Accent Classification Using a Large Non-Homogenous Dataset

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

Presentation Overview

- My motivations and the relation between accent classification and automatic speech recognition (ASR)
- High level introduction to accent classification
- Themes and limitations of previous work
- Research questions
- Overview of my methodology
- Benchmarks from some related works
- My Convolutional Neural Network
- My fine-tuned Wav2Vec2 model
- Experiments
- Results and comparison
- Contribution to the field
- Discussion and avenues for future work

My motivations

ASR technology is a growing field that we use a lot in everyday life. It's useful for convenience as well as essential for accessibility purposes.

However, ASR struggles with speech variability. One major cause of speech variability is accents. Accent robust ASR is a stepping stone to more inclusive ASR technology.

As part of our ethical obligation to produce Responsible AI, research in ASR should focus on giving all users an equal experience regardless of accent.

Accent classification is a sub-task that forms a basis for accent robust ASR. The focus is on improving user experience. However, it also has other uses such as in marketing, linguistics or language learning.

When surveying recent literature, it is clear that the topic of accent classification comes with its own challenges.

Tech > Phones & Gadgets

AYE RIGHT Scottish accents under THREAT as Amazon Alexa, Google Home and Apple's Siri fail to understand different tones

Have you changed your voice to communicate better with a virtual helper? You're not alone

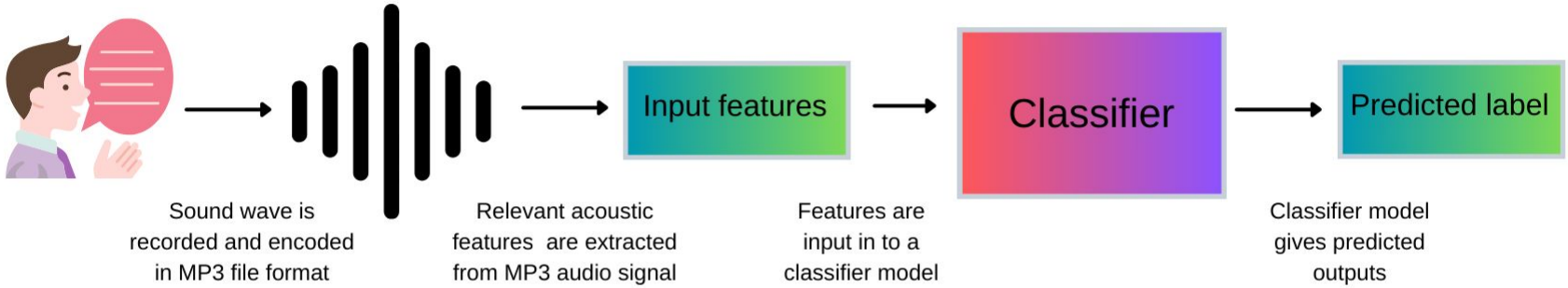
By Sean Keach, Digital Technology and Science Editor
Published: 12:45, 16 Aug 2018 | **Updated:** 12:50, 16 Aug 2018

Background: How does accent classification generally work?

Accent classification, also called Accent Recognition (AR), is a classification task that uses supervised machine learning to determine the accent label of an encoded audio signal.

An audio signal is an electronic representation of sound and can be stored in MP3, Wav and other file formats. In order to be computationally analysed by the accent classifier, acoustic features must be extracted from the signal and input into the machine learning model. The classifier then returns the predicted label class given the input which can be compared to the actual class to determine model accuracy.

The below diagram shows the machine learning pipeline for accent classification on a very high level.



* The diagram refers to the MP3 file format as this is the file format used by the relevant dataset

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

Themes of Previous work

The quality of the dataset appears to have a significant impact on classification accuracy and most models are optimised to perform well given sparse, un-diverse and highly curated data.

For example, the Speech Accent Archive. This dataset is a popular and one of the larger datasets used in accent classification and includes 655 native and non-native speakers (although mainly native) all reading the same sentence using the same recording equipment. It was constructed by linguistic researchers at George Mason University, USA for human use and not specifically for machine learning applications.

Other researchers have historically used similar datasets that do not closely resemble real life data.

Other notable issues include the lack of benchmarks, and the bias towards English language research.

Research Questions

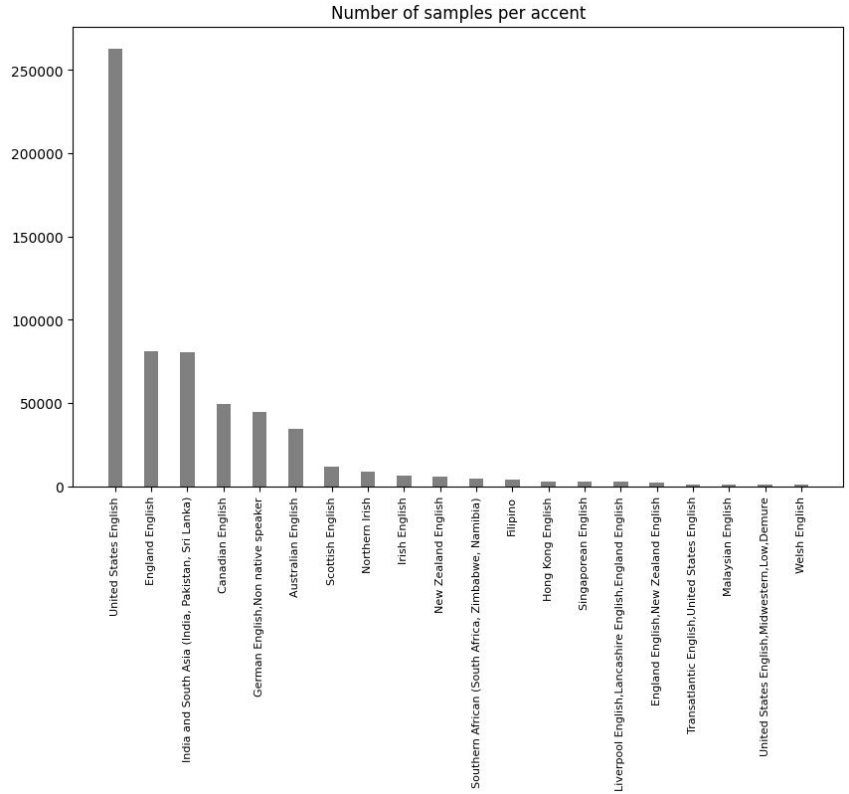
Due to these issues, most most models are optimised to produce accurate results with sparse, homogenous data. It remains unknown how these findings translate to larger, non-homogenous datasets that more resemble real-life speech.

Within the accent classification field, research on larger, less homogenous data is only now beginning to emerge (Nicastro and Inguanez, Zuluaga-Gomez et. al.) but the differences between these approaches due to the high number of variables involved in accent classification (feature extraction technique, model architecture, model parameters) it it difficult to make comparisons between these approaches and their predecessors.

In this project I examine:

a. How previous research on sparse homogenous data translates to larger non-homogenous datasets.
b. The effect of different model designs and feature extraction techniques on classification accuracy.

Methodology



My project performs accent classification using the large, crowd-sourced dataset Common Voice 13.0, developed by the Mozilla Foundation.

This dataset is the largest labeled corpus of accents to date with over 28,118 hours of recorded audio. The corpus contains a wide range of speakers from diverse backgrounds reading a diverse range of sentences - all recorded on their own recording equipment. Hence, CV resembles real-life data much more closely than previously used dataset in this field.

Using as similar data preprocessing pipeline as possible I employ two models, a bespoke CNN model and a fine-tuned Wav2Vec2 model and compare the similarities and differences between the two models and their performances.

Pre-processing involved fixing the sampling rate, balancing and filtering the accents, trimming and cropping the audio (using librosa), encoding the labels, and in the CNN model adding gaussian noise.

I optimise each model for the CV dataset and report my findings in a way that is more easily comparable to the research conclusions and benchmarks set by the related works. These works will be discussed in the following slide.

Benchmarks from related works

Nicastro and Inguanez (2020)

Nicastro and Inguanez supply the first attempt at accent classification using Common Voice, the Wildcat corpus and a custom YouTube dataset.

Using MFCCs and a hierarchical CNN model they provide the first benchmark at 43% when classifying 3 accents in Common Voice. This model uses the pre-defined train, val and test splits included in CV. They further report accuracies using augmented data and classifying further classes.

They compare the Common Voice based model to another model with the same network architecture trained and tested on the Wildcat Corpus and test their models on audio samples taken from YouTube videos - concluding that Common Voice is more useful when the model is applied ‘in the wild’.

Mikhailava et al. (2022)

Mikhailava et al. produce a near state-of-the-art CNN model that uses the sparse Speech Accent Archive.

Designs a CNN model that takes amplitude Mel-Spectrograms on a linear scale as input. They explore further input features and combinations and provide one of the only approaches that performs data-informed input feature selection. They further evaluate kernel size, batch size and other hyperparameters.

Whilst this model does not yield results as successful as Zhang, Wang and Yang (99% versus 98%) this approach excels at providing a thorough analysis of model and hyperparameter optimisation and the appropriate benchmarks. Making this methodology very suitable for use in future comparative studies.

Zuluaga-Gomez et al. (2023)

Zuluaga-Gomez et al.’s very recent work includes a comparison between two fine-tuned models for accent classification.

They fine-tuned both the pre-trained Wav2Vec2-XLSR model and ECAPA-TDNN model on Common Voice data. They conclude that the Wav2Vec2-XLSR is more accurate for accent classification with 97% accuracy.

Their approach takes advantage of the language variants of Common Voice by further applying their methodology to Spanish, German and Italian as well as the English subset of Common Voice. Since Wav2Vec2-XLSR is trained on a multilingual corpus of unlabeled data the model can be fine-tuned in many languages - making the resulting model much more transferable than bespoke examples.

CNN model

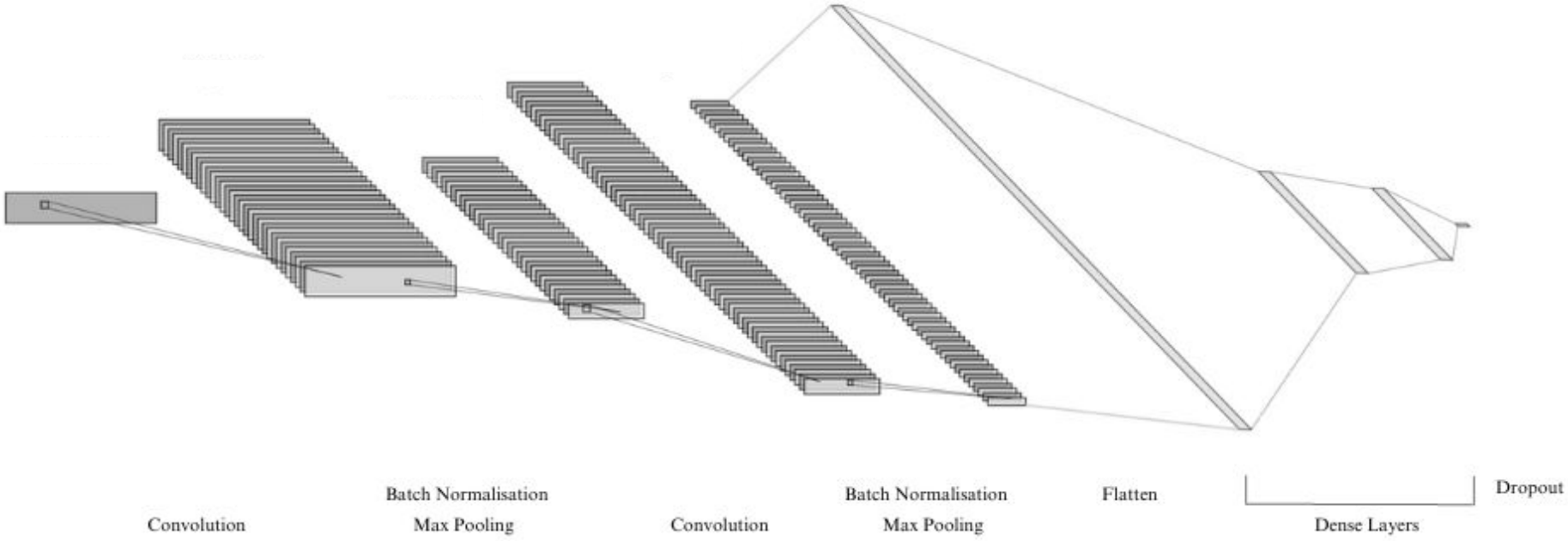
My CNN model is composed of two Convolutional layers proceeded by batch normalisation and max pooling in both cases. The output from these layers is then flattened (width, height and depth are multiplied) and input into three dense layers (with ReLu activation function) to give the output. The model includes a dropout of 0.2 chance of any neuron turning to zero before the final layer to reduce overfitting.

Both a Conv1D and Conv2D variant were designed. The code for the 2D variant can be seen on the right. This involved added a channel dimension to the inputs.

The size of the kernel was (3,3) and the size of the pooling layer were (2,2) in line with previous research. Max pooling similarly aimed to reduce overfitting. Batch normalisation aimed to regularise the inputs and reduce covariant shift where the features of the model can become exponentially small or large with each layer.

The model was initialised with the adam optimiser and a starting batch size of 64, a learning rate of 0.001 and categorical cross entropy (CCE) loss.

```
[ ]  # Neural Network 2D

     def build_2d_net(column: str):
         return Net2D(column).to(dev)

     class Net2D(nn.Module):
         def __init__(self, column: str):
             super().__init__()
             if column =='mfcc':
                 fc1_dim = 3392 # Use for mfccs
             else:
                 fc1_dim = 47488 # Use for mspecs
             self.conv1 = nn.Conv2d(1, 32, (3, 3))
             self.bn1 = nn.BatchNorm2d(32)
             self.pool1 = nn.MaxPool2d(2, 2)
             self.conv2 = nn.Conv2d(32, 64, (3, 3))
             self.bn2 = nn.BatchNorm2d(64)
             self.pool2 = nn.MaxPool2d(2, 2)
             self.fc1 = nn.Linear(fc1_dim, 120)
             self.fc2 = nn.Linear(120, 84)
             self.fc3 = nn.Linear(84, 3)
             self.drop = nn.Dropout(0.2)

         def forward(self, x):
             # Add channel dimension
             x = x.unsqueeze(1)
             x = self.bn1(F.relu(self.conv1(x)))
             x = self.pool1(x)
             x = self.bn2(F.relu(self.conv2(x)))
             x = self.pool2(x)
             x = torch.flatten(x, 1) # flatten all dimensions except batch
             x = F.relu(self.fc1(x))
             x = F.relu(self.fc2(x))
             x = self.drop(x)
             x = self.fc3(x)
             return x
```



Network Architecture of CNN model

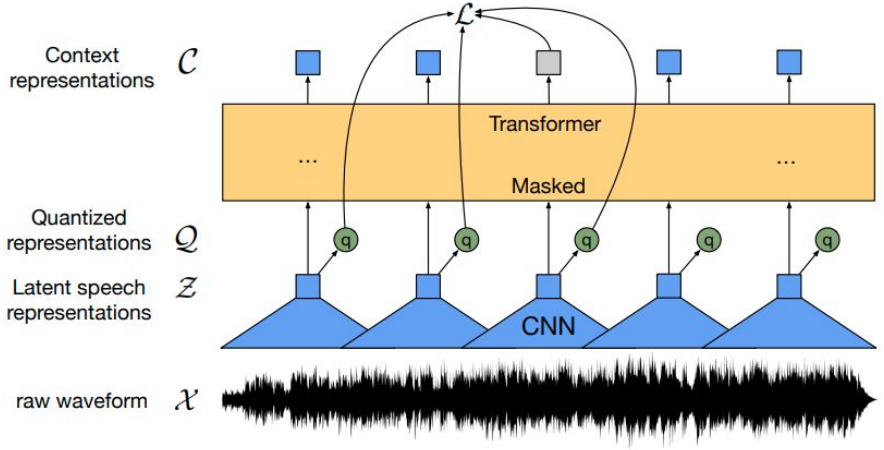
Fine-tuned Wav2Vec2

Wav2Vec2 is a pre-trained model developed by Facebook AI (Baevski et al.). My method uses Wav2Vec2 base model trained on the English language implemented via Hugging Face. It utilises a vast amount of mainly unlabeled audio.

The model includes utilises a CNN for feature extraction in which the data is quantized into discrete units of speech. The feature representations are then masked and input into a transformer encoder. The transformer model consists of stacked encoder and decoder components each with a self-attention and a feed-forward layer. The attention mechanism allows the model to use contextual information when making predictions. The loss is calculated for each quantized unit.

For classification the resulting context representations are pooled and parsed into a linear output layer.

The images on the right show a diagram of the model architecture and a code snippet of the feature extraction process. Training was carried out on a RTX A4500 GPU. The model was implemented with a gradient accumulation size of 4, a warmup ratio of 0.1 and used CCE loss.

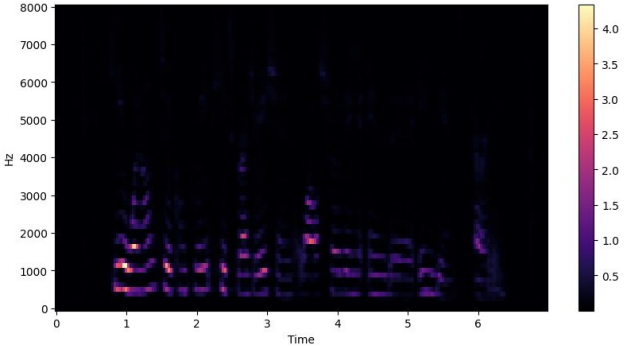


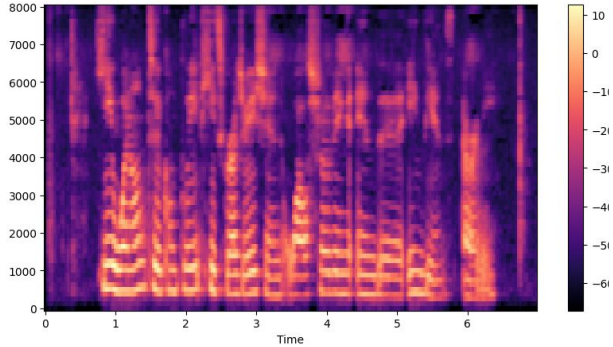

```
[ ]  # Feature Extraction
     feature_extractor = AutoFeatureExtractor.from_pretrained("facebook/wav2vec2-base")
     assert feature_extractor.sampling_rate == SAMPLING_RATE

     def preprocess_function(examples):
         audio_arrays = [x["array"] for x in examples["audio"]]
         inputs = feature_extractor(
             audio_arrays,
             sampling_rate=SAMPLING_RATE,
             max_length=SECONDS*SAMPLING_RATE,
             truncation=True
         )
         return inputs
```

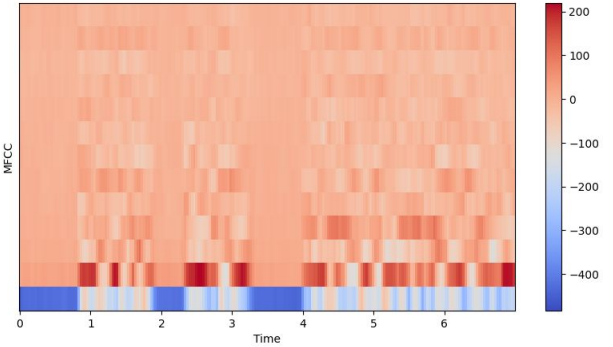
Experimental setup

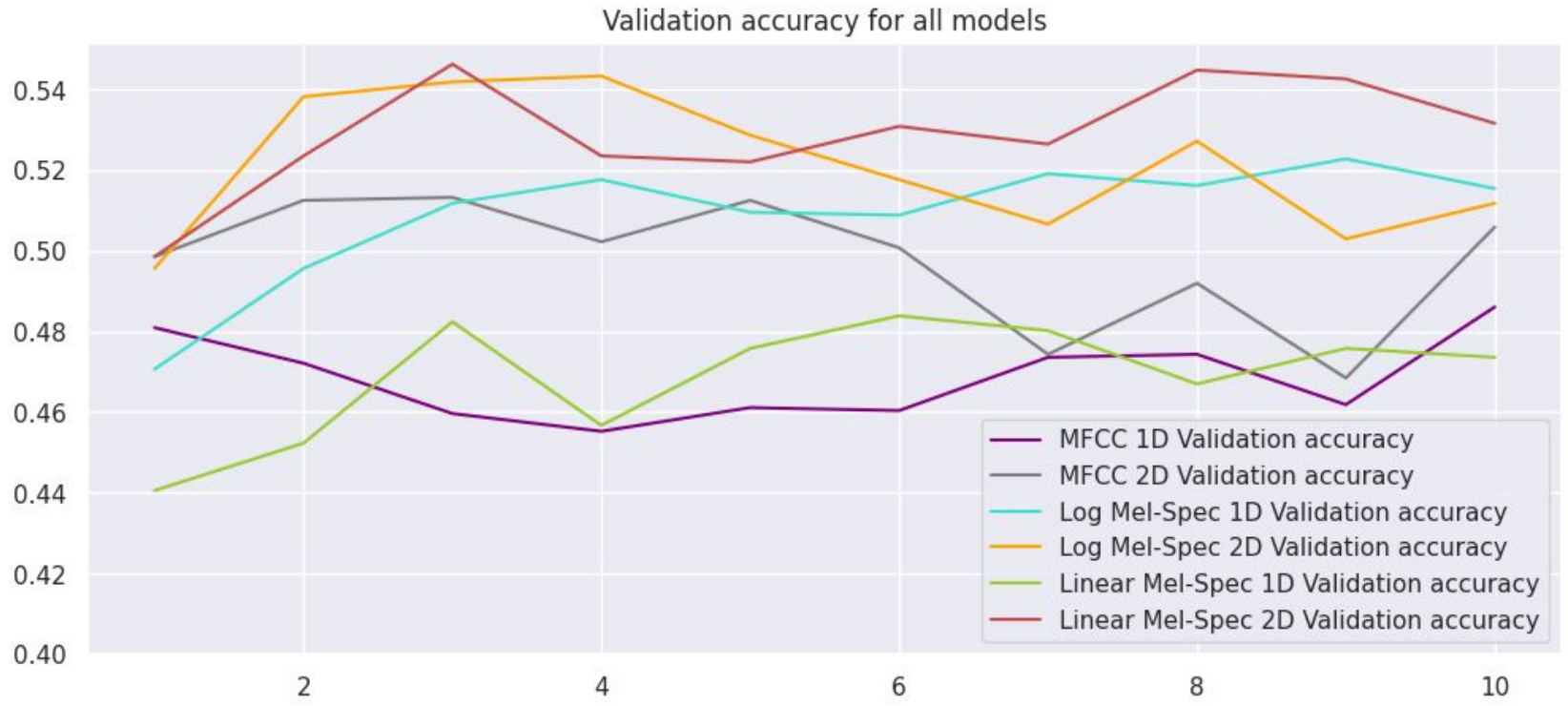
During my research I analysed the effectiveness of multiple different input features with the 1D and 2D CNN models, as well as the optimal learning rate, batch size and number of epochs for both models.

For the first experiment, each set of acoustic features was extracted using built-in functions in librosa, a python library for audio processing. The 1D and 2D variants of the CNN model were trained on each input feature set individually and the results were compared. The input feature sets analysed were the first 13 MFCCs (one of the most popular choices for accent classification and similar tasks such as speaker identification), amplitude Mel-Spectrograms on a linear scale (as suggested by Mikhailava et. al) and Mel-Spectrograms on a logarithmic scale. All the feature sets display the signal on the Mel-Scale a non-linear scale more akin to how it is heard by the human ear. The diagrams below show visual representations of each feature set displayed using librosa.

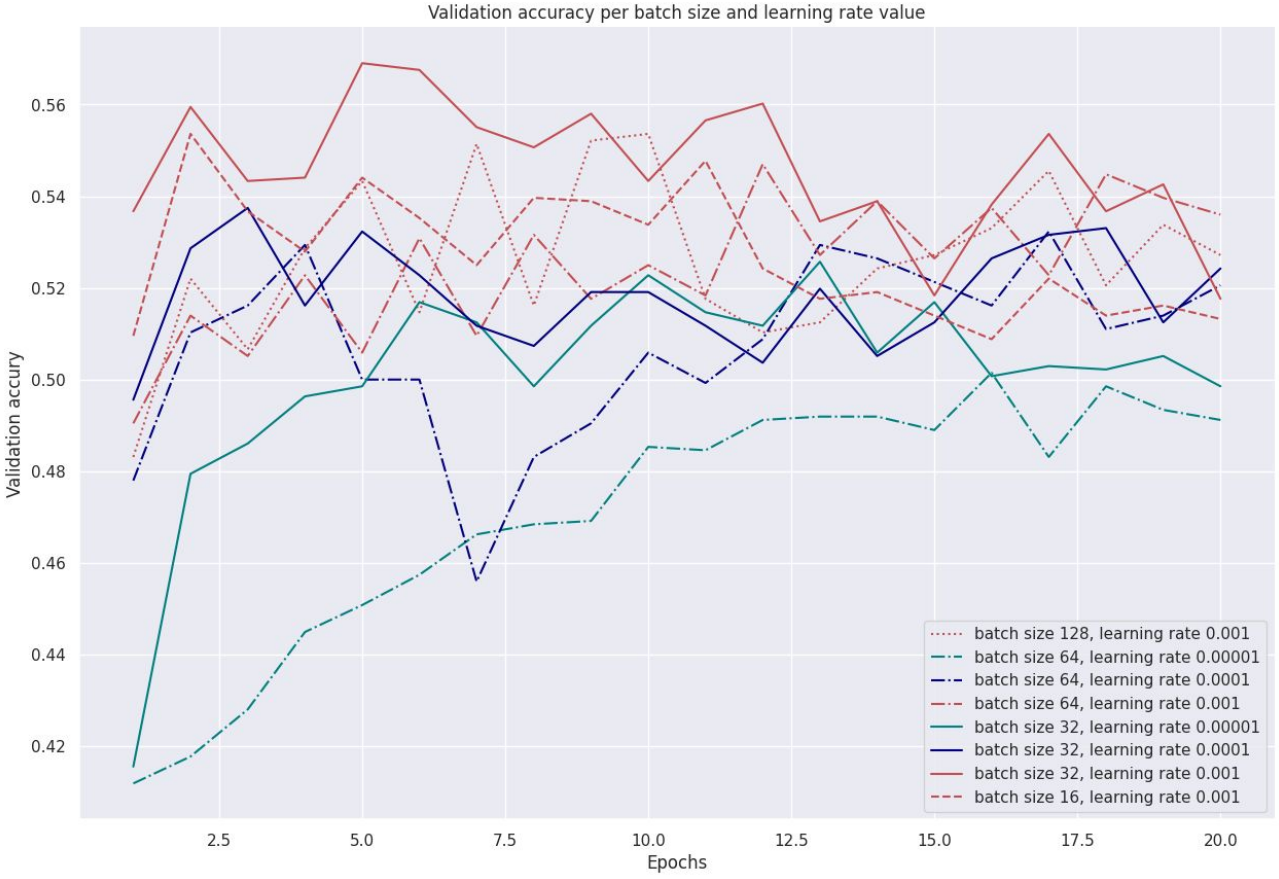








Each model was trained for 10 epochs although most stop learning much before that. The most accurate combination was the 2D CNN using linear Mel-Spectrograms as input. The 2D model has a consistently higher accuracy in both test and train across all inputs. The 2D CNN also performed well using log Mel-Specs but the accuracy varied more and deteriorated early when examining the val accuracy. At its peak the log Mel-Spec reported 82% train accuracy against 54% val accuracy. Linear Mel-Spec on the other hand reported only 75% train accuracy with a similar 55% val accuracy - suggesting that the log Mel-Spec model was more prone to overfitting.



The second experiment for the CNN model involved determining the optimal batch size and learning rate for the 2D CNN using linear mel-spectrograms.

Batch sizes of 128, 64, 32 and 16 were considered with learning rates of 0.001. The 64 and 32 batch sizes were further examined with learning rates of 0.0001, and 0.0001 for 20 epochs.

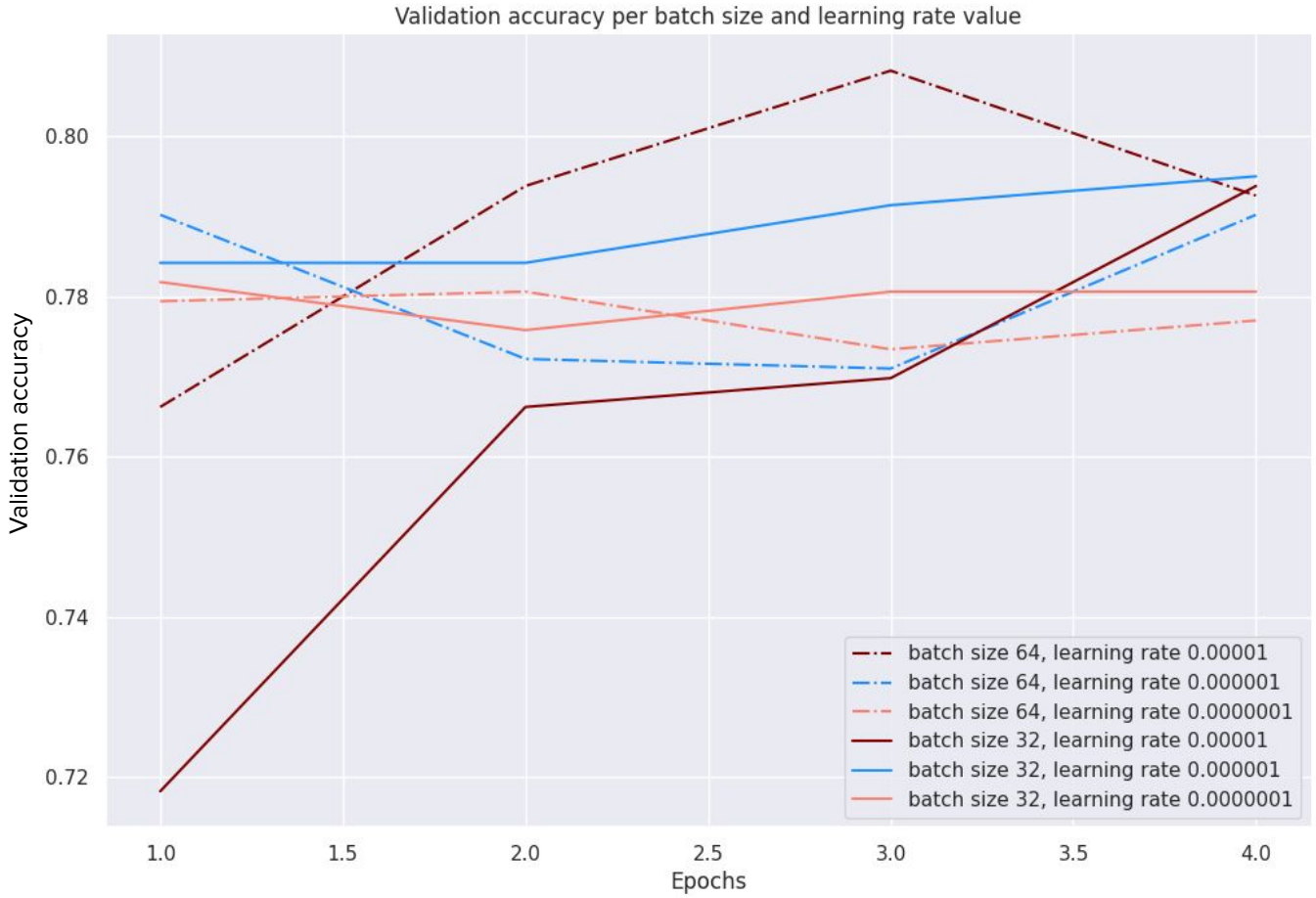
The optimal combination was to use a batch size of 32 with a learning rate of 0.001 for 5 epochs. Overall, larger learning rates performed better.

Similarly, Mikailava et al. optimised their model using a learning rate coefficient of 0.001. They trained their model for many more epochs (average 46).

Comparatively, batch size and learning rate were also analysed for the **Wav2Vec2 model**. Batch sizes of 64* and 32 were investigated with learning rates of 0.00001, 0.000001, and 0.0000001 - for 4 epochs. This was due to lengthier training times than previous model and that the pre-trained nature of the model meant less learning was required.

The optimal combination was to use a batch size of 64 with a learning rate of 0.00001 and train for 3 epochs. Although learning was limited after the initial increase from 23% at the 0th epoch.

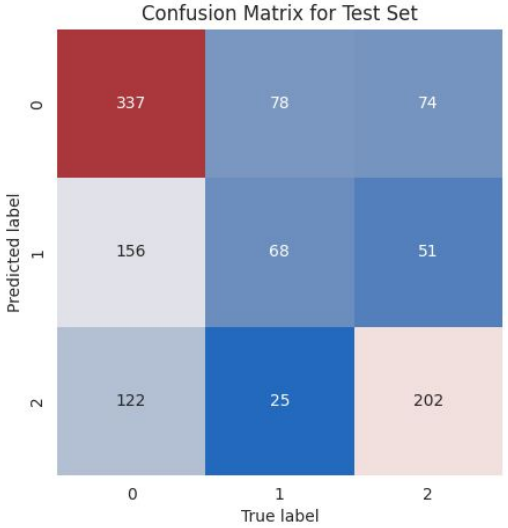
Overall, a larger batch size performed better. And, again, larger learning rates were better suited to the data.

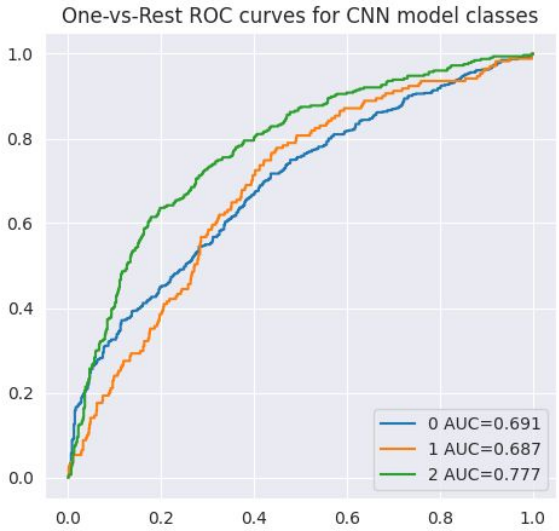


* true batch size of 8 with gradient accumulation step of 4

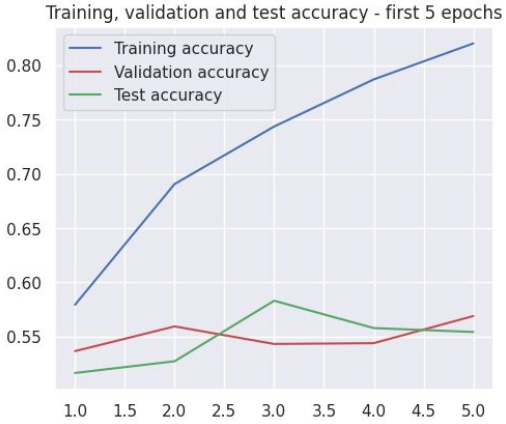
CNN Results

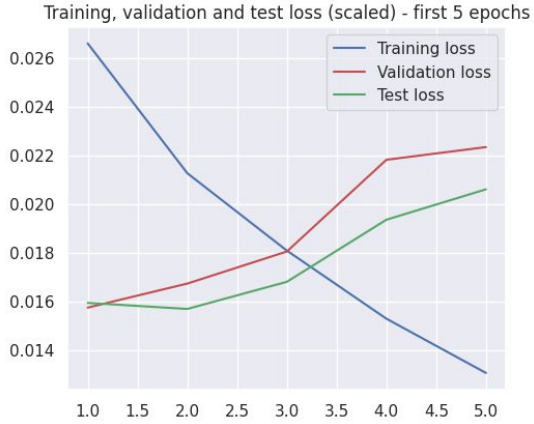
The data was filtered to only include 3 classes, US English, England English and South Asian English (India, Pakistan and Sri Lanka). Encoded as 0, 1 and 2 respectively. The confusion matrix and the ROC curve shows the breakdown of results for each class (CNN model).



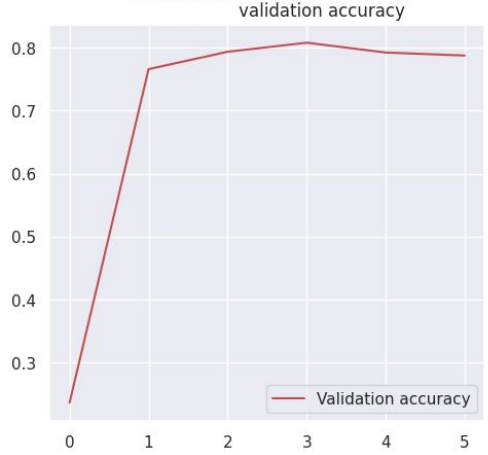


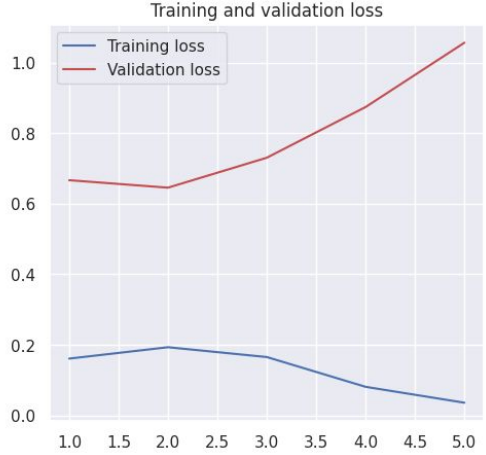
The CNN model was most successful at classifying South Asian accents primarily, then followed by US and England accents. This likely suggests that there is a greater acoustic difference between South Asian accents and US and English accents. English accents, for example were commonly misclassified as US accents over their true label.





CNN model (above), Wav2Vec2 model (below)





Wav2Vec2 Comparison

The CNN model overall achieved an accuracy of 57% of the validation set and 55% on the test set, surpassing Nicastro and Inguanez's baseline. Nicastro and Inguanez's model differs in structure and they segmented clips into 1 second samples.

The fine-tuned Wav2Vec2 proved to be the superior model - achieving a significantly higher accuracy of 80% on the validation set and 76% on the test set after the 3rd epoch.

The Wav2Vec2 model required more extensive data pre-processing and training than the CNN model due to its more complex feature extraction process and network architecture. This meant that I was not able to train the Wav2Vec2 model for as many epochs due to computational constraints.

Contribution

Research findings from Mikhailava et. al such as the effectiveness of Mel-Spectrograms on a linear scale have translated well when applied to large non-homogeneous data. The reduction in accuracy is to be expected due to the crowd-sourced non-homogeneous nature of the data which makes Common Voice difficult to classify - but ideal for modelling real-world interactions with these systems.

However, fine-tuning a pre-trained LAM such as Wav2Vec2 is more favourable than training a CNN model from scratch when dealing with large non-homogenous datasets such as Common Voice. The fine-tuned model achieved 80% and 76% accuracy on the respective validation and test sets, and Zuluaga-Gomez achieved a further 96% and 97% accuracy - compared to only 57% and 55% for a bespoke CNN on the same data. Zuluaga-Gomez et. al's results parallel results achieved by 'easier' datasets.

Overall, this report provides a comparison between two models for accent classification, one inspired by historically popular techniques for sparse data, and another emergent method that has shown great potential - showing favour towards the fine-tuned model. It also demonstrates a CNN-based method that improves upon existing CNN baselines using Common Voice.

Discussion and future work

My model did not exceed the baseline for fine-tuned Wav2Vec2 model on Common Voice set by Zuluaga-Gomez at 97% accuracy. This is likely for multiple reasons.

1. Zuluaga-Gomez used more advanced data preprocessing techniques and data augmentation.
2. They also used the full longer clips of varying length (in my method clips were cropped at 3 seconds and shorter clips were filtered out).
3. They used less samples per accent and classified more classes.
4. They used Wav2Vec2-XLSR with is trained on a multilingual corpus whereas I used the base Wav2Vec2 model which is only trained on English.

Many of these steps were not implemented due to computational and time constraints involved in the scope of this project as well as a desire to keep the preprocessing pipeline relatively similar between the bespoke CNN model and the fine-tuned model.

The use of Common Voice as a dataset in this research opens up many potential avenues for future work, especially if work was updated to use the Wav2Vec2-XLSR model. Research conducted on Common Voice can more easily translated to other languages using a different Common Voice variant which can help mitigate the early issue of bias towards English-language research. The same can be said for Wav2Vec2-XLSR.

Questions?

And thank you for listening!