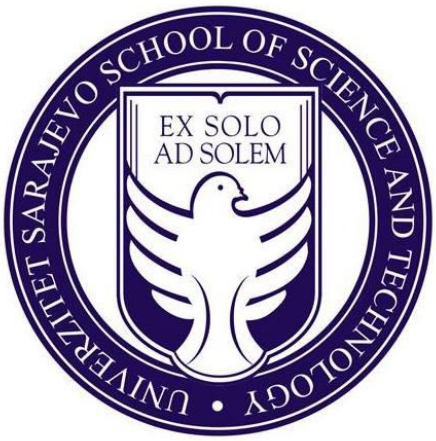
University Sarajevo School of Science and Technology

Department of Computer Science



Speech Recognition and Speech Emotion Recognition

Undergraduate BSc Thesis in Computer Science

Submitted in partial fulfillment of the requirements for graduation in
the Undergraduate Programme

Student name: Neuman Alkhalil (18CI-NA-0080)

Thesis mentor: Associate Professor Jasminka Hasić-Telalović

Sarajevo, July, 2022

ABSTRACT

The purpose of this thesis is to acquire in depth knowledge in the fields of Speech Recognition (SR), Speech Emotion Recognition (SER), Machine Learning (ML), and Artificial Neural Networks (ANN), as well as practical knowledge in building, training, validating, and testing SR and SER models with ML, and attempting to develop a desktop application using a new framework.

Building the SR and SER models was done mainly using the TensorFlow python library where a Convolutional Neural Network (CNN) model and a Long Short-Term Memory (LSTM) model were made respectively. Both of which showed promising results with adequate accuracies as well as other high-scoring performance metrics.

The application incorporating both SR and SER is intended to take audio sources from other apps and translates speech to text as well as portrays the emotion conveyed within the audio signal as a colour surrounding the text, however, many challenges were faced addressed. The application was built on the framework Electron as it is a cross-platform desktop application development framework made by GitHub.

Keywords*: speech recognition, speech emotion recognition, machine learning, artificial neural networks, convolutional neural networks, long short-term memory, artificial intelligence*

DECLARATION OF AUTHORSHIP

I, **Neuman Alkhalil**, hereby **declare** that the work entitled "Speech recognition and Speech Emotion Recognition" is my original work, and that I have not copied from any other work or from any other sources except where reference or acknowledgement is made in the text.

The expressed views are my own, and no one part of this work has been written for me by another person. I have cited all sources from which I have drawn information appropriately. This work has not been submitted toward any degree or award at any other university or place of learning.

_____

Neuman Alkhalil

Sarajevo, July 2022

First and foremost, I would like to thank my parents for their continuous support and for believing in me.

I would like to thank my sisters for making growing up that much more enjoyable.

I would like to thank the rest of my family for always being there for me, and my friends for sharing this adventure with me.

I would like to thank Oda and Milk for the weekly motivation.

Lastly, I would like to thank all my professors who helped, inspired, and guided me in my academic journey.

To the Milk to my Mocha.

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Aim and objectives of the thesis

In “Reading Emotions, Reading People”, Lange et al. [1] argue that the ability to recognize, interpret and react to the emotions of those around you holds the greatest value in terms of one’s own ability to circumnavigate the socio-cultural sphere of life. There exists a variety of methods in which people express their emotions, and one of the most notable is through speech, as indicated by the content of their speech or word selection, as well as their tone, pitch, and different aspect. Additionally, Lange et al. bring up the argument of social hardships that may befall those who lack the ability of emotion recognition.

According to Blanton [2], the impact of emotions upon the voice is recognized, felt, and understood by all, with even the most rudimentary of beings can detect the tones of love, fear, and rage, and animals share this understanding. He argues that the tonal language is the most ancient and ubiquitous of all our modes of communication [2]. According to Schuller [3], it would appear that the time has come for this aforementioned ubiquitous mode of communication to be understood and acquired by computing machinery.

Additionally, Schuller [3] argues that communication with computing machinery, an aspect of which is that of speech recognition (SR), has become a regular phenomenon in today time especially due to the popularity of voice assistants. However, it can be argued that without taking the roles and possible hidden sub context of emotions hidden in speech behind words that communication and understanding is greatly limited. Hence there exists recognition for the importance and need for a discipline of automatically recognizing human emotion and affective states from speech, usually referred to as Speech Emotion Recognition [SER] [4]. It is important to note that this patent dates to the 1970, with the first research paper on the topic

being of Daellert et al. in 1996 [5], today, 26 years later, it is still a field of computer science worth exploring.

With recognizing the crucial role speech recognition and speech emotion recognition play currently and the potential they show in the future, this BSc thesis is created with the aim of obtaining and expanding my knowledge relating to the field, as well as providing a practical learning opportunity for a field not covered by the traditional scope of undergraduate higher education and push the boundaries of applied knowledge and skill development which will primarily be obtained via the process of building and training a model, and an attempt at product creation.

1.2 Natural Language Processing (NLP)

Natural language is a reference to the method of communication used between humans, which is usually in the form of speech or text. [6] Natural language differs from constructed and formal languages, such as the ones used to program computers or within the study of logic.

The challenge of natural language is that human languages are highly ambiguous, ever changing, and constantly evolving. Although humans are great at expressing, perceiving, and interpreting language, we tend to be very poor at formally understanding and describing the rules that govern languages. [7]

Linguistics is the scientific study of language which overlooks grammar, semantics, and phonetics. Progressions in forming syntax and semantics in the field of linguistics have been made, however more focus on discrete mathematical formalisms and theory for natural language need to be prioritized for computers to process natural language. [6] This is where computational linguistics comes into play. Computational linguistics is defined as “the branch of linguistics in which the techniques of computer science are applied to the analysis and

synthesis of language and speech." [8] It creates a rule-based model of the human language which enables computational processing.

NLP is the branch of artificial intelligence concerned with enabling computers to understand text and spoken words in the same fashion as humans. Additionally, NLP includes the understanding of human "utterances", to the point of responding in a more useful manner in its least. This is done by combining the various models of computational linguistics, statistics, machine learning, and deep learning. [9]

A subsidiary to NLPs are n-gram character models. It's crucial to start with the fundamentals and recognize that a written or spoken word is made up of characters such as numerals/digits and punctuation. This leads to one of the most basic models of language: a probability distribution across character sequences. The probability of a sequence of N characters, c1 through cN, is given by P($C_{1:n}$) [10].

An n-gram is a sequence of written symbols with a length of N; hence, the models of probability distribution of n-letter sequences are termed n-gram models. This model is used to forecast forthcoming characters by the system in question, such as in autocorrect circumstances.

As previously stated, NLP is an area of artificial intelligence that enables machines to read, comprehend, and infer meaning from human communication [11]. Models are created with the goal to understand, break down, and separate important features from text and voice. NLP has also been widely utilized in a variety of applications, ranging from autocorrecting texts to plagiarism detectors.

Speech or text may be used as sources in NLP. Regardless of its source there exist six levels of analysis which can combine phonological analysis and tokenization: phonological, morphological, lexical, syntactical, semantic, discourse integration, and pragmatic analysis [10].

Phonological analysis is only utilized if the text in question origin is a speech, because it deals with the interpretation of speech sounds inside and across words in order to logically organize sound. Morphological analysis is concerned with deciphering different words based on their morphemes. The process of lexical analysis entails determining and analyzing the structure of words. Syntactic analysis is the technique of examining the words of a sentence to determine its grammatical structure. Semantic analysis focuses on the interactions between word-level meanings in a phrase to discover the various meanings. The features of the text as a whole that communicate meaning by forming links between component sentences are the focus of discourse integration. Finally, pragmatic analysis elaborates how additional meaning is read into texts without being encoded [12].

Tokenization, stemming, and lemmatization are the three most essential natural language processing activities to be aware of [13]. To begin, Tokenization is the process of breaking down complex data, such as paragraphs or sentences, into words and concepts known as Tokens with the purpose of speeding up the machine's learning process [14]. It's vital to note that these tokens can be bigrams, which are tokens made up of two consecutive words, trigrams, which are tokens made up of three consecutive words, and N-grams, which are tokens made up of "n" number of consecutive words. Second, stemming is the process of reducing tokenized words to their roots by removing suffixes and thereby removing redundancy [13]. Finally, because one of stemming's major drawbacks is the creation of intermediate representations of words, the process of lemmatization is critical because it considers morphological analysis of words in relation to vocabulary utilization (dictionary importance of words) and returns meaningful words in proper form [15].

1.3 What is Speech Recognition and Speech Emotion Recognition?

Speech recognition (also known as automatic speech recognition) utilizes NLP and machine learning (ML) to enable computers to recognize and translate spoken language into text [16]. It is known as the technique of turning human sound impulses into words or instructions for machines. It is a branch of pattern recognition and an important study field in speech signal processing. The purpose of speech recognition is to use phonetic and linguistic information to convert an input speech feature vector series into a sequence of words. A full speech recognition system contains a feature extraction method, acoustic model, language model, and search algorithm, according to the structure of the speech recognition system [17]. The speech recognition system is simply a pattern recognition system with several dimensions.

The majority of voice recognition algorithms are based solely on the sound of individual words, rather than their context. They make an attempt to identify words but are unable to comprehend speech [18]. When compared to human listeners, this puts them at a significant disadvantage.

Speech recognition has advanced significantly over the years, but some claim that its full potential has yet to be realized, since it may be used for voice authorization, typing, and remote health monitoring [19].

Speech recognition has advanced significantly over the years, but some claim that its full potential has yet to be realized, since it may be used for voice authorization, typing, and remote health monitoring [20]. Speech recognition is a crucial part of speech emotion recognition.

The challenge of recognizing the emotional qualities of speech, regardless of the semantic content, is known as Speech Emotion Recognition (SER). The SER system is defined as a set of approaches for processing and classifying speech inputs in order to identify emotions [21]. A system like this might be useful in a variety of situations.

It's worth noting that SER encompasses a wide range of disciplines, including computer science, artificial intelligence, machine learning, effective computing, computational linguistics, pattern recognition, and psychology. The simplest explanation for the problem of SER is the recognition of emotions from various audio samples discovered in enormous volumes in audio file databases [22].

Speech emotion recognition has a lot of promise, especially in terms of customer service and marketing, education especially for neurodivergent people and those with hearing impairments, and even in the healthcare system.

2. IMPORTANT DEVELOPMENTS

2.1 Speech Recognition

When it comes to speech recognition, an interdisciplinary subfield of computer science and computational linguistics, throughout history many new developments were made, and interdisciplinary field found new means of applicability and utilization. SR technologies have grown in popularity in recent years, and as a result, they are transforming the way people interact with computing systems. When it comes to history and the most important developments, Juang and Lawrence[23] traces its roots to the 1950s and the Three Bell Labs researchers which pawed the way for automated speech recognition as it is known today.

2.1.1 Audrey

In the 1950s, the general consensus amongst computer scientists was that speech signals needed to first be split into little phonetic units which could subsequently be grouped into words, as demonstrated in *Figure 1* where the audio sample with the word "test" is split into 4 phonetic units; "t", "eh", "s", and "t". This seemed promising at the time but did not produce good results. The first speech recognizer was called Audrey (Automatic Digit Recognition machine) which was developed by Bell Labs in 1952. There were limitations to Audrey as it could only recognize spoken digits from 0 to 9 from designated speakers, with an accuracy of more than 90% when uttered by its developer, 70-80% accuracy with designated speakers and significantly lower accuracy rates with other voices that were not used to train it [24]. The system also required many electronical equipment and specialized analog circuitry to recognize each digit.

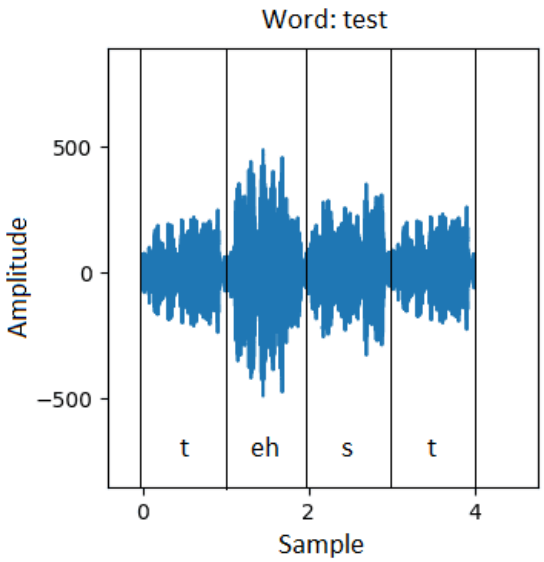


Figure 1 Audio sample split into phonetic units

2.1.2 Harpy

The next innovation in SR was a system called Harpy, developed by DARPA from Carnegie Mellon University in 1976, which consisted of 15,000 interconnected nodes, each of which represented all the possible utterances within the domain. It utilized the Beam Search algorithm, which explores a graph by expanding the most promising node in a limited set, examines near-miss alternatives, and does not backtrack, [25] in order to map the speech to the correct nodes, resulting in the corresponding text being produced. *Figure 2* shows the Beam Search algorithm used by Harpy, the top diagram shows the interconnected nodes that each represent a phoneme in the English language (perceptually distinct unit of sound), the diagram in the center represents the Beam Search algorithm traversing the different possible nodes within the tree, pruning the nodes with significantly lower heuristic values, and the diagram at the bottom is the audio sample used which has been pre-processed to prepare the data for recognition, segmented into acoustic units and analyzed to match the audio segments to one of the possible phonemes.

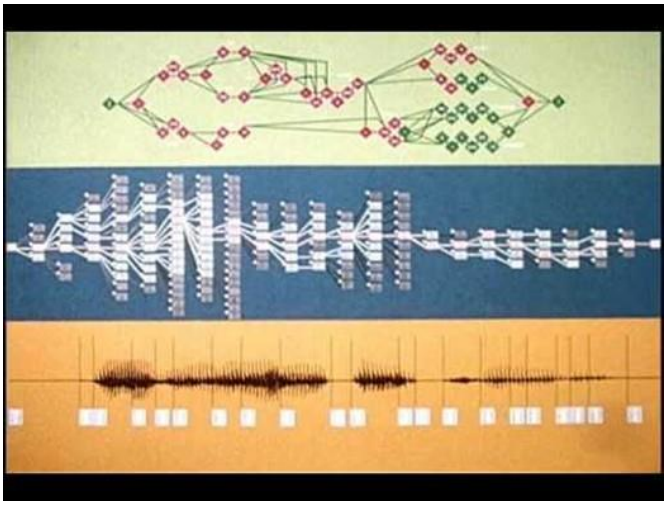


Figure 2 Harpy Beam Search Algorithm [26]

2.1.3 Hidden Markov Models (HMMs)

HMMs are a statistical model developed by Russian mathematician Andrey Andreyevich Markov in the early 1970s. [27] They were first used in speech recognition by the company IBM in the late 1980s. HMMs represented utterances as states and probabilistically predicted

the word based on the phonemes it was made up of. *Figure 3* [28] demonstrates a simple example of how HMMs are used in the modern day. The input audio signal is first segmented into a sequence of fixed size acoustic vectors (feature vectors) by a process called feature extraction. The decoder then takes the feature vectors and computes the most probable words that resulted in the vectors (Y). The decoder is comprised of Acoustic Models, Pronunciation Dictionary, and a Language Model [29]. Acoustic models refer to basic phones (phonemes) within the English language (approximately 40). Each word is decomposed into a sequence of phone models, and these models are concatenated to produce words defined in the pronunciation dictionary. The language model is responsible for computing the probability of a word given the words preceding it to distinguish different homophones and other possible words from one another, the model is usually in the form of an N-gram in which the probability of the word is conditioned specifically to the N-1 word. HMMs in speech recognition continued to produce the best results in the 1980s and 1990s as they were being improved upon and are even in use today.

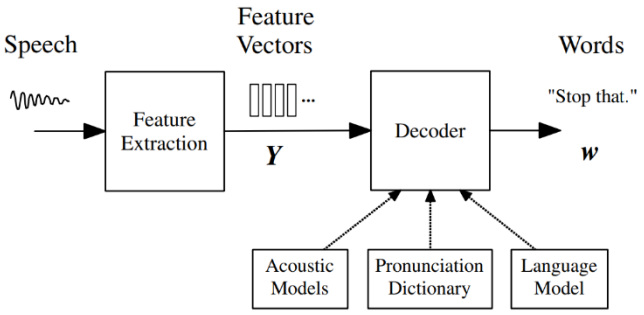


Figure 3 Speech recognition process using HMMs

2.1.4 Neural Networks

In the late 1980s, neural networks became a popular acoustic modeling technique in automatic speech recognition [30]. Since then, neural networks have been used in a wide range of speech recognition applications, including phoneme classification, multi-objective evolutionary algorithms for phoneme classification, isolated word recognition, audiovisual speech recognition, audiovisual speaker recognition, and speaker adaptation [31].

Neural networks are noted for making fewer explicit assumptions on feature statistical attributes than HMMs and have a number of characteristics that make them appealing speech recognition models. When used to estimate the probability of a speech feature segment, neural networks allow for natural and efficient discriminative training. Early neural networks, while their competence in identifying short-time units like individual phonemes and isolated words, were rarely successful in continuous recognition tasks due to their poor capacity to represent temporal connections [32].

The use of neural networks as a pre-processing, feature transformation, or dimensionality reduction step prior to HMM-based recognition was one way to overcome this restriction. However, in this field, LSTM and associated recurrent neural networks (RNNs) and Time Delay Neural Networks (TDNNs) have recently exhibited increased performance [33].

Geoffrey Hinton is noted as one of the pioneers for neural networking and its implementation within speech recognition [34]. Hinton's study looks at how neural networks may be used for machine learning, memory, vision, and symbol processing [35]. He was known for great passion and belief in neural networks, as well as restless efforts in. Geoffrey Hinton has been quoted saying: “I'll prove to you that it works, the key was to give it more data and computing power.” [n]. Due to deep learning, these neural networks have been utilized in technological advancements, especially those which prove services like Siri, Cortana and Google Voice [36].

2.2 Speech Emotion Recognition

Over the past two decades, speech emotion recognition (SER) has attracted a considerable amount of interest, especially from the machine learning community.

According to Schuller [37], the first significant advances toward what is now known as speech emotion recognition were made in accordance with the traditional approach of building an engine capable of detecting and recognizing emotion from speech.

It is important to highlight that reaching automated emotion recognition necessitates the use of an adequate emotion representation model. This creates two major issues: how to express emotion in general, and how to best quantify the time axis [Ibid]. The conventional technique employs the Big Seven model, beginning with adequately describing emotion to guarantee correct match with the psychological literature while selecting a representation that can be handled by a machine [38]. The Big Seven model employs discrete classes based on Ekman's "big seven" emotion categories, including anger, disgust, fear, happiness, surprise, and sorrow, which are frequently supplemented with a "neutral" rest-class [39], rather than a value "continuous" dimension approach. The following crucial issue faced in the field was that of acquisition of labeled data for training and testing that suits the according emotion representation model.

Before putting labeled data into a suitable machine-learning algorithm, one generally needs distinctive audio and textual features and properties [40]. The construction of ideal characteristics that best depict the emotional content and should be resistant against external sounds, changing languages, or even cultural influences is an ongoing active topic of study in the SER domain. The majority of the documented ones are quite low level, such as energy or spectrum information, which may be calculated reliably. However, there is a major emphasis on prosodic aspects in the synthesis of emotion, that is, characterizing the intonation, intensity, and rhythm of the speech with voice quality data [41].

A great number of recent studies have suggested a redirection of attention towards machine-learning-based SER systems. Harar et al. [42] suggested an approach based on convolutional, pooling, and fully connected layers in a Deep Neural Network (DNN) architecture. Their trained model obtained an overall test accuracy of 96.97 percent on full file categorization into three groups of furious, neutral, or sad using raw audio data from the German Corpus (Berlin Database of Emotional Speech). The Interactive Emotional Dyadic Motion Capture

(IEMOCAP) database from USC is another important database that has been utilized by a number of studies to test various SER approaches [43,44,45]. Fayek, Lech and Cavedon [46] had the highest effective models, with accuracies of 64.78 percent and 64.16 percent, respectively, for their Convolutional Neural Network (CNN)architecture and Bi-directional Long Short-Term Memory-Extreme Learning Machine (LSTM-ELM) model. Trigeorgis et al. [47] used a similar strategy, combining CNNs with LSTM networks, and found comparable prediction accuracies when applied to the REmote COLaborative and Affective (RECOLA) dataset (68.6 percent for arousal, 26.1 percent valence) [48]. It should be noted that the audio files from these datasets were recorded by actors and hence are not totally representative of genuine speech. That is not to denigrate the accomplishments of prior publications; further study, particularly on organic speech, is required.

A number of interesting possibilities have lately piqued the community's curiosity. The most noteworthy example is holistic speaker modeling [49]. Consideration of different states and features that momentarily effect voice production is an essential part of enhanced robustness. In other words, one is not just emotional, but also maybe exhausted, suffering from a cold, inebriated by alcohol, or speaking differently due to being in a certain mood [50].

Similarly, current emotion recognition engines should include a speaker's overall state and features in addition to the emotion of interest in order to optimally detect it independent of such co-influencing elements. Once an initial engine has been trained, semi-supervised learning algorithms may be successful in exploiting further unlabeled data [51]. The goal is for the computer to identify previously viewed data on its own, ideally only if a significant confidence level is surpassed.

Speech emotion recognition has dramatically developed as a field, especially considering the amount of works, research and projects in its relation and the possibilities it provides from

those of optimizing understanding for those with neurodevelopmental disorders, to optimizing customer service via online platforms, or even when it comes to security systems [52]. Speech emotion recognition brings the promise of eliminating miscommunication due to the hidden role emotions play being finally uncovered.

3. EXISTING SOLUTIONS

While there were great advancements, innovations, and explorations throughout history of speech recognition and speech emotion recognition in itself, today its utilization is a pressing topic of academic and business discussions, as both aspects have made their way into the lives of consumers with speech recognition now playing an important part in everyday urban living.

3.1 Existing Speech Recognition Products

Speech recognition has dramatically increased in popularity and utilization with each new advancement. SR is currently a widespread type of input that is used to replace techniques like as typing and clicking [53]. Today, SR technology can be found in a variety of industries, including the automobile industry, where it allows drivers to manage some components of the vehicle via voice commands without having to leave the steering wheel. The most widespread application of SR is probably in current cell phones.

There exists a plethora of consumer products which utilize speech recognition, out of which the most notable being so called voice assistants such as Siri, Alexa, Bixby and Cortana. With voice recognition referring to the ability of a machine to receive, interpret, and carry out speech commands, it gained prominence with the rise of Artificial Intelligence and voice assistants.

According to a report by the US Statista Research Department in 2022, Apple's Siri remains the most popular voice assistant with its participants [54]. Siri is the virtual assistant that is part of Apple Inc. operating systems. It fulfils the functions of answering questions, making

suggestions, and performs activities by delegating requests to a collection of Internet services using voice inquiries, gesture-based control, focus-tracking, and a natural-language user interface [55]. It adjusts to users' specific language usages, queries, and preferences over time, giving personalized results. On the other hand, Bixby is a virtual assistant created by Samsung Electronics that marks a significant redesign of S Voice, Samsung's voice assistant software debuted in 2012 which has the same functionality as Siri.

However, voice assistants are not only limited to ones mobile phones. Similarly, Amazon Alexa, or simply Alexa, is a virtual assistant system that is primarily based on Ivona, a Polish voice synthesizer purchased by Amazon in 2013. It has capabilities of speech interactions, playing music, creating to-do lists, setting alarms, streaming podcasts, playing audiobooks, and providing weather, traffic, sports, and other real-time information such as news [56]. Alexa can also act as a home automation system, controlling a variety of smart gadgets.

Lastly, Cortana is a Microsoft virtual assistant that leverages the Bing search engine to conduct things such as creating reminders and answering inquiries for the user [57].

3.2 Existing Speech Emotion Recognition Products

While voice recognition has seen major utilization, especially in terms of a function as voice assistants, when it comes to speech emotion recognition there are less utilized and known products which its implementation. Most notably it is rather yet a field of exploration for those with interests but is of a expanding and innovative character.

There exist a few softwares with the goal of emotional recognition from vocal sources. The most famous one would be EmoVoice, a framework for creating emotional speech corpuses and classifiers, as well as for offline and real-time online speech emotion recognition.

Additionally, there exists Vokaturi, a software that is said to be capable of "understanding the emotion in a speaker's voice in the same manner as a person can" [58]. Developers may incorporate Vokaturi into their projects using the Open Vokaturi SDK. The Vokaturi software will compute percentage likelihoods for five emotional states given a database of voice recordings: neutrality, happy, sorrow, anger, and fear.

And lastly, there is Good Vibrations. The Good Vibrations API detects mood by listening to recorded voice [59]. The API and SDK leverage universal biological cues to do real-time emotion analysis to detect stress, pleasure, or disorder.

4. PRACTICAL BACKGROUND

In addition to newly enquired theoretical knowledge, the aim of this BSc thesis additionally states the equipping of a practical knowledge related to speech recognition and speech emotion recognition, or more notably its functionality and implementation. In order to obtain both theoretical and practical knowledge on the subject it was required to branch out and learn to use a variety of programs, frameworks, models and such in order to be able to comprehend and execute SR and SER model building, training, and implementation.

An important aspect of the process of SR and SER training and implementation was becoming familiar with the software libraries TenorFlow and Keras which was previously foreign.

Machine learning is a branch of computational algorithms which encompasses a broad range of algorithms and modeling tools [60] which are designed with the goal of emulating human intelligence by the process of learning from the surrounding environment [61]. As such, ML is primarily used for the processing of a vast array of data. According to Jordan and Mitchell [62], it is one of the fastest-growing technological fields today, straddling the lines between computer science and statistics, as well as artificial intelligence and data science. The

development of novel learning algorithms and theory, as well as the continual boom in the availability of online data and low-cost processing, have fueled recent advances in machine learning. John Terra conducted a cross comparison of different deep learning frameworks in 2022 [63] and concluded that TensorFlow was the most optimal choice, especially for those with a lack of a background or experience in the use and utilization of deep learning frameworks. Abadi et al. [64] defines TensorFlow as a machine learning and artificial intelligence software library that is free and open-source, and as such may be used for a variety of applications, but with a focus on deep neural network training and inference.

TensorFlow is a symbolic math library for neural networks that is ideally suited for dataflow programming in a variety of applications. It allows you to construct and train models at many abstraction levels [65]. The deep learning framework was developed and released by Google in 2015 aiding to its widespread availability and evident compatibility. Terra [66] argued that it is notable for its documentation and training assistance, as well as its scalable production and deployment options, many abstraction levels, and support for several platforms, such as Android. Pang, Nijkamp and Wu [67] argue that TensorFlow greatly eases and accelerates the research and application of neural network models, which include multilayer perceptron, the convolutional neural network, and stochastic gradient descent. It is important to note that it can be utilized in a wide variety of programming languages, most notably Python, as well as Javascript, C++, and Java. For the purpose of this work, it shall be used with Python as the general-purpose, high-level, interpreted programming language with the design philosophy which prioritizes code readability with extensive indentation [68].

With all this in mind, TensorFlow was be utilized in this work, but not as a part of the application, as it provides the most optimal choice of a deep learning framework one can and should gain knowledge of and use, in order to train the model in speech recognition and speech

emotion recognition, however to make an effective model large amounts of data need to be fed to it in order to train it.

In order to perform the practical aspect of the thesis, knowledge both theoretical and practical was gained of a CNN. A convolutional neural network (CNN) is a type of artificial neural network (ANN) used in deep learning which is most typically used to analyze visual data but may also be used to analyze speech [69], hence the utilization of a convolutional neural network model.

In order for goal execution a fast Fourier transform (FFT) was utilized. FFT is a method that computes a sequence's discrete Fourier transform (DFT) or its inverse (IDFT) [70]. It is the standard approach for analyzing the frequency spectrum of a signal in voice recognition. Due to the enormous number of samples per window, speech recognition demands a lot of processing where FFT consists of complex field computing.

5. SPEECH RECOGNITION

5.1 Dataset

To train the model for speech recognition, an open-source dataset created by Tensorflow and AIY which consisted of 65,000 one-second long utterances of 30 short command words that was said by thousands of different people, with varying levels of clarity, noise, and pitch which was collected through a website where members of the public could contribute their voices as data for the dataset. This method of collection allows large volumes of data to be collected much more easily. Because of this, the dataset was the ideal choice for training the model as it provided a large amount of varying data, perfect for the Machine Learning process.

5.2 Preparing the data

To start with, the dataset must be prepared in order to be used. As aforementioned in the section on HMMs, the speech data goes through a process called feature extraction in which the files will be segmented, and the different phones are split up. Once this is done the features will be stored in a JSON file.

The first step to feature extraction would be defining a data dictionary. This dictionary contains 4 parameters which are 'mappings' (the 30 command words in the dataset), 'labels' (values representing each of the audio files in the dataset), 'MFCCs' (a method for feature extraction, this is where the input data for training the model will be stored), and 'files' (the filepath for each file).

```
# data dictionary
data = {
    "mappings": [],
    "labels": [],
    "MFCCs": [],
    "files": [],
}
```

To populate the dictionary with all 65,000 files would not be practical to do manually and as such will be done programmatically. In order to do so, the program loops through all the sub-directories of the dataset and every directory path (directory paths in this instance happen to be the 30 words in the dataset) is appended to the dictionary under 'mappings'.

```
# loop through all the sub-directories
for i, (dirpath, dirnames, filenames) in enumerate(os.walk(dataset_path)):
    if dirpath is not dataset_path:
        category = dirpath.split("/")[-1]
        data["mappings"].append(category)
        print(f"Processing {category}")
```

The next step is extracting the features, however before this all the files under 1 second in length must be excluded as a CNN is used which should ideally take in data which have the same shape. This can be done with two lines of code:

```
if len(signal) >= SAMPLE_RATE:
    signal = signal[:SAMPLE_RATE]
```

The conditional statement only allows audio signals with a predefined sample rate of 22050 (22 kHz), or 1 second, to be considered, and the line of code following that takes a 1 second 'snip' of the file in the case that it is over 1 second. Once all audio signals are 1 second in length, the MFCCs can be extracted using the python audio processing library Librosa. The function 'librosa.feature.mfcc()' takes as input 4 parameters which are the: audio signal, number of coefficients to extract, length of each segment/coefficient, and size of the window for the FFT.

```
MFCCs = librosa.feature.mfcc(signal, n_mfcc=n_mfcc, hop_length=hop_length, n_fft=n_fft)
data["labels"].append(i-1)
data["MFCCs"].append(MFCCs.T.tolist())
data["files"].append(file_path)
```

After doing so we append the current audio signal's label, MFCCs and filepath to the dictionary.

After traversing every file within the dataset and storing it into the dictionary, the contents of the dictionary are dumped into a json file and ready to be used for training the model.

5.3 Building and Training the Model

Building and training the model can be summarized in 4 steps:

1. Loading data for training, validation, and testing
2. Building the CNN model
3. Training the model
4. Saving the model

To load the data for training, validation and testing, the python library Numpy to turn the data in the dictionary stored in the json file into an array of values:

```
#extract inputs and targets
X = np.array(data["MFCCs"])
Y = np.array(data["labels"])

return X, Y
```

The two arrays are then split further into training, testing and validation sets using the Sklearn python library, where 10% is used for testing, 10% is used for validation, and the rest is used for training:

```
#create train/validation/test splits
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=test_size)
X_train, X_validation, Y_train, Y_validation = train_test_split(X_train, Y_train, test_size=test_validation)
```

Building the CNN model will require the help of the Keras API within the Tensorflow library. Keras is a high-level library that provides a Python interface for artificial neural networks.

The first step to building the model is building the network. The sequential model was used from the Keras library as it groups the linear stack of convolutional layers that would be made to create a sequential model:

```
#build network
model = keras.Sequential()
```

Next is creating the convolutional layers, of which there will be 3 for this project. The first layer will have 64 filters, each of which are of a size of 3x3. The layer implements the activation function ReLU (rectified linear unit), it takes as input the shape of the audio signals used for testing the model as this should be defined in the first layer, and lastly it also uses the L2 regularizer (Ridge Regression) which is used to reduce the chance of the model overfitting.

Once the convolutional layer is formed, batch normalization is done to make the neural networks faster and more stable through normalization of the layers' inputs.

The 3x3 filters are then max pooled, this is an operation, as demonstrated in figure 4, [71] in which the maximum element from each region of the feature

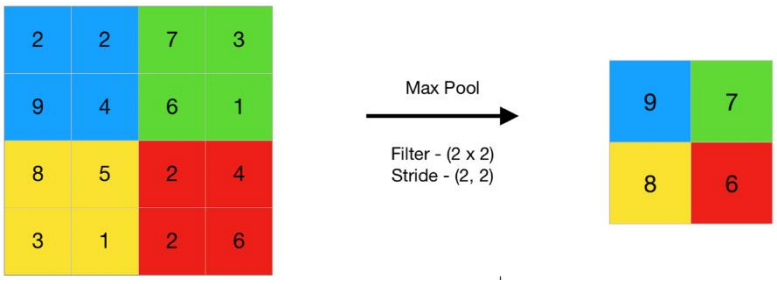


Figure 4 Max Pooling example

map covered by the filter is selected, resulting in a feature map containing the most prominent features of the previous feature map.

The next 2 layers will essentially be the same as the first layer, the only differences being that the input shape does not have to be defined, the 2 layers both have 32 filters as opposed to 64, and the third layer's filters are 2x2 as opposed to 3x3. These differences are evident in the example of code:

```
#conv layer 1
model.add(keras.layers.Conv2D(64, (3, 3), activation="relu", input_shape=input_shape,
                              kernel_regularizer=keras.regularizers.l2(0.001)))
model.add(keras.layers.BatchNormalization())
model.add(keras.layers.MaxPool2D((3, 3), strides=(2, 2), padding="same"))

#conv layer 2
model.add(keras.layers.Conv2D(32, (3, 3), activation="relu",
                              kernel_regularizer=keras.regularizers.l2(0.001)))
model.add(keras.layers.BatchNormalization())
model.add(keras.layers.MaxPool2D((3, 3), strides=(2, 2), padding="same"))

#conv layer 3
model.add(keras.layers.Conv2D(32, (2, 2), activation="relu",
                              kernel_regularizer=keras.regularizers.l2(0.001)))
model.add(keras.layers.BatchNormalization())
model.add(keras.layers.MaxPool2D((2, 2), strides=(2, 2), padding="same"))
```

Once the layers are formed, the multidimensional output feed should be flattened into a linear dense layer (aka fully-connected layer):

```
#flatten the output feed it into a dense layer
model.add(keras.layers.Flatten())
model.add(keras.layers.Dense(64, activation="relu"))
model.add(keras.layers.Dropout(0.3))
```

The last step to building the model is the Softmax classifier. In Layman's terms, the Softmax classifier 'squashes' the raw scores into normalized positive values that sum to one, this allows cross-entropy loss (minimizing loss) to be applied:

```
#softmax classifier
model.add(keras.layers.Dense(NUM_KEYWORDS, activation="softmax"))
```

All that's left is to compile the model which is done using the Adam optimizer with a learning rate of 1e-4. Figure 5 [72] is a diagram that visually represents a similar CNN and how all the components are connected.

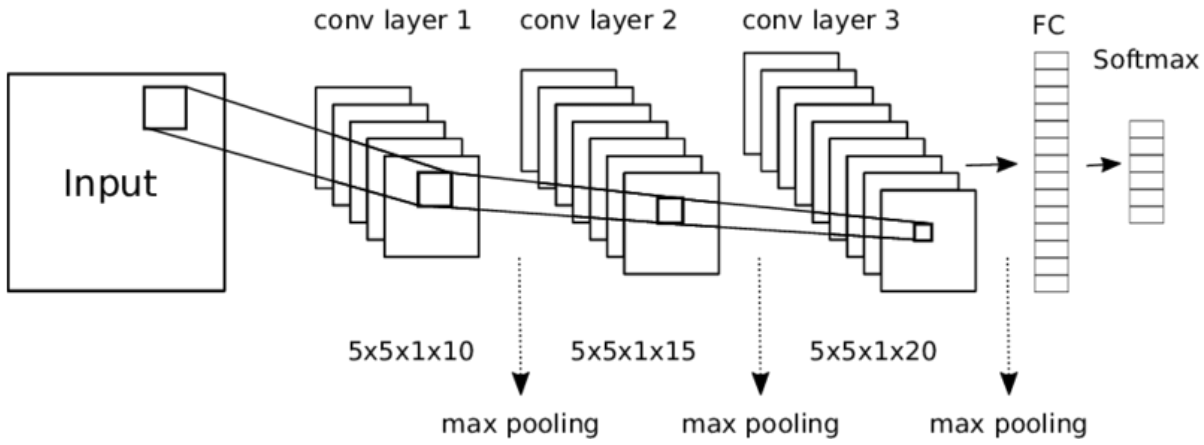


Figure 5 Convolutional Neural Network example

Training this model requires just one function from Keras called 'fit()' which takes the training data that was loaded earlier, the number of epochs which was set to 40, the batch size which

```
#train the model
model.fit(X_train, Y_train, epochs=EPOCHS, batch_size=BATCH_SIZE, validation_data=(X_validation, Y_validation))
```

was set to 32, and the validation data which included the two validation data sets loaded at the start:

5.4 Testing and Evaluating the Model

After the first epoch, the validation accuracy was 0.3769 (37.7% 3.s.f.). This was quickly improved as on the 7th epoch the validation accuracy rose to 0.8037 (80.4%) and gradually increased from there.

```
Epoch 1/40
1475/1475 [==============================] - 27s 18ms/step - loss: 3.1287 - accuracy: 0.1582 - val_loss: 2.3753 - val_accuracy: 0.3769
Epoch 2/40
1475/1475 [==============================] - 27s 19ms/step - loss: 2.2533 - accuracy: 0.3602 - val_loss: 1.6802 - val_accuracy: 0.5615
Epoch 3/40
1475/1475 [==============================] - 27s 18ms/step - loss: 1.7523 - accuracy: 0.4967 - val_loss: 1.2855 - val_accuracy: 0.6636
Epoch 4/40
1475/1475 [==============================] - 26s 18ms/step - loss: 1.4454 - accuracy: 0.5857 - val_loss: 1.0674 - val_accuracy: 0.7185
Epoch 5/40
1475/1475 [==============================] - 27s 18ms/step - loss: 1.2393 - accuracy: 0.6470 - val_loss: 0.9200 - val_accuracy: 0.7536
Epoch 6/40
1475/1475 [==============================] - 26s 18ms/step - loss: 1.0889 - accuracy: 0.6950 - val_loss: 0.7921 - val_accuracy: 0.7950
Epoch 7/40
1475/1475 [==============================] - 26s 18ms/step - loss: 0.9884 - accuracy: 0.7230 - val_loss: 0.7367 - val_accuracy: 0.8037
Epoch 8/40
1475/1475 [==============================] - 26s 18ms/step - loss: 0.9028 - accuracy: 0.7519 - val_loss: 0.6552 - val_accuracy: 0.8352
Epoch 9/40
1475/1475 [==============================] - 26s 18ms/step - loss: 0.8367 - accuracy: 0.7705 - val_loss: 0.6050 - val_accuracy: 0.8453
Epoch 10/40
1475/1475 [==============================] - 26s 17ms/step - loss: 0.7877 - accuracy: 0.7855 - val_loss: 0.5805 - val_accuracy: 0.8510
```

As shown in the last 10 epochs, the validation accuracy stabilized at around 0.91 (91%) with an overall test accuracy of 0.9071 (90.7%) and a test loss of 0.3882 (38.8%) which is ultimately a good overall score for the model.

```
Epoch 31/40
1475/1475 [==============================] - 26s 18ms/step - loss: 0.4393 - accuracy: 0.8861 - val_loss: 0.3725 - val_accuracy: 0.9046
Epoch 32/40
1475/1475 [==============================] - 26s 18ms/step - loss: 0.4349 - accuracy: 0.8864 - val_loss: 0.3829 - val_accuracy: 0.9022
Epoch 33/40
1475/1475 [==============================] - 26s 18ms/step - loss: 0.4262 - accuracy: 0.8902 - val_loss: 0.3727 - val_accuracy: 0.9102
Epoch 34/40
1475/1475 [==============================] - 26s 18ms/step - loss: 0.4192 - accuracy: 0.8921 - val_loss: 0.3697 - val_accuracy: 0.9083
Epoch 35/40
1475/1475 [==============================] - 26s 18ms/step - loss: 0.4133 - accuracy: 0.8915 - val_loss: 0.3607 - val_accuracy: 0.9088
Epoch 36/40
1475/1475 [==============================] - 26s 18ms/step - loss: 0.4046 - accuracy: 0.8965 - val_loss: 0.3679 - val_accuracy: 0.9079
Epoch 37/40
1475/1475 [==============================] - 26s 18ms/step - loss: 0.4058 - accuracy: 0.8955 - val_loss: 0.3669 - val_accuracy: 0.9100
Epoch 38/40
1475/1475 [==============================] - 26s 18ms/step - loss: 0.3993 - accuracy: 0.8978 - val_loss: 0.3611 - val_accuracy: 0.9083
Epoch 39/40
1475/1475 [==============================] - 26s 18ms/step - loss: 0.3917 - accuracy: 0.8997 - val_loss: 0.3707 - val_accuracy: 0.9073
Epoch 40/40
1475/1475 [==============================] - 26s 18ms/step - loss: 0.3871 - accuracy: 0.8999 - val_loss: 0.3532 - val_accuracy: 0.9121
183/183 [==============================] - 1s 6ms/step - loss: 0.3882 - accuracy: 0.9071
Test error: 0.38820430636405945, test accuracy: 0.9071404337882996

Process finished with exit code 0
```

Due to the large number of classes (30) present in this multiclassification problem, the loss function used was sparse categorical cross-entropy, and producing a confusion matrix as well as calculating other metrics such as precision, recall and F1 were not very practical nor possible, especially as the model does not easily mistake one word for another and so the statistics would not show many different results.

To put this model into use, a word prediction service was created in order to test the model using self-recorded audio clips where two of the keywords from the dataset were uttered.

Due to the nature of the CNN model, the self-recorded audio clips had to be prepared in the same way as the dataset was to have the same shape as the data used to train the model.

```
if __name__ == "__main__":
    word_prediction_service = Word_Prediction_Service()

    word_prediction1 = word_prediction_service.predict("test_files\\happy.wav")
    word_prediction2 = word_prediction_service.predict("test_files\\marvin.wav")

    print(f"Predicted words: {word_prediction1}, {word_prediction2}")
```

The model proved to work as it correctly predicted both the words which were ‘happy’ and ‘marvin’:

```
1/1 [==============================] - 0s 15ms/step
Predicted words: happy, marvin

Process finished with exit code 0
```

5.5 Challenges

There were not many complications with building this model besides in preparing the speech data. Compatibility issues with the data not having the same shape was met with a work around of enforcing a 1

```
Total params: 35,998
Trainable params: 35,742
Non-trainable params: 256
```

second audio file for all files. This condition caused almost 30,000 of the 65,000 parameters to be neglected due to them being under 1 second in length as shown:

The main challenge was building the CNN model as it required more knowledge of the Tensorflow framework as well an understanding of neural networks, machine learning, and different algorithms.

6. SPEECH EMOTION RECOGNITION

6.1 Dataset

In order to train the speech emotion recognition model the Toronto Emotional Speech Set (TESS) database containing 2,800 audio files with 2 different actors was used. In accordance with the work of Masuyama [73] the dataset covered the seven fundamental human emotions of anger, disgust, fear, happiness, pleasant surprise, sadness and a neutral state with 400 audio files present for each amotion within the dataset. It is important to note that each audio file was split into multiple segments and that each segment was counted as a parameter within the training process equaling to the total number of parameters being 305,799.

6.2 Preparing the data

As in the model for speech recognition, the data must first be prepared before it is fed to the model.

All the files within the dataset directory are first traversed in order to store their filepaths and labels (the emotion) in 2 arrays. The values within the arrays are then stored within a DataFrame, made using the python library Pandas. A DataFrame is a two-dimensional array with corresponding labels which make it ideal for certain data science and machine learning scenarios, as in this one. An example of the use of a DataFrame can be seen on Figure 6 using the two libraries Seaborn and Matplotlib to plot a count of the number of files with each label:

Figure 6 Count of all the iterations of emotions in the dataset

```
sns.countplot(dataframe['label'])
plt.show()
```

6.3 Analyzing the Different Emotions

To better analyze the difference between the different emotions, spectograms are usually used in order to visually represent the varying emotions. A spectogram can be made for each emotion using the Librosa and Matlibplot libraries as shown in 4 prevalent emotions:

```
def spectogram(data, sr, emotion):
    x = librosa.stft(data)
    xdb = librosa.amplitude_to_db(abs(x))
    plt.figure(figsize=(11,4))
    plt.title(emotion, size=20)
    librosa.display.specshow(xdb, sr=sr, x_axis='time', y_axis='hz')
    plt.colorbar()

def represent_emotion(emotion):
    path = np.array(dataframe['file_path'][dataframe['label'] == emotion])[0]
    data, sampling_rate = librosa.load(path)
    waveplot(data, sampling_rate, emotion)
    spectogram(data, sampling_rate, emotion)
    Audio(path)

represent_emotion('angry')
represent_emotion('sad')
represent_emotion('fear')
represent_emotion('happy')
```

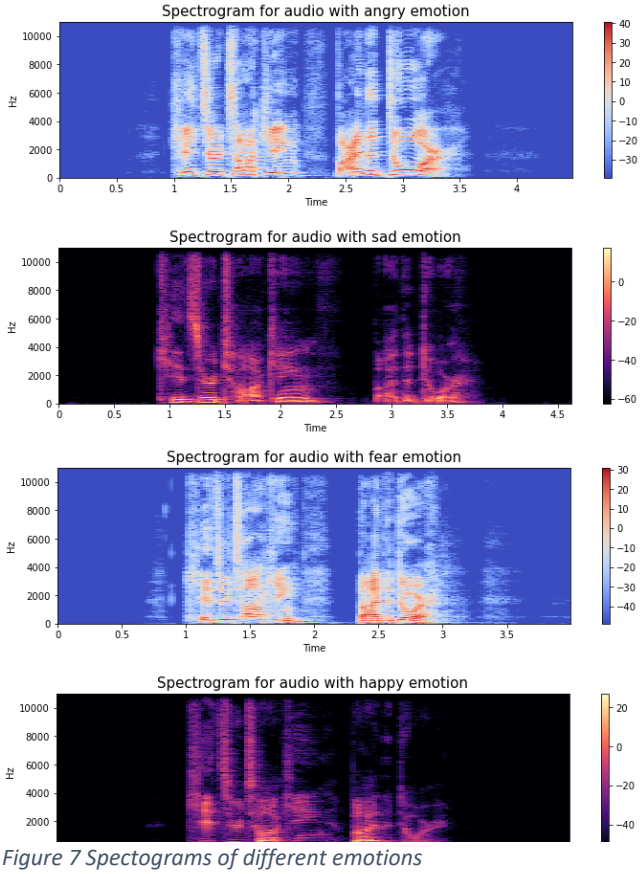


Figure 7 Spectograms of different emotions

6.4 Building and Training the Model

As in the previous model, to prepare the data, the features (MFCCs) must be extracted. This is done via Librosa's 'mfcc()' function:

```
def extract_mfcc(filename):
    Y, sr = librosa.load(filename, duration=3, offset=0.5)
    mfcc = np.mean(librosa.feature.mfcc(y=Y, sr=sr, n_mfcc=40).T, axis=0)
    return mfcc
```

The model uses the sequential model as the previous CNN model for speech recognition did, however this one utilizes LSTM's rather than the CNN of the previous model. LSTM's are not limited to process one data point as they have the ability to also process entire sequences of data.

```
model = Sequential([
    LSTM(256, return_sequences=False, input_shape=(40,1)),
    Dropout(0.2),
    Dense(128, activation='relu'),
    Dropout(0.2),
    Dense(64, activation='relu'),
    Dropout(0.2),
    Dense(7, activation='softmax')
])
```

The model went through a similar flattening process in order to make the output of the LSTM layer into a linear Dense layer using the ReLU activation function.

The model was then compiled using the optimizer Adam and trained with 10% of the dataset being used for validation, 10% being used for testing and the rest being used for training. The training of the model went on for 100 epochs in order to get the most consistent and accurate model possible.

During the training of the model, multiple training sessions were conducted in hopes of increasing accuracy and reliability. During the 20th training of the model the highest value of testing and validation accuracy out of all the models was found.

6.5 Testing and Evaluating the Model

In order to understand whether the models are performing effectively and providing valid and reliable data there are a number of performance metrics which can be utilized. It is important to note that the models del with classification models, meaning that we are measuring the successful prediction of the target class of the data sample provided, aka whether an instance belongs to one class or another [74]. It is important to note that the complexity of the classification system depends on the number of classes utilized, as binary systems are considered the easiest and multiclass ones bring with them their own challenges [75].

When training and creating models, especially in machine learning, it is important to have a evaluation of its reliability as we are dealing with real-world problems that can be utilized to help numerous groups and individuals. In order to have an estimate on the reliability of the predictions we may utilize accuracy, precision, recall and F1-score [76].

6.5.1 Confusion Matrix

According to Bhandari [77] a confusion matrix can be understood as an NxN model which is utilized in order to evaluate the performance of a classification model in which N represents the number of target classes. They are used in machine learning as they provide a holistic view of the accurate performance of the classification models and its errors by comparing the predicted value gained via the machine learning model and the actual target value.

In a binary classification problem it would contain a 2x2 matrix witch 4 possible values which would include: true positive (TP), true negative (TN), false positive (FP – which is a type 1 error), and false negative (FN – which is a type 2 error).

The values bring with them their own understanding and indicators of meaning which are the following presented in *Table 1*.

True Positive (TP)	The predicted value matches the actual value The actual value was positive, and the model predicted a positive value
True Negative (TN)	The predicted value matches the actual value The actual value was negative, and the model predicted a negative value
False Positive (FP)	The predicted value was falsely predicted The actual value was negative, but the model predicted a positive value Also known as the Type 1 error
False Negative (FN)	The predicted value was falsely predicted The actual value was positive, but the model predicted a negative value Also known as the Type 2 error

Table 1: Table representation of the values of a Confusion Matrix

A confusion matrix for a multi-class classification model is similarly implemented, and clearly shows more data.

The multi-class confusion matrix that was utilized in the evaluation of the model can be seen in Figure 8, via which it is possible to not only view the degrees of reliability in the machine learning models predictions but even a greater insight into the phenomena of speech-emotion recognition.

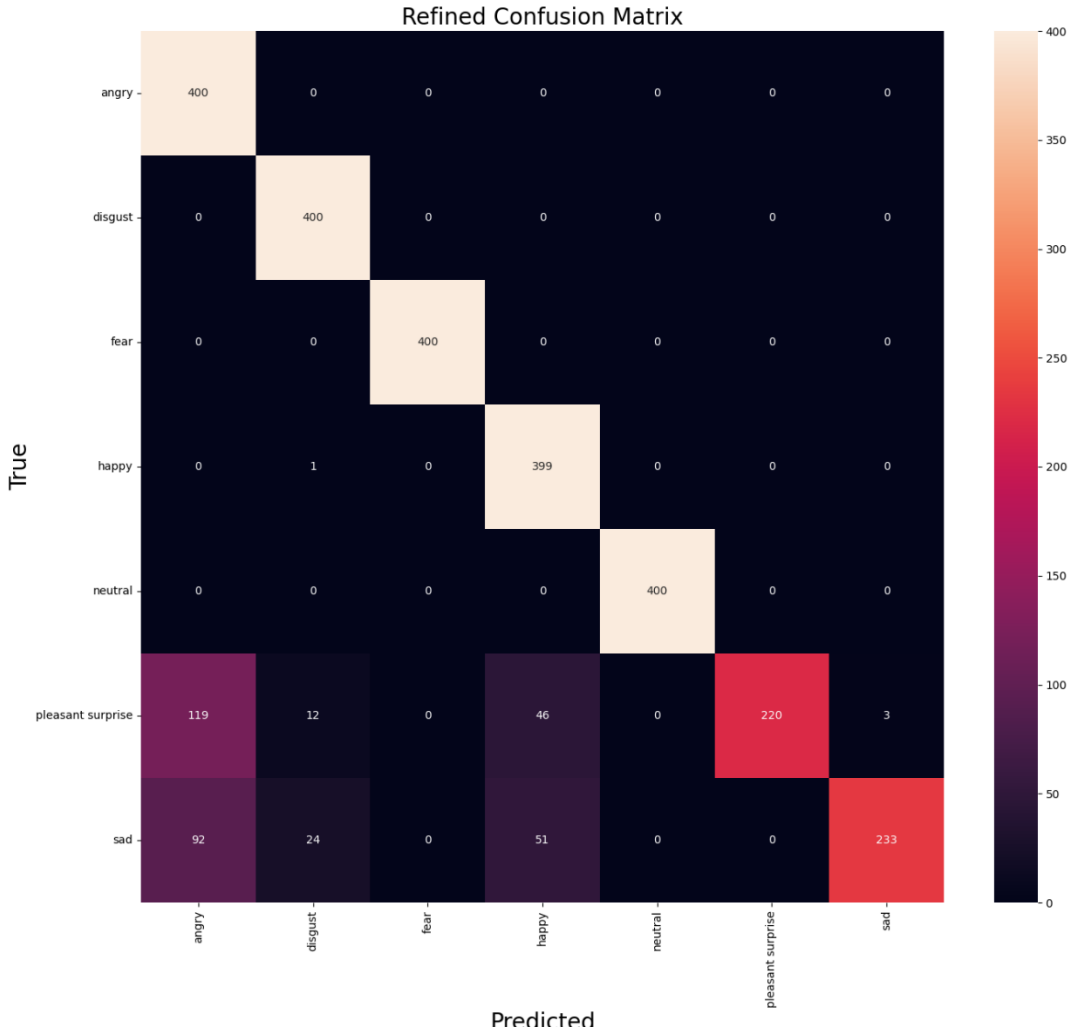


Figure 8: Multi-class confusion matrix for the speech-emotion recognition model

It is possible to conclude from Figure 8, that there is high reliability of the machine learning models predictions of the emotions for five out of the 7 provided emotions of anger, disgust, fear, happiness, and neutral respectively. However, it is important to note that while still presenting a lower degree of reliability, that the phenomena of emotion confusion is especially prevalent for the values and emotions of pleasant surprise and sadness.

Understanding the phenomena of emotion confusion would not be possible without referring to the field of study of psychology. According to Pettinelli [78] in his book “The psychology of emotions, feelings and thoughts”, sadness and anger are not mutually exclusive emotions but those generally felt in unison due to their interlocking value derived from their schema

recall in their working and long term memory. Additionally, Pettinelli argues that the exhibition of emotions such as sadness has been shown to be linked to feelings of injustice and hence concur along with feelings of frustration which can be banalized to feelings of anger. It is important to note that Koolagudi and Rao [79] state this phenomena in arguing how machine learning and speech-emotion recognition is not yet possible to separately consider and identify the emotion of "frustration" and as such rather relates such indicators to that of the emotion of anger for which mode data is provided.

Additionally, research by Jin, Chen and Wu [80] provides an explanation for the confusion of the emotions of positive surprise and anger, by stating that when taking into account phonetic indicators of recorded and provided audio datasets that it is possible to find commonalities due to both experiencing high pitch or intensity. Roach [81] explains that when it comes to phonetic descriptions of emotions that intensity of expression is additionally often related to the emotional expression of disgust which can explain its confusion with the feeling of surprise whether positive or negative. And lastly they add that feelings of happiness and positive surprise are often confused for one another due to the indicators of pitch and presented undertones of excitement [82]

All this would confirm the reliability of the models used as exhibited in the confusion matrix while providing us with a greater understanding not only for the confusion in terms of emotion recognition but the phenomena of their own categorization criteria and its shared presence. As such to properly addressed this issue it would be needed to develop additional criteria for emotion classification for which the contribution of linguists, phoneticians and psychologists are required. However, it is possible to argue that these emotions may be confused even in real life and by humans due to their overlapping nature and the notion that no speech is truly always expression only one emotion. Additionally, it is important to note that the emotional classification criteria is only specific to English language and the same criteria and as such

results would not be able to be obtained for languages such as German, Spanish, Japanese and such which are marked with clear phonetical differences in speech patters and emotion expression according to linguists [83] and anthropologists [84].

6.5.2 Measuring Accuracy

When it comes to the performance metrics for evaluating the reliability of machine learning models the most commonly utilized one is accuracy.

The ratio of true positives and true negatives to all positive and negative observations is referred to as the model accuracy, which is a performance statistic for machine learning classification models [85]. In other words, accuracy indicates the proportion of times our machine learning model will predict a result accurately out of all the predictions it has made. It mathematically denotes the proportion of the total of true positive and true negative forecasts. Mathematically represented as:

$$Accuracy\ score = \frac{TP + TN}{TP + FP + TN + FN}$$

Additionally, it is important to note that for cases of multi-class classification machine learning models that there exist two types of accuracy: testing and validation accuracy. The results of the evaluation and measurement of accuracy can be found below.

As visible in Figure 9 and the console log below, the initial value of testing marks its lowest at 0.1464 (14.6%) but the graph's testing and validation accuracy drastically improves with each new epoch, aka a complete pass of the training dataset through the algorithm. The highest recorded validation accuracy stands at 0.9500 (95.0%), followed by 0.9286 (92.9%). While the validation accuracy took longer to reach stabilization compared to other training attempts, it stabilized around 0.82 (82%) past the 80th epoch. It is important to note that such a high degree of validation accuracy is rarely documented and reached, but one must remember that the dataset utilized only contained 2800 audio files and that the results may vary if a larger dataset was implemented within the model training.

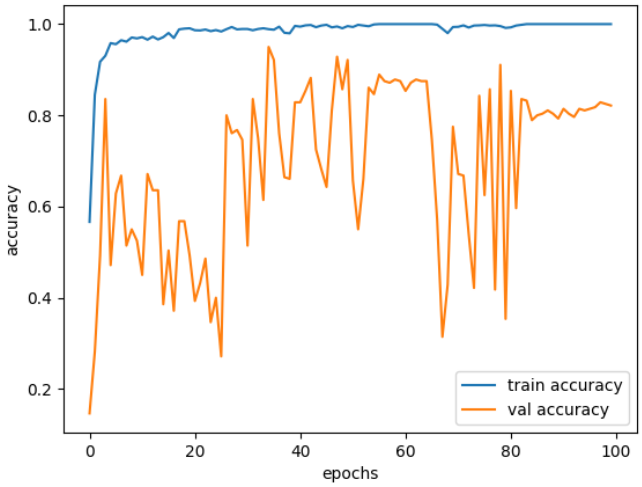


Figure 9 Accuracy over number of epochs

```
Epoch 1/100
40/40 [==============================] - 6s 114ms/step - loss: 1.1321 - accuracy: 0.5663 - val_loss: 1.8098 - val_accuracy: 0.1464
Epoch 2/100
40/40 [==============================] - 4s 107ms/step - loss: 0.4369 - accuracy: 0.8452 - val_loss: 2.2923 - val_accuracy: 0.2821
Epoch 3/100
40/40 [==============================] - 4s 112ms/step - loss: 0.2522 - accuracy: 0.9175 - val_loss: 1.0638 - val_accuracy: 0.4929
Epoch 4/100
40/40 [==============================] - 4s 111ms/step - loss: 0.2236 - accuracy: 0.9306 - val_loss: 0.5796 - val_accuracy: 0.8357
```

```
Epoch 96/100
40/40 [==============================] - 4s 104ms/step - loss: 7.5774e-05 - accuracy: 1.0000 - val_loss: 0.7668 - val_accuracy: 0.8143
Epoch 97/100
40/40 [==============================] - 4s 104ms/step - loss: 7.2544e-05 - accuracy: 1.0000 - val_loss: 0.7524 - val_accuracy: 0.8179
Epoch 98/100
40/40 [==============================] - 4s 104ms/step - loss: 1.6119e-04 - accuracy: 1.0000 - val_loss: 0.7287 - val_accuracy: 0.8286
Epoch 99/100
40/40 [==============================] - 4s 105ms/step - loss: 5.9007e-05 - accuracy: 1.0000 - val_loss: 0.7434 - val_accuracy: 0.8250
Epoch 100/100
40/40 [==============================] - 4s 106ms/step - loss: 1.1496e-04 - accuracy: 1.0000 - val_loss: 0.7413 - val_accuracy: 0.8214

Process finished with exit code 0
```

An accuracy score was also calculated using the Sklearn libraries metric function 'accuracy_score' which provides an accuracy score given inputs for true values and predicted values.

```
#Accuracy = (True Positives + True Negatives) / (Positives + Negatives)
accuracy = accuracy_score(Y_true, Y_pred)
print('Accuracy: %f' % accuracy)
```

```
Accuracy: 0.875714
```

It is important to note that while our accuracy rate is great, the utilization of accuracy does not provide information about the possible errors of the machine learning model in relation to new data that has not yet been implicated. Additionally, accuracy may provide its own challenges and issues as when comparing two models, the same accuracy measurements may show that the models perform differently for certain classes. Also, accuracy measures are not the best metrics to employ when the dataset is unbalanced. As such, Kumar [86] advises on the exercise of caution when using the model's accuracy measures to assess its performance and suggests that it be combined with other performance metrics, especially precision and recall which can provide a deeper understanding of the data in addition to its accuracy.

6.5.3. Measuring Precision

The percentage of labels that were correctly predicted positively is represented by the model precision score. Another name for precision is the positive predictive value. When it comes to the reliability of the machine learning model, false positives and false negatives are traded off using precision together with recall. Huilgo [87] states that precision is the measure of exactness or quality, and as such a model with high precision is the one that should be utilized in order to reduce false negatives. However it is important to note that class distribution has an impact on precision.

When the classes are severely unbalanced, the precision score is a helpful indicator of the accuracy of the forecast. It reflects the ratio of true positives to the total of true positives and false positives mathematically. It is mathematically represented as such:

$$Precision\ score = \frac{TP}{FP + TP}$$

Precision was measured for the speech-emotion recognition model using the Sklearn metric function ‘precision_score’ and provided an average precision score of 0.875714 with a weighted and macro average of 0.88 which indicated a high degree of precision in addition to accuracy for the trained model. While the value did vary with the lowest precision value coming at 0.65 which is still a positive value, the model has on numerous occasions achieved a 1.00. However, precision as a performance metric comes hand-in-hand with recall as they in combination provide a deep dive and understanding for the reliability of the machine learning model.

6.5.4 Measuring Recall

According to Kumar [88], a model's ability to correctly forecast positives out of real positives is measured by the model recall score. This differs from precision, which counts the proportion of accurate positive predictions among all positive predictions given by models. In other words, it assesses how well the machine learning model is able to distinguish between all true positives and all false positives inside a dataset.

Huilgo [89] states that the machine learning model is more adept at recognizing both positive and negative samples if it has a higher the recall score. Sensitivity or the true positive rate are other names for recall. A high recall score shows how well the model can locate examples of success. A model with a high recall would be the one that should be utilized in order to reduce false positives. In order to obtain and be provided with a complete and complex picture of the model's performance, recall is often combined with additional performance metrics like precision and accuracy. It symbolizes the ratio of true positives to the total of true positives and false negatives in mathematics.

$$Recall\ score = \frac{TP}{FP + TP}$$

The recall score for the trained speech-emotion recognition model stands at the average value of 0.875714, and with a macro and weighted average of 0.88 While this provides us with the argument that there is extremely high recall for our model, it is important to note that there is a large discrepancy as the minimum recorded value stands at 0.55 (which is still a positive value) and a maximum value of 1.00 multiple times. The most fascinating aspect would be that there is a predominance of a almost binary relationship between epochs as they are either between 0.55-0.6 or near or at 1.00 value. This however does not take away from the high recall value of the trained model and indicates that the model successful and often indicated cases of success.

6.5.5. Measuring F-1 Score

Lastly, according to Korstanje [90], The model score as a function of recall and precision is represented by the model F1 score. A substitute for accuracy metrics, F-score is a machine learning model performance metric that equally weights Precision and Recall when assessing how accurate the model is. Bhandari [91] states that it is conceptualized mathematically as a harmonic mean of precision and recall score.

$$F1\ score = \frac{2}{\frac{1}{Recall} + \frac{1}{Precision}}$$

The F1 score gained from training the speech-emotion recognition model stands at 0.875714 and a weighted and macro average of 0.87.

6.5.6 Classification Report

	precision	recall	f1-score	support
0	0.65	1.00	0.79	400
1	0.92	1.00	0.96	400
2	1.00	1.00	1.00	400
3	0.80	1.00	0.89	400
4	1.00	1.00	1.00	400
5	1.00	0.55	0.71	400
6	0.99	0.58	0.73	400
accuracy			0.88	2800
macro avg	0.91	0.88	0.87	2800
weighted avg	0.91	0.88	0.87	2800

Using the Sklearn library's 'classification_report' metric function, we can get all the different scores for precision, recall, F1 and the support for each class (emotion) summarized in a table as well as macro and weighted averages and an overall accuracy value. The emotions in this report are represented by numbers ranging from 0 to 6 which are 'angry', 'disgust', 'fear', 'happy', 'pleasant surprise', and 'sad' respectively.

Given this table we can see that the angry emotion scored the least in precision by a considerable margin with a score of 0.65, however it's F1-score was moderately high (0.79) due to its perfect recall score.

The emotions disgust, fear, and neutral all scored highly in both precision and recall, and subsequently in F1 as well, with neutral having perfect scores in all 3, an indicator that those emotions were most likely the most unique in regards to their features, making them the easiest to single out.

The emotion happy has a moderate score in precision, however it is still much lower than most of the emotions excluding angry which seems to be an outlier. However, it does still have an overall F1 score of 0.89. There are 2 other outliers, those being pleasant surprise and sad, There are 2 other outliers, those being the emotions pleasant surprise and sad, which both scored poorly in recall (0.55 and 0.58) which is evident in the confusion matrix, with almost perfect scores in precision.

6.6 Challenges

The main challenge was learning about LSTM as well as finding a database that would work well with it as LSTMs require larger amounts of data than CNNs.

There were difficulties displaying the spectograms as newer versions of the Librosa library does not have the necessary functions to plot audio waves nor spectograms, therefore an older version of the library had to be installed in order to display them.

There were difficulties getting a model with sufficient accuracy as majority of them would have accuracies under 60%, with the exception of 4, one of which did exceptionally well and was therefore used as the model for this thesis.

7. USING SR AND SER PRACTICALLY

The previous section went over data preparation, building, testing and using models with Tensorflow, as well as learning about machine learning concepts. This section of the thesis is focused on the attempted utilization of speech recognition and speech emotion recognition in a practical environment, along with an MVP (minimum viable product) being produced using different frameworks and APIs.

7.1 Technologies

7.1.1 GitHub

Git is software for tracking changes in any group of files and is typically used to coordinate work among programmers who are working on source code collectively during software development [93]. Speed, data integrity, and support for dispersed, non-linear processes are among its objectives.

GitHub, Inc. is a web-based interface that specializes in Git-based software development and version control. It includes Git's distributed version control and source code management (SCM) features as well as its own, today it is frequently used to host open-source projects [94].

As GitHub allows for real-time collaboration, it consequently encourages teams to work together to build and edit their site content [95], while for individuals it provides a variety of useful tools in order to keep track of their software development process. Some of its highlights include access control and collaboration tools including bug tracking, feature requests, task management, continuous integration, and wikis. It is based in California and has been a Microsoft subsidiary since 2018 [96].

GitHub was utilized in order to store the code and track changes and mistakes during the realization of the thesis.

7.1.2 Electron

Electron is an open-source framework developed by GitHub. The framework allows its users to create desktop applications using web technologies. It is built on the Chromium browser engine and will therefore take up more resources than another framework such as WPF, UWP or .NET, however it is much more practical to use and allows for cross-platform app development upon project initialization, allowing the program to run on Windows, MacOS, Linux and more operating systems. Electron also uses Node.js for its backend, making it much more familiar to web developers [97].

Electron allows for integration with other frameworks such as Angular or React, essentially allowing the developer to use the frameworks they know how to use while allowing their apps to run as standalone desktop apps. Because of this it has been used in the development of many applications such as VS Code, Facebook Messenger, Twitch, Microsoft Teams, InVision, and many more [98].

7.1.3 APIs

The speech recognition model created in section 5 only recognizes 30 different words without context and both models in sections 5 and 6 require the input data to be of the same shape to work which poses a problem for implementing them in real-time scenarios [99]. Due to those limitations, APIs for more advanced pre-trained models were used in the development of the application for practicality, especially as this section is focused on implementing the technology in a practical setting.

8. SYSTEM ANALYSIS

8.1 Program Description

The idea was to have a system that will be able to incorporate both speech recognition and speech emotion recognition in real-time by taking audio sources from other applications and using both technologies to produce speech-to-text on the screen as well as a colour indicative of the emotion surrounding the text. It essentially creates subtitles on the screen, allowing for the user to move the subtitle box where they please.

8.2 User Requirements

8.2.1 Functional Requirements

- The user should be able to start and stop the app from working at any moment.
- The user should be able to move and resize the subtitle box.
- The subtitle box should always stay on top when it is activated.
- The system should work cross-platform.
- The system should be easy to use and navigate.
-

8.2.2 Non-Functional Requirements

- The system should accurately translate the speech to text.
- The system should accurately represent the emotion conveyed to the user.
- The system should be safe and secure.

8.3 Target Audience

- Deaf users or users with hearing impairment.
- Neurodivergent users.
- Users in a noisy environment.

9. SYSTEM DESIGN

9.1 Application UI

The UI is very simple, as the application only has one function and therefore would only need two buttons as shown in Figure 10.

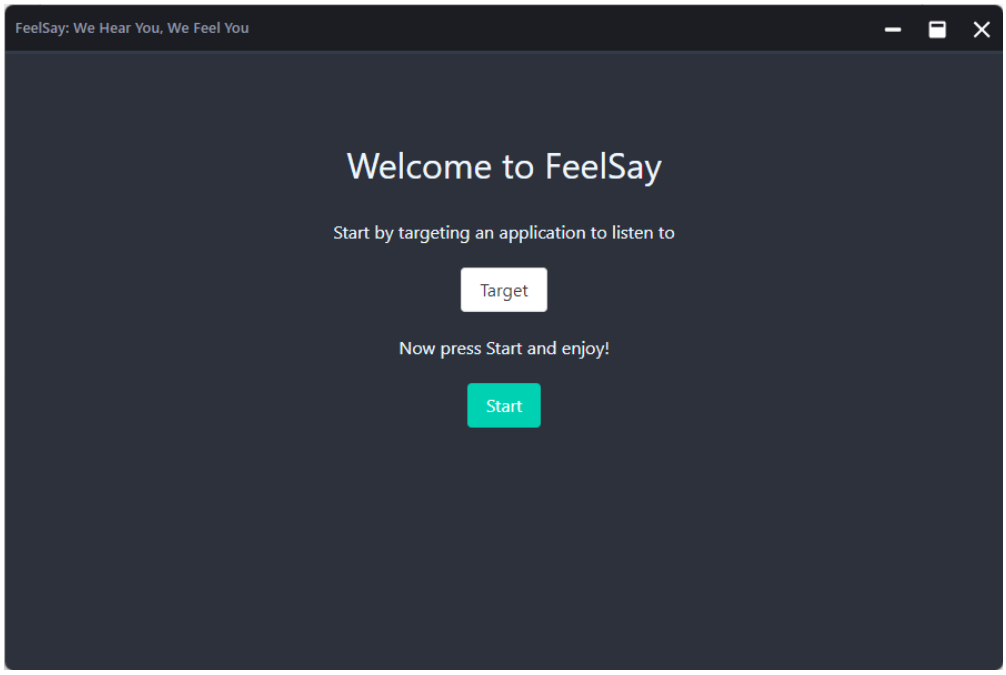


Figure 10 FeelSay starting screen

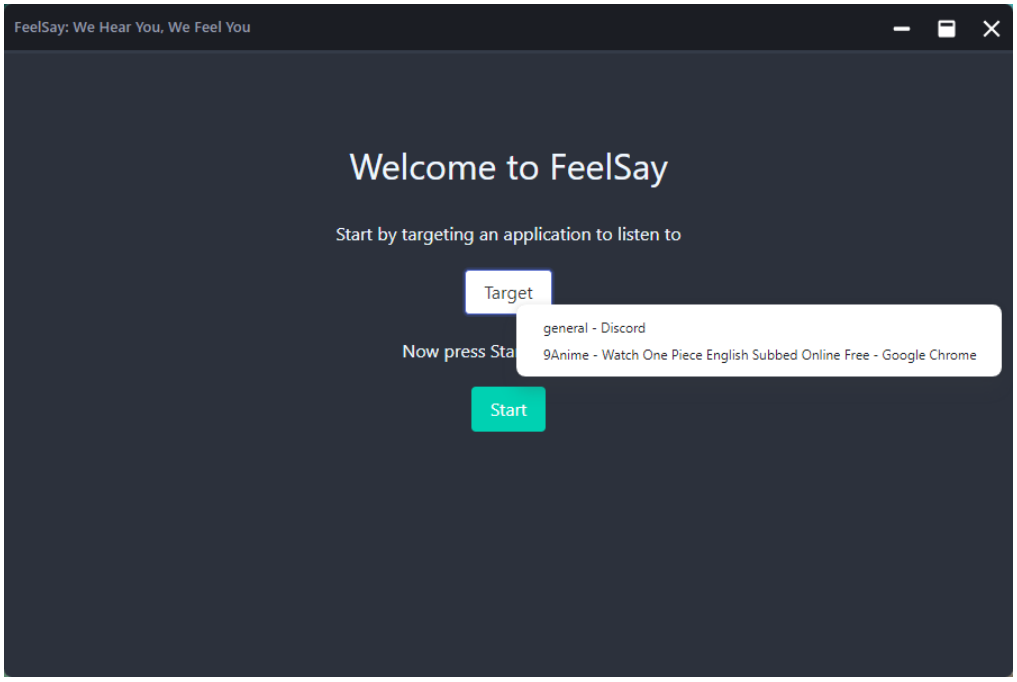


Figure 11 FeelSay starting screen with menu

Figure 11 shows what happens when the Target button is clicked.

Figure 12 shows the draggable and resizable subtitle box that is created when the start button is pressed.

Figure 12 Draggable subtitle box

9.2 Application Architecture

Electron's application structure has changed with a security update in November of 2021. On the very low level, the Electron framework bundles Chromium and Node into one framework, whereby Chromium renders the screens of the application and Node provides OS-level functionality. How Chromium works is that there is a main process which starts the application and allows Node functionality, and a renderer process that the main process creates as shown in Figure 13. [100]

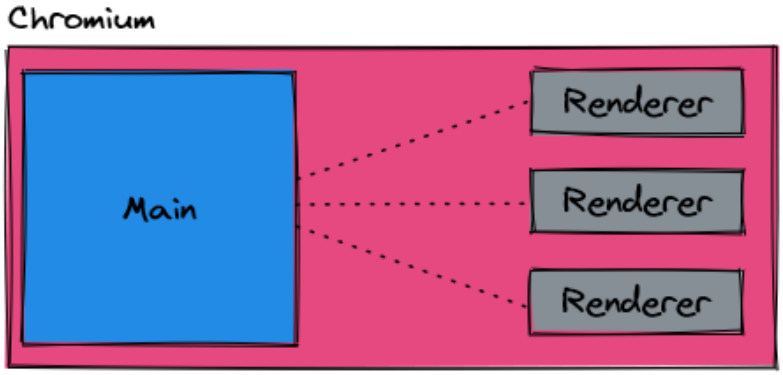


Figure 13 Chromium structure [100]

Due to the security update, the renderer process can no longer directly communicate to the main process as this opens up the possibility of the user being able to access Node functions through the application. Because of this, a new process called preload was made to establish communication between the main process and the renderer process, leaving the project with a structure like the one shown in Figure 14. [100]

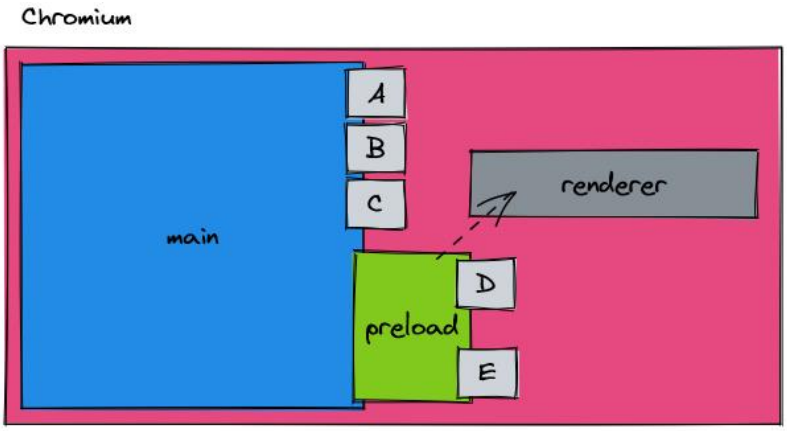


Figure 14 Chromium structure post update [100]

9.3 Programming the Application

To establish communication between the render process and the main process, in the preload script a contextBridge was used to establish a connection to the main process with API's defined for various functions that the render process could use in order to get access to Node functions using inter-process communication functions (ipcRenderer) to both send and receive requests.

```
// All of the Node.js APIs are available in the preload process.
// It has the same sandbox as a Chrome extension.
const { ipcRenderer, contextBridge, desktopCapturer } = require("electron");

// API - communicate with render process
const API = {
    audioSources: (data) => ipcRenderer.invoke("sources/get", data).then((result) => {}),
    closeApp: () => ipcRenderer.send('closeApp'),
    minimizeApp: () => ipcRenderer.send('minimizeApp'),
    maximizeRestoreApp: () => ipcRenderer.send('maximizeRestoreApp'),
    start: () => ipcRenderer.send('start'),
}

// window.electronAPI - communicate with main process
contextBridge.exposeInMainWorld("electronAPI", API);
```

The above code shows how to use the API from the render.js file. The API called would be used to get the available applications the user could target.

```
// //Get the available audio sources
targetBtn.addEventListener('click', () => {
    getAudioSources();
})

async function getAudioSources() {
    const sources = await electronAPI.audioSources();
}
```

The other applications can be targeted through the use of Electron's function desktopCapturer which utilizes the Node API navigator.mediaDevices.getUserMedia, allowing the application to recognize all other open applications on the system and capture data from them. As shown in the code below, the API endpoint in the main process used to get all of the currently open applications, display the names of the applications in a menu built from a template in Electron, and send back the selected application's information.

```
ipcMain.handle("sources/get", async (event, data) => {
  const inputSources = await desktopCapturer.getSources({
    types: ['window']
  });

  const audioOptionsMenu = Menu.buildFromTemplate(
    inputSources.map(source => {
      return {
        label: source.name,
        click: () => {
          console.log(source);
          return source;
        }
      };
    })
  );

  audioOptionsMenu.popup();
});
```

The source's audio stream should then be taken and sent to an API endpoint which converts the speech to text and displays the result which was returned into a second window called subtitleWindow. The API chosen for this task was made by AssemblyAI, and it works with both sending and receiving data, however the problem that stands is sending the audio stream from other applications as the Node API getUserMedia does not have the ability to get audio

data from a single source (targeted application in our case), and therefore sending data to the speech-to-text API poses a challenge.

The subtitle window is a smaller window that opens when the start button is clicked. The window contains a property which always keeps the window ‘on top’, meaning that it will be kept above the other applications as it is more practical for its use case, which satisfies one of the user requirements.

```
subtitleWindow.setAlwaysOnTop(true, 'screen-saver')
```

It is also draggable to allow the user to move the application to any part of the screen, fulfilling another functional requirement.

The first requirement was to start and stop the application from working so as to not intrude in the user’s privacy. This was achieved by initializing the electron project which gives it the ability to start, and to stop the application a button was created on the top right of the application to close the application as well as two other buttons to minimize and make the application full screen. These buttons were made using simple icon designs that stand out to the color scheme used for the application in order to provide a better user experience.

The simple nature of the application fulfills the requirement of it being easy to use and navigate, and due to Electron using Chromium as a platform for it’s applications, it fulfills the requirement of being cross-platform as Chromium supports almost all operating systems.

10. FACED CHALLENGES

The main challenge with the development of the application was with the Electron update in November of 2021. The update deprecated the use of desktopCapturer and other such Node functions within the render script due to security issues, meaning to get the audio sources, create the selection menu, as well as the close, restore/maximize, and minimize buttons a

context bridge would need to be used to establish IPC (inter-process communication) whereby an additional preload script would need to be used as a middleman to send API requests from the render script, which is directly responsible for the frontend of the application, to the main script, which has all the functionality.

Another major challenge when developing this application was finding an API for speech emotion recognition. As the model made in section 6 requires the input audio to be of the same shape as the training dataset, it cannot be used in a real-time speech emotion recognition system. There were also no other API's to use in order to make the app work for emotion recognition.

While on the topic of APIs, the initial API planned for use was the built in Web Speech API which had the limitation of only allowing for the microphone to feed it data and was therefore not optimal for this application. Then the AssemblyAI speech-to-text API was found which worked adequately, however the problem was then finding a way to send the audio stream as the Node API getUserMedia did not have the ability to get audio from a specific application and therefore the audio stream could not be sent to the speech-to-text API which impedes the desktop app to properly function.

Aside from not being able to get the audio data from other applications, the rest of the application's code works, meaning that the application would be finished if it had the audio stream sending data to the real-time API.

11. CONCLUSION

To conclude, the aim of this thesis was achieved as the models for both Speech Recognition and Speech Emotion Recognition turned out to be a success for both acquiring knowledge and putting it into practice by creating the models from the ground up. This taught complicated concepts such as Machine Learning and Neural Networks in an engaging manner.

The opportunity to not only research new areas of computer science and their possible social utilization but additionally gain the abilities and skills of model testing and implementation serve as a final conclusion to the inquiry and application of the gained knowledge and skills throughout undergraduate higher education.

While many challenges were faced, it is important to note that both trained models showed high degrees of validation accuracy, as well as the emotion recognition model scoring highly amongst other performance metrics, while the implementation provided a number of challenges which were turned into learning experiences.

Numerous attempts were made in order to address and uncover this issue; however, this challenge is one which would require a vastly higher skill and educational set. Nonetheless, the code in itself, along with building the models speak of the success of the obtained practical knowledge aspect and aim of the thesis.

The study as a whole can be viewed as an accomplishment as it not only provides an in-debt dive into a variety of fields, programs, software's, concepts and relevant models, as well as an overview of new skills gained, and practical knowledge expanded.

To solve the problem with the application, a method for obtaining another application's exclusive audio stream would have to be found. After having such a method all that would be

left is to preprocess the data before sending it to the speech-to-text API and displaying the response.

However, it is important to note that nothing is without the possibility of improvements, and the greatest that could be applied is that in future projects a larger dataset would be implemented in order to achieve greater accuracy and variation when it comes to audio files and present emotions, as well as possibly branching out to investigate the effects of moods and other external elements such as those of sleep deprivation or alcohol intoxication which are noted to greatly effect emotional processing and expression.

12. BIBLIOGRAPHY

[1] Lange, J., Heerdink, M. W., & Van Kleef, G. A. (2022). Reading emotions, reading people: Emotion perception and inferences drawn from perceived emotions. *Current opinion in psychology*, *43*, 85-90.

[2] Blanton, S. The voice and the emotions. *Q. Journal of Speech 1*, 2 (1915), 154172.

[1] Lange, J., Heerdink, M. W., & Van Kleef, G. A. (2022). Reading emotions, reading people: Emotion perception and inferences drawn from perceived emotions. *Current opinion in psychology*, *43*, 85-90.

[2] Blanton, S. The voice and the emotions. *Q. Journal of Speech 1*, 2 (1915), 154172.

[3] Schuller, B. "Speech Emotion Recognition: Two Decades in a Nutshell, Benchmarks, and Ongoing Trends", *Cacm.acm.org*, 2018. [Online]. Available: https://cacm.acm.org/magazines/2018/5/227191-speech-emotion-recognition/fulltext. [Accessed: 27- May- 2022]

[4] Ghosh, "The Fundamentals of Natural Language Processing and Natural Language Generation - DATAVERSITY", *DATAVERSITY*, 2022. [Online]. Available: https://www.dataversity.net/fundamentals-natural-language-processing-natural-language-generation/. [Accessed: 28- May- 2022]

[5] Dellaert, F., Polzin, T. and Waibel, A. Recognizing emotion in speech. In *Proceedings of ICSLP 3*, (Philadelphia, PA, 1996). IEEE, 19701973.

[6] Brownlee, J., 2022. *What Is Natural Language Processing?*. [online] Machine Learning Mastery. Available at: <https://machinelearningmastery.com/natural-language-processing/> [Accessed 25 January 2022].

[7] Goldberg, Y. and Hirst, G., 2017. *Neural Network Methods in Natural Language Processing*. San Rafael: Morgan & Claypool Publishers.

[8] Oxford Lexico, n.d. *COMPUTATIONAL LINGUISTICS | Meaning & Definition for UK English | Lexico.com*. [online] Lexico Dictionaries | English. Available at: <https://www.lexico.com/definition/computational_linguistics> [Accessed 4 March 2022].

[9] Education, I., 2022. *What is Natural Language Processing?*. [online] Ibm.com. Available at: <https://www.ibm.com/cloud/learn/natural-language-processing> [Accessed 13 January 2022].

[10] Banerjee, D. "Natural Language Processing (NLP) Simplified : A Step-by-step Guide", *Datascience.foundation*, 2020. [Online]. Available: https://datascience.foundation/sciencewhitepaper/natural-language-processing-nlp-simplified-a-step-by-step-guide. [Accessed: 17- May- 2022]

[11] Chowdhary, K. Natural language processing. 2020. *Fundamentals of artificial intelligence*, 603-649.

[12] Hirschberg, J., & Manning, C. D.. Advances in natural language processing. 2015. *Science*, *349*(6245), 261-266.

[13] Malik, U. "Python for NLP: Tokenization, Stemming, and Lemmatization with SpaCy Library", *Stack Abuse*, 2021. [Online]. Available: https://stackabuse.com/python-for-nlp-tokenization-stemming-and-lemmatization-with-spacy-library/. [Accessed: 01- Apr- 2022]

[14] Chavan, J. "NLP: Tokenization , Stemming , Lemmatization , Bag of Words ,TF-IDF , POS", *Medium*, 2020. [Online]. Available: https://medium.com/@jeevanchavan143/nlp-tokenization-stemming-lemmatization-bag-of-words-tf-idf-pos-7650f83c60be. [Accessed: 15- Apr- 2022]

[15] Saravia, E. "Fundamentals of NLP - Chapter 1 - Tokenization, Lemmatization, Stemming, and Sentence Segmentation", *Notebooks by dair.ai*, 2022. [Online]. Available: https://dair.ai/notebooks/nlp/2020/03/19/nlp_basics_tokenization_segmentation.html. [Accessed: 05- Jun- 2020]

[16] Reddy, D. R. Speech recognition by machine: A review. 1976. *Proceedings of the IEEE*, *64*(4), 501-531.

[17] Elliot B. Sloane, Ricardo J. Silva,

Chapter 83 - Artificial intelligence in medical devices and clinical decision support systems, Editor(s): Ernesto Iadanza, 2020. Clinical Engineering Handbook (Second Edition), Academic Press, Pages 556-568,

[18] Yu, Dong, and Li Deng. 2016. *Automatic speech recognition*. Vol. 1. Berlin: Springer.

[19] Ulkuhan Guler, Tuna B. Tufan, Aatreya Chakravarti, Yifei Jin, Maysam Ghovanloo,

Implantable and Wearable Sensors for Assistive Technologies, 2021. Reference Module in Biomedical Sciences, Elsevier, 3:14.

[20] Ibid

[21] Unknown, "Speech Emotion Recognition Project using Machine Learning", *ProjectPro*, 2021. [Online]. Available: https://www.projectpro.io/article/speech-emotion-recognition-project-using-machine-learning/573. [Accessed: 10- May- 2022]

[22] Wadhwa, A., Gupta ,P. and Pandey, B. "SPEECH EMOTION RECOGNITION (SER) THROUGH MACHINE LEARNING", *Analyticsinsight.net*, 2021. [Online]. Available: https://www.analyticsinsight.net/speech-emotion-recognition-ser-through-machine-learning/. [Accessed: 10- May- 2022]

[23] Juang, Biing-Hwang, and Lawrence R. Rabiner. "Automatic speech recognition–a brief history of the technology development." 2015. *Georgia Institute of Technology. Atlanta Rutgers University and the University of California. Santa Barbara* 1:67.

[24] Pieraccini, R., & Director, I. C. S. I. From audrey to siri. 2021. *Is speech recognition a solved problem*, 23. 2:9.

[25] Reddy, R., 2013. CMU Harpy System 1976 - Beam Search. [online] Youtube.com. Available at: <https://www.youtube.com/watch?v=32KKg3aP3Vw> [Accessed 17 March 2022].

[26] Slideplayer.com. 2022. *Speech Recognition SR has long been a goal of AI - ppt download*. [online] Available at: <https://slideplayer.com/slide/6256663/> [Accessed 5 May 2022].

[27] Franzese, M. and Iuliano, A. Hidden Markov Models, 2019. Editor(s): Shoba Ranganathan, Michael Gribskov, Kenta Nakai, Christian Schönbach, Encyclopedia of Bioinformatics and Computational Biology, Academic Press, Pages 753-762,

[28] Gales, M. and Young, S., 2007. The Application of Hidden Markov Models in Speech Recognition. [online] Available at: <https://mi.eng.cam.ac.uk/~mjfg/mjfg_NOW.pdf> [Accessed 9 April 2022].

[29] Nallammal, N., and V. Radha. "Performance evaluation of face recognition based on pca, lda, ica and hidden markov model." 2010. *International Conference on Data Engineering and Management*. Springer, Berlin, Heidelberg.

[30] Waibel, A.; Hanazawa, T.; Hinton, G.; Shikano, K.; Lang, K. J. "Phoneme recognition using time-delay neural networks". 1989. *IEEE Transactions on Acoustics, Speech, and Signal Processing*. **37** (3): 328–339.

[31] Bird, Jordan J.; Wanner, Elizabeth; Ekárt, Anikó; Faria, Diego R. "Optimisation of phonetic aware speech recognition through multi-objective evolutionary algorithms" (PDF). 2020. *Expert Systems with Applications*. Elsevier BV. **153**: 113402

[32] S. A. Zahorian, A. M. Zimmer, and F. Meng. "Vowel Classification for Computer based Visual Feedback for Speech Training for the Hearing Impaired," 2020. *ICSLP*

[33] Waibel, Alex "Modular Construction of Time-Delay Neural Networks for Speech Recognition" (PDF). 1989. *Neural Computation*. **1** (1): 39–46.

[34] Rumelhart, David E.; Hinton, Geoffrey E.; Williams, Ronald J.). "Learning representations by back-propagating errors". 1986. *Nature*. **323** (6088): 533–536.

[35] Smith, C. S."The Man Who Helped Turn Toronto into a High-Tech Hotbed". 2017. *The New York Times.*

[36] Schmidhuber, J. "Deep learning in neural networks: An overview". 2015. *Neural Networks*. **61**: 85–117

[37] Schuller, B. "Speech Emotion Recognition: Two Decades in a Nutshell, Benchmarks, and Ongoing Trends", *Cacm.acm.org*, 2018. [Online]. Available: https://cacm.acm.org/magazines/2018/5/227191-speech-emotion-recognition/fulltext. [Accessed: 27- May- 2022]

[38] Schuller, B. and Batliner, A. *Computational Paralinguistics: Emotion, Affect and Personality in Speech and Language Processing.* Wiley, 2013.

[39] Gunes, H. and Schuller, B. Categorical and dimensional affect analysis in continuous input: Current trends and future directions. *Image and Vision Computing 31*, 2 (2013), 120136.

[40] Devillers, L., Vidrascu, L. and Lamel, L. Challenges in real-life emotion annotation and machine learning based detection. *Neural Networks 18*, 4 (2005), 407422.

[41] Watson, D., Clark, L.A., and Tellegen, A. Development and validation of brief measures of positive and negative affect: the PANAS scales. *J. of Personality and Social Psychology 54*, 6 (1988), 106

[42] Pavol Harar, Jesus B Alonso-Hernandezy, Jiri Mekyska, Zoltan Galaz, Radim Burget, and Zdenek Smekal. 2017. "Voice pathology detection using deep learning: a preliminary study." In Bioinspired Intelligence (IWOBI), 2017. *International Conference and Workshop on. IEEE*, 1–4

[43] Kun Han, Dong Yu, and Ivan Tashev. Speech emotion recognition using deep neural network and extreme learning machine. 2014. In *Fifteenth annual conference of the international speech communication association*

[44] Che-Wei Huang and Shrikanth S Narayanan. Attention Assisted Discovery of Sub-Utterance Structure in Speech Emotion Recognition.. 2016. *In INTERSPEECH*. 1387–1391.

[45] Efthymios Tzinis and Alexandras Potamianos. Segment-based speech emotion recognition using recurrent neural networks. In Affective Computing and Intelligent Interaction (ACII), 2017 *Seventh International Conference on. IEEE*, 190–195

[46] Fayek, H.M., Lech, M., and Cavedon, L. 2017. Evaluating deep learning architectures for Speech Emotion Recognition. *Neural Networks* 92 (2017), 60–68.

[47] Trigeorgis, G., Ringeval, F., Brueckner, R., Marchi, E., Nicolaou, A.N., Schuller, B. and Zafeiriou. S. Adieu features? end-to-end speech emotion recognition using a deep convolutional recurrent network. In *Acoustics, Speech and Signal Processing (ICASSP)*, 2016. *IEEE International Conference on. IEEE*, 5200–5204.

[48] Ringeval, F., Sonderegger, A., Sauer, J. and Lalanne, D. Introducing the RECOLA multimodal corpus of remote collaborative and affective interactions. In Automatic Face and

Gesture Recognition (FG), 2013. *The IEEE International Conference and Workshops* on. IEEE, 1–8

[49] Ververidis, D. and Kotropoulos, C. Emotional speech recognition: Resources, features, and methods. *Speech Commun. 48*, 9 (2006), 11621181.

[50] Zeng, Z., Pantic, M., Roisman, G.I., and Huang, T.S. A survey of affect recognition methods: Audio, visual, and spontaneous expressions. *IEEE Trans. Pattern Analysis and Machine Intelligence 31*, 1 (2009), 3958.

[51] Liu, J., Chen, C., Bu, J., You, M. and Tao, J. Speech emotion recognition using an enhanced co-training algorithm. In *Proceedings ICME.* (Beijing, P.R. China, 2007). IEEE, 9991002.

[52] Devillers, L., Vidrascu, L. and Lamel, L. Challenges in real-life emotion annotation and machine learning based detection. *Neural Networks 18*, 4 (2005), 407422.

[53] Jinyu Li, Li Deng, Reinhold Haeb-Umbach, Yifan Gong, Chapter 2 - *Fundamentals of speech recognition*, Editor(s): Jinyu Li, Li Deng, Reinhold Haeb-Umbach, Yifan Gong, Robust Automatic Speech Recognition, Academic Press, 2016, Pages 9-40,

[54] Statista Research Department, "U.S. voice assistant use by device 2021 | Statista", *Statista*, 2022. [Online]. Available: https://www.statista.com/statistics/1274398/voice-assistant-use-by-device-united-states/. [Accessed: 17- May- 2022]

[55] Jacob Aron. How innovative is Apple's new voice assistant, Siri?, 2011. *New Scientist*, Volume 212, Issue 2836, Page 24

[56] Lopatovska, Irene, et al. "Talk to me: Exploring user interactions with the Amazon Alexa." 2019. *Journal of Librarianship and Information Science* 51.4. 984-997.

[57] Bhat, Heena Reyaz, Tanveer Ahmad Lone, and Zubair M. Paul. "Cortana-intelligent personal digital assistant: a review." 2017. *International Journal of Advanced Research in Computer Science* 8.7. 55-57.

[58] Doerrfeld, B. "20+ Emotion Recognition APIs That Will Leave You Impressed, and Concerned | Nordic APIs |", *Nordic APIs*, 2022. [Online]. Available: https://nordicapis.com/20-emotion-recognition-apis-that-will-leave-you-impressed-and-concerned/. [Accessed: 27- May- 2018]

[59] Ibid

[60] Jordan, M. I., & Mitchell, T. M. Machine learning: Trends, perspectives, and prospects. 2015. *Science*, *349*(6245), 255-260.

[61] El Naqa, I., & Murphy, M. J. What is machine learning?. 2015. In *machine learning in radiation oncology* (pp. 3-11). Springer, Cham.

[62] Jordan, M. I., & Mitchell, T. M. Machine learning: Trends, perspectives, and prospects. 2015. *Science*, *349*(6245), 255-260.

[63] Terra, J. Keras vs Tensorflow vs Pytorch: Key Differences Among the Deep Learning Framework. 2022. In Simply Learn. Available at: <https://www.simplilearn.com/keras-vs-tensorflow-vs-pytorch-article>[Accessed: 03 May 2022]

[64] Abadi, M et al. *TensorFlow: A System for Large-Scale Machine Learning.* 2016. Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation

[65] Pang, B., Nijkamp, E., & Wu, Y. N. Deep learning with tensorflow: A review. 2020. *Journal of Educational and Behavioral Statistics*, *45*(2), 227-248.

[66] Terra, J. Keras vs Tensorflow vs Pytorch: Key Differences Among the Deep Learning Framework. 2022. In Simply Learn. Available at: <https://www.simplilearn.com/keras-vs-tensorflow-vs-pytorch-article>[Accessed: 03 May 2022]

[67] Pang, B., Nijkamp, E., & Wu, Y. N. Deep learning with tensorflow: A review. 2020. *Journal of Educational and Behavioral Statistics*, *45*(2), 227-248.

[68] Kuhlman, D. *A Python Book: Beginning Python, Advanced Python, and Python Exercises*. New York City, NY: Platypus Global Media. 2011.

[69] Valueva, M.V.; Nagornov, N.N.; Lyakhov, P.A.; Valuev, G.V.; Chervyakov, N.I. "Application of the residue number system to reduce hardware costs of the convolutional neural network implementation". 2020. *Mathematics and Computers in Simulation*. Elsevier BV. 177: 232–243

[70] Heideman, Michael T.; Johnson, Don H.; Burrus, Charles Sidney "Gauss and the history of the fast Fourier transform" 1984. *IEEE ASSP Magazine*. **1** (4): 14–21.

[71] GeeksforGeeks. 2022. *CNN | Introduction to Pooling Layer - GeeksforGeeks*. [online] Available at: <https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/#:~:text=Max%20pooling%20is%20a%20pooling,of%20the%20previous%20feature%20map> [Accessed 13 May 2022].

[72] Hinz, Tobias & Barros, Pablo & Wermter, Stefan. (2016). The Effects of Regularization on Learning Facial Expressions with Convolutional Neural Networks. 80-87. 10.1007/978-3-319-44781-0_10.

[73] Masuyama, E. A number of fundamental emotions and their definitions. 1994. In *Proceedings of 1994 3rd IEEE International Workshop on Robot and Human Communication* (pp. 156-161)

[74] Kumar, A. "Accuracy, Precision, Recall & F1-Score – Python Examples", Data Analystics, 2022. [Online]. Available: https://vitalflux.com/accuracy-precision-recall-f1-score-python-example/. [Accessed: 28- Jun- 2022]

[75] Ibid.

[76] Kumar, A. "Confusion Matrix Explained with Python Code Examples", Data Analytics, 2019. [Online]. Available: https://vitalflux.com/models-confusion-matrix-examples/. [Accessed: 29- Jun- 2022]

[77] Bhandari, A. "Confusion Matrix for Machine Learning", Analytics Vidhya, 2020. [Online]. Available: https://www.analyticsvidhya.com/blog/2020/04/confusion-matrix-machine-learning/. [Accessed: 29- Jun- 2022]

[78] Pettinelli, M. *The psychology of emotions, feelings and thoughts*. 2012. Lightning Source.

[79] Koolagudi, S. G., & Rao, K. S. Emotion recognition from speech: a review. *International journal of speech technology*. 2012. 15(2), 99-117.]

[80] JIN, Qin, et al. Speech emotion recognition with acoustic and lexical features. In: 2015 *IEEE international conference on acoustics, speech and signal processing (ICASSP*). IEEE, 2015. p. 4749-4753.

[81] Roach, P. Techniques for the phonetic description of emotional speech. 2000. In *ISCA Tutorial and Research Workshop (ITRW) on Speech and Emotion.*

[82] Ibid.

[83] Eady, S.J. "Differences in the F0 patterns of speech: Tone language versus stress language." *Language and speech* 25.1 (1982): 29-42.

[84] Li, F.. "Language-specific developmental differences in speech production: A cross-language acoustic study." Child development 83.4 (2012): 1303-1315.

[85] Gad, A. "Accuracy, Precision, and Recall in Deep Learning | Paperspace Blog", *Paperspace Blog*, 2020. [Online]. Available: https://blog.paperspace.com/deep-learning-metrics-precision-recall-accuracy/. [Accessed: 28- Jun- 2022]

[86] Kumar, A. "Accuracy, Precision, Recall & F1-Score – Python Examples", *Data Analystics*, 2022. [Online]. Available: https://vitalflux.com/accuracy-precision-recall-f1-score-python-example/. [Accessed: 28- Jun- 2022]

[87] Huilgol, P. "Precision vs Recall | Precision and Recall Machine Learning", Analytics Vidhya, 2020. [Online]. Available: https://www.analyticsvidhya.com/blog/2020/09/precision-recall-machine-learning/. [Accessed: 27- Jun- 2022]

[88] Kumar, A. "Accuracy, Precision, Recall & F1-Score – Python Examples", Data Analystics, 2022. [Online]. Available: https://vitalflux.com/accuracy-precision-recall-f1-score-python-example/. [Accessed: 28- Jun- 2022]

[89] Huilgol, P. "Precision vs Recall | Precision and Recall Machine Learning", *Analytics Vidhya*, 2020. [Online]. Available: https://www.analyticsvidhya.com/blog/2020/09/precision-recall-machine-learning/. [Accessed: 27- Jun- 2022]

[90] Korstanje, J. "The F1 score", *Towards Data Science*, 2021. [Online]. Available: https://towardsdatascience.com/the-f1-score-bec2bbc38aa6. [Accessed: 29- Jun- 2022]

[91] Bhandari, A. "Confusion Matrix for Machine Learning", *Analytics Vidhya*, 2020. [Online]. Available: https://www.analyticsvidhya.com/blog/2020/04/confusion-matrix-machine-learning/. [Accessed: 29- Jun- 2022]

[93] Spinellis, D. Git. 2012. *IEEE software*, *29*(3), 100-101.

[94] Blischak, J. D., Davenport, E. R., & Wilson, G. A quick introduction to version control with Git and GitHub. 2016. *PLoS computational biology*, *12*(1), e1004668.

[96] Unknown., "An Introduction to GitHub", *Digital.gov*, 2022. [Online]. Available: https://digital.gov/resources/an-introduction-github/. [Accessed: 04- May- 2022].

[96] Ibid

[97] Scoccia, Gian Luca, and Marco Autili. "Web frameworks for desktop apps: An exploratory study." 2020. *Proceedings of the 14th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*.

[98] Prylipko, Dmytro, et al. "Zanzibar OpenIVR: an open-source framework for development of spoken dialog systems." 2011. *International Conference on Text, Speech and Dialogue*. Springer, Berlin, Heidelberg,

[99] Ofoeda, Joshua, Richard Boateng, and John Effah. "Application programming interface (API) research: A review of the past to inform the future." 2019. *International Journal of Enterprise Information Systems (IJEIS)* 15.3. 76-95.

[100] Patel, Z. "The ultimate Electron guide", *Debug & Release*, 2022. [Online]. Available: https://www.debugandrelease.com/the-ultimate-electron-guide/]. [Accessed: 30- Jun- 2022]