.4 show that a given (non-predicted) input to the visualisation highlights the expected region on the map. This is a test independent from the prediction.

4.7 Summary

A dataset to train the network has been created specifically for the project as a suitable public dataset was not available. This dataset was suitably pre-processed nd then used to build and test a model, ultimately resulting in up to 65% prediction accuracy during training for regions and 85% for gender classification, when using training and test data. For unseen data, the prediction accuracy was lower than for the training data for region classification but matched the high gender classification accuracy seen for the training data. The real time audio capture was successfully piped to the Google Colab virtual machine, and after pre-processing was passed to the prediction method in order to classify the region and gender of the speaker; these predictions were visualised on a colour-encoded map of the UK regions.

Chapter 5

Discussion and Analysis

5.1 Interpretation of Results

The model validation accuracy is around 60% for the given custom-built dataset. In order to check if this figure could be improved by changing the configuration of the model, or tuning the model, the performance of the model was measured independently of the custom-built dataset. A publicly available dataset comprised of spoken digits (zero to nine) was used as the training data to the same model. With this data, the model showed over 82% validation accuracy for the exact same model layers and hyper parameters.

For the project unseen data, the prediction rate was around 15% for the UK regions and over 80% for the gender. Again, to test the model independently of the training data, the pubic data set was also used; for that unseen data, a successful prediction rate of 90% was attained using the exact same model configuration.

These results suggest that the model is relatively well tuned and that there is confidence in the model as a method of audio classification.

The lower prediction rate for unseen data compared to the higher prediction rate for random seen data suggests that model shows bias towards the training dataset, or that there are simply not enough samples in the custom-built dataset. The project dataset has 30 samples per region per gender, whereas 500 samples per digit were used from the public single word dataset (limited from a full set of 2000, to save computational resources).

The confusion matrix of 3.12 shows that only a single class resulted in 100% prediction accuracy, and that there were significant false positives for some classes, suggesting that some regional accents "overlap" more than others. A regional accent is complex data; there are subtle nuances of variation within an accent. In addition, there may be overlaps between neighbouring accents. A speaker's pitch and tone may also vary, possibly independent of the speaker's accent. Some of these features are subtle and may result in similarity of spectrums. In order to accurately capture the full range of regional accents, including all variations within an accent and any overlap across accents, a greater amount of training data is required.

It should be noted that the gender classification does not suffer from a great loss of accuracy; the unseen data resulted in over 80% accuracy. Indeed, even with some accent mis-classifications, the false positives still match the correct gender. For example, the EE_F class has over 75% mis-classification, but all false positives still classify correctly as female.

5.2 Significance of the Findings

The project has shown that even with a limited dataset, it is possible to create an audio classification learning model that has at least a 60% success rate. The comparison with the public data set for single words shows the model can be have a success rate of at least 80%. This also validates the approach of converting audio from the time domain to the frequency domain prior to using as input to a CNN based model with multiple layers.

The gender classification showed at least 80% even for unseen data, suggesting spectogram analysis of the frequencies associated with individual genders is an effective method to distinguish between them.

It has been shown that a limited dataset size impacts the accuracy of a model, and that a larger dataset would be expected to yield higher success rates in validation.

5.3 Limitations

The greatest challenge has been creating an appropriate dataset that is large enough to provide full coverage of the full spectrum of regional accents, such that the accuracy of the model is sufficiently high that unseen data can be predicted with a level of confidence. The task of creating the dataset was a time-consuming labour intensive task where many hundreds of potential audio samples were inspected and rejected due to "impurities" such as noise, long gaps or multiple speakers. Building an appropriate learning model has also been a challenge; predicting an accent is a very different problem to classifying a single word, and so the neural network for accent prediction is expected to be more complex. The path to the optimal model is likely to take many further iterations of experimentation. The success of this project in predicting unseen audio accents has been limited in particular by the size of the dataset. A larger dataset of many hundreds of samples, maybe thousands per region would be required to further improve this project (Simard et al., 2003), and would do so without any changes to the model. As an extra step, further tuning of the model may also provide improvement in prediction rates.

5.4 Summary

The project has demonstrated some success in classification of audio by UK region and significant success in identifying the gender of a speaker. This has vindicated the approach detailed in the methodology and implementation. However, this success has been limited by the size of the custom-made dataset. It has been shown that a larger dataset used for learning by the same model produces a significantly higher success rate. A larger accent dataset would yield an increased success rate in accent classification. The data visualisation techniques are an effective and reliable way of demonstrating the predictions.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

The aim of this project was to create a machine learning model that is able to classify the UK regional accent and gender from an audio recording, and then to present the prediction using data visualisation techniques. To do so, key objectives were to create a custom training dataset, and then to investigate, implement and measure the performance of a suitable neural network, and finally be able to predict and visualise the characteristics of an audio recorded using the project software.

A suitable pre-existing dataset does not exist. It has been shown that creating such a dataset presents many challenges, since the audio samples are sourced from random longer clips found on YouTube that may contain too many pauses, contain multiple speakers or have background noise. Careful analysis was required to narrow down to a the chosen dataset.

A machine learning model consisting of multiple convolutional layers and dense layers with appropriate hyper parameter tuning was proven to have a successful prediction rate. It was shown that a prediction rate of 60% was achievable for the region of the speaker and over 80% for correct gender prediction, with a relatively small training dataset. The same model was used with a much larger independent dataset consisting of single word audio samples, where a higher prediction rate was observed. This shows that the machine learning model is well tuned for audio prediction, justifying the approach of first transforming audio samples into the frequency domain, prior to feeding to the neural network. It has been shown that a CNN is well suited to classification of audio samples transformed into the frequency domain, and that a complex network of multiple layers performs better than the simpler network initially deployed. It is felt that a larger training dataset is vital to increasing the performance of the model and ultimately improve the prediction rate of unseen data.

The application allows a user to record a sample via the local computer microphone, and uses the neural network to predict the accent and gender of the speaker, and then visualises the predictions via a map of the UK using a colour scheme to highlight both the region and gender without any textual output to explain the prediction. This is an effective way of visualising the result; in many practical applications the prediction may be an input to a subsequent software module, that can leverage that information in some way.

6.2 Future Work

The work of this project has presented a solution to the original problem; however, the accuracy of the solution is limited by the dataset used to train the neural network. Future work should include the creation of a comprehensive dataset of UK regional accents. This could perhaps be achieved by creating a smart phone application that allows individuals to upload audio clips of themselves, along with an identification of accent and gender. Such an application would need widespread adoption across the UK with the hope that many thousands of audio samples could be collected from the public. The objective would be to create a dataset that encompasses all variations of UK accents, including any variations within any given accent. The scale of this dataset should be such that it could be shared globally for use by other audio recognition software developers.

In addition to collecting more samples, data augmentation techniques could be investigated with a view to increase the footprint of existing audio samples. Such techniques are used for image recognition algorithms where it is possible to generate batches of image data from existing image data with real time data augmentation (Keras, n.d.*a*) (Machine Learning Mastery, 2019*a*). This should be investigated as a method to reproduce "mixed" or "overlapping" regional accents, thereby increasing the accent coverage of the dataset.

Given the current small dataset, the effect of using k-fold cross validation as opposed to the current hold-out method of an 80:20 dataset split, should be investigated. This method allows training on multiple train and test sets, resulting in a reduction on bias on the training set and may improve the prediction accuracy of unseen data.

The scope of the project could be expanded to include other English speaking countries to create a global English country accent predictor. In a similar way to the currently used dataset, country specific datasets could be created to identify regions or states of large countries, such as the USA.

For a larger dataset, it is possible that the current high gender prediction rate may fall, as a larger set of samples may see greater overlap in frequencies dividing a perceived male and female voice. To counter this, techniques such as fuzzy logic (Meena et al., 2013) may be used to augment the current techniques.

Further development of the machine learning model is also possible; the accuracy rate could be improved further by investigating how the model configuration could be tuned, or altered. For example, different optimisers could be tested, or variations in hyper parameters. This would be a trial and error type iterative approach.

Currently, a prediction is always given, based on the highest probability of predicted class without checking the actual probability value. Further work can be done to establish a realistic probability threshold under which a prediction should not be provided as confidence in that prediction would be too low.

Chapter 7

Reflection

I have worked on this project for 10 months, starting my planning and meeting my supervisor even before the beginning of the academic year. The project objective seemed quite daunting at first; I soon realised that an effective way to tackle the problem was to first take a high level snapshot of the problem and break it into sub-tasks, and then investigate the sub-tasks and split them further if appropriate.

Many of the problems I faced were solved by an iterative approach, where I hit many dead ends, but soon that realised each dead end and rejected potential solution was a learning experience that ultimately led the project to a solution.

It is also to reflect on things I could have done differently; I now realise that I spent too much time on searching for and rejecting public datasets that would fit the requirements. I could have realised the unsuitability of some datasets sooner, and instead spent that time sourcing my own data.

Embarking on such a long project has developed skills such as organisation, time management, keeping detailed logs of progress, being able to communicate my ideas to others and honed an approach to problem solving. I have also developed a resilience to obtaining non-expected results in experiments and using that as a feedback mechanism to try a different approach.

This project has greatly improved my self-research skills since a lot of reading around the subject of Machine Learning was required, as well as unrelated subjects like audio wave processing techniques.

I feel some disappointment that a richer and comprehensive dataset was not available as this has contributed to the lower than desired prediction accuracy for unseen data. It is unfortunate that building my own dataset was so time consuming and was difficult to balance against other academic requirements. This has taught me that sometimes trade-offs are required in the hope that all tasks can be completed in a timely and acceptable manner.

Sometimes one part of a project can take a long time to complete and impacts overall progress. I learned that it is important to believe in oneself and and one's abilities to overcome problems and to keep working hard towards the end goal.

I learned that a project of this size can require some oversight by a supervisor, and that help should be sought and accepted; even a small adjustment based on feedback can yield significant results, and for that, I am very grateful.

I feel the non-technical skills I have learned during the project are transportable to any situation in future work, academic or otherwise, where I need to approach an complex project that will span multiple months.

References

Altendorf, U. (2003), *Estuary English: Levelling at the interface of RP and south-eastern British English*, Vol. 29, Gunter Narr Verlag.

Apple (2011), 'Apple launches iphone 4s, ios 5 icloud'. (accessed August 10, 2019).
URL: *https://www.apple.com/newsroom/2011/10/04Apple-Launches-iPhone-4S-iOS-5-iCloud/*

Asta Speaks (2014), 'Audrey: The first speech recognition system'. (accessed August 4, 2019).
URL: *https://astaspeaks.wordpress.com/2014/10/13/audrey-the-first-speech-recognition-system/*

Bahdanau, D., Cho, K. and Bengio, Y. (2014), 'Neural machine translation by jointly learning to align and translate', *arXiv preprint arXiv:1409.0473* .

Bahl, L., Brown, P., De Souza, P. and Picheny, M. (1988), Acoustic markov models used in the tangora speech recognition system, *in* 'ICASSP-88., International Conference on Acoustics, Speech, and Signal Processing', IEEE, pp. 497–500.

BBC (2017), 'The machines that learned to listen'. (accessed August 7, 2019).
URL: *https://www.bbc.com/future/article/20170214-the-machines-that-learned-to-listen*

Britannica (2020), 'Stephen fry'. (accessed October 11, 2019).
URL: *https://www.britannica.com/biography/Stephen-Fry*

British Library (n.d.), 'Accents dialects'. (accessed October 9, 2019).
URL: *https://sounds.bl.uk/Accents-and-dialects*

Buscema, M. (1998), 'Back propagation neural networks', *Substance use & misuse* **33**(2), 233–270.

Business Insider (2018), 'Amazon wants your help teaching alexa new languages — and it could help in its fight against google'. (accessed August 10, 2019).
URL: *https://www.businessinsider.com/cleo-new-alexa-skill-for-teaching-foreign-languages-2018-3?r=USIR=T*

Call Rail (2017), 'What is speech recognition software?'. (accessed September 19, 2019).
URL: *https://www.callrail.com/blog/speech-recognition-software/*

Common Voice (n.d.), 'What's inside the common voice dataset?'. (accessed October 9, 2019).
URL: *https://voice.mozilla.org/en/datasets*

Corkill, D. D. (1991), 'Blackboard systems', *AI expert* **6**(9), 40–47.

Das, S. and Picheny, M. (1996), Issues in practical large vocabulary isolated word recognition: The ibm tangora system, *in* 'Automatic Speech and Speaker Recognition', Springer, pp. 457–479.

Data Science Central (2017), 'Artificial neural network (ann) in machine learning'. (accessed September 19, 2019).
URL: *https://www.datasciencecentral.com/profiles/blogs/artificial-neural-network-ann-in-machine-learning*

Erman, L. D., Hayes-Roth, F., Lesser, V. R. and Reddy, D. R. (1980), 'The hearsay-ii speech-understanding system: Integrating knowledge to resolve uncertainty', *ACM Computing Surveys (CSUR)* **12**(2), 213–253.

Forbes (2018), 'The rise in computing power: Why ubiquitous artificial intelligence is now a reality'. (accessed August 10, 2019).
URL: *https://www.forbes.com/sites/intelai/2018/07/17/the-rise-in-computing-power-why-ubiquitous-artificial-intelligence-is-now-a-reality/544f11fc1d3f*

FZ Blogs (2011), 'Speech recognition in 1920s: Radio rex – the first speech recognition machine?'. (accessed August 4, 2019).
URL: *https://ileriseviye.wordpress.com/2011/02/17/speech-recognition-in-1920s-radio-rex-the-first-speech-recognition-machine/*

George Mason University (2020), 'The speech accent archive'. (accessed October 9, 2019).
URL: *https://accent.gmu.edu/*

GitHub (2019), 'Record audio from your microphone'. (accessed March 31, 2020).
URL: *https://gist.github.com/ricardodeazambuja/03ac98c31e87caf284f7b06286ebf7fd*

Google AI Blog (2017), 'Launching the speech commands dataset'. (accessed April 2, 2020).
URL: *https://ai.googleblog.com/2017/08/launching-speech-commands-dataset.html*

Google AI Blog (2019), 'Introducing translatotron: An end-to-end speech-to-speech translation model'. (accessed August 10, 2019).
URL: *https://ai.googleblog.com/2019/05/introducing-translatotron-end-to-end.html*

HowStuffWorks (n.d.), 'How speech recognition works'. (accessed August 4, 2019).
URL: *https://electronics.howstuffworks.com/gadgets/high-tech-gadgets/speech-recognition1.htm*

IBM (n.d.), 'Ibm shoebox'. (accessed August 4, 2019).
URL: *https://www.ibm.com/ibm/history/exhibits/specialprod1/specialprod1_7.html*

Independent (2015), 'British accents dying out across the uk - apart from glaswegian'. (accessed October 10, 2019).
URL: *https://www.independent.co.uk/news/uk/home-news/british-accents-dying-out-across-the-uk-apart-from-one-that-s-flourishing-a6737956.html*

InfoWorld (2019), 'What is tensorflow? the machine learning library explained'. (accessed September 24, 2019).
URL: *https://www.infoworld.com/article/3278008/what-is-tensorflow-the-machine-learning-library-explained.html*

International Dialects of English Archive (n.d.), 'England'. (accessed October 9, 2019).
URL: *https://www.dialectsarchive.com/england*

Jantzen, J. (1998), 'Introduction to perceptron networks', *Technical University of Denmark, Lyngby, Denmark, Technical Report* .

Keras (n.d.*a*), 'Image data preprocessing'. (accessed April 24, 2020).
URL: *https://keras.io/preprocessing/image/*

Keras (n.d.*b*), 'Keras: The python deep learning library'. (accessed September 24, 2019).
URL: *https://keras.io/*

Keras (n.d.*c*), 'Optimizers'. (accessed March 30, 2020).
URL: *https://keras.io/optimizers/*

Keras (n.d.*d*), 'Probabilistic losses'. (accessed March 26, 2020).
URL: *https://keras.io/api/losses/probabilistic_losses/categoricalcrossentropy-class*

KETKAR, N. (2017), 'Deep learning with python, a hands-on introduction, apress edition, 160p'.

Kingma, D. P. and Ba, J. (2014), 'Adam: A method for stochastic optimization', *arXiv preprint arXiv:1412.6980* .

Krizhevsky, A., Sutskever, I. and Hinton, G. E. (2012), Imagenet classification with deep convolutional neural networks, *in* 'Advances in neural information processing systems', pp. 1097–1105.

Lowerre, B. T. (1976), The harpy speech recognition system, Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF COMPUTER SCIENCE.

Machine Learning Mastery (2016), 'Crash course on multi-layer perceptron neural networks'. (accessed September 20, 2019).
URL: *https://machinelearningmastery.com/neural-networks-crash-course/*

Machine Learning Mastery (2019*a*), 'How to configure image data augmentation in keras'. (accessed April 24, 2020).
URL: *https://machinelearningmastery.com/how-to-configure-image-data-augmentation-when-training-deep-learning-neural-networks/*

Machine Learning Mastery (2019*b*), 'Supervised and unsupervised machine learning algorithms'. (accessed September 19, 2019).
URL: *https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/*

Medium (2018*a*), 'Google colab free gpu tutorial'. (accessed September 24, 2019).
URL: *https://medium.com/deep-learning-turkey/google-colab-free-gpu-tutorial-e113627b9f5d*

Medium (2018*b*), 'Hidden markov models — part 1: the likelihood problem'. (accessed August 7, 2019).
URL: *https://medium.com/@Ayra_Lux/hidden-markov-models-part-1-the-likelihood-problem-8dd1066a784e*

Medium (2018*c*), 'Speech recognition technology: The past, present, and future.'. (accessed August 9, 2019).
URL: *https://medium.com/swlh/the-past-present-and-future-of-speech-recognition-technology-cf13c179aaf*

Medium (2019), 'Dense layers explained in a simple way'. (accessed March 25, 2020).
URL: *https://medium.com/datathings/dense-layers-explained-in-a-simple-way-62fe1db0ed75*

Meena, K., Subramaniam, K. R. and Gomathy, M. (2013), 'Gender classification in speech recognition using fuzzy logic and neural network.', *Int. Arab J. Inf. Technol.* **10**(5), 477–485.

Netherlands Worldwide (n.d.), 'Regions in the united kingdom'. (accessed August 4, 2019).
URL: *https://www.netherlandsworldwide.nl/countries/united-kingdom/doing-business/uk-regions*

Nii, H. P. (1986), 'The blackboard model of problem solving and the evolution of blackboard architectures', *AI magazine* **7**(2), 38–38.

Pentathlon GB (n.d.), 'Regions'. (accessed November 6, 2019).
URL: *https://www.pentathlongb.org/about/regions*

ProPublica (2013), 'Faq: What you need to know about the nsa's surveillance programs'. (accessed August 9, 2019).
URL: *https://www.propublica.org/article/nsa-data-collection-faq*

SALu (2017), 'Sgd greater than adam?? which one is the best optimizer: Dogs-vs-cats toy experiment'. (accessed March 25, 2020).
URL: *https://shaoanlu.wordpress.com/2017/05/29/sgd-all-which-one-is-the-best-optimizer-dogs-vs-cats-toy-experiment/*

Schalkwyk, J., Beeferman, D., Beaufays, F., Byrne, B., Chelba, C., Cohen, M., Kamvar, M. and Strope, B. (2010), "your word is my command": Google search by voice: A case study, *in* 'Advances in speech recognition', Springer, pp. 61–90.

Schaller, R. R. (1997), 'Moore's law: past, present and future', *IEEE spectrum* **34**(6), 52–59.

Schmidhuber, J. (2015), 'Deep learning in neural networks: An overview', *Neural networks* **61**, 85–117.

Science Direct (2006), 'Short-time fourier transform'. (accessed March 25, 2020).
URL: *https://www.sciencedirect.com/topics/engineering/short-time-fourier-transform*

Sigmund, M. (2008), 'Gender distinction using short segments of speech signal', *International Journal of Computer Science and Network Security* **8**(10), 159–162.

Simard, P. Y., Steinkraus, D., Platt, J. C. et al. (2003), Best practices for convolutional neural networks applied to visual document analysis., *in* 'Icdar', Vol. 3.

Tech Target (2018), 'Siri'. (accessed August 10, 2019).
URL: *https://searchmobilecomputing.techtarget.com/definition/Siri*

The Economic Times India (2011), 'iphone 4s siri: Apple's voice recognition technology with a brain of its own'. (accessed August 4, 2019).
URL: *https://economictimes.indiatimes.com/tech/hardware/iphone-4s-siri-apples-voice-recognition-technology-with-a-brain-of-its-own/articleshow/10360379.cms*

The Guardian (2018), 'Received pronunciation may be dying out – but its passing is long overdue'. (accessed October 12, 2019).
URL: *https://www.theguardian.com/science/shortcuts/2018/may/22/received-pronunciation-may-be-dying-out-but-its-passing-is-long-overdue*

The Intercept (2015), 'How the nsa converts spoken words into searchable text'. (accessed August 9, 2019).
URL: *https://theintercept.com/2015/05/05/nsa-speech-recognition-snowden-searchable-text/*

The Intercept (2018), 'Finding your voice'. (accessed August 9, 2019).
URL: *https://theintercept.com/2018/01/19/voice-recognition-technology-nsa/*

Towards Data Science (2017*a*), 'Learning rate schedules and adaptive learning rate methods for deep learning'. (accessed March 29, 2020).
URL: *https://towardsdatascience.com/learning-rate-schedules-and-adaptive-learning-rate-methods-for-deep-learning-2c8f433990d1*

Towards Data Science (2017*b*), 'Stochastic gradient descent with momentum'. (accessed March 26, 2020).
URL: *https://towardsdatascience.com/stochastic-gradient-decent-with-momentum-a84097641a5d*

Towards Data Science (2018*a*), 'A comprehensive guide to convolutional neural networks'. (accessed March 25, 2020).
URL: *https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53*

Towards Data Science (2018*b*), 'Over 1.5 tb's of labeled audio datasets'. (accessed October 9, 2019).
URL: *https://towardsdatascience.com/a-data-lakes-worth-of-audio-datasets-b45b88cd4ad*

Towards Data Science (2018*c*), 'Understanding neural networks. from neuron to rnn, cnn, and deep learning'. (accessed September 20, 2019).
URL: *https://towardsdatascience.com/understanding-neural-networks-from-neuron-to-rnn-cnn-and-deep-learning-cd88e90e0a90*

Towards Data Science (2019), 'An introduction to perceptron algorithm'. (accessed March 25, 2020).
URL: *https://towardsdatascience.com/an-introduction-to-perceptron-algorithm-40f2ab4e2099*

UK Data Service (n.d.), 'Uk boundary datasets'. (accessed March 28, 2020).
URL: *https://borders.ukdataservice.ac.uk/easy_download.html*

Venture Beat (2019), 'Why companies like amazon manually review voice data'. (accessed August 10, 2019).
URL: *https://venturebeat.com/2019/04/15/why-companies-like-amazon-manually-review-voice-data/*

Voquent (n.d.), 'Any language. any accent.'. (accessed October 9, 2019).
URL: *https://www.voquent.com/voice-actors/?language*

Wilson, A. C., Roelofs, R., Stern, M., Srebro, N. and Recht, B. (2017), The marginal value of adaptive gradient methods in machine learning, *in* 'Advances in Neural Information Processing Systems', pp. 4148–4158.

YouTube (2014), 'One woman, 17 british accents - anglophenia ep 5'. (accessed August 4, 2019).
URL: *https://www.youtube.com/watch?v=FyyT2jmVPAk*

YouTube (2015*a*), 'Hidden markov models'. (accessed August 4, 2019).
URL: *https://www.youtube.com/watch?v=9yl4XGp5OEg*

YouTube (2015*b*), 'Stephen fry - full address'. (accessed October 11, 2019).
URL: *https://www.youtube.com/watch?v=IporlmXXDeY*

YouTube (2016), 'Radio rex - first voice controlled device - circa 1920'. (accessed August 4, 2019).
URL: *https://www.youtube.com/watch?v=AdUi_St-BdMt=365s*

Appendix A

Dataset Sources

Appendix A of the report presents the sources used to construct the UK Regional Accent dataset. Each audio file is labelled with its relevant gender and number from 0-9, including an extra audio file per gender used as 'unseen data'.

A.0.1 Northern Ireland

A.0.1.1 Female

- [F00] - https://www.youtube.com/watch?v=UXQKMDZk_4A
- [F01] - https://www.youtube.com/watch?v=1A-9tdJ8SMg
- [F02] - https://www.youtube.com/watch?v=u-qmA7aMEOg
- [F03] - https://www.youtube.com/watch?v=8Rqsixnzurc
- [F04] - https://www.youtube.com/watch?v=VWXYrjc5NG8
- [F05] - https://www.youtube.com/watch?v=O3wIY_DgBds
- [F06] - https://www.youtube.com/watch?v=kS8-0NlhtD8
- [F07] - https://www.youtube.com/watch?v=nKPBzlxPPq0
- [F08] - https://www.youtube.com/watch?v=vAqkBxg-aik
- [F09] - https://www.youtube.com/watch?v=IbBCl8IJ8YM
- [UNSEEN] - https://www.youtube.com/watch?v=zkMZIbRYhO8

A.0.1.2 Male

- [M00] - https://www.youtube.com/watch?v=2-gYCQT6Wng
- [M01] - https://www.youtube.com/watch?v=j6RzSoTO8jk
- [M02] - https://www.youtube.com/watch?v=mwFmBIsCmPct=213s
- [M03] - https://www.youtube.com/watch?v=qAE3_U1VzOg
- [M04] - https://www.youtube.com/watch?v=kJfOeAR_awAt=678s
- [M05] - https://www.youtube.com/watch?v=jmsaHDh0Qtk

- [M06] - https://www.youtube.com/watch?v=d3qB3RPtN9U
- [M07] - https://www.youtube.com/watch?v=Oe6JPSJF6xg
- [M08] - https://www.youtube.com/watch?v=7jk7XFirTNg
- [M09] - https://www.youtube.com/watch?v=X8iXhQeg4SI
- [UNSEEN] - https://www.youtube.com/watch?v=GQs-sbF8I9w

A.0.2 Scotland

A.0.2.1 Female

- [F00] - https://www.youtube.com/watch?v=_vPvtFjB71U
- [F00] - https://www.youtube.com/watch?v=t9AJsB2LtM0
- [F02] - https://www.youtube.com/watch?v=r9hluC75MX0
- [F03] - https://www.youtube.com/watch?v=0mZwG9w6LHQ
- [F04] - https://www.youtube.com/watch?v=M8UrrVnmZQE
- [F05] - https://www.youtube.com/watch?v=j5cbghpL180
- [F06] - https://www.youtube.com/watch?v=vM91-PHYRgwt=67s
- [F07] - https://www.youtube.com/watch?v=_knxIRZCHTQ
- [F08] - https://www.youtube.com/watch?v=1p-SxJsIlFk
- [F09] - https://www.youtube.com/watch?v=tnjqSTnY9Ps
- [UNSEEN] - https://www.youtube.com/watch?v=SGKoekcThLE

A.0.2.2 Male

- [M00] - https://www.youtube.com/watch?v=0P5i3pIWU8Q
- [M01] - https://www.youtube.com/watch?v=aA1H4iLap44t=3141s
- [M02] - https://www.youtube.com/watch?v=AFuc2CuCZUc
- [M03] - https://www.youtube.com/watch?v=HXJI8N7WM1E
- [M04] - https://www.youtube.com/watch?v=_L3kyGyJD0M
- [M05] - https://www.youtube.com/watch?v=FognyQZ_PPQ
- [M06] - https://www.youtube.com/watch?v=C14hkhY7DuU
- [M07] - https://www.youtube.com/watch?v=cPdn5zwZcgk
- [M08] - https://www.youtube.com/watch?v=QJftaIBp_7o
- [M09] - https://www.youtube.com/watch?v=QyOZTqTIF6s
- [UNSEEN] - https://www.youtube.com/watch?v=NRwIYjqvEI8

A.0.3 Wales

A.0.3.1 Female

- [F00] - https://www.youtube.com/watch?v=vMtpcvPwvRY
- [F01] - https://www.youtube.com/watch?v=m6SrwWtloxg
- [F02] - https://www.youtube.com/watch?v=hawPzbzZDeY
- [F03] - https://www.youtube.com/watch?v=GNGije2cUT8
- [F04] - https://www.youtube.com/watch?v=DqnujaoDV7c
- [F05] - https://www.youtube.com/watch?v=vWk6ekzSc_M
- [F06] - https://www.youtube.com/watch?v=7fULJ8l-uVE
- [F07] - https://www.youtube.com/watch?v=somVQmbtZQ4
- [F08] - https://www.youtube.com/watch?v=n4h5iAHGeaE
- [F09] - https://www.youtube.com/watch?v=RwOJE4UII0Q
- [UNSEEN] - https://www.youtube.com/watch?v=WYJWalemraY

A.0.3.2 Male

- [M00] - https://www.youtube.com/watch?v=Y6sXh9uv3mo
- [M01] - https://www.youtube.com/watch?v=mO-Kcn37-ME
- [M02] - https://www.youtube.com/watch?v=URa3ANsUT-o
- [M03] - https://www.youtube.com/watch?v=ICJXtVdSvkU
- [M04] - https://www.youtube.com/watch?v=6J1I7rap-Gc
- [M05] - https://www.youtube.com/watch?v=5icrv1YJr8g
- [M06] - https://www.youtube.com/watch?v=aS77MkauFYk
- [M07] - https://www.youtube.com/watch?v=AXVcHrMBgkI
- [M08] - https://www.youtube.com/watch?v=04KTau1b29E
- [M09] - https://www.youtube.com/watch?v=qMbontVP2AQ
- [UNSEEN] - https://www.youtube.com/watch?v=B8daes-0vSot=255s

A.0.4 England

A.0.4.1 North East - Female

- [F00] - https://www.youtube.com/watch?v=m16bq0YdVHw
- [F01] - https://www.youtube.com/watch?v=KY-5bgR-0ect=3s
- [F02] - https://www.youtube.com/watch?v=w2woc46LXkw
- [F03] - https://www.youtube.com/watch?v=xIKOOLQUxgU
- [F04] - https://www.youtube.com/watch?v=zXqFpSUg5Ow
- [F05] - https://www.youtube.com/watch?v=9tQjdUiZi3Yt=80s
- [F06] - https://www.youtube.com/watch?v=TBeUBSyN920
- [F07] - https://www.youtube.com/watch?v=dvRxnOHXtks
- [F08] - https://www.youtube.com/watch?v=Zpq43RA_bGc
- [F09] - https://www.youtube.com/watch?v=KXn5S3n6QIE
- [UNSEEN] - https://www.youtube.com/watch?v=BccFksS–z0

A.0.4.2 North East - Male

- [M00] - https://www.youtube.com/watch?v=ZBD46hbwjyc
- [M01] - https://www.youtube.com/watch?v=n_AtePILQPA
- [M02] - https://www.youtube.com/watch?v=Nubd83V2jkU
- [M03 - https://www.youtube.com/watch?v=309PAADTL2st=95s
- [M04] - https://www.youtube.com/watch?v=c98ypUPZfW0
- [M05] - https://www.youtube.com/watch?v=FVVMIrKbWdQ
- [M06] - https://www.youtube.com/watch?v=hgGyUGXTP7I
- [M07] - https://www.youtube.com/watch?v=KUJlpnaa13s
- [M08] - https://www.youtube.com/watch?v=LDDZsAnFouY
- [M09] - https://www.youtube.com/watch?v=RpE_vEzZYD0
- [UNSEEN] - https://www.youtube.com/watch?v=PxZm6CahqFU

A.0.4.3 North West - Female

- [F00] - https://www.youtube.com/watch?v=8Hqz6EiShpct=2s
- [F01] - https://www.youtube.com/watch?v=vXWooQiq2oQ
- [F02] - https://www.youtube.com/watch?v=UydM0NvTkg8
- [F03] - https://www.youtube.com/watch?v=gP5qDdPmku4
- [F04] - https://www.youtube.com/watch?v=IvviO_E6c5c
- [F05] - https://www.youtube.com/watch?v=wVpMgq8KZFQ
- [F06] - https://www.youtube.com/watch?v=wMldw7-1nIwt=5s
- [F07] - https://www.youtube.com/watch?v=q3rwdKSeeW4
- [F08] - https://www.youtube.com/watch?v=5KePjmrdEWM
- [F09] - https://www.youtube.com/watch?v=x01wky3NtKc
- [UNSEEN] - https://www.youtube.com/watch?v=zild32BvIOY

A.0.4.4 North West - Male

- [M00] - https://www.youtube.com/watch?v=wILrmpvTdnY
- [M01] - https://www.youtube.com/watch?v=1qskq_3kblw
- [M02] - https://www.youtube.com/watch?v=zs1Zn9KSeuA
- [M03] - https://www.youtube.com/watch?v=7BWKY3ZiFug
- [M04] - https://www.youtube.com/watch?v=XBhKC_yDGhk
- [M05] - https://www.youtube.com/watch?v=priqnITOqWA
- [M06] - https://www.youtube.com/watch?v=6j8B-DFuIbY
- [M07] - https://www.youtube.com/watch?v=9fWn6t4VhLA
- [M08] - https://www.youtube.com/watch?v=HocXPsCrh7s
- [M09] - https://www.youtube.com/watch?v=5Fq2ZCj6o3kt=829s
- [UNSEEN] - https://www.youtube.com/watch?v=yjpGey7E0DU

A.0.4.5 Yorkshire and the Humber - Female

- [F00] - https://www.youtube.com/watch?v=vhkZgNkmk0A
- [F01] - https://www.youtube.com/watch?v=57DeyUEY8xY
- [F02] - https://www.youtube.com/watch?v=zi_uBEOF1mo
- [F03] - https://www.youtube.com/watch?v=WB98O0QzuRQ
- [F04] - https://www.youtube.com/watch?v=kQlfHWfgx9k
- [F05] - https://www.youtube.com/watch?v=CsUJYaWVd3M
- [F06] - https://www.youtube.com/watch?v=Zh5xq6sNV-w
- [F07] - https://www.youtube.com/watch?v=t-abFrgb4Mo
- [F08] - https://www.youtube.com/watch?v=6V54g0314UA
- [F09] - https://www.youtube.com/watch?v=f3HxTLj_Zwg
- [UNSEEN] - https://www.youtube.com/watch?v=TJKm1DvDJ-Q

A.0.4.6 Yorkshire and the Humber - Male

- [M00] - https://www.youtube.com/watch?v=AXikoUGihVQ
- [M01] - https://www.youtube.com/watch?v=nMERO8spsD4
- [M02] - https://www.youtube.com/watch?v=-7wvcOXXzsY
- [M03] - https://www.youtube.com/watch?v=PYcP1YxhvTM
- [M04] - https://www.youtube.com/watch?v=UF1oshULgng
- [M05] - https://www.youtube.com/watch?v=Uohritdc0ys
- [M06] - https://www.youtube.com/watch?v=cDQHbzAZIhs
- [M07] - https://www.youtube.com/watch?v=63IhJOnUEq0
- [M08] - https://www.youtube.com/watch?v=NC2AMUX_44Qt=560s
- [M09] - https://www.youtube.com/watch?v=e-DIwEvCD6U
- [UNSEEN] - https://www.youtube.com/watch?v=dCk6fSQ21rY

A.0.4.7 East Midlands - Female

- [F00] - https://www.youtube.com/watch?v=J4YvM1PiB2I
- [F01] - https://www.youtube.com/watch?v=d_SsWqd2wN0
- [F02] - https://www.youtube.com/watch?v=V-zt1vdPCn8
- [F03] - https://www.youtube.com/watch?v=03fE8cWqudA
- [F04] - https://www.youtube.com/watch?v=ITWH1YoYxS0
- [F05] - https://www.youtube.com/watch?v=xRYJdE7YJmE
- [F06] - https://www.youtube.com/watch?v=sjWagfNZL7E
- [F07] - https://www.youtube.com/watch?v=hdiCXgoLykI
- [F08] - https://www.youtube.com/watch?v=fge2hfMkJ1w
- [F09] - https://www.youtube.com/watch?v=ZM47GRpQpVQ
- [UNSEEN] - https://www.youtube.com/watch?v=HUeeDbHwRDQt=42s

A.0.4.8 East Midlands - Male

- [M00] - https://www.youtube.com/watch?v=-SEzTUPxiLo
- [M01] - https://www.youtube.com/watch?v=CxFwUnFY36c
- [M02] - https://www.youtube.com/watch?v=Uy4NCy0QgMU
- [M03] - https://www.youtube.com/watch?v=dll0DeugRRw
- [M04] - https://www.youtube.com/watch?v=PEIX1yFmtuUt=187s
- [M05] - https://www.youtube.com/watch?v=p2tdZiXiEa4
- [M06] - https://www.youtube.com/watch?v=vz3XVPGWMUI
- [M07] - https://www.youtube.com/watch?v=aOIkYy57L34
- [M08] - https://www.youtube.com/watch?v=caGrcgZHWv4
- [M09] - https://www.youtube.com/watch?v=i0UttkgxIIo
- [UNSEEN] - https://www.youtube.com/watch?v=7WIy1vX3rW4

A.0.4.9 West Midlands - Female

- [F00] - https://www.youtube.com/watch?v=yCzIrGmJiX4t=165s
- [F01] - https://www.youtube.com/watch?v=HquCFc_HwMc
- [F02] - https://www.youtube.com/watch?v=Y3CBv4JRNdg
- [F03] - https://www.youtube.com/watch?v=yOG8012f03Q
- [F04] - https://www.youtube.com/watch?v=qzQvq1Jyk5I
- [F05] - https://www.youtube.com/watch?v=tb_1BRs7Rsw
- [F06] - https://www.youtube.com/watch?v=a6l5EWSSPF0
- [F07] - https://www.youtube.com/watch?v=mF9NE1t0C3E
- [F08] - https://www.youtube.com/watch?v=W4o0pjIW240
- [F09] - https://www.youtube.com/watch?v=x4ACN_hXmYw
- [UNSEEN] - https://www.youtube.com/watch?v=fSQ36vQfo3w

A.0.4.10 West Midlands - Male

- [M00] - https://www.youtube.com/watch?v=bgp0AatNDLU
- [M01] - https://www.youtube.com/watch?v=vpCQYPaB5ao
- [M02] - https://www.youtube.com/watch?v=7T84pISwKdg
- [M03] - https://www.youtube.com/watch?v=cLAY0osSkFI
- [M04] - https://www.youtube.com/watch?v=JmP6DyY8pfM
- [M05] - https://www.youtube.com/watch?v=ANUZXcdHzYY
- [M06] - https://www.youtube.com/watch?v=yIoBgpaCLLs
- [M07] - https://www.youtube.com/watch?v=iS8JDa837s8
- [M08] - https://www.youtube.com/watch?v=-47sqwkLEtk
- [M09] - https://www.youtube.com/watch?v=dSRgQvCvRiY
- [UNSEEN] - https://www.youtube.com/watch?v=K6776kra9RE

A.0.4.11 East of England - Female

- [F00] - https://www.youtube.com/watch?v=HpcGhkDLPZk
- [F01] - https://www.youtube.com/watch?v=Q90GV88Cq74
- [F02] - https://www.youtube.com/watch?v=ugmw8VJ3zV8
- [F03] - https://www.youtube.com/watch?v=ejdr7_HIsRg
- [F04] - https://www.youtube.com/watch?v=kpG4T8xUqF0
- [F05] - https://www.youtube.com/watch?v=MgbINwqN4xw
- [F06] - https://www.youtube.com/watch?v=0EHWCv0PSj8
- [F07] - https://www.youtube.com/watch?v=aul-XS7iNzY
- [F08] - https://www.youtube.com/watch?v=miIg8hGqSxQ
- [F09] - https://www.youtube.com/watch?v=DYPEBr4DkVw
- [UNSEEN] - https://www.youtube.com/watch?v=UResfbrjTw8

A.0.4.12 East of England - Male

- [M00] - https://www.youtube.com/watch?v=UnkyMuVD_Sw
- [M01] - https://www.youtube.com/watch?v=DaOIQ8CD2VM
- [M02] - https://www.youtube.com/watch?v=qjyCq-9fJuw
- [M03] - https://www.youtube.com/watch?v=4zTHPRc-INM
- [M04] - https://www.youtube.com/watch?v=d2bHBmXXdJo
- [M05] - https://www.youtube.com/watch?v=WN0Vixbfpbs
- [M06] - https://www.youtube.com/watch?v=hziQm1a5irE
- [M07] - https://www.youtube.com/watch?v=L7OQ2bab6HY
- [M08] - https://www.youtube.com/watch?v=8epX0SaZyv0
- [M09] - https://www.youtube.com/watch?v=7Wwoz5MSaUI
- [UNSEEN] - https://www.youtube.com/watch?v=3xt3VvaH6f8

A.0.4.13 Greater London - Female

- [F00] - https://www.youtube.com/watch?v=JbCtn0vNvek
- [F01] - https://www.youtube.com/watch?v=JNyBO5Y3Upc
- [F02] - https://www.youtube.com/watch?v=dxWXl1ZABhU
- [F03] - https://www.youtube.com/watch?v=v1ZUCsmeCPE
- [F04] - https://www.youtube.com/watch?v=umlJgWK_Xxo
- [F05] - https://www.youtube.com/watch?v=pG_GCoeyed8
- [F06] - https://www.youtube.com/watch?v=j3mend1pWl8
- [F07] - https://www.youtube.com/watch?v=-bYloTHJWx8
- [F08] - https://www.youtube.com/watch?v=QAqxN_4jVPE
- [F09] - https://www.youtube.com/watch?v=97KMKN378lst=187s
- [UNSEEN] - https://www.youtube.com/watch?v=ldYoSVO5K4Q

A.0.4.14 Greater London - Male

- [M00] - https://www.youtube.com/watch?v=YJtZMxNz0KI
- [M01] - https://www.youtube.com/watch?v=amk77G1xpIEt=869s
- [M02] - https://www.youtube.com/watch?v=NLd-tcss33M
- [M03] - https://www.youtube.com/watch?v=-o9z3UB5dwMt=331s
- [M04] - https://www.youtube.com/watch?v=nkn5zLrIR-0
- [M05] - https://www.youtube.com/watch?v=ocFJLWCHH4w
- [M06] - https://www.youtube.com/watch?v=er3s6L3Dw8o
- [M07] - https://www.youtube.com/watch?v=j3r10dQETe0
- [M08] - https://www.youtube.com/watch?v=s-FwtWOYttot=5s
- [M09] - https://www.youtube.com/watch?v=MvtuPHPNLHk
- [UNSEEN] - https://www.youtube.com/watch?v=LVuDJ5aO2fs

A.0.4.15 South East - Female

- [F00] - https://www.youtube.com/watch?v=8VEipx4wV-8t=87s
- [F01] - https://www.youtube.com/watch?v=Rxt9gdjVWdM
- [F02] - https://www.youtube.com/watch?v=qgrcV26rxPU
- [F03] - https://www.youtube.com/watch?v=lmfHzx6jyco
- [F04] - https://www.youtube.com/watch?v=HTzLCdi77uw
- [F05] - https://www.youtube.com/watch?v=EIt_qW4ebZQ
- [F06] - https://www.youtube.com/watch?v=5mmB7fkJUU8
- [F07] - https://www.youtube.com/watch?v=kxqlPzTCFcU
- [F08] - https://www.youtube.com/watch?v=rY3vB8eWFNw
- [F09] - https://www.youtube.com/watch?v=IpJrPEhX2m0
- [UNSEEN] - https://www.youtube.com/watch?v=Vpjp-CPHk_s

A.0.4.16 South East - Male

- [M00] - https://www.youtube.com/watch?v=m4SoDdXBwcg
- [M01] - https://www.youtube.com/watch?v=Aa_2JI4eblQt=420s
- [M02] - https://www.youtube.com/watch?v=oGykXkakx10t=14s
- [M03] - https://www.youtube.com/watch?v=M__yZUcm2UQt=100s
- [M04] - https://www.youtube.com/watch?v=-_ij_ZyDwVI
- [M05] - https://www.youtube.com/watch?v=9FajTBDwHLA
- [M06] - https://www.youtube.com/watch?v=iUUpvrP-gzQt=74s
- [M07] - https://www.youtube.com/watch?v=oF2qYst_7II
- [M08] - https://www.youtube.com/watch?v=JqnSI60__kE
- [M09] - https://www.youtube.com/watch?v=NFYnT-b5IGst=53s
- [UNSEEN] - https://www.youtube.com/watch?v=NhH7S-_B3d8t=205s

A.0.4.17 South West - Female

- [F00] - https://www.youtube.com/watch?v=_Aax6_rga1s
- [F01] - https://www.youtube.com/watch?v=bmM7etrlKcs
- [F02] - https://www.youtube.com/watch?v=oIBRbSCN-IM
- [F03] - https://www.youtube.com/watch?v=vTo0crpK8Zst=290s
- [F04] - https://www.youtube.com/watch?v=0uGxaPrJgzg
- [F05] - https://www.youtube.com/watch?v=ZwHFazeeTvA
- [F06] - https://www.youtube.com/watch?v=pKy-XarmFPQ
- [F07] - https://www.youtube.com/watch?v=M0rfQBSQvnEt=7s
- [F08] - https://www.youtube.com/watch?v=OujQcbSrRZs
- [F09] - https://www.youtube.com/watch?v=evdCi7ZyQQs
- [UNSEEN] - https://www.youtube.com/watch?v=MAs5fDVp96k

A.0.4.18 South West - Male

- [M00] - https://www.youtube.com/watch?v=PzxuDyNC1bQt=74s
- [M01] - https://www.youtube.com/watch?v=AB4EAfuv65st=1135s
- [M02] - https://www.youtube.com/watch?v=6Vn6ecRR8LE
- [M03] - https://www.youtube.com/watch?v=n0JGT5SkpKo
- [M04] - https://www.youtube.com/watch?v=P24wyd1bNBg
- [M05] - https://www.youtube.com/watch?v=KZWQ2KpsUVs
- [M06] - https://www.youtube.com/watch?v=buG_Ne3C5Ew
- [M07] - https://www.youtube.com/watch?v=WKvk1tk2wO8
- [M08] - https://www.youtube.com/watch?v=aJuyUMmASTw
- [M09] - https://www.youtube.com/watch?v=Qz48ClAquzI
- [UNSEEN] - https://www.youtube.com/watch?v=wosinmfH4Pw