Final Report

Stephen Bourke

A00290961

Word Count: 3604

**Introduction:**

I have done the placement under the guidance of Mehwish Tahir through TUS Athlone itself. As such I am working on my own project rather than in a workplace. I applied to more than a dozen companies hoping that one would take me on as an intern for the duration of the twelve weeks but had also signed up to do the placement with the college if I could not find anywhere else.

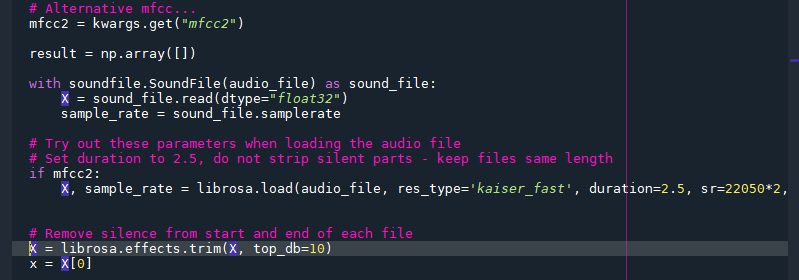
After some discussion with my supervisor, I decided to work on an application which can detect a user’s emotion from their speech through the use of neural networks. I chose this topic as I felt that it would allow me to build on what I had learned about neural networks during the course. It has been beneficial in that I have learned more about constructing neural networks, and more about dealing with files and building a fully functioning program. The results of the project have been broadly in line with similar projects which are available on the internet, with the predictive model performing well on the test data of a train-test split, with an accuracy of 77%. However, when used on a completely different set of data from a new dataset accuracy drops to roughly 36%. This result is also in line with a popular speech emotion recognition project that is available on github.

**Responsibilities and Duties:**

The first task was to find suitable datasets containing labelled examples of emotional speech. My supervisor, Ms. Tahir, suggested the website superkogito.github.io which contains a large selection of such datasets, of which I used the Ravdess, Tess, Crema, Savee and Emo-DB datasets. Each of these consist of hundreds or thousands of audio files containing a phrase or sentence spoken by an actor, adopting one of a number of emotional styles, for example, happy, angry or fearful. Each file is also labelled according to the emotion that is being conveyed.

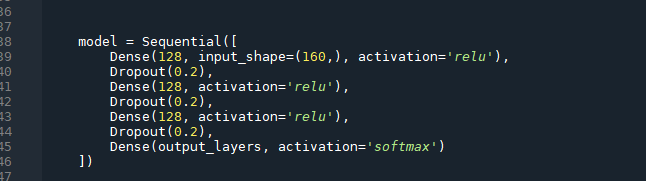
I created a number of functions in python for importing the audio files and retrieving the information about each file, including the emotion label itself, which formed part of the filename. I also created a function which converted all of my audio files into mono and set the sample rate to 16000 Hz, to make them uniform. After some research online, it appeared that the python package, librosa, was frequently used for extracting some features from audio files which would be useful for speech emotion recognition. Ideally I would have had a better knowledge of sound-waves and been able to manipulate them myself to extract those features most relevant for identifying the emotion of a speaker. I did some reading on the internet about the nature of speech files and how they might be manipulated in order to make them more useful for the task at hand. However, since I was having difficulty understanding the finer mathematical points of this area, I decided to simply utilise librosa since this was a common approach amongst those doing similar projects. My program allows the user to include a number of different audio representations, namely ‘MFCC’ (mel frequency cepstral co-efficient), ‘Mel Spectrogram’, ‘Chroma’, ‘Tonnetz’ and ‘Contrast’. I generally used ‘MFCC’, ‘Mel Spectrogram’ and ‘Chroma’ since these were the most commonly used parameters in the online projects that I had seen.

After feeding the audio files into librosa, the output for each audio file was a series of numbers (160 or 180 depending on some of the parameters used with librosa) which gave information about the audio sample, and a label representing the emotion relating to each series. Since many online projects were using a mean value of these parameters I did not have to worry about the differing duration of speech samples as the same number of numbers (160) was being returned for each audio sample. Some online projects removed any sections of silence from the beginning and end of an audio sample, while others did not do this. I tried this at a later stage of the project when I was just using the MFCC values but it did not affect the accuracy of the model either way. I came across the use of ‘kwargs’ in one of the online projects which allowed me to implement whichever librosa features that I required by adding them in when the function was being called.



*Figure 1. Using Librosa to extract relevant features from speech sample*

The next task was to create a suitable neural network which could process this data. As shown in Figure 2 below, my initial neural network consisted of two hidden layers, with 128 nodes at each layer. This simple neural network was then used to build a predictive model for each of the datasets that I decided to utilise.



*Figure 2. Initial Neural Network Configuration*

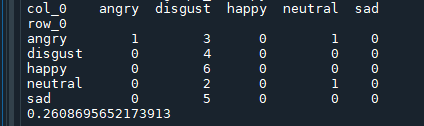
Of the five datasets, Tess performed the best on its own test data, with a remarkable accuracy score of 99.5%. The other datasets generally managed to achieve results of between sixty and seventy percent accuracy on their test data. I believe that the reason for Tess’s exceptional results is that the actors in this example speak in an extremely exaggerated manner, so that distinguishing between the emotions is an easier task.

When it came to dealing with speech examples from the other datasets, however, the model based on the Tess data performed poorly. For example, the Tess model only achieved an accuracy of 17.6% when used to predict the emotions in the Ravdess dataset. I decided to record some of my own speech samples for testing purposes, and ended up with 37 samples. Again, Tess performed poorly when used to predict my samples. Indeed, the models for each dataset performed poorly when given data from any other dataset. The exception here was the model produced from the Ravdess dataset, which gave a slightly better result of 24% on my own data.

The next task was obviously to try to improve the accuracy of the models. I decided to try some other machine learning algorithms, namely decision trees, random forests and SVMs, to see how they would perform. Interestingly it was the model created using the Crema dataset which performed the best here, giving an accuracy of 29% on my data when using an SVC algorithm. I attempted to improve this performance further by trying out different parameters with the aid of GridSearch but my computer didn’t seem to be able to handle this task and I decided to stop the operation after it had been running for many hours, an issue which I had come across during the second semester of the course as well.

Given this difficulty and the fact that most of the examples of speech emotion recognition projects on the internet favoured neural networks, I decided to focus on using them. Having already tried a basic neural network, I next used a convolutional neural network which I used with the Ravdess dataset. The accuracy on the test data was roughly the same as for my initial network, but the accuracy on my own dataset was only 13%.

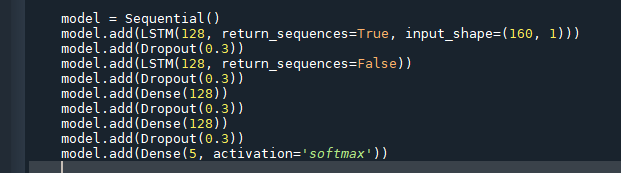
My next approach was to combine the datasets in the hopes that the models produced would be less closely associated with a particular style of speech. Again, most of the combinations performed poorly on unseen data, with the best result being 26% accuracy on my own data using Ravdess, Tess and Crema datsets combined. As can be seen from figure 3 below, however, almost all of my speech samples were being labelled as disgust.



*Figure 3. Model from combination used to predict my own data*

My next task was to divide the actors used in the speech samples by gender, but this did not improve results by much. I also created models using subsets of the emotions in an attempt to boost the accuracy of the results. Using a subset of four emotions, namely ‘angry’, ‘happy’, ‘sad’ and either ‘disgust’ or ‘fear’, the accuracy when used to predict my data reached 55% using Ravdess and 40% using Crema models. The Ravdess model also reached an accuracy of 45% when used to predict the emotions of the Crema dataset. While this was a big improvement over the earlier results, I was keen to pursue a model which could predict more than just four emotions.

Given how poorly my models were performing in general, I next tried to replicate a recurrent neural network from a project that was available on github (<https://github.com/x4nth055/emotion-recognition-using-speech>), as shown to me by my supervisor. I chose to use an LSTM, the structure of which is shown in figure 4 below.



*Figure 4. Structure of RNN*

I used just five emotions for training this model, using a combination of Ravdess, Tess and Emo-DB datasets. Again the librosa parameters I used were MFCC, mel and chroma. This resulted in an accuracy of 77% on the test data, which was exactly the same as the figure achieved in the original project on github. Unfortunately, yet again the model struggled to correctly predict the emotions of completely new audio samples. It returned a result of approximately 36% on both my own speech samples and the Crema dataset. I also downloaded the original project itself and built a model from it, just in case there was some discrepancy between my code and the github code, but the accuracy results that this model achieved were the same. The original project contained series of 180 numbers as opposed to the 160 numbers in my project, due to the fact that the author used a slightly different set-up for the MFCC calculation. I set about tweaking my model, and reprocessing the datasets so that they would return series of 180 as well. The results, however, were identical whether using 160 or 180 markers for the processed audio.

Finally, I decided to try out a different approach, using just the MFCC parameter in librosa, rather than using three librosa features (MFCC, mel, chroma). This approach had been used in another project on github (<https://github.com/MiteshPuthran/Speech-Emotion-Analyzer>). However, when trying to replicate this code I found that I was not getting back the same array length for a processed audio file as the original. From subsequently looking at the feedback for this project, I saw that I was not the only one who was getting different results from the original project. I also tried out using the raw MFCC output as well as the mean values. Using the raw output meant that each speech sample had to be the same duration. However, the accuracy went down when just using this single audio feature so I didn’t pursue it further.

**Technical Achievements:**

The primary technical achievement of the project was in learning more about neural networks. I have gotten a better understanding of how to implement different architectures and hyper-parameters in order to improve the accuracy of the model. I also gained experience creating convolutional and recurring neural networks and increased my knowledge of the shapes of the input layers used in such networks.

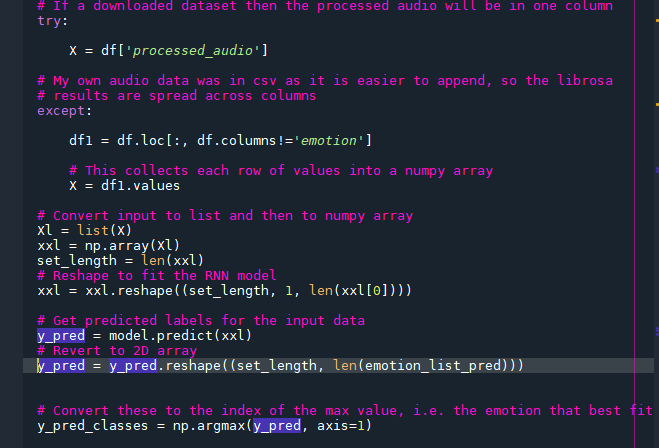
While it may not be a specific technical area that I mastered, simply getting the code to work was a technical achievement in and of itself. I cannot necessarily recall all the little tweaks and adjustments that had to be made to my code to get everything to work properly, but this process was a great learning experience and has helped me to be more comfortable with debugging code. In particular I enjoyed trying to diagnose a problem by setting breakpoints and trying to figure out why a piece of code was not working as it should.

Figure 5 below is a sample of some code which illustrates some of the technical work that needed to be done. The try except block is needed to deal with different input data. While I was able to save the processed data from each of the datasets in a json file, I decided to use a csv file for saving the processed data for my own speech samples as it would be easier to append to this than to json as new recordings were added. There might have been a more efficient way of doing this but at the time that was what I chose to do. I was having a slight problem putting my lists of numbers into the csv files and then retrieving them again so I gave each number in the list its own column and then concatenated them again when using them. This is shown in the code below where the df1 object is made up of all of the columns of the dataframe except the column containing the emotion label.

The next section shows where I had to alter the shape of the array so that it would work with the RNN. Although this looks very simple it took me a while to figure out exactly how to achieve this. Interestingly, while I followed the advice of online articles in reshaping my array, it appears as though the project on github that I downloaded has reshaped their array in a different way. My model achieved the same results as this online one so it does not seem to be a problem. The examples in online articles were often very simple ones, and didn’t really illustrate how more complex arrays could be reshaped, so I was unable to understand exactly why both shapes seemed to work okay.

Finally, to retrieve the emotion which achieved the highest score and was thus the best fit for a given audio sample, I employed the np.argmax() function.

While all of this only amounts to a few lines of simple code, I did not have prior experience of most of it, so each line had to be researched and understood before I could implement it.



*Figure 5. Excerpt of Code*

Another technical achievement of my work has simply been in working with the various files and creating functions which are usable with different input files. Not having worked much with external files until now, it has been interesting to learn about how to work with them and to learn about setting up robust, flexible code which is reusable and will not throw errors unexpectedly. I have also become more comfortable creating functions and have enjoyed this object-oriented aspect of the project, where I have had to create functions which are able to work with datasets which are different from each other in their structure.

**Challenges:**

Before beginning the project I had envisaged that I would achieve a lot and create a model which worked very well. However, progress was slower than I thought it was going to be. Actions which I thought would be straightforward threw errors quite a lot and I was forced to go searching online for an answer to an issue quite frequently. I also found that I often didn’t understand what the problem was as the error messages can be a bit difficult to decipher, making it harder again to find a quick solution. For instance, on attempting to use GridSearchCV with my neural network the results were a lot poorer than expected, and I was unsure as to why. The online documentation is often rather dry and spare in style.

However, I tried to be content with taking it slowly and tried to read about the theory behind a piece of code rather than just copying it and using it blindly. I also found a number of good websites which explain things in detail and don’t assume lots of prior experience and knowledge, such as machinelearningmastery.com.

Another challenge I encountered was in creating functions which I could use with different datasets. Each of the datasets used a different set of emotions in its speech files thus it was important to create functions which could deal with these different inputs. In reality, I feel that I could have done better here, as while my functions were adequate to begin with, I ended up relying on hard-coding as the project become larger and more complex. For example, when working with the recurrent neural network, it was necessary to alter the shape of the input data from a 2-D array to one which is 3-D. I probably should have created a different function to deal with this scenario but I resorted to simply commenting / uncommenting blocks of code as needed. One thing which I could have done but forgot about was to use a more structured approach to the creation of the project. I had previously learned about the agile process and the importance of drawing up a blueprint of how the project should be constructed. I feel that had I implemented this approach I would have focused on creating more open and flexible functions which could deal with different datasets from the start, rather than rushing to create a function straight away that would only be suitable for one specific dataset.

At times I also found it hard to stay motivated as the accuracy of my models was not very high. The results reported by online articles seemed to be very high and this was disheartening at times. However, after I had downloaded and recreated a model from github it appears that the high accuracy results achieved by these projects was generally just referring to test data. When I used these models on an entirely new unseen set of data the accuracy dropped a lot and performance levels were very similar to that of my model.

I also found it difficult to work on the same project all the time without a huge amount of variety. Here I feel that an internship would have been more interesting and varied and I might have gotten a more rounded experience of working as a programmer, as I might have been exposed to different areas such as testing. However, I did at least get to work with building up an entire program through multiple functions, as well as working with files and debugging errors.

**Analysis / Evaluation:**

Overall I am reasonably happy with the way the project worked out. In retrospect I probably would not have chosen to work with audio files as my knowledge of speech signals is limited and I struggled to understand the details of them when I tried to research them further. As such I simply ended up using the same librosa features that others had used without gaining a better understanding which elements of the speech samples my network was being trained on. Thus I wasn’t able to tweak the data to emphasise certain aspects of it. If I had worked with visual data I feel that perhaps it would have been more intuitive and that I would have had a better grasp of the data I would be working with.

While my final model performed quite well on the test data from its own dataset, correctly predicting the emotion of the speaker 77% of the time, it only achieved a result of 36% accuracy when used on my own speech samples, and on the Crema dataset. Thus, it would appear as though the model might be useful if one could be trained specifically for each individual user, but it does not seem to work well when faced with a completely unknown speaker. This could be due to the fact that everyone has a different style of speaking and it is quite difficult to generalise from one person’s speech patterns to those of everyone else. Certainly, the speech samples contained in the Tess dataset were extremely exaggerated and unrepresentative of general everyday speech. Again, I feel that had I better understanding of sound waves I may have been better able to identify how to pinpoint specific features within a speech sample that might be linked to the emotion of the speaker. After having worked hard over both semester one and two, I was a bit tired and not as sharp as I had been earlier on in the course and thus found it tricky to get a good grasp on the more mathematical side of sound waves.

Despite this, I do feel that I gained good experience of not just working with neural networks, but also of creating a fully functioning program made up of many different parts. I am also currently seeking employment and the project has helped in keeping my coding skills ticking over. It has also shown me that it is possible for me to work mostly independently on a project that interests me and to see the project through to the end. I already have another idea that I would like to work on next that is in a similar domain, which involves using machine learning to try to decipher what notes or chords are being played in a given song, something which could be useful when trying to learn how to play a song oneself on piano or guitar for example.

**Conclusion:**

I had hoped to find a company with which to do an internship as this would obviously have given me some real-world experience of programming and would have also been a valuable addition to my CV. However, I feel that I still learned a lot and found the experience to be a positive one. I feel more confident in working on a project all the way through from start to finish. While it was not perfect I feel that my project was of a decent standard and working on it has taught me a lot. Finally, I would like to thank my supervisor, Mehwish Tahir, for her kind encouragement and helpful advice throughout the course of the twelve weeks.

**References:**

<https://github.com/x4nth055/emotion-recognition-using-speech>

<https://github.com/MiteshPuthran/Speech-Emotion-Analyzer>

<https://superkogito.github.io/SER-datasets/>

[https://zenodo.org/record/1188976#.YrMcdOzMLIW](https://zenodo.org/record/1188976" \l ".YrMcdOzMLIW)

<https://machinelearningmastery.com/grid-search-hyperparameters-deep-learning-models-python-keras/>

<https://www.analyticsinsight.net/speech-emotion-recognition-ser-through-machine-learning/>

<https://blog.dataiku.com/speech-emotion-recognition-deep-learning>

<https://www.thepythoncode.com/article/building-a-speech-emotion-recognizer-using-sklearn>

<https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html>