University of Worcester

Evaluating Approaches to Speech Emotion Recognition Through an Investigative Deep Learning Project

Student
Thomas Elliott Mably

Supervisor
Pete Moody

Towards the Degree of
BSc (Hons) Computing

University of Worcester
May 2021

Abstract

The past decade has seen significant interest in the research of machine learning for speech processing. However, more recently, research trends in the field have moved towards Deep Learning (DL), a subset of machine learning often yielding far improved results for speech recognition compared to other approaches (Nassif et al. 2019). In this paper, we present an overview of existing Deep Learning systems for SER, perform a comparative evaluation on some of the world-leading literature in the area, and present a practical investigation into which deep learning approach may be most suitable for SER. Our investigation aims to consider the maximisation of accuracy with minimal data availability, providing insight into the feasibility of implementing deep learning systems for SER outside of the laboratory environment and with limited resources.

Preface

As an Artificial Intelligence enthusiast and prospective Computing graduate, diving into an advanced Machine Learning topic was profoundly interesting. With Pete Moody's guidance, honing my research has laid the foundations for the realisation of my ambition to continue onto a Masters degree. Working independently on a project over such a long timescale was a rewarding experience and a worthy test of resolve and an open-minded attitude to research. As artificial intelligence is a broad topic, it was rewarding to be immersed in a specific and cutting-edge area of the subject. It has introduced areas I might like to specialise within in the future.

Human-Computer Interaction has been a fascinating research topic throughout the past year. Bridging the gap between humans and computers stands to impact the broader context of computing profoundly. The abstraction of input devices away from the synthetic form, such as a keyboard, and facilitating interaction through the most natural forms of communication is sure to revolutionise how we utilise computer systems. Speech Emotion Recognition is an integral part of advancing this interaction. Evaluating the wealth of recent literature and learning how to implement cutting-edge technology as part of this paper was extremely rewarding.

The artefact has many dependencies (Tensorflow, Keras, Librosa, Numpy, Pydub…), making it challenging to run across computers. Furthermore, Python is an interpreted language, so we can't easily create an executable. Therefore, we have provided demonstrations of the artefact throughout the report. Please watch these videos should you require further explanation regarding the functionality of the artefact.

Acknowledgements

I want to thank my supervisor Pete Moody for his support throughout a challenging year. Furthermore, his words of encouragement, especially through the last couple of months where I have faced personal difficulty, have been vital to this project's success. Navigating a new topic whilst distance learning would not have been possible without his consistent guidance.

Table of Contents

1 Introduction

Speech Emotion Recognition (SER) is a challenging but essential step in optimising Human-Computer Interaction (HCI). After a storied two decades of research, SER has matured to the point where its application is considered the natural next step in bringing speech user interfaces into the commonplace. By enriching the next generation of Artificial Intelligence (AI) with the ability to interpret the emotion from spoken voice input, increased accuracy and levels of connection can be made between the intelligence and the human user (Schuller 2018).

1.1 Initial Literature Search and Criticisms

After an initial literature search on the topic of Human-Computer Interaction, it was apparent that a great deal of uncertainty exists regarding the methods of approaching Speech Emotion Recognition. In terms of implementation, the most recent papers on the subject almost exclusively leverage Deep Learning as the approach to classification. Although this has led to exciting results (Niu et al. 2017), Deep Learning is far more computationally intensive and requires far larger datasets to learn from than traditional machine learning approaches (Velardo, 2020). We aim to undertake a practical investigation into the efficacy of deep learning solutions with realistic resource and data availability and compare with secondary research using more traditional approaches to evaluate the feasibility of deep learning implementations.

There is also significant uncertainty in the theoretical sense, most notably regarding how a set of features can indicate an emotion, as emotive response varies wildly from person to person. The labelling of data for emotion recognition training as performed in traditional supervised machine learning workflows is problematic (Barrett et al. 2019). Ekman, whose work is continually referenced as 'proof' that emotional classification by features of expression is possible (Calder et al. 2001), has described 'most' of the research papers derived from his work as pseudoscience, citing insufficient care in the labelling of emotional responses (Murgia, 2021). Perhaps the difficulty in attaining verifiable labelling contributes to Deep Learning's ever-increasing popularity for audio classification. As Deep Learning models performs feature extraction without human input, labelling data features is no longer a concern. However, we must ensure that the data we provide to the model is indeed an accurate representation of a particular emotional state for correct classification. Else, the features extracted by the model may not be a valid indication of a specific emotional state. As part of our research, we aim to evaluate the theoretical basis of emotion recognition, drawing from some of the most influential literature on emotion classification, to conclude whether the discrete classification of emotions is possible for emotion-recognition systems.

Wijayasingha and Stankovic (2021), among other sources, point to significant hurdles facing the integration of SER systems into real-world scenarios. For example, most models proposed in research are trained with noiseless data. Thus, their success may be low when applied in real-world conditions, as noise is almost always present. As part of our investigation, we aim to discuss these hurdles and consider how we can apply corruption-minimising solutions to our models for confident deployment in the future.

1.2 Research Aims (and objectives)

It is essential to ensure that the aims of this project are of a suitable scope, narrow enough to be achievable with the knowledge and resources available, whilst providing enough material for scholarly investigation.

1.2.1 Research Aim One

Evaluate the most popular approaches for Speech Emotion Recognition with consideration of criticisms regarding their theoretical basis, robustness in real-world implementations and ethical concerns that may arise from their use.

1.2.1.1 Justification

Much of our evaluation of the architectures behind SER solutions will draw from secondary research. Here we can identify feasible approaches for assessment in our practical investigation whilst also considering the very best processes whose success would be impossible to replicate in our own implementations.

1.2.1.2 Objectives to meet the aim:

- Perform first search for sources for outline literature review.
- Compile reading list.
- Read resources in reading list.
- Write Literature Review.

1.2.2 Research Aim Two

Utilise industry best practices (Géron, 2019) to perform a successful end-to-end Deep Learning project, culminating in the production of accurate, performant Speech Emotion Recognition (SER) models for discrete classification.

1.2.2.1 Justification

The practical investigation will involve the implementation of two highly relevant approaches identified during literature review. By undertaking a practical study, we can gain insight into the difficulties in implementing deep learning solutions with a limited data set, produce models on which we can perform primary research to evaluate their efficacy, and gain a qualitative insight on the deep learning workflow.

1.2.2.2 Objectives to meet the aim:

- Take Course Deep Learning (for audio) with Python.
- Reflect on course, consider impact the project direction.
- Take Course Natural Language Processing with Deep Learning.
- Finalise Product Specification
- Pre-Processing.
- Source Data.
- Determine means of pre-processing.
- Pre-process data.
- Prepare the dataset.
- Implement the Neural Network
- Train Model.
- Solve Overfitting.
- Fine Tune Model.
- Analyse the accuracy of each model using unseen data.
- Determine the computational efficiency of each approach and justify whether any increases in accuracy are worth compromises in computational requirements.

1.3 Research Questions

As research questions are integral in guiding the methodology of a project, the research question must point to a need for deliberate investigation (Ratan, Anand and Ratan 2019). As such, drawing from the project aims, we will practically investigate the implementation of a Deep Learning system for SER outside of the laboratory environment, seeking to answer the following research questions:

Can emotions be accurately classified into discrete categories?

Are Deep Learning solutions for SER feasible for real-world implementation in scenarios with limited computational and data resources?

What challenges faced the implementation of SER into existing systems?

1.4 Research Significance

Deep Learning has emerged as a revolutionary solution to problems across the field of artificial intelligence. Its application to SER has been subject to significant research in recent times. Whilst this is an exciting prospect, the relative youth of this topic has resulted in uncertainty regarding best practices, along with approach validity and verifiability. By performing an end-to-end practical investigation, we hope to evaluate Deep learning for SER whilst highlighting further uncertainties that require research in future work.

2 Literature Review

2.1 Human-Computer Interaction (HCI)

As technology becomes ever more pervasive in our lives, there is a growing need for accurate and efficient interaction with computer systems. Human-Computer Interaction (HCI) is an interdisciplinary field involving the design and implementation of such interactions (Ozseven 2019). Ideally, a computer system would facilitate natural interaction with humans, such as through direct vocal interaction, rather than traditional input devices such as keyboards (Interaction Design Foundation, Unknown). In recent times, Human-Computer Interaction has grown from its focus on individual user behaviour to include the use of technology in networked communication systems (UCSB Social Computing Group, 2020), accessibility concerns, and the study of the interaction between computer systems and many users (crowds).

2.2 Automatic Emotion Recognition (AER)

Automatic Emotion Recognition (AER) systems for the automatic recognition of an emotional state by analysing input signals have become an essential component of HCI (Khalil et al. 2019). AER systems can be grounded in different modalities, such as visual and auditory, or use a combination of several (Sapiński et al. 2019). Facial Emotion Recognition (FER) systems, the most common optical AER systems, use cameras to classify emotional state based on the visible aspects of human emotions, such as micro-expressions or familiar cues such as smiling. In contrast, Speech Emotion Recognition (SER) systems utilise microphones to capture speech signals to classify emotional state by pattern recognition (Wijayasingha and Stankovic 2021). While AER benefits most existing speech and facial recognition systems, it has critical applications in social security, safe driving, and health care (Shu et al. 2018).

2.3 Issues with Emotion Classification theory

There are two fundamental approaches for emotion classification, those focusing on discrete classification into a fixed number of emotions and those that consider the continuous nature of emotions discerned with consideration to speech's temporal features, such as the valence and potency of delivery (Khalil et al. 2019). Human emotions are inherently ambiguous and impure (Mao, Ching and Lee 2020). Some papers give little reference to the disputed notion in psychology that emotions can be classified based on measurable features. Particularly in Deep Learning applications, where feature classification is more of a more automated process as the implementation spots patterns, it might be taken as a given that emotion is indeed classifiable. However, various literature reviews include the theoretical basis of SER as an ongoing discussion that should directly inform research in the years to come (Khalil et al., 2019; Schuller, 2018). Much of the current literature mistakenly takes decades old and oft-questioned (Cherry 2017) psychological studies of the classification of emotion as fact (Barrett et al. 2019). A review of the challenges surrounding Facial Emotion Recognition (FER) systems found that it is not possible to confidently infer emotion from a given facial cue (such as from a frown or smile) (Shu et al. 2018).

Mao, Ching and Lee (2020) present an alternative to the discrete labelling employed by many speech recognition systems, instead offering a means of attributing a time series of soft labels to capture the blends of emotional cues and build a profile of an emotional state. There is clear merit to this system, but it is too complex for implementation within this project.

2.3.1 Cultural differences

While cultural differences in emotional perception have been reported for facial and emotional expressions, combining the two into a single emotion detection system also presents unique challenges. How a perceiver combines auditory and visual cues has been subject to cultural variability. A study demonstrated that Japanese participants paid more attention to vocal cues than Dutch participants, whose basis of emotional perception was skewed toward facial cues (Tanaka et al. 2010). Furthermore, there are also discrepancies in emotion intensity, proven especially apparent in the direct comparison of eastern and western cultures (Lim 2016).

2.3.2 Ethical Issues

As with all AI projects that have the potential to impact our interaction with technology, it is essential to be cautiously aware of the near-term adverse effects of its implementation, particularly where there are applications for surveillance (Amnesty International, 2021). As is evidenced in Oxford Union's debate regarding the ethical standing of facial recognition technology, the negative impacts of the advancement of SER could disproportionately and problematically affect individuals depending on their ethnicity (Oxford Union 2019). This is particularly pressing as our solution focuses on emotion, which is loosely defined across cultures (see 2.3.1).

Facial Emotion Recognition (FER) systems have recently been adopted into surveillance of public spaces, with funds for FER systems capable of 'identifying suspicious people' allocated by Lincolnshire Police to significant criticism (Murgia, 2021). It is entirely plausible that SER systems will require the same ethical considerations.

2.4 Approaches to Audio Processing

A range of techniques have been employed for the task of Emotion Recognition; from traditional Artificial Intelligence (AI) workflows, such as Expert System implementations, to the cutting-edge Deep Learning approaches. Machine Learning (ML) is often used interchangeably with AI, likely because of its increased relevancy whilst traditional AI systems tend towards obsolescence. However, Machine learning is a subset of Artificial intelligence (Figure 1). Deep Learning is a further subset of Machine Learning, whose focus shifts from manual feature extraction to an automated process driven by intensive analysis of a data set.

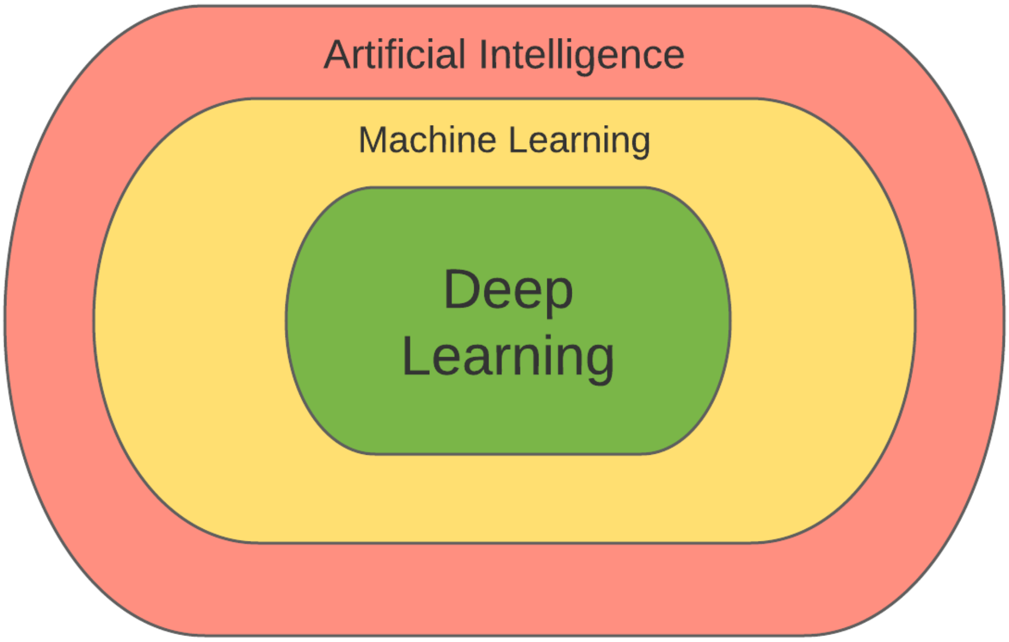


Figure 1. A model of Deep Learning and Machine Learning as subsets of artificial intelligence

2.4.1 Traditional Artificial Intelligence Systems vs Machine Learning for Audio Processing

This example (Velardo, 2020) will examine two solutions for onset detection or where an audio utterance begins. First, we will create a solution using an Expert System, where the AI is given a set of rules, rather than coming to these rules on its own accord.

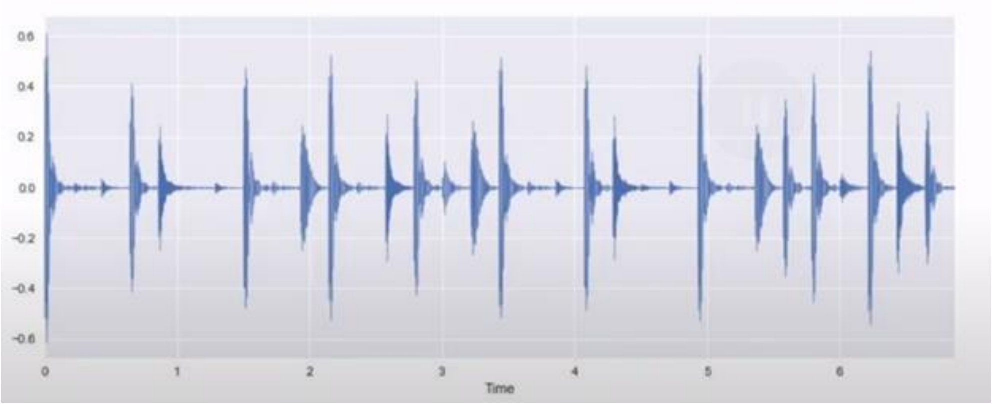


Figure 2. (Valerdo, 2020) An unlabelled waveform

Looking to Figure 2, where there is a burst of amplitude, there is an onset. To solve this problem with an expert system, we would define specific rules into the system to identify an onset, for example, where the amplitude exceeds the bounds of 0.2 or -0.2. Figure 3 shows this rule superimposed onto the waveform.

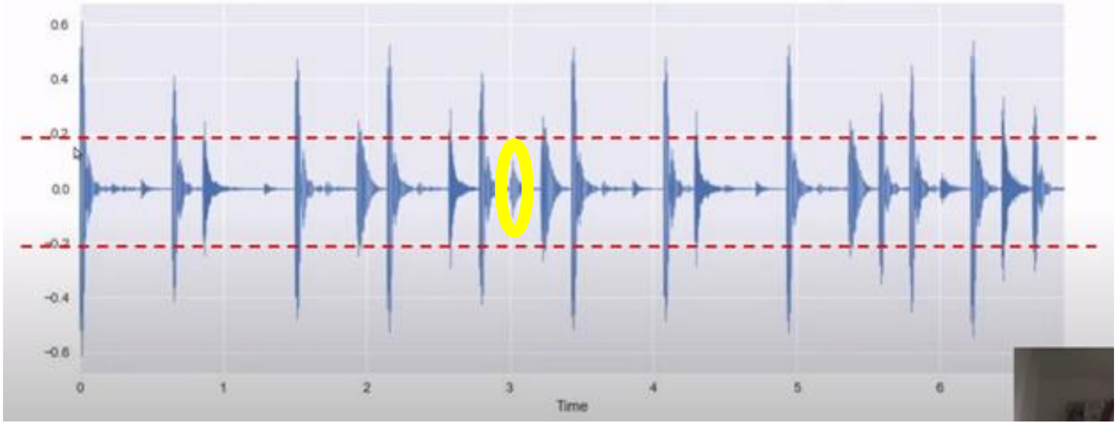


Figure 3. (Valerdo, 2020) A waveform to be bounded by amplitude values -0.2 - 0.2. Waves outside of these values are expected to indicate the onset of a note.

Figure 3 evidences that an expert system with a singular rule would have missed a minor outburst of energy (circled). Granted, this is an extreme simplification of an expert system; whose rule sets are usually far more extensive. To rectify this error, we would make changes to our ruleset. As such, the implementation of artificial intelligence projects is agile in nature, with continual prototyping and adjustments taking the place of coordinated testing (Figure 4).

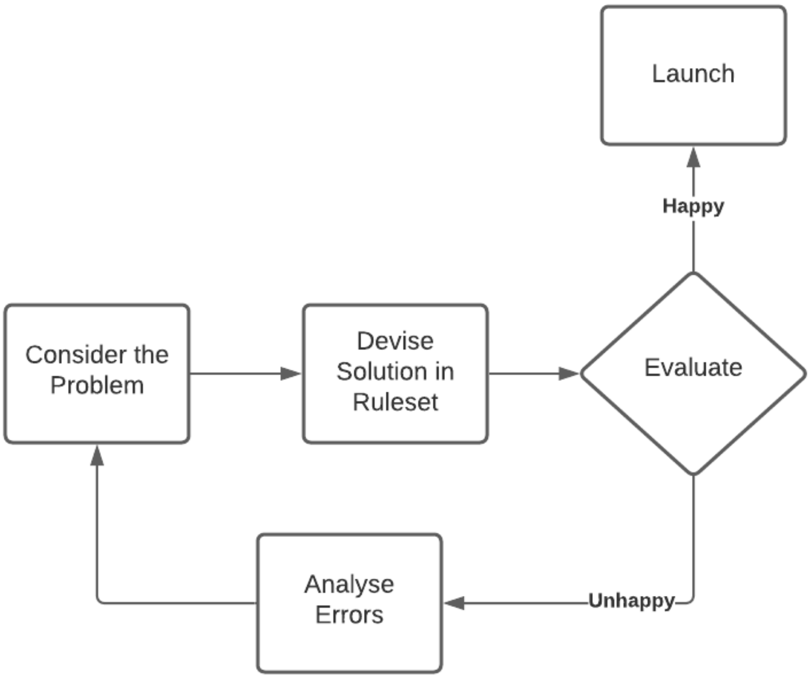


Figure 4. The workflow for traditional artificial intelligence solutions.

2.4.2 Machine Learning Systems

If we were to approach the same problem with Machine Learning, we would work backwards from the labelled data rather than immediately design something for data labelling. As shown in Figure 5, a machine learning algorithm will automatically derive the ruleset by analysing pre-processed (labelled) data (IBM Cloud Education, 2020). Therefore the ruleset is not directly weighted by human influence, so long as the data provided is labelled accurately.

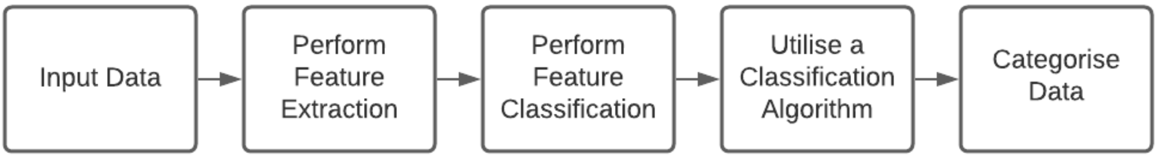

Input Data
Perform Feature Extraction
Perform Feature Classification
Utilise a Classification Algorithm
Categorise Data

Figure 5. The workflow of traditional machine learning solutions

2.4.3 Deep Learning Systems

Artificial Neural Networks (ANNs) are machine learning models inspired by the networks of biological neurons found in the human brain. However, ANNs have gradually become quite different from their biological inspiration. ANNs are versatile, robust and scalable, making them ideal for large and complex tasks, such as classifying vast datasets. As data becomes more abundant, ANNs are becoming more frequently favoured over other ML approaches due to their scalability (Geron, 2019). In ANNs such as the Multi-Layer Perceptron, backpropagation allows the adjustment of neuron weights dependent on minimising error. Essentially, an ANN can adjust its values to achieve the optimal classification of input data. Training refers to continual minute adjustments over many passes of a dataset. As shown in Figure 6, this removes the need for feature extraction and classification by human means.

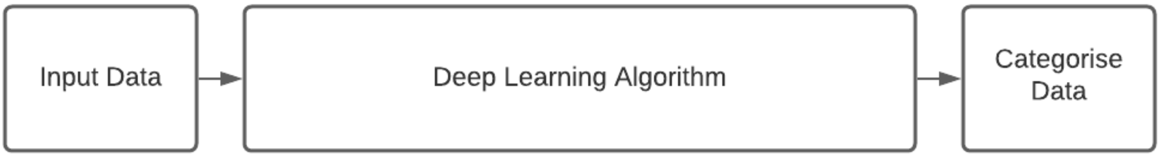


Figure 6. The workflow of Deep Learning solutions.

2.4.3.1 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) have proven highly effective for the classification of images (Qin et al. 2020)but also show promise for audio-classification tasks (Hershey et al., 2017). Lei writes that, whilst CNNs are effective for classification, their implementations are computationally intensive (Lei, Pan and Huang 2019). However, evidence that CNN implementations are feasible for systems with limited computational capability, with end-to-end audio classification CNNs presented with low computational complexity when compared with other Deep Learning architectures (Huang and Leanos, 2018). Furthermore, Hershey et al. (2017) offer a highly scalable solution capable of classifying audio datasets with a cumulative length of 5.24 million hours.

CNNs are particularly effective when applied to datasets whose features can be represented as a multi-dimensional array. A dataset of images or stacks of images will likely be of this format, which explains CNNs consistent use in image-classification systems. A Fast Fourier Transform (FFT) can be performed on digital audio to spectrally describe the data, which can then be presented on a spectrogram (Figure 7).

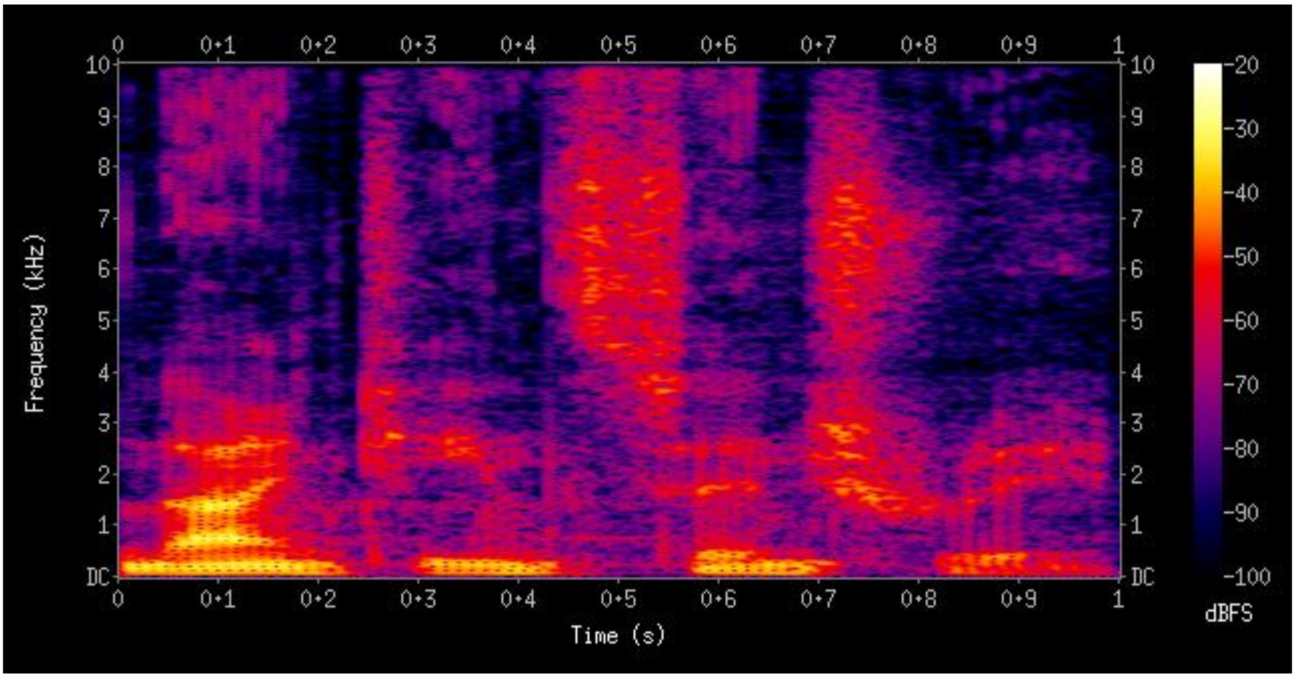


Figure 7. Spectrogram of the spoken words "nineteenth century" (Aquegg, 2008).

Magnitude Spectrograms, essentially an image of sound's intensity at given frequencies, are stored as multidimensional arrays. As such, they are appropriate learning data for CNNs. (Wijayasingha and Stankovic, 2021).

2.4.3.2 Long Short-Term Memory (LSTM) Networks

LSTM based audio processing methods have been shown to offer state of the art performance for various speech recognition tasks (Li and Wu, 2015). LSTM has been proven highly effective for SER specifically, though these systems utilise additional features such as attention mechanisms in the vast majority of cases. We will hope to evaluate the performance of LTSM for SER without these other features and compare the results with the more often utilised CNN.

2.5 Limitations Common to Deep Learning Implementations

State-of-the-art deep learning implementations, which achieve an excellent level of accuracy, typically start processing audio at the end of an utterance. Intrinsically, a mechanism for the detection of speech onset and conclusion complicates the SER system. More significantly, waiting until the end of an utterance to begin processing poses issues for real-time application of the model. EmoRL, a model for the continuous detection of emotion, will utilise a degree of confidence to determine when to start processing speech, achieving a similar level of accuracy with lower latency (Lakomkin et al., 2018).

A spectrogram's size is relative to its time interval. As utterances are of variable length, this poses a problem to some Deep Learning architectures, such as CNNs, that cannot accept input data of varying size. RNNs can take inputs of variate size and have been combined with CNN systems to circumvent the issue (Zhang et al., 2019) Zhang *et al.*'s two-dimensional architecture also outperformed regular CNN benchmarks. Thus, hybrid architectures are becoming increasingly common in contributions to the area (van Houdt, Mosquera and Nápoles, 2020).

Ensuring the quality of training data is crucial in producing a successful audio classification model. Ambient noise during the recording process (see Figure 8) has the potential to impair the accuracy of feature extraction and classification. Therefore, it is crucial to minimise unnecessary variation in training data during pre-processing, whilst ensuring features indicative of emotional discrimination remain intact (Vogt, André and Wagner, 2008).

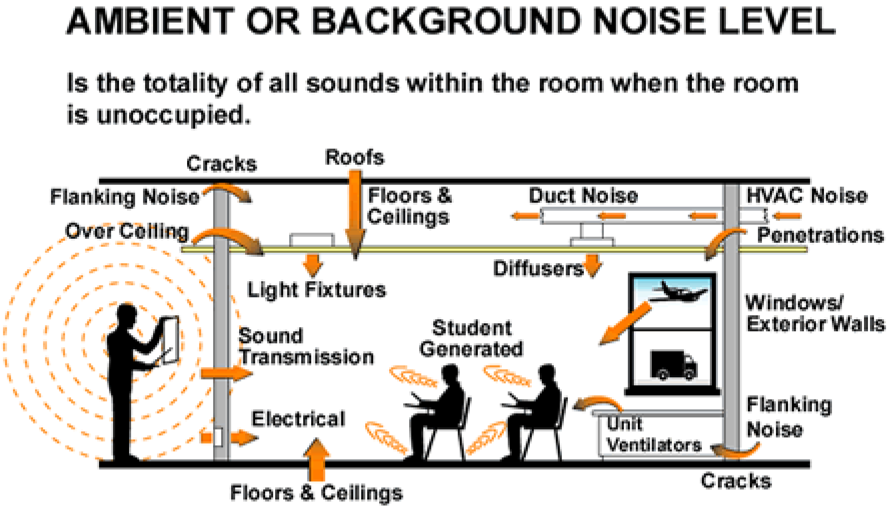


Figure 8. A diagram of ambient noise (Acoustical Services, Unknown)

3 Methodology/Sources of data

3.1 Procedure

Geron (2020) describes a framework, mainly focused on ensuring data quality, for Design-Build-Test projects leveraging machine learning. Specifically, as we are utilising Deep Learning, there is a need for immaculate data practices. The most significant challenge of design and implementation will be the pre-processing of audio data. In the first instance, we will have to determine what type of data will be used to train the model. Secondly, we will need to create a scalable solution that can convert raw audio data into this desired format. The significant dataset that we will utilise poses a considerable challenge; we must be sure to approach its management in a considered way.

3.2 Requirements Elicitation

To prioritise functional requirements, we will utilise the MoSCoW process, categorising the needs of the practical investigation. Performing this step ensures that we know what is essential to achieving the project aims and answering the research questions.

3.2.1 Functional Requirements

These are the requirements of the investigation that directly impact its function.

3.2.1.1 Must-Have

- We must implement two or more models for SER so that we can perform a comparison.
- We must pre-process a dataset into a datatype suitable for classification.
- We must pre-process data into elements of uniform length for application to all Deep Learning architectures.
- We must take a metric of the accuracy of each model for comparison.
- We must test each model using an unseen testing set to achieve a fair result for its accuracy.

3.2.1.2 Should-Have

- We should fine-tune each model to achieve the best result for each architectural implementation.
- We should take metrics of the build time and efficiency of each model.

3.2.1.3 Could Have

- We could provide a means of taking real-time input for the model.
- We could deploy our model as a web application/RESTful API.
- We could incorporate additional datasets to improve the accuracy of the model.
- We could investigate how noise affects model accuracy.

3.2.1.4 Won't-Have (due to a lack of expertise)

- We will not take composite inputs (such as both MFCCs and Group Delay Spectrograms)

3.2.2 Non-Functional Requirements

No.	Category	Non-Functional Req.	Priority
1	Performance	The performance of the models should be such that they could be applied to scenarios requiring low latency if it is indeed possible to achieve this with the deep learning architecture used.	High
2	Scalability	The models should be utterly scalable to process datasets of enormous size. This is because deep learning data sets are typically immense. To achieve scalability, we need to automate the means of detecting both data and the categories for classification. There can be no hardcoding of parameters regarding data input.	Medium

3.3 Types of Data source

Khalil *et al.* (2019) describe data sources for emotion recognition modelling as one of three types.

3.3.1 Simulated databases

Experienced performers have provided the speech data. Among all database types, simulated databases are the simplest to obtain, as there are no ethical concerns or ambiguity as to which emotion the actor is *trying* to convey. That said, the feelings expressed in this speech data are not "true". There is a possibility that the performer's delivery may be archetypal to how they have been taught to convey emotion, which may be unrealistic compared to natural emotion.

3.3.2 Induced databases

Participants unknowingly enter an artificial situation designed to elicit a specific emotional response. Whilst this gives a natural reaction, the deception of subjects presents an ethical issue. As such, these data sources are controversial and less common than simulated databases.

3.3.3 Natural databases

Data taken from organic conversations (such as at call centres) is classified into discrete categories. However, these are difficult to produce as categorising data correctly and verifiably is problematic (see section 2.3).

3.4 Dataset Choice and Justification

3.4.1 RAVDESS

The Ryerson Audio-Visual Database of Emotional Speech and Song (Livingstone and Russo, 2018) provides data of 24 actors (12 male, 12 female) vocalising "two lexically-matched statements in a neutral North American accent." The actors offer line reading with calm, happy, sad, angry, fearful, surprise, disgusted expressions, two degrees of emotional strength, one akin to everyday conversation, and the other exaggerated. Two hundred forty-seven untrained individuals have verified this data conforms with the categories given.

3.4.2 TESS

The Toronto Emotional Speech Set (Dupuis and Pichora-Fuller, 2010) provides a set of 200 target words spoken by two actresses (aged 26 and 64 years). Each actress delivered a target word in seven emotions (anger, disgust, fear, happiness, pleasant surprise, sadness, and neutral).

3.4.3 Evaluation of Data Availability

Over 3000 audio files of varying duration are available through the combination of the RAVDESS and TESS datasets. Assuming the average length of a track is three seconds, each audio file will then be cut into 0.5-second segments, resulting in 18,000 (3000*(3/0.5)) audio files. Each audio file will produce spectrograms of 13 MFCC coefficients. As such, there should be sufficient data to train the model.

Both the TESS and RAVDESS datasets are simulated datasets. Whilst this ensures no ethical concerns, this does bring into question the trustworthiness of the data's representation of an emotional state. However, both datasets have received significant external verification, which was my primary reason for selecting these data sets. Furthermore, the data will be simple to collate, as several classifications of emotion appear in both datasets.

3.5 Testing

The only way to know how well a machine learning model generalises to new data is to apply it to an unseen data set (Géron, 2019). As previously discussed, data with a proven emotional context is scarce. Therefore, we will hold back 20% of the training data set for testing, provide us with adequate data to test the model's accuracy whilst ensuring that the quality of the data used remains consistent.

To measure the system's performance, we will be using the Root Mean Square Error (RSME) calculation. The use of this performance measure is typical for regression problems (Géron, 2019). RSME uses weighted errors to approximate how much error a system makes in its prediction, with higher weight allocated to more significant errors.

3.6 Technologies Used

The development environment we will be utilising for implementation is Visual Studio Code (Microsoft, 2021) because of its direct integration with the open-source package manager and development environment Anaconda (Anaconda, 2021). The program for pre-processing will be implemented in Python, making use of the Librosa library (McFee *et al.*, 2015) for processing the audio signal. We will use Scikit-Learn (Scikit, 2021) to manage the dataset and split the resource into a training and testing set. TensorFlow (TensorFlow, 2021), and its high-level implementation Keras, will be used to implement the model designs and perform training and testing.

Python is an industry-standard programming language for developing AI solutions, mainly due to its readability and extensive libraries (Costa, 2020). Tensorflow is an end-to-end open-source machine learning platform, leveraged as a library within Python. It is the standard platform when implementing deep learning solutions with a wealth of documentation available considering the youth of the research area.

The program will be stored and executed locally instead of using workbooks on a cloud-hosted service such as Jupyter. Thereby, we can have complete control over the capture of efficiency metrics and draw more realistic conclusions regarding the feasibility of running such a model on limited hardware.

4 Design

4.1 Framing the Problem With a Detailed Specification

Geron (2019) suggests that the first step in designing a machine learning solution is to frame the problem within the context of the broader information on the subject. Doing so allows us to achieve a more focused specification of what we would like our solution to achieve.

4.1.1 Problem Analysis

For the pre-processing of data, we must find a means of processing the data that requires minimal manual input. Not only will this save time during this development, but it is also a non-functional requirement that the solution is highly scalable. Most Deep Learning implementations utilise datasets far more extensive than even the 18,000+ MFCC spectrograms that will be used for training our system. By selecting two segments per one-second clip duration, we will average approximately two thousand spectrogram images per classification, far higher than the one thousand per class suggested in Mitsa's (2019) rule of thumb for data required vs model success.

We must implement a variety of models to produce a comparison of their accuracies. It makes sense to include a baseline implementation, something with a storied history for success but is not necessarily the best method to compare with the other deployments. The Multilayer perceptron is an excellent choice of architecture for this purpose, containing only a weighted network of fully connected neurons. Following this, we will test two further, recently more utilised, architectures for the problem; CNNs and LSTM-RNN networks.

Once we have built these models, we need to save and load the models to avoid repeatedly compiling and training them. To do this, we can utilise TensorFlow's load/save functionality. This once again fits in with the non-functional requirement of scalability, as some models build times are days long rather than minutes, as ours should be.

4.2 Methodology of Pre-Processing

As mentioned previously, the structure for my project design will follow Geron's (2019) framework for an end-to-end Deep Learning project. This approach mainly focuses on ensuring data quality during pre-processing and thus the validity of the model. To ensure data quality, we should perform the visualisation of the data to be fed to the model and carefully plot out how we will achieve an optimal dataset for training our solution.

4.2.1 Ensuring the Uniformity of Data Through Visualisation

Deep Learning architectures often require input data to be of uniform shape (Muccino, 2020). The length of an MFCC, our proposed input data medium, is dependent on time. As the audio samples provided in the RAVEDESS and TESS datasets vary, we must standardise the duration to ensure uniformity for training on our models.

There are two common approaches for achieving uniformity; padding the data out with dummy information that will have minimal effect during training or cutting the data to specific lengths (Brownlee, 2019). In our case, we are going to use a combination of these methods. Using the Pydub library (Jiaaro, 2021), we can 'chunk' clips into one-second segments, with a singular shorter chunk being taken from the end of the file. This shorter clip will then be brought up to the one-second duration by adding a period of silence at the end. Figure 9 shows a visualisation of this process. By processing the raw data this way, we ensure uniformity whilst minimising the loss of valuable training data.

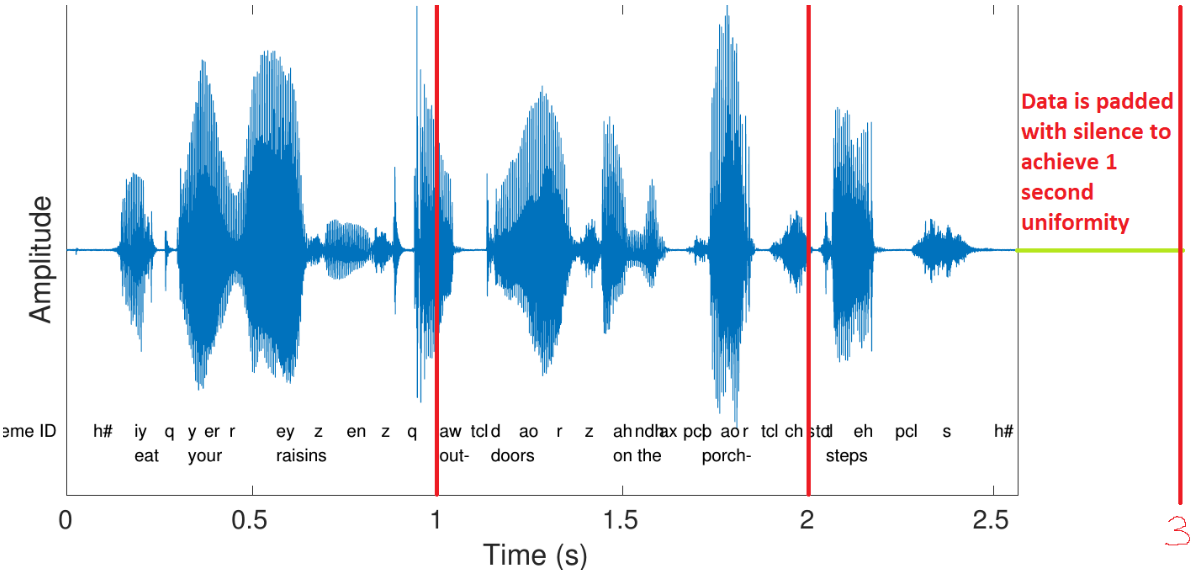
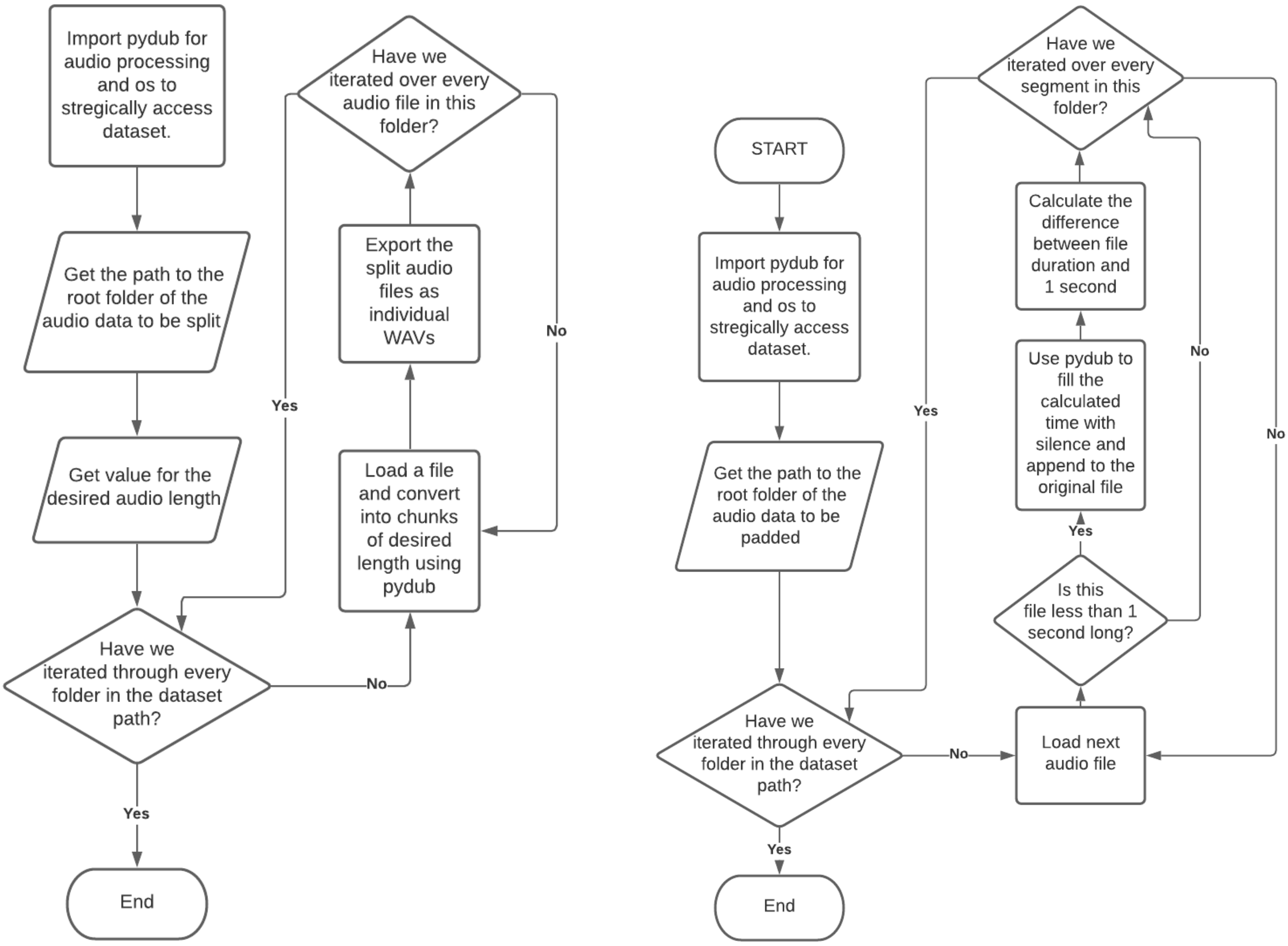


Figure 9. A visualisation of the 'chunking' and padding processes. Original waveform image courtesy of Aalto University (2020)

4.2.1.1 Flowcharts of proposed cut/pad script design


Import pydub for audio processing and os to stregically access dataset.
Get the path to the root folder of the audio data to be split
Get value for the desired audio length
Have we iterated through every folder in the dataset path?
Yes
End
No
Load a file and convert into chunks of desired length using pydub
Export the split audio files as individual WAVs
Have we iterated over every audio file in this folder?
Yes
No
START
Import pydub for audio processing and os to stregically access dataset.
Get the path to the root folder of the audio data to be padded
Have we iterated through every folder in the dataset path?
Yes
End
No
Load next audio file
Is this file less than 1 second long?
Yes
Use pydub to fill the calculated time with silence and append to the original file
Calculate the difference between file duration and 1 second
Have we iterated over every segment in this folder?
Yes
No
No

4.3 Extracting MFCCs from data

4.3.1 Waveform sampling

We will perform waveform sampling of audio input using the Librosa library, providing a plot of the waveform data at a given sample rate (Figure 10).

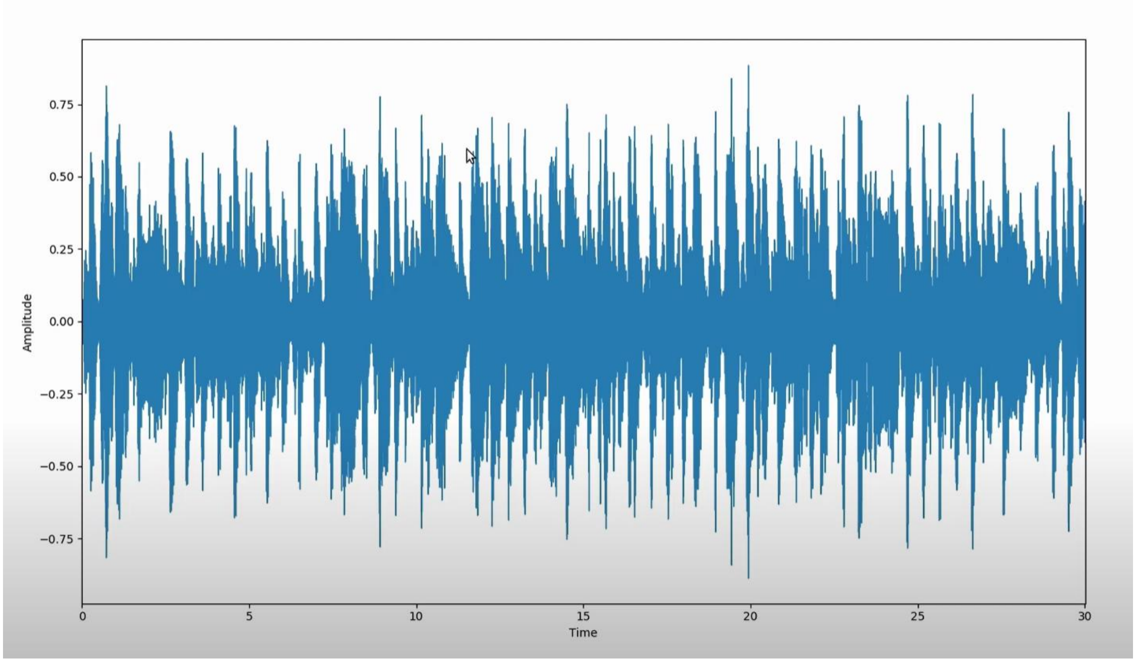


Figure 10. A waveform as sampled by the Libroasa Library

4.3.2 Spectral analysis

Performing a Fast Fourier Transform on the waveform data allows us to decompose the data into a form dependent on the temporal frequency. To accomplish this transformation, we will utilise the NumPy library function fft (Numpy, 2021). Here, we pass waveform, in the form of an array, into the Fourier transform function. Once transformed, we take the absolute value of each data point, resulting in a plot of magnitude. These magnitude values will indicate the contribution of each frequency to the overall sound of the data.

4.3.3 MFCCs

Taking a magnitude spectrogram, we perform a final transformation using the Librosa library. The result is an MFCC spectrogram (Figure 6), representing the timbre of sound as a spectrogram. These are multivariate time series; thus, they can be classified well using a Convolutional Neural Network (CNN).

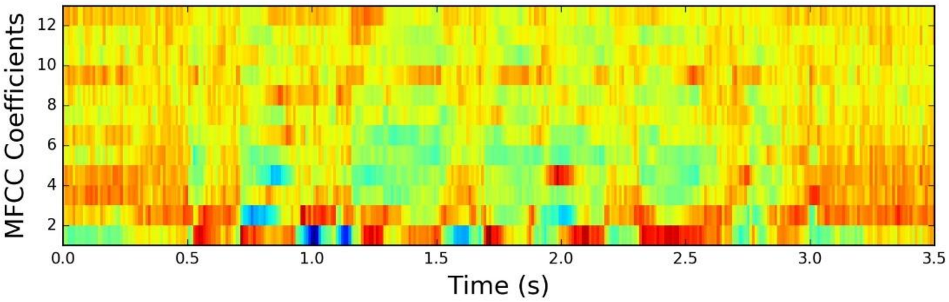
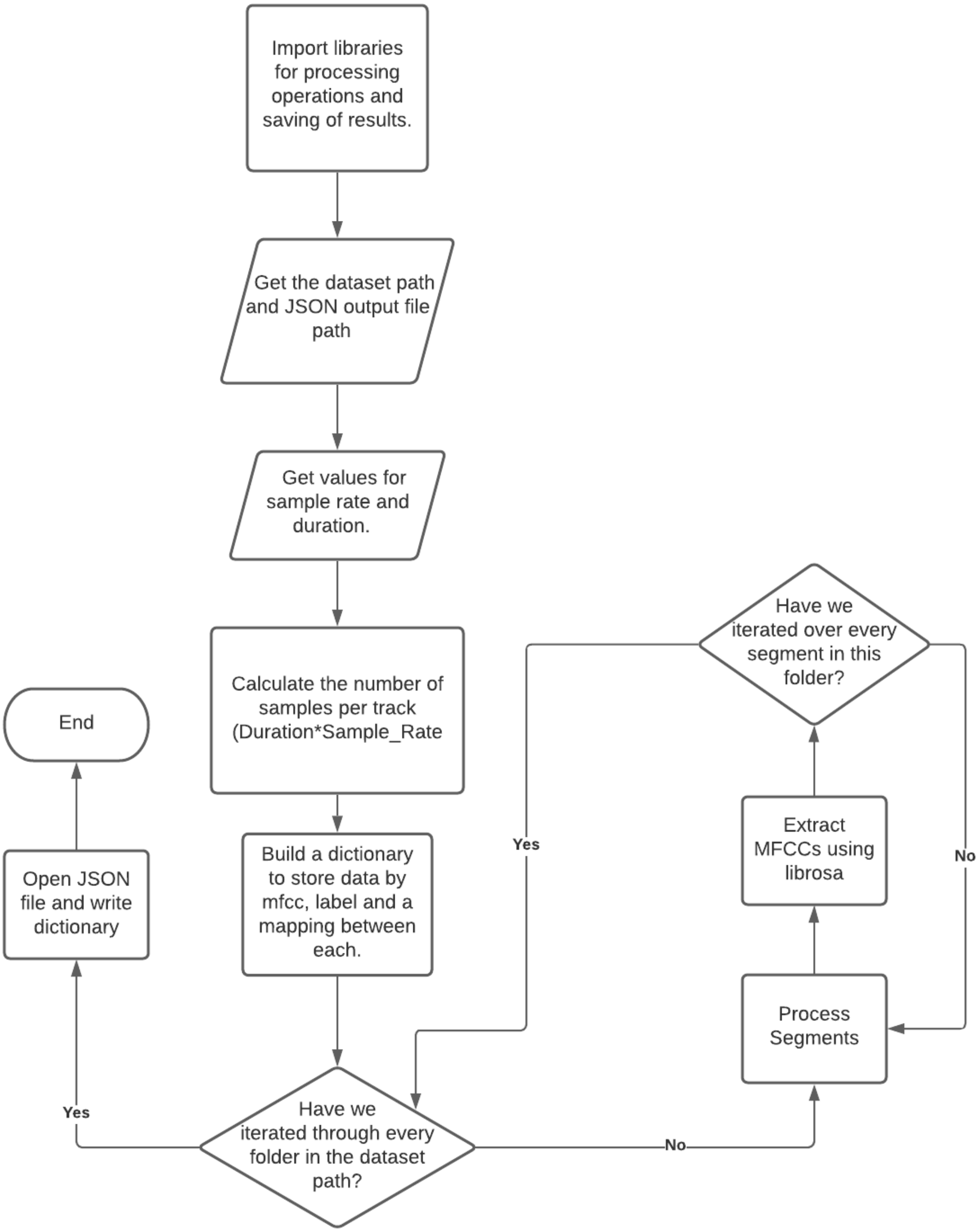


Figure 11. A Mel-Frequency spectrogram plotting MFCCs (Fayek, 2016).

4.3.4 Flowchart of the Proposed Script

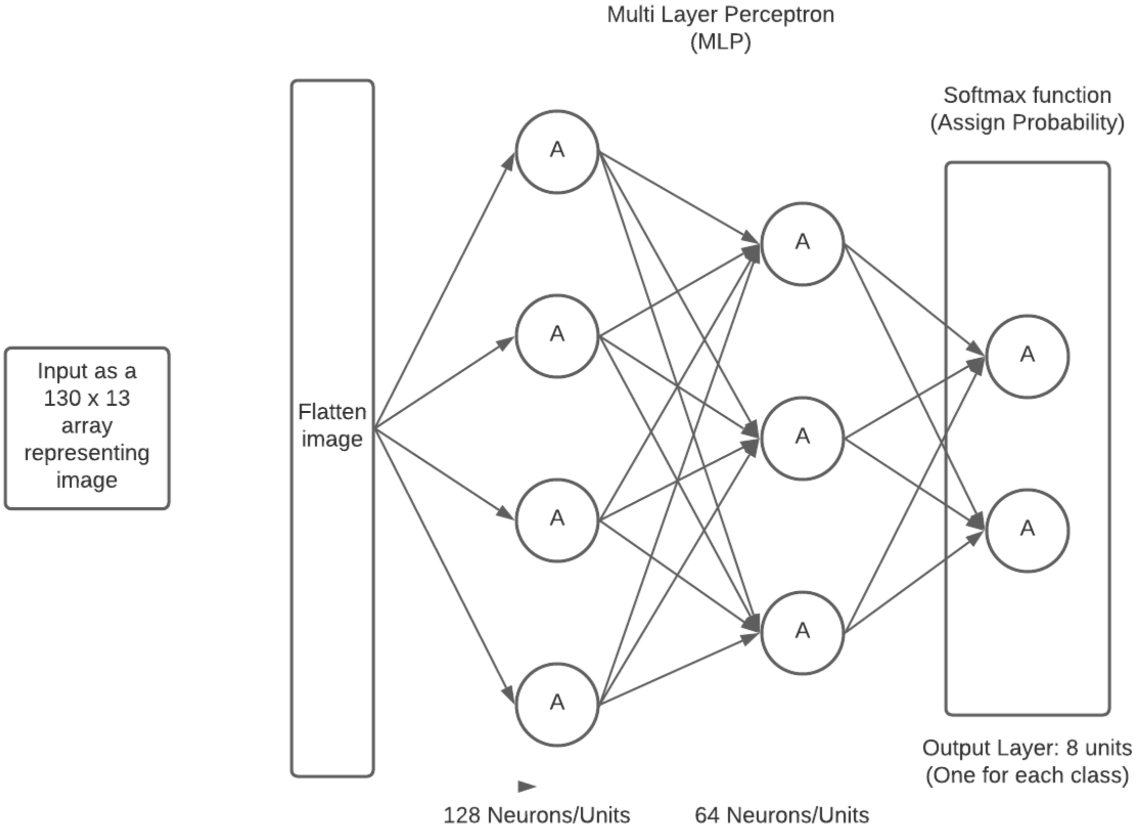

Import libraries for processing operations and saving of results.
Get the dataset path and JSON output file path
Get values for sample rate and duration.
Calculate the number of samples per track (Duration*Sample_Rate
Build a dictionary to store data by mfcc, label and a mapping between each.
Have we iterated through every folder in the dataset path?
Yes
No
Open JSON file and write dictionary
End
Process Segments
Extract MFCCs using librosa
Have we iterated over every segment in this folder?
Yes
No

4.4 Methodology of Model Implementation

Most of the implementation phase involves collating, sorting and pre-processing a complex dataset. Once this has been achieved, there is only a need to implement well-documented models for neural networks whilst justifying their parameters (for example, the number of neurons per layer and which solver is used). We identified two neural network architectures that were feasible to implement for our SER system during the literature review, the Convolutional Neural Network (CNN) and the Long Short-Term Memory (LSTM) network. Significant research evidence suggests that these methods are highly suited to spectrogram classification (Hershey et al., 2017; Huang and Leanos, 2018; Wijayasingha and Stankovic, 2021). Therefore, we will implement solutions using these architectures, comparing their results with one another, along with baseline results from a Multilayer Perceptron implementation.

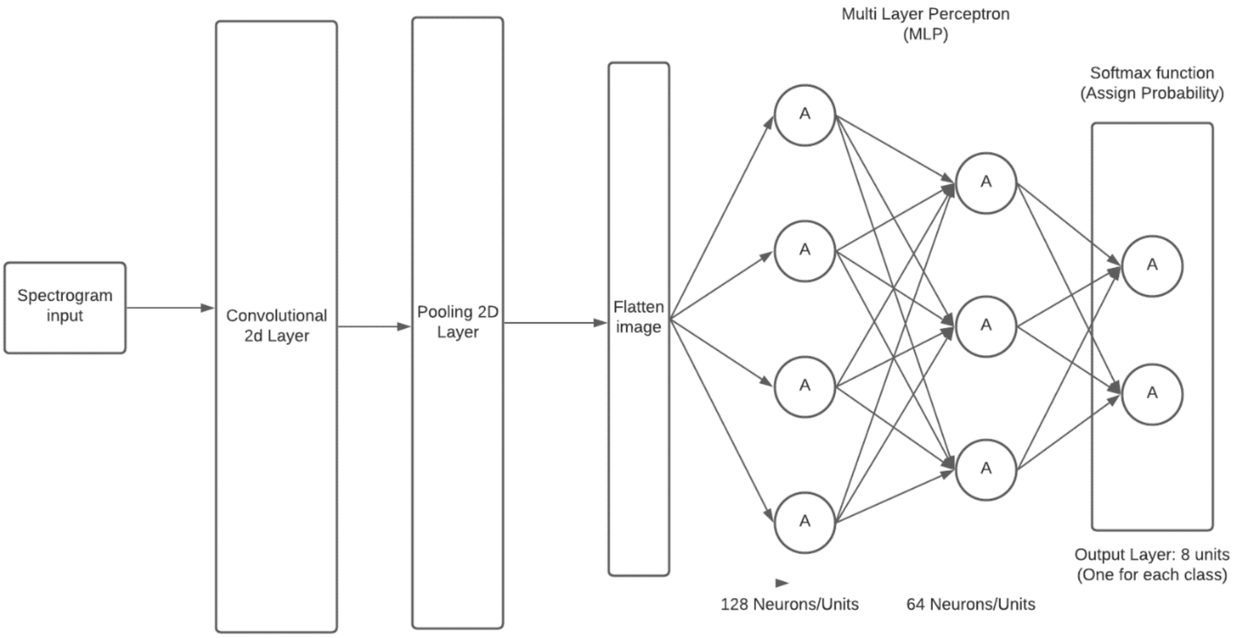
Following the guidance of Ramesh (2018) we will create a basic implementation of each architecture, then tune our parameters based on observations during trial-and-error training.

4.4.1 Designing our Multi-Layer Perceptron



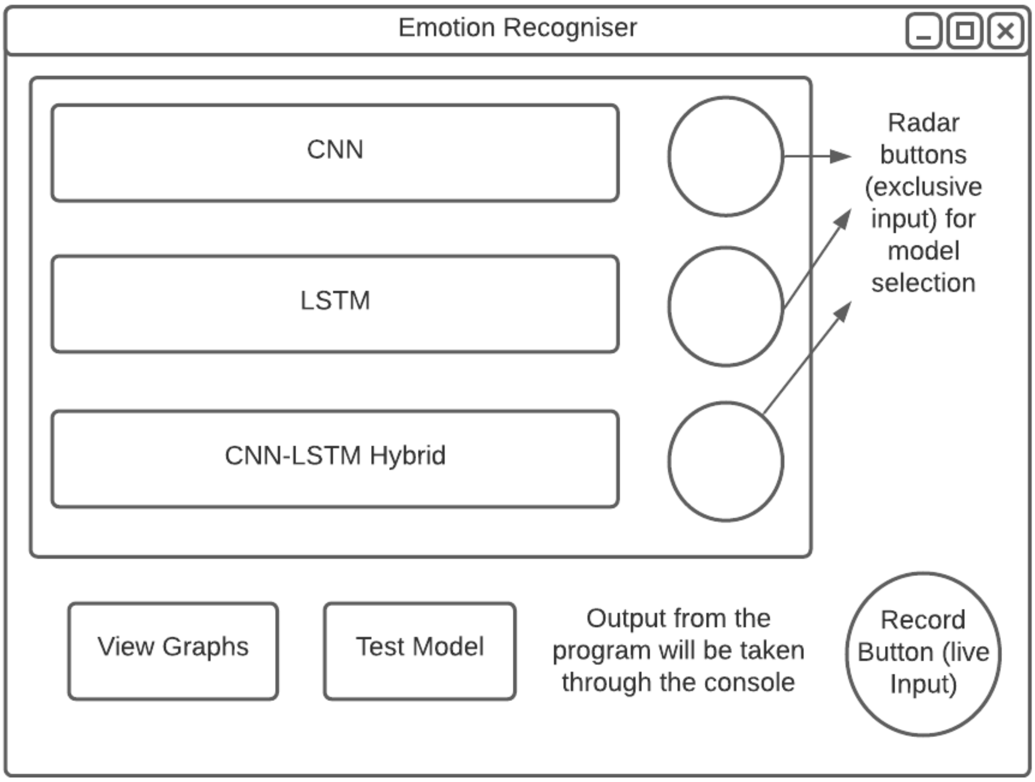
4.4.2 Designing our CNN

The basic design of the CNN will be three convolutional layers, with max-pooling and batch normalisation. The initial convolutional layers are used for rich feature extraction. The convolutional layer strategically applies learned filters (whose weights are continually adjusted during training) to input images, creating feature maps that summarise input features. Pooling is used to improve the invariability of the feature map when introduced to features that are slightly translated compared to the learned features of the map, making it more robust (Brownlee, 2019). These extracted features are then passed into an MLP for classification.


Spectrogram input
Convolutional 2d Layer
Pooling 2D Layer
Flatten image
Multi Layer Perceptron (MLP)
Softmax function (Assign Probability)
A
128 Neurons/Units
64 Neurons/Units
Output Layer: 8 units (One for each class)

4.5 Program for loading and testing saved models

As part of the ‘could-have’ specification, we included a possibility of taking audio input and allowing the model to predict this input. The following details a design for the front-end, should the need arise for one. The most important aspect of the project is ensuring the success of the models, but the front end would be useful for demonstration purposes.


Emotion Recogniser
CNN
LSTM
CNN-LSTM Hybrid
Radar buttons (exclusive input) for model selection
View Graphs
Test Model
Output from the program will be taken through the console
Record Button (live Input)

4.5.1 Model Selection Pseudocode:

```
SWITCH
 CASE button = CNN
      model = CNN.h5
 CASE button = LSTM
```

```
      model = LSTM.h5
 CASE button = CNN-LSTM
      model = CNN-LSTM.h5
```

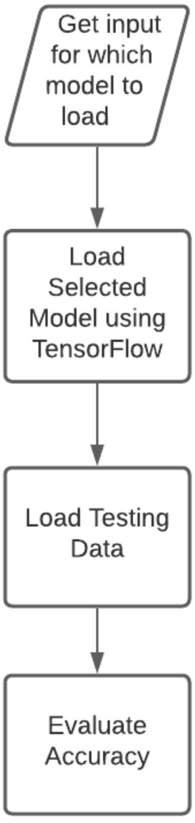
4.5.2 ‘Test Model’ button Pseudocode:

```
ON TEST BUTTON CLICK
 output = model.predict(TestData)
PRINT output
```

4.5.3 Record Button Pseudocode:

```
IMPORT audiolibrary
SET SAMPLE RATE
ON RECORD BUTTON CLICK

 input = audiolibrary.record()
 split(input)
 pad(input)
 recorded = librosa.mfcc(input)
 model.predict(recorded)
```

4.5.4 Flowchart


Get input for which model to load
Load Selected Model using TensorFlow
Load Testing Data
Evaluate Accuracy

4.6 Justification of design

As the Artefact is simply a component of a practical investigation regarding the backend model, there is no need for advanced features such as direct input through a microphone or a GUI. However, including these, if there is time, would be welcome. The Python console will supply the required output functionality for insights into the model performance. Matplotlib, a python library used for the plotting of data, will also be used to gain insight into a model’s training performance by plotting the error against the number of epochs for both the training and testing set.

5 Implementation

Some code (TM_AudioPreProcessor.py and CNN_emotion_classifier) has been adapted from Velardo's course Deep Learning (for audio) with Python (2020). Links are provided in each code file to the relevant part of the course where the subject is covered.

5.1 Implementing the Pre-Processing scripts

Firstly, we must tackle the issue of creating the scripts for the pre-processing of our data set.

5.1.1 Pre-Processing the Raw Data

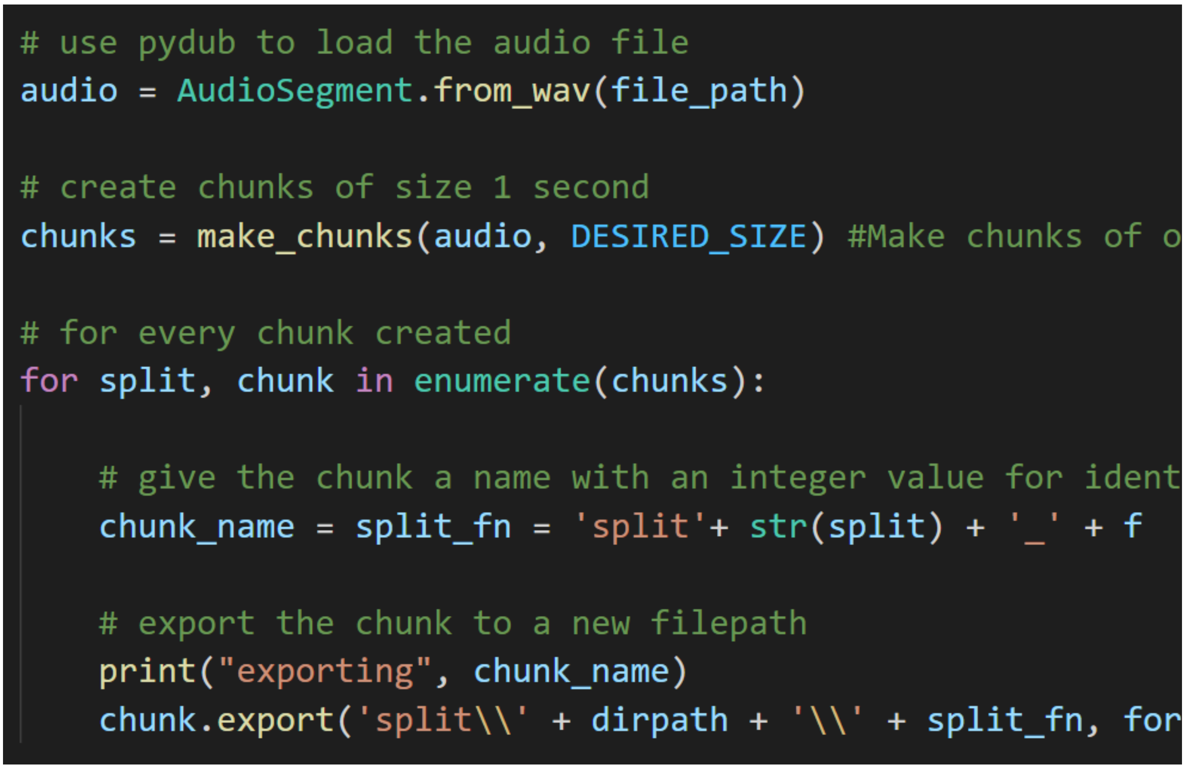
A video demonstration of this phase of the implementation is available at: https://youtu.be/I2aN73vZCHQ

The first step of this implementation is to create the means of ensuring the uniformity of the duration of raw data (See 4.2.1). For the sake of simplicity, the padding and 'chunking' functionality will be implemented in two separate files, splitter.py and padder.py.

Both implementations will walk through the directories of a given path, performing their respective processing on each file within a directory. The general logic of the program is as described in the design documentation. This section focuses on explaining the processing logic. Please see the design documentation for anything else.

5.1.1.1 Splitter.py

Figure 12 shows that the Pydub audio processing library includes all of the functionality for loading the .wav file and performing the chunking operation. We must supply the make_chunks function of Pydub with a duration in milliseconds. Pydub then splits the audio file into multiple chunks of length equal to the given duration and a final chunk that remains of varying duration. Each chunk is then exported to a new directory within the folder, called split.

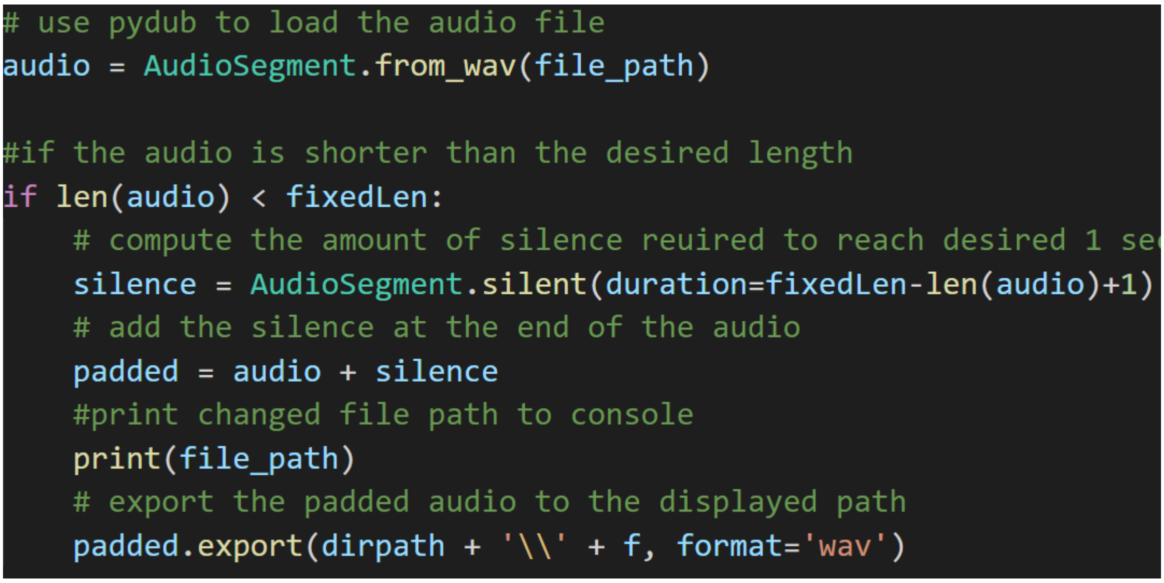


```
# use pydub to load the audio file
audio = AudioSegment.from_wav(file_path)

# create chunks of size 1 second
chunks = make_chunks(audio, DESIRED_SIZE) #Make chunks of o

# for every chunk created
for split, chunk in enumerate(chunks):

    # give the chunk a name with an integer value for ident
    chunk_name = split_fn = 'split'+ str(split) + '_' + f

    # export the chunk to a new filepath
    print("exporting", chunk_name)
    chunk.export('split\\' + dirpath + '\\' + split_fn, for
```

Figure 12. The 'chunking' pre-process implementation

5.1.1.2 Padder.py

This file provides the functionality for padding out audio files that are below the desired length of one second. As shown in Figure 13, this functionality utilises the Pydub library to add silence to the end of the file, such that it will total 1 second in duration. The padded file then overwrites itself once the operation is complete.



```
# use pydub to load the audio file
audio = AudioSegment.from_wav(file_path)

#if the audio is shorter than the desired length
if len(audio) < fixedLen:
    # compute the amount of silence reuired to reach desired 1 se
    silence = AudioSegment.silent(duration=fixedLen-len(audio)+1)
    # add the silence at the end of the audio
    padded = audio + silence
    #print changed file path to console
    print(file_path)
    # export the padded audio to the displayed path
    padded.export(dirpath + '\\' + f, format='wav')
```

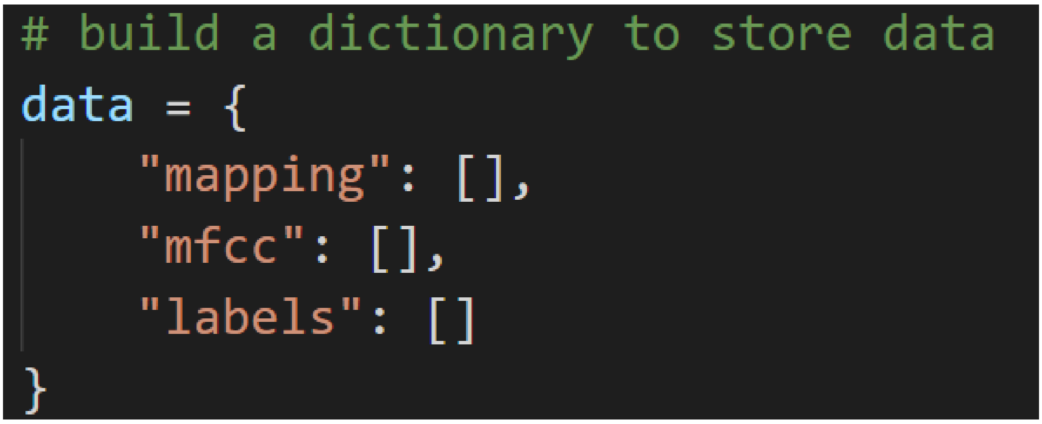
Figure 13. The padding pre-process implementation

5.1.1.3 TM_Audio_Processor

This file provides the functionality for iterating over a given directory, segmenting the audio files into half-second bites and extracting the MFCCs. The MFCCs are stored in a dictionary before being written to an external file as JSON objects.

If we refer to section 4.3.4, we must get values for sample rate and duration as an input. Then, we must perform a calculation to determine the number of samples taken per audio track.

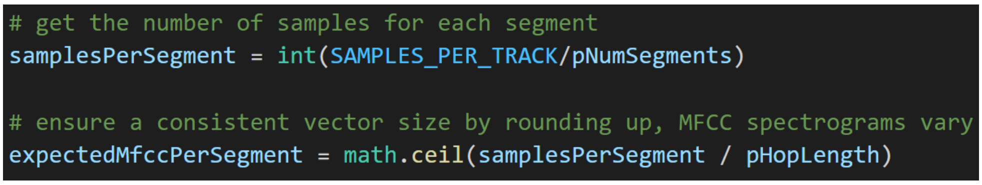
Next, we must build a dictionary (Figure 14) to store our desired output (MFCCs) against the label of emotion that they are supposed to represent. To do this, we declare a dictionary called data and set out the key values. MFCC is the MFCC spectrogram data, and Labels refers to the class label that we will give to the MFCC for identification. Mapping is used to bind the label and MFCC in external operations.



```
# build a dictionary to store data
data = {
    "mapping": [],
    "mfcc": [],
    "labels": []
}
```

Figure 14. Declare a dictionary for the storage of processed data.

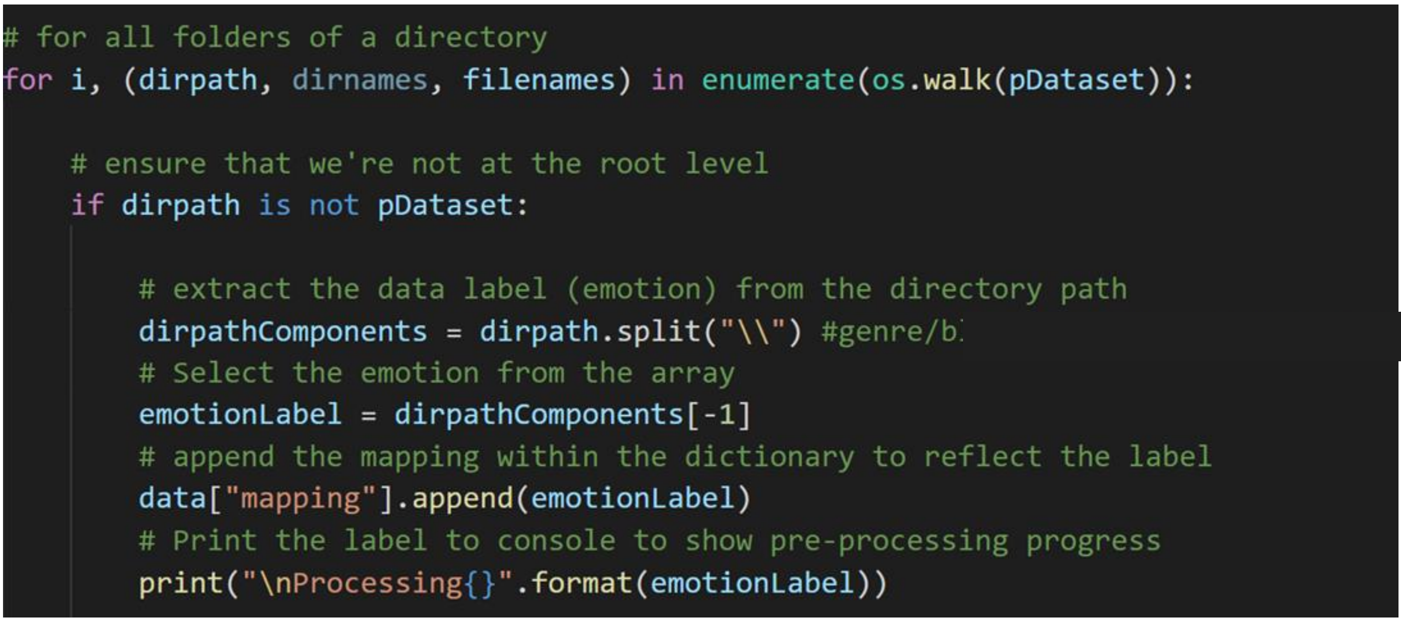
The next step is to standardise the size of each vector by rounding it up (Figure 15). This shouldn't be necessary in our case, as we normalised the raw data, but it is good practice to do so in case we use non-standard data in the future. Uniform MFCCs are crucial for input into Deep Learning architectures. As MFCC vary by time, it is essential to undertake this step manually.



```
# get the number of samples for each segment
samplesPerSegment = int(SAMPLES_PER_TRACK/pNumSegments)

# ensure a consistent vector size by rounding up, MFCC spectrograms vary
expectedMfccPerSegment = math.ceil(samplesPerSegment / pHopLength)
```

Figure 15. calculate how many samples per segment and round up the value of the expected number of MFCCs per segment.

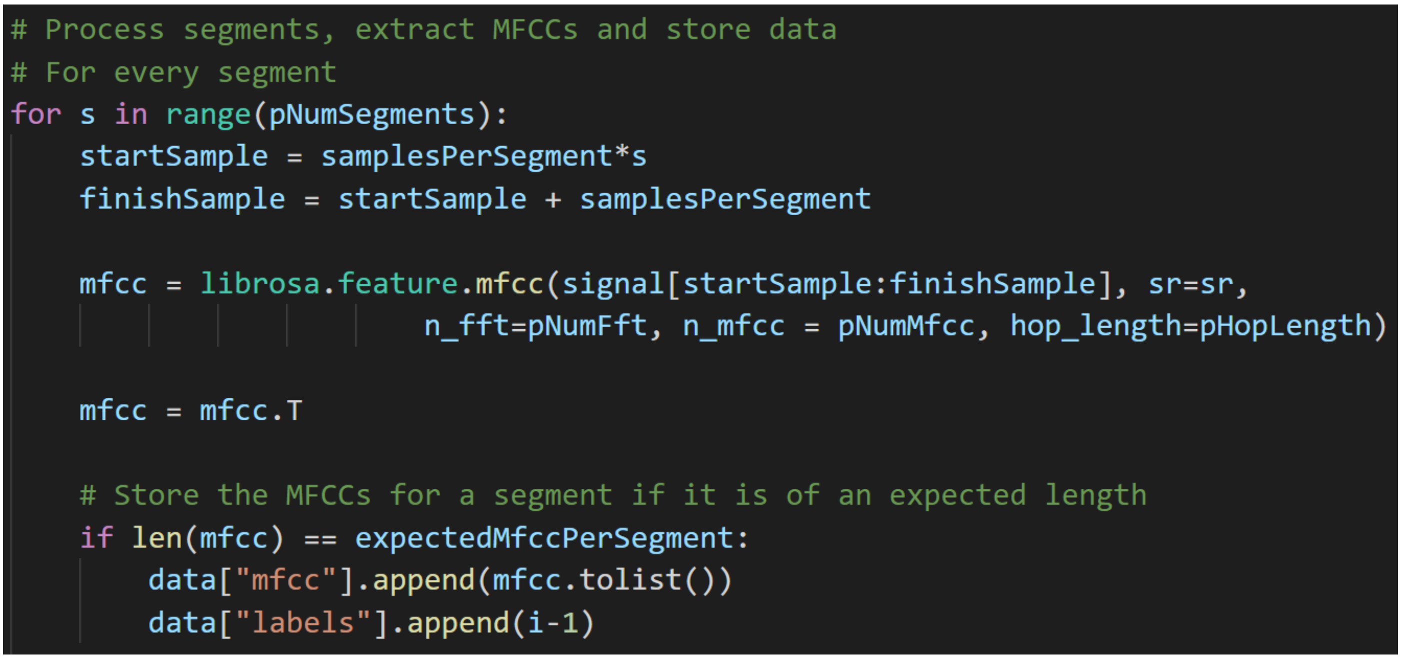
Next, we must iterate through the directory of the dataset. In this implementation, the folder names are representative of the label that should be used for the data. Thus, we perform a split of the file path to get the final value of the directory and assign this to a local variable as the 'emotionLabel' (Figure 16).



```
# for all folders of a directory
for i, (dirpath, dirnames, filenames) in enumerate(os.walk(pDataset)):

    # ensure that we're not at the root level
    if dirpath is not pDataset:

        # extract the data label (emotion) from the directory path
        dirpathComponents = dirpath.split("\\") #genre/b
        # Select the emotion from the array
        emotionLabel = dirpathComponents[-1]
        # append the mapping within the dictionary to reflect the label
        data["mapping"].append(emotionLabel)
        # Print the label to console to show pre-processing progress
        print("\nProcessing{}".format(emotionLabel))
```

Figure 16. the code responsible for moving through the root directory of the data set. The folder names are used as labels for classification.

As shown in Figure 17, the Librosa library both loads the waveform and performs MFCC extraction in a single line of code each. Once we have extracted the data we require (stored in variable mfcc), we must append the value to our dictionary.



```
# Process segments, extract MFCCs and store data
# For every segment
for s in range(pNumSegments):
    startSample = samplesPerSegment*s
    finishSample = startSample + samplesPerSegment

    mfcc = librosa.feature.mfcc(signal[startSample:finishSample], sr=sr,
                        n_fft=pNumFft, n_mfcc = pNumMfcc, hop_length=pHopLength)

    mfcc = mfcc.T

    # Store the MFCCs for a segment if it is of an expected length
    if len(mfcc) == expectedMfccPerSegment:
        data["mfcc"].append(mfcc.tolist())
        data["labels"].append(i-1)
```

Figure 17. The implementation of the transformation from a waveform to MFCC coefficients. The desired data (MFCC)is then appended to the dictionary and assigned a label

5.2 Implementing Neural Networks

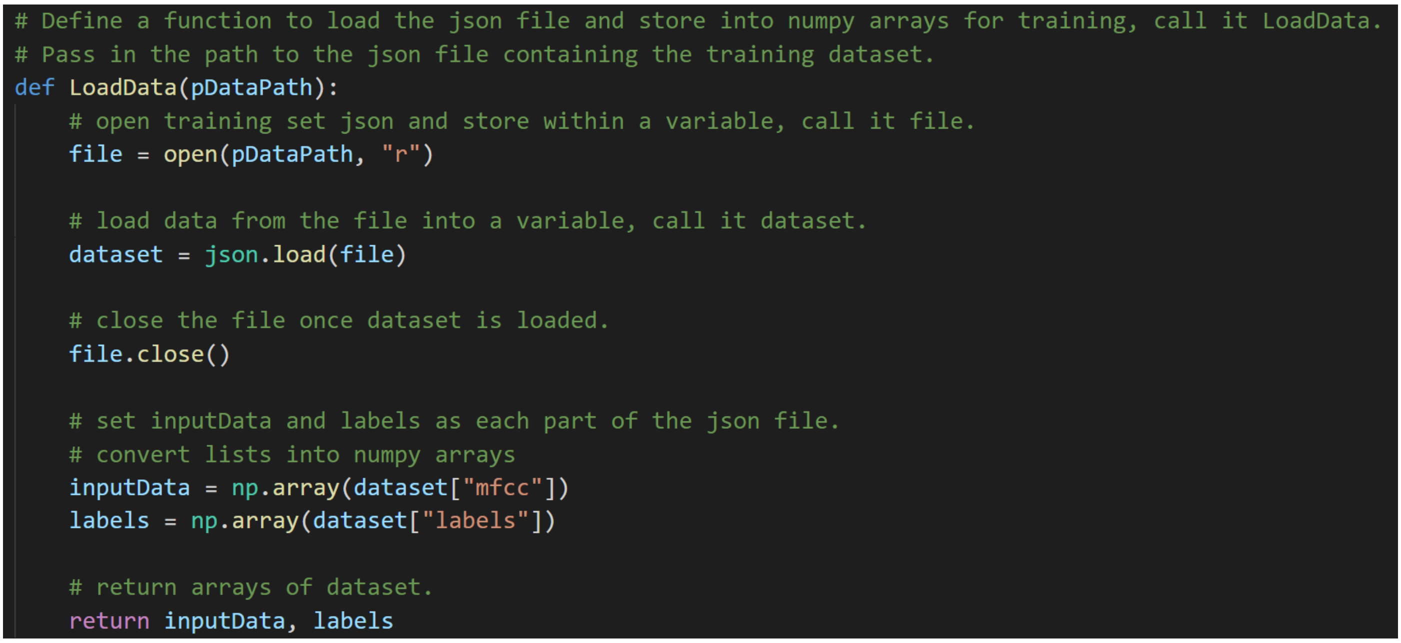
Keras, the high-level platform of TensorFlow, allows the ability to monitor specific metrics during model training (Keras, 2021.) These metrics are recorded after each epoch, which is one pass through the entire dataset (Brownlee, 2018). Analysing these metrics can detect model inefficiencies, such as overfitting, that indicate how the model's parameters should be tuned to achieve an optimal result (Digitalsreeni, 2020). Keras offers metrics for both regression problems, relating to the prediction of a numeric value, and classification problems regarding the prediction of a class label. As we are investigating a classification problem, we will utilise classification metrics.

5.2.1 Functionality Common Across Our ANNs

The implementation of neural network models will follow the following implementation, though with differing values for the structure of the model.

5.2.1.1 LoadData()

The LoadData function (Figure 18) will load the dataset from a given path using the JSON library. It will then store the loaded file into two arrays, input data and labels, dependent on the key for each value.

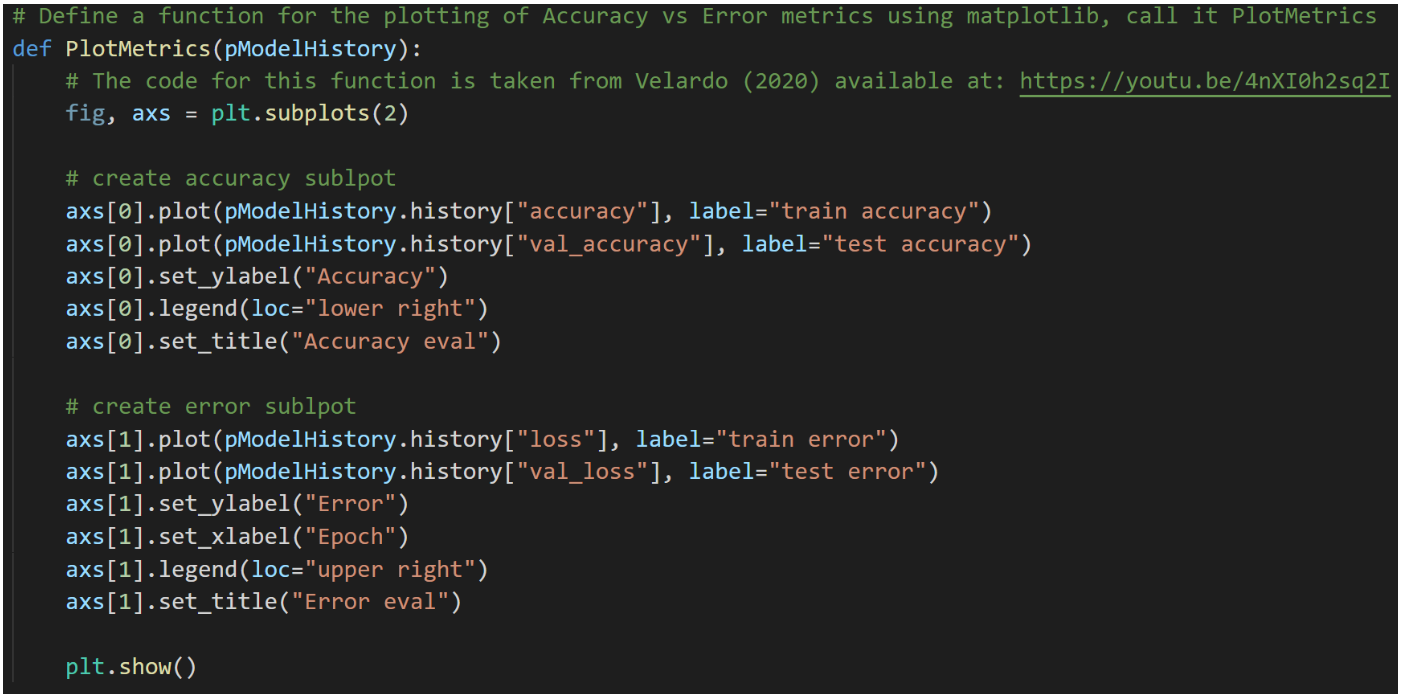


```python
# Define a function to load the json file and store into numpy arrays for training, call it LoadData.
# Pass in the path to the json file containing the training dataset.
def LoadData(pDataPath):
    # open training set json and store within a variable, call it file.
    file = open(pDataPath, "r")

    # load data from the file into a variable, call it dataset.
    dataset = json.load(file)

    # close the file once dataset is loaded.
    file.close()

    # set inputData and labels as each part of the json file.
    # convert lists into numpy arrays
    inputData = np.array(dataset["mfcc"])
    labels = np.array(dataset["labels"])

    # return arrays of dataset.
    return inputData, labels
```

Figure 18. The LoadData function standard across all of our model implementations

5.2.1.2 PlotMetrics()

This function is the work of Velardo (2020). PlotMetrics() (Figure 19) leverages matplotlib to draw a plot of the model accuracy vs the number of epochs (dataset passes). This graph is crucial for efficient hyperparameter tuning for the attainment of a more performant solution.

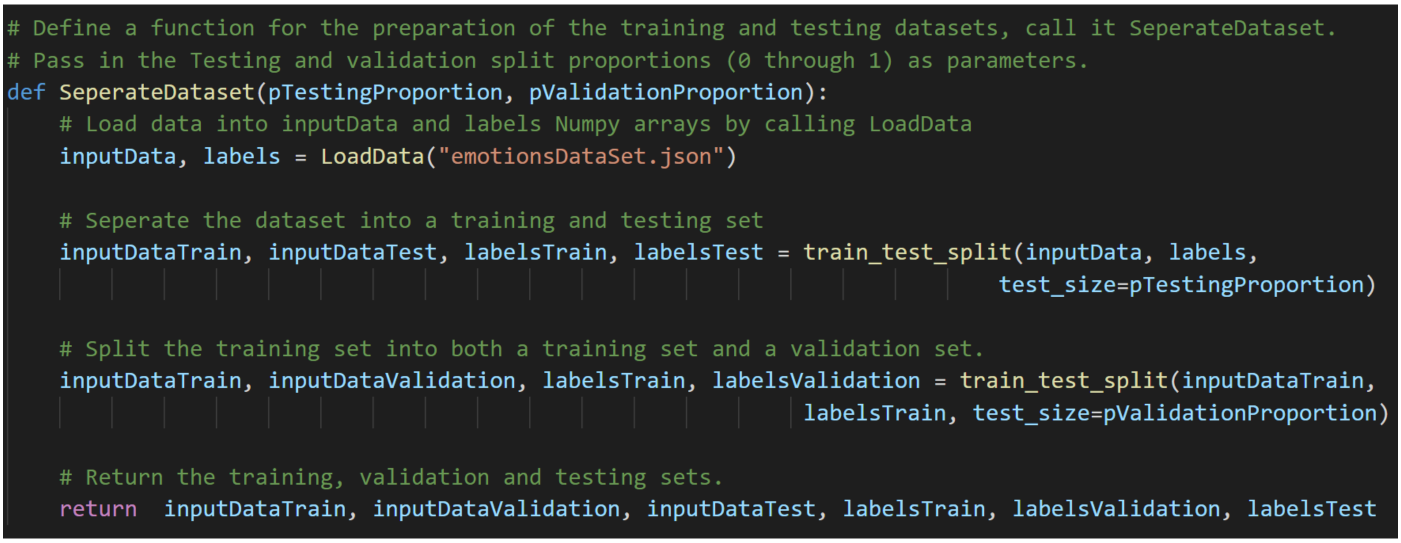


```python
# Define a function for the plotting of Accuracy vs Error metrics using matplotlib, call it PlotMetrics
def PlotMetrics(pModelHistory):
    # The code for this function is taken from Velardo (2020) available at: https://youtu.be/4nXI0h2sq2I
    fig, axs = plt.subplots(2)

    # create accuracy sublpot
    axs[0].plot(pModelHistory.history["accuracy"], label="train accuracy")
    axs[0].plot(pModelHistory.history["val_accuracy"], label="test accuracy")
    axs[0].set_ylabel("Accuracy")
    axs[0].legend(loc="lower right")
    axs[0].set_title("Accuracy eval")

    # create error sublpot
    axs[1].plot(pModelHistory.history["loss"], label="train error")
    axs[1].plot(pModelHistory.history["val_loss"], label="test error")
    axs[1].set_ylabel("Error")
    axs[1].set_xlabel("Epoch")
    axs[1].legend(loc="upper right")
    axs[1].set_title("Error eval")

    plt.show()
```

Figure 19. The PlotMetrics function used to plot a graph of the solution accuracy vs epochs.

5.2.1.3 SeparateDataset()

This SeperateDataset() function (Figure 20) provides the utilises Scikit-Leams train_test_split function to separate the data into training, testing and validation sets. The training set is what the model uses to leam. It is unsuitable to use this data to evaluate the model's accuracy as the model's weights will be adjusted to classify the data more accurately. Instead, we should use unseen data (validation and test set) to test the model. The model predicts the test set on every epoch completion to determine the model accuracy whilst leaving weights unaffected. The validation data is used once the model is fully trained to produce a final accuracy value for the model.



```
# Define a function for the preparation of the training and testing datasets, call it SeperateDataset.
# Pass in the Testing and validation split proportions (0 through 1) as parameters.
def SeperateDataset(pTestingProportion, pValidationProportion):
    # Load data into inputData and labels Numpy arrays by calling LoadData
    inputData, labels = LoadData("emotionsDataSet.json")

    # Seperate the dataset into a training and testing set
    inputDataTrain, inputDataTest, labelsTrain, labelsTest = train_test_split(inputData, labels,
                                                                              test_size=pTestingProportion)

    # Split the training set into both a training set and a validation set.
    inputDataTrain, inputDataValidation, labelsTrain, labelsValidation = train_test_split(inputDataTrain,
                                                                         labelsTrain, test_size=pValidationProportion)

    # Return the training, validation and testing sets.
    return  inputDataTrain, inputDataValidation, inputDataTest, labelsTrain, labelsValidation, labelsTest
```

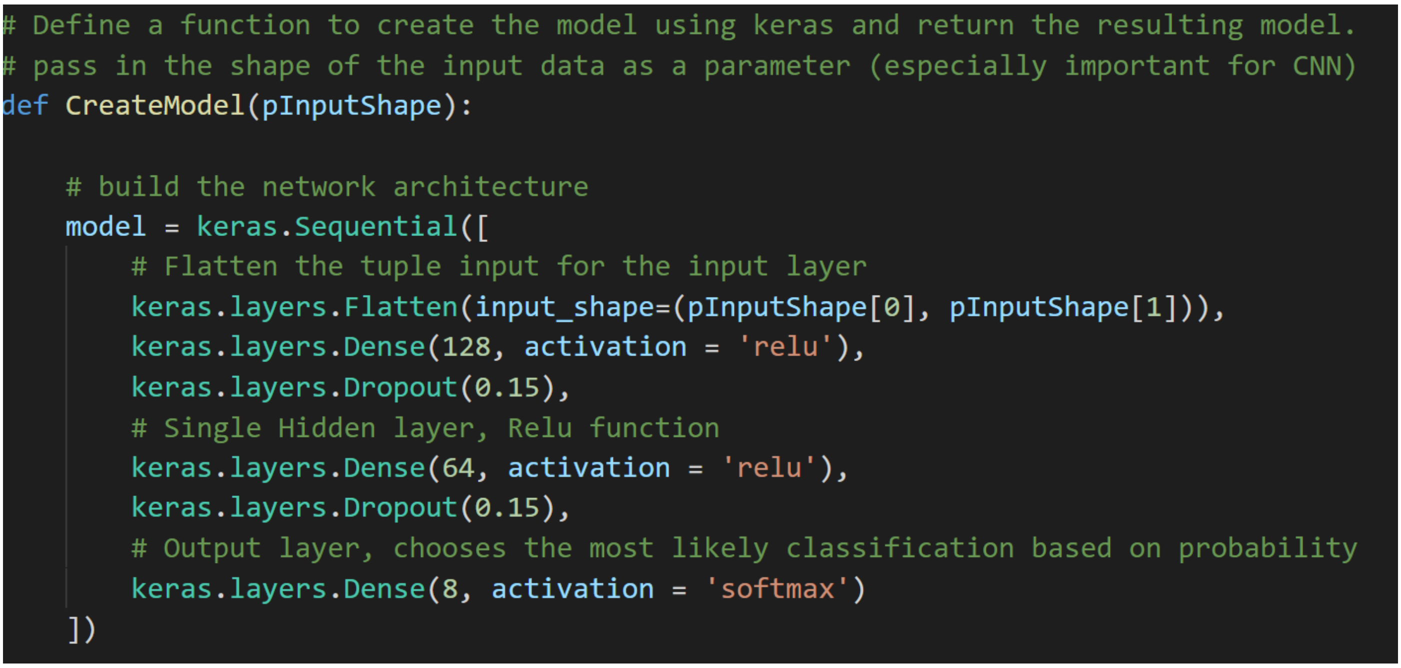
Figure 20. The SeperateDataset Function used to split the input dataset into individual testing, training and validation sets.

5.2.2 The Multi-Layer Perceptron.

A video demonstration of this phase of the implementation is available at: https://youtu.be/Mg4yn4VUSHg

5.2.2.1 Implementation and Training

Figure 21 shows the final implementation of the MultiLayer perceptron. The model is composed of three fully connected layers, as shown in section 4.4.1. We include a dropout layer in order to account for overfitting (please see section 5.2.2.2).



```
# Define a function to create the model using keras and return the resulting model.
# pass in the shape of the input data as a parameter (especially important for CNN)
def CreateModel(pInputShape):

    # build the network architecture
    model = keras.Sequential([
        # Flatten the tuple input for the input layer
        keras.layers.Flatten(input_shape=(pInputShape[0], pInputShape[1])),
        keras.layers.Dense(128, activation = 'relu'),
        keras.layers.Dropout(0.15),
        # Single Hidden layer, Relu function
        keras.layers.Dense(64, activation = 'relu'),
        keras.layers.Dropout(0.15),
        # Output layer, chooses the most likely classification based on probability
        keras.layers.Dense(8, activation = 'softmax')
    ])
```

Figure 21. The final implementation of the MLP structure.

5.2.2.2 Analysing Training and Tuning Hyperparameters

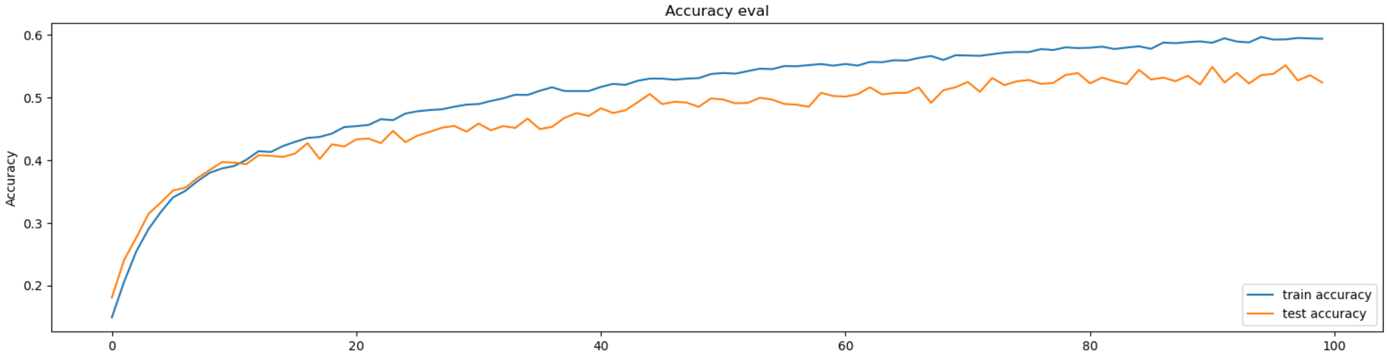


Figure 22. The metrics of the MLP model before we applied any dropout.

Initially, we trained the Multi-Layer Perceptron without any anti-overfitting measures, such as dropout (Figure 22). The result was severe overfitting, with 60% accuracy on the training data and 48% on the validation data.

By applying dropout layers with parameters of 0.15, we achieve very similar results for training and test data. Therefore, we have a more suitably generalisable model with a final accuracy of 0.51% (Figure 23).

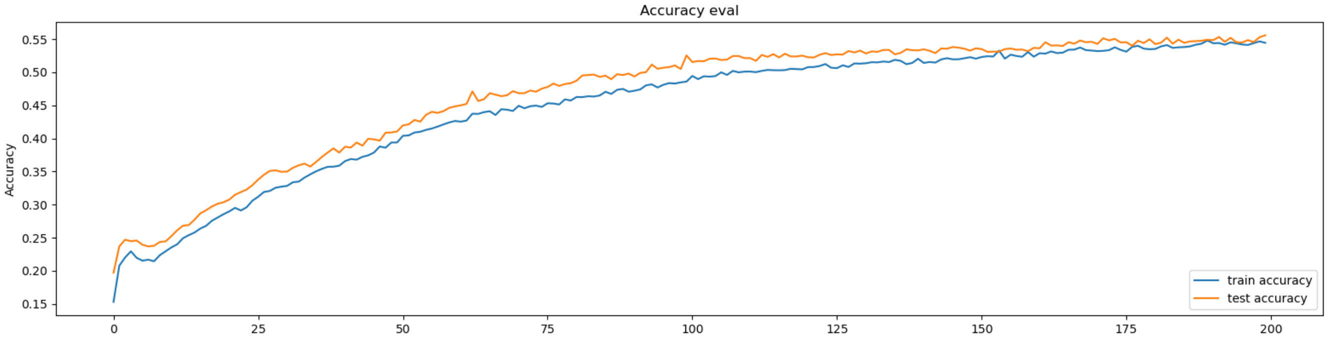


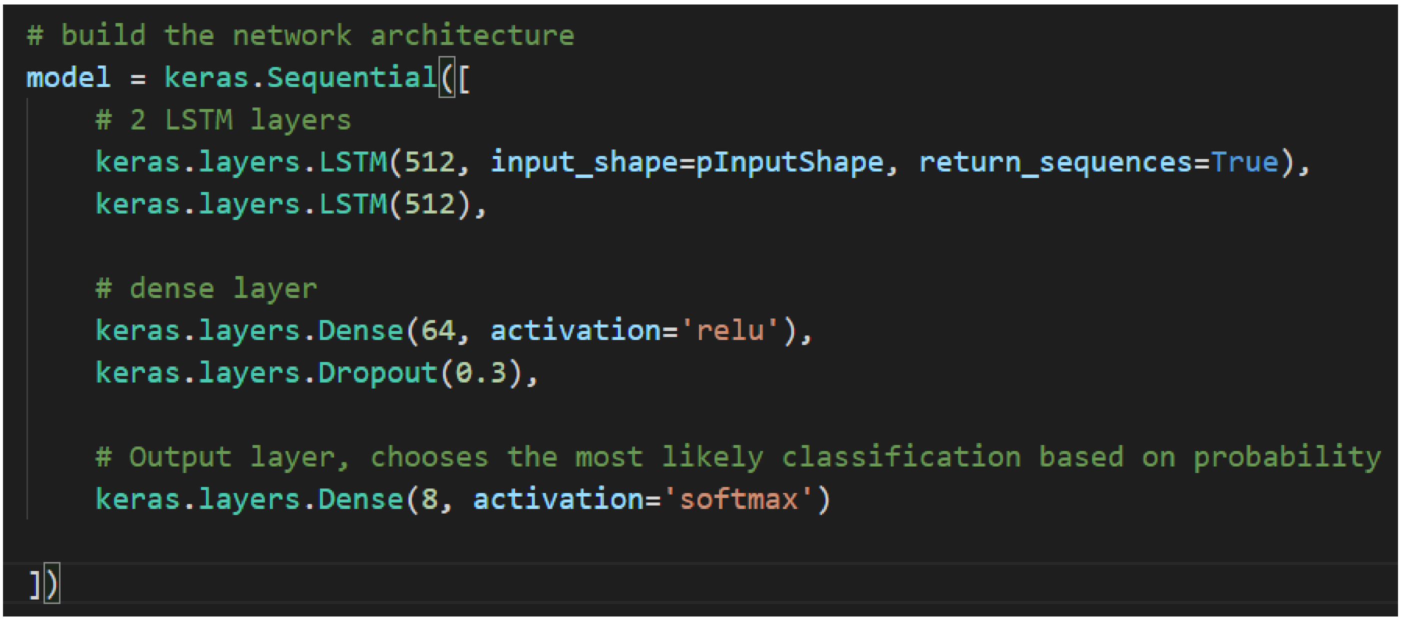
Figure 23. The same MLP model after we implemented dropout.

5.2.3 The LSTM Model

A video demonstration of this phase of the implementation is available at: https://youtu.be/CmZALx_nV60

5.2.3.1 Implementing and Training the initial model

For the initial LSTM model, we decided to begin with a high number of neurons (greater complexity) and work backwards to reach a compromise between accuracy and training time.



```
# build the network architecture
model = keras.Sequential([
    # 2 LSTM layers
    keras.layers.LSTM(512, input_shape=pInputShape, return_sequences=True),
    keras.layers.LSTM(512),

    # dense layer
    keras.layers.Dense(64, activation='relu'),
    keras.layers.Dropout(0.3),

    # Output layer, chooses the most likely classification based on probability
    keras.layers.Dense(8, activation='softmax')

])
```

Figure 24. The initial implementation of the LSTM network.

5.2.3.2 Analysing Training and Tuning Hyperparameters

Training the highly complex LSTM model took around 90 minutes. This was significantly longer than the training of the MLP, which took approximately two minutes for 200 total epochs. The LSTM solution only ran for 30 epochs due to the significant amount of time required.

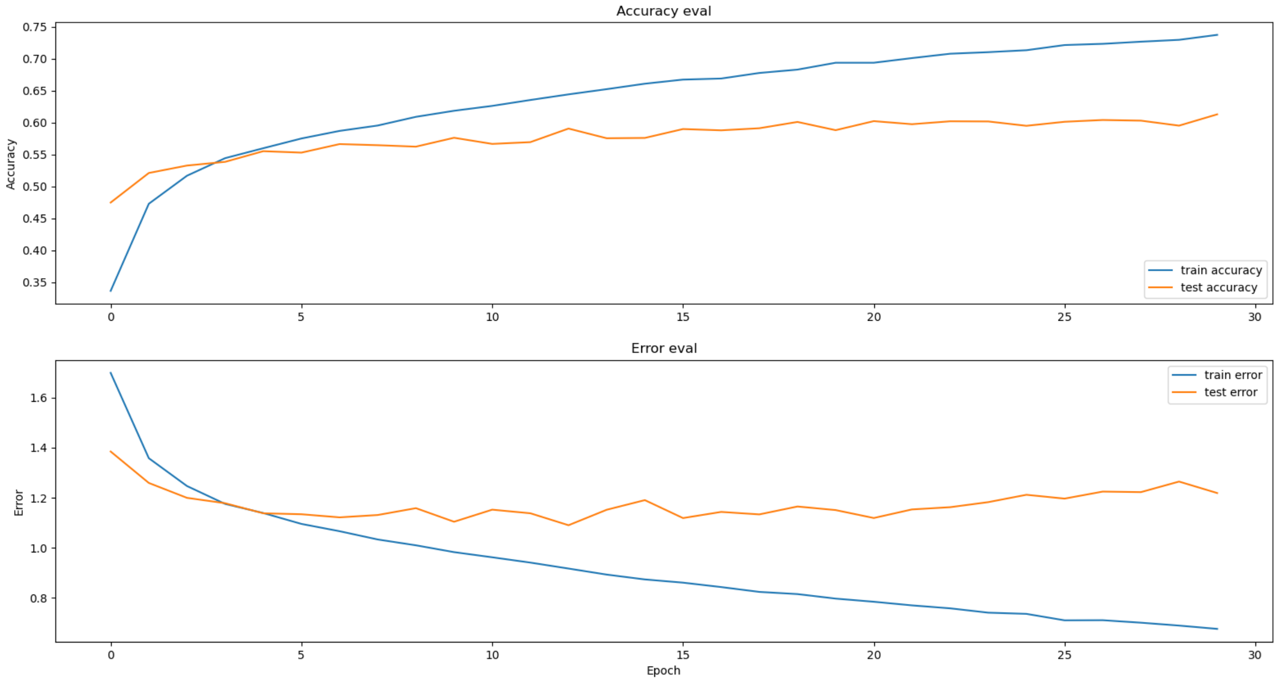
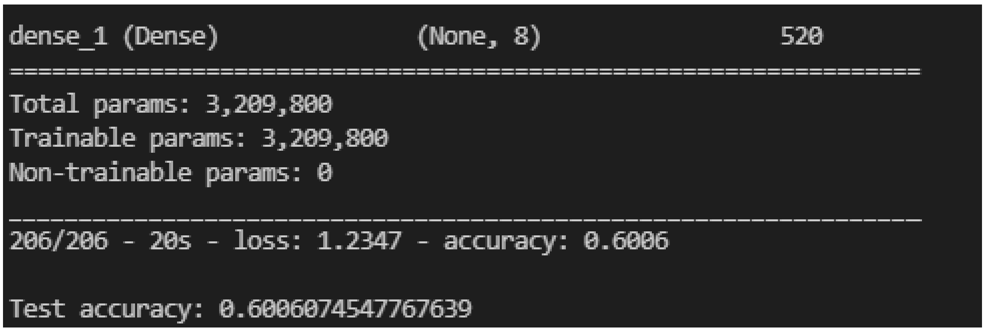


*Figure 25. Accuracy vs epoch for LSTM with the high complexity of 512*512 units*

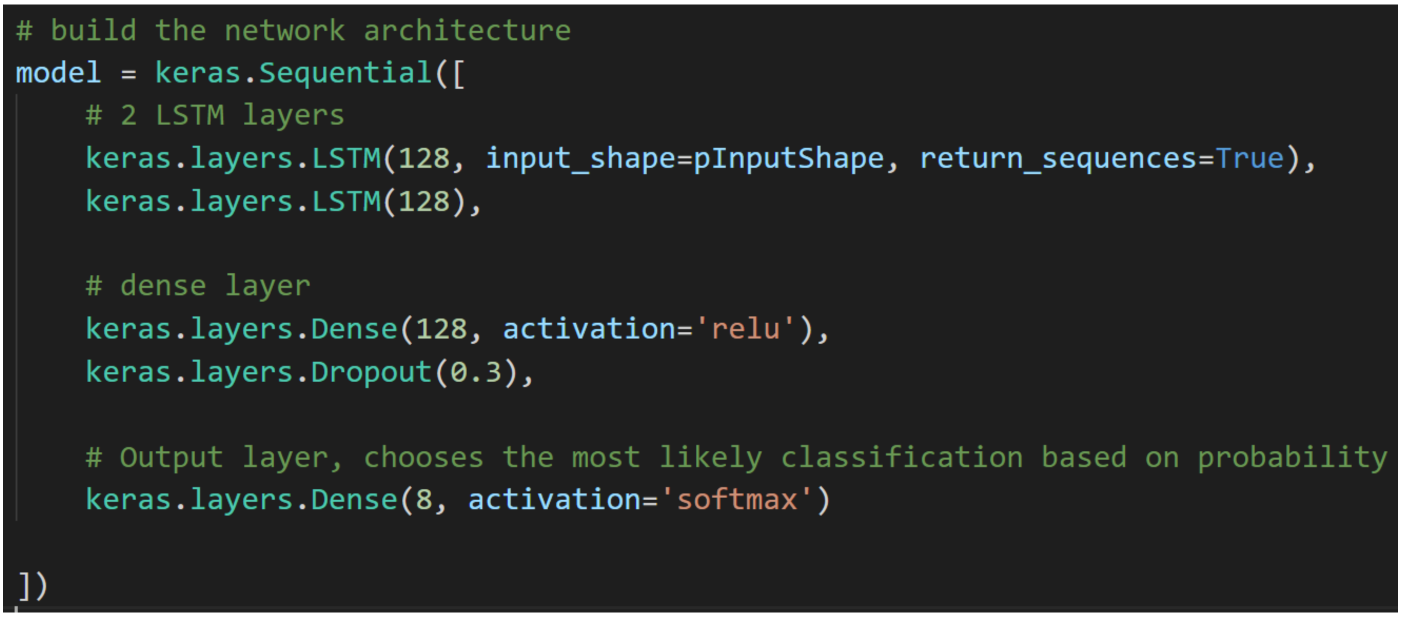
The clear asymptote shown in Figure 25 indicates that the majority of progress occurred before epoch 15. Therefore, we can reduce the number of epochs should we wish to retrain the model. Furthermore, there was significant overfitting, beginning at around epoch five, so we should increase dropout. Our opinion was that the complexity was too great for the marginal improvement of accuracy. We can achieve similar levels using a more straightforward implementation and compile and train far more quickly. It is unrealistic to tune the accuracy of a network with 3 million parameters (Figure 26) without a powerful workstation. As the running time was around an hour and a half for compilation, it is pretty much impossible to perform trial and error on a network of this complexity, as the build time is prohibitive n the home environment. Instead, we will seek to reduce the complexity and address th e overfitting problem. As mentioned previously, the tuning of parameters is a trial and error problem.



```
dense_1 (Dense)              (None, 8)                 520
=================================================================
Total params: 3,209,800
Trainable params: 3,209,800
Non-trainable params: 0
_________________________________________________________________
206/206 - 20s - loss: 1.2347 - accuracy: 0.6006

Test accuracy: 0.6006074547767639
```

Figure 26. The final summary of the highly complex LSTM model, showing both the accuracy and complexity of the model.

We will make slight adjustments to this model. There are issues with overfitting; thus, we will increase the dropout value. Furthermore, we will reduce the complexity of the LTSM layer to 128 units, which should significantly improve training time. Figure 27 shows the improved implementation.



```
# build the network architecture
model = keras.Sequential([
    # 2 LSTM layers
    keras.layers.LSTM(128, input_shape=pInputShape, return_sequences=True),
    keras.layers.LSTM(128),

    # dense layer
    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dropout(0.3),

    # Output layer, chooses the most likely classification based on probability
    keras.layers.Dense(8, activation='softmax')

])
```

Figure 27. The improved LSTM model with tuned hyperparameters.

The time per epoch (single dataset pass) reduced from around two and a half minutes to 8 seconds on running this model of reduced complexity, which is more viable when training with limited resources. However, looking to Figure 28, the validation accuracy started to lag behind the training accuracy, suggesting significant overfitting. Furthermore, there is once again evidence that past epoch 13, gains in accuracy begin to diminish. Furthermore, the final accuracy result for this implementation was significantly lower, with a value of 52% compared to 60% on the more complex model (Figure 29).

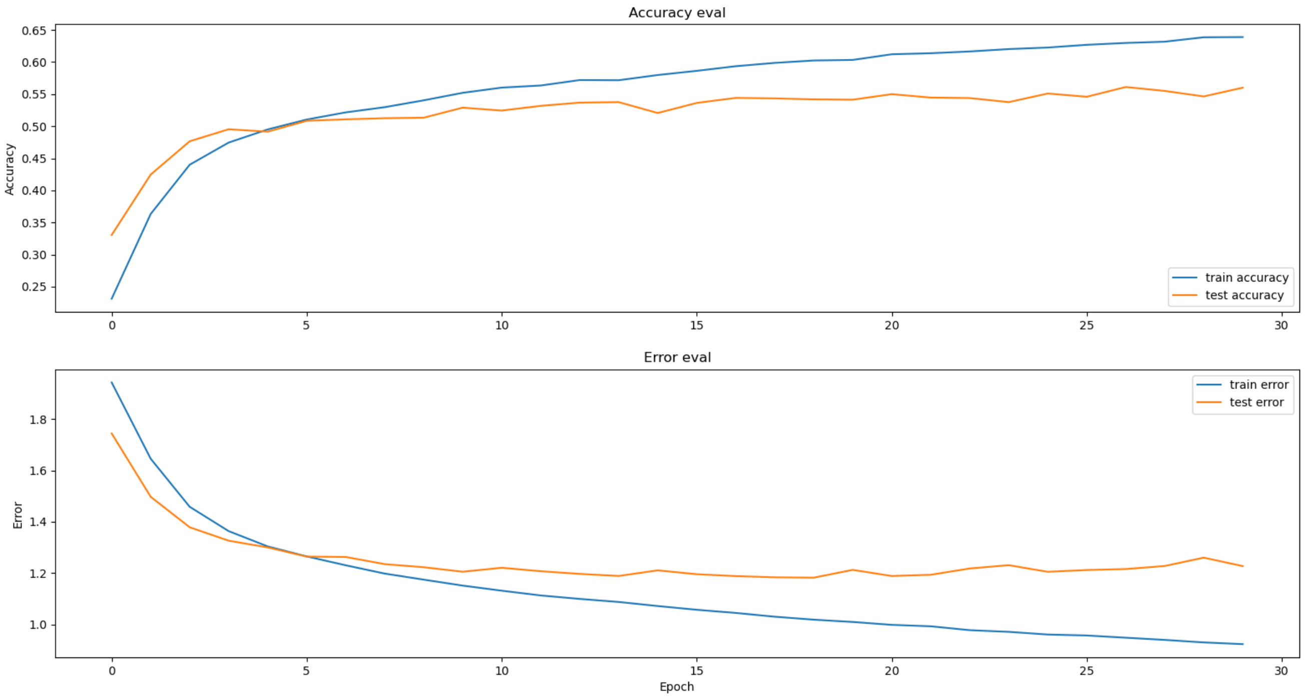
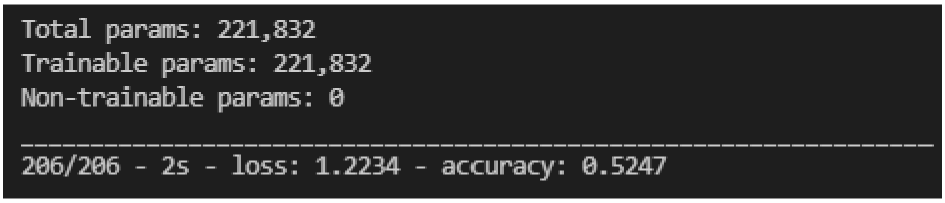


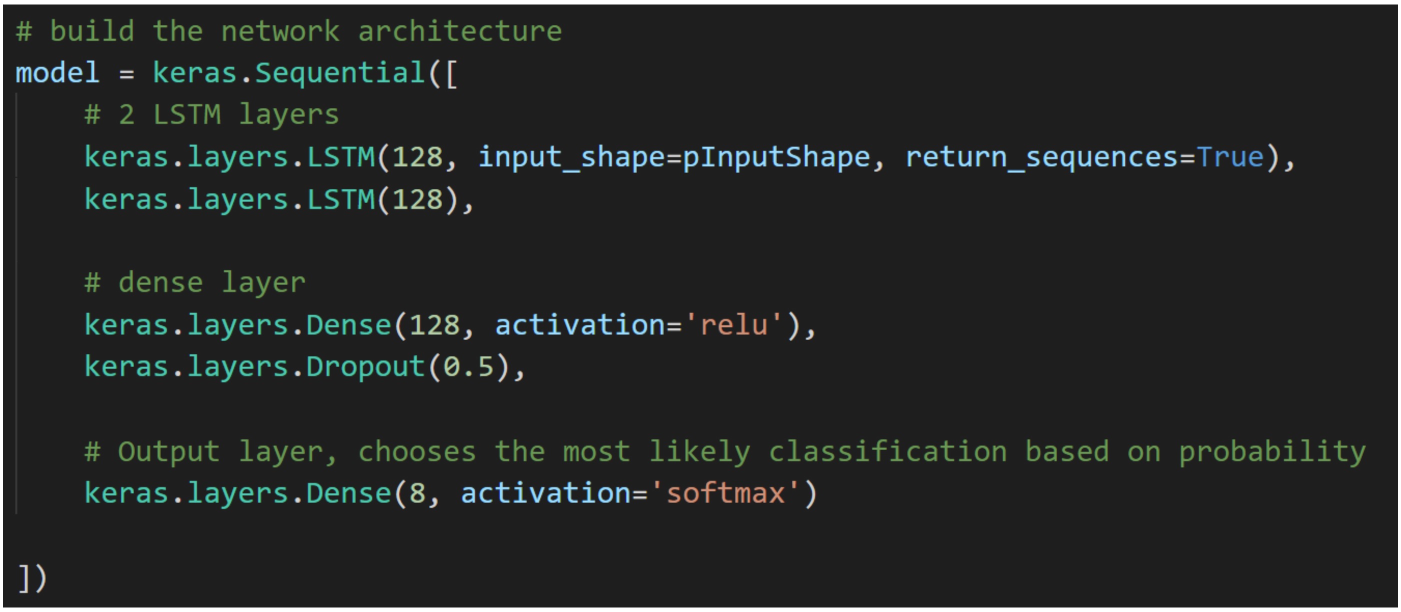
Figure 28. The graph of accuracy vs epochs for the improved LSTM model.



```
Total params: 221,832
Trainable params: 221,832
Non-trainable params: 0

________________________________________________________________
206/206 - 2s - loss: 1.2234 - accuracy: 0.5247
```

Figure 29. The final accuracy and complexity values for the improved LSTM model

Suppose we could get similar levels of accuracy between the training and test data. In that case, we potentially could see an increase in validation accuracy from 55 to 60%, which would be a significant improvement if the model were to be implemented in the real world. To try and achieve this, we will increase the dropout figure to trigger with 0.5 probability (Figure 30). Doing so should narrow the difference between the train and test accuracies.



```python
# build the network architecture
model = keras.Sequential([
    # 2 LSTM layers
    keras.layers.LSTM(128, input_shape=pInputShape, return_sequences=True),
    keras.layers.LSTM(128),

    # dense layer
    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dropout(0.5),

    # Output layer, chooses the most likely classification based on probability
    keras.layers.Dense(8, activation='softmax')

])
```

Figure 30. The final implementation of the LSTM model.

The result of this change is shown in Figure 31. There is a significant improvement regarding the discrepancy between training and testing values. However, the time per epoch increased to 11 seconds. The final accuracy result achieved for the less complex LSTM model was 54%.

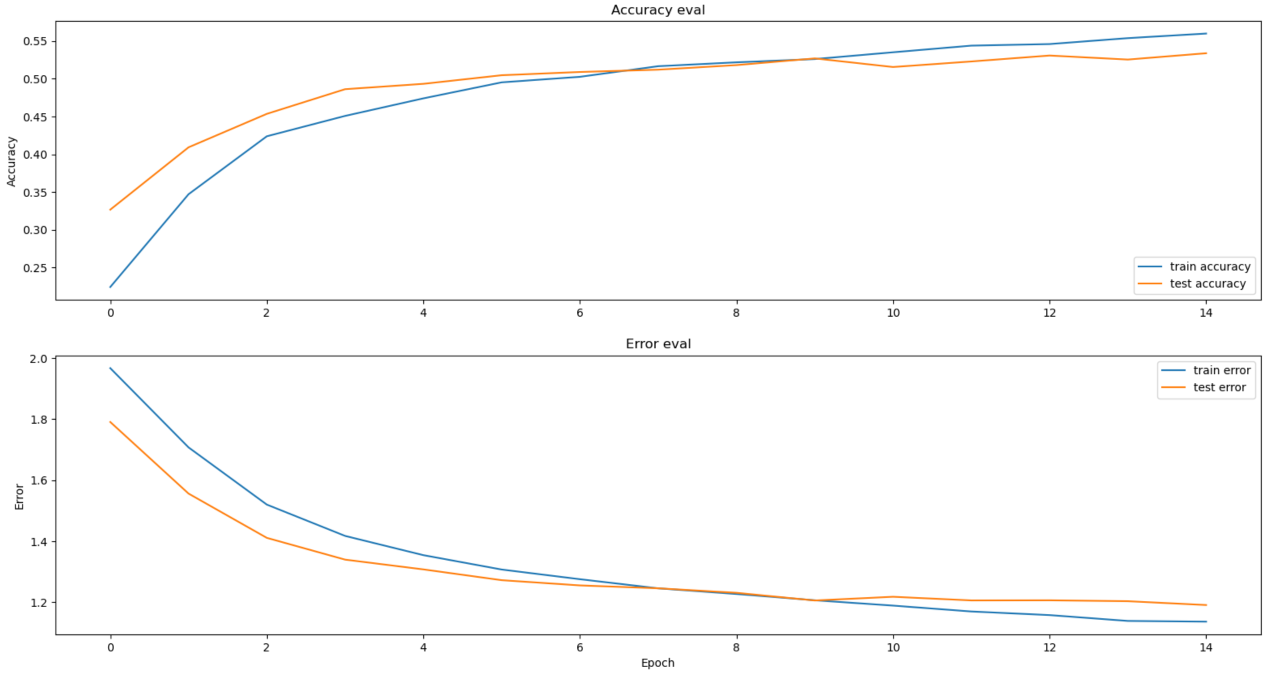
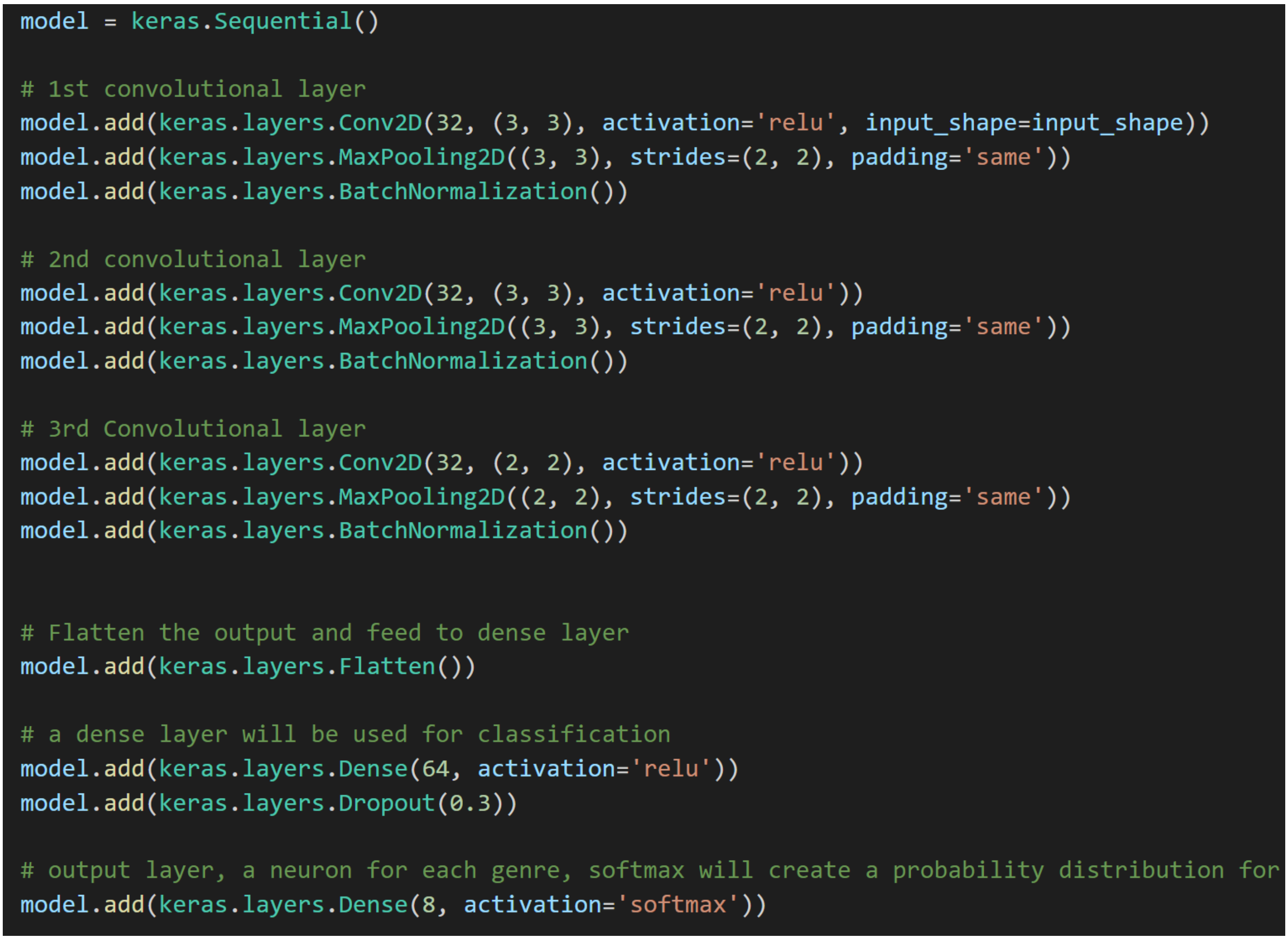


Figure 31. The final results for the less complex LSTM model.

5.2.4 The Convolutional Neural Network

5.2.4.1 Implementation and Training

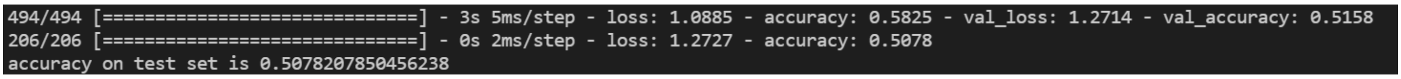
To implement the convolutional neural network, we utilised the model provided by Velardo (2020) (Figure 32) in his course *Deep Learning (for audio) with Python*.



```
model = keras.Sequential()

# 1st convolutional layer
model.add(keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=input_shape))
model.add(keras.layers.MaxPooling2D((3, 3), strides=(2, 2), padding='same'))
model.add(keras.layers.BatchNormalization())

# 2nd convolutional layer
model.add(keras.layers.Conv2D(32, (3, 3), activation='relu'))
model.add(keras.layers.MaxPooling2D((3, 3), strides=(2, 2), padding='same'))
model.add(keras.layers.BatchNormalization())

# 3rd Convolutional layer
model.add(keras.layers.Conv2D(32, (2, 2), activation='relu'))
model.add(keras.layers.MaxPooling2D((2, 2), strides=(2, 2), padding='same'))
model.add(keras.layers.BatchNormalization())


# Flatten the output and feed to dense layer
model.add(keras.layers.Flatten())

# a dense layer will be used for classification
model.add(keras.layers.Dense(64, activation='relu'))
model.add(keras.layers.Dropout(0.3))

# output layer, a neuron for each genre, softmax will create a probability distribution for
model.add(keras.layers.Dense(8, activation='softmax'))
```

Figure 32. Velardo (2020) presented this implementation of a CNN during his course

Upon running (see Figure 33), this model had similar accuracy to the Multi-Layer Perceptron without any parameter tuning (around 50%). However, the train time per epoch was significantly longer, at around 3 seconds compared to one second for the MLP.

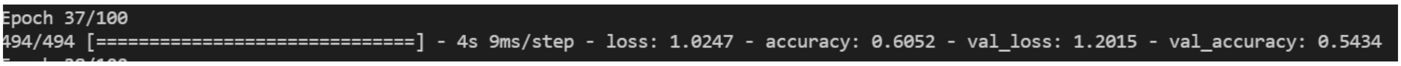


```
494/494 [==============================] - 3s 5ms/step - loss: 1.0885 - accuracy: 0.5825 - val_loss: 1.2714 - val_accuracy: 0.5158
206/206 [==============================] - 0s 2ms/step - loss: 1.2727 - accuracy: 0.5078
accuracy on test set is 0.5078207850456238
```

Figure 33. The initial run of Velardo's (2020) model without any tuning

5.2.4.2 Tuning the Hyperparameters

To improve the accuracy of the CNN model, we increased the complexity from 32 to 64 units per convolutional layers resulting in a final tuned accuracy of 54% (Figure 33) at the 37th epoch. This result is slightly better than the accuracy achieved by the MLP (51%).



```
Epoch 37/100
494/494 [==============================] - 4s 9ms/step - loss: 1.0247 - accuracy: 0.6052 - val_loss: 1.2015 - val_accuracy: 0.5434
```

Figure 34. The final, tuned implementation of Velardo's (2020) model for a CNN architecture.

Further tuning of the convolutional layer parameters would require significant processing power, which is currently beyond the scope of what is available to us.

6 Discussion

Looking back to the requirements specification, it is clear that efficient and highly scalable solutions for both data pre-processing and classification are entirely feasible with limited computational resources. Clear trends corroborated what was seen in the literature review. The multilayer perceptron provides a suitable baseline performance but is outclassed by other architectures. The Convolutional Neural Network proved to be highly effective in classifying MFCCs, because MFCC spectrograms are essentially images. Similarly, the LSTM-RNN architecture was the most accurate, providing a classification accuracy of approx. 60% on unseen data. However, the LSTM took significantly longer to train (around 90 minutes).

During the literature search, it became apparent that there was little available research testing the robustness of deep learning implementations of SER across cultures. It is likely the fact that, as discussed in section 3.3, there is a significant challenge to achieving a reliable and broad enough data set for emotion classification. If I were to undertake funded research, I believe testing my implementation with inputs of other languages would be highly worthwhile. Doing so could prove that it is unnecessary to develop an emotion classifier specific to each language/dialect. However, looking at this with a critical eye, I expect that the lack of literature on the subject may be because of the assumption that this sort of recognition must be specific to a given range of inputs. There is just too extensive a range of differences between cultures, dialects and meaning. As discussed in the literature review, it does truly boil back to whether emotion can be classified at all. Perhaps we shouldn't ever expect the classification of emotion to be undoubtedly accurate. Instead, we could consider it simply a metric that we can factor in improving the accuracy of our current systems.

Having a greater understanding of hyperparameters would have opened the door for a wealth of additional analysis. The conclusion drawn from looking at the accuracy statistics is that different Deep Learning architectures are similarly successful for audio classification, regardless of which architecture I tested. Therefore, I would like to look more deeply into achieving the high levels of accuracy present in much of the literature. I expect that the reason for the discrepancy between my results (60%) and those achieved by many research examples (85-90%) comes down to the ability to tune the model dynamically, with vast numbers of parameter variations, to ensure that the model is performing as well as it possibly can.

7 Reflection

7.1 Analysis of Gannt Chart

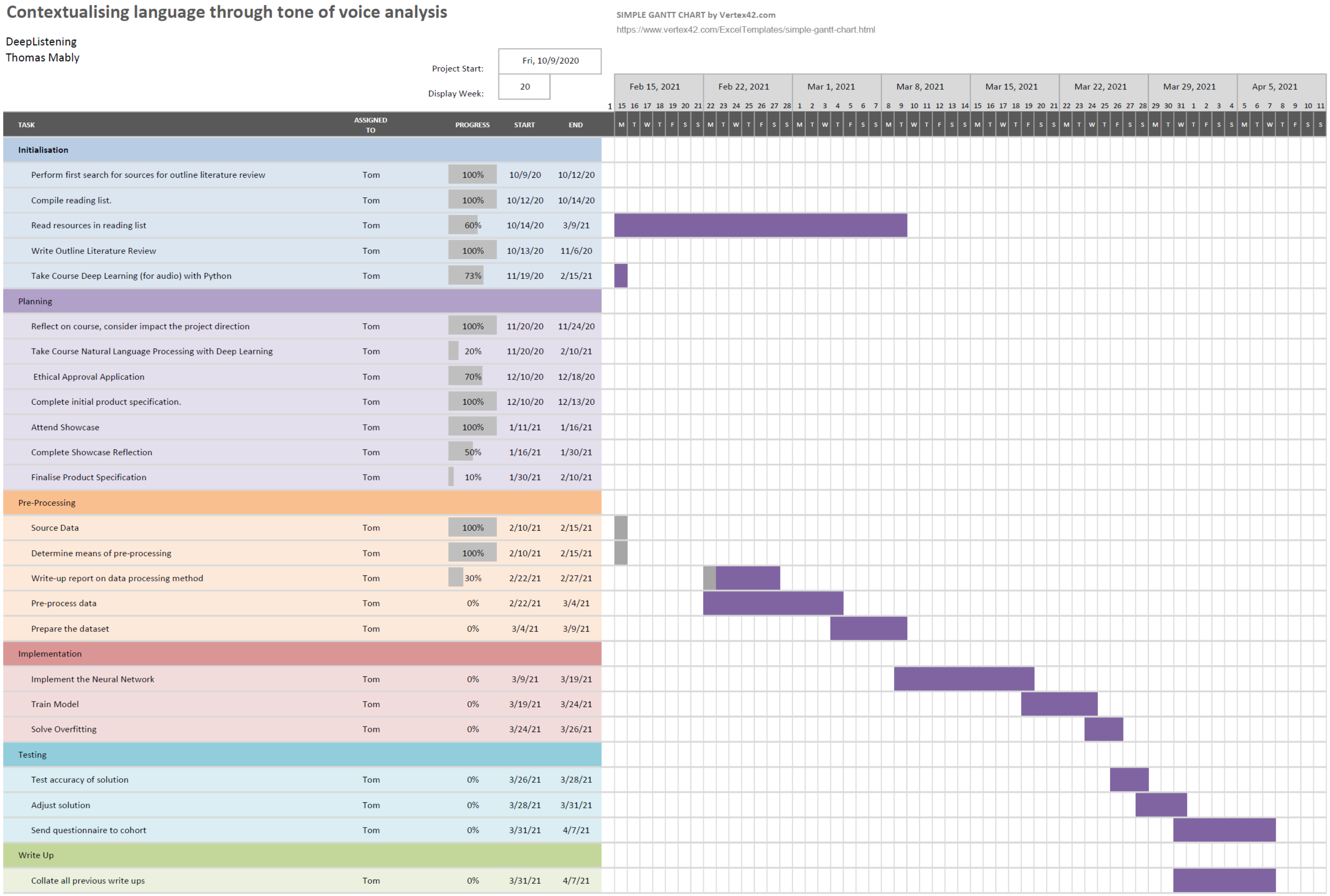
The Gantt chart shown in section 7.2 indicates that the project was well on track for completion before the deadline at the date shown (22nd February). However, around this time, I started facing personal issues that threw the project off course. As such, the project was delayed significantly, and progress was slow through march. By the beginning of April, I set aside time to work solely on the project and get it to a completed state. Unfortunately, as time ticked away, external factors made this an impossibility.

With the benefit of hindsight, I would have continued the consistent progress made in the initial months of the project. It is essential to work on research projects every week. Consistent progress over an extended period is the whole purpose of a research project such as this. Up until the progress started to slow down, I was thoroughly enjoying the research project.

Choosing a topic I had not worked on before greatly increased the difficulty of the project. Leveraging Deep Learning also required different testing, validation, and verification strategies than the more usual projects I have completed in the past. When compared to other projects with elements such as GUIs and databases, creating Deep Learning models offers less opportunity for the production of design documentation (Wireframes, UMLs, ERDs.) Instead, a lot of time in this project went into learning the methodology from courses (such as Velardo 2020) and continual reading (particularly of Russell and Norvig's (2020) *AI: A Modern approach*) to gain the background knowledge necessary to undertake the practical investigation. I hope to have documented the design and implementation process coherently, despite not having practice doing so for an artificial intelligence project.

Despite all the effort, trial-and-error and frustration that went into them, the resulting implementations may appear simplistic, as they have no front end. However, these models are successful solutions to a complex problem, achieved using languages and methods that were entirely unbeknown to me at the beginning of the project. For this reason, I am happy to have gone down this path and am proud to have something to show for it, considering the difficult circumstances.

7.2 Gannt Chart at Halfway Point



Contextualising language through tone of voice analysis

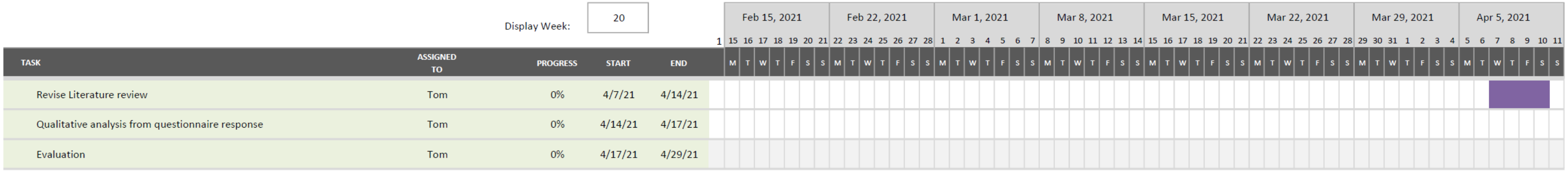
DeepListening
Thomas Mably

SIMPLE GANTT CHART by Vertex42.com
https://www.vertex42.com/ExcelTemplates/simple-gantt-chart.html

Project Start: Fri, 10/9/2020

Display Week: 20

TASK	ASSIGNED TO	PROGRESS	START	END
Initialisation				
Perform first search for sources for outline literature review	Tom	100%	10/9/20	10/12/20
Compile reading list.	Tom	100%	10/12/20	10/14/20
Read resources in reading list	Tom	60%	10/14/20	3/9/21
Write Outline Literature Review	Tom	100%	10/13/20	11/6/20
Take Course Deep Learning (for audio) with Python	Tom	73%	11/19/20	2/15/21
Planning				
Reflect on course, consider impact the project direction	Tom	100%	11/20/20	11/24/20
Take Course Natural Language Processing with Deep Learning	Tom	20%	11/20/20	2/10/21
Ethical Approval Application	Tom	70%	12/10/20	12/18/20
Complete initial product specification.	Tom	100%	12/10/20	12/13/20
Attend Showcase	Tom	100%	1/11/21	1/16/21
Complete Showcase Reflection	Tom	50%	1/16/21	1/30/21
Finalise Product Specification	Tom	10%	1/30/21	2/10/21
Pre-Processing				
Source Data	Tom	100%	2/10/21	2/15/21
Determine means of pre-processing	Tom	100%	2/10/21	2/15/21
Write-up report on data processing method	Tom	30%	2/22/21	2/27/21
Pre-process data	Tom	0%	2/22/21	3/4/21
Prepare the dataset	Tom	0%	3/4/21	3/9/21
Implementation				
Implement the Neural Network	Tom	0%	3/9/21	3/19/21
Train Model	Tom	0%	3/19/21	3/24/21
Solve Overfitting	Tom	0%	3/24/21	3/26/21
Testing				
Test accuracy of solution	Tom	0%	3/26/21	3/28/21
Adjust solution	Tom	0%	3/28/21	3/31/21
Send questionnaire to cohort	Tom	0%	3/31/21	4/7/21
Write Up				
Collate all previous write ups	Tom	0%	3/31/21	4/7/21



Display Week: 20

TASK	ASSIGNED TO	PROGRESS	START	END
Revise Literature review	Tom	0%	4/7/21	4/14/21
Qualitative analysis from questionnaire response	Tom	0%	4/14/21	4/17/21
Evaluation	Tom	0%	4/17/21	4/29/21

8 Conclusion

To conclude this paper, I will seek to answer my research questions.

8.1 Can emotions be accurately classified into discrete categories?

While the classification of emotions is complicated, it is possible, especially when looking at emotion classification for an individual. The task is entirely complex when attempting to generalise the classification model to work across cultures, differing social norms, and personality types. That said, if the training data is an accurate representation of genuine emotion, using a computer algorithm to detect patterns in this data is wholly unbiased by human perception. It seeks to find only the undeniable patterns that exist across the data set. Therefore, it is undoubtedly possible to somehow categorise emotion, but it is most likely that these categories need to be loose by design and not entirely discrete. The level of variability is too great to assert that, given a particular cue (frequency, speech rate), we can accurately label an emotional response. Still, these cues in combination, such as in patterns displayed in MFCC spectrograms, can evidently be discerned by a computer algorithm. As a computer can categorise these MFCCs with a high level of success, so long as the data supplied to these algorithms is verifiably representative of a discrete emotion, it is indeed possible to discretely categorise emotion so long as the theory of discrete emotions is taken as fact.

8.2 Are Deep Learning solutions for SER feasible for implementation in scenarios with limited computational and data resources?

Having performed the investigation offline solely running on a modest laptop, Deep Learning is entirely feasible for implementations outside the laboratory environment. As data science platforms move increasingly into the cloud, the dependency on data and hardware resource availability is likely declining. That said, achieving highly accurate systems requires a great deal of hyperparameter tuning. This is a case of trial and error, resulting in continuous model rebuilds. Therefore, it should be considered that implementing *optimal* deep learning systems of high complexity (say for less feature-rich input data) is very difficult in a home environment.

8.3 What challenges faced the implementation of SER into existing systems?

Both the CNN and LSTM-RNN models produced promising results across a limited dataset. That said, a result of 60% (achieved by the highly complex LTSM system) is potentially not suitable in real-world scenarios, particularly as the most powerful application for SER is in safety-critical scenarios such as healthcare. Considerable challenges exist, both from a theoretical standpoint (looking to section 2.4) and in terms of the practicality of implementing these systems in the real world. Solely looking at implementation, there has been substantial research in improving how SER systems work in real-time (Lakomkin et al. 2018) and how they can remain robust through adverse environmental factors, such as noise (Wijayasingha and Stankovic 2021). As the systems become more robust, they will likely become increasingly viable from a practical perspective. However, I've yet to find a system capable of recognising emotions with excellent accuracy across my full literature search. Any implementation of an SER system will likely be deployed at scale. The 2-4% error rate that is continually exhibited in the most successful literature will come to light for users at such a scale. However, inevitably, the trend for the accuracy of deep learning implementations is constantly upward. This, coupled with improved resilience, will likely see the deployment of a growing number of SER systems in the years to come.

9644 words

9 References

Aalto University (2020) *Waveform* Available at: https://wiki.aalto.fi/display/ITSP/Waveform (Accessed 21st May 2020)

Acoustical Surfaces (Unknown) *Acoustics 101 – Learn About Acoustics* Available at: https://images.app.goo.gl/oNnMFWocTvHAqMhP8 (Accessed 26th March 2021)

Amnesty International (2021) *Digital volunteers to expose scale of facial recognition technology in New York City* Available at: https://www.amnesty.org/en/latest/news/2021/05/decode-surveillance-nyc-launches/ (Accessed 5th May 2021)

Anaconda (2021) *Anaconda3* Available at: https://www.anaconda.com/ (Accesssed 16th December 2020)

Aquegg (2008) *Spectrogram-19thC* Available at: https://commons.wikimedia.org/wiki/File:Spectrogram-19thC.png (Accessed 12th May 2021)

Aurélien Géron (2019) *Hands-on machine learning with Scikit-Learn, Keras and TensorFlow: concepts, tools, and techniques to build intelligent systems. O'Reilly Media.*

Barrett, L.F. et al. (2019) *Emotional Expressions Reconsidered: Challenges to Inferring Emotion From Human Facial Movements. Psychological Science in the Public Interest.* [Online] 20 (1). Available from: doi:10.1177/1529100619832930.

Brownlee, J. (2018) *Difference between a Batch and an Epoch in a Neural Network* Available at: https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/#:~:text=The%20number%20of%20epochs%20is%20a%20hyperparameter%20that%20defines%20the,of%20one%20or%20more%20batches. (Accessed 20th May 2021)

Brownlee, J. (2019) *A Gentle Introduction to Padding and Stride for Convolutional Neural Networks* Available at: https://machinelearningmastery.com/padding-and-stride-for-convolutional-neural-networks/ (Accessed 20th May 2020)

Calder, A.J. et al. (2001) *A principal component analysis of facial expressions. Vision Research*. [Online] 41 (9). Available from: doi:10.1016/S0042-6989(01)00002-5.

Cherry, K. (2017) *What Are Emotions and the Types of Emotional Responses? Verywell.*

Digitalsreeni (2020) *135 - A quick introduction to Metrics in deep learning. (Keras & TensorFlow)* Available at: https://youtu.be/rHgQrdME-DA

Dupuis, K. and Pichora-Fuller, M. K. (2010) *Toronto emotional speech set (TESS)* University of Toronto, Psychology Department. https://doi.org/10.5683/SP2/E8H2MF

Fayek, H. (2016) *Speech Processing for Machine Learning: Filter banks, Mel-Frequency Cepstral Coefficients (MFCCs) and What's In-Between* Available at: https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html (Accessed 20th December 2020)

Costa, C. (2020) *Top Programming Languages for AI Engineers in 2021* Available at: https://towardsdatascience.com/top-programming-languages-for-ai-engineers-in-2020-33a9f16a80b0 (Accessed 20th December 2020)

Hershey, S. et al. (2017) *CNN architectures for large-scale audio classification*.In: *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*. [Online] Available from: doi:10.1109/ICASSP.2017.7952132.

Huang, J.J. & Leanos, J.J.A. (2018) *Aclnet: Efficient end-to-end audio classification CNN. arXiv.*

IBM Cloud Education (2020) *Machine Learning* Available at: https://www.ibm.com/uk-en/cloud/learn/machine-learning (Accessed 7th January 2021)

Interaction Design Foundation (Unknown) *Human-Computer Interaction* Available at: https://www.interaction-design.org/literature/topics/human-computer-interaction (Accessed 24th March 2021)

Jiaaro (2021) *Pydub* Available at: http://pydub.com/ (Accessed 20th May 2021)

Keras (2021) *Module: tf.keras.metrics* Available at: https://www.tensorflow.org/api_docs/python/tf/keras/metrics (Accessed 20th May 2021)

Khalil, R.A. et al. (2019*) Speech Emotion Recognition Using Deep Learning Techniques: A Review*. *IEEE Access*. [Online] 7. Available from: doi:10.1109/ACCESS.2019.2936124.

Lakomkin, E. et al. (2018) *EmoRL: Continuous acoustic emotion classification using deep reinforcement learning*.In: *Proceedings - IEEE International Conference on Robotics and Automation*. [Online] Available from: doi:10.1109/ICRA.2018.8461058.

Lei, X., Pan, H. & Huang, X. (2019) *A dilated cnn model for image classification. IEEE Access*. [Online] 7. Available from: doi:10.1109/ACCESS.2019.2927169.

Li, X. & Wu, X. (2015) *Constructing long short-term memory based deep recurrent neural networks for large vocabulary speech recognition*.In: *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*. [Online] 2015-August. Available from: doi:10.1109/ICASSP.2015.7178826.

Lim, N. (2016) *Cultural differences in emotion: differences in emotional arousal level between the East and the West. Integrative Medicine Research*. [Online] 5 (2). Available from: doi:10.1016/j.imr.2016.03.004.

Livingstone SR, Russo FA (2018) *The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS): A dynamic, multimodal set of facial and vocal expressions in North American English*. PLoS ONE 13(5): e0196391.

Mao, S., Ching, P.C. & Lee, T. (2020) *Emotion Profile Refinery for Speech Emotion Classification*. *arXiv*.

McFee, B. *et al.* (2015) *Librosa: Audio and Music Signal Analysis in Python.* In Proceedings of the 14th Python in science conference, pp. 18-25. 2015.

Microsoft (2021) *Visual Studio Code* Available at: https://code.visualstudio.com/ (Accessed 16th December 2020)

Mitsa T. (2019) *How Do You Know You Have Enough Training Data?* Available at: https://towardsdatascience.com/how-do-you-know-you-have-enough-training-data-ad9b1fd679ee#:~:text=Computer%20Vision%3A%20For%20image%20classification,%2Dtrained%20models%20%5B6%5D. (Accessed 24th March 2021)

Muccino (2020) *Image Classification with Variable Input Resolution in Keras* Available at: https://medium.com/mindboard/image-classification-with-variable-input-resolution-in-keras-cbfbe576126f

Murgia, M. (2021) *Emotion Recognition: can AI detect human feelings from a face?* Available at: https://www.ft.com/content/c0b03d1d-f72f-48a8-b342-b4a926109452 (Accessed 12th May 2021)

Nassif, A.B. et al. (2019) Speech Recognition Using Deep Neural Networks: A Systematic Review. *IEEE Access*. [Online] 7. Available from: doi:10.1109/ACCESS.2019.2896880.

Niu, Y. et al. (2017) *A breakthrough in speech emotion recognition using deep retinal convolution neural networks. arXiv*.

Numpy (2021) *Discrete Fourier Transform* Available at: https://numpy.org/doc/stable/reference/routines.fft.html (Accessed 20th December 2020).

Oxford Union (2019) *Artificial Intelligence: An Inhuman Future?*

Ozseven, T. (2019) *Human-computer interaction. Human-Computer Interaction*. [Online] Available from: doi:10.1145/1226736.1226761.

Qin, J. et al. (2020) *A biological image classification method based on improved CNN. Ecological Informatics.* [Online] 58. Available from: doi:10.1016/j.ecoinf.2020.101093.

Ramesh, S. (2018) *A guide to an efficient way to build neural network architectures- Part II: Hyper-parameter selection and tuning for Convolutional Neural Networks using Hyperas on Fashion-MNIST* Available at: https://towardsdatascience.com/a-guide-to-an-efficient-way-to-build-neural-network-architectures-part-ii-hyper-parameter-42efca01e5d7 (21st May 2021)

Sapiński, T. et al. (2019) *Multimodal Database of Emotional Speech, Video and Gestures*.In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics).* [Online] 11188 LNCS. Available from: doi:10.1007/978-3-030-05792-3_15.

Schuller, B.W. (2018) *Speech emotion recognition: Two decades in a nutshell, benchmarks, and ongoing trends. Communications of the ACM.* [Online] 61 (5). Available from: doi:10.1145/3129340.

Scikit-Learn (2021) *SciKit-Learn* Available at: https://scikit-learn.org/stable/ (Accessed 16th December 2020)

Shu, L. et al. (2018) *A review of emotion recognition using physiological signals. Sensors (Switzerland).* [Online] 18 (7). Available from: doi:10.3390/s18072074.

Tanaka, A. et al. (2010*) I feel your voice: Cultural differences in the multisensory perception of emotion. Psychological Science*. [Online] 21 (9). Available from: doi:10.1177/0956797610380698.

TensorFlow (2021) *Tensorflow 3* Available at: https://www.tensorflow.org/ (Accessed 16th December 2020)

UCSB Social Computing Group (2020) What is Social Computing? Available at: http://socialcomputing.ucsb.edu/index8067.html?page_id=14 (Accessed 24th March 2021)

Valerdo, V. (2020) *Deep Learning (for audio) with Python* Available at: https://www.youtube.com/watch?v=fMqL5vckiU0&list=PL-wATfeyAMNrtbkCNsLcpoAyBBRJZVlnf&ab_channel=ValerioVelardo-TheSoundofAI (Accessed 16th December 2020)

van Houdt, G., Mosquera, C. & Nápoles, G. (2020) *A review on the long short-term memory model. Artificial Intelligence Review.* [Online] 53 (8). Available from: doi:10.1007/s10462-020-09838-1.

Vogt, T., André, E. & Wagner, J. (2008) *Automatic recognition of emotions from speech: A review of the literature and recommendations for practical realisation*.In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics).* [Online] 4868 LNCS. Available from: doi:10.1007/978-3-540-85099-1_7.

Wijayasingha, L. & Stankovic, J.A. (2021) *Robustness to noise for speech emotion classification using CNNs and attention mechanisms. Smart Health*. [Online] 19. Available from: doi:10.1016/j.smhl.2020.100165.

Zhang, P. et al. (2019) *SR-LSTM: State refinement for lstm towards pedestrian trajectory prediction*.In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. [Online] 2019-June. Available from: doi:10.1109/CVPR.2019.01236.

10 Appendices

10.1 Project Proposal Form

Milestone 3: Computing Projects Proposal Form 2020/21

See the Computing Projects Blackboard VLE for further guidance on how to complete this form.

You can change the sizes of the boxes.

A STUDENT DETAILS (to be completed by the student)			
Name:	Thomas Mably		
University email:	Mabt1_17@uni.worc.ac.uk		
Student Number:	17006297	**Module code:**	COMP3008
Project Completion Date:	15.00 Thursday 29th April 2021	**Pathway:**	Computing
Part time / Full time:	Full Time	**Year started course:**	2018
Single/Major/Joint Honours:	Single	**If joint/major, other subject:**	

B PROJECT DETAILS (to be completed by the student from milestone 1)	
Supervisor:	Pete Moody
Project Title:	Mood Detection Through Natural Language Processing
Type of Project:	Design-Build-Test (DBT)
Project Aims: (*A broad statement of the expected outcomes)*	Analyse vocal data to determine a set of rules used discern a person's mood from their tone of voice. Create an Expert System artificial intelligence, based on the ruleset gathered from analysis. Use machine learning to improve the accuracy of mood recognition. Discuss how this technology could be used in a business context.

Research Question(s):	Can someone's mood be discerned by computational analysis of waveforms? Can the utilisation of machine learning improve a working Expert System solution? For an artificial intelligence, what are the difficulties in the production and understanding of human language?
Proposed Primary Research	Taking samples of the speech of people in different moods. Analysing the waveforms to compile a set of rules. Composing an expert system solution capable of determining mood. Using data in the public domain to collate a large data set for each class of mood. Using this data set in a machine learning solution to improve the expert system model.
Proposed Secondary Research	Defining Artificial Intelligence, Machine Learning and Deep learning. Comparing their relationship and which is most suitable for this project. Discussing the Composition of an artificial neuron and how it compares to a biological neuron. Discuss how a solution built of neurons will differ from more traditional programming paradigms. A discussion of how Natural Language processing is a challenging computational task. Discussion of how the ambiguity and imprecision of natural language are at odds with more easily computable tasks. Key Words: Deep Learning, Natural Language Processing, Mood Classification, Machine Learning, Artificial Intelligence, Rational Agents. Courses to be followed: Deep Learning (for audio) with Python, Valerio Velardo, University of Huddersfield https://www.youtube.com/watch?v=fMqL5vckiU0&list=PL-wATfeyAMNrtbkCNsLcpoAyBBRJZVlnf&ab_channel=ValerioVelardo-TheSoundofAI Natural Language Processing with Deep Learning, Winter 2019, Professor Christopher Manning, Stanford University https://www.youtube.com/watch?v=8rXD5-xhemo&ab_channel=stanfordonline Deep Learning State of the Art, Lex Fridman, MIT https://www.youtube.com/watch?v=0VH1Lim8gL8&ab_channel=LexFridman Key Resources: Russell, S, Norvig, P. (2009) *Artificial intelligence — A Modern Approach*. Pearson. This text is considered a go to textbook for Undergrad/Masters level study in Artificial Intelligence.

C OUTLINE LITERATURE REVIEW (from milestone 2)

Artificial Intelligence

Defining Artificial Intelligence

An artificial intelligence is an intelligent agent, capable of perceiving and acting on its environment (Russell and Norvig, 2009). An intelligent agent will direct its activity to achieve a goal, acting in its best interests. These agents are highly rational, acting based on its own morals (ruleset) whilst attaining its goal.

The subsets of artificial intelligence

The subsets of artificial intelligence vary in how these rulesets are constructed. The following are some examples of these subsets:

Evolutionary Algorithms

Evolutionary algorithms. Stemming from biology, this approach is highly randomised. Implementation is suitable in situation where a finding an optimal solution is too computationally intensive and where a near-optimal solution is sufficient (Dyer, 2010).

Expert Systems

The ruleset and heuristics are handcrafted by numerous experts to collate their knowledge in the hope of creating an intelligence with superhuman reasoning on a particular subject (Silver et al, 2018). This model is usually made up of many procedural functions, who are actioned by the intelligence if certain conditions are met. IBM’s Deep Blue (1997) was an early example of an expert system performing at a superhuman level in a particular field. Deep Blue managed to beat chess grandmaster Garry Kasparov in a series of six chess matches.

Machine Learning

Although sometimes used interchangeably, Artificial Intelligence is the larger set, Machine learning is a part of AI and deep learning is a subset of machine learning (Velardo, 2020).

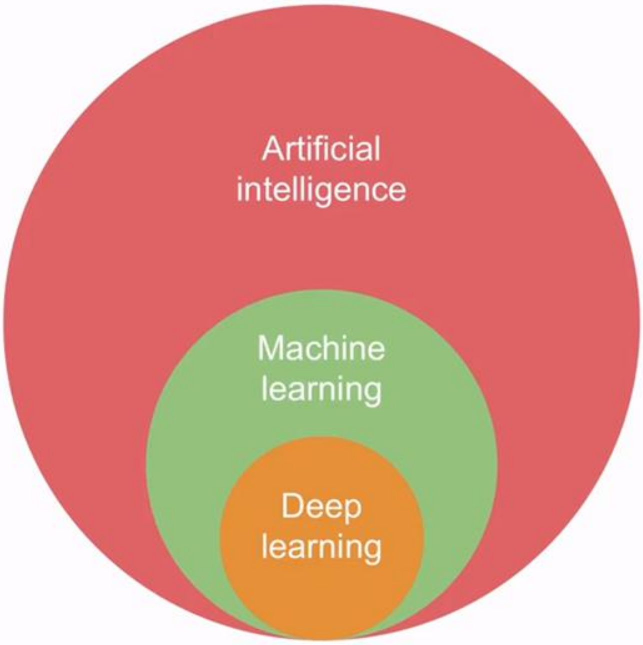


Figure 35. (Velardo, 2020) A model of Deep Learning and Machine Learning as subsets of artificial intelligence Machine learning has a computer that performs a task without using explicit instructions. As researchers and developers, we do not give specific rules, we allow these to be discerned by the agent itself.

The Paradigms of Machine Learning

Supervised learning

A model is supplied with both unlabelled and labelled data. The goal for the algorithm is to learn the operation that links the unlabelled data to the desired result. Once an approximation is made for this function, the mapping can be applied to other data to predict the output (Brownlee, 2020). As we have data that is congruent with achieving the result, providing more labelled data allows the ML algorithm to adjust its functionality.

Unsupervised learning

This is used when there is no availability of structured (labelled) data. In this case the machine learning algorithm will go over datasets, without telling the algorithm what the data is. The system will analyse this unstructured data and create clusters (differentiation) between the data. In other words, the algorithm can learn to separate the data without knowing the labels beforehand.

Reinforcement learning

A branch of machine learning where the learning occurs through environmental interaction. Rational agents are not taught what action to take, instead learning from the consequences of its actions within the environment (Ravichandiran, 2018). Each action will result in the agent receiving either a positive or negative ‘reward’. As a rational agent, it will act in its own best interest, thus performing actions that resulted in a positive reward. As such, it is a trial-and-error process. An example of the power of reinforcement learning is AlphaZero, a project of Deepmind. As opposed to IBM’s deep-blue chess engine, built on the expert system model, AlphaZero replaces hand crafted rules and heuristics drawn from the knowledge of experts, with a neural network with general purpose algorithms (Silver et al, 2018). To master chess, an untrained neural network will play millions of games against itself, receiving a reward per action as per reinforcement learning. Eventually, the moves that the engine shifted from random to considered, as the system adjusted the parameters of the neural network based on what moves resulted in positive rewards, and which brought negative consequences. AlphaZero outperformed world computer chess champion Stockfish after just 4 hours of reinforcement learning.

Artificial Intelligence vs Machine Learning, an example.

In this example (Velardo, 2020) we will examine two solutions for onset detection. This is the detection of when a musical note begins. First, we will create a solution using an Expert System artificial intelligence, where the AI is given a set of rules, rather than coming to these rules on its own accord.

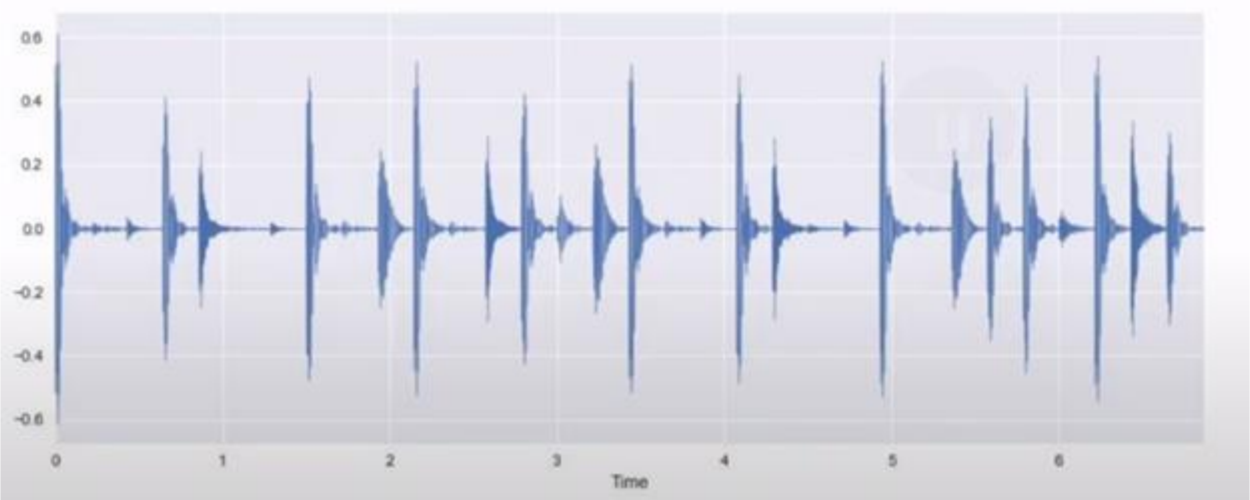


Figure 36. (Valerdo, 2020) An unlabelled waveform

Looking to Figure 2, where there is a burst of amplitude, there is an onset. To solve this problem with an expert system, we would define certain rules into the system for the identification of an onset. For example, where the amplitude exceeds the bounds of 0.2 or -0.2, then we have an onset. Figure 3 shows this rule superimposed onto the waveform.

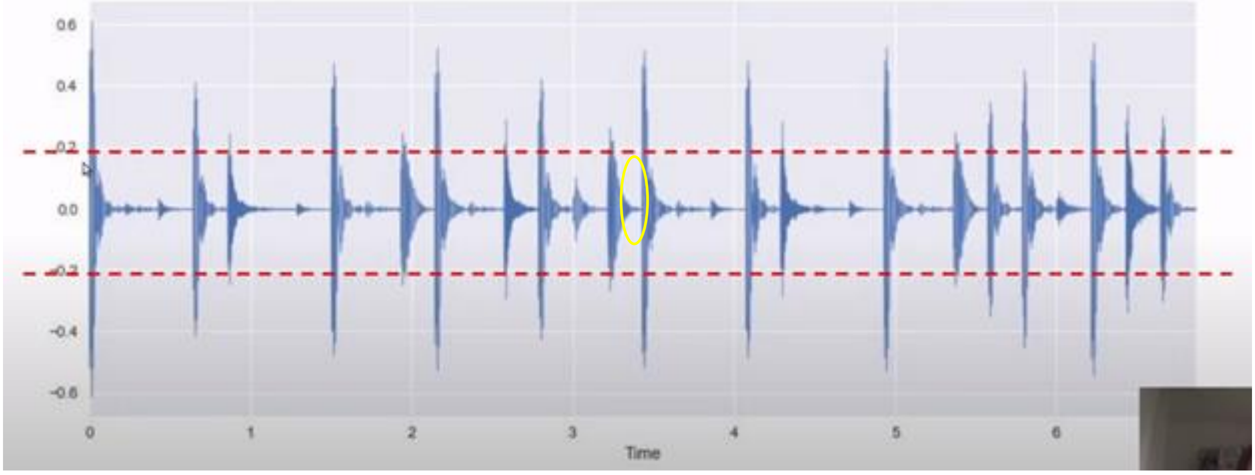


Figure 37. (Valerdo, 2020) A waveform to be bounded by amplitude values -0.2 - 0.2. Waves outside of these values are expected to indicate the onset of a note.

Looking to figure 3, as humans we can discern that certain onsets will be missed by this rule. However, an expert system with a single rule would have missed the smaller outburst of energy circled. This is an extreme simplification of an expert system; whose rule sets are usually far larger.

Now let us take the same problem but approach a solution using machine learning. We can provide numerous waveforms, whose onsets have been pre-processed (labelled). This data is then passed to a machine learning system, which derives a set of rules from this data. This ruleset is not imposed by human influence, instead only by the labelled data which is provided to the system. This process is the essence of supervised machine learning.

Deep learning

Defining Deep Learning

Deep learning vs Machine Learning

When should you use Deep Learning over Machine Learning?

Using an exceptionally large dataset.

If you are tackling a complex problem where traditional ML would fail.

Where you have access to extensive computational resources.

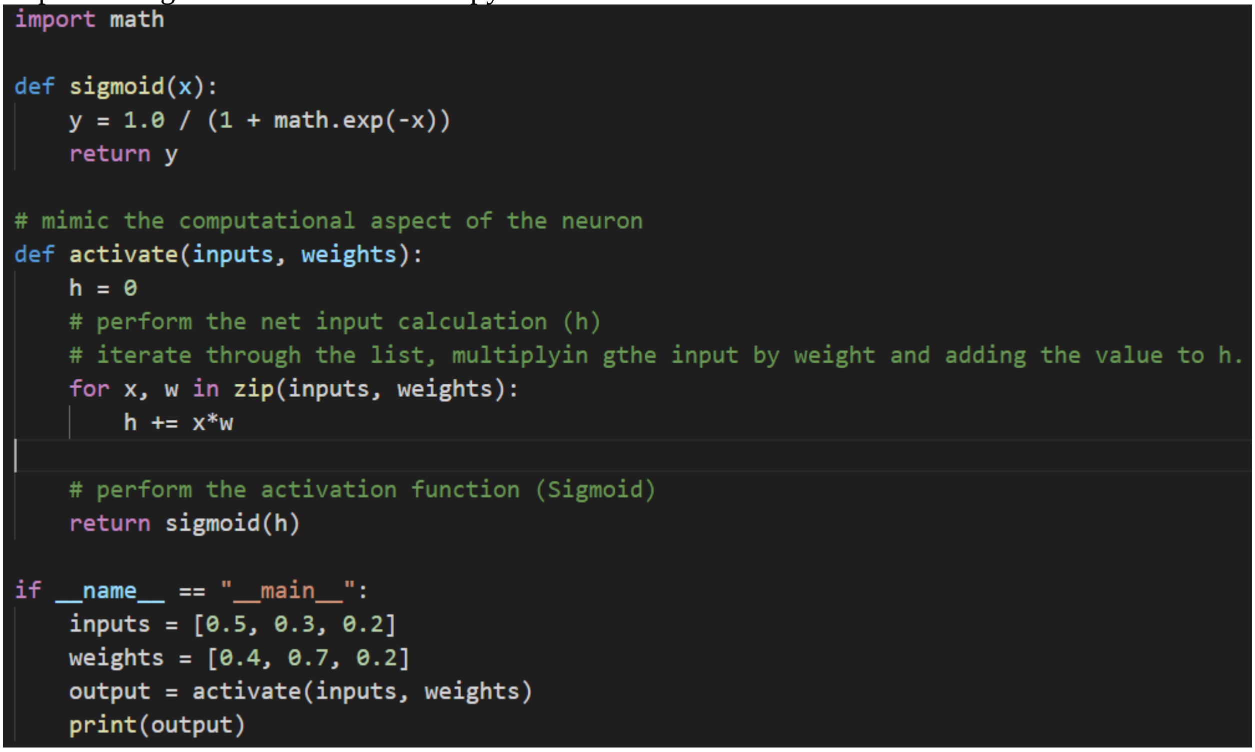
The Fundamentals of Neural Networks

Biological Neurons

Artificial Neurons

The mathematics behind the artificial neuron

Implementing an artificial neuron with python



```
import math

def sigmoid(x):
    y = 1.0 / (1 + math.exp(-x))
    return y

# mimic the computational aspect of the neuron
def activate(inputs, weights):
    h = 0
    # perform the net input calculation (h)
    # iterate through the list, multiplyin gthe input by weight and adding the value to h.
    for x, w in zip(inputs, weights):
        h += x*w

    # perform the activation function (Sigmoid)
    return sigmoid(h)

if __name__ == "__main__":
    inputs = [0.5, 0.3, 0.2]
    weights = [0.4, 0.7, 0.2]
    output = activate(inputs, weights)
    print(output)
```

Computation in Neural Networks
Explanation of neural networks

Implementing a Neural Network

Technologies Used
The technologies we will be using are python and TensorFlow. These are industry standard for artificial intelligence. TensorFlow is used everywhere from academia to corporations. On top of TensorFlow, you have a high-level interface called Keras that lets you create complex networks using small amounts of code. Furthermore, tensor flow is open source.

Comparing Industry Standard Technologies

Training a neural Network

Pre-processing Audio Data for Deep Learning
Spectrograms
Preparing Datasets

References

Ravichandiran, S. (2018) *Hands-On Reinforcement Learning with Python : Master Reinforcement and Deep Reinforcement Learning Using OpenAI Gym and TensorFlow* Packt Publishing, Limited, Birmingham.

Russell, S, Norvig, P. (2009) *Artificial intelligence — A Modern Approach.* Pearson.

Silver et al. (2018) *AlphaZero: Shedding new light on chess, shogi and Go* Available at: https://deepmind.com/blog/article/alphazero-shedding-new-light-grand-games-chess-shogi-and-go (Accessed 16th December 2020)

Dyer, D. (2010) *Evolutionary Computation in Java* Available at: https://watchmaker.uncommons.org/manual/ch01s02.html (Accessed 16th December 2020)

IBM (2011) *Deep Blue* Available at: https://www.ibm.com/ibm/history/ibm100/us/en/icons/deepblue/ (Accessed 16th December 2020)

Brownlee, J. (2020) *Supervised and Unsupervised Machine Learning Algorithms* Available at: https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/ (Accessed 16 December 2020)

Valerdo, V. (2020) *Deep Learning (for audio) with Python* Available at: https://www.youtube.com/watch?v=fMqL5vckiU0&list=PL-wATfeyAMNrtbkCNsLcpoAyBBRJZVlnf&ab_channel=ValerioVelardo-TheSoundofAI (Accessed 16th December 2020)

D SMART Project Objectives

A list of Specific Measurable Achievable Realistic and Time-Bound objectives
These will obviously include all the milestones, but further objectives will be required, particularly between milestones 5 and 7. Most of them will appear on your Gantt chart. Examples might include:

Complete first search for sources for outline literature review.
Compile reading list.
Begin reading sources on reading list.
Begin Course Deep Learning (for audio) with Python.
Complete Outline Literature Review using knowledge gained from reading list and partial course completion.
Complete Course Deep Learning (for audio) with Python.
Reflect on course, consider how knowledge gained will impact the project direction.
Begin Course Natural Language Processing with Deep Learning
Complete reading (and making notes on) sources for literature review
Complete Ethical Approval Application.
Complete Gantt chart
Complete initial product specification.
Attend Showcase
Reflect on showcase to impact the planning for the first prototype.
Finish Course Natural Language Processing with Deep Learning
Reflect on course, consider how new knowledge will impact first prototype.
Begin the pre-processing of data to determine a rule set for an expert system.
Continue to expand knowledge through MIT videos.
Devise Expert system
Test Expert System using help of other students.
Label data for machine learning.
Complete Literature Review.
Improve the Solution using machine learning.
Complete Final Prototype.
Test Final Prototype.
Reflect on improvements between first prototype (Expert System) and second (Machine Learning)
Discuss findings in report.
Complete written report
(There is no need to demonstrate that all your objectives are SMART.)

E REQUIREMENTS (A list of resources you will need such as software, hardware, data sources etc.)

<table>
<tr><td colspan="2">Software – Python, Tensorflow, Keras, PyTorch (all open-source).
Hardware – Capable computer with powerful GPU (one I own should handle this project), Microphone (already owned)
Data Sets – I hope to train this application using audio samples from the public domain. I intend to ask a several subjects to interact with the application during development and testing.
Supervision – Weekly Meeting to keep the project on track and to discuss finding of my research.</td></tr>
<tr><td><strong>Agreed Word Count</strong></td><td>Max 9000 words</td></tr>
</table>

10.2 Grading Matrix

University of Worcester

This matrix captures the assessment criteria for the Design-Build-Test Computing Project Assessment.

Student Name:	**Student Number:**	**Academic Year:** 2020/2021	**Module Code:** COMP300x
Assessment Criteria **1.** Demonstrate a critical understanding of a significant computing issue aligned with the degree pathway [LO1]. **2.** Provide a comprehensive critical review of relevant and contemporary literature [LO2]. **3.** Design and construct an appropriate software or hardware computing artefact [LO3]. **4.** Gather and analyse original data [LO4]. **5.** Provide an accurate and critical evaluation of project outcomes against the stated objectives [LO5]. **6.** Provide a personal assessment of the project contribution to addressing a significant computing problem and a reflection on your management of the project [LO6].			**Assignment Description:** Project Report + Artefact **Assignment Weighting:** 100% **Module Title:** Computing Project

Assessment Criteria						
	Demonstrate a critical understanding of a significant computing issue aligned with the degree pathway	**Provide a comprehensive critical review of relevant and contemporary literature**	**Design and construct an appropriate software or hardware computing artefact informed by primary research (requirements elicitation) and/or secondary research and/or critical literature review**	**Gather and analyse primary data, either using the Artefact, and/or regarding the design (requirements elicitation) and/or performance (user testing) of the Artefact**	**Provide an accurate and critical evaluation of the Artefact against the stated design objectives or requirements specification**	**Provide a personal assessment of the project contribution to addressing a significant computing problem and a reflection on your management of the project**

For students: **The sort of stuff that could be included** These lists are indicative (suggestions). They are not prescriptive (you have to do them all) or exhaustive (include everything you should do)	**Why am I doing this project? Why is it useful? What problem is it solving? How does it link to my specialist pathway and/or employment aspirations? Research question(s)**	**Literature review; citations; reference list; what other work has been done in this area? (Do authors agree with each other?) How does my proposed work link in with this? Best practice**	**Planning, including Gantt chart; methodology; Project aims; Project objectives; product specification; strategy; justification of strategy; design documentation (wireframes, DFDs, ERDs, etc.); commentary on the design and construction process; limitations**	**Data acquisition; justification of data gathering technique(s); data description; data analysis; conclusions based on data; limitations of data gathering techniques, data gathered and conclusions reached**	**Does the project achieve its aims and meet its objectives? What is/are the answer(s) to the research question(s)? Do the findings agree with previous work? Limitations, how might they be overcome in the absence of resourcing restrictions? Suggestions for further work**	**What did the project achieve? What did I learn from doing this project, from my supervisor/ other students/the showcase/…? What went well? Why? What didn't work? Why? How might it be done even better next time? Deviations from planning**
Approximate weighting (%)	**20**	**20**	**20**	**20**	**10**	**10**
A	1. An original research question is formulated independently and without ambiguity and is clearly linked to the investigation of a significant computing problem area. 2. A persuasive demonstration of the nature and significance of the problem is presented articulately.	1. A comprehensive and thorough literature review is presented which includes relevant principles, concepts and research performed in the area of the study. 2. The literature review is systematic, articulate and ordered in a clear and coherent fashion. Clear and correctly formatted referencing is used throughout.	1. A complete and precise system requirements specification is formulated from an articulate synthesis of a full range of functional and non-functional requirements. 2. A significant and high quality artefact has been created. 3. A systematic and coherent approach to designing an artefact to address a specification is documented. 4. The development process of an original artefact has been clearly and thoroughly documented.	1. Primary data is gathered systematically. 2. The findings are clearly described and subject to close critical analysis. The analytical method used is clearly demonstrated to be appropriate. 3. A coherent interpretation based on a synthesis of a range of results leads to a reasoned and well-articulated critical conclusion.	1. An articulate and critical evaluation of the Artefact is undertaken independently and with full consideration of the specification. 2. The evaluation is clearly presented within the context of the research objectives, with clear outcomes demonstrated for each objective. 3. A detailed and thorough critical reflection on the effectiveness of the Artefact in addressing the project objectives is clearly demonstrated.	1. An articulate and persuasive argument for the nature and value of the contribution of the project outcomes is made in the context of related literature and theory. 2. An assessment of the management and planning of the project is present in the form of a critical analysis of the initial project plan with the actual project life cycle.

	3. Recognition of potential contribution to the broader context and theory of the subject discipline (degree pathway) is evident.	3. A full range of contemporary and relevant sources is used. Sources largely consist of high quality and reliable sources such as peer reviewed academic articles. 4. The literature review demonstrates an ability to synthesise available evidence and to evaluate conflicting interpretations to reach a critical, independent, resolution.	5. A systematic and coherent justification for the design of the system has been provided. This will include a critical and persuasive argument demonstrating the value of formulating a detailed system requirements specification in the context of the proposed design. 6. Challenges for the design and implementation of the Artefact are acknowledged and addressed.	4. Specific limitations of the data gathering and analytic approach are acknowledged and addressed.	4. Limitations of the design-build-test process and the final Artefact are acknowledged and addressed through critical reflection. 5. A clear and insightful account of how future work might improve or build upon this process and product is provided.	3. An informed and honest personal reflection of progress on the project overall with key challenges identified and potential improvements outlined.
B	1. A research question is formulated without ambiguity and is clearly linked to the investigation of a significant computing problem area. 2. A clear demonstration of the nature and significance of the problem is presented effectively. 3. Recognition of potential contribution to the subject discipline (chosen degree pathway) is evident.	1. A comprehensive literature review is presented which includes relevant principles, concepts and research performed in the area of the study. 2. The literature review is well structured and presented in a clear and coherent fashion. Clear and correctly formatted referencing is used.	1. A system requirements specification is formulated coherently. A range of functional and non-functional requirements is defined 2. A significant artefact has been created. 3. The design of the Artefact to address a specification is clearly documented. 4. The development process of an original artefact has been documented fully. 5. A critical justification for the design of the system has been provided.	1. Primary data is gathered. 2. The findings are clearly described and subject to close analysis. The analytical method used is demonstrated to be appropriate. 3. A coherent interpretation of the findings leads to a reasoned critical conclusion. 4. Some limitations of the data gathering and/or analytic approach are acknowledged.	1. A systematic evaluation of the Artefact is undertaken with full consideration of the specification. 2. The evaluation is presented within the context of the research objectives, with clear outcomes demonstrated for each objective. 3. A critical reflection on the effectiveness of the Artefact in addressing the project objectives is clearly demonstrated.	1. A coherent argument for the nature and value of the contribution of the project outcomes is made. 2. An assessment of the management and planning of the project is present in the form of a critical comparison of the initial project plan with the actual project life cycle. 3. An informed and honest personal reflection of progress on the project overall with some key challenges identified.

		3. A range of contemporary and relevant sources is used. Sources largely consist of high quality and reliable sources such as peer reviewed academic articles. 4. The literature review demonstrates an ability to synthesise available evidence to reach a critical resolution.	6. Some challenges and limitations of the design and implementation of the Artefact are acknowledged and addressed.		4. Some limitations of the design-build-test process and the final Artefact are acknowledged and addressed through reflection. 5. Some account of how future work might improve or build upon this process and product is provided.	
C	1. A research question is formulated and is clearly linked to the investigation of a significant computing problem area. 2. A clear demonstration of the nature and significance of the problem is presented appropriately. 3. Relevance to the subject discipline (chosen degree pathway) is strongly evident.	1. A literature review is presented which covers a range of relevant topics. 2. The literature review is presented in a clear and coherent fashion. Format of referencing is used correctly is most cases. 3. A range of relevant sources is used including peer reviewed academic articles. 4. The literature review demonstrates an ability to collate a variety of theories or opinion and present an informed argument.	1. A system requirements specification is clearly defined and includes some functional and non-functional requirements. 2. A functional artefact has been created. 3. The design of the Artefact to address a specification is evident. 4. Documentation of the development process of an artefact is evident but may be lacking some details. 5. Some justification for the design of the system has been provided. 6. No assessment of limitations of the process.	1. Primary data is gathered. 2. The solution or findings are described and subject to analysis. 3. An interpretation of the findings leads to a reasoned conclusion. 4. An attempt to state some limitations of the data gathering and/or analytic approach	1. A comprehensive evaluation of the Artefact is undertaken with consideration of the specification and research objectives. 2. A conclusion is reached regarding the evaluation that addresses most requirements. 3. Some reflection on the system specification, design and development process is evident. 4. Some key challenges and limitations are identified and discussed. 5. No indication of how the work may be developed.	1. An argument for the nature and value of the contribution of the project outcomes is made. 2. An assessment of the management and planning of the project is present in the form of a systematic comparison of the initial project plan with the actual project life cycle. 3. An informed and honest personal reflection of progress is discussed.

D	1. A research question is formulated and is aligned with a computing problem. 2. A demonstration of the nature of the problem is presented. 3. Relevance to the subject discipline (chosen degree pathway) is evident.	1,3. A literature review is presented which includes some relevant sources. 2. Some attempt to format references correctly is evident. 4. The literature review demonstrates an ability to collate and present a variety of theories or opinions evident from within the related literature.	1. An attempt to formulate a requirements specification is evident but may not be complete. 2. A limited but functional artefact has been created. 3. Some attempt to design the Artefact with consideration of the specification is evident. 4. Some documentation of the development process of an artefact is evident but may be incomplete. 5,6. No justification for the design or consideration of the limitations of the process.	1. Some primary data is gathered. 2. An attempt to describe and analyse the primary data is demonstrated. 3. An interpretation of the findings leads to a conclusion. 4. No attempt to consider limitations of the data gathering and analysis	1. An evaluation of the Artefact is undertaken with some consideration of the specification and research objectives. 2. A conclusion is reached regarding the evaluation that addresses some requirements. 3. An attempt to reflect on the system specification, design and development process is evident but may be lacking in detail. 4. No consideration of limitations of the design-build-test process or the resulting Artefact 5. No indication of how the work may be developed.	1. An attempt to define the contribution of the project is evident. 2. A coherent description of the management and planning of the project is present. 3. A reflection of progress is discussed.
Narrow Fail (E)	1,2,3. A problem area is defined but a specific research questions is lacking.	1,2,3. A literature review is presented which includes a restricted range of sources. 4. The literature review demonstrates a limited ability to collate and present a variety of theories or opinions evident from within the related literature. Recapitulation of original sources may contain factual errors.	1. Limited attempt to formulate a requirements specification. 2. An attempt to create an artefact is evident but it may not be fully functional. 3. Limited attempt to design the Artefact with consideration of the specification is evident. 4. Documentation of the development process is largely incomplete.	1. Limited primary data is gathered. 2. Limited and unsubstantiated attempt to describe and analyse the primary data is evident. 3. A conclusion based on the analysis of primary data is evident. 4. No attempt to consider limitations of the data gathering and analysis	1,2,3. An attempt to evaluate the Artefact is evident but may be incomplete or lacking consideration of the specification and research objectives. 4. No consideration of limitations of the design-build-test process or the resulting Artefact 5. No indication of how the work may be developed.	1. There is limited attempt to define the contribution of the project. 2. A description of the management and planning of the project is present. 3. No reflection of personal progress.

			5,6. No justification for the design or consideration of the limitations of the process.			
Clear Fail (F-G)	1,2,3. A problem area is not clearly defined. **Symptoms may include:** • No research question or defined 'deliverable' • No purpose for the work - who will benefit? • It is not clear what is to be done (at the start) and/or what has been done (at the end) • No links to Computing degree	1,2,3,4. No literature review is evident or it lacks sufficient sources of appropriate quality. **Symptoms may include:** • Only a small area covered • Depth very restricted • Sources used are mainly web sites • No obvious structure • Information from sources is presented, but not evaluated • No 'compare and contrast' between sources	1. No attempt to formulate a requirements specification. 2,3,4,5,6. No serious attempt to design and/or implement an artefact is evident. **Symptoms may include:** • No requirements specification • Missing, trivial or non-functional Artefact • Very limited design documentation • No explanation of or justification for the development process and tools used • No evaluation of the design and development process	1,2. No attempt to gather and/or describe primary data is evident. 3,4. No conclusion based on the analysis of primary data is provided. **Symptoms may include:** • Primary data is absent or trivial • Treatment of data is descriptive, and not analytical • Conclusions are absent or trivial or unjustified • No evaluation of the data gathering and processing - how could they have been improved?	1,2,3. No evaluation of the Artefact is evident. 4,5. No attempt to assess limitations or potential for development. **Symptoms may include:** • No evidence of testing of the Artefact (unit testing, system testing, user testing) • No comparison of the Artefact with the design specification • No evaluation of the design-build process- how might it have been improved? • No suggestions for improving or developing the Artefact	1. There is no attempt to define the contribution of the project. 2. No description of the management and planning of the project is present. 3. No personal reflection. **Symptoms may include:** • No evaluation of the project outcomes against the objectives - were they achieved? • No evidence of planning, or evaluation of actual progress of project against the plan • No personal action plan: what have I learnt? What went wrong? Why? How am I going to do it better next time?

Ethics Procedures

Although not part of the explicitly assessed work in a Computing Project, students are expected to engage with the University 's ethics approval process.

Failure to do so constitutes academic misconduct and carries various penalties.

Ethics Process: Supervisors, please insert a tick (✓) in the coloured box next to the appropriate description of the student's ethics engagement.

If you place a tick in any of the red boxes, the grade for the project should be entered as 'SM' (Suspected Misconduct), and the Academic Integrity Tutor should be informed.

Ethics process completed, and project approved. All changes (e.g. modified questionnaires) also approved.	🟩
Ethics process **partly completed**, low-risk project (not involving human participants)	🟥
Ethics process **partly completed**, high-risk project (human participants)	🟥
No engagement with ethics process, low-risk project (not involving human participants)	🟥
No engagement with ethics process, high-risk project (human participants)	🟥
If the ethics process was only **partly completed**, please indicate the extent of the student's engagement: • what was done • what more should have been done	

Overall Comments:

Recommendation for future assignments:

Employability & Engagement:

Agreed Assignment Grade:	**First Marker:**	**Second Marker*:***	**Third Marker:** *if required*

RESULTS ARE PROVISIONAL UNTIL AGREED BY THE BOARD OF EXAMINERS