<u>Deep Learning Methods to Classify Audio Segment Speaker Gender</u>

Craig Threadgold

Udacity Machine Learning Engineer Nanodegree Capstone Project

September 2018

Definition

Project Overview
It is becoming ever more important for automatic classification of audio data to be able to be deployed to the masses with companies such as Apple, Google, and Microsoft all having audio analytics functions as a core feature to their products. Gender identification plays a crucial role in automatic speaker recognition, by being used as a feature in subsequent models (Gaul, 2016). However, demand from less technically advanced business is growing at a rapid rate. For example, financial service companies are looking to collect metadata on customers who contact them by telephone. By using autonomous systems to extract information, it allows human resources to be more efficiently targeted to at-risk customers and to optimise the wider operational landscape. The technology used to perform this function is increasingly moving away from statistical models (such as Hidden Markov Models) towards machine learning: the emergence of deep learning has revolutionised the domain of speech recognition (Schmidhuber, 2015).

Artificial neural networks (ANNs) are inspired by cognitive biological nervous systems (Gaul, 2016). ANNs are created through the combination of many individual units (neurons) into a complex network, which work together to produce a global behaviour (ibid.). Neurons are architected into layers where each layer inherits the outputs of the previous layer. The middle layers are called hidden layers as they have contact with neither the input features nor the output classification. Deep learning is characterised by many hidden layers, where each one can use features from previous hidden layers to learn complex features, such as curves or typical shapes of a cat, for instance. These networks are intelligent because the neurons 'learn' by updating their respective weights through a backpropagation of a loss value from the final classification layer through to the input layers. ANNs are becoming ubiquitous in the technology sector and are increasingly being used in wider industry as their ability to classify images and, more recently, audio data is being realised.

Problem Statement
The aim of this project is to explore the applicability of deep learning techniques to be able to accurately classify tagged audio data as either from a male or a female. The audio features will include the Mel Spectrogram, MFCCs, Chroma, and Spectral Contrast. These features shall be fed through a deep convolutional network in order to attempt to find features that are endemic to either a male or a female speaker, and therefore to be able to classify the gender of the model based on solely the audio file. Previous models have been built for this purpose, such as Gaul (2016), Jerzy (2013), and Wang (2017). These models are highly developed, and I shall attempt to use inspiration from each of them. As discussed, the applicability of this automated service to industry will be vast, where companies will be able to collect information for further analysis about the demographics of people who are contacting them.

The first step will be to engineer features from the input data. The dataset has audio files in an MP3 format. These can be transformed into features by loading the audiofile into memory as a numpy array. After concatenating all the files together, the feature engineering shall be applied and the features split into batches in preparation for deep learning. The deep learning model will take a feature segments as an input, labelled as either male or female. The model will learn certain characteristics of what determines an audio segment to be from a male or a female, and then will be able to use these features to predict unseen segments. The prediction will be whether the segment has been spoken by either a male or a female.

A high-level workflow overview is as follows:

1. Extract audio segments and labels from dataset
2. Pre-process data by removing silence and splitting into uniform size
3. Train deep learning model
4. Evaluate results

Metrics

A popular loss function for machine learning problems is the mean square error. This method assumes that the error follows a normal distribution where the loss is a real-valued function of the input. However, this is not appropriate for a classification task because classes are not normally distributed. Cross-entropy loss methods are preferred because they assume a binomial error distribution (between the classes). Cross-entropy also has the benefit of providing faster learning when the model predictions differ substantially from the labels, which is generally the case for the first few epochs (Varma, 2018).

At each epoch of the deep learning model, the categorical cross-entropy will be calculated to quantify the predictive power of the model. Cross-entropy calculates the difference between the predicted classes of the probabilistic model (y) and the ground truth ($\hat{y}$) (DiPetro, 2016). By giving a probability for each one-hot encoded category of potential classification (i), the categorical cross-entropy sums these over n training examples to give one summary statistic. It takes the negative log loss for each training example as follows:

$$-logL(\{y^n\},\{\hat{y}^n\}) = \sum_n \left[-\sum_i y_i \log \hat{y}_i^n \right]$$

At each epoch of the model, the accuracy of the prediction of the validation set will also be used to compare the model's weights (as set by the cross-entropy) to an independent dataset, which can be classified. The accuracy will determine which epoch has the optimal weights set.

Although accuracy suffices to test our model at the training and validation stage, it is not sufficient to only use accuracy to determine the predictive power of the model. The accuracy paradox states that "predictive models with a given level of accuracy may have a greater predictive power than models with higher accuracy" (Afonja, 2017). For example, a model which always classifies an audiofile as female will have a very good accuracy if 90% of the input datapoints are female. Therefore, I shall use the precision, recall, and F1 score in addition to the accuracy to evaluate the testing set. For discussion on the definition of True Positives, False Positives, True Negatives, and False Negatives, please see Google Machine Learning (2018). The Precision, Recall, and F1-Score can be defined as the following metrics are defined as follows:

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

All these metrics have a value between 0 and 1, where 1 is the optimal solution and 0 indicates that the model has no predictive power. The Precision details the predictive power over the positively classified test datapoints. This is good when the cost of a False Positive is high, such as in email spam detection. Recall details how many of the actual positives have been classified as positive. This is to be used when the cost of a False Negative is high, such as in cancer detection. The F1-Score attempts to balance Precision and Recall and is useful when there is an uneven class distribution (Shung, 2018).

Analysis

Data Exploration

The aim of this project is to be open-source and therefore an open-source dataset has been used. The Common Voice Dataset (Common Voice, 2018) has 12GB of recorded and validated data. This dataset has labels for upvotes and downvotes, which can filter out poor-quality audiofiles (due to a bad transcription or poor-quality voice, for instance). Although in future I can use 12GB of data, I shall limit this due to training time and course restraints. 2000 audiofile datapoints shall be used (each file is approximately 5 seconds long giving nearly 2 hours of data). The datapoints shall be evenly split between male and female labels: 1,000 male audiofiles and 1,000 female audiofiles. The licence for the dataset is CC0 (CC0, 2018), which means that the author of the dataset gives anyone the right to use it.

The audiofiles in this dataset are given as mono .mp3s and a 48,000 sampling rate. This is all sufficient to process the files into any form that is required for machine learning. The audiofiles are one sentence long, which is sufficient for analysis, although longer files would perhaps have made the pre-processing stages easier. Features shall be taken directly from the audiofiles so no other information is required in this respect. There is enough labelled data to train, cross-validate, and test.

The train/validation/test datasets is pre-split by Common Voice. However, Wang et al.'s (2017) distribution of 60% train, 20% cross-validate, 20% test shall be replicated. To do this, the dataset shall be amalgamated into one list, disregarding Common Voice's train/validation/test split. The dataset is skewed towards male voices. Of the 73,000 datapoints for training, 75% of the voices are male. In order to make this more equitable, I shall select more from the categories that are under-represented in order to make a more generalised model. This dataset has been peer-reviewed, which makes the inputs very reliable. Some additional quality modifications shall be discussed in the methodology section.

The maximum audiofile length is 20.2s, the minimum is 1.2s, and the mean is 4.6s. The standard deviation is 1.58s demonstrating that the duration is not widely dispersed from the mean. The standard deviation of the male datapoints is slightly lower than the female datapoints at 1.48s compared to 1.67s. Despite this, the maximum and minimum values for the dataset are both male. The similarity of the distribution of male and female audiofiles shows that both classes follow the same format of one audiofile being one sentence. The characteristics of the classes should be similar, making the dataset choice congruent with a fair experiment.

The data is loaded to be manipulated as a one-dimensional numpy array in the shape (*duration,*) with type float (32bit) data:

```
array([-9.6513881e-09,  1.3265871e-08, -1.7428913e-08,  2.2109800e-08,
       -2.7251639e-08,  3.2764046e-08, -3.8520479e-08,  4.4351978e-08,
       -5.0034121e-08,  5.5285227e-08, -5.9747649e-08,  6.2977854e-08,
       -6.4434879e-08,  6.3468640e-08, -5.9312207e-08,  5.1082583e-08,
       -3.7788244e-08,  1.8353123e-08,  8.3441156e-09, -4.3406118e-08,
        8.7833179e-08, -1.4240022e-07,  2.0750262e-07, -2.8294980e-07,
        3.6769626e-07, -4.5947837e-07,  5.5427046e-07, -6.4544389e-07,
        7.2223980e-07, -7.6672126e-07,  7.4674904e-07, -5.9681327e-07,
        1.5200379e-07,  1.1933113e-06, -2.0046591e-05, -1.5644573e-05,
        6.9349171e-06, -5.5698456e-06,  5.2337709e-06, -6.5004124e-06,
       -2.2889924e-05, -6.2098570e-06, -5.4067464e-06, -6.5768036e-06,
       -1.6908709e-05,  1.0354515e-06, -9.1435027e-07,  1.5919918e-06,
       -3.8221256e-06, -2.5552041e-05], dtype=float32)
```

Figure 1 - Example Raw Audiofile Encoding

Exploratory Visualization

An audiofile is difficult to visualize. Commonly, it done by taking the Mel Spectrogram. A spectrogram is the representation of the sampled power of a sound based on a non-linear Mel scale of frequency. The dataset has been split into male and females, so shall be displayed as two independent inputs. Mel Spectrograms are a human-readable way of displaying data. By taking the logarithm of the Mel scale, the non-linear way that audio data is distributed can be condensed and presented in a way where meaningful patterns can be ascertained. Without doing this, long arrays of audio data are extremely hard to gather trends. The same principle can be applied for machines: Mel Spectrograms hold information on shapes and features of audiofiles. This is especially powerful in deep learning methods because the multi-dimensional data format can be used in convolutional architectures. In addition, speech is a non-stationary signal so normal signal processing cannot be applied (Bhattacharjee, 2017). By taking small samples of the speech, such as is done during a Mel filter bank, content of small duration can be treated as stationary. Mel Spectrograms use this and allow speech to become analysable. They are useful for this project because they present multi-dimensional data on speech for convolutional networks and because they allow a speech segment to be transformed into a series of stationary data.

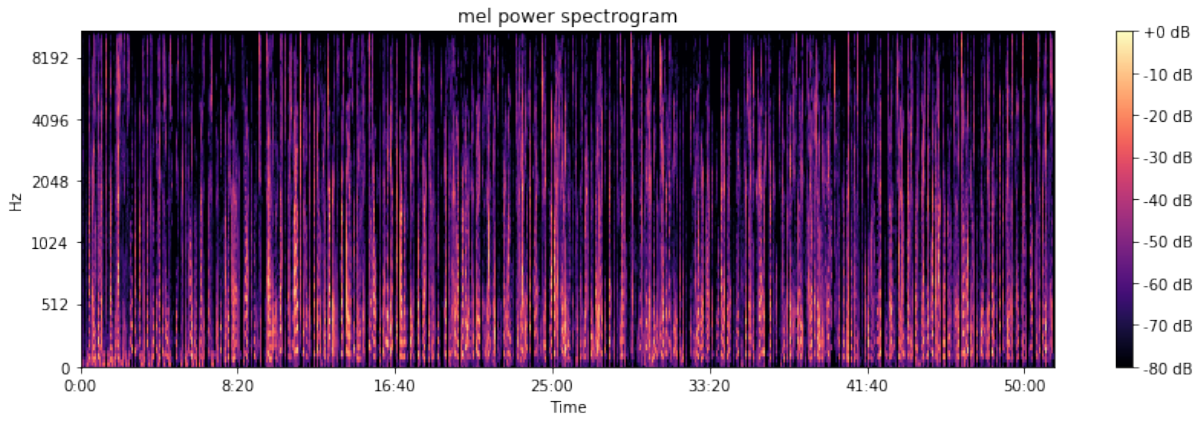

Figure 2 - Male Mel Spectrogram

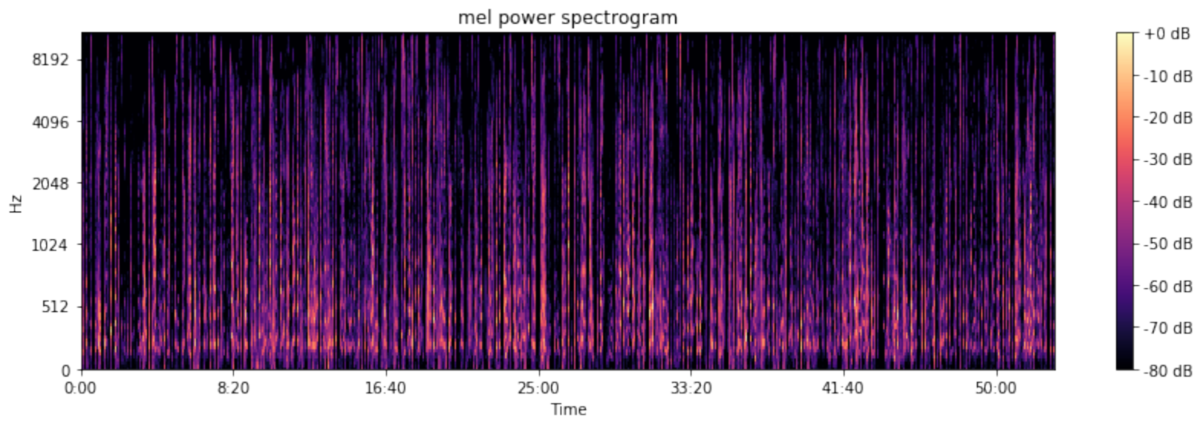

Figure 3 - Female Mel Spectrogram

The first extracted feature is the Mel Frequency Cepstral Coefficients (MFCCs). These coefficients collectively make up a Mel Frequency Cepstrum. Frequency warping in MFCCs allow for better compression of information, and is commonly used in deep learning to be able to reduce the training time without losing information. Mel Spectrograms are highly correlated, so by applying a Discrete Cosine Transform, the Mel Spectrogram filter bank is decorrelated and create a compressed representation of the filter bank. This is discussed further in the Algorithms and Techniques section. Correlated data has been found to be problematic in machine learning algorithms at times, meaning MFCCs can yield better results. However, MFCCs are an extension of Mel Spectrograms and display a stationary representation of our audio data, which machines can learn the common features of. The common features are likely to be different between males and females, and the difference in these features in the MFCCs is how the model will predict between male and female segments. The coefficients are scalars with no units.

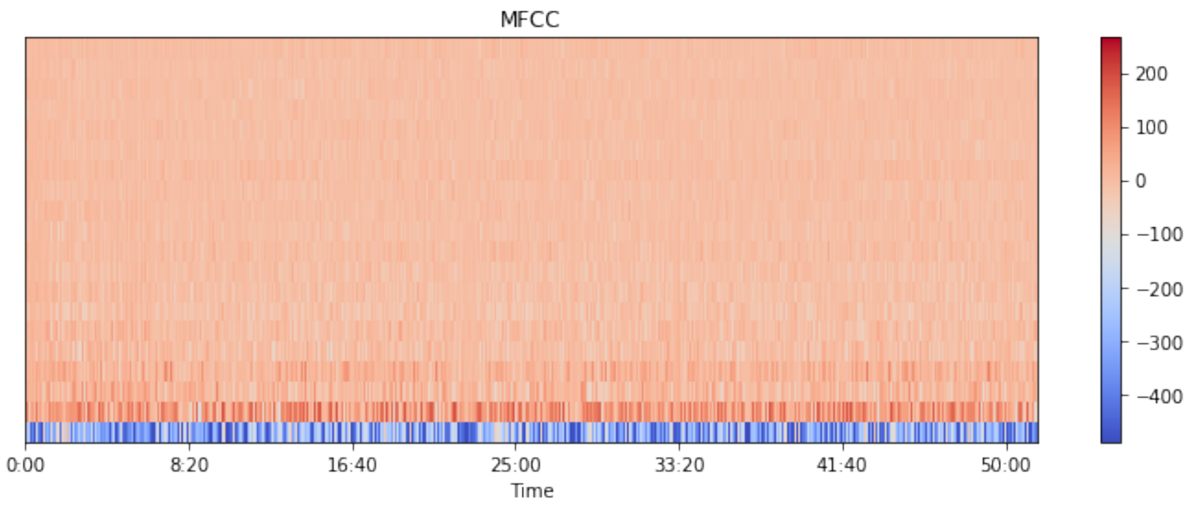

Figure 4 - Male MFCCs

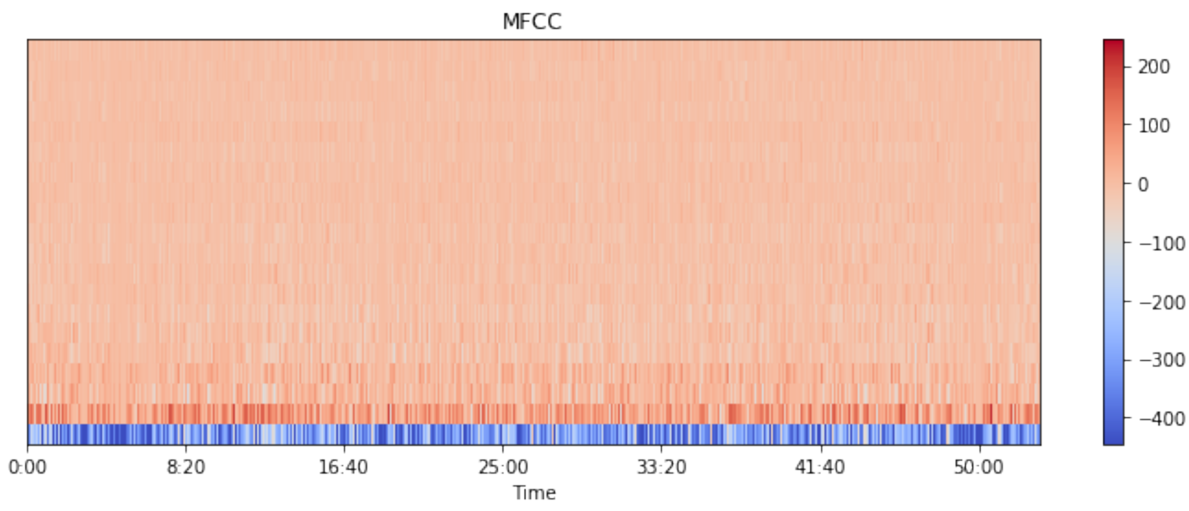

Figure 5 - Female MFCCs

The Chroma of an audiofile summarises the pitch. Coefficients are created to summarize the propensity for each of the twelve pitch classes to appear in the audio sample. A Chromagram displays each pitch class with a coefficient, between 0 and 1. Chroma is useful for machine learning applications because male and females typically have a different average pitch and propensity to produce different classes of pitch (Pepiot, 2014). The Chromagram will display the pitches observed and the model will be able to classify typical pitches for each gender.

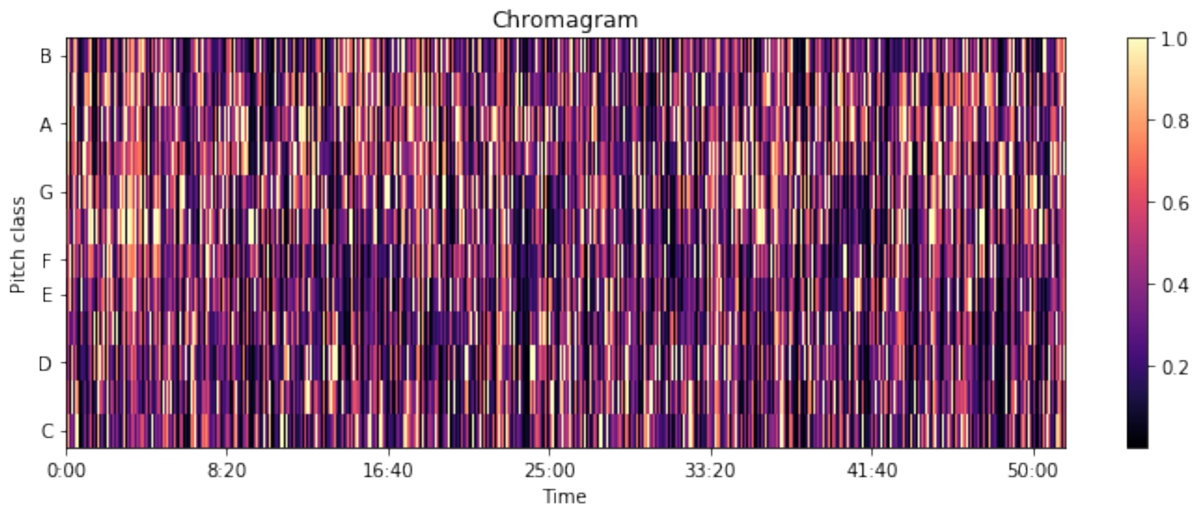

Figure 6 - Male Chromagram

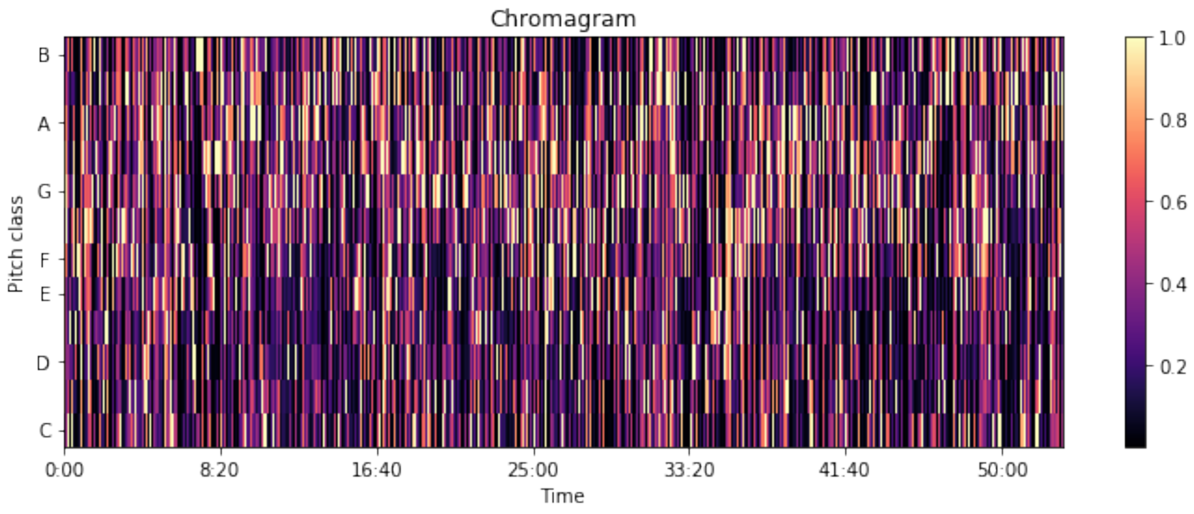

Figure 7 - Female Chromagram

Finally, the Spectral Contrast is the decibel contrast between the peaks and troughs in an audio sample (Yang, 2003). Much like the Chromagram, the average biological difference between how males and females contrast their speech amplitude through frequency bands is different (ibid.). Some studies have found that spectral contrast has a better discriminatory ability amongst different music types than MFCCs (Jiang, 2002) so the same principles can be applied to male/female speech.

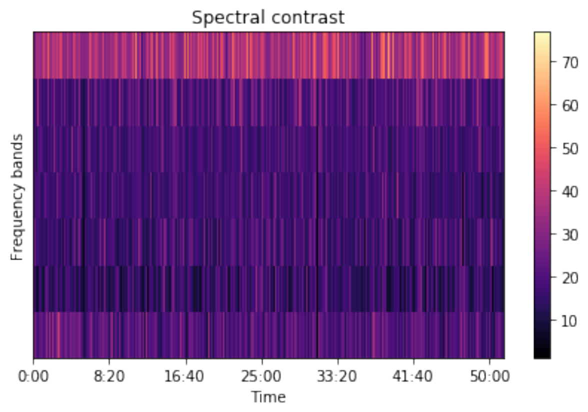

Figure 8 - Male Spectral Contrast

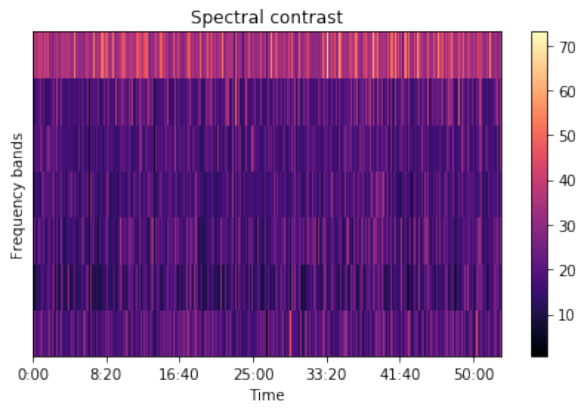

Figure 9 – Female Spectral Contrast

Algorithms and Techniques

As discussed, a neuron is an abstraction of how a biological unit in the brain works. Each neuron receives multiple inputs (x) which have associated weight (w). The result of the neuron (a) is the output of the weighted inputs passed through an activation function (f).

$$a_i = f(n_i)$$

$$n_i = \sum_j w_{ij} x_j + b_i$$

In addition to other inputs a bias value (b) always features in a neuron's summation. x_j is the j-th unit's input to the neuron, and w_{ij} is the weight applied to the j-th input. In ANN literature, this mathematical formulation can be depicted as follows:

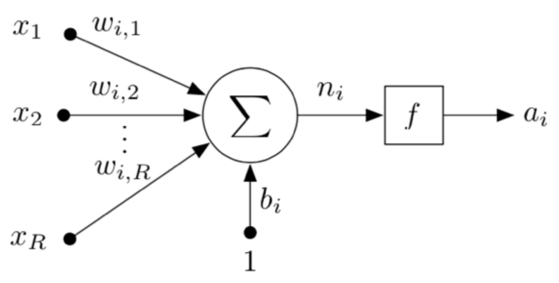

Figure 10 - Pictorial depiction of a neuron (Gaul, 2016)

There are many activation functions that are available to use. An activation function is an abstraction of how a biological unit transforms the summation of the input values (n_i) into a final result. In biological terms, the 'final result' is whether a neuron fires or not. The activation function takes the summation of weighted inputs and bias as an input and applied a mathematical function to give a final result. One of the most common activation functions is the Sigmoid function, which has a continuous output between 0 and 1 (ibid.).

$$Sigm(x) = \frac{1}{1 + e^{-\theta x}}$$

In this project, a Rectified Linear Unit (ReLU) is used as the main activation function, which is becoming increasingly common in deep learning ANNs. When the ReLU input is less than 0, the output is always 0 and when above 0, the output is the weighted sum of the inputs to the neuron (Yang, 2017). This method has been shown to have a far faster training time compared to the Sigmoid activation function. A drawback of ReLU is that with a large gradient, the neuron can update weights so that there is no output on a datapoint (i.e. the weights are set to 0). This can be mitigated by correctly setting the learning rate (ibid.).

So far, a single neuron has been defined. Neurons are connected together in a layer where they have the same inputs. Each neuron still has its own activation function, which is passed to the next layer (which can be another layer of neurons). The figure below shows a layer with S neurons and R inputs.

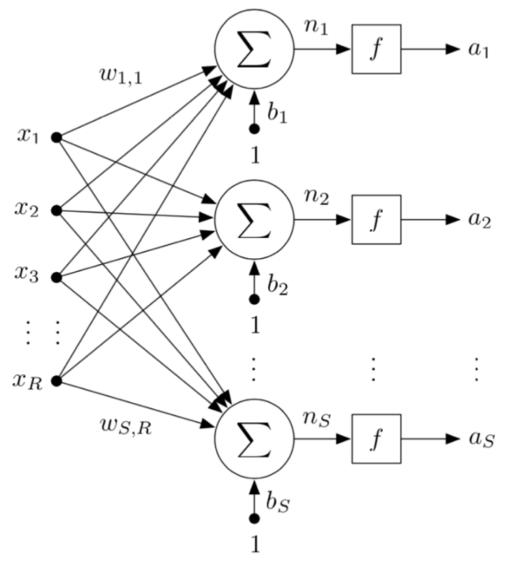

Figure 11 - A layer of neurons (Gaul, 2016)

As shown by *Figure 2,* each layer of neurons can then be passed to another layer. It is through this technique where hidden layers are able to identify extremely complex features. But to set the weights of a network, in order to create the summation of inputs to pass through the activation function, the network needs to be trained. The network initially starts with randomly initialized weights. The forward pass passes an input through the network and then the loss function determines a loss based on the classification of that image. Through the differentiation of the loss with respect to the weight, the weights that contributed the most to the loss is calculated. The weights are then changed in the opposite direction to this gradient (moderated by the learning rate) (Deshpande, 2016).

Recently, convolutional neural networks (CNNs) have built upon the discussed principles to be able to effectively classify data that is space-dependent. A classic example of spatial data is an image. A grayscale image is stored as a four-dimensional array, where each pixel (value) is between 0 and 255. The pixels in the array are not independent: many pixels can form the shape of a bus, for instance. The weights discussed before are not applied directly to the whole image, however, but are applied to parts of the image at a time: this is the convolution. A filter is an array containing weights, which is placed over the image. The weights are multiplied by the pixel value before the filter is moved over a different area of the image. This gives summary values for areas of the image and produces a new image which is smaller than the original image. The filters' weights are updated until they can detect features such as curves and even faces (ibid.).

Aside from convolutional layers, CNNs have pooling, dropout, and fully connected layers. Pooling layers can be considered a down-sampling activity: they tend to take the maximum pixel value from a larger area to reduce the amount of information processed. Dropout layers randomly reduce the number of activations of neurons and are used to avoid overfitting. Fully connected layers are created through a linear operation on the complex feature space that CNNs create. This layer can then be used for a classification process.

An issue with CNNs is that they tend to require a large amount of data to be able to build the complex feature spaces to recognise developed patterns, such as individual faces. In addition, a lot of computational power is required to do enough forward and back propagations to update the filter weights and build filters able to detect these complex shapes. This is largely not possible without large cloud computing infrastructure such as those provided by Google, Amazon, and Microsoft. To overcome this, transfer learning has been a powerful addition to the CNN methodology. Transfer learning allowed pretrained models to be used in personalised workflows. The weights for the lower layers of a pretrained model are frozen and act as a feature extractor. The top layers can then be cut off and replaced with layers that are specific for the task. For example, this project uses the VGG16 pretrained model, which has been trained on the ImageNet dataset. ImageNet is a dataset with over 14 million images that can predict over 1000 classes. VGG16 can, for instance, be used to predict the gender of an audiofile rather than one of the 1000 classes of animals, people, objects, which it was originally designed for (Yosinski, 2014).

The techniques that power the conversion of raw audio data into useable features is powerful but complex. An audio signal is passed through a pre-emphasis filter, sliced into overlapping frames, and then transformed to compute the power spectrum (Fayek, 2016). To compute further features, such as MFCCs, further transformations are applied to retain selected coefficients. For both the power spectrum and more developed features, a mean normalization is applied.

The first step to creating audio features is to apply a pre-emphasis filter, which amplifies the higher frequencies. This balances the frequency spectrum, avoids numerical issues during the Fourier transform operation, and improves the Signal-to-Noise ratio. A filter (α) can be applied to a signal (x) based on the time (t) of each datapoint:

$$y(t) = x(t) - \alpha x(t-1)$$

Typical values for the filter are 0.95 or 0.97. Then the signal needs to be split into short-time frames. Frequencies of a signal constantly change with respect to time so applying a Fourier transform over the whole signal would lose frequency contours over time. To overcome this, frequencies are assumed to be stationary for a short-time frame, usually 20 – 40 milliseconds. The frequency contours are therefore maintained for each frame and through frame-by-frame analysis, can be seen to change as the signal progresses. A window function is then applied to each frame (such as the Hamming window) to reduce spectral leakage (ibid.).

A Short-Time Fourier-Transform (STFT) is applied to each frame to calculate the frequency spectrum. The constant, N, is usually 256 or 512. For this project, a constant of 512 has been used. The power spectrum (P) is calculated based on the STFT for each frame:

$$P = \frac{|STFT(x_i)|^2}{N}$$

Finally, a filter bank is constructed by applying triangular filters on a Mel-scale to the power spectrum (P) to extract frequency bands. This is to mimic the human ear by being discriminative at lower frequency bands. The frequency bands (f) produced by the power spectrum can be converted to a Mel scale (m):

$$m = 2595log_{10}(1 + \frac{f}{700})$$

A Discrete Cosine Transform (DCT) decorrelates and compresses the filter bank coefficients to create Mel Frequency Cepstral Coefficients (MFCCs). MFCCs are usually limited between two and thirteen coefficients for Automatic Speech Recognition but this project shall investigate with 20 cepstral coefficients. The MFCC's feature vector displays the power spectral envelope of a single frame. Further features can be derived by calculating the change in coefficients over time (delta MFCCs) (Huang, 2001) but this is out of scope for this project.

Benchmark

The first benchmark model that was undertaken was a random classification. By comparing the model's results to a random classification, it will immediately show if the model has any predictive power at all. The random metrics should give an Accuracy, Recall, Precision, and F1-Score of 0.5.

	Predicted Male	**Predicted Female**
True Male	658	704
True Female	668	652

Figure 12 - Random Baseline Model Confusion Matrix

	Precision	Recall	F1-Score
Male	0.50	0.48	0.49
Female	0.48	0.49	0.49
Average	0.49	0.49	0.49

Figure 13 - Random Baseline Model Metrics

The next baseline model was to classify all datapoints as the same class. The aim of this was to check that the dataset was balanced. With an unbalanced dataset, initial models were able to be up to 80% accurate. Although this accuracy is good, the model predicted the same class for all inputs. The model had a very poor predictive power. By systematically predicting the results, it proves that the model is not defaulting to predicting a majority class.

	Predicted Male	Predicted Female
True Male	1362	0
True Female	1320	0

Figure 14 - Same Class Model Confusion Matrix

	Precision	Recall	F1-Score
Male	0.51	1.00	0.67
Female	0.00	0.00	0.00
Average	0.26	0.51	0.34

Figure 15 - Same Class Model Metrics

Finally, the best performing feature (Chroma) was passed through a very simple architecture. The metrics from this will be form the benchmark against my more complex models. The simple architecture has one convolutional layer, with a (2,2) kernel, and a ReLU activation function followed by a dense layer with 2 units. The baseline model proves that audio features have predictive power. The model continually learnt over 20 epochs but did show overfitting after epoch 7.

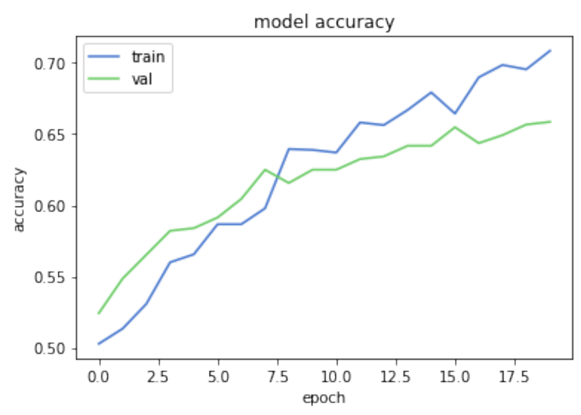

Figure 16 - Simple Baseline Model Accuracy

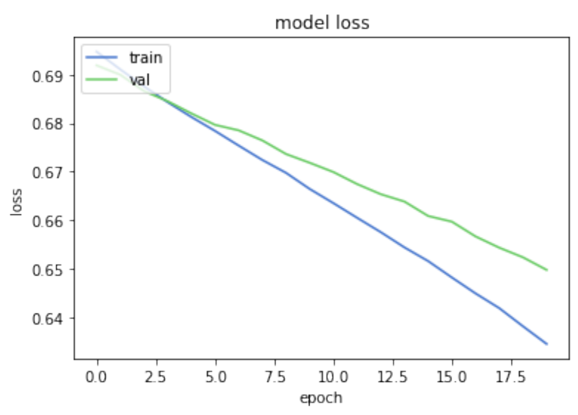

Figure 17 – Simple Baseline Model Loss

The confusion matrix shows that the correctly predicted classes outnumber the incorrectly predicted classes. This is emphasised by an F1-Score of 0.64, which shows that both the precision and recall were non-zero, disproving that the model was predicting the same class. More developed models should outperform this simple architecture, but the baseline does prove that there is enough information in the features to be able to build a predictive model.

	Predicted Male	**Predicted Female**
True Male	152	126
True Female	68	191

Figure 18 - Simple Baseline Confusion Matrix

	Precision	**Recall**	**F1-Score**
Male	0.69	0.55	0.61
Female	0.60	0.74	0.66
Average	0.65	0.64	0.64

Figure 19 - Simple Baseline Model Metrics

Methodology

Data Pre-processing

The Common Voice dataset has been partially pre-processed. It is an open-source, peer-contributed collection of audiofiles with associated metadata. All entries have been sense-checked by two layers. Firstly, the dataset is split into 'valid' and 'other' metadata CSV files. Valid audiofiles have been verified of good quality with accurate metadata. In addition, users can up and down vote entries. This project shall use the most upvoted audiofiles from the valid dataset, which should reduce the need for pre-processing.

The following steps were taken:

1. Identify CSV metadata to be used
    a. For this project 'cv-valid-train', 'cv-valid-test', and 'cv-valid-dev' were used
2. Combine metadata into one dataframe
3. Rank audiofiles by upvotes and split equally into male and female lists
4. Load the most upvoted audiofiles
5. Remove silence using Voice Activity Detection
    a. As discussed by Wang (2017), creating a new label for silence can unbalance the data. To overcome this, voice activity detection will be used in an attempt to remove all silence.
6. Create features
7. Split features into segment batches
8. Add channel dimension
9. Split into train/validation/test sets
10. For transfer learning, additional steps were taken
    a. Save Mel Spectrograms to the melspectrograms directory as a PNG
    b. Rescale the image by dividing each pixel value by 255
    c. Import VGG16 pretrained model

Parameter	**Description**	**Value Used**
feature_set_count	the number of total audiofiles to be processed	2000
df_paths	metadata CSVs to use	*discussed above*
sr	sampling rate	16,000
frame_length	*discussed below*	2048
hop_length	*discussed below*	512
top_db	decibel threshold for silence	60
n_mfcc	number of MFCC coefficients	20
datapoint_length	size of feature batch matrix	100
mono	mono or stereo audio file inputs	mono
n_mels	the number of mel filter banks used	128

Figure 20 - Audio Feature Parameters and Descriptions

Having concatenated all audiofiles into one audiofile for pre-processing, the global audiofile needs to be split into batches for deep learning. To improve the time resolution of the classification signal, the global audiofile is split into overlapping rather than adjacent segments (Gaul, 2016). The amount of overlapping is determined by the hop length (H). This indicates the number of samples by which two adjacent segments overlap. The frame length (W) is the length of each segment (Librosa, 2018). This is outlined by Figure 17 below.

The features were chosen based on some baseline model testing. Chroma showed to be successful for a model that performed better than the random and single class baseline models. Based on this, it was chosen to be used for the developed models. However, testing with Mel Spectrograms on transfer learning models proved to be effective, despite the failure on the baseline model, so Mel Spectrograms were used for all transfer learning models.

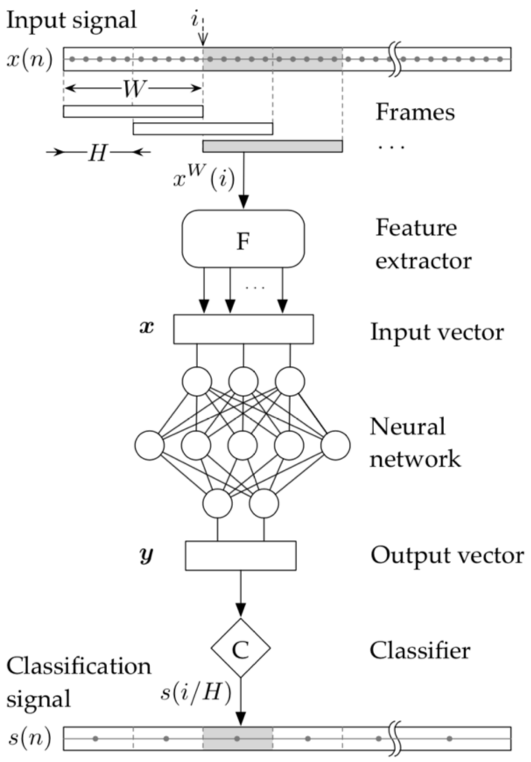

Figure 21 - Workflow Summary (Gaul, 2016)

Implementation
The implementation is different depending on whether transfer learning is used. The most significant investigative models that have been built have been saved in the Jupyter Notebook where this workflow was built. The non-transfer learning implementation used Chroma as a feature, whereas transfer learning used Mel Spectrogram images as a feature.

The following steps were taken to apply the algorithm to the dataset:

1. Pre-process data as per Data Pre-processing section
2. Define the model architecture and parameters
    a. Architectures were defined in either:
        i. class ModelGenerator
        ii. def build_and_train_model
        iii. Top models were individually created for transfer learning
    b. Loss functions, optimizers, metrics also defined
3. Train the model
    a. For transfer learning, checkpoint the model and save in parent directory
4. Plot the train/validation loss and accuracy
5. Test the model on the testing set
    a. Creation of metrics and confusion matrix
6. Compare the metrics to the baseline model and previously built models

It should be noted for transfer learning that features are reprocessed and saved in the following directories:

- './melspectrograms/training'
- './melspectrograms/validation/'
- './melspectrograms/testing/'

The PNG files in these directories are then used instead of the features that are saved in memory.

The architecture used for non-transfer learning contains 6 layers, including three convolutional layers and two dense layers. The input shape for this architecture is (12, 100, 1) where 12 pitch coefficients are given for 100 frames. A channel has been added as part of the pre-processing step but no additional information is created: the channel is to emulate an image, which is required for the TenorFlow backend. Using a kernel of (2,2) reduces the size of the feature space whilst MaxPooling and Dropout layers were used to reduce overfitting. Finally, the output was flattened and passed to a Dense layer with two units with a softmax activation function for classification. There were 2,987,146 trainable parameters.

Layer	Output Shape	Kernel	Activation	Units/Dropout
Convolution2D	(11, 99, 32)	(2, 2)	ReLU	32
Convolution2D	(10, 98, 48)	(2, 2)	ReLU	48
Convolution2D	(9, 97, 120)	(2, 2)	ReLU	120
MaxPooling2D		(2, 2)		
Dropout				0.25
Flatten	(23040)			
Dense	128		ReLU	128
Dropout				0.25
Dense	64		ReLU	64
Dropout				0.4
Dense	2		Softmax	2

Figure 22 - Non-Transfer Learning Original Architecture

Transfer learning has a different architecture. Firstly, the VGG16 pretrained model was imported. This architecture has 16 layers composed of convolutional layers (using a (3,3) kernel), and max pooling layers. Layers can be frozen where the weights are kept the same, or unfrozen where the weights can be updated through the training and backpropagation process. The base layer contained 14,714,688 trainable parameters.

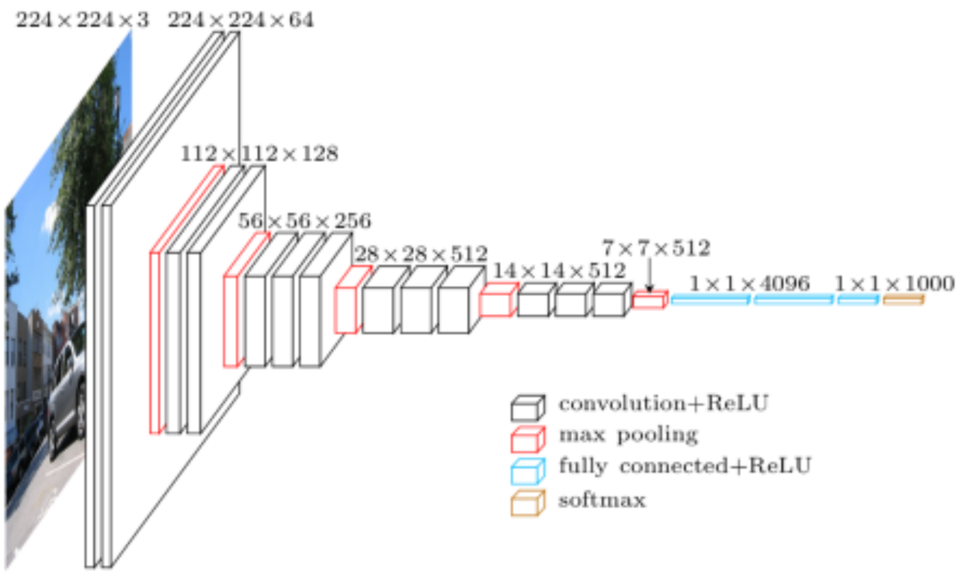
$224 \times 224 \times 3$
$224 \times 224 \times 64$
$112 \times 112 \times 128$
$56 \times 56 \times 256$
$28 \times 28 \times 512$
$14 \times 14 \times 512$
$7 \times 7 \times 512$
$1 \times 1 \times 4096$
$1 \times 1 \times 1000$
convolution+ReLU
max pooling
fully connected+ReLU
softmax

Figure 23 - VGG16 Architecture

In addition to the VGG16 architecture, a small architecture was appended in order to use VGG16 to classify the specific male-female problem. This included a Dense layer with 256 units and a dropout of 0.5 in order to reduce overfitting.

A significant difficulty whilst coding was transforming the data into a format that could be used to train a pretrained model based on images. The tensor had to be transformed to having three channels rather than one. To overcome this, the from_directory function provided in the keras library was employed. It took a long time to reach this solution and it only partially solves it: by saving to disk rather than to memory, the processing time is vastly increased. Additional coding issues included splitting the features into frames and batches. This was resolved by using the feature functions from librosa to apply the sampling, Fourier, and Mel filter bank transformations and only after splitting the features into batches.

Refinement

As mentioned, the baseline model with a simple architecture achieved an F1-Score of 0.64. The first attempt to improve on this used an architecture by Mandel (2017). This model performed significantly better than the baseline model. An F1-Score of 0.81 is an improvement on the baseline model's 0.64 demonstrating the value of a more complex architecture. However, the model accuracy and loss shows that it has a tendency to overfit. After the third epoch, the training and validation accuracy diverges. A training accuracy greater than the validation accuracy indicates that the model is learning characteristics of the training set that is not based on the gender of the audiofile. After the fourth epoch, the validation accuracy does not improve and the validation loss increases. The first refinement step is to reduce overfitting.

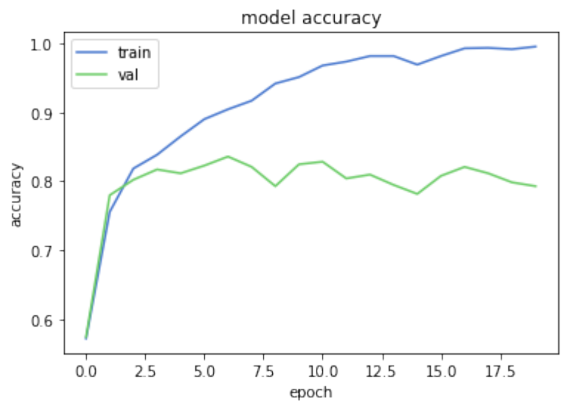

Figure 24 - Mandel Model Accuracy

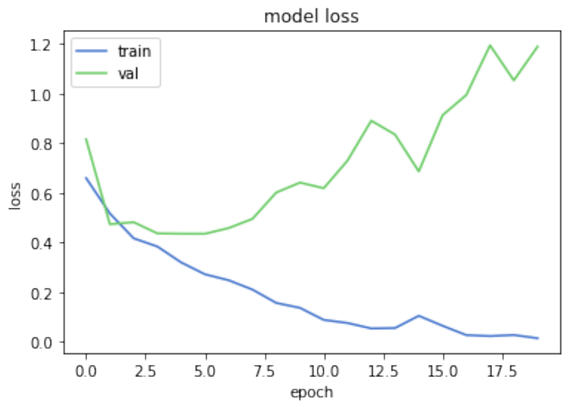

Figure 25 – Mandel Model Loss

	Predicted Male	Predicted Female
True Male	242	36
True Female	63	196

Figure 26 - Mandel Model Confusion Matrix

	Precision	Recall	F1-Score
Male	0.79	0.87	0.83
Female	0.84	0.76	0.80
Average	0.82	0.82	0.81

Figure 27 - Mandel Model Metrics

In an attempt to reduce the overfitting, the number of layers were reduced whilst keeping the same dropout layers. This does seem to have slightly reduced over fitting until the fourth epoch but then similar patterns of overfitting are shown. The F1-Score is slightly lower than the initial architecture at 0.79, and therefore the predictive power of the model has not been increased.

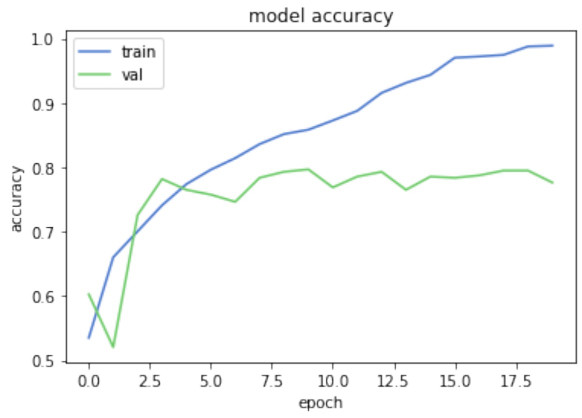

Figure 28 - Updated Mandel Model Accuracy

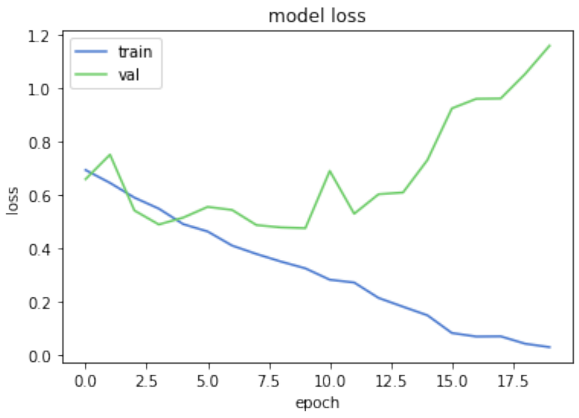

Figure 29 – Updated Mandel Model Loss

A large amount of literature has been written on the use of Recurrent Neural Networks, which take time sequence into account for learning (Hannun, 2014). These have been applied on audio datasets because what or how something is said it dependent upon what has been said before. A LSTM model was also used in the refinement process and did not outperform a more traditional convolutional model with an F1-Score of 0.73.

Layer	Output Shape	Units	Activation	Dropout	Recurrent Dropout	Return Sequences
LSTM	(12, 128)	128	tanh	0.05	0.35	True
LSTM	(32)	32	tanh	0.05	0.35	False
Dense	(2)	2	Softmax			

Figure 30 - LSTM Architecture

For transfer learning, lessons from the initial refinement process were learnt: use a simple convolutional network. The initial model used flattened the output from VGG16 into a Dense layer of 256 units before applying a Dropout layer at 0.5. A stochastic gradient descent optimizer was used with a learning rate of 0.0001 and momentum of 0.9. The initial results gave a F1-Score of 0.93 and far reduced overfitting over the same number of epochs compared to non-transfer learning methods.

	Predicted Male	Predicted Female
True Male	236	23
True Female	17	261

Figure 31 - Initial Transfer Learning Model Confusion Matrix

	Precision	Recall	F1-Score
Male	0.93	0.91	0.92
Female	0.92	0.94	0.93
Average	0.93	0.93	0.93

Figure 32 – Initial Transfer Learning Model Confusion Matrix Metrics

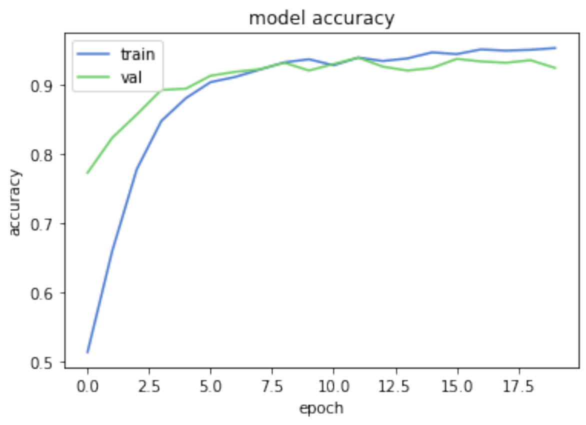

Figure 33 - Initial Transfer Learning Model Accuracy

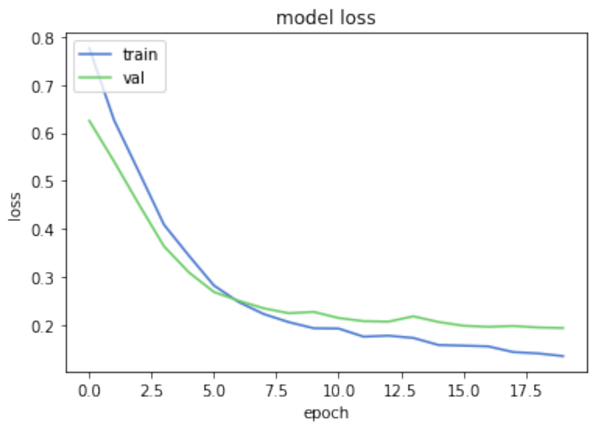

Figure 34 - Initial Transfer Learning Model Loss

The next step was to reduce the number of frozen layers from the VGG16 model. Previously it was set at 15, so only the final layer was trained. By reducing this to 10, the F1-Score reduced to 0.91 so this method was disregarded. The next refinement was to increase the learning rate to 0.01. It was clear from this that the learning rate was far too high as the model only predicted the male class so did not outperform the baseline models. Returning the learning rate to 0.0001, the optimizer was changed from SGD to Adam, which had the same impact of all predictions being the male class. The first model was therefore selected as the final architecture and parameters.

<u>Results</u>

Model Evaluation, Validation, and Justification

A training, validation, and testing set were used to develop the final model. A checkpointer was used to capture the outputs from the model at each epoch and only save the best model. The best model was determined by the accuracy achieved on the validation set, which is tested at each epoch. In addition to this, the final metrics were derived from an unseen testing set. The number of datapoints used to train, validate, and test this model was substantial. Two thousand audiofiles were used from the Common Voice dataset, and they were split into 2,681 batches to be individually processed by the model. The metrics were tested on a set that had 531 batches, which is statistically significant. The model was trained over 500 epochs (over two days), which gives sufficient time for trends to be seen and the best model to be found. The input data changes significantly because each datapoint is a different speaker. The model is therefore robust to changes in input data.

Model parameters:

- Optimizer – Stochastic Gradient Descent
- Learning rate – 0.0001
- Momentum – 0.9
- Number of frozen VGG16 layers – 15
- Top model
    - Dense layer – 256 units, ReLU activation function
    - Dropout layer – 0.5
    - Dense layer – 2 units, softmax activation function

The accuracy of the model throughout the training phase improved at pace until the 59th epoch where the validation accuracy started to plateau. There were some improvements in the validation accuracy, but it took many epochs to materialise. This is expected when using a cross-entropy-based loss function, where the loss is rapidly reduced during the first few epochs. The final model does show signs of overfitting to the training set, but the overfitting is far less pronounced compared to the baseline models. The best model was found at epoch 135 and beyond this the overfitting out-weighed caused the validation accuracy to decrease.

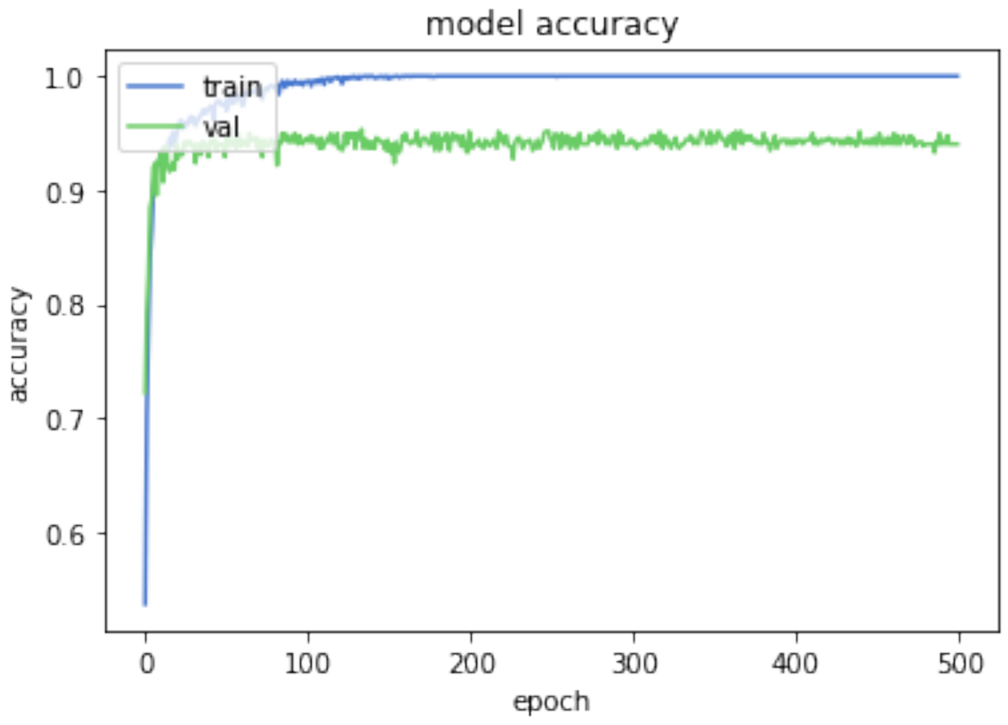

Figure 35 - Final Model Accuracy

The model loss shows a very similar story to the accuracy, except the overfitting is more pronounced. Even from the first few epochs the validation loss deviated from the training loss. This increased as the model was optimised to reduce the training set loss.

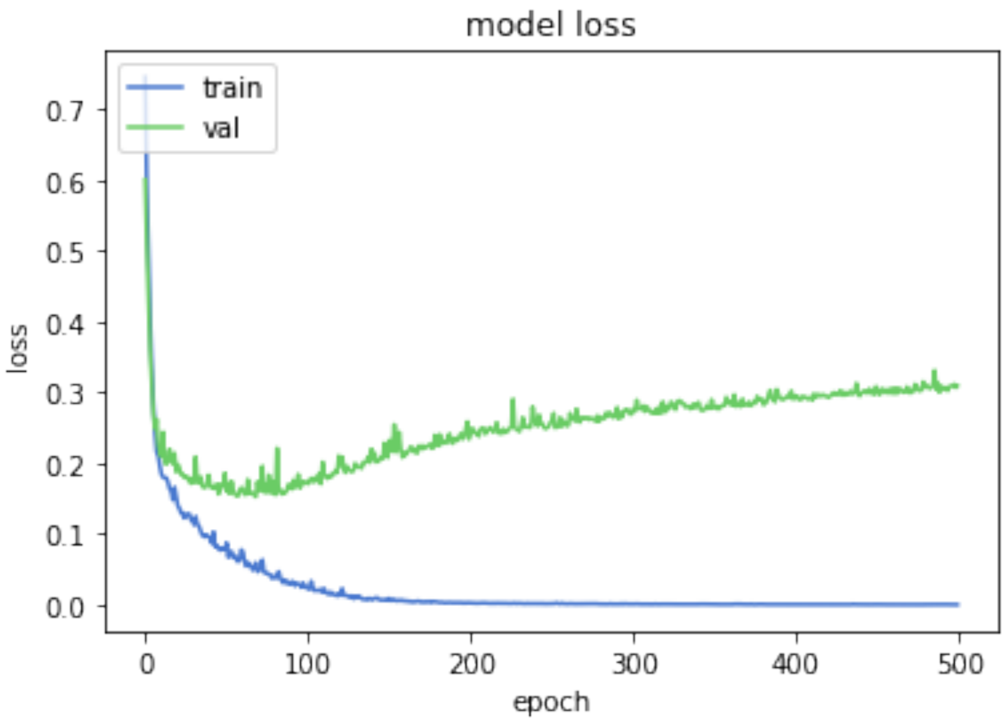

Figure 36 - Final Model Loss

The confusion matrix demonstrates that the model has predictive power. The testing set is independent of the sets used to develop the model and can be assumed to be representative for real-world applications. With 509 datapoints correctly classified out of 537 the model performs well with a 94.8% accuracy. These evaluative metrics show that the model is fit for purpose and the architecture/parameters are justified.

	Predicted Male	**Predicted Female**
True Male	245	14
True Female	14	264

Figure 37 - Final Model Confusion Matrix

Because 14 datapoints of each gender were misclassified, the Precision, Recall, and F1-Score do not differ between classes. An F1-Score of 0.95 is better than any of the previous models tested.

	Precision	**Recall**	**F1-Score**
Male	0.95	0.95	0.95
Female	0.95	0.95	0.95
Average	0.95	0.95	0.95

Figure 38 - Final Model Metrics

The initial benchmark model had an F1-Score of 0.64. On all metrics, the final model outperformed this benchmark. In addition, the model outperformed all non-transfer learning models and transfer learning models with other parameters. The final solution has overfitting typical of the benchmark model but to a far reduced extent. The high accuracy and F1-Score achieved on the independent test set demonstrate that this problem has been solved to an extent (for further discussion, see the Improvement section).

Conclusion

Free-Form Visualization

Audio files are hard to visualize. The Mel Spectrograms have been visualized in previous sections (along with other features). The output from this model is a one-hot encoded label, which has been statistically analysed and presented in the Results section.

Reflection

The following steps were taken to complete this project:

1. A problem statement was found
2. Open-source datasets were identified
3. The dataset was pre-processed
4. A benchmark was created
5. An investigation into available features was undertaken
6. Features were chosen and models were built with non-transfer and transfer learning techniques
7. Model parameters were refined to find a final model to train
8. The final model was trained over a period of two days

There were two steps that were very difficult for this project. Firstly, understanding how to transform audiofiles into features that could be read by a machine was challenging. This included having to research transformations from arrays into Mel-Spec Spectrograms, which required knowledge on the sampling rate, audio encoding, frame length etc. In addition, the shape and type of data that was created by the Librosa library was difficult to understand because of the number of abstracted transformations that are done by the library. This took a significant amount of time to understand and then implement. The second step that was difficult was understanding the whether a model could not learn because the features did not contain enough information or because the model architecture was poor. This is apparent because Mel Spectrograms do not outperform the baseline model on a non-transfer learning approach, whereas they perform very well as a transfer learning feature: they have the potential to be used, but only with the correct architecture. The literature on audio feature inputs to deep learning is far sparser than for images, and the final solution was found by making an audio feature into an image that can be processed.

The transfer learning workflow did not fit my expectation for the problem. I was predicting to be able to directly use the numpy array, and I am sure that there is a way to do this. Apart from this, the results exceeded my expectation. VGG16 is trained on the ImageNet dataset, and learns features for images. These include low level curves, which I expected to work with the curves shown in Mel Spectrograms. However, I was surprised that the best model was achieved by freezing nearly all the layers. The higher layers would have far more developed features, such as faces, animals, and objects, which are not found in Mel Spectrograms. Despite this, the results outperformed what I thought was possible with such a short amount of time. It is clear that this workflow can be used in a general setting. With the Common Voice dataset, it will be easy to apply the workflow to other labels, such as the age and accent.

Improvement

There are a few areas that could be improved. Firstly, there are multiple features created in the workflow and many are not used. A more thorough investigation into models that can be created with features such as Chroma and Spectral Contrast will allow multiple features to be passed to the transfer learning base model. In addition, more pre-processing steps can be added to allow the workflow to deal with poorer-quality data. This project was completed with a very good dataset, but this might not be reality for some applications (phone calls in high winds, for example).

The code base should be refactored to apply a lot of the transformation from pre-processing of the train/validation/test split into the *PreprocessGenderAudioData class*. This will make it neater and easier to integrate to existing workflows. By doing this, parameters for the model can be specified at the class initialisation stage, which will make it easier to build different models with various parameters.

The most significant improvement that could be made would be to change the transfer learning model used. As discussed, VGG16 is optimised for images, not audiofiles. The DeepSpeech (2018) project has pretrained a TensorFlow model on audiofiles, which will be far more optimised for this task. This is the most important improvement that can be made, as it will allow the models to be able to detect a finer level of labels such as who is speaking, what they're saying, and how it's being said (emotion analysis). The difficulty of this improvement is that DeepSpeech models have not been integrated into Keras but are available by using the TensorFlow package.

Finally, as discussed, an improvement will be how to pass the numpy array to the transfer learning model rather than encoding the features in a PNG file and then reading from these. This will significantly improve the processing time as it will reduce the need for the *def preprocess_for_transfer_learning* step, which uses disk space rather than completing the tasks in memory.

By using a different transfer model, having time to change more parameters and architectures, and by increasing the number of features used simultaneously, this model will surely be able to be bettered. It is a good benchmark for the next round of my investigation.

<u>Reference List</u>

Afonja, T. (2017). Accuracy Paradox. Available: https://towardsdatascience.com/accuracy-paradox-897a69e2dd9b. Last accessed 22nd September 2018.

Apple. (2017). Hey Siri: An On-device DNN-powered Voice Trigger for Apple's Personal Assistant. Available: https://machinelearning.apple.com/2017/10/01/hey-siri.html. Last accessed 13th July 2018.

Bhattacharjee, M. (2017). How do we use MFCC coefficients in machine learning algorithms?. Available: https://www.quora.com/How-do-we-use-MFCC-coefficients-in-machine-learning-algorithms. Last accessed 24th September 2018.

Bourlard, H and Morgan, N (1994). Connectionist Speech Recognition: A Hybrid Approach. Boston: Kluwer Academic Publishers.
CC0. (2018). CC0. Available: https://creativecommons.org/choose/zero/. Last accessed 13th July 2018.

Common Voice. (2018). Common Voice Dataset. Available: https://voice.mozilla.org/en. Last accessed 13th July 2018.

DeepSpeech. (2018). Project DeepSpeech. Available: https://github.com/mozilla/DeepSpeech#continuing-training-from-a-frozen-graph. Last accessed 23rd September 2018.

Deshpande, A. (2016). A Beginner's Guide To Understanding Convolutional Neural Networks. Available: https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks/. Last accessed 22nd September 2018.

Fayek, H. (2016). Speech Processing for Machine Learning: Filter banks, Mel-Frequency Cepstral Coefficients (MFCCs) and What's In-Between. Available: https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html. Last accessed 24th September 2018.

Gill, C. (2017). Automatic Speaker Recognition using Transfer Learning. Available: https://towardsdatascience.com/automatic-speaker-recognition-using-transfer-learning-6fab63e34e74. Last accessed 13th July 2018.

Google Machine Learning. (2018). Classification: True vs. False and Positive vs. Negative. Available: https://developers.google.com/machine-learning/crash-course/classification/true-false-positive-negative. Last accessed 22nd September 2018.

Hannun, 2014. (2014). Deep Speech: Scaling up end-to-end speech recognition. Baidu Research. 1 (1), 1-12.
Hinton, G et al.. (2012). Deep Neural Networks for Acoustic Modeling in Speech Recognition: The shared views of four research groups. IEEE Signal Processing Magazine. 29 (6), 82–97.

Huang, X. (2001). Spoken Language Processing: A guide to theory, algorithm, and system development. Prentice Hall.

Jiang, D. (2002). Music type classification by spectral contrast feature. Proceedings. IEEE International Conference on Multimedia and Expo, Lausanne, Switzerland, pp. 113-116 vol.1.

Librosa. (2018). Glossary. Available: https://librosa.github.io/librosa/glossary.html. Last accessed 23rd September 2018.

Mandel, M. (2017). Building a Dead Simple Speech Recognition Engine using ConvNet in Keras. Available: https://blog.manash.me/building-a-dead-simple-word-recognition-engine-using-convnet-in-keras-25e72c19c12b. Last accessed 23rd September 2018.
McFee, B et al. (2015). A Software Framework For Musical Data Augmentation. Available: http://bmcfee.github.io/papers/ismir2015_augmentation.pdf. Last accessed 15th July 2018.

Erwan Pépiot. Male and female speech: a study of mean f0, f0 range, phonation type and speech rate in Parisian French and American English speakers. Speech Prosody 7, May 2014, Dublin, Ireland. pp.305-309, 2014.

Schmidhube, J. (2015). Deep learning in neural networks: An overview. Neural Networks. 61 (1), 85–117.

Shung, Koo. (2018). Accuracy, Precision, Recall or F1?. Available: https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9. Last accessed 22nd September 2018.

Varma, R. (2018). Picking Loss Functions - A comparison between MSE, Cross Entropy, and Hinge Loss. Available: http://rohanvarma.me/Loss-Functions/. Last accessed 22nd September 2018.

Wang, Z et al.. (2017). Learning utterance-level representations for speech emotion and age/gender recognition using deep neural networks. Available: https://www.researchgate.net/profile/Ivan_Tashev/publication/316009054_Learning_utterance-level_representations_for_speech_emotion_and_agegender_recognition_using_deep_neural_networks/links/5a29e14eac. Last accessed 13th July 2018.

Wang, Z et al.. (2017). Learning utterance-level representations for speech emotion and age/gender recognition using deep neural networks. Available: https://www.researchgate.net/profile/Ivan_Tashev/publication/316009054_Learning_utterance-level_representations_for_speech_emotion_and_agegender_recognition_using_deep_neural_networks/links/5a29e14eac. Last accessed 13th July 2018.

Yang, Ji. (2003). Spectral contrast enhancement: Algorithms and comparisons. Speech Communication. 39 (1), 33–46.

Yang, Ji. (2017). ReLU and Softmax Activation Functions. Available: https://github.com/Kulbear/deep-learning-nano-foundation/wiki/ReLU-and-Softmax-Activation-Functions. Last accessed 2nd September 2018.

Yosinski J, Clune J, Bengio Y, and Lipson H. How transferable are features in deep neural networks? In Advances in Neural Information Processing Systems 27 (NIPS '14), NIPS Foundation, 2014.