A COMPREHENSIVE EVALUATION ON THE APPLICATIONS OF DATA AUGMENTATION, TRANSFER LEARNING AND IMAGE ENHANCEMENT IN DEVELOPING A ROBUST SPEECH EMOTION RECOGNITION SYSTEM

By

Kyle Philip Calabro, B.Sc CS

A thesis submitted to the Graduate Committee of

Ramapo College of New Jersey in partial fulfillment

of the requirements for the degree of

Master of Science

with a Major in Data Science

December 2021

Committee Members:

Scott Frees, Advisor

Osei Tweneboah, Reader

John Kerrigan, Reader

COPYRIGHT

by

Kyle Philip Calabro

2021

FAIR USE AND AUTHOR'S PERMISSION STATEMENT

Fair Use

Duplication Permission

ACKNOWLEDGEMENTS

The work accomplished within this thesis would not have been possible without the assistance of my mentors. First, I would like to thank my thesis advisor Dr. Scott Frees, whose poise and professionalism during an open house event in November of 2013 was a large influence in my decision applying to Ramapo College of New Jersey through early decision for undergrad as a Computer Science major. Since then, I have thoroughly enjoyed his tutelage and am thankful for him having guided me throughout the process of developing this thesis. Next, I would like to thank Dr. Osei Tweneboah, my first thesis reader, for igniting my passion for Machine Learning and maintaining a wealth of knowledge with regards to neural networks, in turn being an excellent resource for the research and work performed in this thesis. I would also like to thank Dr. John Kerrigan of Rutgers University for serving as a reader on the thesis committee. Dr. Kerrigan's expertise in mathematics and assistance allowed me to fully grasp the concepts behind the tools and techniques I utilized throughout this project to form a fundamental level of understanding for the real mechanisms at work on a low level. Dr. Kerrigan has also been a long-time mentor of mine that has been with me throughout my academic career, and for that I could not be more grateful.

Finally, I would like to thank my family. My journey throughout academia has been extensive in nature, without their support, motivation and influence to consistently strive to think differently, push boundaries and master new skills, the completion of this thesis would not have come to fruition.

TABLE OF CONTENTS

Page

LIST OF TABLES

TABLE **PAGE**

LIST OF FIGURES

Figure **Page**

ABSTRACT

Within this thesis work, the applications of data augmentation, transfer learning, and image enhancement techniques were explored in great depth with respect to speech emotion recognition (SER) via convolutional neural networks and the classification of spectrogram images. Speech emotion recognition is a challenging subset of machine learning with an incredibly active research community. One of the prominent challenges of SER is a lack of quality training data. The methods developed and presented in this work serve to alleviate this issue and improve upon the current state-of-the-art methodology.

A novel unimodal approach was taken in which five transfer learning models pre-trained on the ImageNet dataset were used with both the feature extraction and fine-tuning method of transfer learning. Such transfer learning models include the VGG-16, VGG-19, InceptionV3, Xception and ResNet-50. A modified version of the AlexNet deep neural network architecture was utilized as a baseline for non-pre-trained deep neural networks. Ensemble learning methods were deployed to further improve classification performance.

Two speech corpora were utilized to develop these methods. The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS) and the Crowd-sourced Emotional Multimodal Actors Dataset (CREMA-D). Data augmentation techniques were applied to the raw audio samples of each speech corpora to increase the amount of training data, yielding custom datasets. Raw audio data augmentation techniques include the addition of Gaussian noise, stretching by two different factors, time shifting and shifting pitch by three separate tones. Image enhancement techniques were implemented with the aim of improving classification accuracy by constructing invariant features that may be identified as more prominent features in the

spectrograms by a convolutional neural network. Image enhancement techniques include conversion to grayscale, contrast-stretching and the combination of grayscale conversion followed by contrast-stretching. In all, 180 experiments were conducted to provide a comprehensive overview of all techniques that were proposed as well as a definitive methodology. Such methodology yields improved or comparable results to what is currently considered to be state-of-the-art when deployed on the RAVDESS and CREMA-D speech corpora.

The research performed in this thesis produces a unimodal methodology yielding 85% and 66% classification accuracy when deployed on the RAVDESS and CREMA-D speech corpora, respectively, through model averaging ensembles. The individual models composing such ensembles exhibit classification accuracies of 81 - 84% and 61 - 64% when deployed on the RAVDESS and CREMA-D speech corpora, respectively. Such performance is achieved through utilizing transfer learning employed through convolutional neural networks pre-trained on the ImageNet dataset and convolutional neural networks utilizing a modified AlexNet architecture in conjunction with raw audio data augmentation techniques that serve to increase the amount of available training data. This is representative of a five to twenty and four to eight percent increase in classification performance over unimodal state-of-the-art methods found in literature for the RAVDESS and CREMA-D speech corpora, respectively.

I. INTRODUCTION

Inherently, humans are social creatures by nature. Interdependence is a concept that has been deeply rooted in our society for centuries in which communication serves as the foundation for any community. Through communication, valuable information is exchanged, thoughts and feelings are shared with others, together this empowers advancement on a societal level. Emotional *valence* is a quantity that categorizes the various types of human emotions. Valence is measured across a spectrum in which emotions such as happiness and excitement are labeled as emotions exhibiting positive valence. In contrast, emotions such as sadness and fear are considered to exhibit negative valence. By accurately determining the emotional valence in a social interaction, the listener can make an informed decision on a best course of action in return [1]. Similarly, arousal is also a quantity that is utilized when categorizing emotions. Arousal measures an emotion's intensity. As such, emotions such as calm and boredom exhibit lower levels of arousal, while emotions such as nervousness and excitement exhibit higher levels of arousal. Together, valence and arousal form the two dimensions of emotion classification in a dimensional model. Figure 1 displays such a model.

Researchers who agree upon this dimensional model of emotions believe that all emotions fall somewhere within this two-dimensional space. In turn, they believe that a standard neurophysiological system is responsible for creating all emotions we exhibit as humans. Other psychologists in the field disagree with this categorization, however, and feel that every individual emotion is generated from a different neural system.

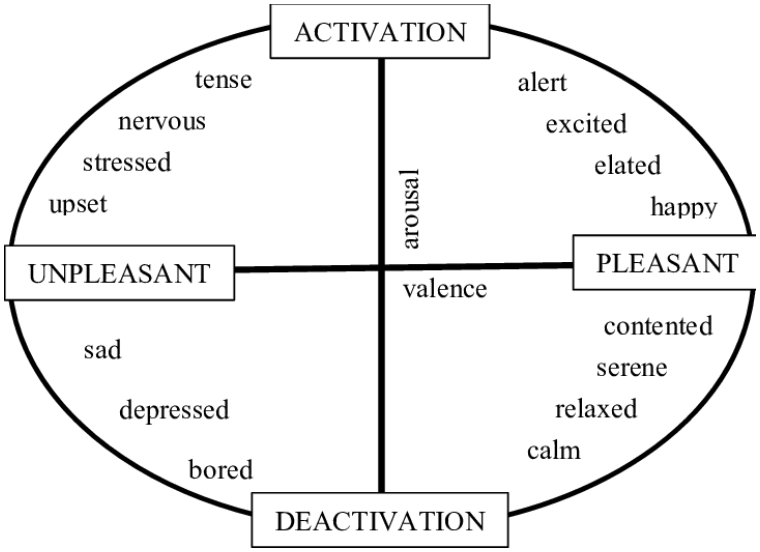

Figure 1. The Circumplex Model [2].

Human interaction can be divided into two distinct categories, verbal and non-verbal. Within non-verbal communication there exist two subcategories - communication through facial expression and communication via body language. Dating back to 1971, Albert Mehrabian developed the *7-38-55 rule* of personal communication [3]. As defined by this rule, in any given social interaction, seven percent of the information being conveyed comes from spoken words, thirty-eight percent comes from vocal tone and fifty-five percent comes from a speakers' body language.

Also in 1971, the work published in [4] outlined the six universally recognized emotions observed throughout all cultures around the world. These emotions include happiness, sadness, surprise, anger, fear and disgust. Within this thesis, eight total emotions were considered, the six aforementioned emotions listed in [4] as well as the neutral and calm emotions.

In general, understanding a person's emotions conveyed through speech is a relatively elementary task that we as humans practice daily. However, the rise of COVID-19 also generated an increase in the number of hospital patients utilizing ventilators as a form of treatment.

Intubation from a ventilator can result in pressure damage which “may lead to ulverations, inflammation, scarring, or a laryngeal granuloma, an inflammatory growth caused by injury to the covering of arytenoids” as well as “paralysis/paresis of the vocal cords” [5]. Such injuries and their accompanying ailments largely result in difficulties with speech and communication in patients. Traditionally, a Speech Language Pathologist would be brought in to provide therapy that can alleviate such difficulties. Traditional speech therapy, however, is a lengthy process that can take months, leaving a unique opportunity in which a Speech Emotion Recognition (SER) system could assist in bridging the gap between therapy and everyday life for such patients. SER is formally defined as “the task of recognizing the emotional aspects of speech irrespective of the semantic contents” [6].

SER is a challenging and ongoing subject of research in the discipline of machine learning due to a number of compounding factors [6]. To begin, there does not exist a large amount of databases that are suitable to be utilized for SER. Furthermore, within the databases that do exist, “only minimal data is available for training purposes” [6]. To further complicate matters, these databases tend to be imbalanced with respect to gender representation and emotional classes [6]. These factors have resulted in relatively low performance in terms of classification accuracy for SER studies in machine learning [6]. Many studies have also shown that performance is dependent upon the type of emotional expressions used in generating the SER database [6]. The aim of the research within this thesis is to explore new avenues of machine learning techniques to improve the classification accuracy of SER systems and alleviate some of the issues associated with the task.

The research performed in this thesis produces a unimodal methodology yielding 85% and 66% classification accuracy when deployed on the RAVDESS and CREMA-D speech corpora, respectively, through a model averaging ensemble. Such performance is achieved through utilizing transfer learning employed through convolutional neural networks pre-trained on the ImageNet dataset and convolutional neural networks utilizing a modified AlexNet architecture in conjunction with raw audio data augmentation techniques that serve to increase the amount of available training data. This is representative of a five to twenty and four to eight percent increase in classification performance over unimodal state-of-the-art methods currently found in literature for the RAVDESS and CREMA-D speech corpora, respectively.

Related Works

In recent years speech signal processing has made considerable technological advancements. Some applications of these advancements include mental stress detection [7], automatic speech recognition [8], language recognition [9] and speaker recognition. Another sizable application of speech signal processing is speech emotion recognition (SER). A speech emotion recognition system has a wide variety of use-cases. These applications range from speech language pathology therapy to assessing customer satisfaction in call centers or even deployment in artificial intelligence systems such as robots to provide the capability of expressing more empathy towards humans dynamically based on the emotions they exhibit [10], [6].

The deployment of machine learning techniques for applications in speech signal processing has become increasingly more popular given the advancements in computational power over the past few decades. Some particular machine learning and deep learning methods have been proven to be better suited for applications involving speech signal processing. Such methods include the support vector machine (SVM), the multilayer perceptron (MLP) and the recurrent neural network (RNN). Classical approaches to speech emotion recognition extract features from the raw audio such as energy, pitch, mel-frequency cepstral coefficients (MFCC), perceptual linear predictive (PLP) and filter banks as utilized in [11]. Many state-of-the-art methods employ the use of convolutional neural networks (CNN) by generating spectrogram images from the raw audio, transforming speech emotion recognition into an image classification task. Some recently proposed methods have also turned to multimodal approaches which employ the use of both audio and visual data (video), known as audio-visual emotion recognition [12],

[13]. Unfortunately, any approach to speech emotion recognition suffers from common issues in a lack of quality training data and feature selection [6]. As such, many different approaches have been taken to work around the hardships that encompass speech emotion recognition with respect to machine learning.

The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS) speech corpus is renowned for its high quality of audio recording, thanks to its sampling rate, and is widely used amongst researchers as a benchmark dataset. The work presented in [14] implemented an attention-based CNN model that exhibited 77.4% classification accuracy when trained and evaluated on the RAVDESS speech corpus utilizing the addition of noise in various forms. Similarly, in [15] a one-dimensional CNN model was trained on raw audio features such as chromagram, mel-scale spectrogram values, tonnetz representation, and spectral contrast features achieving a classification accuracy of 71.61% when trained and evaluated on the RAVDESS speech corpus. Multimodal methods have also been utilized within speech emotion recognition. Within the work of [16], a multimodal neural framework was proposed to perform fine-grained emotion recognition via a temporal alignment mean-max pooling operation and a cross modality excitation mechanism to capture intercorrelations between different modalities. This proposed method achieved 72% classification accuracy when trained and evaluated on the RAVDESS speech corpus [16]. In [17], an artificial intelligence-assisted deep stride CNN architecture using the plain nets strategy to learn salient and discriminative features from spectrograms was proposed. The proposed technique exhibited a classification accuracy of 80% when trained and evaluated on the RAVDESS speech corpus [17]. The authors of [18] proposed

a gated Residual Networks (GResNets) model and achieved a classification accuracy of 64.48% on the RAVDESS speech corpus using spectrogram images.

The audio recordings of the Crowd-source Emotional Multimodal Actors Dataset (CREMA-D) speech corpus is considered by many to be slightly lower quality than that of RAVDESS, as such there is not a well-defined technical evaluation baseline for the CREMA-D corpus. A multimodal temporal deep network framework embedding video clips using their audio-visual content onto a metric space where their gap is reduced and their complementary and supplementary information is explored was proposed in [12]. This multimodal approach achieved 74% and 67.7% classification accuracy on the CREMA-D and RAVDESS corpora, respectively. In [13], a radial basis function based support vector machine and *metric learning* was utilized to achieve a classification accuracy of 58.2% when trained and evaluated on the CREMA-D speech corpus. Metric learning approaches "learn distances to bring similar inputs closer and dissimilar ones further apart, which are more discriminative than the conventional Euclidean distance. This transformation is done through convex optimization with pairwise constraints" [13]. Some examples include Large Margin Nearest Neighbour and Geometric Mean Metric Learning. By combining the audio and visual modalities, the researchers of [13] improved classification accuracy to 63.6% for the CREMA-D corpus. The research performed in [13] also showed that the classification of emotions such as fear, disgust, and surprise benefit from the combination of the audio and visual modalities. Conversely, it was also shown that the anger emotion does not benefit from the addition of the visual modality [13].

Image enhancement techniques have also been shown to potentially yield a positive impact on the performance of CNNs and can be viewed as a form of data augmentation. Image

enhancement serves to construct invariant features that may be recognized as prominent features by a CNN during training [19]. CNNs have proven to be sensitive to small variations in image data such as contrast [20]. The work in [20] shows that image enhancement can provide large generalization gains. The authors of [20] found that adjusting contrast and brightness in combination with affine transformations yielded classification accuracy performance gains of approximately eight percent on the ImageNet dataset. Combining a variety of image enhancement techniques has also been shown to be a viable option based on the domain of the image classification task. Within [21] a number of techniques are combined to aid in classifying weakly illuminated images. The authors of [21] utilize a combination of contrast limited adaptive histogram equalization to lighten brightness and enhance contrast as well as log correction to further adjust image brightness. In addition, bright channel enhancement was also used to improve the vividness of an image [21]. The method proposed in [21] is shown to yield improved results over comparable state-of-the-art techniques.

Data augmentation techniques have also proven to be a valuable asset when deployed on raw audio data. Data augmentation assumes that more useful features can be extracted from the original data through perturbation and combination [22]. The work performed in [22] shows that raw audio data augmentation techniques such as adding Gaussian noise, time stretching, and pitch shifting can yield performance improvements when used in conjunction with spectrograms. These augmentation techniques serve to combat overfitting and improve the generalizability of deep neural networks.

Transfer learning has also proven to be an incredibly powerful technique with respect to deep learning. Transfer learning is based on the concept that a neural network trained on a

different domain, for a different source task may be adapted for a new domain and target task [23]. The authors of [24] show that transfer learning has a number of applications when used for other forms of sentiment analysis such as natural language processing.

In this thesis, five transfer learning models pre-trained on the ImageNet dataset were used for speech emotion recognition via the classification of spectrograms. A modified version of the AlexNet deep neural network architecture was utilized as a baseline for non-pre-trained deep neural networks. Two speech corpora were utilized to develop these methods. The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS) and the Crowd-source Emotional Multimodal Actors Dataset (CREMA-D). Data augmentation techniques were applied to the raw audio of the speech corpora to increase the amount of training data, yielding custom training datasets. Image enhancement techniques were implemented with the aim of improving classification accuracy by unveiling more prominent, invariant features in the spectrograms. Ensemble learning methods were also deployed in an attempt to further classification performance. In all, 180 experiments were conducted with the objective of discovering new methodologies yielding improved or very similar performance to what is currently considered to be state-of-the-art.

II. AUDIO FEATURES

Recording Audio

Sound is generated by the vibration of an object, which causes local air molecules to oscillate and produce sound waves. Sound waves are mechanical in nature and require a medium to transfer energy from one point to another. With regards to sound waves, this medium is air. In nature, sound waves are analog, continuous-time signals. Therefore, they must be converted into a digital, discrete-time signal to record and store sound. This conversion is achieved by sampling the audio amplitudes at discrete points in time. The number of audio samples taken per second is defined as the *sampling rate*. Figure 2 shows the results different sampling rates may have on sound waves.

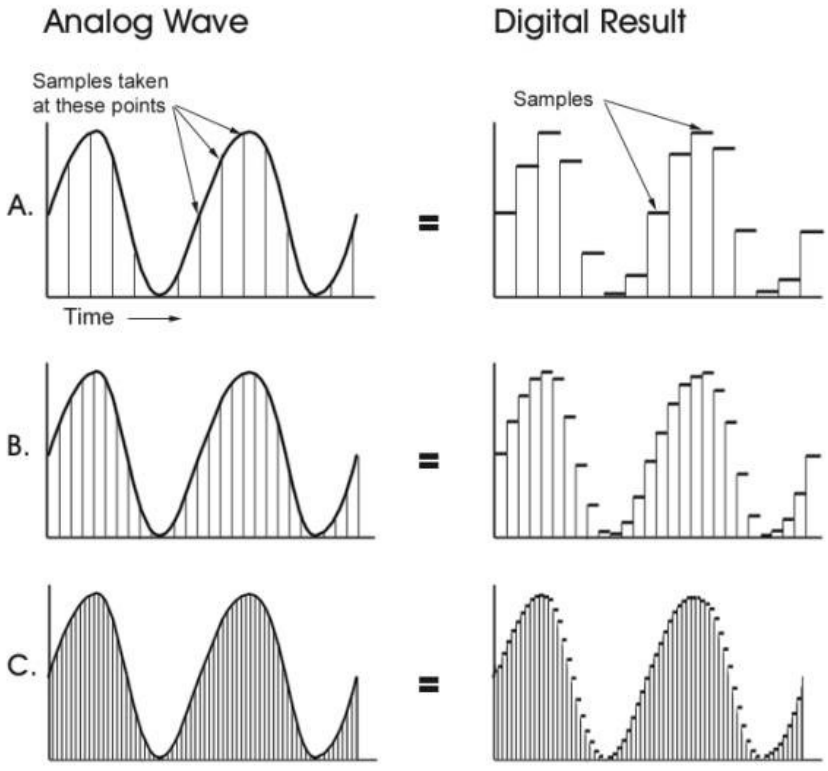

Figure 2. Effect of Increasing Sampling Rate of an Audio Signal [25].

While increasing the sampling rate allows for a better approximation of the actual analog sound signal, it also increases the amount of data that is being recorded. Therefore, it is important to strike a balance between sampling rate and data consumption. A sampling rate of sixteen thousand samples per second (kHz) is typically used within most applications of audio signal processing. Analog audio signals may also exhibit an infinite number of possible values with respect to amplitude. It is necessary for the amplitude to be discretized when converting an analog signal to a digital signal. The *bit depth* defines the number of possible amplitude values for a single sample of a discrete-time audio signal. Similar to the sampling rate, a higher bit depth yields a higher resolution of the discrete-time audio signal. Most standard audio recordings today are sixteen bit audio, with 2^{16} = 65,536 possible amplitude values. Figure 3 displays an analog signal sample at a bit depth of four bits per sample, which yields a total of 2^4 or sixteen possible amplitude values for each audio sample.

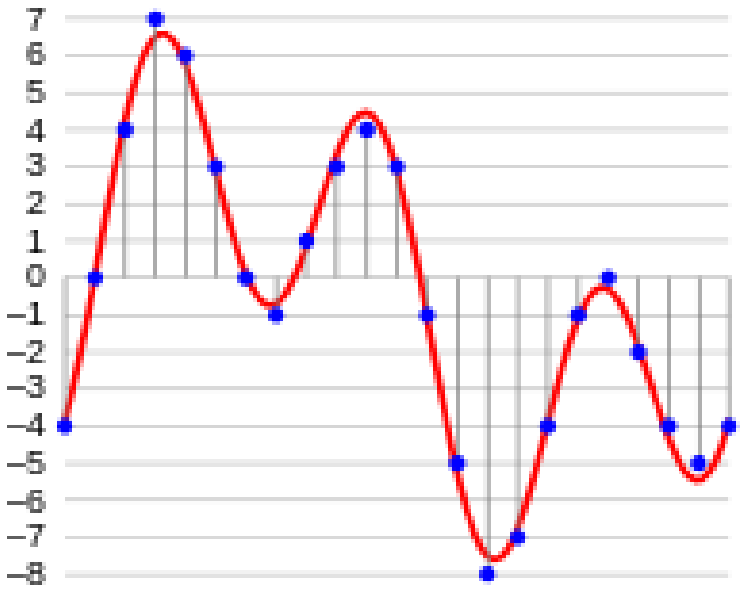

Figure 3. Analog Signal Being Sampled at a Bit Depth of Four Bits per Sample [26].

Audio Preprocessing

A speech signal is the result of a dynamic process and in turn generates dynamic data. This results in the statistical properties of the data, being that of the mean amplitude, standard deviation, and other metrics, changing over time. In order for a Fast Fourier Transform (FFT) to be applied for spectral analysis the speech signal is divided into multiple overlapping audio segments called *frames* to make the data statistically stationary [27]. Typically, audio frames are approximately between twenty and forty milliseconds in length [27].

When a FFT is applied to a signal, it is assumed that the signal is periodic in nature. Speech signals inherently abide by this assumption and since they do not drop to an amplitude of zero at the end of each audio frame, the FFT will generate high-frequency artifacts at these locations of the frame. To avoid this issue, a window function is applied to the audio frames through a technique referred to as *windowing* [27]. A window function is a mathematical function which has an amplitude of zero outside some defined interval. If and when a window function is multiplied with the speech segment in an audio frame, the resulting speech segment will exhibit an amplitude of zero outside the interval defined within the window function [27]. This serves to smoothen the edges of the signal in each audio frame. Therefore, the overlapping regions in audio frames, which are approximately ten milliseconds in length, ensure none of the audio segment is lost during preprocessing [27]. Figure 4 provides a visualization of framing and windowing a speech signal.

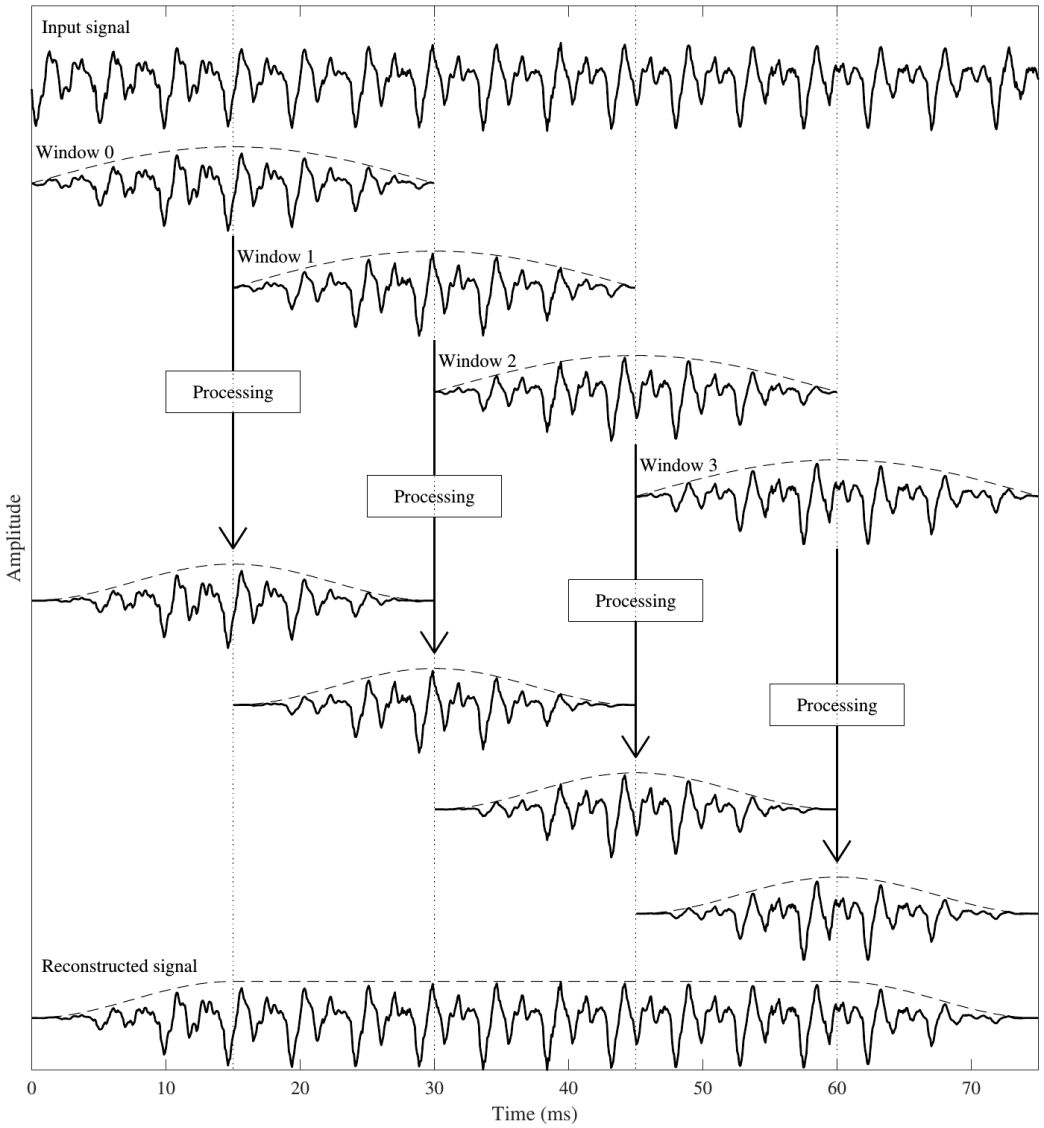

Figure 4. Framing and Windowing of an Audio Signal [28].

Features in Speech Emotion Recognition

Within the discipline of machine learning, features are defined as properties of data that assist a machine in learning to differentiate between data classes. In any audio classification task, whether it be SER, speaker recognition, automatic speech recognition or mental stress detection, audio features are an essential component to developing a solution. In SER however, "identification of the 'best' or the most characteristic acoustic features that characterize different emotions has been one of the most important but also the most elusive challenges" [6]. With this challenge still persisting, some researchers have transitioned to transforming SER into a hybrid image classification task rather than a strictly audio classification task as:

> Given the success of Deep Neural Networks architecture's design to classify 2-dimensional arrays, classification of speech emotions followed the trend, and several studies investigated the possibility of using spectral magnitude arrays known as speech spectrograms to classify emotions. Spectrograms provide 2-dimensional image-like time-frequency representations of 1-dimensional speech waveforms. Although the calculation of spectrograms does not fully adhere to the concept of the end-to-end network, as it allows for an additional preprocessing step (speech-to-spectrogram) before the DNN model, the processing is minimal and most importantly, it preserves the signal's entirety. [6]

Furthermore, a number of research projects have shown that log-scaled mel spectrograms offer better performance as the input to a Convolutional Neural Network (CNN) [22].

This method is advantageous compared to traditional machine learning techniques which require the calculation of feature parameters from raw data [6]. Generally, "it is not known which features can lead to the most efficient clustering of data into different categories (or classes)" [6]. This requires the use of feature selection methods in which the "quality of the resulting hand-crafted features can have a significant effect on classification performance" [6]. Deep Neural Networks (DNN) bypass this issue as "there is no need to compute hand-crafted features,

nor to determine which parameters are optimal from the classification perspective" [6]. These processes are handled by the DNN itself. This solution does come at a cost, however. Typically, DNNs require much larger requirements with respect to labeled training data-samples than conventional machine learning techniques [6]. As previously mentioned, the availability of labeled training data for use in SER is already limited. Still though, DNNs have been shown to outperform many conventional approaches in a variety of classification tasks, of which the classification of images has been particularly successful [29].

Generating Log-mel Spectrograms

The first step in generating a log-mel spectrogram is to calculate *the mel-frequency cepstral coefficients* (MFCCs), first introduced in [30]. To calculate the MFCCs for a given audio signal, the first step is to frame the given signal into small overlapping audio frames of twenty-five to forty milliseconds [30]. With a sampling rate of sixteen kHz and a frame length of thirty-two milliseconds, this yields a total of $16 \times 32 = 512$ audio samples per frame [30]. If the frame step size (hop length) is sixteen milliseconds, it yields $16 \times 16 = 256$ audio samples in the overlapping regions [30].

Following this process, periodogram estimates of the power spectrum must then be calculated for each audio frame [30]. The spectrum's powers are mapped onto the *mel-scale* using a mel-filterbank that contains a set of twenty to forty triangular overlapping filters, as displayed in figure 5 [30]. These filters are spaced according to the mel-scale and given filterbank energies when applied to the power spectrum.

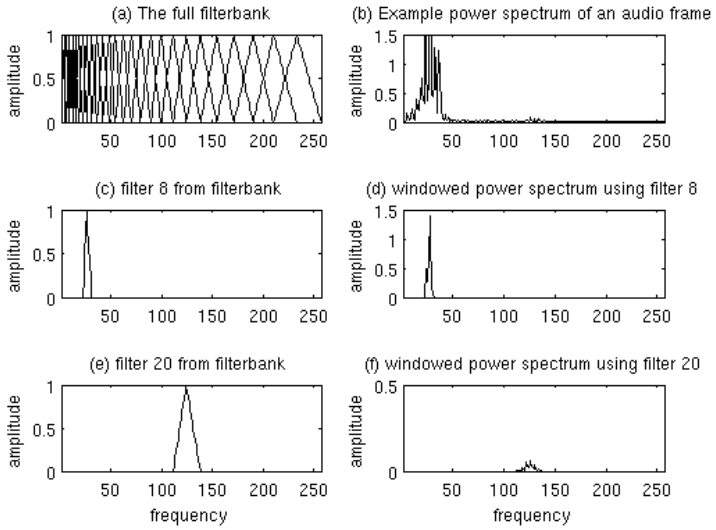

Figure 5. Applying a Mel-filterbank to the Power Spectrum of an Audio Frame [31].

After obtaining the filterbank energies, the logarithmic function is applied to them. The logarithmic function is applied in this step as human beings do not hear loudness on a linear scale. The logarithmic operation therefore serves to compress the features so that they match more closely to what humans actually hear. The final step is to perform a discrete cosine transform (DCT) on the log mel-filterbank energies, which in turn yields the MFCCs. The MFCCs are then mapped as a spectrogram in which the MFCCs make up the y-axis values and the x-axis values serve as a representation of the time-series of the audio signal. Figure 6 displays an example of a log-scaled mel spectrogram.

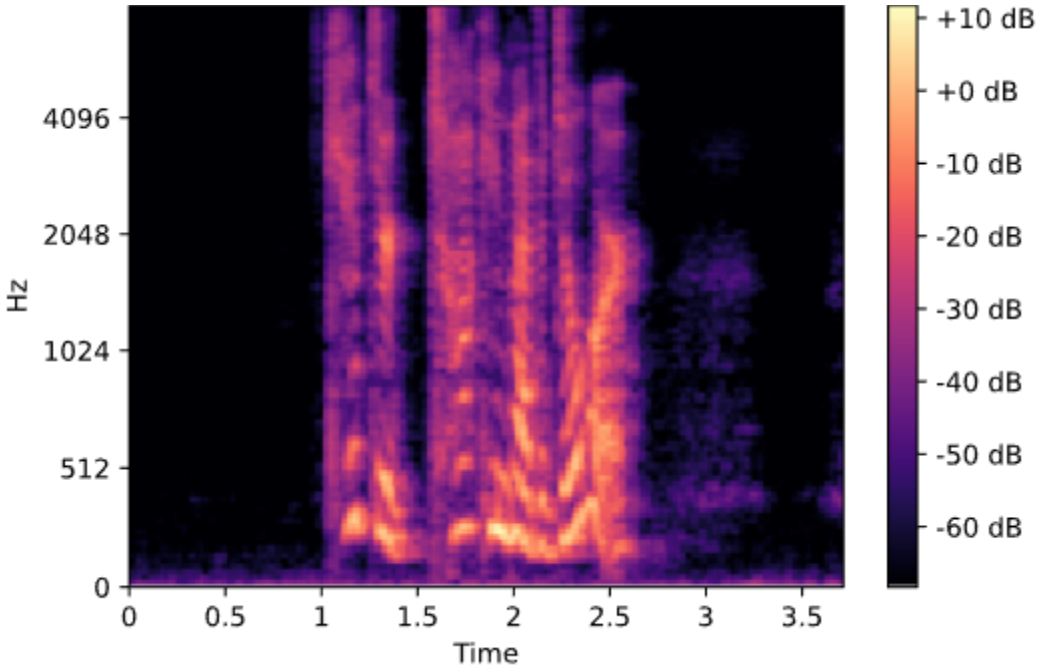

Figure 6. Log-scaled Mel Spectrogram.

III. DATASETS

For the work performed in this thesis, two speech corpora were selected. The first speech corpus is formally known as the Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS). This particular dataset is available online to be used by researchers at no cost. RAVDESS was designed and created specifically to be utilized for SER and has been evaluated and validated by a number of fellow researchers. The second speech corpus utilized was the Crow-sourced Emotional Multimodal Actors Dataset (CREMA-D). There exist two main types of speech datasets within the field of SER. The first contains recordings of people who express genuine emotions by being subjected to external stimuli such as image, video or audio. The second type contains recordings of professional actors reading script outlines while acting out the desired emotion. The former is formally known as a spontaneous dataset, the latter is referred to as a simulated dataset. The RAVDESS and CREMA-D corpora are both simulated datasets. Researchers are keen to use simulated datasets for SER tasks as they maintain accurate labels and actors are inherently good at expressing particular emotions with decent accuracy. The RAVDESS speech corpus was initially used to develop and fine-tune the machine learning methodology. In turn, the CREMA-D speech corpus was used to determine the robustness of the method and its generalizability.

Ryerson Audio-Visual Database of Emotional Speech and Song

The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS) was released in 2018 by researchers of the SMART lab at Ryerson University located in Toronto, Ontario, Canada [32]. It is a simulated, multimodal dataset that contains both video and audio data [32]. For the audio data, the actors recorded sentences both as normal speech and songs. The data pertaining to songs was not considered for this thesis as the work is aimed at SER. Each actor recorded two lexically matched sentences in eight different emotions. The two sentences are “Kids talking by the door” and “Dogs are sitting by the door” [32]. The eight acted emotions include neutral, calm, happy, sad, anger, fear, disgust and surprise [32]. Of these eight emotions, seven of them were recorded twice per sentence. One recording exhibits a normal intensity, whereas the second recording exhibits a stronger intensity [32]. There was only one recording per sentence for the neutral emotion since there is no strong intensity for this emotion [32]. The audio files were recorded at sixteen bits per sample, at a sampling rate of forty-eight kHz, and in the WAV audio format [32].

A total of twenty-four actors in the age range of twenty-one to thirty-three years old took part in forming this speech corpus, where half of the samples are male actors and the remaining half consists of female actors [32]. This yields a total of 1,440 recording samples given 24 actors $\times$ 2 sentences $\times$ 8 emotions $\times$ 2 emotional intensities (with the exception of the neutral emotion) [32]. Therefore, the neutral class consists of 96 data samples, the remaining seven emotional classes consist of 192 data samples each. The RAVDESS speech corpus is available at [33].

Crowd-sourced Emotional Multimodal Actors Dataset

The Crowd-sourced Emotional Multimodal Actors Dataset (CREMA-D) was released in 2014 as a collaborative project amongst researchers from Ursinus College, the University of Pennsylvania and the University of Illinois Chicago. The CREMA-D speech corpus is composed of twelve different emotionally neutral sentences in six different emotions including anger, disgust, fear, happy, neutral and sad at four different intensity levels. Examples of the emotionally neutral sentences include “Don’t forget a jacket,” “I think I have a doctor’s appointment” and “I think I’ve seen this before” amongst others [34].

Each emotion class has 1,271 audio samples, save for the neutral class which has 1,087 audio samples. The actors whose recordings form the speech corpus range in age from twenty to seventy-four. There were forty-eight male actors and forty-three female actors, for a total of ninety-one. The audio files were recorded at sixteen bits per sample at a sampling rate of sixteen kHz, also in the WAV audio format. The CREMA-D dataset is available at [34]. Table 1 provides a summary of the speech corpora used throughout this research.

Corpus	Age of Participants	No. of Sentences	No. of Participants	Emotions	Samples per Class	Total Samples
RAVDESS	21-33 Years Old	2 (two repetitions and intensities)	24 (12 male, 12 female)	8	192 (96 for neutral)	1,440
CREMA-D	20-74 Years Old	12 (four intensities)	91 (48 male, 43 female)	6	1,271 (1,087 for neutral)	7,442

Table 1. Summary of Speech Corpora.

Data Augmentation

The speech recordings that compose both the RAVDESS and CREMA-D speech corpora were performed in the absence of background noise, in a "noiseless" environment [32]. This of course does not accurately reflect the real world in which conversations in a "noiseless" environment are incredibly rare as there is always some form of environmental noise around people during a conversation. The only noise audible in the recording of the RAVDESS and CREMA-D corpora is a combination of static noise generated by the recording equipment and echoes from the surrounding environment [32]. As such, it is important to account for the lack of background noise and introduce some noise to the audio samples to improve the real-world generalizability of the final SER model.

With the relatively small amount of data available from the RAVDESS corpus, 1,440 samples total, a deep neural network (DNN) of any type would be prone to severe overfitting. Overfitting occurs when a machine learning model fits the training data too closely, resulting in poor generalizability to new, previously unseen data. Therefore, it is necessary to deploy various data augmentation techniques to not only introduce noise to the audio recordings to improve real-world generalizability, but also to increase the amount of training data available. Data augmentation is therefore largely considered to be a regularization method with respect to machine learning [22].

Data augmentation has been proven to be "very helpful for the improvement of audio classification performance, especially directly used for the spectrogram" [22]. Data augmentation assumes that more useful features can be extracted from the original data through perturbation and combination [22]. There exist two main types of data augmentation techniques:

> One is to perturb the samples to generate new samples, and then add the new ones to the original dataset to explicitly expand the dataset. Other methods mix, cut and combine the samples, which does not substantially increase the number of samples. [22]

Within the work of this thesis project, the former is utilized extensively. The data augmentation techniques utilized for this work, described in the next section, serve to maintain the invariance of the original distribution of data from both the RAVDESS and CREMA-D speech corpora [22]. Figure 7 displays the waveplot and corresponding log-scaled mel spectrogram of a male actor acting a neutral emotion from the RAVDESS dataset.

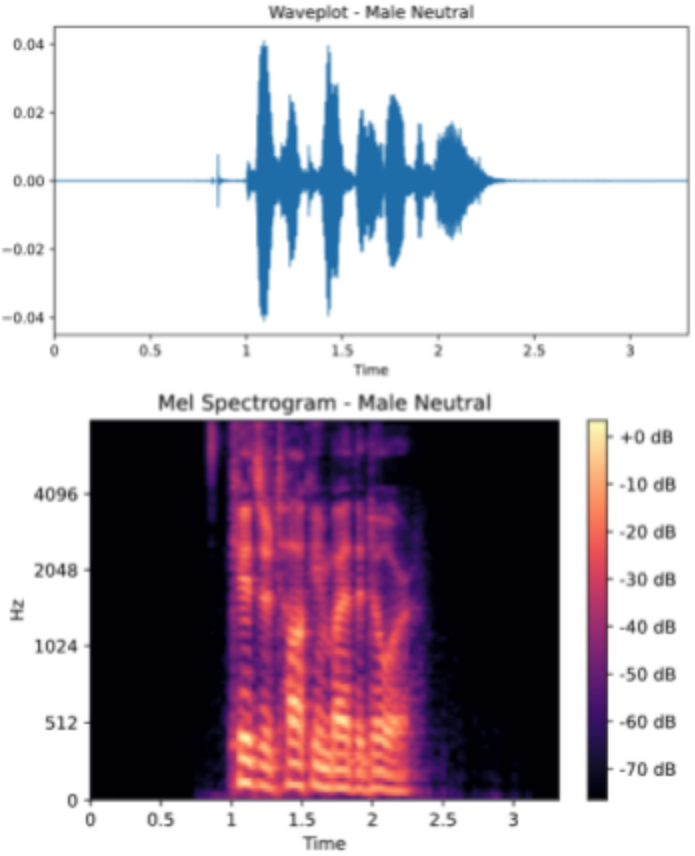

Figure 7. Waveplot and Spectrogram of Unaugmented Male Neutral Audio Signal.

Methods of Data Augmentation for Raw Audio

Noise Addition

Adding Gaussian noise to the audio samples serves to make the input space smoother and easier to learn [22]. The mean of Gaussian noise $n(t)$ is 0 and the variance is 1 [22]. This allows for the utilization of a pseudo-random number generator. Noise amplitude σ is an important component of noise. It serves as a hyper-parameter that can be configured. With a smaller noise amplitude, it is difficult to disturb, the larger the noise amplitude, the more difficulty the classifier experiences in learning [22]. To obey uniform distribution, an accepted range for noise amplitude has been defined as: $\sigma \in [0.001, 0.015]$ [22]. The newly generated sample after the addition of Gaussian noise can be expressed through equation 1:

$$x(t) = x(t) + \sigma \times n(t), \textit{ where } x(t) \textit{ is the raw signal} \qquad (1)\ [22]$$

This augmentation was applied to every audio sample within the set of training data from both the RAVDESS and CREMA-D speech corpora. Figure 8 displays the effect of the noise addition augmentation on the raw signal displayed in figure 7.

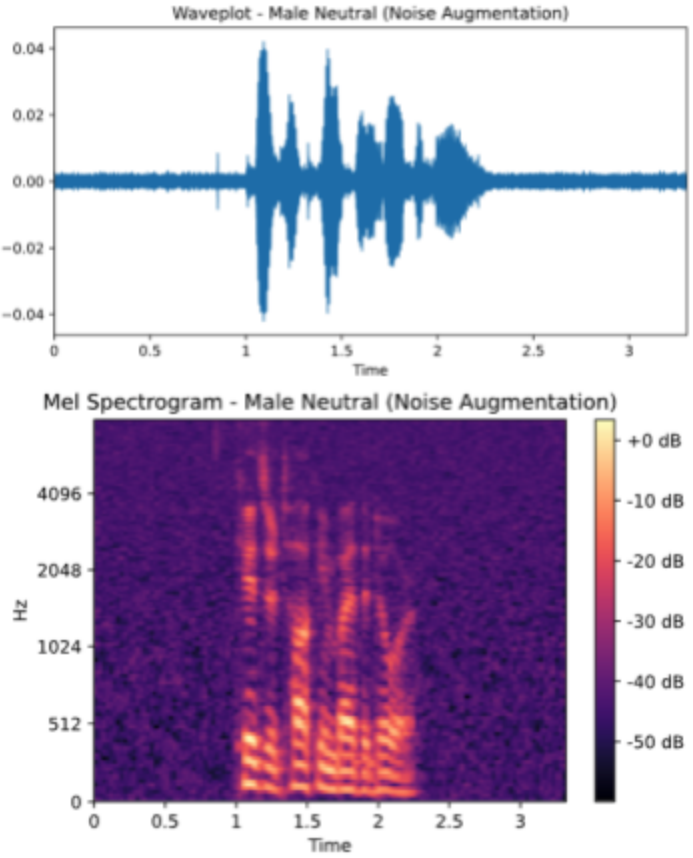

Figure 8. Results of Noise Augmentation on Raw Audio Signal.

Stretching of Audio

Stretching an audio signal changes the tempo and length of the audio clip without changing its pitch, where γ is the stretch factor [22]. To obey uniform distribution, the acceptable range for the stretch factor is defined as: $\gamma \in [0.8, 1.25]$ [22]. When γ is less than one, the signal is slowed down. Conversely, when γ is greater than one, the signal is sped up [22]. For the augmented signal to fit the input size of the neural network, it is necessary to maintain the original length for the stretched signal. This is accomplished through zero padding and cropping if the augmented signal is not long enough, or too long, respectively. Within the work of this thesis, stretch augmentations with a stretch factor of 0.8 and 1.2 were applied to every audio sample within the set of training data for both the RAVDESS and CREMA-D speech

corpora. This was accomplished through the use of the *time_stretch* function in the open source librosa toolkit [35]. Figure 9 displays the effect of stretching the audio signal by a factor of 0.8 for the audio sample from figure 7. Figure 10 displays the effect of stretching the audio signal by a factor of 1.2 for the audio sample from figure 7.

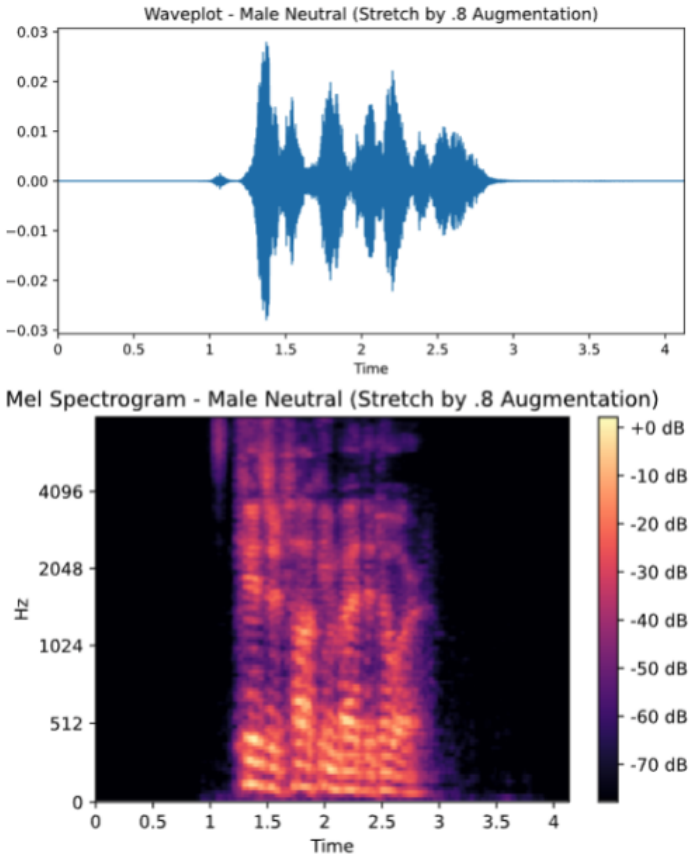

Figure 9. Results of Stretch Augmentation by Factor of 0.8.

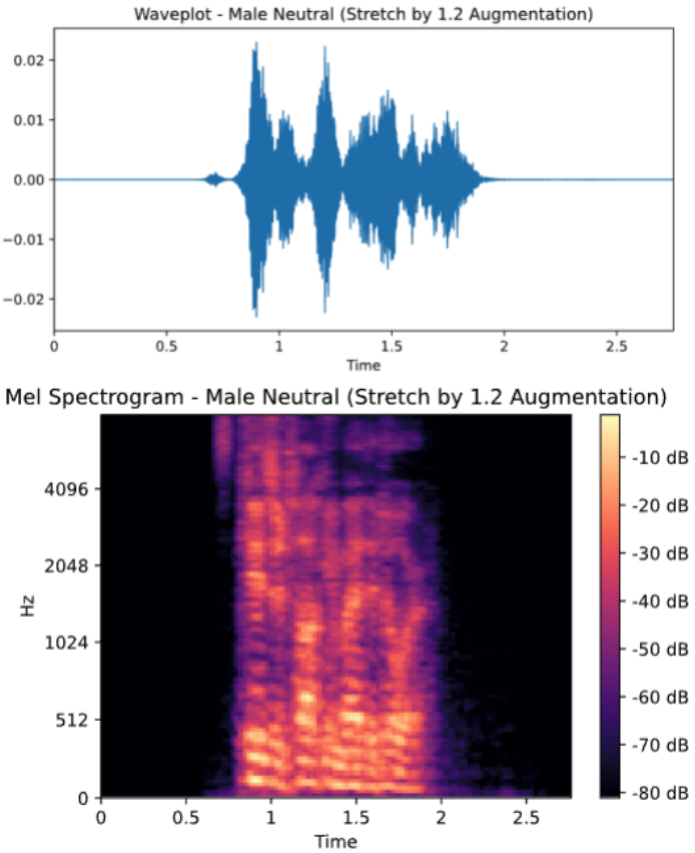

Figure 10. Results of Stretch Augmentation by Factor of 1.2.

Time Shifting of Audio

Time shifting an audio signal simply shifts the analog signal either forwards or backwards within the length of the given recording. This has no physical effect on the actual sounds but will still provide more training data. Any data that rolls beyond the last position of the recording is re-introduced at the front. This augmentation was applied to both training datasets of the RAVDESS and CREMA-D speech corpora. Figure 11 displays the effect of shifting the audio signal previously shown in figure 7.

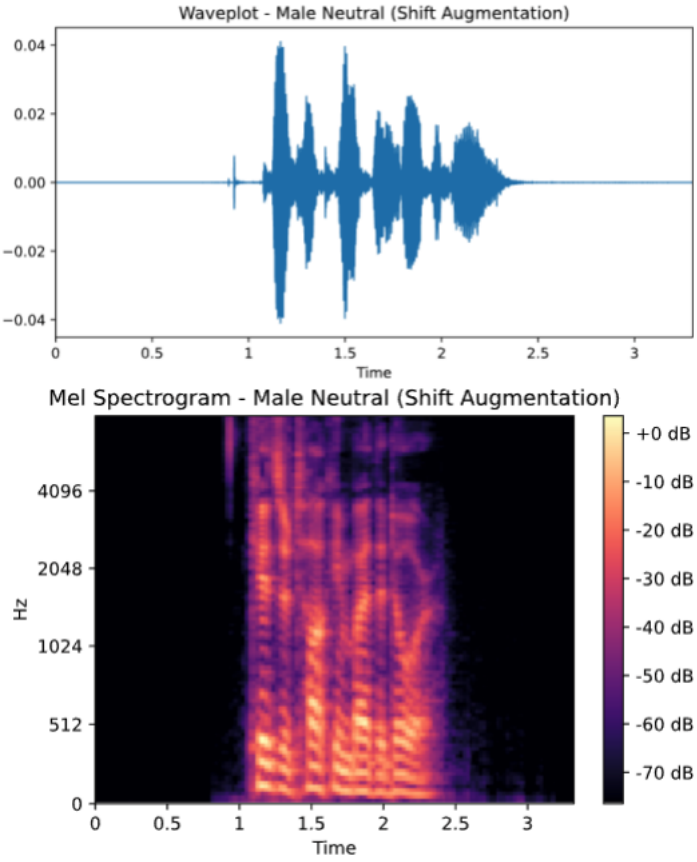

Figure 11. Results of Time Shift Augmentation.

Shifting of Pitch

The *pitch* of a sound is the perceived fundamental frequency F_0, the frequency at which vocal cords vibrate in voiced sounds [36]. Shifting the pitch of an audio sample is the reciprocal process of the aforementioned stretch augmentation [22]. A shift pitch serves to change the pitch of an audio signal by *n_steps* semitones without changing the tempo where *n_steps* is the number of fractional steps to shift [22]. To obey uniform distribution, an acceptable range for *n_steps* is defined as: $n\_steps \in [-4, 4]$ [22]. For the work in this thesis, three separate pitch augmentations were applied. First, the pitch was increased by a major third for each audio sample in the set of training data for both the RAVDESS and CREMA-D speech corpora. This

augmentation utilizes an *n_steps* value of (4). Figure 12 displays the result of this particular augmentation.

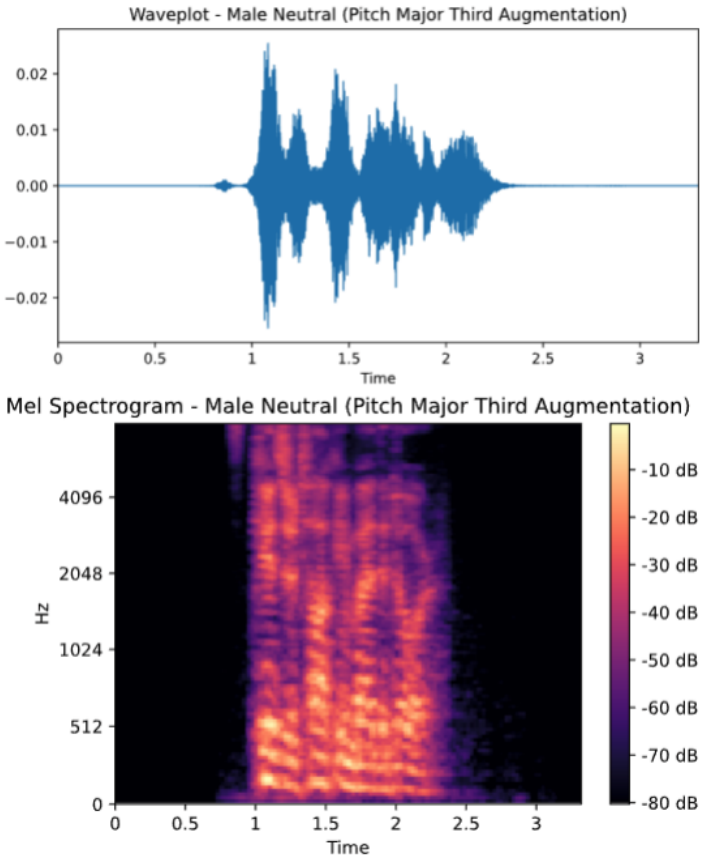

Figure 12. Results of Major Third Pitch Increase.

Next, the pitch of each audio sample in the set of training data was increased by three quarter-tones. This augmentation utilizes an *n_steps* value of (3). Figure 13 displays the result of this particular augmentation.

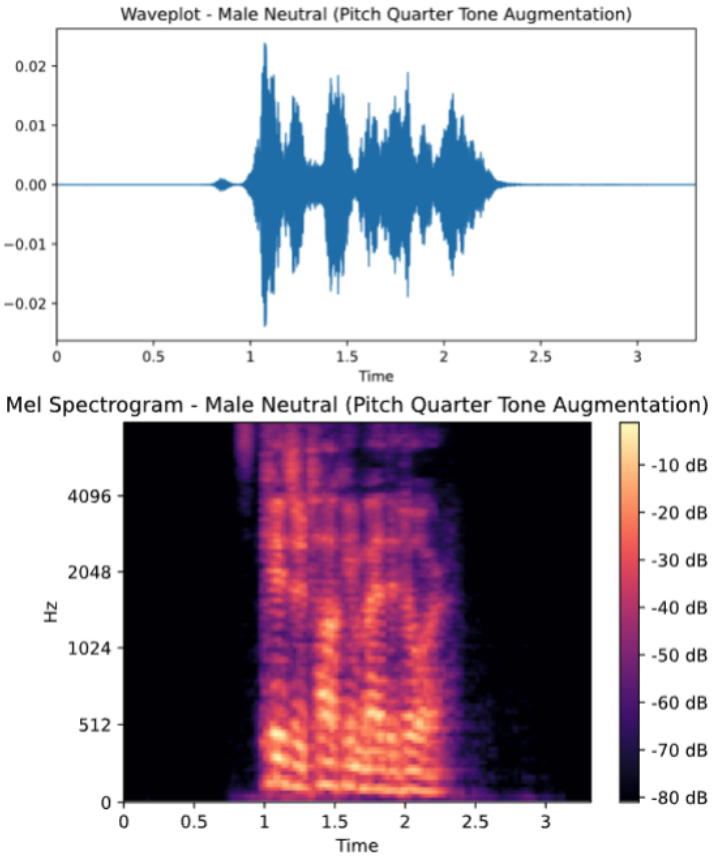

Figure 13. Results of Three Quarter-Tone Pitch Increase.

Finally, the pitch of each audio sample in the set of training data was decreased by a tritone. This augmentation utilizes an *n_steps* value of (-6). Initially, this augmentation violates uniform distribution, the number of steps per an octave was then configured to abide by uniform distribution, effectively yielding an *n_steps* value of (-4). Figure 14 displays the result of this particular augmentation.

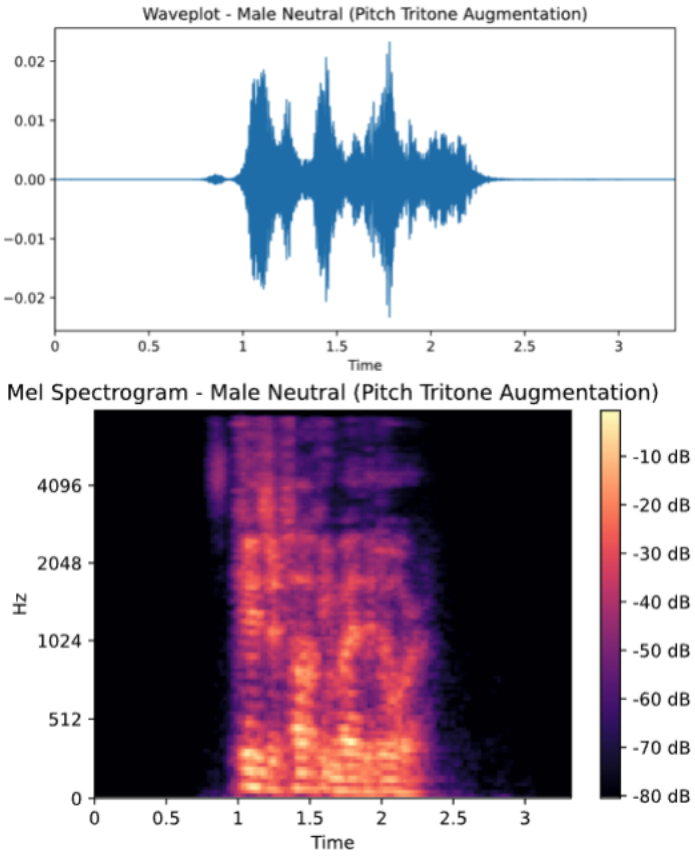

Figure 14. Results of Tritone Pitch Decrease.

These augmentations were accomplished through the use of the *pitch_shift* function in the open source librosa toolkit [35].

IV. MACHINE LEARNING METHODS AND ALGORITHMS

Machine learning is the science of and art of programming computers so that they can learn from data [37]. From an engineering perspective, the concept of machine learning is formally defined as:

> A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E. [37]

Machine learning therefore is encompassed by the broader field that is artificial intelligence. In *supervised* machine learning, a *learning* algorithm is provided with labeled data and learns the different data categories by matching the features to the labeled data samples. Upon completion of training on the data, it can make predictions on new, previously unseen data. *Deep learning* is a subset of machine learning that refers to the use of advanced neural network architectures with multiple layers of neurons. There exist two branches of supervised machine learning in classification and regression tasks.

Classification tasks are performed on discrete data whereas regression tasks are performed on continuous data. There exist numerous learning algorithms associated with supervised learning. Each of these algorithms are applied to different types of data. A Convolutional Neural Network (CNN) is designed to behave in a similar fashion to the human brain's visual cortex and is therefore solely used in image processing applications. In this work, in which audio signals have been transformed into spectrogram images, CNNs with transfer learning methods were primarily used to perform the Speech Emotion Recognition (SER) classification task.

Multilayer Perceptron

The Perceptron

Artificial neurons were inspired by the real biological neurons of the human body. Nerve cells, or neurons serve as the building blocks of the human nervous system. Figure 15 displays the body of a typical neuron. The dendrites collect electrical and chemical signals, known as synapses, from other neurons, which are combined at the nucleus. If the aggregate signal exceeds a threshold, the neurons fire a synapse through its axon to other neurons.

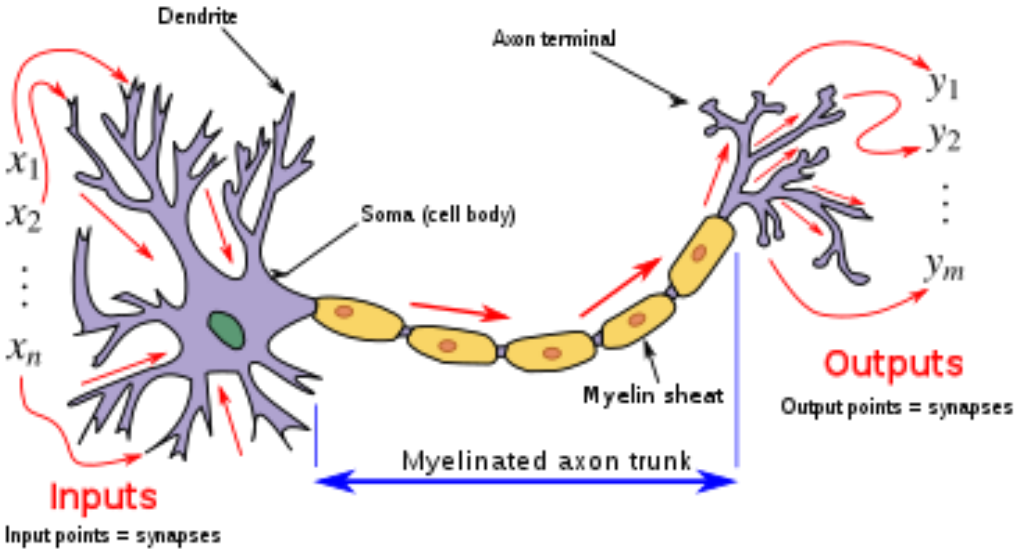

Figure 15. The Structure of a Biological Neuron [38].

Warren McCulloch and Walter Pitts developed the concept of the McCulloch-Pitts (MCP) neuron in 1943 [37]. The MCP neuron was described by them to be a simple logic gate with a binary output. In 1957, Frank Rosenblatt introduced the perceptron learning algorithm [37]. It is known as one of the simplest Artificial Neural Network (ANN) architectures [37].

The perceptron is based on a *threshold logic unit* (TLU). Within a TLU, the inputs and outputs are numbers rather than binary on/off values and each input connection is associated with a weight [37]. The TLU computes a weighted sum of its inputs shown in equation 2.

$$z = w_1 x_1 + w_2 x_2 + \ldots + w_n x_n = x^T w \qquad (2)$$

The TLU then applies a step function to that sum and outputs the result found in equation 3.

$$h_w(x) = step(z), \; where \; z = x^T w \qquad (3)$$

Figure 16 visualizes the TLU utilized in a perceptron.

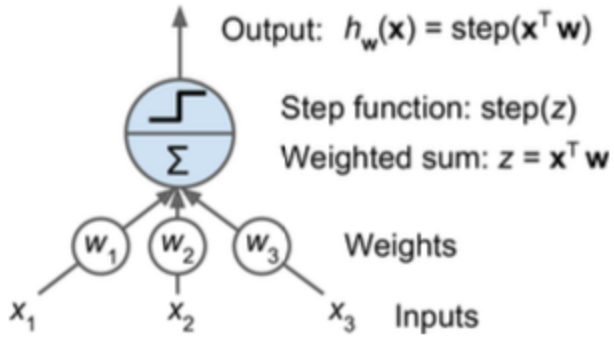

Figure 16. Threshold Logic Unit (TLU) [37].

A single TLU may be used for simple linear binary classification by computing a linear combination of the inputs and if the result exceeds a threshold, it outputs the positive class or else outputs the negative class [37].

A perceptron is simply composed of a single layer of TLUs in which each TLU is connected to all the inputs [37]. When all the neurons in a layer are connected to every neuron in the previous layer, it is referred to as a *fully connected layer* or a *dense layer* [37]. Typically, a *bias neuron* is added which simply outputs 1 all the time [37]. Figure 17 displays a perceptron with two inputs and three outputs, representing a multi-output classifier. Input neurons are simply passthrough neurons which output whatever input they are fed, together all the input neurons form the input layer.

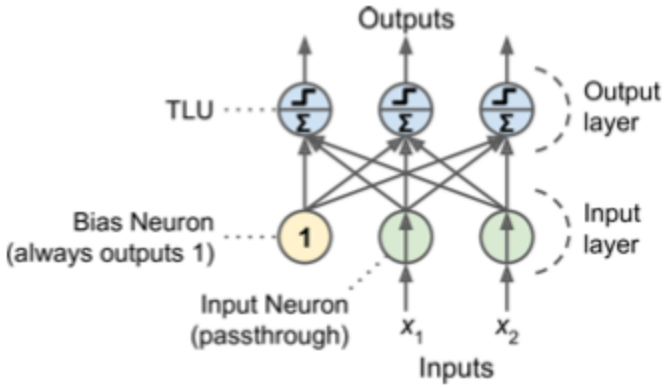

Figure 17. Perceptron Diagram [37].

It is possible to compute the outputs of a layer of artificial neurons for several instances at once through equation 4.

$$h_{W,b}(X) = \varphi(XW + b) \qquad (4)\ [37]$$

Where X represents the matrix of input features with one row per instance and one column per feature. W represents the weight matrix containing all the connection weights except for the ones from the bias neuron; it has one row per input neuron and one column per artificial neuron in the layer. b represents the bias vector which contains all the connection weights between the bias neuron and artificial neurons, it has one bias term per artificial neuron. Lastly, the function φ is the activation function, in the case of TLUs, a step function.

In 1949, Donald Hebb suggested that when a biological neuron often triggers another neuron, the connection between these two neurons grows stronger [37]. This is known as Hebb's rule. The training algorithm for perceptrons proposed by Frank Rosenblatt is a variant of Hebb's rule. The algorithm takes into account the error made by the network by reinforcing connections that help reduce the error [37]. The perceptron is fed one training instance at a time, and for each instance it generates its predictions [37]. For each output neuron that produced an incorrect

prediction, it reinforces the connection weights from the inputs that would have contributed to the correct prediction. Equation 5 demonstrates the perceptron learning rule.

$$w_{i,j}^{(next\ step)} = w_{i,j} + \eta(y_j - \hat{y}_j)x_i \qquad (5)\ [37]$$

In which, $w_{i,j}$ is the connection weight between the i^{th} input neuron and the j^{th} output neuron, x_i is the i^{th} input value of the current training instance, $\hat{y}_j$ is the output of the j^{th} output neuron for the current training instance, y_j is the target output of the j^{th} output neuron for the current training instance and η is the learning rate.

The decision boundary of each output neuron is linear, therefore perceptrons are incapable of learning complex patterns. If training instances are however linearly separable, Rosenblatt demonstrated that the perceptron training algorithm would converge to a solution. This is referred to as the *perceptron convergence theorem* [37].

There are a number of weaknesses and limitations associated with perceptrons. In particular, they are incapable of solving some trivial problems such as the exclusive or classification problem [37]. Such limitations however can be eliminated by stacking multiple perceptrons [37]. The resulting ANN is called a Multilayer Perceptron (MLP).

Feedforward Artificial Neural Networks

An MLP is a feedforward artificial neural network composed of one passthrough input layer, one or more layers of TLUs, referred to as hidden layers, and one final layer of TLUs, called the output layer [37]. Each layer except for the output layer includes a bias neuron and is fully connected to the next layer. Figure 18 displays the structure of a MLP.

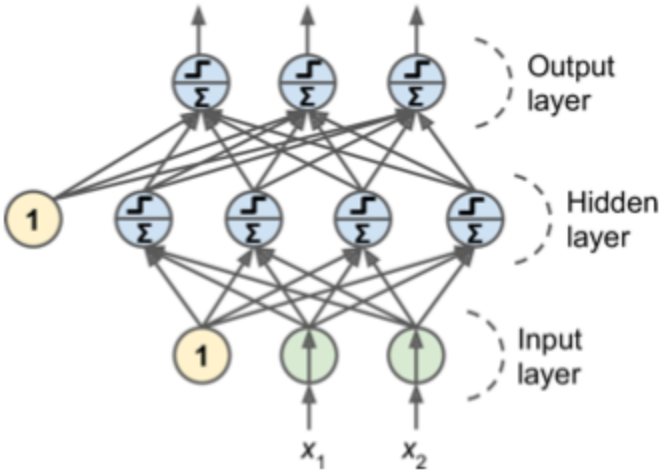

Figure 18. Multilayer Perceptron [37].

Given that the signal flows only in one direction from the inputs to the outputs, an MLP is clearly a feedforward artificial neural network. When an ANN contains a deep stack of hidden layers, it is referred to as a *deep neural network* (DNN). MLPs are trained via the *backpropagation* training algorithm.

Backpropagation Training Algorithm

First introduced in 1986 by David Rumelhart, Geoffrey Hinton and Ronald Williams, the backpropagation training algorithm is still used today [37]. In just two passes through the network (one forward, one backward) the algorithm is able to compute the gradient of the network's error with regards to every single model parameter. In simpler terms, the algorithm can determine how each connection weight and each bias term should be tweaked in order to reduce the error [37]. Once it has these gradients it simply performs a regular Gradient Descent step, repeating the process until the network converges to the solution [37]. For the algorithm to work properly, it was necessary to tweak the MLPs architecture by replacing the step function with the logistic function found in equation 6 to serve as the activation function.

$$\sigma(z) = 1/(1 + exp(-z)) \qquad (6)\ [37]$$

This modification was necessary as the step function contains only flat segments, therefore there are no gradients to work with, whereas the logistic function has a well-defined nonzero derivative everywhere, allowing Gradient Descent to progress some at every step [37].

The backpropagation training algorithm begins by handling one mini-batch at a time and it traverses the full training set multiple times. Each pass is referred to as an *epoch*. Each mini-batch is passed to the network's input layer, which passes it on to the first hidden layer. Then the outputs of all the neurons in this layer are computed for every instance in the mini-batch. The result is passed on to the next layer, its output is computed and passed to the next layer, and so on until the output of the last layer, the output layer, is reached [37]. This is the *forward pass* of the algorithm, it is exactly like making predictions, except the intermediate results are preserved since they are needed for the backwards pass. Next, the algorithm measures the network's output error through a loss function that compares the desired output and the actual output of the network, returning some measure of the error. The algorithm then efficiently computes how much each output connection contributed to the error by applying the *chain rule*. The *backwards pass* of the algorithm is now performed by measuring how much of these error contributions came from each connection in the layer below, again using the chain rule, and so on until the algorithm reaches the input layer. This pass efficiently measures the error gradient across all connection weights in the network by propagating the error gradient backward through the network [37]. Lastly, the algorithm performs a Gradient Descent step to tweak all the connection weights in the network, using the previously computed error gradients [37]. Today, the backpropagation algorithm is still used in ANNs, albeit with a variety of other activation functions, not just the logistic function.

Optimization

Gradient descent is a rather generic optimization technique capable of finding optimal solutions to a wide variety of problems. It is used broadly within machine learning and serves as the basis for other optimization techniques. Gradient descent measures the local gradient of a given error function with regards to the parameter vector θ, and it goes in the direction of a descending gradient. Once the gradient is zero, a minimum has been reached. To start, θ is filled with random values, referred to as *random initialization,* it is then improved gradually, one step at a time, each step attempting to decrease the cost function until the algorithm converges to a minimum [37]. Figure 19 visualizes gradient descent.

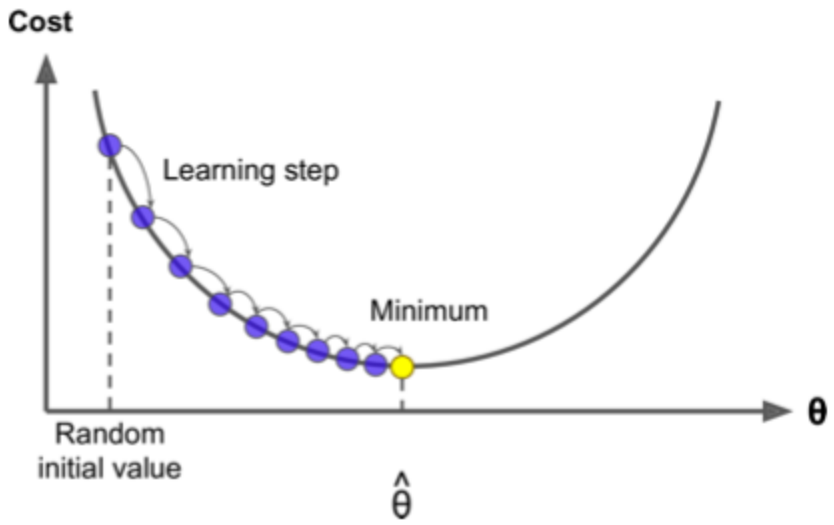

Figure 19. Gradient Descent [37].

Faster optimizers, which take inspiration from gradient descent, now exist that are capable of greatly improving training times with respect to deep neural networks. Examples include the *RMSProp* and *Adam* optimizers amongst others. Both RMSProp and Adam are adaptive learning rate methods of optimization. The RMSProp or *root mean square propagation* optimizer was first introduced by Geoffrey Hinton and Tijmen Tieleman in 2012 in [39].

RMSProp is unique in that it accumulates only the gradients from the most recent iterations by using exponential decay in the first step of the algorithm [39]. Equation 7 displays the RMSProp algorithm.

$$1.\ s \leftarrow \beta s + (1 - \beta)\ \nabla_\theta J(\theta) \otimes \nabla_\theta J(\theta)$$
$$2.\ \theta \leftarrow \theta - \eta\ \nabla_\theta J(\theta) \oslash \sqrt{s + \varepsilon} \qquad (7)\ [37]$$

Where β is the decay rate, typically set to 0.9, s is a vector accumulating the square of the gradients, $J(\theta)$ is the cost function, which is discussed in the following section. ε is a smoothing term that prevents division by zero and is typically initialized to 10^{-10} for RMSProp and η is the learning rate [37].

The Adam or *adaptive moment estimation* optimizer was first introduced in [40]. This optimizer combinees the ideology of momentum optimization and RMSProp. Thus, it keeps track of an exponentially decaying average of past gradients as well as an exponentially decaying average of past squared gradients. Equation 8 traverses the various steps of the Adam optimization algorithm.

$$1.\ m \leftarrow \beta_1 m - (1 - \beta_1)\ \nabla_\theta J(\theta)$$
$$2.\ s \leftarrow \beta_2 s + (1 - \beta_2)\ \nabla_\theta J(\theta) \otimes \nabla_\theta J(\theta)$$
$$3.\ \hat{m} \leftarrow \frac{m}{1 - \beta_1^t}$$
$$4.\ \hat{s} \leftarrow \frac{s}{1 - \beta_2^t}$$
$$5.\ \theta \leftarrow \theta + \eta \hat{m} \oslash \sqrt{\hat{s} + \varepsilon} \qquad (8)\ [37]$$

Where β_1 is the momentum decay hyperparameter, typically initialized to 0.9. β_2 is the scaling decay hyperparameter usually initialized to 0.999. m is the momentum vector initialized at 0. s is a vector accumulating the square of the gradients also initialized to 0. $J(\theta)$ is the cost function, which is discussed in the following section. ε is a smoothing term that prevents division by zero and is typically initialized to 10^{-7} for Adam. Lastly, η is the learning rate in

which 0.001 is generally utilized [37]. Adam is regarded to be computationally efficient and is well suited for problems with large amounts of data or parameters while minimizing memory requirements [40].

Cost Function

With respect to deep neural networks and multiclass classification tasks, categorical cross-entropy is typically used as the cost function. Categorical cross-entropy measures how well a set of estimated class probabilities match the target class. Equation 9 displays the cross-entropy cost function.

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} log(\widehat{p_k^{(i)}}) \qquad (9)\ [37]$$

Where $y_k^{(i)}$ is the target probability that the i^{th} instance belongs to class k. Generally, this is equivalent to 1 or 0, depending on whether the instance belongs to the class or not. The cross-entropy gradient vector for a class k may be calculated through equation 10.

$$\nabla_{\theta(k)} J(\theta) = \frac{1}{m}\sum_{i=1}^{m} (\widehat{p_k^{(i)}} - y_k^{(i)}) x^{(i)} \qquad (10)\ [37]$$

By computing the gradient vector for every class, in conjunction with an optimization algorithm, the parameter matrix θ that minimizes the cost function can be found.

Activation Function

The function $\varphi()$ denotes an activation function. Ideally, activation functions should be differentiable to optimally learn the weights using gradient descent. Examples of these activation functions include the sigmoid function, softmax function, tanh function and many others. The rectified linear unit (ReLU) function is a nonlinear function that is often used in the hidden layers of deep neural networks. Equation 11 displays the ReLU function.

$$ReLU(z) = max(0,\ z) \qquad (11)\ [37]$$

The ReLU function returns 0 if it receives any negative input, but for any positive value z it returns that value back. Thus, the ReLU function gives an output that has a range from 0 to infinity. Despite not being differentiable at $z = 0$, in practice the ReLU function works very well and is fast to compute [37].

For classification tasks, in the case of binary classification, the sigmoid function is typically used as the activation function for the output layer of a deep neural network. When working with a multiclass classification task, the softmax function is used for the output layer. The softmax function will ensure that all the estimated probabilities are between 0 and 1 and that they add up to one [37]. The softmax function can be found in equation 12.

$$\hat{p_k} = \sigma(s(x))_k = \frac{exp(s_k(x))}{\sum_{j=1}^{K} exp(s_j(x))} \qquad (12)\ [37]$$

In which K is the number of classes, $s(x)$ is a vector containing the scores of each class for the instance x and $\sigma(s(x))_k$ is the estimated probability that the instance x belongs to class k given the scores of each class for that instance. Figure 20 displays an MLP with ReLU used as the activation function for the hidden layers and an output layer utilizing the softmax function.

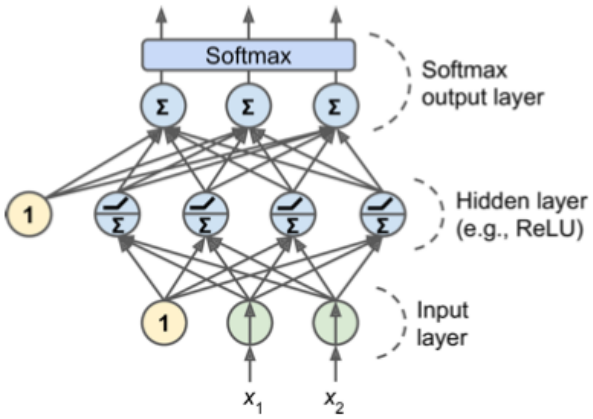

Figure 20. MLP with ReLU and Softmax Activation Functions [37].

Convolutional Neural Networks

Convolutional neural networks (CNN) are a biologically-inspired variation of the multilayer perceptron (MLP). Unlike MLPs in which each neuron has a separate weight vector, neurons in CNNs share weights. In turn, this sharing of weights results in reducing the overall number of trainable weights, introducing sparsity. Utilizing the weights sharing strategy, neurons are able to perform convolutions on the data with the convolution filter being formed by the weights, this forms the convolutional layers of the network. This is then followed by a pooling operation which, as a form of non-linear downsampling, progressively reduces the spatial size of the representation thus reducing the amount of computation and parameters in the network, forming a pooling layer.

Between the convolution and the pooling layer there exists an activation function, such as the rectified linear unit (ReLu) layer. The activation function is applied element-wise and after several convolutional and pooling layers, the feature map size is reduced and more complex features are extracted.

Eventually, with a small enough feature map, the contents are reduced into a one dimension vector and fed into a fully-connected MLP for processing. The last layer of this fully-connected MLP, seen as the output, is a loss layer which is used to specify how the network training penalizes the deviation between the predicted and true labels.

Convolutionional Layers

Convolutional layers are the fundamental building blocks of a CNN. Neurons in the first convolutional layer are not connected to every single pixel in the input image, but rather only to pixels in their receptive field. Each neuron in the second convolutional layer is connected only to

neurons located within a small rectangle in the first layer. This methodology can be seen in Figure 21. This allows the network to concentrate on small low-level features in the first hidden layer, then assemble them into larger higher-level features in the next hidden layer, and so on [37].

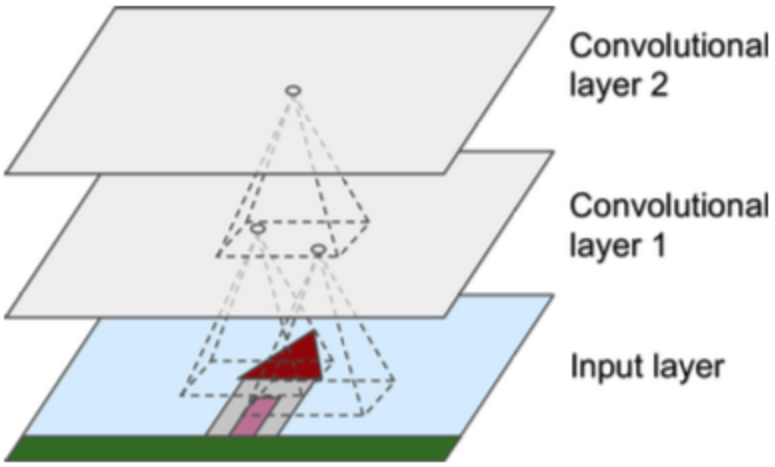

Figure 21. CNN Layers with Local Receptive Fields [37].

It is possible to connect a large input layer to a much smaller layer by spacing out the receptive fields; this shift from one receptive field to the next is called the *stride* [37]. Figure 22 shows a 5 x 7 input layer connected to a 3 x 4 layer, using 3 x 3 receptive fields and a stride of 2.

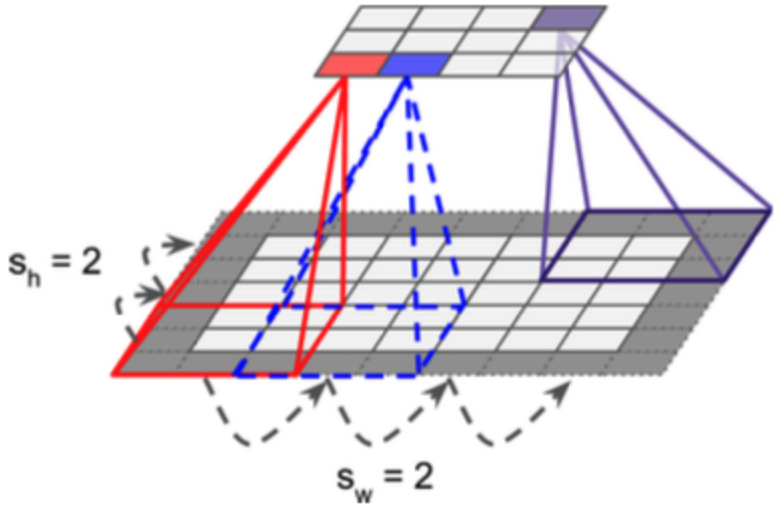

Figure 22. Dimensionality Reduction in CNN with a Stride of 2 [37].

A neuron located in row i, column j in the upper layer is connected to the outputs of the neurons in the previously layer located in rows $i \times s_h \; to \; i \times s_h + f_h - 1$, columns $j \times s_w \; to \; j \times s_w + f_w - 1$, where s_h and s_w are the vertical and horizontal strides, f_h and f_w are the height and width of the receptive field [37].

A neuron's weights can be represented as a small image the size of the receptive field, referred to as a *filter* or *convolution kernel* [37]. A layer full of neurons using the same filter outputs a *feature map*. A feature map highlights the areas in an image that activate the given filter the most. Convolutional layers automatically learn the most useful filters for its task, and the layers above will learn to combine the filters to combine them into progressively more complex patterns [37]. A convolutional layer typically has multiple filters, outputting one feature map per filter. To accomplish this there is one neuron per pixel in each feature map, and all neurons within a given feature map share the same parameters or weights. This mechanism drastically reduces the number of parameters in the model [37]. Thus, a convolutional layer simultaneously applies multiple trainable filters to its inputs, making it capable of detecting multiple features anywhere it is inputs [37]. Therefore, a neuron located in row i, column j of the feature map k in a given convolutional layer l is connected to the outputs of the neurons in the previously layer $l - 1$ located in rows $i \times s_h \; to \; i \times s_h + f_h - 1$, columns $j \times s_w \; to \; j \times s_w + f_w - 1$, across all feature maps in layer $l - 1$, where s_h and s_w are the vertical and horizontal strides, f_h and f_w are the height and width of the receptive field [37]. Equation 13 computes the output of a given neuron in a convolutional layer.

$$z_{i,j,k} = b_k + \sum_{u=0}^{f_h-1} \sum_{y=0}^{f_w-1} \sum_{k'=0}^{f_{n'}-1} x_{i',j',k'} \cdot w_{u,v,k',k} \; with:$$
$$i' = i \times s_h + u,$$
$$j' = j \times s_w + v \qquad (13)\ [37]$$

Where $z_{i,j,k}$ is the output of the neuron located in row i, column j in feature map k of the convolutional layer l, s_h and s_w are the vertical and horizontal strides, f_h and f_w are the height and width of the receptive field, and $f_{n'}$ is the number of features maps in the previous layer $l-1$, $x_{i',j',k'}$ is the output of the neuron located in layer $l-1$, row i' and column j', feature map k', b_k is the bias term for feature map k and $w_{u,v,k',k}$ is the connection weight between any neuron in feature map k of the layer l and its input located at row u, column v and feature map k' [37].

Pooling Layers

Pooling layers serve to subsample, or shrink, the input image in order to reduce the computational load, memory usage, and number of parameters, hence limiting risk of overfitting. In similar fashion to a convolutional layer, each neuron in a pooling layer is connected to the outputs of a limited number of neurons in the previous layer, located within a small receptive field [37]. A pooling neuron though, has no weights, it simply aggregates the inputs using an aggregation function such as the max or mean. *Max pooling layers* are the most common types of pooling layers. In such a layer, only the max input value in each receptive field makes it to the next layers, while the other remaining inputs are dropped. Inserting a max pooling layer every few layers in a CNN, it is possible to get some level of translation invariance with respect to rotation and scale at a larger scale [37]. In classification tasks this invariance can be useful as the prediction should not depend on these details.

In modern CNN architectures *global average pooling layers* are also widely used. All this layer does is compute the mean of each entire feature map, meaning that it just outputs a single number per feature map and per instance. Inherently, this is rather destructive as most of the information in the feature map is lost [37]. It can be useful to use such a layer as the output layer however in certain cases [37].

General Architecture

Typical architectures of CNNs stack a few convolutional layers, each one followed by a ReLU layer, then a pooling layer, then another few convolutional layers, with the accompanying ReLU layer, then another pooling layer and so on. As it progresses through the network the input image gets smaller and smaller, but also deeper and deeper with more feature maps. At the top of this stack, a regular feedforward ANN is added. This feedforward ANN is composed of a few fully connected layers, with the accompanying ReLU layer(s), and the final layer outputs the prediction. For a multiclass classification task this final layer uses the softmax activation function to output the estimated class probabilities. Figure 23 visualizes the architecture of a typical CNN.

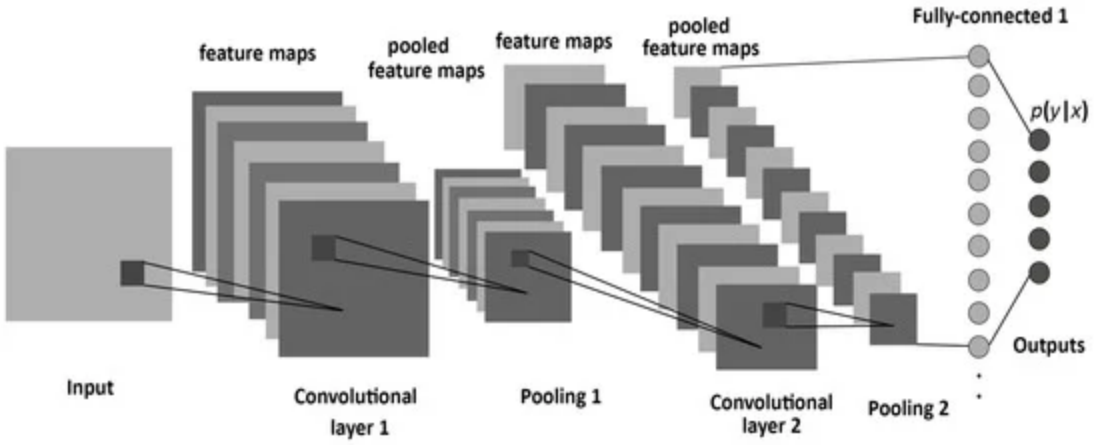

Figure 23. Typical CNN Architecture [41].

Regularization Techniques

Dropout is one of the most popular regularization techniques for deep neural networks. It was first proposed by Geoffrey Hinton in 2012 in [42]. Dropout is a fairly simple algorithm in which at every training step, every neuron, including the input neurons, but excluding the output neurons, is given a probability p of being temporarily "dropped out" [42]. Being dropped out means that the given neuron will be entirely ignored during this training step, but may be active during the next step. The parameter p refers to the *dropout rate.* This concept is founded upon the fact that a unique neural network is generated at each training step [42]. Since each neuron can be either present or absent, there are a total of 2^N possible networks, where N is the total number of droppable neurons. Neurons trained with dropout cannot co-adapt with their neighboring neurons. Thus, neurons must be as useful as possible on their own. This results in neurons that are less sensitive to slight changes in the inputs, yielding a more robust network that generalizes better [42]. Figure 24 visualizes the concept of dropout.

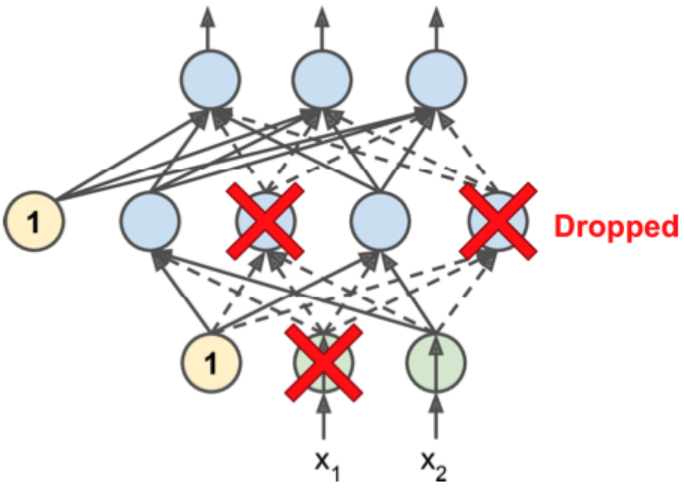

Figure 24. Dropout Regularization [37].

There is a small caveat to utilizing dropout in that if $p = 50\%$, during testing a neuron will be connected to twice as many input neurons as it was on average during training. To compensate for this, it is necessary to multiply each neuron's input connection weights by 0.5 after training [42]. If this is not performed, each neuron will get a total input signal roughly twice as large as what the network was trained on yielding poor performance [42].

Transfer Learning

With respect to machine learning, *transfer learning* is based on the concept that a neural network trained on a different domain, for a different source task may be adapted for a new domain and target task [23]. A formal definition of transfer learning is as follows:

> Given a learning task T_t based on domain D_t, we can get help from D_s for the learning task T_s. Transfer Learning aims to improve the performance of the predictive function $f_T(\cdot)$ for learning task T_t by the discovery and transfer of latent knowledge from D_s and T_s, where $D_s \neq D_t$ and/or $T_s \neq T_t$. In addition, in most cases, the size of D_s is much larger than the size of D_t, $N_s \gg N_t$. [23]

A domain D can be represented by $D = \{\chi, P(X)\}$, which contains two parts in the feature space χ and the edge probability distribution $P(X)$ where $X = \{x_1, ..., x_n\} \in \chi$. A task can be represented by $T = \{y, f(x)\}$, consisting of two parts in the label space y and the target prediction function $f(x)$, which could also be regarded as a conditional probability function $P(x \mid y)$ [23].

Transfer learning has a number of advantages. With respect to certain specialized domains such as speech emotion recognition (SER), insufficient training data is an inescapable problem. The collection of data for SER is complex and expensive, making it difficult to build a large-scale, high-quality annotated dataset. Transfer learning relaxes the hypothesis that training data must be independent and identically distributed with the test data, which makes it a viable option for domains exhibiting insufficient training data, such as SER [23]. Provided this relaxed hypothesis, models in the target domain need not to be trained from scratch, significantly reducing the demand of training data and training time in the target domain [23].

There exist two main forms of transfer learning for deep learning methods in *fine-tuning* and *feature extraction*. In feature extraction, pre-trained network parameters can be applied as

features to train conventional classifiers that require lower numbers of training data [6]. Fine-tuning involves training some of the layers of a pre-trained network, gently adjusting the weights of the given network in accordance with the target domain [6]. The work performed throughout this thesis made heavy utilization of both methods.

Feature Extraction

The feature extraction method of transfer learning modifies the last fully connected layer(s) to fit the number of output classes of a target task while maintaining the existing parameters of the convolutional layers without updating them [43]. Thus, the weights of the pre-trained network are not updated when training a model with feature extraction. This method offers the advantage of faster training times compared to fine-tuning at the expense of some classification accuracy, typically.

Fine-tuning

Fine-tuning aims to create the highest learning impact on the final, fully-connected (data-dependent) layers of the network while leaving the earlier (data-independent) layers mostly intact [6]. This is accomplished by allowing training to occur on some of the layers of a pre-trained network. Therefore, with fine-tuning the weights of the pre-trained network will be adjusted in the layers the user has defined as trainable. In general, the closer the source domain is to the target domain, the more layers can be unfrozen and trained on. Fine-tuning incurs a high risk of overfitting though, so a very low learning rate must be used in order to prevent the weights of the network changing too abruptly [43].

VGG-16 and VGG-19 Model

The VGG-16 and VGG-19 models were first introduced in [44] by Karen Simonyan and Andrew Zisserman of the Visual Geometry Group of the Department of Engineering Science at the University of Oxford. This particular model placed first and second in the 2014 ImageNet challenge for the localisation and classification tasks, respectively [44]. The VGG models are unique in that they utilize very small (3×3) convolutional filters. The VGG-16 model consists of thirteen convolutional layers with five max pooling layers, three fully connected layers and a softmax output layer [44]. The convolutional stride is fixed to one pixel and the spatial padding of a convolutional layers' input is such that the spatial resolution is preserved after convolution, meaning the padding is one pixel for (3×3) convolutional layers [44]. The max pooling is performed over a (2×2) pixel window with a stride of two [44]. The VGG-19 model adds an additional three convolutional layers distributed throughout the architecture. Figure 25 displays the architecture of both the VGG-16 and VGG-19 models.

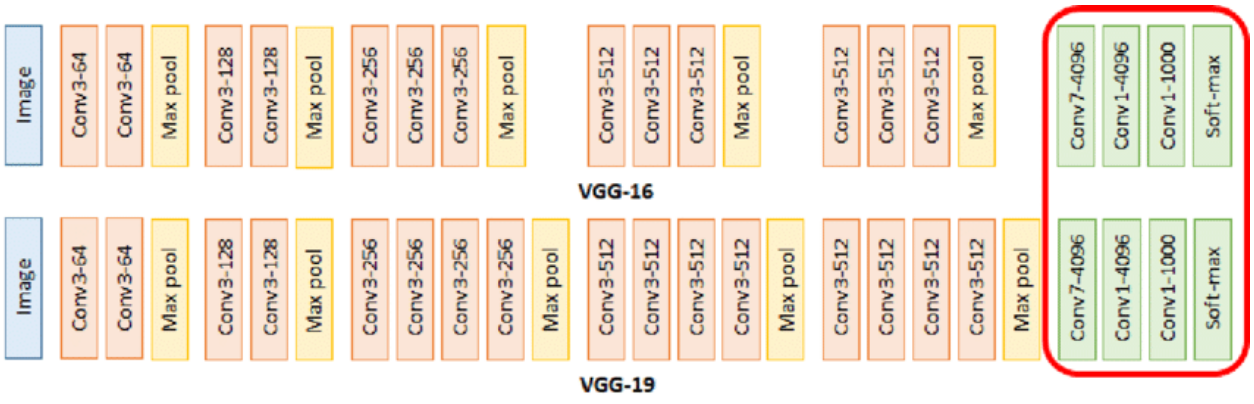

Figure 25. VGG-16 and VGG-19 Architecture [45].

InceptionV3 Model

InceptionV3, first introduced in [46], is the third edition of Google's Inception Convolutional Neural Network (GoogLeNet). InceptionV3 makes use of techniques such as *label*

smoothing, factorized (7 × 7) convolutions, batch normalization and the use of an auxiliary classifier to propagate label information lower down the network. The founding principle of InceptionV3 is that it exchanges architectural simplicity for a lower number of parameters. This is accomplished through *inception modules*. This in turn allows InceptionV3 to be deployed in environments where memory and computational constraints are of concern [46]. Figure 26 shows the composition of a typical inception module used in the original GoogLeNet architecture.

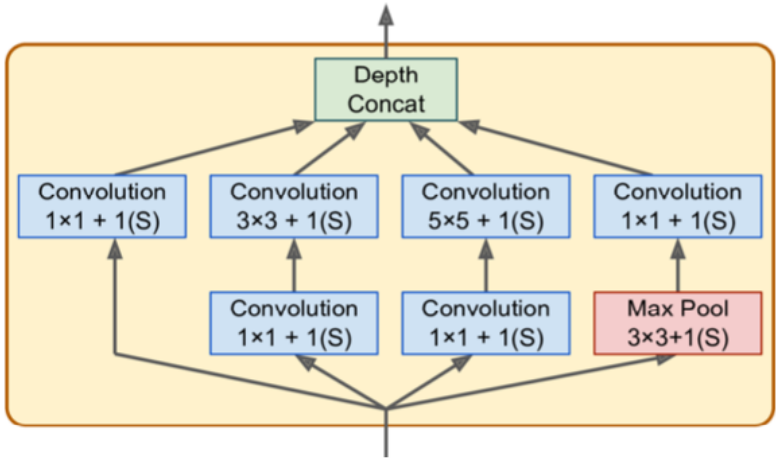

Figure 26. Inception Module Composition [37].

In an inception module, the input signal is first copied and fed to four different layers. The second set of convolutional layers utilizes different kernel sizes. This allows them to capture patterns at different scales [37]. Each of the layers utilizes a stride of one and the same padding, thus their outputs all have the same height and width of their inputs. This in turn allows for the concatenation of all the output along the depth dimension in the *depth concat layer*. This essentially stacks the feature maps from all four top convolutional layers. This allows inception modules to output fewer feature maps than their inputs, reducing dimensionality, cutting both computational cost and the number of parameters [46]. Figure 27 displays the architecture of the InceptionV3 model.

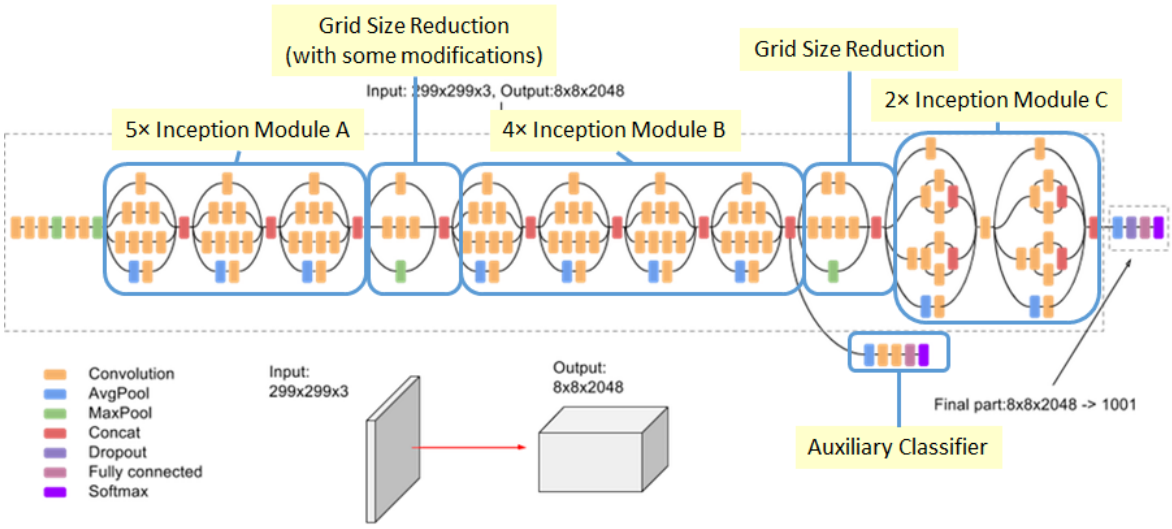

Figure 27. InceptionV3 Architecture [47].

Label smoothing is a mechanism to regularize the classifier layer by estimating the marginalized effect of label-dropout during training [46]. This serves to allow the model to be more adaptable. The aim of factorizing convolutions is to reduce the number of parameters without decreasing the efficiency of the network [46]. For example, utilizing one layer of (5×5) filters results in $5 \times 5 = 25$ parameters, whereas two layers of (3×3) filters results in $(3 \times 3) + (3 \times 3) = 18$ parameters. This is a twenty-eight percent reduction in the number of parameters [46]. These various techniques allow InceptionV3 to be a considerably more efficient model than the VGG-16 and VGG-19 models.

Xception Model

The Xception model was first introduced in [48] by François Chollet. The architecture of the Xception model is largely derived from that of the InceptionV3 model. Hence the name Xception is derived from "extreme inception" [48]. The Xception architecture has roughly the same number of parameters as InceptionV3 but makes a more efficient use of the model parameters yielding performance gains [48]. Within the architecture of Xception, inception

modules are replaced with *depthwise separable convolutions* [48]. A depthwise separable convolution consists of:

> A depthwise convolution, i.e. a spatial convolution performed independently over each channel of input, followed by a pointwise convolution, i.e. a (1 × 1) convolution, projecting the channel's output by the depthwise convolution onto a new channel space. [48]

Regular convolutional layers use filters that try to simultaneously capture spatial patterns (an oval for example) and cross-channel patterns (for example, mouth + eyes + nose = face) [37]. A separable convolutional layer assumes that spatial patterns and cross-channel patterns can be modeled separately. Thus, a depthwise separable convolution serves to apply a single spatial filter for each input feature map and look exclusively for cross-channel patterns. Figure 28 displays the composition of a depthwise separable convolutional layer. Figure 29 shows the architecture of the Xception model.

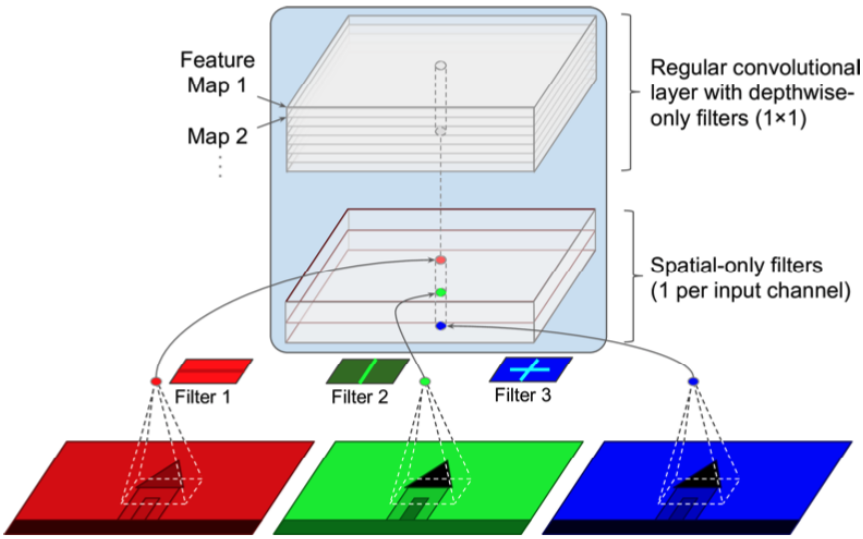

Figure 28. Depthwise Separable Convolutional Layer [37].

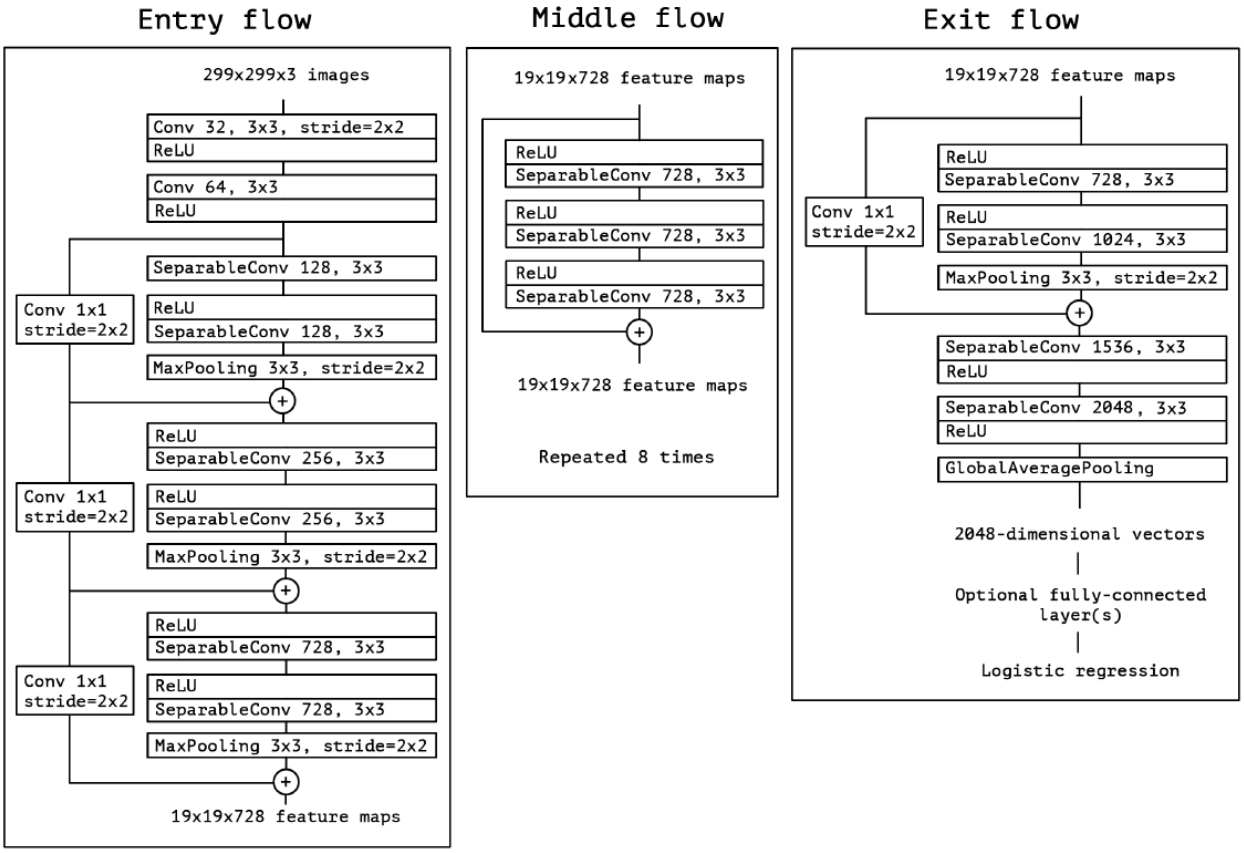

Figure 29. Xception Model Architecture [48].

In figure 29, SeperableConv layers are the depthwise separable convolutions treated as inception modules placed throughout the whole architecture of the model.

ResNet-50 Model

ResNet-50 is a variant of the ResNet model first introduced in [49]. Various ResNet models exist with depths of 18, 34, 50, 101 and 152 layers [49]. For the research performed in this work, the 50-layer version of ResNet was utilized. The primary concept of the work in [49] is to reformulate layers as learning residual functions with reference to the layer inputs, rather than learning unreferenced functions, i.e. the signal feeding into a layer is also added to the output of a layer located higher up the stack. The stack of these reformulated layers known as *residual units* creates what is referred to as a *residual network* [49]. When training a neural

network, the goal is to make it model a target function $h(x)$. By adding input x to the output of the network via a skip connection, the network will be forced to model $f(x) = h(x) - x$ rather than $h(x)$ [49]. This concept is referred to as *residual learning* [49]. Figure 30 displays the concept of residual learning.

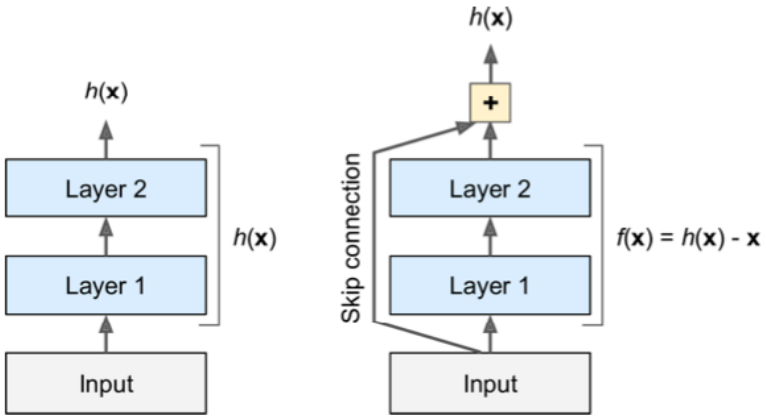

Figure 30. Residual Learning [37].

Utilizing many skip connections allows the network to start making considerable progress even if several layers have not yet begun training. Figure 31 displays the architectures of the various ResNet variants based on the depth of the network.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 31. ResNet Variant Architectures [49].

AlexNet

AlexNet was introduced in 2012 within [50], winning the ImageNet challenge that year by a considerable margin. AlexNet was a massive leap forward for CNNs. It was the first network architecture to stack convolutional layers directly on top of each other rather than stacking a pooling layer on top of each convolutional layer [37]. The concepts detailed in [50] have largely carried through into the other aforementioned neural networks and are still largely used today. AlexNet was the first to utilize the rectified linear unit (ReLU) activation function, rather than the tanh function [50]. This offered an incredible reduction in training time by a factor of six on the CIFAR-10 dataset compared to the same experiment with the tanh function [50].

AlexNet also allowed for multi-GPU training by placing half of the model's neurons on one GPU and the other half on another GPU [50]. This not only allowed for a larger model to be trained, but also reduced the training time [50]. The work in [50] also introduced the concept of overlapping pooling. Traditionally, CNNs pooled outputs of neighboring groups of neurons with no overlapping. When overlapping was introduced, it yielded a reduction in error by roughly half a percent. The authors of [50] also found that models utilizing overlapping pooling were more difficult to overfit. AlexNet also made considerable use of the regularization method known as *dropout* throughout the architecture. At the time, dropout was a relatively new concept and largely unexplored. Figure 32 shows the architecture of AlexNet from [50].

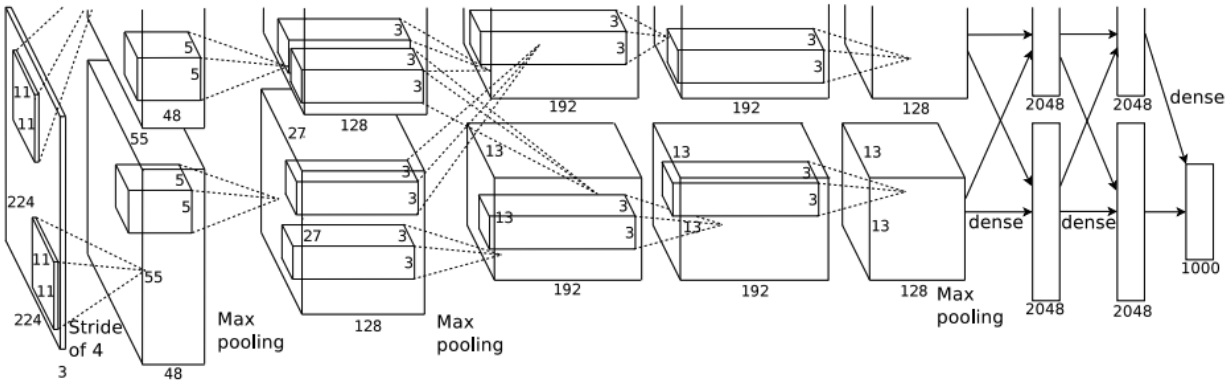

Figure 32. AlexNet Architecture [50].

A competitive normalization step known as *local response normalization* was used throughout the architecture of AlexNet after certain ReLU steps. With local response normalization, the most strongly activated neurons inhibit other neurons located at the same position in neighboring feature maps [50]. Thus, local response normalization encourages different feature maps to specialize, pushing them apart and forcing them to explore a wider range of features, improving generalization [50]. Equation 14 displays how local response normalization is applied.

$$b_i = a_i\left(k + \alpha \sum_{j=j_{low}}^{j_{high}} {a^2}_j\right)^{-\beta} \quad with:$$
$$j_{high} = min\,(i + \tfrac{r}{2}, f_n - 1)\,,$$
$$and\; j_{low} = max\,(0,\, i - \tfrac{r}{2}) \qquad (14)\;[37]$$

In which, b_i is the normalized output of the neuron located in feature map i, at some row u and column v, a_i is the activation of that neuron after the ReLU step, but prior to normalization, k, α, β and r are hyper-parameters where k is the bias, r is the depth bias and f_n is the number of feature maps [37].

Ensemble Learning

Ensemble learning is a branch of machine learning that combines multiple machine learning algorithms and models to make predictions. There exist several methods of ensemble learning, *voting classifiers* are particularly popular. In a voting ensemble classifier, the input data is fed to all the classifiers in the ensemble and predictions are made separately. There are two options for the style of voting to be used, hard and soft voting. With hard voting, the predictions of each classifier in the ensemble are aggregated and the class that gets the most votes is the overall prediction of the ensemble. In a soft voting ensemble the class with the highest class probability, averaged over all the individual classifiers is the prediction of the ensemble. Soft voting ensembles give more weight to highly confident votes and in turn tend to achieve higher performance than hard voting ensembles. Voting ensemble classifiers typically offer some marginal performance gains over the individual classifiers they are composed of.

Other ensemble methods such as model averaging ensemble classifiers and stacked generalization ensembles are common as well. A *model averaging ensemble* combines the prediction from each individual classifier equally. A *weighted average ensemble* allows multiple models to contribute to a prediction in proportion to their trust or estimated performance. *Stacked generalization ensembles* take a weighted average ensemble a step further by training an entirely new model that learns how to best combine the contributions from each submodel. This is commonly referred to as *stacking* and can result in better predictive performance than any single contributing model. Overall, ensemble learning methods offer a relatively inexpensive opportunity to increase classification performance from a computational resource perspective,

and can be particularly useful in situations where there is a lack of training data, such as speech emotion recognition, or where robustness is a priority.

V. RESEARCH METHODOLOGY

Experimental Procedure

For the research conducted in this work, two speech emotion corpora were utilized - the Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS) and the Crowd-sourced Emotional Multimodal Actors Dataset (CREMA-D). To begin, the RAVDESS corpus was selected for training and tuning the various transfer learning based convolutional neural networks with a variety of image enhancement techniques applied to the spectrograms. RAVDESS was used as the primary dataset due to its recording quality as well as its overall size. With 1,440 samples total, training models on RAVDESS is considerably quicker than training on CREMA-D with 7,442 samples. These factors made RAVDESS a more appealing choice for initial implementations of modeling and tuning.

Data augmentation methods were deployed on the training sets of both the RAVDESS and CREMA-D speech corpora. This not only served to increase the amount of available training data for each speech corpus but also to improve the generalizability of the models to the real-world by introducing noise to the audio and promoting robustness. If the speech emotion recognition (SER) model were only trained on clean speech data, it would not perform very well in real-world applications as noise is present in almost every environment found in the real-world. This would therefore impact the classification accuracies of the model in a negative fashion when deployed in the real-world.

Therefore, the various convolutional neural network models were trained on both the original, clean speech datasets and their corresponding counterparts with augmented data. This serves to determine what impact the ensemble of data augmentation methods have on the

classification accuracy of the various models. In addition to the data augmentation performed on the raw audio files, experiments in which the spectrograms were slightly augmented and enhanced were also performed. The impact of converting the spectrograms to grayscale, contrast-stretching and the combination of these two techniques was studied within this work. Again, the various models and datasets were all trained individually with each of these techniques to determine the impact of the given technique.

With respect to transfer learning, both methods of fine-tuning and feature extraction were used for each transfer learning model. This serves to provide performance data with regards to the tradeoff between the two techniques.

Performance Metrics

To assess the results of the experiments conducted in this research, a variety of individual performance metrics were selected. They are detailed below.

Training and Test Accuracy

In machine learning, the dataset for the task at hand is initially split into two parts - the *training set* and the *test set*. The training set is the portion of the dataset used for actually training the machine learning algorithm. The test set is used to get an unbiased evaluation of the performance of a given model. Thus, the test set is separated from the model during the training process. A common ratio for partitioning the training and test set is 80:20. With respect to speech emotion recognition (SER) this ratio was broadly used in most research publications. To remain consistent with other research in SER, within this thesis all datasets were split with the 80:20 ratio, with eighty percent of data being used for training, while the remaining twenty percent was used for the test set. Typically, a validation split may also be used. In the case of transfer learning, however, there are not many hyper-parameters to tune as the models are largely pre-trained. Thus, a validation split does not serve as much benefit in this use-case. Not making use of a validation split also preserves more data for the training set. All splits were also *stratified,* meaning that within a split there are an equal number of data samples per emotional class.

The formula for calculating the *training accuracy* can be found in equation 15 . The *true labels* are the ground truths, meaning the actual labels of data classes included in the given dataset. The number of correct predictions is calculated by simply comparing the predicted labels to the true labels. The training accuracy serves to measure how well the machine performed

when predicting on samples from the training set. For computing the training accuracy, the machine makes predictions on the data it was trained on. Thus, training accuracy is typically the highest among the classification accuracies.

$$Training\ Accuracy\ (\%) = \frac{\#\ of\ correct\ predictions\ on\ training\ samples}{Total\ \#\ of\ data\ samples\ in\ training\ set} \times 100 \quad (15)$$

The *test accuracy* is given in equation 16. This classification accuracy measures the model on previously unseen data by dividing the total number of correct predictions made on the test set data with the data samples' total number in the test set. The test accuracy serves as an unbiased representation of model performance as it is only computed once after training is complete and all model parameters have been finalized.

$$Test\ Accuracy\ (\%) = \frac{\#\ of\ correct\ predictions\ on\ test\ samples}{Total\ \#\ of\ data\ samples\ in\ test\ set} \times 100 \quad (16)$$

The training and test accuracies are the most popular performance metrics for machine learning and deep learning. A quality machine learning model exhibits high scores for both classification accuracies, which is indicative of strong prediction capabilities.

Precision and Recall

While classification accuracy provides insight as to how well a model performs overall, it is not indicative of how well the model is performing with respect to classifying individual classes. The metrics of *precision* and *recall* are relevant in the case of evaluating different classification rates.

The average precision score is calculated by dividing the sum of true positives across all classes by the sum of true positives and false positives. For example, when classifying the neutral emotion of the RAVDESS dataset, the true positives are the samples that were correctly classified as "neutral." The false positives are the samples classified as "neutral," but were, in

actuality, one of the other seven emotions. The false negatives are the "neutral" samples that were incorrectly classified as other emotions. The true negatives are the samples that were correctly classified as other emotions. Equation 17 shows the calculation for the average precision score.

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives} \qquad (17)$$

Therefore, precision quantifies the number of correct predictions made on the test set out of all the predictions labeled by the model as "true" [51].

The average recall score is calculated by dividing the sum of true positives by the sum of true positives and false negatives across all classes. Equation 18 shows the calculation for the average recall score.

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives} \qquad (18)$$

Thus, recall quantifies the number of correct predictions made on the test set out of all the actual "true" instances [51].

The metrics of precision and recall are beneficial for classification problems, particularly in cases when the classes are unbalanced. For the research performed in this work, the average precision and recall scores were calculated for models making predictions on the test set.

F-score

The *F-score* combines precision and recall into a single metric. The F-score is the harmonic mean of precision and recall [37]. It is particularly useful for easily comparing two classifiers. Whereas the regular mean treats all values equally, the harmonic mean gives more weight to low values. As a result, a classifier will only receive a high F-score if both the recall and precision are high. Equation 19 shows the computation of the F-score, denoted F_1.

$$F_1 = \frac{2}{\frac{1}{precision} + \frac{1}{recall}} = 2 \times \frac{precision \times recall}{precision + recall} = \frac{True\ Positives}{True\ Positives + \frac{False\ Negatives + False\ Positives}{2}} \quad (19)\ [37]$$

The F-score favors classifiers that exhibit similar scores for precision and recall. There is however a tradeoff associated with precision and recall, increasing precision reduces recall, and increasing recall reduces precision. This is referred to as the *precision/recall tradeoff* [37].

Accuracy Curves

Accuracy curves are plots useful in providing visual cues as to whether a model is *overfitting* or *underfitting*. The vertical axis is representative of the accuracy. The horizontal axis shows the number of *epochs*, being the number of passes over the training set. If a model were to be trained for thirty epochs, it is trained over the entire training set thirty times. In machine learning, the *bias* error is the error generated by the learning algorithm's wrong assumptions, and the *variance* error is the error generated by the model being over-sensitive to the small changes in the training set. If a model is not complex enough to learn the data's properties, it is said to have high bias and will fail to correctly classify the inputs, which will yield inadequate training and test accuracies. This phenomenon is known as underfitting and may be alleviated by increasing the number of parameters used in the model or decreasing the degree of regularization.

Conversely, if too many parameters are being used in training a model, the model will closely fit to the data in the training set. This results in a high degree of variance, and the model will fail to generalize when deployed on previously unseen data from the test set. This phenomenon is known as overfitting. Overfitting may be corrected by using more training data, reducing the number of parameters, or increasing the regularization. Ideally, a good balance between bias and variance is desired. With the limited amounts of labeled data in smaller

datasets, mitigating the potential of overfitting becomes increasingly more difficult [52]. Figure 33 shows the accuracy curve plot of an overfitted model.

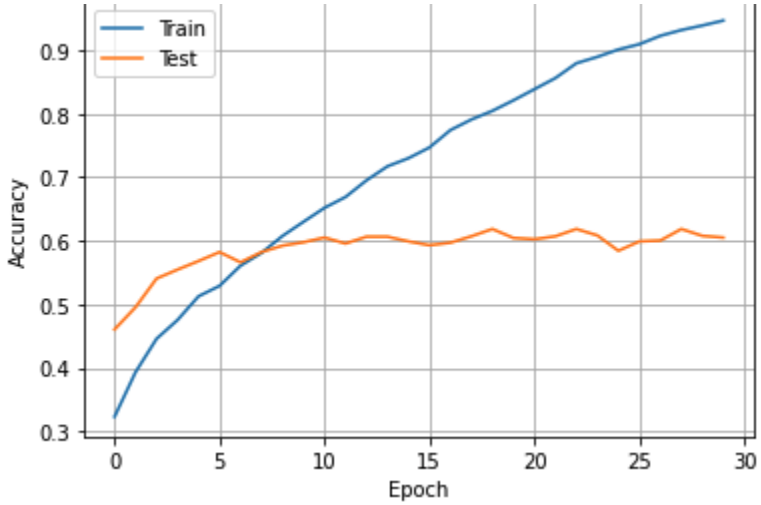

Figure 33. Accuracy Curve of an Overfitted Model.

Loss Curves

Loss curves plot the training and test set losses against the number of epochs used during model training. Similar to accuracy curves, loss curves are used to evaluate a models' performance in this thesis with respect to overfitting and underfitting. If the test set loss continues to increase with the number of epochs, this is indicative that the model is overfitting. Figure 34 displays the loss curve of a model exhibiting overfitting. With respect to multiclass classification tasks, categorical cross-entropy is the loss function that is typically used. This particular loss function calculates the cross-entropy between the class labels and the predictions made by the model. Throughout this thesis, the categorical cross-entropy loss is the cost function that was used with all models.

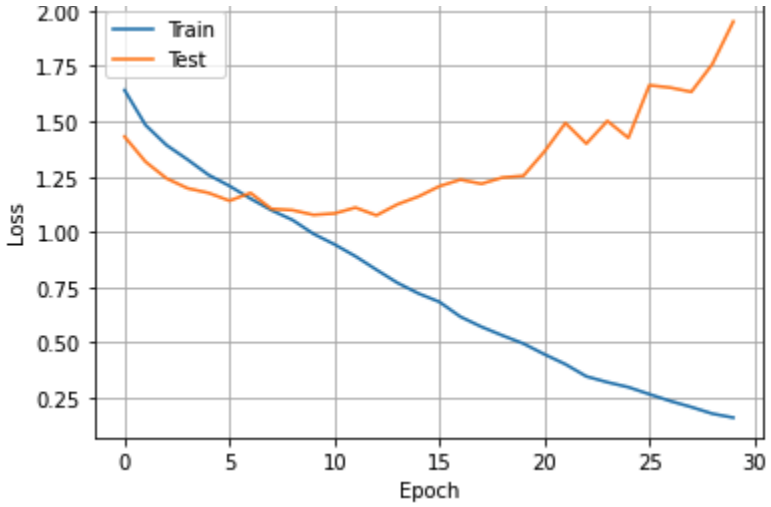

Figure 34. Loss Curve of an Overfitted Model.

Confusion Matrix

A *confusion matrix* is a simple visual representation of how well a particular machine learning model is predicting data from each class. It is composed of a two-dimensional array of numbers. One of its axes represents the true labels of the test data, while the other axis is representative of the predicted labels of the test data. Figure 35 displays an example of a typical confusion matrix. In this particular example, the horizontal axis is for the predicted labels and the vertical axis is composed of the true labels. Each of the eight emotional classes are labeled on both the axes. The diagonal numbers, from the top left to bottom right corners, represent the two axes' labels. These numbers represent the number of data samples the particular model classified correctly as the predicted labels match the true labels for this particular set of numbers. Numbers outside this diagonal represent the number of samples misclassified by the model. The total number of data samples belonging to any class in the test set can be found by adding all the numbers on a straight line corresponding to that label on the true label axis, i.e. all numbers in a row. A confusion matrix is particularly helpful in identifying the class a particular sample was misclassified as.

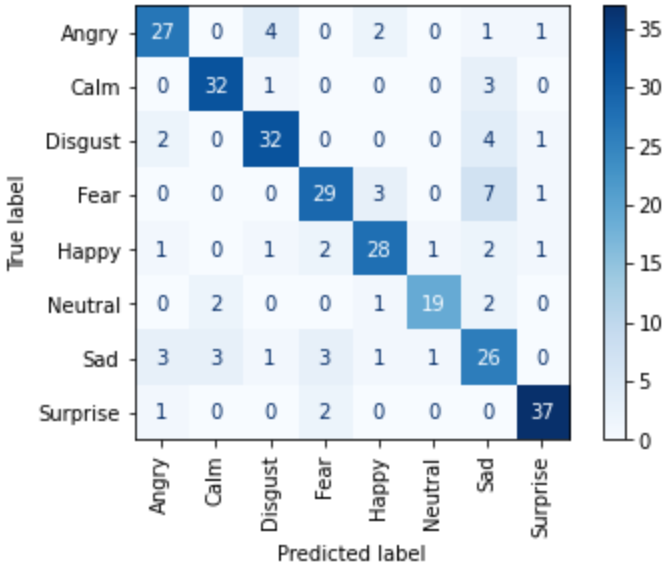

Figure 35. Example of A Confusion Matrix.

Training Time

Given the various methods and transfer learning models utilized throughout the research performed in this thesis, the training time of each model was recorded. Although training time is somewhat dependent on implementation, there may be vast differences in training times in which the differences cannot be attributed to implementation alone. Provided the differences in depth and complexity amongst the transfer learning models as well as the varying amounts of training data that the models were trained on, training time also serves to provide another point of comparison amongst the different models.

Methodology

The flow diagram displayed in figure 36 explains the end-to-end procedure for the work performed throughout this thesis. To begin, a speech corpus was selected. Data preprocessing was performed, generating metadata that relates the actor, emotion performed, etc. to each of the raw audio files within the selected corpus. Next, the data was split into training and test sets. If data augmentation was to be performed, it was done so at this point in the procedure. Then the log-scaled mel spectrograms were generated. When generated, the axes, padding and any corresponding labels were removed from the spectrogram figures as this data was not relevant. The spectrograms were then uploaded to a Google Cloud Storage bucket along with a CSV file that contained the corresponding metadata for the given dataset. A TensorFlow dataset input pipeline was created. This allows for iterations of the data to be passed in a streaming fashion, so the full dataset need not fit in memory. The pipeline utilizes a batch size of 32 and prefetch. Prefetch allows later elements to be prepared while the current element is being processed, in parallel. This improves both latency and throughput. Within the pipeline, image enhancement techniques were applied if necessary and the images were scaled and standardized appropriately to be used as input to the various models. Next, the various transfer learning models were trained on the original and augmented sets of training data. A modified version of the AlexNet deep neural network architecture was also implemented and serves as a baseline for deep neural networks not utilizing transfer learning. The modifications were made to the fully connected layers of the AlexNet architecture to better suit the task at hand. Finally, an unbiased performance evaluation was obtained by utilizing the test set in accordance with the various performance metrics previously mentioned. Model averaging ensembles composed of the

top-performing models for each speech corpus were developed and evaluated on the respective test sets as well.

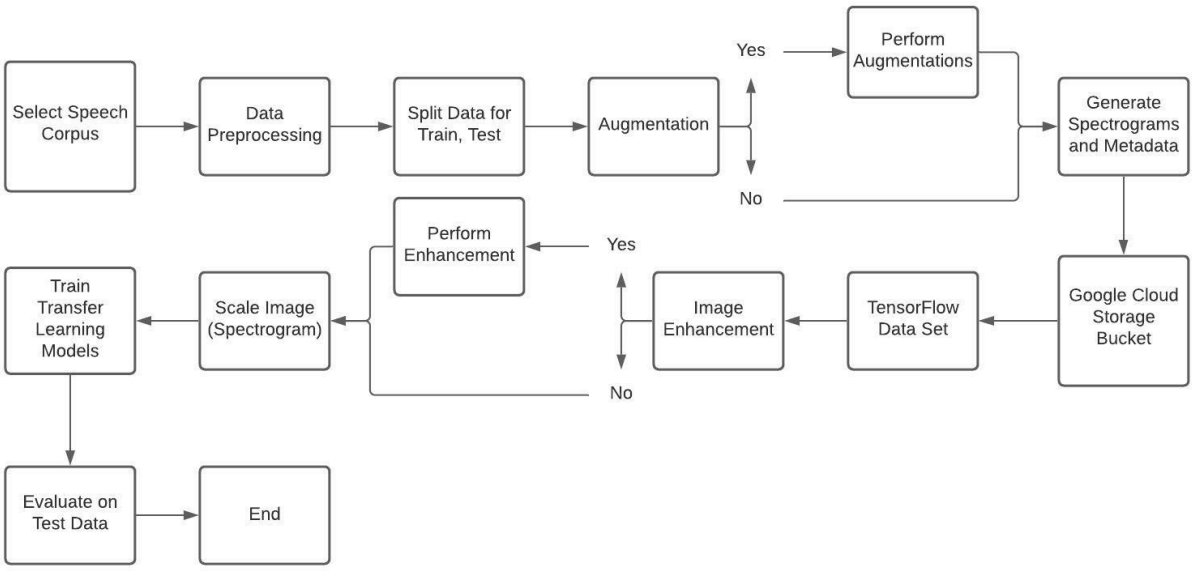

Figure 36. Flow Diagram Showing End-to-End Procedure.

The overall procedure is broken down into ten Jupyter Notebooks, each serving a specific purpose. Table 2 lists the various notebooks and briefly describes their corresponding purpose.

Notebook Name	Purpose
NB1	Initial preprocessing of data
NB2	Generate Log-scaled Mel-spectrogram values for 1D CNN
NB3	Deploy 1D CNN on original data
NB4	Deploy 1D CNN on augmented data
NB5	Log-scaled Mel Spectrogram generation and data augmentation
NB6	Transfer Learning: Feature Extraction for original datasets
NB7	Transfer Learning: Feature Extraction for augmented datasets
NB8	Transfer Learning: Fine-tuning and AlexNet for original datasets
NB9	Transfer Learning: Fine-tuning and AlexNet for augmented datasets
NB10	Model Averaging Ensembles

Table 2. Overview of Jupyter Notebooks.

Multiple versions of NB1, NB2, NB3, NB4 and NB5 exist for each of the speech corpora. Furthermore, for each speech corpus, multiple versions of NB6, NB7, NB8 and NB9 exist for each of the processed datasets and image enhancement techniques. Two versions of NB10 exist, one for each speech corpus.

160 experiments were completed with transfer learning models through this research. Four datasets were generated, four types of image enhancement (or lack thereof) were utilized, five transfer learning models were utilized with both the feature extraction and fine-tuning methods of implementation. Figure 37 displays an overview of the various transfer learning experiments performed. An additional 16 experiments were completed with the various datasets and image enhancement techniques with the AlexNet baseline DNN model. Each of the trained models utilized in these experiments were saved so they could be made available and easily be

reused at a later time. Lastly, four model averaging ensembles were developed and evaluated on the corresponding test sets. In total, 180 experiments were conducted throughout this research.

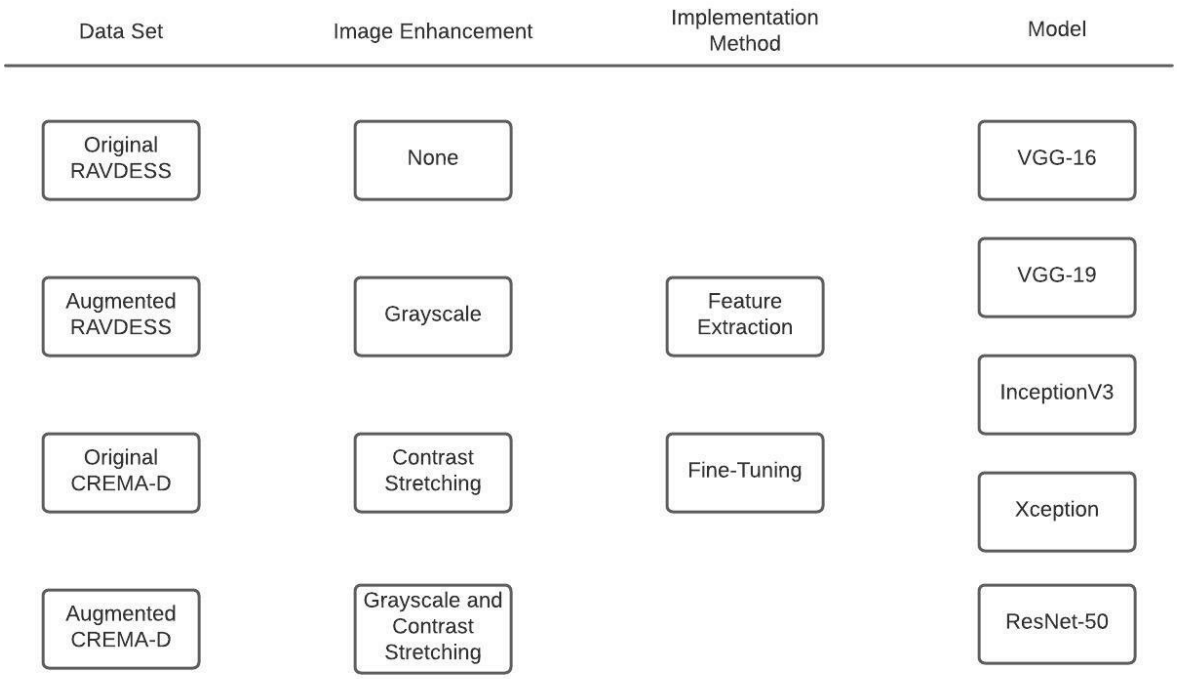

Figure 37. Overview of Transfer Learning Experiments Completed.

Training and Test Set Size

As previously mentioned, an 80:20 split was used to generate the training and test sets for modeling purposes. Each of the data augmentation methods detailed in the previous section were performed on the audio samples in the sets of training data derived from the RAVDESS and CREMA-D datasets. This served to not only increase the amount of available training data, but also improve the generalizability of the machine learning models to the real-world. The audio samples in the test set derived from each of the datasets remained unaugmented. Table 3 summarizes the dimensionality of the training and test sets used throughout this work.

Dataset	Training Set Size	Test Set Size
Original RAVDESS	1,152 Samples	288 Samples
Augmented RAVDESS	9,216 Samples	
Original CREMA-D	5,953 Samples	1,489 Samples
Augmented CREMA-D	47,624 Samples	

Table 3. Training and Test Set Size.

Pre-Trained Models and the ImageNet Dataset

The transfer learning based neural networks mentioned in previous sections were utilized in this work with weights pre-trained on the ImageNet dataset. The ImageNet dataset these models are pre-trained on contains approximately 1.2 million training images from 1,000 classes [53]. The size of the ImageNet dataset makes it an ideal candidate for use in transfer learning. The performance of the aforementioned models on the ImageNet dataset has also been verified through the ImageNet challenge, in which all models performed quite well in various years. Table 4 summarizes the dimensionality of the neural networks utilized for transfer learning in this research.

Model	Size (MB)	Parameters	Depth
VGG-16	528	138,357,544	23
VGG-19	549	143,667,240	26
InceptionV3	92	23,851,784	159
Xception	88	22,910,480	126
Resnet-50	98	25,636,712	50

Table 4. Dimensionality of Transfer Learning Neural Networks [54].

Data Input Pipeline

The TensorFlow input pipeline utilized for this research makes use of parallel computing to improve latency and throughput. This is largely powered by TensorFlow's autotune feature. Autotune sets the number of parallel calls dynamically based on CPU resource availability at runtime. The pipeline uses multiple threads to process elements as they are passed as input to the various machine learning models. Prefetch is also used in the pipeline allowing elements to be prepared while the current element is being processed, in parallel, via autotune. The prefetch operation utilizes a background thread and internal buffer to fetch elements from the Google Cloud Storage bucket ahead of the time they are requested. The input pipeline utilizes a batch size of 32. The primary purpose of implementing a data input pipeline is to conserve memory resources. This is particularly helpful with larger datasets that may either heavily tax or exhaust memory resources when attempting to hold the entirety of the dataset in memory.

Image Enhancement Techniques

With the aim of increasing classification performance by constructing invariant features not present in the original spectrogram images, three image enhancement techniques were examined. These techniques include performing *contrast-stretching* on the spectrograms, converting the spectrograms to *grayscale* and combining the two aforementioned techniques. Combining the contrast-stretching and grayscale conversion techniques was performed in the hope that it would yield a synergistic effect. The image enhancement techniques were applied to spectrograms in the data pipeline prior to being passed as input to a given model. Figure 38

displays the effect of contrast-stretching on a spectrogram exhibiting the disgust emotion from the RAVDESS corpus.

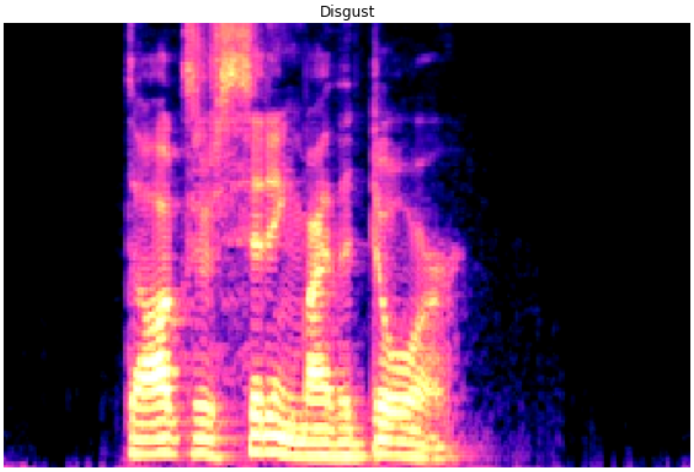

Figure 38. Spectrogram with Contrast-stretching.

Contrast-stretching is a transformation function that expands a narrow range of input levels into a wide, stretched, range of output levels, resulting in an image of higher contrast [55]. Equation 20 displays how to perform contrast-stretching at the individual pixel level.

$$s = T(r) = \frac{1}{1 + (m/r)^E} \quad (20)\ [55]$$

In which, s is the intensity of the output pixel, r is the intensity of the input pixel and E controls the slope of the function, i.e. the factor of stretching to apply. Figure 39 visualizes the transformation function utilized when performing contrast-stretching. For the contrast-stretching performed in this research, a stretch factor of two was utilized.

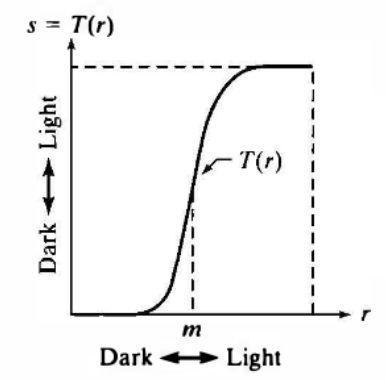

Figure 39. Contrast-stretching Transformation Function [55].

A grayscale image is essentially a data matrix whose values represent shades of gray. Thus, grayscale images have only one color channel, conserving memory compared to RGB images. This reduction in data could improve latency and throughput with respect to a real-time speech emotion recognition system, yielding faster predictions and a more enjoyable user experience. Figure 40 displays a grayscale converted spectrogram exhibiting the disgust emotion from the RAVDESS corpus.

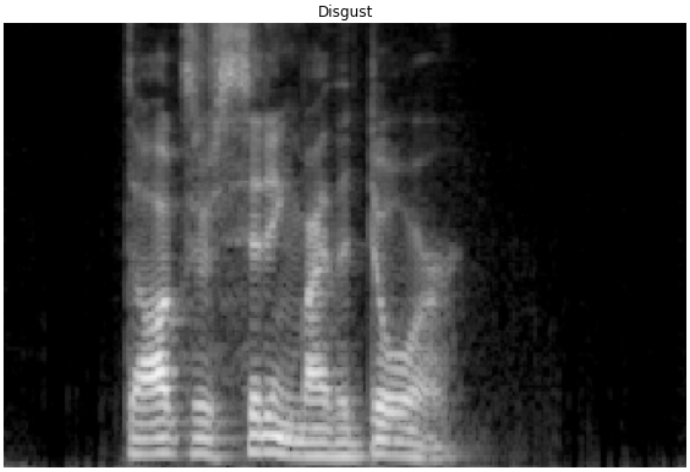

Figure 40. Grayscale Converted Spectrogram.

Given that the transfer learning models utilized for this research were pre-trained on the ImageNet database, which contains RGB images, these neural networks require a three channel color input. Therefore, it is necessary to convert the grayscale images to RGB by stacking the

one (gray) color channel three times to create an image with three color channels, albeit with the absence of actual color. Figure 41 shows the grayscale converted spectrogram with contrast-stretching applied exhibiting the disgust emotion from the RAVDESS corpus.

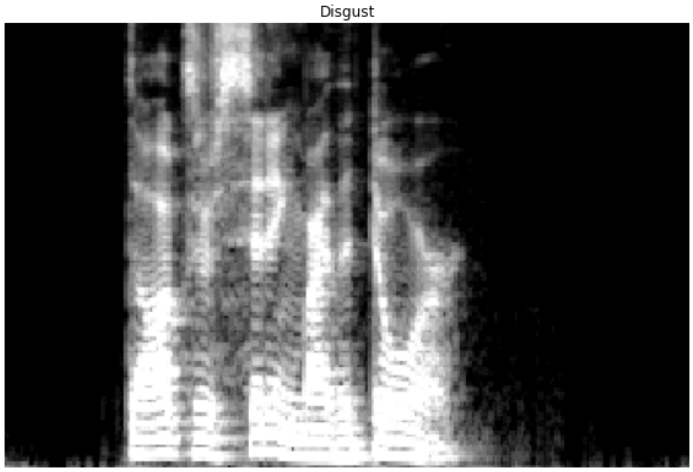

Figure 41. Grayscale Converted Spectrogram with Contrast-stretching.

Model Training and Tuning Parameters

Each of the neural networks were trained for thirty epochs. As the various models take input from the previously mentioned input data pipeline, the batch size is 32 as is used by the pipeline. The various transfer learning models used in this research utilize the previously mentioned RMSProp optimizer. The AlexNet models that serve as baseline DNNs utilize the Adam optimizer. A learning rate of 0.00001 was utilized for each of the transfer learning neural networks. With transfer learning it is necessary to use a lower learning rate as fine-tuning allows user-defined layers of the given transfer learning model to be trainable. This means the weights for these trainable layers of the given model will also get updated with backpropagation in each epoch as each batch of data is passed. Utilizing a lower learning rate prevents getting stuck at a local minimum and also prevents sudden updates to the weights of the trainable layers of the

given transfer learning model. Such sudden updates to the weights may adversely affect the given model.

Given that the AlexNet models do not utilize transfer learning, an *annealing learning rate* was used for these models. An annealing learning rate anneals the learning rate over time. The work performed in [56] shows that "training with a large initial learning-rate followed by a smaller annealed learning rate can vastly outperform training with the smaller learning rate throughout." This is implemented through a callback function that monitors the test accuracy as the model progresses through epochs during training. The learning rate is reduced by a factor of .01 after three epochs with no improvement to the test accuracy. The initial learning rate is set to .001 and the minimum learning rate is set to 0.00001.

For the VGG-16 and VGG-19 models the last two convolutional blocks were set to trainable when utilizing the fine-tuning method. With the InceptionV3 model the last thirty-two layers were set to trainable for use in fine-tuning. With respect to the Xception model, the first two convolutional blocks were left untrainable, the remaining layers were set to trainable. The layers within the last convolutional block of the ResNet-50 model were unfrozen as the feature extraction experiments of this particular model did not perform as well as the others. Thus, this model would be even more prone to overfitting with more layers unfrozen.

Architecture of Fully Connected Layers

The transfer learning models utilized in this research were pre-trained on the ImageNet dataset which contains 1,000 classes [53]. The RAVDESS and CREMA-D datasets contain eight and six classes, respectively. Therefore, it was necessary to modify the fully-connected layers of the pre-trained models to accommodate the RAVDESS and CREMA-D datasets. A common

architecture for the fully-connected layers was implemented for all of the models utilized in this research. Figure 42 visualizes the architecture of these fully connected (dense) layers.

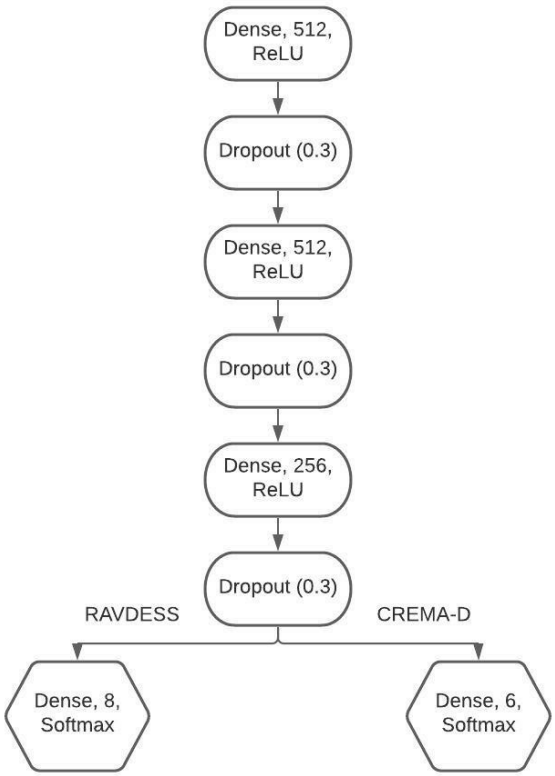

Figure 42. Architecture of Dense Layers.

The first dense layer is composed of 512 neurons, with ReLU as the activation function. It is followed by a dropout layer using a dropout rate of 0.3. This is followed by a second dense layer with 512 neurons and ReLU as the activation function. It is followed by a second dropout layer again using a dropout rate of 0.3 A third dense layer utilizing 256 neurons and ReLU as the activation function comes next. This is followed by one final dropout layer with a dropout rate of 0.3. Lastly, there is a dense output layer that utilizes eight or six neurons for the RAVDESS and CREMA-D datasets, respectively, with softmax as the activation function. For models utilizing the feature extraction method of transfer learning, the extracted features were passed as input to

these fully connected layers. Models that utilized the fine-tuning method of transfer learning had their originally fully connected layers replaced with those outlined above.

Computational Resources

Programming

For this thesis, Python version 3.7.12 was used for development throughout the entirety of the research procedure(s). TensorFlow and Librosa were also used extensively. Table 5 lists all the Python libraries used, along with their respective versions.

Library Name	Version
scikit-learn (sklearn)	0.22.2
tensorflow	2.6.0
matplotlib	3.2.2
numpy	1.19.5
pandas	1.1.5
librosa	0.8.1
seaborn	0.11.2
IPython	5.5.0
scikit-image (skimage)	0.16.2

Table 5. Python Libraries and Versions Utilized.

Hardware

Aside from personal workstations, Google Colab Pro Plus virtual machines were utilized extensively in this research as they provided access to a Tensor Processing Unit (TPU). TPUs were utilized to train virtually all of the machine learning models throughout this research. A TPU is a domain-specific matrix processor specialized for neural network loads. TPUs accelerate tensor operations through the systolic array architecture and quantization [57]. Systolic arrays are used to perform dot products of tensors on one core of a TPU rather than being spread out in

parallel across multiple GPU cores [57]. Quantization is the process of approximating an arbitrary value between two preset limits; this is leveraged by TPUs to compress floating-point calculations by converting continuous numbers to discrete numbers [57]. Through testing, it was observed that utilizing a TPU for training decreased training time by a factor of approximately three to four, depending on model, compared to training with a GPU. For example, a VGG-16 fine-tuning model trained on the augmented RAVDESS dataset with a GPU took approximately 6 hours and 43 minutes to train. This same model, when trained with a TPU, completed training in approximately 1 hour and 43 minutes.

VI. RESULTS AND DISCUSSION

The datasets labeled as “original” are composed of all the original clean speech recording samples of the Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS) and the Crowd-sourced Multimodal Actors Dataset (CREMA-D), respectively. The datasets with the label of “augmented” were generated by applying the various previously mentioned data augmentation techniques to the set of training data generated from each of the speech corpora. The test set of the augmented datasets are the same as those of the original datasets. The data split of 80:20 was used, where eighty percent of the data was used for training the models, and the remaining twenty percent formed the test set for evaluating model performance. Each data split was *stratified*, meaning that there were an equal number of data samples per emotion class in each of the data splits. The neutral class of both the RAVDESS and CREMA-D corpora had a lower number of samples compared to the other emotions included within the corpora. Experiments were conducted utilizing both the feature extraction and fine-tuning methods of transfer learning with the VGG-16, VGG-19, InceptionV3, Xception and ResNet-50 models pre-trained on the ImageNet dataset. A modified version of AlexNet serves as a baseline for a non-pre-trained deep neural network for these experiments. Each of the models were trained for 30 epochs. The various experiments were performed on all of the datasets with and without the previously mentioned image enhancement techniques.

Experiments with RAVDESS Speech Corpus

Twenty experiments each were conducted on the original and augmented RAVDESS datasets utilizing the feature extraction method of transfer learning. An additional twenty experiments each were conducted on the original and augmented RAVDESS datasets utilizing the fine-tuning method of transfer learning. Furthermore, four experiments each were performed on the original and augmented RAVDESS datasets using the modified AlexNet model. An additional two experiments utilizing model averaging ensembles were completed. This totals 90 experiments for the RAVDESS speech corpus.

Experiments with Feature Extraction Method of Transfer Learning

The results of the following experiments with transfer learning utilizing the feature extraction method are separated by the original and augmented datasets generated from the RAVDESS speech corpus. Table 6 displays the classification accuracy performance results for the transfer learning models utilizing the feature extraction method deployed on the original RAVDESS dataset.

Original RAVDESS Dataset: Feature Extraction Accuracy Performance				
Model	**No Image Enhancement**	**Contrast-stretching Applied**	**Grayscale Conversion**	**Grayscale Conversion and Contrast-stretching**
VGG-16	65%	66%	64%	63%
VGG-19	69%	63%	61%	60%
InceptionV3	61%	62%	64%	61%
Xception	64%	65%	67%	69%
ResNet-50	12%	43%	31%	44%

Table 6. Feature Extraction Accuracy Performance on Original RAVDESS Dataset.

The classification accuracy performance results for the feature extraction method of transfer learning on the original RAVDESS dataset displays a few notable trends. Firstly, the VGG-19 model with no image enhancement and the Xception model with grayscale and contrast-stretching applied to the spectrograms exhibit the highest classification accuracy with 69% accuracy each. Secondly, the deeper models such as InceptionV3, Xception and ResNet-50 generally benefit from the addition of image enhancement techniques with respect to classification accuracy. Table 7 displays the classification accuracy performance results of the same experiments performed on the augmented RAVDESS dataset.

Augmented RAVDESS Dataset: Feature Extraction Accuracy Performance				
Model	**No Image Enhancement**	**Contrast-stretching Applied**	**Grayscale Conversion**	**Grayscale Conversion and Contrast-stretching**
VGG-16	76%	76%	74%	74%
VGG-19	72%	75%	69%	76%
InceptionV3	67%	72%	72%	67%
Xception	68%	68%	69%	72%
ResNet-50	12%	45%	40%	52%

Table 7. Feature Extraction Accuracy Performance on Augmented RAVDESS Dataset.

The VGG-16 model with no image enhancement as well as with contrast-stretching performed on the spectrograms yields a classification accuracy of 76%. The VGG-19 model with the grayscale conversion and contrast-stretching image enhancement also yields this same accuracy figure. This is a 7% increase over the highest performing model from the same experiments on the original RAVDESS dataset. Again, it is observed that the deeper models exhibit greater benefits from the use of image enhancement techniques.

Experiments with Fine-tuning Method of Transfer Learning

The results of the following experiments with transfer learning models utilizing the fine-tuning method are separated by the original and augmented datasets generated from the RAVDESS speech corpus. Table 8 displays the classification accuracy performance results for the transfer learning models utilizing the fine-tuning method when trained on the original RAVDESS dataset.

Original RAVDESS Dataset: Fine-tuning Accuracy Performance				
Model	**No Image Enhancement**	**Contrast-stretching Applied**	**Grayscale Conversion**	**Grayscale Conversion and Contrast-stretching**
VGG-16	76%	76%	78%	74%
VGG-19	75%	75%	69%	70%
InceptionV3	62%	64%	62%	59%
Xception	60%	49%	49%	50%
ResNet-50	12%	40%	25%	42%

Table 8. Fine-tuning Accuracy Performance on Original RAVDESS Dataset.

The VGG-16 model utilizing the grayscale conversion technique of image enhancement exhibits the highest classification accuracy performance at 78%. This is a 2% improvement over the best performing model utilizing the feature extraction method from either dataset. It can be noted that the benefits of image enhancement techniques are not as significant with the fine-tuning method compared to the feature extraction method, though. Table 9 displays the classification accuracy performance results of the same experiments performed on the augmented RAVDESS dataset.

Augmented RAVDESS Dataset: Fine-tuning Accuracy Performance				
Model	**No Image Enhancement**	**Contrast-stretching Applied**	**Grayscale Conversion**	**Grayscale Conversion and Contrast-stretching**
VGG-16	84%	82%	80%	79%
VGG-19	81%	76%	81%	76%
InceptionV3	69%	68%	68%	67%
Xception	72%	66%	63%	66%
ResNet-50	31%	47%	39%	52%

Table 9. Fine-tuning Accuracy Performance on Augmented RAVDESS Dataset.

Once again, the VGG-16 model exhibits the highest classification accuracy performance at 84% with no image enhancement. This is a 6% increase over the highest performing model from the fine-tuning experiments on the original RAVDESS dataset. It can also be observed that the image enhancement techniques are not providing any benefit to the models aside from the ResNet-50 model. Overall, the VGG-16 model utilizing fine-tuning when trained on the augmented RAVDESS dataset yields the highest classification accuracy of any transfer learning model upon evaluation on the test set of the RAVDESS speech corpus.

Experiments with AlexNet Baseline DNN Model

The experiments conducted with a modified AlexNet deep neural network (DNN) serve to reinforce the impact of the data augmentation that was applied and determine what a non-pre-trained DNN is capable of. Table 10 displays the classification accuracy performance of the AlexNet model trained on the original and augmented RAVDESS datasets with the various image enhancement techniques (or lack thereof).

AlexNet Baseline DNN: Accuracy Performance				
Dataset	**No Image Enhancement**	**Contrast-stretching Applied**	**Grayscale Conversion**	**Grayscale Conversion and Contrast-stretching**
Original RAVDESS	71%	40%	38%	33%
Augmented RAVDESS	83%	72%	67%	61%

Table 10. AlexNet Accuracy Performance on Original and Augmented RAVDESS Dataset.

It can be observed that the AlexNet model, when deployed on the augmented RAVDESS dataset with no image enhancement yields a classification accuracy of 83%. This is a one percent reduction in performance compared to the best performing transfer learning model at 84% accuracy. The use of image enhancement techniques had a considerable negative impact on the classification accuracy, particularly when the model was deployed on the original RAVDESS dataset.

Further Evaluation of Top Performing Models

A VGG-16 model utilizing the feature extraction method of transfer learning and contrast-stretching yielded a classification accuracy of 76% when trained on the augmented RAVDESS dataset. Similarly, the VGG-16 model now utilizing the fine-tuning method of transfer learning and no image enhancement yielded the highest classification accuracy of 84% when trained on the augmented RAVDESS dataset of all individual classifier experiments conducted on the RAVDESS speech corpus. Following closely, the AlexNet DNN model using no image enhancement, when trained on the augmented RAVDESS dataset yielded a classification accuracy of 83%. The accuracy and loss curves of the feature extraction VGG-16

model are plotted in figures 43 and 44, respectively. Table 11 summarizes the performance metrics for the top performing VGG-16 model from the set of experiments conducted with feature extraction.

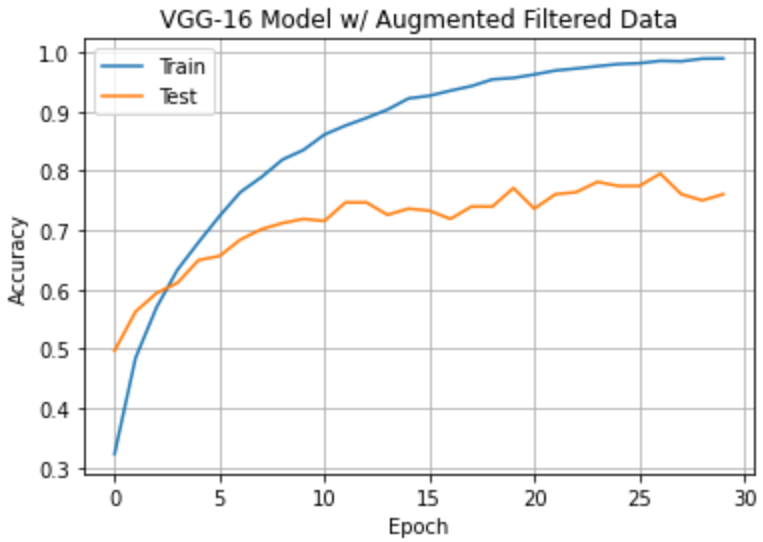

Figure 43. Accuracy Curve: VGG-16 Feature Extraction Model with Contrast-stretching Trained on Augmented RAVDESS Dataset.

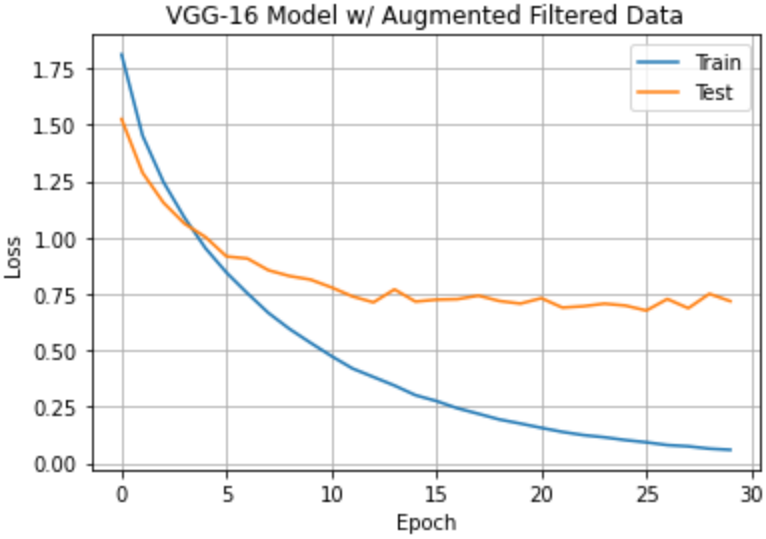

Figure 44. Loss Curve: VGG-16 Feature Extraction Model with Contrast-stretching Trained on Augmented RAVDESS Dataset.

Emotion	Precision	Recall	F1-Score
Angry	74%	91%	82%
Calm	70%	92%	80%
Disgust	94%	74%	83%
Fear	76%	72%	74%
Happy	73%	61%	67%
Neutral	77%	83%	80%
Sad	69%	58%	63%
Surprise	78%	80%	79%
Training Time: 2 Minutes 44 Seconds			

Table 11. Performance Metrics: VGG-16 Feature Extraction Model with Contrast-stretching Trained on Augmented RAVDESS Dataset.

From the accuracy curves of this model it can be observed that the training and test accuracies converge at approximately three epochs with roughly 60% accuracy. Beyond that, the two accuracy curves begin to diverge slightly but then remain consistent throughout the duration of training, with the test accuracy reaching 76%. The loss curve indicates that the error on the test set is greater than that of the training error, but remains consistent as training progresses. This is indicative that the model has slightly overfitted the training data. It can be observed that the model performs best at classifying the disgust emotion with an F-score of 83%. Conversely, the model exhibits its lowest F-score of 63% when classifying the sad emotion. The time required to train this particular model was 2 minutes and 44 seconds utilizing a tensor processing unit (TPU) on a Google Colab Pro Plus virtual machine. The accuracy and loss curves of the top

performing fine-tuning VGG-16 model are plotted in figures 45 and 46, respectively. Table 12 summarizes the performance metrics for the top performing fine-tuning VGG-16 model.

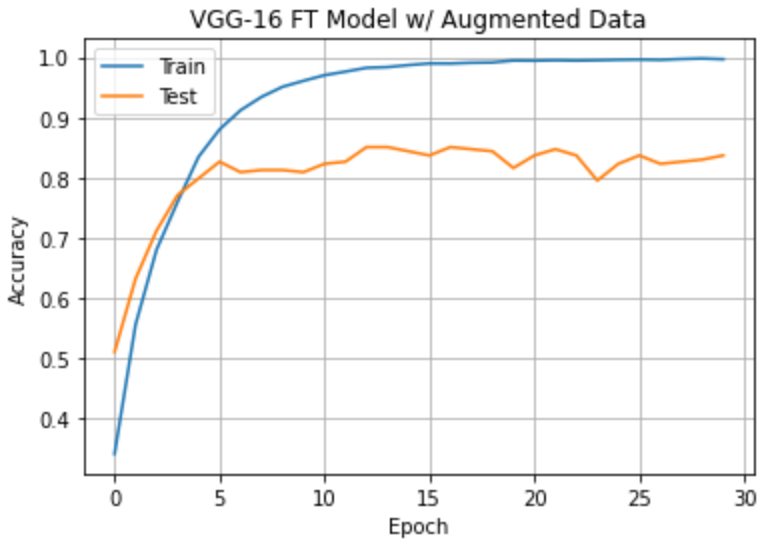

Figure 45. Accuracy Curve: VGG-16 Fine-tuning Model with No Image Enhancement Trained on Augmented RAVDESS Dataset.

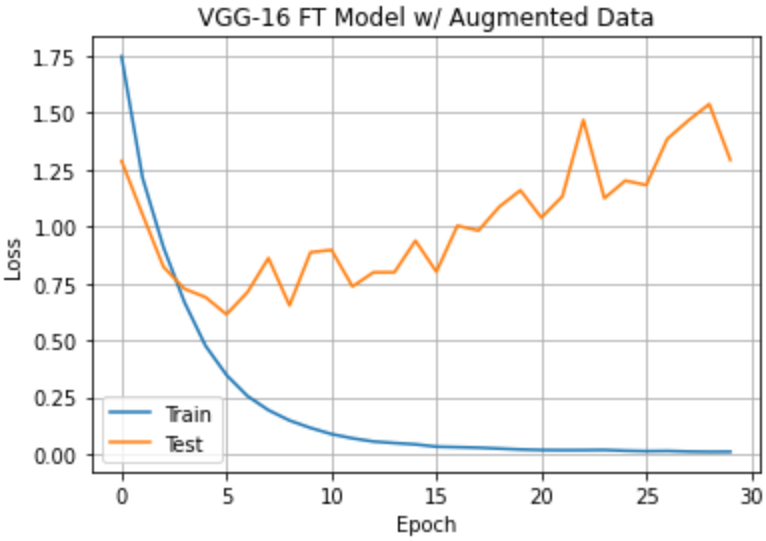

Figure 46. Loss Curve: VGG-16 Fine-tuning Model with No Image Enhancement Trained on Augmented RAVDESS Dataset.

Emotion	Precision	Recall	F1-Score
Angry	79%	77%	78%
Calm	81%	94%	87%
Disgust	79%	87%	83%
Fear	91%	80%	85%
Happy	85%	78%	81%
Neutral	88%	88%	88%
Sad	76%	68%	72%
Surprise	91%	97%	94%
Training Time: 1 Hour 43 Minutes			

Table 12. Performance Metrics: VGG-16 Fine-tuning Model with No Image Enhancement Trained on Augmented RAVDESS Dataset.

The accuracy curves of this model show that the training and test accuracies converge at approximately four epochs with roughly 80% accuracy. Beyond that, the two accuracy curves begin to diverge slightly but then remain consistent throughout the duration of training, with the test accuracy reaching 84%. The loss curve indicates that the error on the test set is much greater than that of the training error, particularly as the training epochs progress. This is indicative that the model is likely overfitting the training data. It can be observed that the model performs best at classifying the surprise emotion with an F-score of 94%. Conversely, the model exhibits its lowest F-score of 72% when classifying the sad emotion. The time required to train this particular model was 1 hour and 43 minutes utilizing a tensor processing unit (TPU) on a Google Colab Pro Plus virtual machine. The accuracy and loss curves of the second-best performing model for the RAVDESS speech corpus experiments, being the AlexNet DNN with no image

enhancement and trained on the augmented RAVDESS dataset, are plotted in figures 47 and 48, respectively. Table 13 summarizes the performance metrics for the AlexNet DNN model.

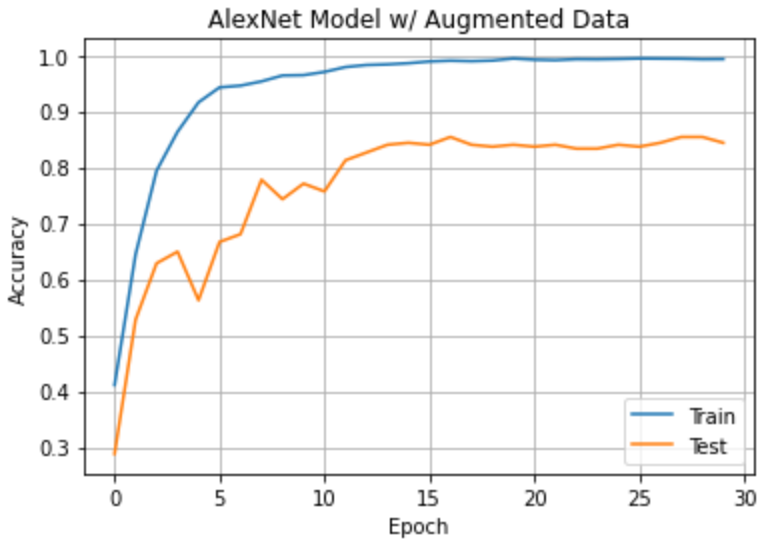

Figure 47. Accuracy Curve: AlexNet Model with No Image Enhancement Trained on Augmented RAVDESS Dataset.

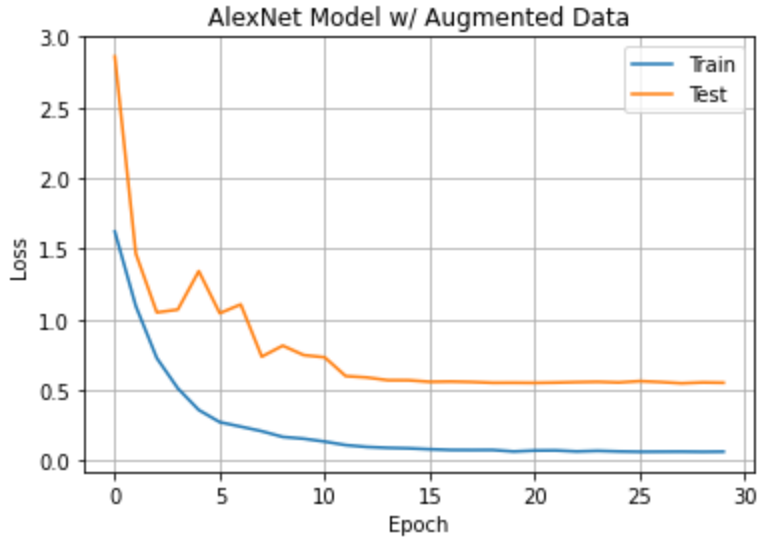

Figure 48. Loss Curve: AlexNet Model with No Image Enhancement Trained on Augmented RAVDESS Dataset.

Emotion	Precision	Recall	F1-Score
Angry	77%	86%	81%
Calm	79%	94%	86%
Disgust	85%	87%	86%
Fear	91%	75%	82%
Happy	85%	78%	81%
Neutral	95%	88%	91%
Sad	78%	76%	77%
Surprise	90%	93%	91%
Training Time: 1 Hour 42 Minutes			

Table 13. Performance Metrics: AlexNet Model with No Image Enhancement Trained on Augmented RAVDESS Dataset.

The accuracy curves of the AlexNet DNN model display that the training and test accuracies never truly converge. The test accuracy does however follow the training accuracy somewhat closely throughout the duration of training, eventually reaching 83%. The loss curve indicates a similar trend in the test error following the training error throughout the duration of training. Overall, this indicates that the model may be ever so slightly underfitting the data. It can be observed that the model performs best at classifying the surprise and neutral emotions with an F-score of 91% each. Conversely, the model exhibits its lowest F-score of 77% when classifying the sad emotion, similar to the VGG-16 model. The time required to train this particular model was 1 hour and 42 minutes utilizing a tensor processing unit (TPU) on a Google Colab Pro Plus virtual machine.

Experiments with Model Averaging Ensembles

Two model averaging ensembles were implemented, composed of models trained on the original and augmented versions of the RAVDESS dataset. The ensemble utilizing models trained on the original version of the RAVDESS dataset is comprised of the fine-tuning VGG-16 and VGG-19 models as well as the AlexNet model trained with no image enhancement techniques applied to the spectrograms. The ensemble utilizing models trained on the augmented version of the RAVDESS dataset is formed with the same models, also trained with no image enhancement techniques applied to the spectrograms. Table 14 displays the performance metrics for the ensemble containing models trained on the original RAVDESS dataset, figure 49 displays the confusion matrix of this ensemble.

Emotion	Precision	Recall	F1-Score
Angry	79%	77%	78%
Calm	86%	89%	88%
Disgust	82%	82%	82%
Fear	81%	72%	76%
Happy	80%	78%	79%
Neutral	90%	79%	84%
Sad	58%	68%	63%
Surprise	90%	93%	91%
Ensemble Test Accuracy: 80%			
VGG-16 Individual Test Accuracy: 76%			
VGG-19 Individual Test Accuracy: 75%			
AlexNet Individual Test Accuracy: 70%			

Table 14. Performance Metrics: Model Averaging Ensemble with Models Trained On Original RAVDESS Dataset.

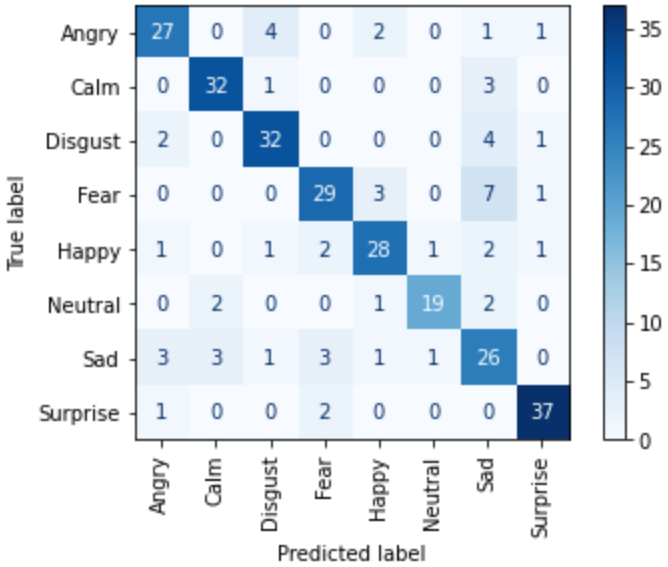

Figure 49. Confusion Matrix: Model Averaging Ensemble with Models Trained On Original RAVDESS Dataset.

From the table, it can be observed that the model averaging ensemble yields a 4% gain in classification accuracy over the best performing individual classifier in the ensemble. This is a considerable improvement over the individual classifiers. It is important to note, the ensemble is still performing particularly poorly when classifying the sad emotion, with an F-score of 63%. From the confusion matrix it can be inferred that the sad emotion is most commonly misclassified as the angry, calm, and fear emotions. Conversely, the ensemble performs very well when classifying the surprise emotion with an F-score of 91%. Table 15 displays the performance metrics for the ensemble containing models trained on the augmented RAVDESS dataset, figure 50 displays the confusion matrix of this ensemble.

Emotion	Precision	Recall	F1-Score
Angry	81%	83%	82%
Calm	80%	89%	84%
Disgust	82%	85%	84%
Fear	94%	82%	88%
Happy	83%	81%	82%
Neutral	88%	92%	90%
Sad	80%	74%	77%
Surprise	90%	95%	93%
Ensemble Test Accuracy: 85%			
VGG-16 Individual Test Accuracy: 84%			
VGG-19 Individual Test Accuracy: 81%			
AlexNet Individual Test Accuracy: 83%			

Table 15. Performance Metrics: Model Averaging Ensemble with Models Trained On Augmented RAVDESS Dataset.

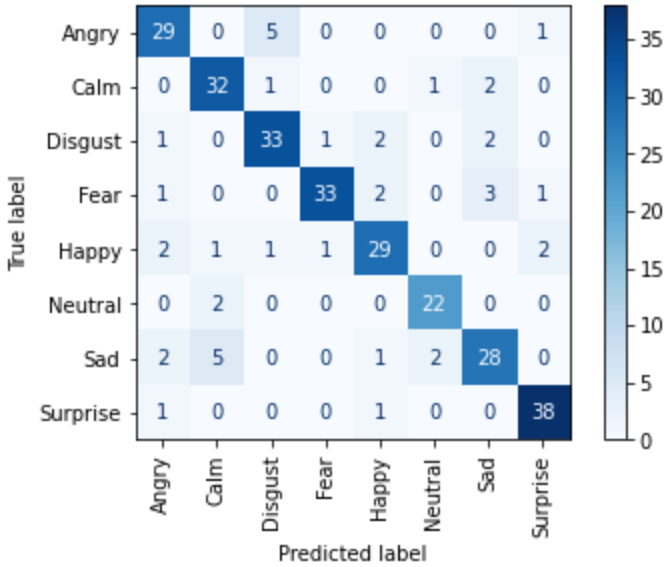

Figure 50. Confusion Matrix: Model Averaging Ensemble with Models Trained On Augmented RAVDESS Dataset.

The model averaging ensemble yields a 1% gain in classification accuracy over the best performing individual classifier in the ensemble. The results of the model averaging ensemble are consistent with the other experiments on the RAVDESS speech corpus as it exhibits the lowest performance when classifying the sad emotion with an f-score of 77%. Following this trend, the confusion matrix displays that the sad emotion is most commonly misclassified as the calm emotion, consistent with the other experiments. The ensemble performs best when classifying the surprise emotion with an F-score of 93%, also consistent with other experiments.

Overall, it is clear that model averaging ensembles are an effective way to aggregate the predictions of individual classifiers and improve upon the performance of the individual classifiers that compose the ensemble. The ensemble composed of models trained on the augmented version of the RAVDESS dataset now exhibit the highest classification accuracy of all experiments performed on the RAVDESS dataset at 85% accuracy. The ensemble formed with models trained on the original RAVDESS dataset exhibits 80% classification accuracy, which is quite impressive compared to the performance of the individual classifiers in the ensemble.

Experiments with CREMA-D Speech Corpus

Twenty experiments each were conducted on the original and augmented CREMA-D datasets utilizing the feature extraction method of transfer learning. An additional twenty experiments each were conducted on the original and augmented CREMA-D datasets utilizing the fine-tuning method of transfer learning. Furthermore, four experiments each were performed on the original and augmented CREMA-D datasets using the modified AlexNet model. An additional two experiments utilizing model averaging ensembles were completed. This totals 90 experiments for the CREMA-D speech corpus.

Experiments with Feature Extraction Method of Transfer Learning

The results of the following experiments with transfer learning utilizing the feature extraction method are separated by the original and augmented datasets generated from the CREMA-D speech corpus. Table 16 displays the classification accuracy performance results for the transfer learning models utilizing the feature extraction method deployed on the original CREMA-D dataset.

Original CREMA-D Dataset: Feature Extraction Accuracy Performance				
Model	**No Image Enhancement**	**Contrast-stretching Applied**	**Grayscale Conversion**	**Grayscale Conversion and Contrast-stretching**
VGG-16	58%	17%	57%	57%
VGG-19	57%	17%	55%	57%
InceptionV3	54%	50%	52%	52%
Xception	54%	54%	54%	54%
ResNet-50	38%	45%	40%	47%

Table 16. Feature Extraction Accuracy Performance on Original CREMA-D Dataset.

The classification accuracy performance results for the feature extraction method of transfer learning on the original CREMA-D dataset displays a few notable trends. Firstly, the VGG-16 model with no image enhancement applied to the spectrograms exhibits the highest classification accuracy with 58% accuracy. Secondly, the Xception model does not benefit from any form of image enhancement. Conversely, the ResNet-50 model exhibits accuracy performance gains of 2 - 9% through the utilization of image enhancement techniques. Lastly, contrast-stretching has a detrimental impact on the classification accuracy performance of the VGG-16 and VGG-19 models by roughly 40%. Table 17 displays the classification accuracy performance results of the same experiments performed on the augmented CREMA-D dataset.

Augmented CREMA-D Dataset: Feature Extraction Accuracy Performance				
Model	**No Image Enhancement**	**Contrast-stretching Applied**	**Grayscale Conversion**	**Grayscale Conversion and Contrast-stretching**
VGG-16	62%	57%	56%	58%
VGG-19	56%	56%	57%	55%
InceptionV3	54%	53%	53%	52%
Xception	56%	55%	54%	54%
ResNet-50	33%	49%	36%	51%

Table 17. Feature Extraction Accuracy Performance on Augmented CREMA-D Dataset.

The VGG-16 model with no image enhancement performed on the spectrograms yields a classification accuracy of 62%. This is a 4% increase over the highest performing model from the same experiments on the original CREMA-D dataset. It can be observed that image enhancement techniques do not provide benefit to any models aside from the ResNet-50 model.

Experiments with Fine-tuning Method of Transfer Learning

The results of the following experiments with transfer learning utilizing the fine-tuning method are separated by the original and augmented datasets generated from the CREMA-D speech corpus. Table 18 displays the classification accuracy performance results for the transfer learning models utilizing the fine-tuning method deployed on the original CREMA-D dataset.

Original CREMA-D Dataset: Fine-tuning Accuracy Performance				
Model	**No Image Enhancement**	**Contrast-stretching Applied**	**Grayscale Conversion**	**Grayscale Conversion and Contrast-stretching**
VGG-16	64%	62%	61%	58%
VGG-19	63%	61%	61%	60%
InceptionV3	17%	17%	17%	17%
Xception	17%	17%	17%	17%
ResNet-50	34%	43%	17%	17%

Table 18. Fine-tuning Accuracy Performance on Original CREMA-D Dataset.

The VGG-16 model with no image enhancement applied to the spectrograms achieves the highest classification accuracy performance at 64%, the VGG-19 model also without image enhancement follows close behind at 63%. This is a 2% increase in performance over the top-performing feature extraction model. Interestingly, the deeper transfer learning models perform particularly poorly on the original CREMA-D dataset when using the fine-tuning method, with the ResNet-50 model using the contrast-stretching image enhancement achieving just 43% accuracy. The InceptionV3 and Xception models exhibit accuracies of a lackluster 17%. Table 19 displays the classification accuracy performance results of the same experiments performed on the augmented CREMA-D dataset.

Augmented CREMA-D Dataset: Fine-tuning Accuracy Performance				
Model	**No Image Enhancement**	**Contrast-stretching Applied**	**Grayscale Conversion**	**Grayscale Conversion and Contrast-stretching**
VGG-16	64%	64%	62%	60%
VGG-19	63%	61%	61%	62%
InceptionV3	54%	53%	51%	51%
Xception	57%	55%	55%	54%
ResNet-50	17%	51%	29%	49%

Table 19. Fine-tuning Accuracy Performance on Augmented CREMA-D Dataset.

Once again, the VGG-16 models utilizing both no image enhancement and contrast-stretching applied to the spectrograms achieve the highest classification accuracies of 64% in this set of experiments. This does not offer any improvement over the highest performing model from the experiments on the original dataset using fine-tuning. Interestingly though, the deeper models greatly benefit from the additional training data generated through the data augmentation methods. The InceptionV3 and Xception fine-tuning models trained on the augmented CREMA-D dataset exhibit 34 - 40% gains in accuracy performance compared to those same models trained on the original CREMA-D dataset. The ResNet-50 model also exhibits an 8% increase in accuracy performance when trained on the augmented CREMA-D dataset.

Experiments with AlexNet Baseline DNN Model

The experiments conducted with a modified AlexNet deep neural network (DNN) serve to reinforce the impact of data augmentation and determine what a non pre-trained DNN is capable of. Table 20 displays the classification accuracy performance of the AlexNet model

deployed on the original and augmented CREMA-D datasets with the various image enhancement techniques (or lack thereof).

AlexNet Baseline DNN: Accuracy Performance				
Dataset	**No Image Enhancement**	**Contrast-stretching Applied**	**Grayscale Conversion**	**Grayscale Conversion and Contrast-stretching**
Original CREMA-D	17%	17%	17%	17%
Augmented CREMA-D	61%	62%	61%	61%

Table 20. AlexNet Accuracy Performance on Original and Augmented CREMA-D Dataset.

It can be observed that the AlexNet model, when deployed on the augmented CREMA-D dataset with contrast-stretching yields a classification accuracy of 62%. This is a two percent reduction in performance compared to the best performing transfer learning model at 64% accuracy. The AlexNet models performed particularly poorly when trained on the original, unaugmented CREMA-D dataset.

Further Evaluation of Top Performing Models

A VGG-16 model utilizing the feature extraction method of transfer learning and no image enhancement yielded a classification accuracy of 62% when trained on the augmented CREMA-D dataset. Of all experiments conducted on the CREMA-D speech corpus, the VGG-16 model now utilizing the fine-tuning method of transfer learning yielded the highest classification accuracies of 64% when trained on the augmented and original CREMA-D datasets, separately, with no image enhancement. Following closely, the AlexNet DNN model using contrast-stretching, when trained on the augmented CREMA-D dataset yielded a classification

accuracy of 62%. The accuracy and loss curves of the feature extraction VGG-16 model are plotted in figures 51 and 52, respectively. Table 21 summarizes the performance metrics for the top performing VGG-16 model from the set of experiments conducted with feature extraction.

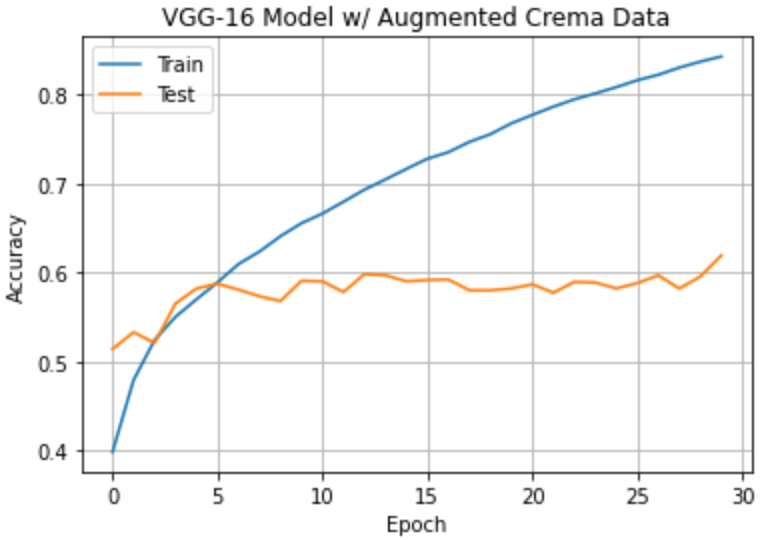

Figure 51. Accuracy Curve: VGG-16 Feature Extraction Model with No Image Enhancement Trained on Augmented CREMA-D Dataset.

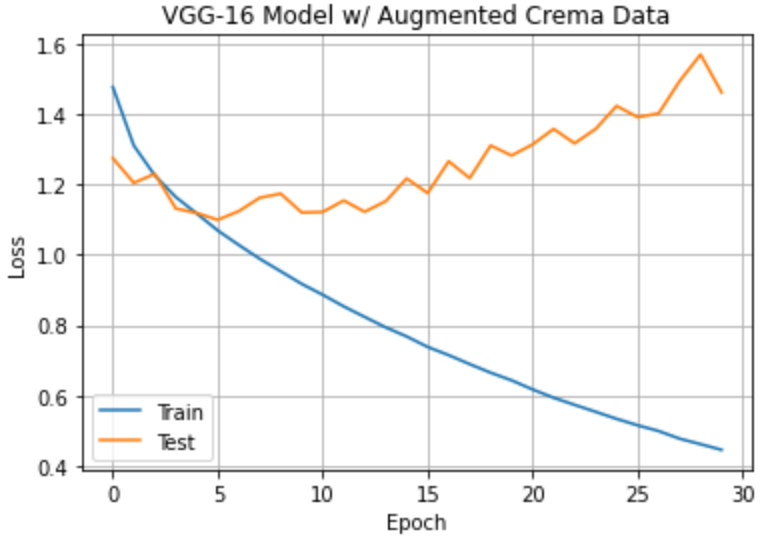

Figure 52. Loss Curve: VGG-16 Feature Extraction Model with No Image Enhancement Trained on Augmented CREMA-D Dataset.

Emotion	Precision	Recall	F1-Score
Angry	75%	82%	79%
Disgust	60%	53%	56%
Fear	57%	45%	50%
Happy	61%	62%	62%
Neutral	57%	69%	62%
Sad	59%	63%	61%
Training Time: 40 Minutes 30 Seconds			

Table 21. Performance Metrics: VGG-16 Feature Extraction Model with No Image Enhancement Trained on Augmented CREMA-D Dataset.

From the accuracy curves of this model it can be observed that the training and test accuracies converge at approximately five epochs with just under 60% accuracy. Beyond that, the two accuracy curves begin to diverge slightly but then remain consistent throughout the duration of training, with the test accuracy reaching 62%. The loss curve indicates that the error on the test set is greater than that of the training error and continues to increase as training progresses. This is indicative that the model has overfitted the training data. It can be observed that the model performs best at classifying the angry emotion with an F-score of 79%. Conversely, the model exhibits its lowest F-score of 50% when classifying the fear emotion. The time required to train this particular model was 40 minutes and 30 seconds utilizing a tensor processing unit (TPU) on a Google Colab Pro Plus virtual machine. The accuracy and loss curves of the top performing fine-tuning VGG-16 model are plotted in figures 53 and 54, respectively. Table 22 summarizes the performance metrics for the top performing fine-tuning VGG-16 model.

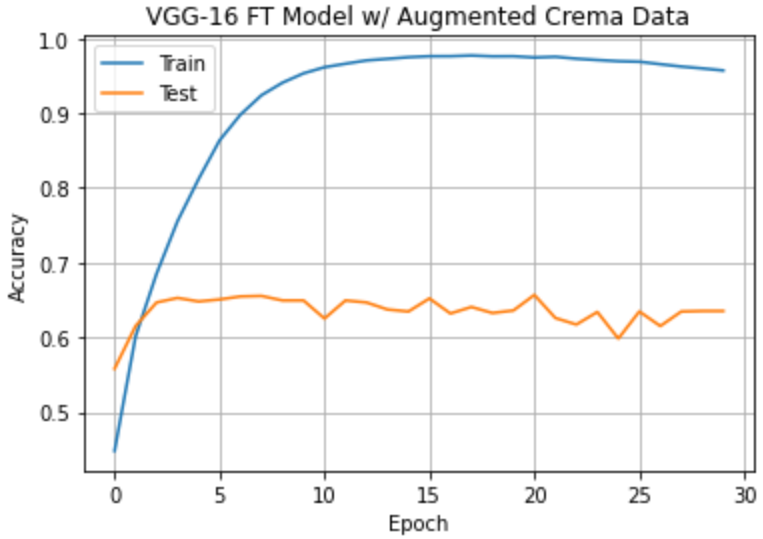

Figure 53. Accuracy Curve: VGG-16 Fine-tuning Model with No Image Enhancement Trained on Augmented CREMA-D Dataset.

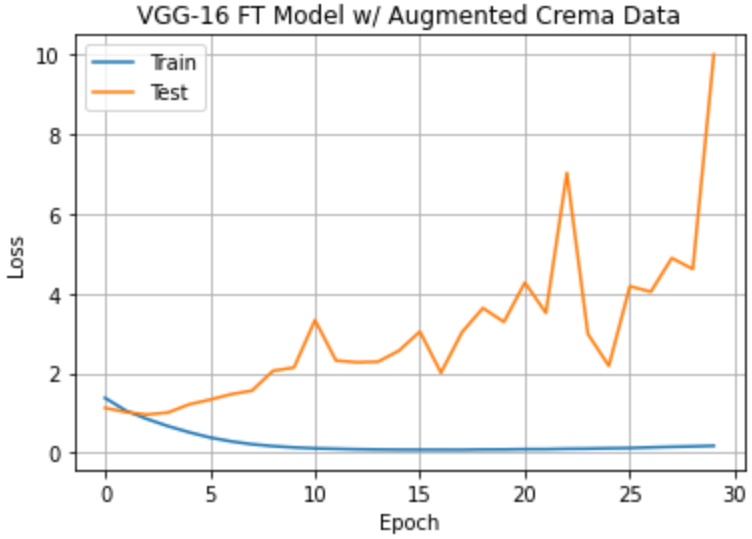

Figure 54. Loss Curve: VGG-16 Fine-tuning Model with No Image Enhancement Trained on Augmented CREMA-D Dataset.

Emotion	Precision	Recall	F1-Score
Angry	72%	79%	76%
Disgust	60%	62%	61%
Fear	64%	43%	51%
Happy	66%	61%	63%
Neutral	60%	77%	68%
Sad	58%	64%	61%
Training Time: 8 Hours 53 Minutes			

Table 22. Performance Metrics: VGG-16 Fine-tuning Model with No Image Enhancement Trained on Augmented CREMA-D Dataset.

From the accuracy curves of this model it can be observed that the training and test accuracies converge after the first few epochs with approximately 65% accuracy. Beyond that, the two accuracy curves begin to diverge severely with the test accuracy not improving throughout the duration of training, completing training with 64% accuracy. The loss curve indicates that the error on the test set is much greater than that of the training error, particularly as the training epochs progress. This is indicative that the model is severely overfitting the training data after the first few epochs. It can be observed that the model performs best at classifying the angry emotion with an F-score of 76%. Conversely, the model exhibits its lowest F-score of 51% when classifying the fear emotion. The time required to train this particular model was 8 hours and 53 minutes utilizing a tensor processing unit (TPU) on a Google Colab Pro Plus virtual machine.

The accuracy and loss curves of the second-best performing model for the CREMA-D speech corpus experiments, being the AlexNet DNN with contrast-stretching and trained on the

augmented CREMA-D dataset, are plotted in figures 55 and 56, respectively. Table 23 summarizes the performance metrics for the AlexNet DNN model.

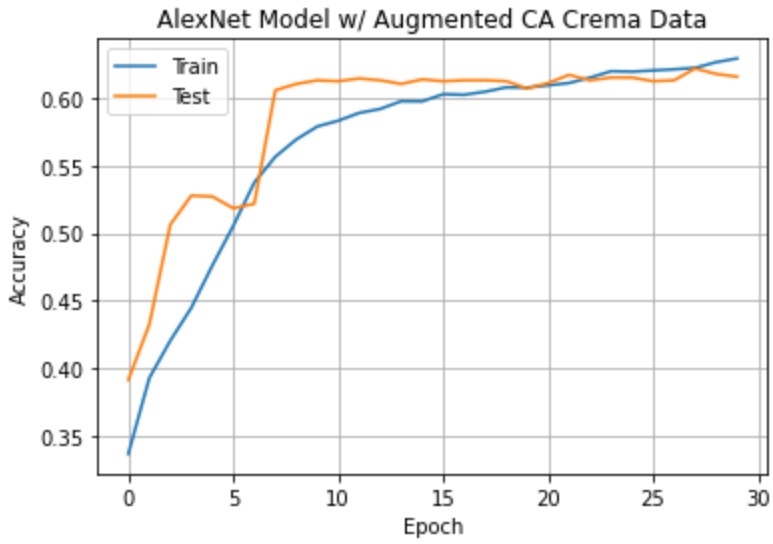

Figure 55. Accuracy Curve: AlexNet Model with Contrast-stretching Trained on Augmented CREMA-D Dataset.

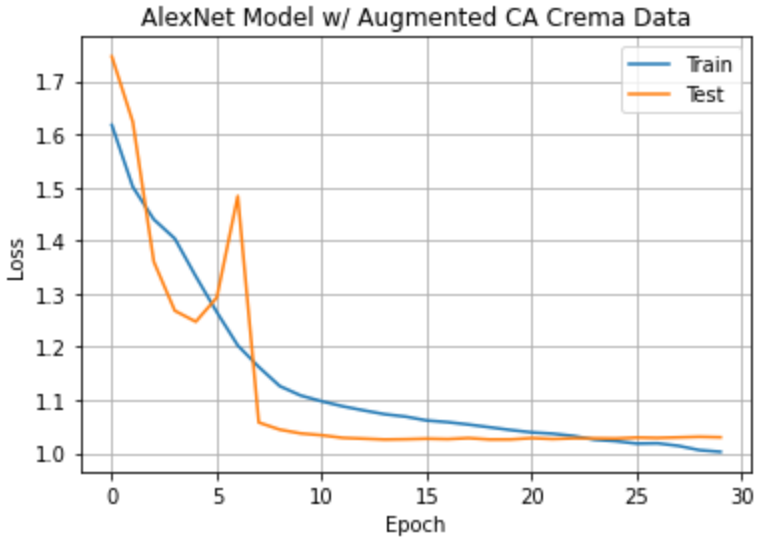

Figure 56. Loss Curve: AlexNet Model with Contrast-stretching Trained on Augmented CREMA-D Dataset.

Emotion	Precision	Recall	F1-Score
Angry	73%	76%	74%
Disgust	58%	57%	57%
Fear	55%	53%	54%
Happy	62%	62%	62%
Neutral	61%	69%	65%
Sad	60%	56%	58%
Training Time: 9 Hours 12 Minutes			

Table 23. Performance Metrics: AlexNet Model with Contrast-stretching Trained on Augmented CREMA-D Dataset.

The accuracy curves of the AlexNet DNN model display that the training and test accuracies converge multiple times throughout training. The test accuracy does however follow the training accuracy closely throughout the duration of training, eventually reaching 62%. The loss curve indicates a similar trend in the test error following the training error throughout the duration of training. Overall, this indicates that the model is neither overfitting or underfitting the data. It can be observed that the model performs best at classifying the angry emotion with an F-score of 74%. Conversely, the model exhibits its lowest F-score of 54% when classifying the fear emotion, similar to the two VGG-16 models. The time required to train this particular model was 9 hours and 12 minutes utilizing a tensor processing unit (TPU) on a Google Colab Pro Plus virtual machine.

Experiments with Model Averaging Ensembles

Two model averaging ensembles were implemented composed of models trained on the original and augmented versions of the CREMA-D dataset. The ensemble utilizing models

trained on the original version of the CREMA-D dataset is comprised of the fine-tuning VGG-16, VGG-19, and ResNet-50 models trained with no image enhancement techniques applied to the spectrograms. The ensemble utilizing models trained on the augmented version of the CREMA-D dataset is formed with the fine-tuning VGG-16 and VGG-19 models as well as the AlexNet model, also trained with no image enhancement techniques applied to the spectrograms. Table 24 displays the performance metrics for the ensemble containing models trained on the original RAVDESS dataset, figure 57 displays the confusion matrix of this ensemble.

Emotion	**Precision**	**Recall**	**F1-Score**
Angry	77%	79%	78%
Disgust	60%	65%	63%
Fear	59%	63%	61%
Happy	71%	59%	64%
Neutral	68%	74%	71%
Sad	62%	58%	60%
Ensemble Test Accuracy: 66%			
VGG-16 Individual Test Accuracy: 64%			
VGG-19 Individual Test Accuracy: 63%			
ResNet-50 Individual Test Accuracy: 34%			

Table 24. Performance Metrics: Model Averaging Ensemble with Models Trained On Original CREMA-D Dataset.

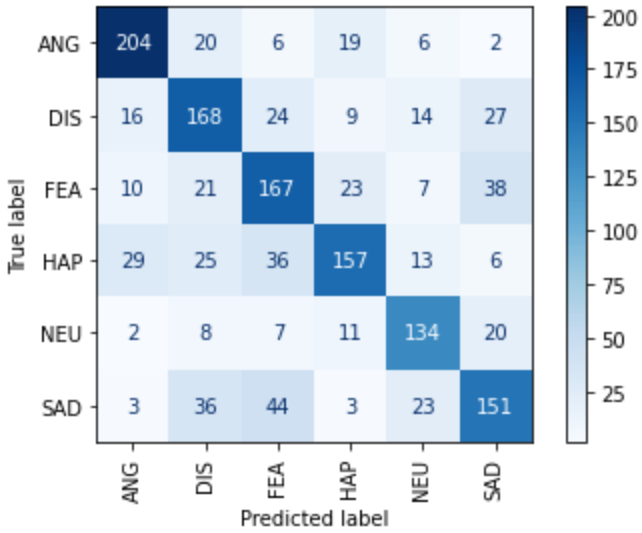

Figure 57. Confusion Matrix: Model Averaging Ensemble with Models Trained On Original CREMA-D Dataset.

From the table, it can be observed that the model averaging ensemble yields a 2% gain in classification accuracy over the best performing individual classifier in the ensemble. The ensemble is performing particularly poorly when classifying the sad emotion, with an F-score of 60%, similar to the ensemble experiments performed on the RAVDESS speech corpus. From the confusion matrix it can be inferred that the sad emotion is most commonly misclassified as the fear and disgust emotions, also consistent with the RAVDESS speech corpus experiments. Conversely, the ensemble performs very well when classifying the angry emotion with an F-score of 78%. Table 25 displays the performance metrics for the ensemble containing models trained on the augmented RAVDESS dataset, figure 58 displays the confusion matrix of this ensemble.

Emotion	Precision	Recall	F1-Score
Angry	76%	82%	79%
Disgust	64%	61%	63%
Fear	66%	47%	55%
Happy	69%	68%	68%
Neutral	61%	73%	66%
Sad	58%	65%	61%
Ensemble Test Accuracy: 66%			
VGG-16 Individual Test Accuracy: 64%			
VGG-19 Individual Test Accuracy: 63%			
AlexNet Individual Test Accuracy: 61%			

Table 25. Performance Metrics: Model Averaging Ensemble with Models Trained On Augmented CREMA-D Dataset.

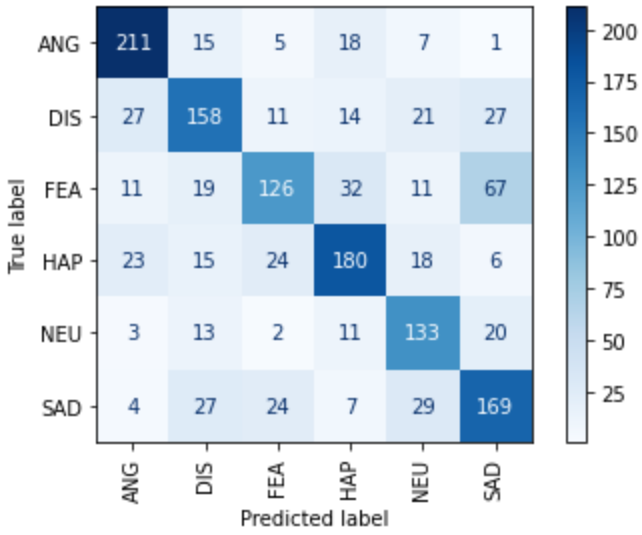

Figure 58. Confusion Matrix: Model Averaging Ensemble with Models Trained On Augmented CREMA-D Dataset.

The model averaging ensemble yields a 2% gain in classification accuracy over the best performing individual classifier in the ensemble and exhibits the same classification performance as the ensemble composed of models trained on the original CREMA-D dataset with 66% accuracy. This particular ensemble exhibits the lowest performance when classifying the fear emotion with an f-score of 55%. The confusion matrix displays that the fear emotion is most commonly misclassified as the happy and sad emotions. The ensemble performs best when classifying the angry emotion with an F-score of 79%, also consistent with other experiments on the CREMA-D speech corpus.

The ensembles composed of models trained on the original and augmented versions of the CREMA-D dataset now exhibit and share the highest classification accuracy of all experiments performed on the CREMA-D dataset at 66% accuracy. These results further reinforce the impact ensemble techniques may have with respect to improving classification performance within machine learning tasks.

Discussion

Overall, the models and methods developed in this work yielded improved classification performances on the RAVDESS speech corpus compared to what is found in literature. When deployed on the CREMA-D speech corpus, the methods and corresponding models yielded classification performance typical of what is currently found in literature. The variation in performance between the speech corpora could be attributed in part to the sampling rate. The RAVDESS speech corpus was composed with a sampling rate of 48 kHz, whereas the CREMA-D speech corpus was composed with a sampling rate 16 kHz [32], [34]. Thus, the audio recordings of the RAVDESS speech corpus contain more data, potentially yielding higher resolution spectrograms.

The transfer learning models, particularly those utilizing fine-tuning were prone to overfitting. This is a relatively common issue with transfer learning and could be alleviated through a few techniques. One such technique would be *early stopping*. Early stopping monitors a given performance metric such as the validation accuracy, test accuracy or the loss. The technique is implemented through a callback function. It will interrupt training when it measures no progress on the given metric for a certain number of epochs. It can also be used to roll back to the best model if checkpoint saving is used when training the model. Examining the accuracy and loss curves of the various models in this work, early stopping would have significantly reduced the overfitting exhibited by the models without much, if any, sacrifice to classification performance.

Conversely, the AlexNet models developed in this work did not suffer from overfitting and exhibited fairly similar classification performance to the transfer learning models. This can

largely be attributed to the data augmentation methods that were used to increase the amount of training data. This is also indicative that the source domain of the transfer learning models, being the ImageNet dataset, does not exhibit a strong association with the target domain of this work, being speech emotion recognition through the classification of spectrogram images.

With respect to the image enhancement techniques, the usage of such techniques did not provide much benefit overall. In some certain cases, the models did exhibit performance gains when utilizing these techniques but in general, the original spectrograms with no image enhancement yielded the best classification performances. Overall, only three techniques were examined within this work, there are a large number of other image enhancement techniques that could be utilized. Unfortunately, there is not a great deal of literature on image enhancement techniques and their usage in conjunction with convolutional neural networks, specifically when deployed on spectrograms for the purpose of speech emotion recognition.

For the CREMA-D datasets, the models were particularly accurate when classifying the angry emotion. Conversely, the models exhibited poor performance when classifying the fear emotion. With the RAVDESS datasets, the models classified the surprise, neutral, and angry emotion most accurately. Within these datasets, the models were particularly poor at classifying the sad emotion. These results somewhat resemble what was detailed through the work in [13], being that the fear and disgust emotions are more difficult to classify.

The model averaging ensembles implemented within this work exhibit the highest classification performance of all experiments conducted on both the RAVDESS and CREMA-D speech corpora. Through the deployment of model averaging ensembles, the maximum classification accuracy of all experiments conducted on the RAVDESS and CREMA-D speech

corpora is 85% and 66%, respectively. This is representative of a considerable improvement over comparable unimodal state-of-the-art methods implemented on the RAVDESS speech corpus exhibiting a minimum of a 5% improvement in classification accuracy over such methods. The method proposed in this work exhibits a 66% classification accuracy when deployed on the CREMA-D speech corpus, with and without data augmentation methods, which is similar to multimodal methods and slightly improved on unimodal methods that are currently found in literature.

VII. Conclusion

In this thesis, the applications of transfer learning, raw audio data augmentation and image enhancement techniques with respect to speech emotion recognition (SER) via image classification of spectrograms were explored in great detail. The objective of the research was to explore a variety of new techniques in the hope of producing a machine learning model that yielded improved performance over the current state-of-the-art implementations. The research conducted in this work leveraged Python with an emphasis on the TensorFlow, Librosa and Sci-Kit Learn libraries.

Five transfer learning models pre-trained on the ImageNet dataset were utilized with both the feature extraction and fine-tuning methods of transfer learning. These models include the VGG-16, VGG-19, InceptionV3, Xception and ResNet-50 models. A modified version of the AlexNet deep neural network (DNN) was implemented to serve as a baseline of performance for non-pre-trained deep neural networks. Three image enhancement techniques were used. A variety of raw audio data augmentation techniques were implemented. The datasets used to train these models include the Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS) and the Crowd-sourced Emotional Multimodal Actors Dataset (CREMA-D) speech corpora in both their original and augmented versions. The RAVDESS datasets were utilized for initial model and methodology development and the CREMA-D dataset was used to test the robustness of the proposed methodologies.

In total, 180 experiments were conducted in this research using various combinations of the proposed techniques. The performance of the models generated through these experiments were compared. It was observed that the models trained on the augmented version of both speech

corpora yielded the highest performance. Models utilizing the fine-tuning method of transfer learning yielded higher performance compared to those utilizing the feature extraction method. The AlexNet DNN models yielded relatively similar performance as the transfer learning models utilizing fine-tuning. It was also observed that image enhancement techniques provide greater benefit to deeper transfer learning models including InceptionV3, Xception and ResNet-50.

Through the various experiments and analysis a variety of methods and techniques were discovered that offer either improved or very similar performance to what is currently considered to be state-of-the-art with respect to speech emotion recognition. In particular, utilizing a VGG-16 deep convolutional neural network pre-trained on the ImageNet dataset in conjunction with various data augmentation techniques yielded a minimum of a four percent improvement in classification accuracy on the RAVDESS speech corpus and comparable performance on the CREMA-D speech corpus to what is currently considered to be state-of-the-art in speech emotion recognition. Model averaging ensembles exhibit even further improvement over comparable state-of-the-art methods when deployed on the RAVDESS speech corpus exhibiting a minimum of a 5% improvement in classification accuracy over such state-of-the-art methods, yielding 85% classification accuracy. The ensemble learning method proposed in this work exhibits a 66% classification accuracy when deployed on the CREMA-D speech corpus, with and without data augmentation methods, which is similar or slightly improved compared to what is found throughout literature for this particular speech corpus.

VIII. Future Work

With regards to continuing the research performed within this thesis, the ultimate goal is to develop a real-time speech emotion recognition system. As such, the next area to explore would be in determining the viability of a *weighted model averaging ensemble*. A weighted model averaging ensemble classifier builds upon a model averaging ensemble by allowing the contribution of each ensemble member to a prediction to be weighted proportionally to the respective trust or estimated performance of the given member. This can typically yield slightly improved classification performance upon implementation. A *stacked generalization ensemble* would also be an interesting method to investigate as well. Stacked generalization ensembles take a weighted average ensemble a step further by training an entirely new model that learns how to best combine the contributions from each member of the ensemble. This is commonly referred to as *stacking* and can result in better predictive performance than any single contributing model. In general, weighted average ensembles and stacking ensembles are more expensive from a computational resource perspective in implementation as finding the best set of weights for the ensemble in a weighted average ensemble is accomplished through grid-search or direct optimization. Similarly, a stacking ensemble requires training a new model based on the models within the ensemble and can be quite time-consuming when utilizing deep neural networks.

To follow the trend of increasing the amount of training data through data augmentation and engineering it would also be of interest to attempt to combine the RAVDESS and CREMA-D datasets through the use of *oversampling* and *undersampling*. Oversampling is used to duplicate samples from a minority class. Conversely, undersampling serves to delete samples

from a majority class. Given that the RAVDESS and CREMA-D dataset do not contain the same emotional classes it would be necessary to perform oversampling and undersampling to generate one balanced dataset. It has also been shown in literature that it may be possible to combine certain emotional classes which could aid in reducing the amount of oversampling and undersampling. This would serve to improve the generalizability of any model trained on the combined dataset and may offer an increase in classification performance given the additional amount of training data and diversity.

Lastly, with respect to developing a real-time speech emotion recognition (SER) system, a process would need to be developed for efficiently making predictions. The potential pseudo-code for such a system is given below:

Run loop indefinitely until user interrupts process either via keyboard or mouse:

1) *Record microphone audio for three seconds*
2) *Convert audio to log-scaled mel spectrogram*
3) *Standardize and scale the spectrogram image*
4) *Input image (spectrogram) to model for prediction*
5) *Retrieve prediction label for the given image (spectrogram)*
6) *Display prediction to the screen via text*
7) *Repeat*

The model utilized by the system could potentially be one of the ensemble classifiers previously mentioned, or a new model trained on the dataset generated from the combined RAVDESS and CREMA-D speech corpora. The SER system could be built around a mobile device or desktop application which would allow for it to be deployed in a number of different environments.

Overall however, there exist many different applications and use-cases for a SER system ranging from speech language pathology to customer call centers.

REFERENCES

[1] N. H. Frijda, *The Emotions*, Cambridge, England, UK: CUP, 1986.

[2] O. Korn, L. Stamm, and G. Moeckel, "Designing authentic emotions for non-human characters. A study evaluating virtual affective behavior", *Designing Interactive Systems*, pp. 477-487, Jun 2017.

[3] A. Mehrabian, *Silent Messages*, Belmont, CA, USA: Wadsworth Pub. Co, 1971.

[4] P. Ekman and W. V. Friesen, "Constants across cultures in the face and emotion", *Journal of Personality and Social Psychology*, vol. 17, no. 2, pp. 124–129, Feb 1971.

[5] B. Stephenson, "4 valuable benefits of speech therapy after intubation," *Short & Long Term Rehab Blog*, 15-Apr-2020. [Online]. Available: http://blog.rehabselect.net/4-valuable-benefits-of-speech-therapy-after-intubation. [Accessed: 8-August-2021].

[6] M. Lech, M. Stolar, C. Best, and R. Bolia, "Real-time speech emotion recognition using a pre-trained image classification network: Effects of bandwidth reduction and companding," *Frontiers in Computer Science*, vol. 2, 2020.

[7] G. Shanmugasundaram, S. Yazhini, E. Hemapratha, and S. Nithya, "A comprehensive review on stress detection techniques," *International Conference on System, Computation, Automation and Networking*, pp. 1-6, Mar 2019.

[8] S. Sadhu, R. Li, and H. Hermansky, "M-vectors: sub-band based energy modulation features for multi-stream automatic speech recognition," *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6545- 6549, May 2019.

[9] W. Liu et al., "State-time-alignment phone clustering based language-independent phone recognizer front-end for phonotactic language recognition," *14th Int. Conf. on Computer Science & Education*, pp. 863-867, Aug 2019.

[10] X. Huahu, G. Jue, and Y. Jian, "Application of speech emotion recognition in intelligent household robot," *International Conference on Artificial Intelligence and Computational Intelligence*, vol. 1, pp. 537-541, 2010.

[11] N. Kurpukdee, S. Kasuriya, V. Chunwijitra, C. Wutiwiwatchai and P. Lamsrichan, "A study of support vector machines for emotional speech recognition," *2017 8th International Conference of Information and Communication Technology for Embedded Systems (IC-ICTES)*, pp. 1-6, 2017.

[12] E. Ghaleb, M. Popa, and S. Asteriadis, "Multimodal and temporal perception of audio-visual cues for emotion recognition," *2019 8th International Conference on Affective Computing and Intelligent Interaction (ACII)*, 2019.

[13] E. Ghaleb, M. Popa, and S. Asteriadis, "Metric learning based multimodal audio-visual emotion recognition," *IEEE MultiMedia*, vol. 27, no. 1, pp. 37–48, 2019.

[14] M. Xu, F. Zhang, and W. Zhang, "Head fusion: Improving the accuracy and robustness of speech emotion recognition on the IEMOCAP and Ravdess Dataset," *IEEE Access*, vol. 9, pp. 74539–74549, 2021.

[15] D. Issa, M.F. Demirci, and A.Yazici,"Speech Emotion Recognition With deep convolutional neural networks," *Biomed. Signal Process. Control*, vol. 59, May 2020, Art. no. 101894.

[16] H. Li, W. Ding, Z. Wu, and Z. Liu, "Learning fine-grained multimodal alignment for speech emotion recognition," 2020, *arXiv:2010.12733*. [Online]. Available: http://arxiv.org/abs/2010.12733

[17] Mustaqeem and S. Kwon, "A CNN-assisted enhanced audio signal processing for speech emotion recognition," *Sensors*, vol. 20, no. 1, p. 183, 2019.

[18] Y. Zeng, H. Mao, D. Peng, and Z. Yi, "Spectrogram based multi-task audio classification," *Multimedia Tools and Applications*, vol. 78, no. 3, pp. 3705–3722, 2019.

[19] N. Mitschke, Y. Ji, and M. Heizmann, "Task specific image enhancement for improving the accuracy of cnns," *Proceedings of the 10th International Conference on Pattern Recognition Applications and Methods*, 2021.

[20] Hernández-García, A. and König, P. Data Augmentation Instead of Explicit Regularization. *arXiv preprint arXiv:1806.03852*. [Online]. Available: https://arxiv.org/abs/1806.03852

[21] Y. Cheng, J. Yan, and Z. Wang, "Enhancement of weakly illuminated images by deep fusion networks," *2019 IEEE International Conference on Image Processing (ICIP)*, 2019.

[22] S. Wei, S. Zou, F. Liao, and weimin lang, “A comparison on data augmentation methods based on Deep Learning for Audio Classification,” *Journal of Physics: Conference Series*, vol. 1453, p. 012085, 2020.

[23] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, “A survey on Deep Transfer Learning,” *Artificial Neural Networks and Machine Learning – ICANN 2018*, pp. 270–279, 2018.

[24] R. Liu, Y. Shi, C. Ji, and M. Jia, “A survey of sentiment analysis based on Transfer learning,” *IEEE Access*, vol. 7, pp. 85401–85412, 2019.

[25] “Digital Audio Basics: Sample Rate and Bit Depth,” [Online] Available: https://www.izotope.com/en/learn/digital-audio-basics-sample-rate-and-bit- depth.html. [Accessed: 1-Oct-2021].

[26] “Audio bit depth,” *Wikipedia*, 05-Oct-2021. [Online]. Available: https://en.wikipedia.org/wiki/Audio_bit_depth. [Accessed: 10-Oct-2021].

[27] A. Subramanian, “Fundamentals of Signal Processing,” *Data Science Blog*, 17-Jul-2021. [Online]. Available: http://blog.dominodatalab.com/fundamentals-of-signal-processing. [Accessed: 11-Oct-2021].

[28] T. Bäckström, “Windowing,” *Aalto University Wiki*, 06-Aug-2019. [Online]. Available: https://wiki.aalto.fi/display/ITSP/Windowing. [Accessed: 11-Oct-2021].

[29] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[30] S. Davis and P. Mermelstein, “Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences,” *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 28, no. 4, pp. 357–366, 1980.

[31] J. Lyons, “Mel Frequency Cepstral Coefficient (MFCC) tutorial,” *Practical Cryptography*. [Online]. Available: http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/. [Accessed: 11-Oct-2021].

[32] S. R. Livingstone and F. A. Russo, “The Ryerson Audio-Visual Database of emotional speech and Song (RAVDESS): A Dynamic, multimodal set of facial and vocal expressions in North American English,” PLOS ONE, vol. 13, no. 5, 2018.

[33] “The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS),” [Online] Available: https://zenodo.org/record/1188976. [Accessed: 11-Aug-2021].

[34] “CREMA-D (Crowd-sourced Emotional Mutimodal Actors Dataset),” [Online] Available: https://github.com/CheyneyComputerScience/CREMA-D. [Accessed: 14-Oct-2021].

[35] Librosa development team. (2019) LibROSA. https://librosa.github.io/librosa/.

[36] “Fundamental Frequency, Pitch, F0,” [Online] Available: https://link.springer.com/referenceworkentry/10.1007%2F978-0-387-73003-5_775. [Accessed: 12-Oct-2021].

[37] A. Géron, Hands-on machine learning with scikit-learn and tensorflow: Concepts, tools, and techniques to build Intelligent Systems. Sebastopol, CA: O'Reilly Media, 2019.

[38] “Artificial neuron,” [Online] Available: https://en.wikipedia.org/wiki/Artificial_neuron. [Accessed: 1-Nov-2021].

[39] G. Hinton, N. Srivastava, K. Swersky, and T. Tieleman, “Neural Networks for Machine Learning.”

[40] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *Proc. Int. Conf. Learn. Representations*, 2015.

[41] S. Albelwi and A. Mahmood, “A framework for designing the architectures of deep convolutional Neural Networks,” *Entropy*, vol. 19, no. 6, p. 242, 2017.

[42] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, “Improving neural networks by preventing co-adaptation of feature detectors,” *CoRR*, vol. abs/1207.0580, 2012.

[43] D.-ho Lee, Y. Lee, and B.-seok Shin, “Mid-level feature extractor for transfer learning to small-scale dataset of Medical Images,” *Advances in Computer Science and Ubiquitous Computing*, pp. 8–13, 2019.

[44] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.

[45] G. A. Shadeed, M. A. Tawfeeq, and S. M. Mahmoud, “Automatic Medical Images segmentation based on Deep Learning Networks,” *IOP Conference Series: Materials Science and Engineering*, vol. 870, p. 012117, 2020.

[46] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[47] S.-H. Tsang, “Review: Inception-V3 - 1st runner up (image classification) in ILSVRC 2015,” *Medium*, 23-Mar-2019. [Online]. Available: https://sh-tsang.medium.com/review-inception-v3-1st-runner-up-image-classification-in-ilsvrc-2015-17915421f77c. [Accessed: 01-Nov-2021].

[48] F. Chollet, “Xception: Deep learning with depthwise separable convolutions,” *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[49] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[50] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097--1105.

[51] Hossin, “A novel performance metric for building an optimized classifier,” *Journal of Computer Science*, vol. 7, no. 4, pp. 582–590, 2011.

[52] H. M. Bui, M. Lech, E. Cheng, K. Neville, and I. S. Burnett, “Object recognition using deep convolutional features transformed by a recursive network structure,” *IEEE Access*, vol. 4, pp. 10059–10066, 2016.

[53] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, “Imagenet Large Scale Visual Recognition Challenge,” *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[54] “Keras Documentation: Keras applications,” *Keras*. [Online]. Available: https://keras.io/api/applications/. [Accessed: 10-Nov-2021].

[55] R. C. Gonzalez, S. L. Eddins, and R. E. Woods, *Digital Image Processing using MATLAB*. Knoxville, TN: Gatesmark Publishing, 2009.

[56] P. Nakkiran, “Learning rate annealing can provably help generalization, even for convex problems,” *arXiv preprint arXiv:2005.07360*, 2020.

[57] K. Sato, “What makes tpus fine-tuned for deep learning? | google cloud blog,” *Google*, 30-Aug-2018. [Online]. Available: https://cloud.google.com/blog/products/ai-machine-learning/what-makes-tpus-fine-tuned-for-deep-learning. [Accessed: 06-Nov-2021].