Accent Classification Using a Large Non-Homogenous Dataset

A Comparative Study of Bespoke CNN and Fine-tuned Wav2Vec2 Model

NCHAI781 Dissertation Project AE2

Isobel Bourne

Supervised by Dr. Mahsa Abazari Kia

In partial fulfilment of the requirements for the degree of

MSc Responsible Artificial Intelligence

Northeastern University London

Abstract

ASR systems report much higher error rates when used with 'accented' speech, thus, many ASR researchers and developers pursue Accent Recognition from a both an ethical and a quality assurance standpoint in order to fulfill their responsibility to their users to ensure all experiences with ASR technology are equal, regardless of accent. They hope to achieve this either by using AR to elucidate distinguishing acoustic features of accents that can be incorporated into ASR research or by directly incorporating an accent classification sub-task into an ASR model. Working on the assumption that a classifier that is more robust to non-homogeneous data will generalise better on real-life examples, this project aims to expand and test previous accent classification findings on the large non-homogeneous dataset Common Voice 13.0 - the largest labeled corpora of accented speech to date. This project will be one of few works in the AR field that performs accent classification using a large dataset. It also provides a direct comparison of the effectiveness of a bespoke CNN model with fine tuning a large pre-trained acoustic model (Wav2Vec2) on Common Voice. Overall, when evaluated on unseen data, the final models report a classification accuracy of 55% for the CNN model, surpassing previous CNN benchmarks for Common Voice, and 75% for the Wav2Vec2 model.

Keywords: Accent Classification, Convolutional Neural Network, Wav2Vec2.0, Common Voice, Mel-Spectrograms

Abbreviations

Accent Recognition	AR
Audio Speech Recognition	ASR
Speech-to-Text	STT
Large Acoustic Model	LAM
Virtual Reality	VR
Common Voice	CV
Mel-Frequency Cepstral Coefficients	MFCCs
Gaussian Mixture Models	GMM
Phone Recognition followed by Language Modelling	PRLM
Support Vector Machine	SVM
Feed-forward Neural Network	FFNN
Extreme Learning Machine	ELM
Hidden Markov Model	HMM
Long Short-Term Memory	LSTM
Convolutional Neural Network	CNN
Dragon Age: Origins	DA:O
Cross-Lingual Representation Learning For Speech Recognition	XLSR
Emphasized Channel Attention, Propagation and Aggregation in TDNN Based Speaker Verification	ECAPA-TDNN
Categorical Cross Entropy	CCE
Negative Likelihood Loss	NLL Loss

Contents

List of Figures

List of Tables

1 Introduction

1.1 History and motivations

Accent classification, also known as Accent Recognition (AR) aims identify the accent class of a speech sample via computational means. In recent times this is achieved through the use of machine learning techniques, or more generally 'AI', including neural network based models. Accent classification belongs to the domain Audio Speech Recognition (ASR) and even more broadly sits within the field of Human-Computer Interaction. As a field, AR is similar to language identification and speech identification as it takes a (usually) full-length utterance or segment of speech and identifies it as a particular class[22].

Most work in AR is motivated from the perspective of improving ASR systems and improving interactions that individuals with 'accented' speech have with this technology[13, 16, 21]. Some very common examples of ASR systems are Virtual Assistants (the most well known being Apple's Siri, Amazon's Alexa, and Google Assistant). ASR is also commonly used in business, marketing and analytics as it can be used to automatically create customer profiles or pick up keywords in speech and online videos, which are often used to suggest tailored advertisements and recommendations[14]. A major use of ASR is in speech-to-text (STT) transcribing systems[21]. These are often used as accessibility tools, for example as a way for visually impaired individuals to transcribe documents or to automatically generate closed-captions for hearing-impaired people. All of these systems increase digital accessibility and are becoming a large part of everyday life in the UK and other countries.

However, ASR struggles with speech variability, in which accents are one of the greatest sources[13]. Large scale acoustic models (LAMs) are primarily used for ASR tasks and these models are trained on vast amounts of unlabeled data via self-supervised learning[22]. However, this data is usually unbalanced favouring a 'standard' accent for the country in which is it being developed and rarely featuring the accent of non-native speakers. The result of this is an algorithmic bias towards speakers of the 'standard' accent and as performance is specifically optimised for these users, those with non-'standard' accents encounter a much higher rate of inaccurate results[21]. For example, Zuluaga-Gomez et al.[22] states that the LAM Wav2Vec2.0[5] exhibits roughly a 50% relative increase in word error rate for the Malaysian accent versus the standard US accent for text-to-speech transcription. This increase in error rates will also apply to non-standard native accents which are considered 'strong' regional accents, such as the UK Scottish accent. Hence, those with non-'standard' accents experience a technological gap in ASR technology.

Many ASR and AR researches hope to mitigate the lack of diversity in ASR technology through developing accent classification models and incorporating them into ASR systems[7, 16, 11, 13, 21, 22]. This can be referred to as "accent-robust speech recognition". For example, some technologies, e.g. Google Home, allow the user to select certain accent options, such as Irish-accented English or Indian-accent English, which use LAMs fine-tuned on data specific to that accent, to improve ASR on variant speech patterns pertaining to that particular accent. However, accent classification does have other applications, such as in the media, VR and video games industries[9]. Another primary motivator for AR research is to use machine learning alongside linguistics to reveal more about dialects such as which phonetic and acoustic features make up a particular accent[10].

1.2 Research questions

One major issue with accent classification and AR research to date is the quality of training data. Most research in surveyed literature conduct their work on limited and un-diverse datasets, with one of the reviewed models only using 27 audio samples for training[9]. Most of these datasets also use the same speakers multiple times, and when data is sparse this can lead to the model performing speaker identification rather than accent identification, and hence performs badly on unseen data where different speakers are present[21]. On top of this, with a low number of speakers in the data, the corpus is not able to represent the diverse population in terms of age or gender. Moreover, the quality of data varies significantly, and most research appears to be conducted on small, curated datasets made superficially for accent classification and thus do not generalise well on real-life unseen data where speech is less clear or in the presence of background noise[14].

Larger datasets appropriate for training accent classification models are rare and pose unique challenges[11, 14]. These datasets are often crowd-sourced and have clips that are very poor included in the corpus which can impact AR performance. Because of this, researchers tend to avoid such datasets and the limited benchmarks available are conducted on sparse homogeneous data. It is important, however, to conduct research on non-homogeneous data where speech varies from sample to sample in terms of speaker, script and quality, in order to carry over research findings to real-life use - where data encountered 'in the wild' is less standardised and controlled[14].

This project aims to apply past research methodologies that have warranted success on sparse data onto the larger, more diverse Common Voice 13.0 corpus in order to analyse the suitability of these techniques on non-homogeneous data more akin to real-world speech and provide a comparison of differing model structures on large data. Throughout this project I will evaluate the effect of using large non-homogeneous data such as Common-

Voice for accent classification. I will compare the performance and optimal hyper-parameters to similar CNN methodologies which have proven to achieve over 95% accuracy on sparse homogeneous data, to my proposed CNN model using the large non-homogeneous CommonVoice. Finally, I contrast this to the results of fine-tuning Wav2Vec2 the same CommonVoice data in order to evaluate which network architecture is more favourable.

1.3 Structure

For this project, I will design, train and optimise a bespoke (not pre-trained) CNN model for accent classification and compare this with my own fine-tuned Wav2Vec2 model using the same data from Common Voice. This will provide a comparison of how modern CNN modelling methods translate from a sparse to a large non-homogeneous dataset and also contribute a comparative review of training a CNN wholly on a large dataset against fine-tuning a pre-existing Wav2Vec2 model.

This chapter serves as an introduction to this project and the topic of accent classification as well as providing insight on the motivations of AR as a step towards creating more inclusive ASR systems. In the second chapter, titled 'Relevant Work' I supply a comprehensive review of both initial and more recent AR research. The third chapter outlines my methodology, including my motivations for utilising the Common Voice corpus and the network architecture of both the CNN model and fine-tuned Wav2Vec2 model. In chapter four, I outline my pre-processing pipeline, discuss and explain my experimental setup and document the results. The results of each experiment and the optimal model is discussed in more detail in the fifth chapter alongside the specific challenges encountered during this research, the trajectory of AR and possible future work on this project and my contributions in the AR field.

2 Relevant Work

The surveyed literature can be divided into approaches which use traditional machine learning methods and those approaches that use neural network models for classification. Table 2.1 below provides a comparison of recent related works.

Authors	Year	Dataset	Inputs	Model	Classes
Zuluaga-Gomez et al.	2023	Common Voice	features extracted via CNN	w2v2-XLSR & ECAPA-TDNN	16 (English)
Mikhailava et al.	2022	Speech Accent Archive	MFCCs and additional features	CNN	9
Graham	2021	IViE, Cambridge English Corpus	Spectrogram	CNN (LeNet)	5
Zhang, Wang and Yang	2021	AESRC dataset	40-dim F-bank spectrum	CNN (Jasper) with Attention	8
Nicastro and Inguanez	2020	Wildcat Corpus, Common Voice	13 MFCCs	Hierarchical CNN	3 (Wildcat), 4 (CV)
Najafian and Russell	2020	ABI (train and val), WSJCAM0 (test)	GMM i-vectors	Multi-class SVM	4
Bird et al.	2019	Custom dataset of vowel sounds	26 MFCCs	LSTM	4
Rizwan and Anderson	2018	TIMIT	12 MFCCs and normalised energy parameter	ELM	7
Ensslin et al.	2017	Speech Accent Archive (train and val), Dragon Age: Origins (test)	Spectrogram	CNN (AlexNet)	3
Tverdokhleb et al.	2017	Wildcat Corpus	MFCCs	FFNN	7

Table 2.1: Table of related works

2.1 Input Features

Before discussing the different approaches in model architecture, it is important to discuss the different approaches to feature extraction amongst the surveyed literature. Before a raw audio waveform can be classified it must first transformed into a certain feature set. However, researchers have found it difficult to determine which approach to feature extraction is the most effective for AR. Hence, there is a large discussion around which input feature set is the most appropriate for accent classification, with some researchers considering this factor just as significant as model selection in creating a highly accurate model[11]. Representing the appropriate features with enough detail to display subtle phonetic discrepancies between audio samples[10] is the focus of much research in this field. Frequently used feature extraction techniques include calculating Mel-frequency cepstral coefficients (MFCCs)[16, 7, 18, 14] and spectrogram images[10, 9], as well as a variety of other feature sets[13, 16, 18, 21]. Input feature sets can also be combined in a single model[11, 16, 13].

Recently, MFCCs, specifically the first 13 coefficients, have been chosen frequently as inputs for accent classifiers. Bird et al.[7] use 26 MFCCs, Rizwan and Anderson[16] use the first 12 MFCCs and Najafian and Russell[13] use 19 to construct their Gaussian Mixture Model (GMM) i-vectors. However, Nicastro and Inguanez[14] along with Mikhailava et al.[11] suggest that only the first 13 MFCCs provide a substantial contribution. MFCCs are commonly calculated using the Librosa Python library[3] or Auditory Toolbox.

Spectrograms are also a common choice of feature set such as by Ensslin et al.[9] and Graham[10]. Ensslin et al. segment an extracted spectrogram into four quadrants, representing frequency and amplitude through both logarithmic and linear scales in a combinatory approach. Ensslin et al. also breaks down the spectrogram image further into smaller length segments. Despite this Graham emphasises the merits of using 'raw' spectrogram visualisations without any explicit phonetic separation. Mikhaialava et al.[11] also highlights the effectiveness of using amplitude Mel-Spectrograms on a linear scale with a CNN model. Graham obtains the spectrograms using the Praat software. However, like MFCCs, spectrograms can also be calculated using the Librosa Python library.

2.2 Traditional Machine Learning Models

According to Nicastro and Inguanez[14] early accent classification tasks were performed by phonotactic systems which used Phone Recognition followed by Language Modelling (PRLM) to compute output predictions. These systems measured the acoustic similarity of certain accents based on specific units of speech such as phonemes. Accents were then clustered to-

gether via their similarity through statistical methods[14]. However, most recent approaches today use some form of supervised machine learning to automatically classify accents based on extracted acoustic features. Traditional machine learning models that are frequently employed as Decision Tree Classifiers and k-Nearest Neighbour algorithms[11] as well as Support Vector Machines[13], Gaussian Mixture Models[18] and Hidden Markov Models[7].

Najafian's and Russell's[13] exploration of accent recognition aims to improve transcription for accented speech by combining a neural ASR system with a SVM-based accent classifier. Najafian and Russell use a novel feature set obtained by using a Gaussian Mixture Machine (GMM) to cluster the inputs into Gaussian i-vectors. They use the ABI (Accent of the British Isles) corpus with 14 accents, 285 speakers and 57.7 hours of homogeneous scripted speech. Najafian and Russell group the accents into 4 broad classes and categorise the samples with 89.8% accuracy and 76.8% on the original 14 classes. Their study concludes that there is a negative correlation between AR and ASR accuracy with the easier to classify 'extreme' accents (those which are located on the periphery of the i-vector space) experienced a notably high error rate in transcription.

Rizwan and Anderson[16] attempt to improve speech recognition through accent classification whilst working on the TIMIT dataset. They present a method which utilises SVMs, but can be substituted with an equivalent deep learning network such as ELMs, in order to classify regional US accented speech from 630 speakers. Rizwan and Anderson propose training $\frac{n(n-1)}{2}$ SVM-based models per n classes, i.e. a model per pair of of classes. Using MFCCs and the normalised energy parameter and delta coefficients, extracted using Auditory Toolbox, each model 'votes' for a class for each sample and the overall model determines the predicted class by the number of 'votes'. Whilst using five words they achieve 60.58% accuracy with the SVM under-performing its neural equivalent. One cause for this could be that Rizwan and Anderson's SVM approach acts on the assumption that accents are acoustically independent from each other - the complexity of neural networks can allow for a more nuanced representation of the relationship between accent classes. Their neural equivalent, on the other hand, classifies accents with 77.88% accuracy using the same methodology.

2.3 Neural Network Based Models

In recent times, deep learning and neural networks are favoured for accent classification. Approaches began by using feed-forward neural networks (FFNNs)[16, 18] and then progressed onto more advanced architectures such as Long Short Term Memory (LSTMs)[7] and Convolutional Neural Networks (CNNs)[9, 10, 14, 11, 21]. Bird et al.[7], Rizwan and

Anderson[16], and Tverdokhleb et al.[18] all study the effectiveness of deep learning against traditional machine learning methodologies.

Tverdokhleb et al.[18] design a simple FFNN using extracted MFCCs from audio in the Wildcat Corpus. The Wildcat corpus is relatively sparse containing on 1342 audio recordings including both scripted and non-scripted accented speech. Tverdokhleb et al. aim to integrate AR with ASR systems in order to help language learners with more personalised pronunciation feedback and spell-checking and could be integrated into the ASR technology. Their model achieves 91.43% accuracy. In sum, throughout the surveyed literature neural and deep learning approaches increases performance as well quicker training and tuning time compared to similar traditional machine learning models.

Bird et al.[7] propose a hybrid model that combines Random Forest and a LSTM network which performs with 94.87% accuracy, improving upon the standard provided by traditional networks that only classified the data with 89.65% accuracy. Bird et al. work with a custom dataset composed of vowel sounds in both British English accented speech and Mexican English accented speech, in which they extract MFCCs for input into their model. Overall, they demonstrate the greater applicability of neural based approaches, such as LSTM in particular, in accent classification research than traditional machine learning methods.

2.3.1 Convolutional Neural Networks

However, the most popular approach in recent years is to use a CNN-based model for audio classification using visualisations of audio signals such as spectrogram or MFCC heatmap values as input into the classifier[9, 10, 14, 11]. A separate neural model can also be used to extract features from audio input such as in the case of Zuluaga-Gomez et al.[22] and Zang, Wang and Yang[21].

One of the first CNN-based surveyed approaches, Ensslin et al.[9] use an AlexNet to perform binary classification between British and US accented speech, using spectrgoram images as input. Their model is trained using audio from a mere subset of the Speech Accent Archive and then applied to manually labelled data collected from the video game *Dragon Age: Origins* (DA:O). The final model only yielded an accuracy of 52.2%, a notable low score especially considering the data's unbalanced split. They catalogue numerous reasons for this poor performance, including the fact that only 27 samples from the Speech Accent Archive was used for training the model. Moreover, since the audio samples for testing were taken from the DA:O video game the samples contain a variety of background noise, emotionally variant acoustic features and other differences between samples not related directly to accent. Furthermore, many of the accents in the test data are

not authentic and are performances by a largely American cast of voice actors - making classifying these accents, especially with a model trained on such different data, a mammoth task. Hence, the models degradation in the models performance may not reflect how the model would perform when tested on unseen data more similar to the Speech Accent Archive's samples. Unfortunately such an analysis was not provided. One conclusion that can be taken from this work, however, is that a model trained on a homogeneous dataset of clear, scripted audio struggles to generalise on more real-life variant data.

Graham[10] similarly uses a LeNet CNN trained on raw spectrogram images. Graham utilises both the IViE corpus nad the Cambridge English Corpus to create a dataset of 'Standard Southern British' accented speech alongside four non-native accents of over 60 speakers for both training and testing his model. His aims align with linguists who endeavour to harness the deep learning capabilities of Convolutional Neural Networks (CNNs) to model and analyse acoustic features of an accented audio signal in order to infer certain forensic attributes that may pertain to the speaker. Graham encourages little pre-processing of the data and uses a raw spectrogram image as input for his model, unlike other researchers who have segmented spectrograms into phonetic features. This approach serves to reduce unnecessary data processing times. The data provides a balanced distribution of speakers, gender across the five classes which each occupy the same allotment in the data - Graham does not comment on the similarity of the IViE corpus and the CEC corpus samples and makes the assumption that the concatenation of the two datasets is appropriate and that the native speakers are not identifiable due to differences in the collection of the data. Overall, he reports 83.6% accuracy and suggests that phonetic modelling should be used with the deep learning model to allow the network to identify the most significant attributes of the given input features and filter out unnecessary acoustic information for accent classification.

Zhang, Wang and Yang[21] propose a hybrid approach to accent recognition which automatically extracts feature embeddings from F-banks for input into a CNN-Transformer model. On a similar vein to previously discussed works, Zhang, Wang and Yang's motivation stems from a desire for accent-robust ASR systems. However, Zhang, Wang and Yang go one step further by incorporating an ASR front-end into their proposed AR system. The model uses 40-dim F-bank spectrums as input into two acoustic Jasper models pre-trained on non-accented data from LibriSpeech, an audio corpus for ASR. The weights of the first Jasper model remain fixed, whereas the second model is further fine-tuned on to AESR dataset of accented speech. The resulting output emeddings are fused using a Concatenation-ChannelAttention-based fusion technique that utilises Transformers to allow the model to pay more attention to transcriptions provided by the

most accurate model when evaluating the transcriptions with data from the validation set. Zhang, Wang and Yang's impressive and novel approach performs with an overall accuracy of 96.7% for accent recognition and a further 79.9% accuracy for transcriptions using the ASR front-end.

Working from a marketing perspective, Nicastro and Inguanez[14] design a Hierarchical CNN for accent classification with the eventual intention of incorporating this research into ASR in order to improve brand mention detection in accent speech. Akin to Tverdokhleb et al.[18], Nicastro and Inguanez train their model using the Wildcat corpus. They achieve 88% accuracy on unseen data from the Wildcat corpus. However, they further train and test the model using the Common Voice corpus using a range of different accent classes and achieving 43% accuracy with three accent classes for example. They further test the model on a custom dataset collated from YouTube audio clips achieving 47% classification accurate at the third level of the hierarchical classifier. Nicastro and Inguanez are primarily concerned with the evaluation of Common Voice as a dataset, providing benchmarks and tools for evaluating corpora for accent classification. They encourage testing on multiple datasets in order to ensure transfer ability onto unseen real-life data and further highlight the importance of using large datasets such as Common Voice for improved generalisation.

Mikhailava et al.[11] take a more standard approach to accent classification compared to researchers such as Zhang, Wang and Yang[21]. However, this report excels by providing a thorough investigation into input feature and hyper-parameter selection with a CNN model. Mikhailava et al's extensive research analyses the optimal non-arbitrary combination of input features and most appropriate hyper-parameters including batch size, kernel size and learning rate. They experiment and report the performance of their model with a wide range of input features such as MCFFs as well as fundamental frequency, spectral centroid, spectral decay, chromogram, zero crossing and root mean square. Mikhailava et al. provide a modern, near state-of-the-art approach that does not utilise transformer models which achieves a competitive accuracy. The highest average overall accuracy across all experiments was 98.4% when using all tested features and accuracy over 90% was common for input combinations with this model. They design a CNN model with two Convolutional layers and two dense layers with max pooling and batch normalisation applied after each layer trained and tested on the sparse Speech Accent Archive. This corpus contains scripted speech from a limited number of speakers reading identical sentences. Overall, Mikhailava et al.[11] provide benchmarks for CNN model optimisation and suggest that their proposed accent classification ought to perform equally well on both homogenous and non-homogenous data.

Finally, Zuluaga-Gomez et al.[22] attempt a novel approach comparing

the results of fine-tuning Facebook AI's Wav2Vec2.0-XLSR against a fine-tuned ECAPA-TDNN model. They use a self-defined train, validation and test split created from samples from Common Voice 7.0. This model uses a CNN front-end to extract input features from the raw audio signal and feeds this input into a transformer attention based network. Akin to previous research, their motivation is for AR to be integrated into future ASR technology to provide a more inclusive experience for users. They produce a 'recipe', CommonAccent, that can be applied multi-lingually and achieves a classification accuracy of 96% and 97% on their respective defined English validation and test sets. They demonstrate this multi-lingual transferability by successfully further fine-tuning their model on Common Voice 11.0's Spanish, German and Italian subsets. Overall, the wav2vec2-XLSR model performs better than the ECAPA-TDNN model likely due to the scale of pre-training. Finally they provide an analysis of Wav2Vec2.0's internal self-supervised operations by implementing a clustering algorithm to display the phonetic similarity detected between accents on a vector space.

3 Methodology

Throughout this project I aim to evaluate the effectiveness of using Convolutional Neural Networks (CNNs) on the Common Voice dataset for Accent Classification. Up until recently, most accent classification research has been conducted on relatively sparse datasets, this project aims to test these methods using the large, crowd-sourced Common Voice corpus. CNNs are at the forefront of accent classification research and have proved to produce highly accurate results [11]. For this project two CNN models were designed building on research conducted by Mikhailava et al. and compared to a pre-trained Wav2Vec, fine-tuned for this task [4].

3.1 Common Voice

Common Voice is a large, non-homogeneous, crowd-sourced dataset hosted by Mozilla [12]. It is intended to improve real life implementation of speech recognition software by increasing diversity of research data. It aspires to bridge the technological gap between English and non-English languages as well as sharing the voices of underrepresented groups so that ASR systems work equally for everybody. Common Voice aims to include speakers from a diverse range of backgrounds, such as age, gender, race, sexuality, and accent. At the time of writing, CV supports 112 different languages, has over 28,118 recorded hours of audio. In some cases Common Voice is the only available dataset for ASR taks in the specified language [12].

Common Voice is fully crowd-sourced and volunteer based. Contributors are prompted to record themselves reading one of the approved sentences using their own device and recording equipment. The sentences are similarly contributed by users, and then validated by at least 2 other users to enter the pool of approved sentences. Each sentence must be fewer than 15 words and in the public domain and can be mined from websites such as Wikipedia. One example sentence included in the dataset is:

"He was a nephew of Rear-Admiral Sir Francis Augustus Collier."

As we can see from this example, the sentences often contain words such as names or places that the average person may be unsure about their pronunciation. The audio sample of the vocalised sentence is also approved by users, requiring at least 2 others to validate the recording for it to become enter the main dataset. Users are urged not to reject based on the speaker 'having an accent' or making mispronunciations, as long as the sample is clear enough to be understandable and without excessive background noise.

Common Voice is considered the largest dataset for audio processing and speech recognition tasks to date [22]. The first Common Voice Corpus

was made publicly available in February 2019 and whilst it contained over 20GB of data (compared to the now 77GB), it has only recently received large-scale attention in the academic and research space. Hence, much of the contributions from previous research remain untested on Common Voice.

3.2 CNN Model

3.2.1 Network Architecture

CNNs are deep learning neural networks, originally designed for image classification tasks. They are comprised of convolutional layers which transform an input image using a matrix called a 'kernel'. In general, the kernel is significantly smaller than the image and for each channel of an input image, this kernel slides across the image for each patch, which is then multiplied by the pixel values of the patch to produce the output of the convolution. This output is generally then flattened and input into another classifying network such as a FFNN (a simple neural network composed of dense layers of connected nodes), for example in the case of my model the input is flattened and input into three dense layers. Convolutional neural networks are ideal for image classification but can be generalised beyond images to other learning tasks with varying inputs.

My CNN model was designed as a 7 layer network following from the network structure used by Mikhailava et al.[11] that produced results with over 90% classification accuracy using the sparse dataset, the Speech Accent Archive. My bespoke (non pre-trained) model comprises of two convolutional layers, each followed by a batch normalisation layer and a max pooling layer using ReLu activation. The output is then flattened an input into 2 dense layers. Following this the model includes a dropout layer with a dropout rate of 0.2, this helps to prevent overfitting by setting a 20% probability of any specific neuron become zero during the transformation. The batch normalisation layers were also added to reduce overfitting by normalising the inputs and make the CNN more stable - thus speeding up training time. The max pooling layers work to downscale the input since, with max pooling, the kernel instead takes the maximum output of each patch thereby reducing the computational complexity of the model. Figure 3.1 depicts the network architecture of the model.

Two variations of the model were built, a model using 1D Convolutional Layers and a second using 2D convolutions. Generally, 1D and 2D CNNs are used based on the dimensions of the input tensors, for example, 1D CNNs are usually used for a list of values, whereas a 2D CNN would be used for a matrix of values such as an image. In these cases, the kernel slides over the respective number of dimensions. The feature sets calculated via the Librosa python library can be inputted into a 1D CNN as a list

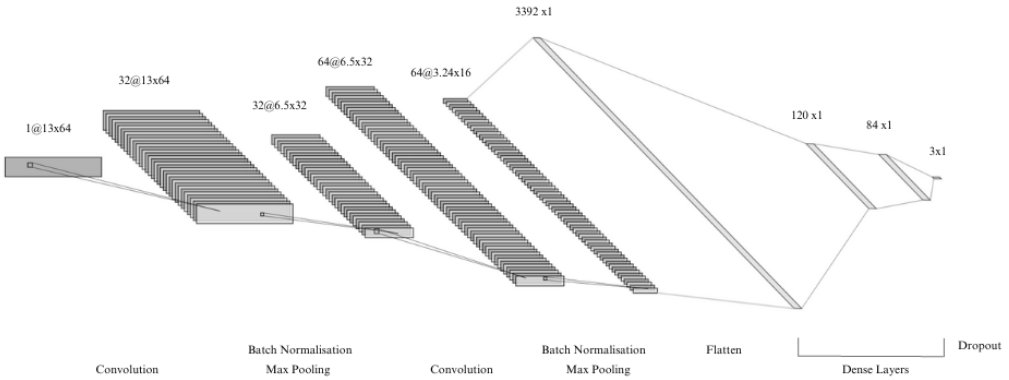

Figure 3.1: Diagram of CNN Model Architecture

or reshaped as a 2D input akin to an image. For this project I plan to compare the accuracy of 1D and 2D Convolutions with calculated input feature values such as MFCCs, and Mel-Spectrograms.

3.2.2 Feature Extraction

Both the one dimensional and two dimensional model were evaluated with a variety of different input features. When sound is recorded, the microphone converts the sound wave into an electrical signal. However in order to classify these sounds, features must first be extracted from the audio signals. These statistical methods that aim to capture the 'shape' of the audio signal and represent acoustic features. The most popular choice for input features is to use the first 13 MFCCs which has proven to yield positive results. However, Mel-Spectrograms are also popular. Usually, Mel-Spectrograms are preferably represented on the logarithmic scale. Despite this, Mikhailava et al. [11] emphasise the promise of the Mel-Spectrograms on the linear scale with a CNN model. Hence, all three input feature sets were evaluated with the two models.

MFCCs are calculated by splitting the (enhanced) signal into short overlapping frames, which allows information about how the signal changes over time to be preserved. Subsequently a windowing function is applied and the frequency spectrum and power spectrum are calculated. This latter process is called the Short-Time Fourier-Transform. Further filter banks are then applied on a Mel-scale (a non-linear scale that emphasises lower frequencies much like the human ear) to the amplitude/power spectrum. This is then represented on the logarithmic scale to give the MFCCs. MFCCs show the changes in rate of different bands in the spectrum[17].

Another common feature extraction technique is the visualise the audio signal as a Spectrogram image. Spectrograms show the spectral density of a signal[17] by showing the frequency and amplitude of an audio signal

over time and are usually depicted as a heat map. They are also calculated using a Fourier Transformation. A popular form of Spectrogram are Mel-Spectrograms, which are Spectrograms cast on the Mel-Scale instead of frequency. This type of Spectrogram is referred to in this paper as a linear/amplitude Mel-Spectrogram or simply as a Mel-Spectrogram. However, amplitude is often succeeded by the Decibel Scale (dB) to account for the way that humans hear amplitude in a logarithmic (as loudness) rather than linear way [8]. This type of Mel-Spectrogram is referred to as a logarithmic Mel-Spectrogram. Figure 3.2 visualises an audio clip in the Common Voice dataset as an amplitude Mel-Spectrogram on the linear scale.

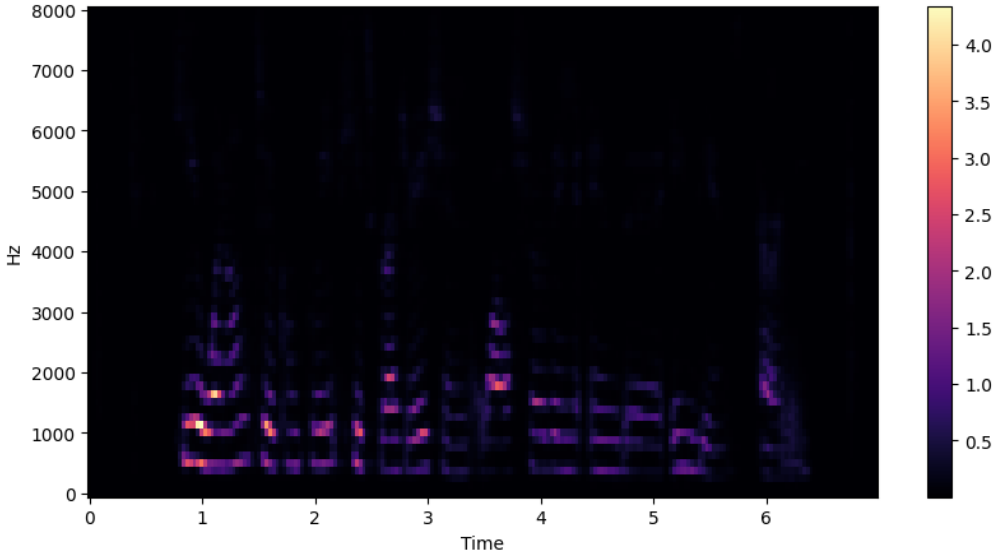

Figure 3.2: Linear Mel-Spectrogram representation of audio signal from Common Voice

3.2.3 Implementation

Pre-processing, feature extracting and model training was carried out on an Intel i7-12700H with 64GB RAM. The Common Voice dataset was accessed and downloaded via Hugging Face [2] and explored with the pandas library. The sampling rate of Common Voice is re-sampled to be 16000Hz. The pre-processing steps as well as the calculation of the input features was carried out using the Librosa Python library, since Librosa includes built-in functions which, given an audio signal, transform the signal into the specified feature. Hence, built-in functons with librosa were used to calculate the Mel-Frequency Cepstral Coefficients, Mel-Spectrograms (on a linear scale) and Mel-Spectrograms on the logarithmic scale. Both types of Mel-Spectrogram used 64 bands. The Hugging Face datasets library allowed these pre-processing transformations to be calculated in parallel. The Librosa and Matplotlib libraries were also used to generate the visualisations.

The model was built and trained using PyTorch using `CrossEntropyLoss` as its loss function. Moreover, the Adam optimiser was used to train the model. The model was evaluated according to validation accuracy at each epoch. Although the data was re-balanced to include less US accents, the Scikit-Learn library's `compute_class_weight()` function was used to further implement weight balancing during training.

3.3 Fine Tuned Wav2Vec2 Model

3.3.1 Network Architecture

Wav2Vec2[4] is a pre-trained model that was developed by Baevski et al.[5] for Facebook AI. It is a large scale model for Automatic Speech Recognition using a vast amount of unlabelled audio data. The Wav2Vec2 model is a start-of-the-art system used for ASR that, when fine-tuned, performs successfully on the dataset with only a small amount of data required for fine-tuning. Von Platen states that Wav2Vec2 produces an error rate as low as 5% on the LibriSpeech test set using only 10 mins worth of audio from the training set. The extremely promising results in ASR achieved by means of Wav2Vec2 have led to further research into the use of Wav2Vec2 in accent classification tasks, such as that by Zuluaga-Gomez[22] who successful fine-tuned Wav2Vec2 on Common Voice 7.0, as discussed in Chapter 2.

The core of the wav2vec2 model is implemented by a transformer. Transformers consist of stacks of encoder and decoder components. Each encoder and decoder consist of a self-attention later and a feed-forward layer, with the decoders also having an extra attention layer. The self-attention layers enables the encoders/decoders to look at other positions while encoding/decoding each discrete unit, allowing for the use of 'context clues' in the outputs[19]. The Wav2Vec2 model uses its attention mechanism to identify and acquire 'context representations' of speech. First the inputs are inputted into a CNN and then the feature vectors are 'quantized' into discrete speech units and inputted into a transformer encoder[6]. Figure 3.3 shows the network architecture of the Wav2Vec2 model. In order to use this model for an (accent) classification task, the resulting context representations are then pooled and parsed into a linear layer, with the outputs representing each accent class.

My fine-tuned Wav2Vec2 model uses input features extracted from Common Voice 13.0 via Wav2Vec2's feature extractor, with audio normalisation enable. The feature extractor is pre-trained and does not need to be further fine-tuned as part of the main model. Similarly to the CNN model, the Wav2Vec2 model uses 3 second audio clips of US, South Asian and English accents from Common Voice that have been trimmed and balanced. The same samples from Common Voice from the validation and test set used by the CNN model were used for hyperparamter optimisation and testing,

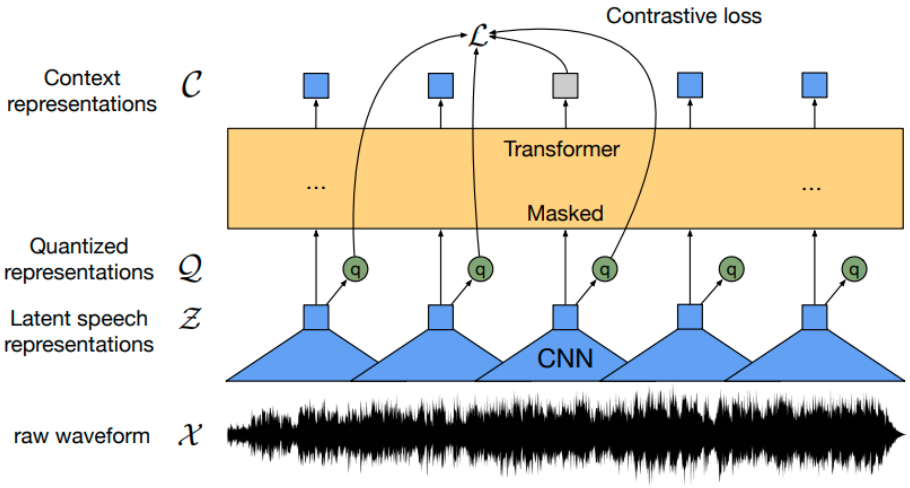

Figure 3.3: Wav2Vec2 Network Architechure (figure designed by Patrick von Platen[20])

respectively. Like with the previous CNN model, the validation and test sets were not balanced.

3.3.2 Implementation

Code provided by Hugging Face[1] was used to implement the Wav2Vec 2.0 model for accent classification. The use of PyTorch and the Hugging Face Transformers and Accelerate libraries allowed for easy integration with a GPU which sped up training by two orders of magnitude. The pre-processing and feature extraction was carried out on an Intel i7-12700H with 64GB RAM. The data was pre-processed according to the same pipeline described in Section 3.1.2. A CNN-based feature extractor model was then used to process the audio waveform using a sampling rate of 16kHz. The sampling rate of Common Voice is 48kHz so the data needs to be downsampled in order to match the 16kHz sampling rate of the Wav2Vec2 was pre-trained on.

The model training was carried out on an RTX A4500 GPU with 30GB RAM, rented via `runpod.io`. A Google Cloud Storage bucket was used to transfer data between the two stages due to the ephemeral nature of the rented machine. The model was fine-tuned with Common Voice data using using a gradient accumulation size of 4 to enable larger batch sizes with a lower memory footprint, as well as a warmup ratio of 0.1 to negate the primacy effect. The model was evaluated based on the accuracy of the predictions of the validation set at each epoch using CCE loss.

4 Experiments and Results

4.1 Data Pre-processing

This project utilises Common Voice Corpus 13.0 in the English language. This contains 2,429 hours of validated, English-language speech, involving 86,942 speakers. When contributing their voice, users are given the option to record their own accent using natural language rather than selecting from a list. Whilst this allows the user more choice in how they refer to their accent, it means that the data requires more pre-processing for the accents to be grouped accordingly.

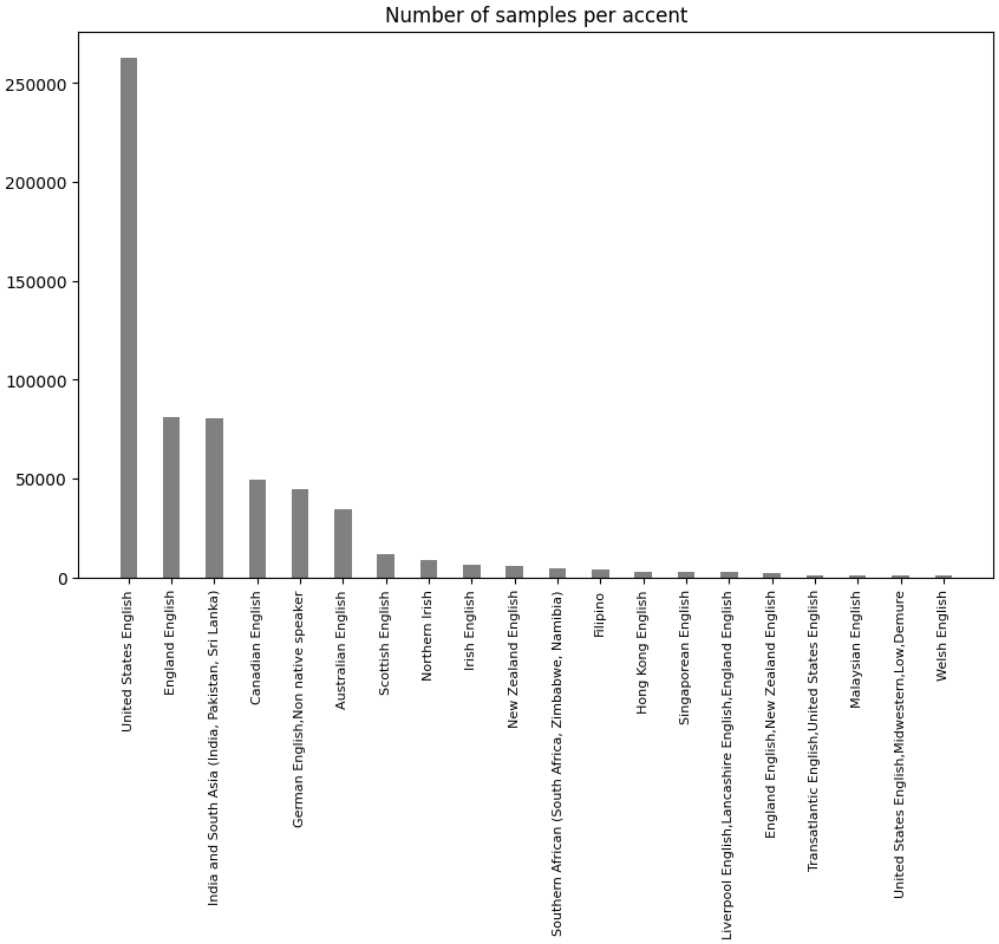

Figure 4.1: The number of samples per top 20 accent labels.

Initially, the dataset contains 618,196 unique accent values. The top three accents are 'United States English', 'England English' and 'India and South Asia (India, Pakistan, Sri Lanka)'. As can be seen in Figure 4.1, after the first six labeled accent groups the number of the samples per label drops significantly. Furthermore, this does contain many duplicate accents

or accents that could be categorised as multiple accents. For example, the accent 'Liverpool English,Lancashire English,England English' could be included in the broader category of 'England English' and added to that accent label during pre-processing to create broader groups. However, due to there being a suitable amount of data in the top three categories of the unedited dataset I have opted to only utilise data with those values, filtering out all other accent classes.

After this, silence at the beginning and end of the clips were removed using the trim function included with the Librosa Python library[3]. Some researches opt to remove other moments of silence from the audio clips. However, Mikhailava et al.[11] encourages moments of silence not to be excluded due to correlating with accent classes and demonstrates a boost in classification accuracy when including these pauses. Hence, only silence at the beginning and end of the clip were removed. Audio clips in the Common Voice corpus are of varying length, hence, all clips were then cropped to be a uniform length of three seconds, or filtered if not meeting this length. Instead of segmenting the data into multiple clips of the same length, as is performed by some previous researchers to boost the number of samples in the dataset, the remaining audio post three seconds was removed. This reduction in data was performed to control the size of the dataset which is initially very large and hence reduce the length of training time (due to certain computational constraints this was considered an important factor) whilst still maintaining substantially more samples than previously used datasets in related works.

However, the dataset required further filtering. Of the three accents included - United States, England, and South Asia - the majority are labeled 'United States English', resulting in an unbalanced corpus. In order to balance the data, a further filter was applied to reduce the number of US accents by one third, aligning more so with the number of English and Indian and South Asian accents included in the data.

Moreover, Gaussian noise was added to the audio files which adds random noise values to the input values according to normal (Gaussian) distribution. Per research conducted by Nicastro and Inguanez[14] in which data augmentation made the model more robust by reducing over-fitting and showed improvement in the classification accuracy of Common Voice samples using a CNN model. This pre-processing step was not applied to the Wav2Vec model.

Common Voice Corpus 13.0 contains three metadata files: `train.tsv`, `validation.tsv` and `test.tsv`. The `train.tsv` split of the data was used to train the model and `validation.tsv` was used for hyper-parameter optimisation. Both `train.tsv` were pre-processed before use. Finally, `test.tsv` was used to test how the final models generalise onto unseen data.

After filtering, the training, validation and test sets contained 142672, 1362 and 1113 audio clips respectively.

4.2 Experiments

4.2.1 Feature set

Prior to training the model, the Mel-Frequency Cepstral Coefficients, Mel-Spectrograms (on a linear scale) and Mel-Spectrograms on the logarithmic scale were calculated. Experiments were carried out to evaluate the suitability of each input feature set individually with the proposed 1D and 2D CNN models. Using an initial batch size of 64 and a learning rate of 10^{-2}, each feature set was tested with both models for 10 epochs.

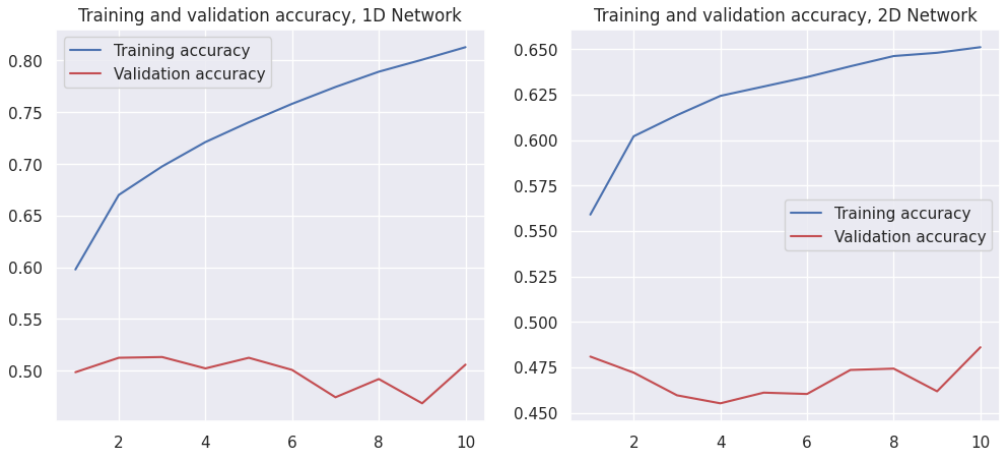

Figure 4.2: Training and validation accuracy using MFCCs.

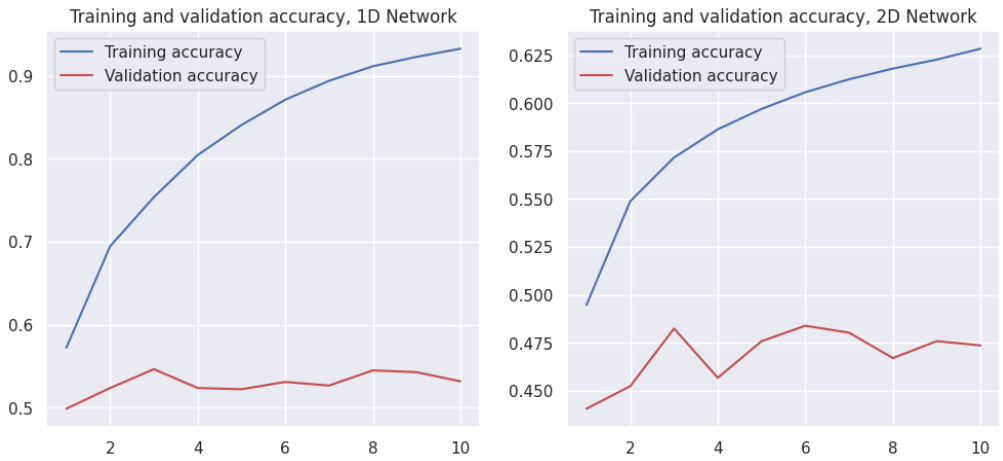

Figure 4.3: Training and validation accuracy using (linear) amplitude Mel-Spectrograms.

Figure 4.4: Training and val accuracy using logarithmic Mel-Spectrograms.

Input	CNN Model	Train Loss	Train Accuracy	Val Loss	Val Accuracy	Best Epoch
MFCCs	1D	0.011729	65.1%	0.018193	48.6%	10
MFCCs	2D	0.010518	69.75%	0.016651	51.32%	3
Linear Mel-Spec	1D	0.012583	60.57%	0.017547	48.38%	6
Linear Mel-Spec	2D	0.008651	75.37%	0.019383	54.62%	3
Log Mel-Spec	1D	0.011192	66.66%	0.016491	51.9%	7
Log Mel-Spec	2D	0.006501	82.13%	0.022114	54.33%	4

Table 4.1: Feature and Model experimentation results

Table 4.1 shows the best epoch based on validation accuracy for each input feature set-model combination. It gives the value for both train and validation accuracy at the best scoring out of the 10 training epochs, as well as the training and validation loss. Here the loss values are scaled by the number of samples in the respective datasets to allow for comparing the training and validation values. Figures 4.2 to 4.4 show the training history over all 10 epochs. The 2D CNN model using the linear amplitude Mel-Spectrograms as input yielded the most accurate results at this stage, in line with the research carried out by Mikhailava et al.[11]. The results did not show huge contrasts in validation accuracy, however, this model in particular could be at risk of overfitting due to the discrepancy in its training accuracy and its validation accuracy compared to the other models. The validation accuracy was closely followed by the 2D model using Mel-Spectrograms on the logarithmic scale. However, as can be seen in Figure 4.5 the Logarithmic Mel-Spectrogram based 2D model showed a dip in

learning after the 4^{th} epoch, whereas the linear Mel-Spectrogram produced more even results across all epochs, despite the increase in training accuracy - suggesting that this model may overfit less than the latter. Using MFCCs led to worse results with this CNN model than Mel-Spectrograms. Notably, with the MFCC-based models the training accuracy does not increase so sharply, showing a tendency to overfit less, but the validation accuracy appears to have plateaued from just the first epoch.

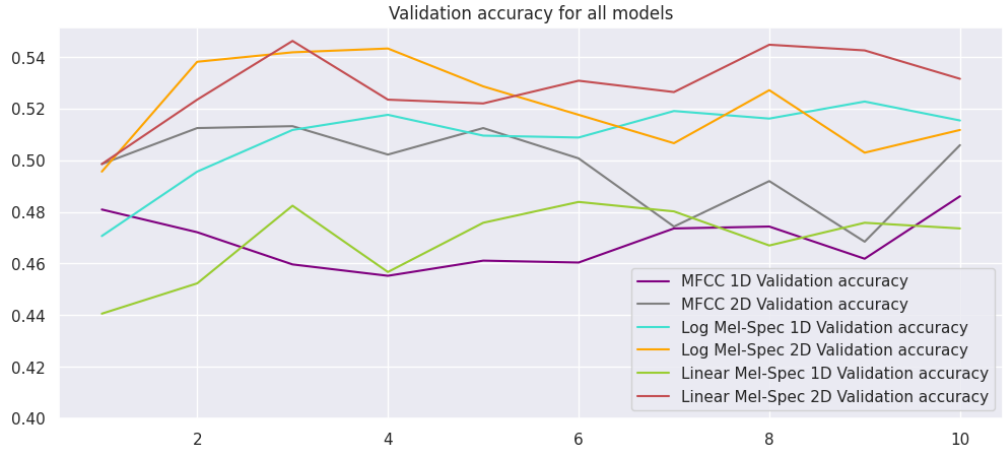

Figure 4.5: The validation accuracy for each input feature for 1D and 2D CNN model.

4.2.2 Batch Size, Learning rate and number of epochs

Hence, further analysis was carried out using only the linear Mel-Spectrogram-based 2D CNN model. Further experiments were implemented to determine the appropriate batch size, number of epochs and learning rate of this model. Batch sizes of 128, 64, 32 an 16 were tested using the initial learning rate of 10^{-2}. Furthermore, models using batches of size 32 and 64 were further trained on smaller learning rates of 10^{-3} and 10^{-4}. Training was carried out for 20 epochs. Figure 4.6 shows the validation accuracy of each model across the 20 epochs.

Using a batch size of 32 showed the least range in validation accuracy across different learning rates, generally performing better than the other batch sizes for each learning rate. The lowest performing variant was the model with a batch size of 64 with a learning rate of 10^{-4}. Due to its small learning rate this model initially evaluated very poorly and learned slowly, gradually increasing in performance throughout. Whilst the model did eventually perform closer to the other compared models it still remained the least accurate combination. Indeed, the lower learning rates performed worse on the validation set than the higher learning rates. The validation set appeared to favour models with a small batch size and comparatively high learning rate. In general, the smaller the batch size the less time

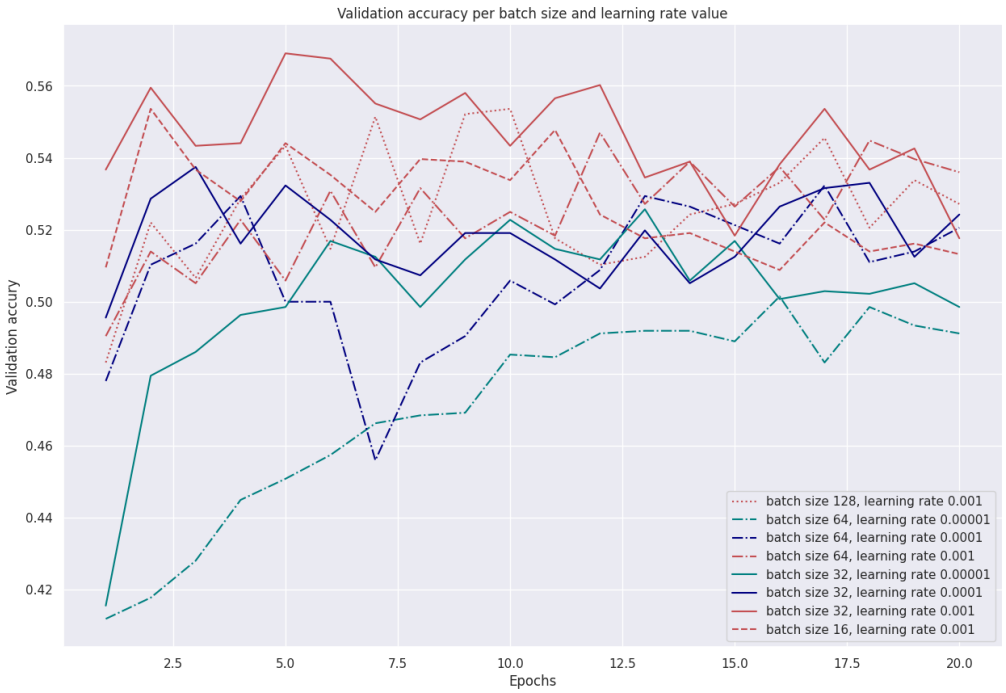

Figure 4.6: Validation accuracy per CNN batch size and learning rate adjustment.

the model took to converge. For example, the $(64, 10^{-2})$ model yielded its highest result after 12 epochs, whereas the $(32, 10^{-2})$ reached its high point after 5 epochs in which after it started to decline and the $(16, 10^{-2})$ model began to decline after only two epochs. The overall optimal hyper-parameters were to use a batch size of 32 with a learning rate of 10^{-2} as this model consistently outperformed the other variants in validation accuracy across almost all epochs.

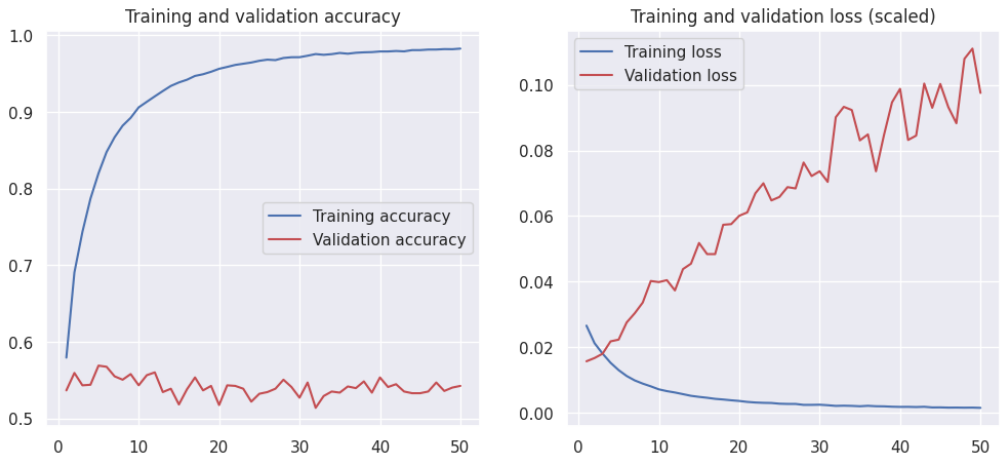

Figure 4.7: Validation and loss for 50 epochs of final CNN model.

The final model was trained for a further 30 epochs to establish the optimal point to cease learning. The validation accuracy across all epochs remained reasonably steady throughout whilst training accuracy improved. The model remained at its highest accuracy of 56.9% on the validation set after only 5 epochs, after which its accuracy gradually began to decline. As training accuracy increased substantially around the 9^{th} epoch, the validation accuracy does begin to drop. Yet there is no steep reduction in validation accuracy, suggesting that the over-fitting of the model does not have a hugely detrimental effect on its ability to generalise to unseen data. However, increased training beyond the 5^{th} or 6^{th} epoch does not improve the model. Hence, I suggest that training ought to be ceased between the 5^{th} and 9^{th} epoch to preserve accuracy on an unseen test set.

Overall, the optimised 2D CNN model using amplitude Mel-Spectrograms on a linear scale achieved 57% classification accuracy on the validation set at the 5^{th} epoch.

4.2.3 Fine-tuned Wav2Vec2

To provide a better comparison of the research findings, a second model was trained by fine-tuning Wav2Vec2 for AR on Common Voice. During the fine-tuning experimentation of this second model, the Wav2Vec2 model was only fine-tuned for 4 epochs. This was due to the length of training time associated with training further epochs, that Wav2Vec2 is claimed require little training[20] to achieve optimal results and the fact that initial testing showed learning to flatten out very early on. Batch sizes of 8 and 16 were tested with the model with 4 gradient accumulation steps, equating to effective batch sizes of 32 and 64 comparatively with the CNN model. Learning rates 10^{-4}, 10^{-5} and 10^{-6} were tested with each batch size also. The model was initially trained with a batch size of 32 and a learning rate of 10^{-4}, however, due to the large number of samples in the Common Voice corpus, it seemed worthwhile to investigate a larger batch size and this could also show an increase in training speed. The models also seemed to stop learning after very few epochs so smaller learning rates were also investigated.

Table 4.2 shows the training loss (scaled), validation loss (scaled) and validation accuracy at each model's best epoch and Figure 4.8 plots the validation accuracy across all 4 epochs for each model against each other.

It appears that larger learning rates are more suitable to the data (similarly to the previous CNN model) since, by the 4^{th} epoch the least effective learning rate was 10^{-6} for both batch sizes. The learning rate of 10^{-5} performs reasonably well with both batch sizes but shows no real increase in validation accuracy after the first epoch. Using a batch size of 32 with the same learning rate also showed the potential for promising results - despite

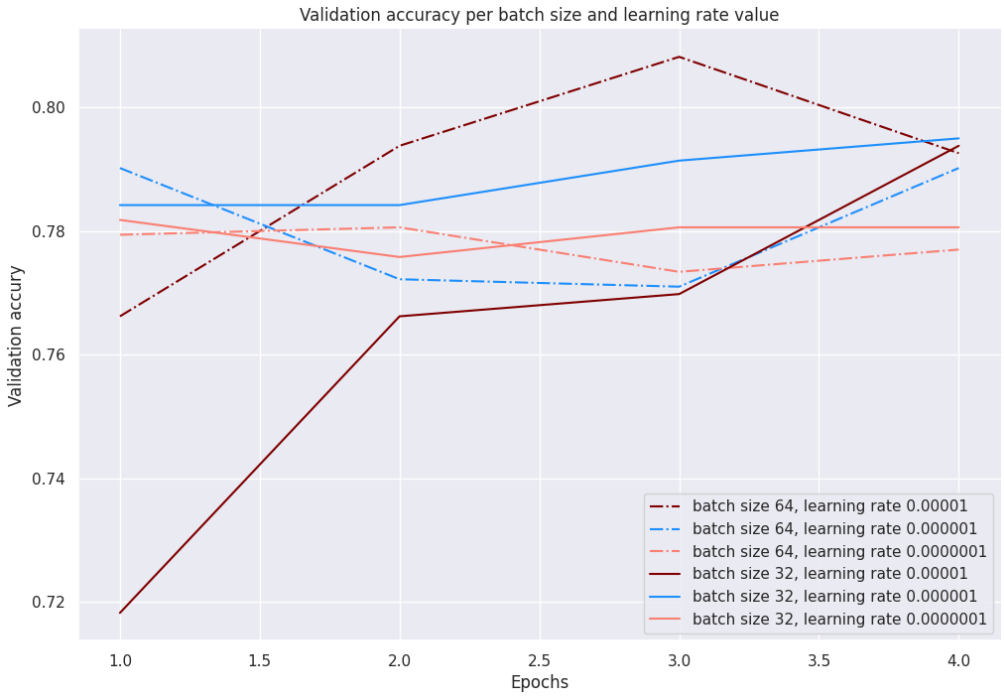

Figure 4.8: Validation accuracy per CNN batch size and learning rate adjustment.

this model's low starting it achieves high accuracy after 4 epochs with perhaps the potential to increase with more training epochs. However, using a batch size of 64 with a learning rate of 10^{-4} achieved the highest accuracy on the validation set, peaking at 80.81% accuracy and then declining after the 3rd epoch. Hence, this was chosen as the optimal model.

Batch Size	Learning Rate	Train Loss	Val Loss	Val Accuracy	Best Epoch
64	10^{-6}	0.225300	0.570457	78.05%	2
64	10^{-5}	0.142900	0.637826	79.01%	1
64	10^{-4}	0.165400	0.730128	80.81%	3
32	10^{-6}	0.193400	0.590846	78.17%	1
32	10^{-5}	0.073000	1.001559	79.49%	4
32	10^{-4}	0.124200	0.872529	79.61%	4

Table 4.2: Results of batch size and learning rate experimentation on fine-tuned model.

4.3 Other Benchmarks

Throughout AR research there is a notable lack of uniformity in approaches[14], with researchers using a wide range of models with different combinations of input features and on numerous different datasets. Historically, this has made it difficult to for appropriate benchmarks to be established and specific model comparisons challenging due to the vast differences in each approach. Mikhailava et al.'s[11] CNN model served as the basis for my

model design. As a 7-layer CNN using amplitude Mel-Spectrograms on a linear scale, Mikhailava et al. produce results of 98.6% accuracy using data from the sparse Speech Accent Archive. This project tests their ideas on the Common Voice dataset, which varies greatly from the Speech Accent Archive as it is a large, non-homogeneous dataset. Common Voice has not yet been widely used as a dataset for accent classification[14, 22], and thus such accuracy is not paralleled in the surveyed literature.

Of the surveyed literature there are few baselines provided for the Common Voice corpus. One baseline on Common Voice is established by Nicastro and Inguanez[14] who use a CNN model with MFCCs. Whilst using the train, val and test split defined by Mozilla in the Common Voice corpus (the same split used as a starting point for this project) Nicastro and Inguanez's CNN model achieves 43% accuracy for three-class accent classification.

Zuluaga-Gomez et al.[22] also work in Common Voice in recent AR research conducted earlier this year. They show that fine-tuning an existing model, Wav2Vec 2.0, over training a CNN model from scratch, is more fruitful when classifying data from Common Voice. They achieved a classification accuracy of 96.5% on their validation set 97.1% on their test set, setting the state-of-the-art for Common Voice.

4.4 Results

Both the optimised CNN model and the optimised fine-tuned Wav2Vec2 model were subsequently tested on unseen data from Common Voice: `test.tsv`. The test set was pre-processed using the same processing pipeline as the validation set described in Section 3. Neither the validation or test sets were filtered and hence contain a third more US accents than the other accent classes.

During hyper-parameter optimisation, the results for the CNN model suggested ceasing training after around 5 epochs as this is when the validation accuracy was at its peak. I suggested training could perhaps be extended to the 9^{th} epoch as this is where the validation accuracy began to decline more sharply. Figure 4.9 shows the training, validation and test accuracy and loss at each epoch from 1 to 25 (top) as well as for just the suggested first 5 epochs (bottom). The test accuracy and loss after 5 epochs is 55.43% and 22.93% respectively.

It should be noted however, as an aside, that despite suggesting training could be continued between 5^{th} and 9^{th} epoch, the test data shows a significant dip after the 5^{th} epoch. The test and validation accuracy are most closely aligned between the 1^{st} and 5^{th} epochs and the 15^{th} and 20^{th} epochs - although the latter, is less accurate and has a much higher loss than the former. Thus, this implies that the model would generalise well

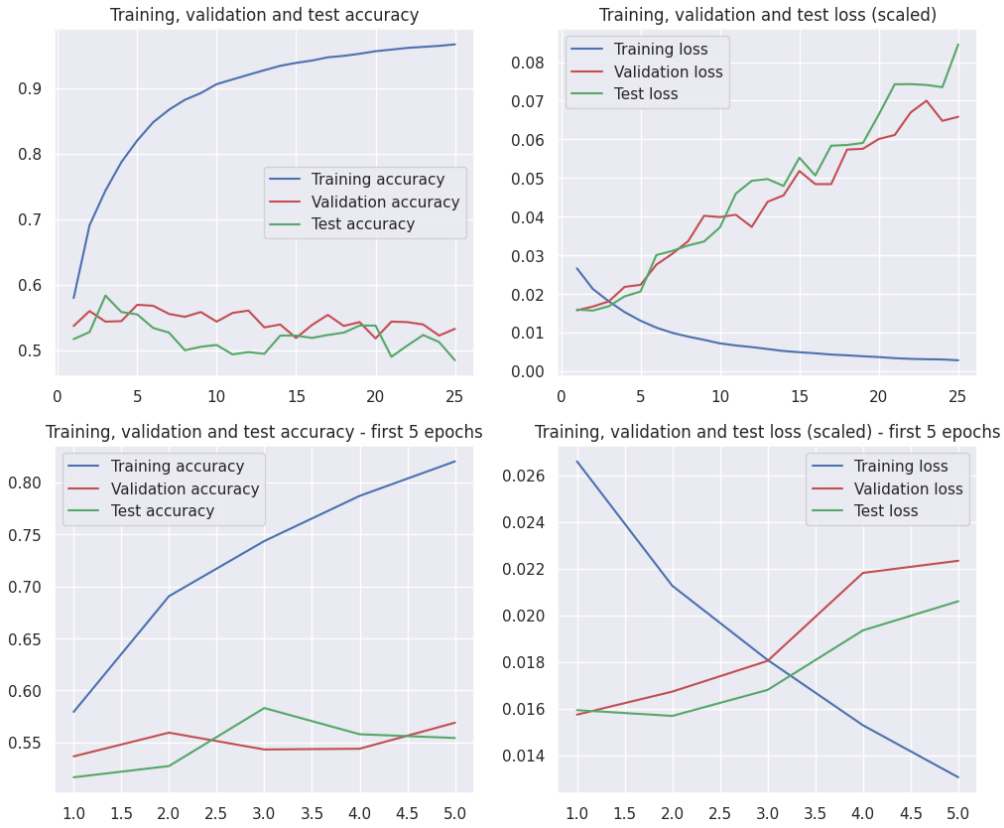

Figure 4.9: Comparison of test to training and validation accuracy and loss for final CNN model.

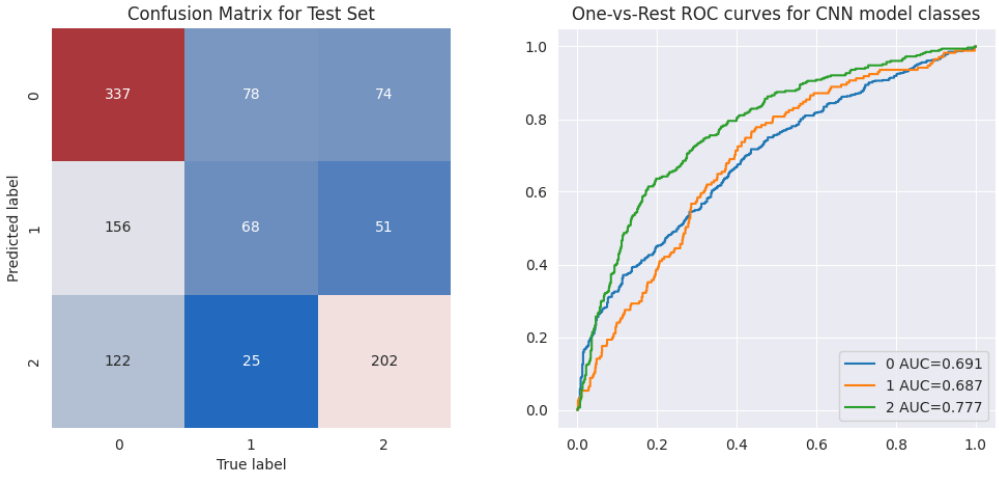

Figure 4.10: Confusion matrix and ROC for CNN model.

to a further unseen Common Voice set (beyond the test set) when trained between these first five epochs. This solidifies the idea that, to perform

most accurately on unseen data, the model should be trained for 5 epochs.

The final model has overall generalised well to the unseen test data by ceasing training after the 5^{th} epoch. Figure 4.10 shows the confusion matrix and ROC curve for the resulting model where 0 corresponds to US accents, 1 to English accents and 2 to South Asian accents. US accents were classified correctly the most, however this is unsurprising considering the skew of the test set. Overall, the model is particularly successful at classifying South Asian accents, perhaps due to this class being more acoustically unique.

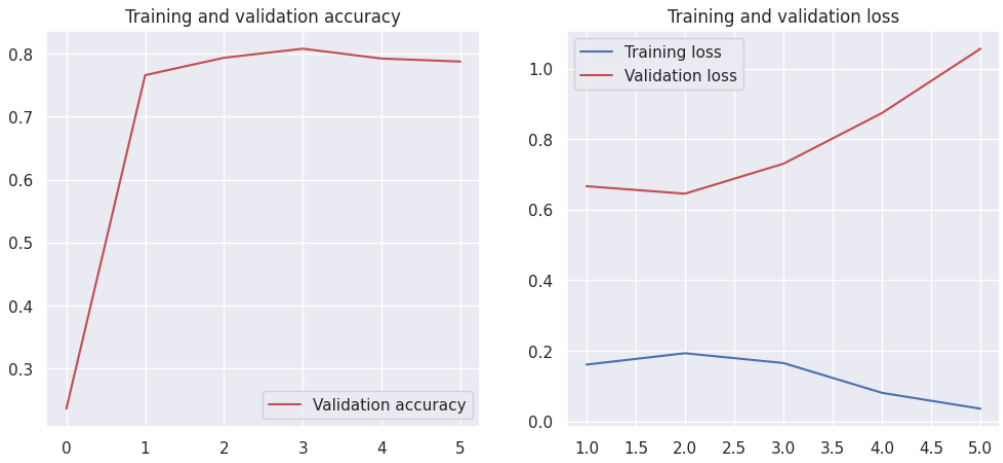

Figure 4.11: Accuracy and loss for fine-tuned Wav2Vec2 model

Figure 4.11 shows the accuracy and loss for the fine-tuned Wav2Vec2 model. The left shows the validation accuracy across the training epochs including the 0^{th} epoch which shows the initial validation accuracy of the model with no fine-tuning sits at 23%. The right shows the validation and training loss throughout training and does not include the 0^{th} epoch. The epoch with the best loss and accuracy is epoch 2 and 3 respectively. One can see that after only one training epoch using a cumulative batch size of 64 with a learning rate of 10^{-4}, the validation accuracy increases drastically. There is a slight increase in learning between the 2^{nd} and 3^{rd} epochs before the loss begins to grow. This suggests the optimal number of epochs for this model is to be between 2 and 3 epochs. Hence, the final fine-tuned model is trained for 3 epochs before being tested on unseen data, in which it generalised to the unseen test data with 75.59% percent accuracy.

5 Conclusion

5.1 Conclusion

The CNN model was more effective when using extracted linear amplitude Mel-Spectrograms (64 bands) as input than with MFCCs or logarithmic Mel-Spectrograms (64 bands). Hyper-parameter optimisation also showed that the optimal batch size was 32 with a learning rate of 10^{-2} (using the Adam optimizer and CrossEntropyLoss) after batch sizes of 128, 64 and 16 and learning rates of 10^{-2}, 10^{-3} and 10^{-4} were compared. After examining validation accuracy across 25 epochs, the resulting model was trained for 5 epochs and achieved an accuracy of 57% on the validation set and 55% on the test set.

However, fine-tuning the pre-trained Wav2Vec2 model developed by Facebook AI[5] produces superior results on Common Voice. The model was tested with batch sizes of 32 (a true batch size of 8 with 4 gradient accumulation steps) and 64 (a true batch size of 16 with 4 gradient accumulation steps), with learning rates of 10^{-4}, 10^{-5} and 10^{-6}. The optimal models demands a batch size of 32, learning rate of 10^{-4} and that learning should cease after 3 epochs. Whilst the Wav2Vec2-based approach is certainly more computationally expensive than training a CNN model from scatch, my fine-tuned model managed to produce an accuracy of 80.1% on the validation set and 75.6% on the test set with only minor differences in pre-processing between the two models. The similar model that also fine-tunes Wav2Vec2 on Common Voice developed by Zuluaga-Gomez et al. achieves a further accuracy of 97% on an unseen subset of Common Voice.

5.2 Contribution

In sum, this project has presented a novel approach to the Common Voice dataset. Using a CNN comprising of 2 Convolutional layers with both Max Pooling and Batch Normalisation and 2 additional layers, I have implemented modern research findings on the large under-researched dataset. Mikhailava et al.[11] encouraged the use of amplitude Mel-Spectrograms on a linear scale, a hypothesis that I tested and proved correct in my case as well. Moreover, both 1D and 2D models were evaluated with each feature set, with the overall model being most effective using Linear Mel-Spectrograms with a 2D CNN. Finally, numerous batch sizes and learning rate (using Adam optimiser) were investigated. The model favoured a larger learning rate and used the optimal batch size of 32. Overall the model achieved a validation accuracy of 57% and a test accuracy of 55% - over 10% above the existing CNN baseline[14].

On top of this, experiments were conducted on a further model. This

further model extracts quantized representations of raw audio through a CNN-based feature extractor model, which are in turn masked and parsed into the Wav2Vec2 transformer-based architecture. This identifies context representations to inform classifications. Models of this type have achieved state-of-the-art results in AR when fine-tuned on Common Voice[22]. Both the Wav2Vec2 based model and the CNN based model were deliberately kept as similar as possible in order to establish an easier comparison between the two network structures. The drop in performance between the CNN and Wav2Vec2 is likely due to the sheer amount of unlabelled data that the Wav2Vec2 fed to the model prior to fine-tuning which allows the Wav2Vec2 model to generalise to a wide-variety of tasks from the get-go. Furthermore, this will assist in accent classification due to the lack of commonality of sparse datasets in this field, since Wav2Vec2 claims to require little data for effective fine-tuning. Finally the increased complexity of the Wav2Vec2 model and transformer based attention mechanism likely further boosts the accuracy compared to the CNN model. These differing factors could be further explored in future works. Overall, this experimental setup has led to the conclusion that future research should favour fine-tuning large acoustic models like Wav2Vec2 on large non-homogeneous data such as Common Voice 11.0.

5.3 Discussion and future work

Using Mikhailava et al.'s[11] design as a basis, my bespoke CNN model achieves 57% accuracy on the validation set and 55% accuracy on the test set from Common Voice, surpassing Nicastro and Inguanez's[14] model who set the CNN baseline for Common Voice at 43%. However, this has not reached the success of models trained and tested on other, perhaps more-classification friendly, datasets such as The Speech Accent Archive. This is likely due to the vast differences in the two datasets. Common Voice is an extremely diverse and entirely crowd-sourced dataset intended to be used to improve real life implementation of ASR software and hence does not deliver an easy setup. As the data is entirely crowd-sources many of the audio samples are poor quality, incorrectly labeled or void of meaningful content entirely [14]. Moreover, each accent label likely contains more diverse samples in terms of accent, as the classes allow for vast regional variations in accent, as well as gender, age and level or social background. The vast amount of data utilised, however, did not mitigate these issues as despite the lack of training homogeneous sparse datasets still appear easier to classify - however, it is unclear how well these models would generalise when put to real-world use. Models trained on Common Voice can be assumed to be more robust due to the data being more reflective of real-world encounters and research conducted on Common Voice can be expanded to include the full range of accents included in the Common Voice corpora.

Furthermore, the size of the data led to numerous challenges during model development and optimisation. Due to the expansive amount of training data for each class, even after filtering and cropping, the size of the training set was still substantial - resulting in lengthy training times. Initially, implementation of the CNN model was performed on a standard CPU and only 12.7 GB of RAM. Training and data pre-processing times were extremely lengthy so this setup was upgraded to use a Tesla T4 GPU with 16GB of memory which rapidly reduced training times. For the Wav2Vec2 based model the time complexity of the model was even more severe, with training times using the CPU predicted as over 140 hours. The T4 GPU reduced the training time to around 20 hours, however, due to the feature extraction component of the Wav2Vec2 model the model exceeded the allotted RAM usage. Finally, an RTX A4500 GPU with 30GB of RAM was rented, reducing training time to roughly 1 hour per epoch with data pre-processing implemented separately.

The fine-tuned model achieves a accuracy of 80% on the validation set and 76% on the test set. Whilst this is a result competitive with other models surveyed in Chapter 2 (which again were carried out their work on cleaner, more homogeneous data) this does not surpass Zuluaga-Gomez's et. al's[22] accuracy of 97%. Zuluaga-Gomez et al.'s model differs to mine in a number of ways. Firstly, Zuluaga-Gomez et al. perform more extensive data pre-processing and using fewer samples overall whilst defining their own train, val, test split. Clips were also lengthier, usually maintaining their original length rather than being cropped to 3 seconds. Futhermore, SpecAugment, a data augmentation strategy[15], improved the Wav2Vec2's model accuracy by 4%. The network architechure of their model also slightly differed as they use w2v2-XLSR with NLL loss for multi-lingual compatibility and add a StatPooling layer at the end of the model. Future work could seek to implement further experiments based around these differences. Moreover, due to the success of the fine-tuned LAM Wav2Vec2, it may not be necessary to use such a vast amount of data for fine-tuning the model. Wav2Vec2 claims to require only 10 mins of audio for fine-tuning to produce competitive results[5]. Hence, the size of the training corpus could be reduced to determine how this effects the models performance. Indeed, using less data would have benefits in reducing training time.

Further optimisation of the Wav2Vec2 model could have increased the model's accuracy. However, due to the lengthy training times and computationally taxing nature of the model only limited experimentation with learning rate and batch size was observed. However, using a similar pre-processing pipeline for both the CNN and fine-tuned Wav2Vec2 model conveniently allows a direct comparison between the two model architectures to be made. It could also be worthwhile for further research to be carried out surrounding how the data inbalance contributes to the incorrect re-

sults on accented speech and comparative work ought to be carried out on an unbalanced dataset to discuss techniques to improve results for lesser represented classes despite having unbalanced data - and without being a detriment to applicability to the majority accent. This will help researchers mitigate issues when data cannot be accessed or augmented or when training on real-life data where exposure for certain accents is limited.

References

[1] Audio classification. https://huggingface.co/docs/transformers/tasks/audio_classification, August 2023.

[2] Hugging face datasets. https://huggingface.co/datasets/mozilla-foundation/common_voice_13_0, August 2023.

[3] Librosa 0.10.1. https://librosa.org/doc/main/index.html, August 2023.

[4] Wav2vec2-base. https://huggingface.co/facebook/wav2vec2-base, September 2023.

[5] A. Baevski, H. Zhou, A. Mohamed, and M. Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Facebook AI*, 2020.

[6] J. Bgn. An illustrated tour of wav2vec 2.0. https://jonathanbgn.com/2021/09/30/illustrated-wav2vec-2.html, September 2021.

[7] J. J. Bird, E. Wanner, A. Ekárt, and D. R. Faria. Accent classification in human speech biometrics for native and non-native english speakers. *PETRA '19: Proceedings of the 12th ACM International Conference on PErvasive Technologies Related to Assistive Environments*, 2019.

[8] K. Doshi. Audio deep learning made simple - why mel spectrograms perform better. https://ketanhdoshi.github.io/Audio-Mel/, February 2021.

[9] A. Ensslin, T. Goorimoorthee, S. Carleton, V. Bulitko, and S. P. Hernandez. Deep learning for speech accent detection in videogames. *Proceedings from the AIIDE-17 Workshop on Experimental AI in Games*, 2017.

[10] C. Graham. L1 identification from l2 speech using neural spectrogram analysis. *Proceedings of the Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2021.

[11] V. Mikhailava, M. Lesnichaia, N. Bogach, I. Lezhenin, J. Blake, and E. Pyshkin. Language accent detection with cnn using sparse data from a crowd-sourced speech archive. *Mathematics*, 2022.

[12] Mozilla. Common voice. https://commonvoice.mozilla.org/en, August 2023.

[13] M. Najafian and M. Russell. Automatic accent identification as an analytical tool for accent robust automatic speech recognition. *Speech Communication*, 2020.

[14] D. Nicastro and F. Inguanez. Multi-tier accent classification for improved transcribing. *Proceedings from the IEEE 10th International Conference on Consumer Electronics (ICCE)*, 2020.

[15] D. S. Park. Specaugment: A new data augmentation method for automatic speech recognition. `https://blog.research.google/2019/04/specaugment-new-data-augmentation.html`, April 2019.

[16] M. Rizwan and D. V. Anderson. A weighted accent classification using multiple words. *Neurocomputing*, 2018.

[17] T. Singh. Mfccs made easy. `https://medium.com/@tanveer9812/mfccs-made-easy-7ef383006040`, June 2019.

[18] E. Tverdokhleb, H. D. N. Keberle, and N. Myronova. Implementation of accent recognition methods subsystem for elearning systems. *Proceedings of the 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications*, 2017.

[19] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *Proceedings of the 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications*, 2017.

[20] P. von Platen. Fine-tune wav2vec2 for english asr with hugging face transformers. `https://huggingface.co/blog/fine-tune-wav2vec2-english`, March 2021.

[21] Z. Zhang, Y. Wang, and J. Yang. Accent recognition with hybrid phonetic features. *Sensors*, 2021.

[22] J. Zuluaga-Gomez, S. Ahmed, D. Visockas, and C. Subakan. Commonaccent: Exploring large acoustic pretrained models for accent classification based on common voice. *Proceedings of the Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2023.