Multi-lingual emotion detection in speech using Hybrid Neural Network Architectures

by

MICHAEL JOHN NORVAL

submitted in accordance with the requirements for

the degree of

DOCTOR OF PHILOSOPHY

in the subject

ENGINEERING

&

at the

UNIVERSITY OF SOUTH AFRICA

SUPERVISOR:

Prof. Z. Wang

14 July 2026

DECLARATION

Name: Michael John Norval

Student number: 36825050

Degree: PHD ENG

Exact wording of the title of the thesis as appearing on the electronic copy submitted for examination:

Multi-lingual emotion detection in speech using Hybrid Neural Network Architectures

I declare that the above thesis is my own work and that all the sources that I have used or quoted have been indicated and acknowledged by means of complete references.

I further declare that I submitted the thesis to originality checking software and that it falls within the accepted requirements for originality.

I further declare that I have not previously submitted this work, or part of it, for examination at Unisa for another qualification or at any other higher education institution.

(The thesis will not be examined unless this statement has been included.)

14 July 2026

SIGNATURE DATE

LIST OF PUBLICATIONS

1. Norval, M. and Wang, Z. (2024) ‘Speech Emotion Recognition using Hybrid Architectures’, International Journal of Computing, 23(1), pp. 1–10. doi: 10.47839/ijc.23.1.3430
2. Norval, M. and Wang, Z. (2025) ‘Explainable Artificial Intelligence Techniques for Speech Emotion Recognition: A Focus on XAI Models’, Inteligencia Artificial, 28(76), pp. 85–123. doi: 10.4114/intartif.vol28iss76pp85-123
3. Norval, M. and Wang, Z. (2025) ‘Quantum AI in Speech Emotion Recognition’, Entropy, 27(12), 1201. doi: 10.3390/e27121201
4. Norval, M., Wang, Z. and Sun, Y. (2024) ‘Synthetic Speech Data Generation Using Generative Adversarial Networks’, in Signals and Communication Technology. Springer, pp. 117–126. doi: 10.1007/978-3-031-47100-1_11
5. Norval, M. and Wang, Z. (2022) ‘Creation of an Afrikaans Speech Corpora for Speech Emotion Recognition’, in 2022 2nd International Conference on Robotics, Automation and Artificial Intelligence (RAAI). IEEE. doi: 10.1109/RAAI56146.2022.10092988
6. Norval, M. and Wang, Z. (2024) ‘Hybrid Architecture and pre-processing for Speech Emotion Recognition’, in 2024 International Conference on Advanced Mechatronic Systems (ICAMechS). IEEE, pp. 31–36. doi: 10.1109/ICAMechS63130.2024.10818817

ABSTRACT

The advancement of Deep Learning has revolutionised numerous fields, including human-computer interaction (HCI) and emotion detection. In speech recognition, voice assistants such as Apple's Siri, Microsoft's Cortana, and Google's Assistant have become integral to everyday technology use. However, current emotion detection systems often need more support for African languages, presenting a significant research gap. The aim and objectives of the study are to develop a Speech Emotion Recognition (SER) system tailored to South African languages, with a specific focus on Afrikaans. The primary objective is to compile an Afrikaans speech corpus for emotion recognition. Furthermore, it explores hybrid neural network architectures combining CNN, Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks for improved accuracy. Finally, the aim is to investigate the optimisation of SER models for multilingual cross-language support.

This study adopts a pragmatic research paradigm, utilising the Design Science Research Framework (DSR). Data is collected quantitatively. The literature review covers human speech, speech physiology, neural network architectures, speech corpus, and data extraction methods. Hybrid architectures and preprocessing techniques are examined to identify the most effective configurations. The Afrikaans speech corpus is sourced from Creative Commons (CC) sources, with data augmentation using Generative Adversarial Networks (GANs).

The study successfully compiled an Afrikaans speech corpus. Synthetic speech samples were created using frameworks such as Tacotron 2 and the WaveNet vocoder, thereby enhancing sound quality. For the Hybrid Neural Network Architectures: The novel Dendritic Convolutional Long Short-Term Memory (DCLSTM) architecture outperformed traditional CNN and LSTM models, effectively capturing nuanced and long-term emotional dependencies in audio data. The second Kalman filter variant was identified as the optimal preprocessing technique for noisy speech signals. A final accuracy of 78.50% was achieved, improving on baseline CNN and LSTM models. For the Multilingual SER System, the DenCaps model achieved up to 71.91% accuracy on multilingual datasets, demonstrating improved generalisation and better capture of temporal dynamics and spatial hierarchies in emotional speech.

The findings highlight the effectiveness of hybrid neural network architectures and advanced preprocessing techniques in improving SER accuracy. The development of the Afrikaans speech corpus and the introduction of the DenCaps model represent significant advancements in emotion detection. These results underscore the potential for creating robust, multilingual SER systems that cater to diverse linguistic needs. The research objectives were successfully achieved through the development of a specialised Afrikaans speech corpus, the exploration of innovative neural network architectures, and the optimisation of models for multilingual support. The academic contributions pave the way for more accurate, contextually aware human-computer interaction systems that address the research gap in emotion detection for African languages.

Keywords: Afrikaans, Capsule Networks, Dendritic, Hybrid, Kalman Filter, Multilingual, SER

DEDICATION

I dedicate this work to my wife, Lizelle, and our daughter, Karli. I also want to thank my brother and parents for their ongoing support.

ACKNOWLEDGEMENT

Years of collaboration, learning, research, and discovery are represented in this PhD thesis. The thesis would not have been possible without the support of several institutions and people. I received significant support and assistance from the following individuals and institutions:

1. My whole family.
2. My supervisor, Prof. Zenghui Wang, for his extraordinary support throughout my PhD study and his patience and knowledge while allowing me to extend my scientific research abilities. Thank you very much!
3. The University of South Africa.
4. Institutions that shared their audio
   i. Creative Commons audio sources
   ii. Japanese National Institute of Informatics - Speech Resource Consortium

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AF	Activation Functions
AI	Artificial Intelligence
API	Application Programming Interface
ASR	Automatic Speech Recognition
CEC	Constant Error Carousels
CNN	Convolutional Neural Network
DSP	Digital Signal Processing
DSR	Design Science research
FIR	Finite Impulse Response
GAN	Generative Adversarial Networks
GMM	Gaussian Mixture Models
HCI	Human-Computer Interaction
HMM	Hidden Markov Models
IIR	Infinite Impulse Response
IoT	Internet Of Things
IRB	Institutional Review Board
JPEG	Joint Photographic Experts Group
KNN	K-Nearest Neighbours
LMS	Least Mean Squares
LMT	Logistical Model Tree
LPC	Linear Prediction Coefficient
LPCC	Linear Prediction Cepstral Coefficients
LSF	Line Spectral Frequencies
LSTM	Long Short-Term Memory
MEDC	Mel Energy-spectrum Dynamic Coefficients
MFCC	Mel Frequency Cepstral Coefficient
ML	Machine Learning
MPEG	Moving Pictures Expert Group
NLP	Natural Language Processing
ONNX	Open Neural Network Exchange
PCC	Prediction Cepstral Coefficient
PLP	Perceptual Linear Prediction coefficients
ReLU	Rectified linear unit
RNN	Recurrent Neural Networks
SDROM	Signal Dependent Rank Order Mean Algorithm
SER	Speech Emotion Recognition
SMO	Sequential Minimal Optimisation
SR	Speech Recognition
SVM	Support Vector Machines
VR	Virtual Reality

CHAPTER 1: Introduction

1.1 Background

Human–Computer Interaction (HCI) has undergone significant evolution over the past decade, moving beyond traditional input methods such as the mouse and keyboard toward more natural and intuitive interfaces. Emotion can be detected through multiple channels, including facial expressions, gestures, physiological signals, and speech (Papakostas, et al., 2017). Among these modalities, audio is considered the least intrusive, as it does not require specialised hardware or direct user interaction. The research objectives were achieved through the development of a specialised Afrikaans speech corpus, the exploration of innovative neural network architectures, and the optimisation of models for multilingual support.

Modern smart devices and home automation systems, such as Amazon Alexa, Google Home, and Microsoft Cortana, rely heavily on speech recognition technologies. These systems are widely deployed in practical applications, including customer service automation, healthcare monitoring, education, and intelligent personal assistants. However, despite their widespread adoption, current systems primarily process linguistic content and largely ignore the emotional context of speech.

This limitation represents a significant drawback in existing systems, as the absence of emotional awareness leads to interactions that are often perceived as mechanical, insensitive, or contextually inappropriate. In applications such as call centres, mental health monitoring, and assistive technologies, the inability to recognise and respond to emotional cues can reduce system effectiveness and user satisfaction.

Integrating emotion recognition into speech-based systems has the potential to address these shortcomings by enabling more adaptive, empathetic, and context-aware interactions. Speech Emotion Recognition (SER) is pivotal in enhancing interactions between humans and machines, especially in multilingual environments and for languages that are often underrepresented, such as Afrikaans. By accurately detecting emotional cues in speech, SER systems can improve the user experience, enabling more nuanced understanding and responses in automated communication settings. This is particularly important for languages and dialects that may lack extensive training data, presenting unique challenges and opportunities for the development of robust SER algorithms.

Prominent commercial speech recognition systems include Apple Siri, Google Assistant, Amazon Alexa, Microsoft Cortana, and IBM Watson Speech-to-Text. These systems are widely used across domains such as virtual assistants, smart home environments, customer service automation, and healthcare applications. While they demonstrate high accuracy in speech-to-text conversion, their primary focus remains on linguistic content, with limited capability to detect and interpret emotional context in speech.

1.2 Human Emotion

Communication is integral to daily human life, with emotion serving as a critical component. Conveyed through both visual and auditory channels, emotion underpins effective interaction; its absence risks a breakdown in communication. The significance extends to human-computer interfaces, as illustrated by a humanoid robot expressing emotions in Figure 1.1.

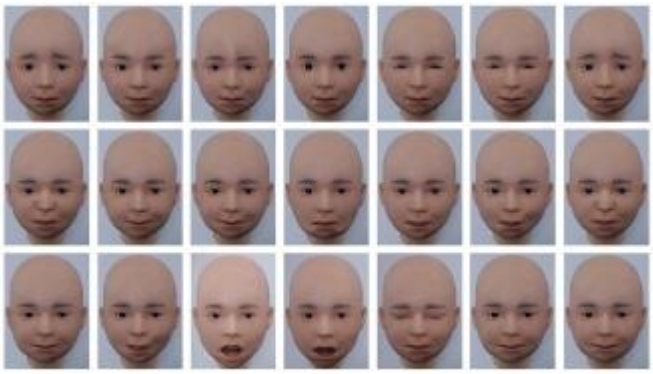

Figure 1.1: Android expressing emotion *(Houser, 2022)*

1.3 Deep Learning and Emotion Detection

Handcrafting features like statistics of short-term feature vectors can achieve SER (Papakostas, et al., 2017; Zhao, Li, Zeng, Wang, & Zhang, 2022). Using Deep Learning to process low-level features, such as energy, pitch, and Linear Prediction Cepstrum Coefficients (LPCC) and Mel Frequency Cepstrum Coefficients (MFCC), is much more effective. CNN methods are superior to handcrafted features. A graphical representation of a sample MFCC is illustrated in Figure 1.2., which illustrates how speech energy is distributed across frequency bands over time, providing a compact representation of temporal and spectral characteristics relevant to emotion recognition.

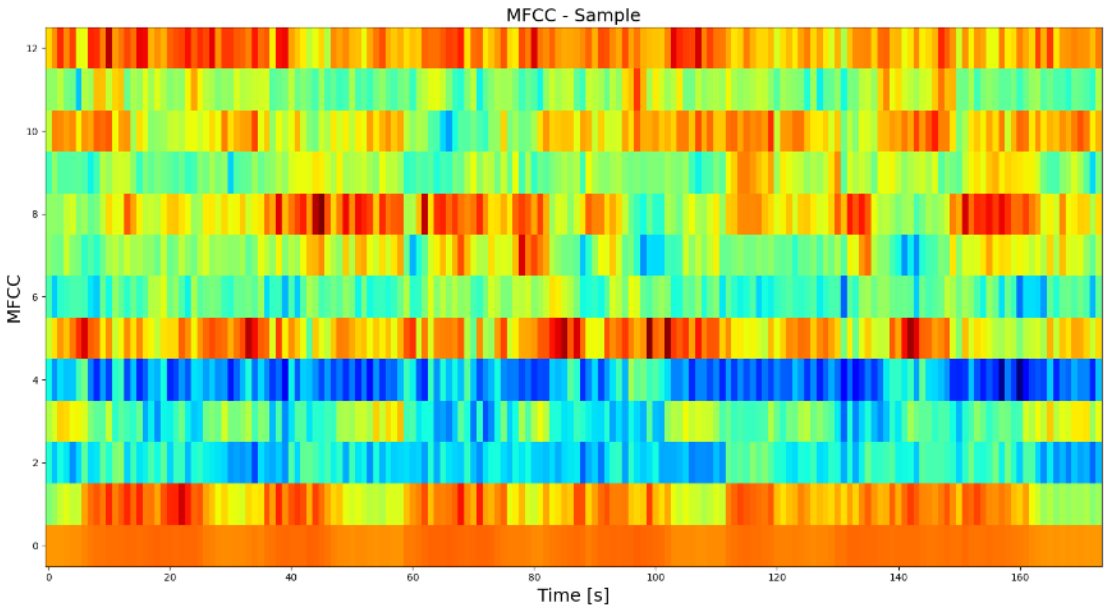

Figure 1.2: Sample MFCC (*Generated using Python*).

1.4 Rationale

SER will play a critical role in next-generation HCI devices. It will enhance and augment every industry in which it is used. SER is very effective because it utilises multimodal methods, including video, wearable fitness devices, and audio. However, regarding privacy concerns, speech is far less intrusive. Regarding data that can train the network, the most common feature is the Mel-frequency Cepstrum Coefficients (MFCC). Other feature extraction methods like Linear Predictor Coefficients (LPC), Mel Energy-spectrum Dynamic Coefficients (MEDC), Linear Predictor Coefficients (LPC) and Perceptual Linear Prediction cepstrum coefficients (PLP) are used (Khalil, et al., 2019; Casale, Russo, G. Scebba, & Serrano, 2008; Kilimci, Bayraktar, & Küçükmanisa, 2025).

Despite its promise, SER faces several research challenges:

i. South Africa has eleven official languages. One of these languages, namely Sepedi, has been researched, and a corpus created (Manamela, Manamela, Modipa, Joseph, & Billson, 2018). However, one of the other eleven languages, Afrikaans, has not been researched, and no SER corpus exists.
ii. Limitations exist in current SER training methods. The limitations are reduced efficiency for temporally varying input data, overlearning and large-layer-internal-architecture limitations (Khalil, et al., 2019; George & P. Muhamed Ilyas, 2024; Barhoumi & BenAyed, 2025)
iii. SER systems are trained and respond well to a specific language and dialect. However, multilingual systems are less accurate and challenging (Zehra, Javed, Jalil, Gadekallu, & Kahn, 2021; Shakil, et al., 2025; Adeel, Tao, Jin, Guo, & Alsuhaibani, 2026). A model trained on a single-language corpus does not achieve the same accuracy when tested on different languages.

There are limited African and South African SER training data available. By having more of these languages sampled, the way is paved for more research. A typical corpus is created using elicited emotional, actor-based, or natural speech (Basu, Chakraborty, Bag, & Aftabuddin, 2017). Collected samples will be verified and corrected in terms of amplitude, sampling rate and length.

Research shows that hybrid models, particularly in the context of speech, outperform conventional feed-forward networks. Hybrid CNN and RNN-LSTM architectures are researched (Shi, Wang, Wang, Liu, & Yan, 2019). Pre-processing research addresses several shortcomings. Some examples of pre-processing are silence removal, normalisation, noise reduction, spectral subtraction, and windowing (Nema & Abdul-Kareem, 2017; Basu, Chakraborty, Bag, & Aftabuddin, 2017).

Cross-language multi-corpus models are researched and optimised. The optimisation will be accomplished by looking at the prosodic and spectral features of audio (Naderi & Nasersharif, 2023).

Emotion has been researched since the 1800s (Thanapattheerakul, Mao, Amoranto, & Chan, 2018). Despite extensive research on emotion, its exact definition remains debatable. Emotion plays a critical role in human communication, and without it, humans would not be able to communicate correctly.

The following commercial sectors can significantly benefit from SER: Call centres, education and gaming. Commercial products exist, but there is much room for growth. It was found that anger was the easiest of the emotions to detect (Petrushin, 2000).

In education and e-learning, the curriculum is adjusted accordingly to keep the focus of the student (Chen, Yue, Yu, Shen, & Zhu, 2007). In VR multiverse games, a built-in engine detects the player's emotions. The gameplay is adjusted according to the detected emotion to engage the player more. Affect Aware Video Games is the term used for such a gaming system (Szwoch & Szwoch, 2015).

1.4.1 Problem Statement and Research Gap

Problem Statement.

Despite recent advancements, several limitations remain in current SER research.

Speech Emotion Recognition (SER) systems are trained on high-resource languages (e.g. English, German, Mandarin) and perform poorly in multilingual and under-resourced settings. In South Africa specifically, no Afrikaans emotional speech corpus exists, and state-of-the-art monolingual models degrade sharply when applied cross-lingually. Conventional CNN and LSTM architectures further struggle to jointly capture the spatial (spectral) and long-range temporal dependencies that characterise emotional speech, limiting accuracy and generalisation.

Research Gap.

From the preceding review, the following gaps are identified and directly addressed by:

- No publicly available Afrikaans speech corpus annotated for emotion recognition exists, despite Afrikaans being one of South Africa's eleven official languages.
- Existing SER architectures (standalone CNN, LSTM, or simple CLSTM fusions) inadequately model the combined spatial-temporal and hierarchical structure of emotional speech; dendritic and capsule-based hybrids remain under-explored in SER.
- Multilingual SER models suffer significant accuracy degradation across languages; optimisation strategies that leverage prosodic and spectral features for cross-lingual generalisation are not systematically benchmarked.
- Pre-processing techniques for noisy emotional speech (particularly Kalman-filter variants) have not been comparatively evaluated as part of an end-to-end hybrid SER pipeline.

This thesis addresses these gaps by (i) compiling a novel Afrikaans SER corpus, (ii) proposing the DCLSTM and DenCaps hybrid architectures, and (iii) benchmarking them against standard baselines on monolingual and multilingual configurations.

1.5 General Aim and Objectives

1.5.1 Aim of the study

The primary aim is to improve SER accuracy for multi-corpus systems and to incorporate the Afrikaans speech corpus into a novel hybrid neural architecture—the Dendritic Convolutional Long Short-Term Memory (DCLSTM) network, together with its Dendritic-Capsule variant (DenCaps).

1.5.2 Objectives

- **Objective 1: Compiling a South African-based Afrikaans language corpus**

Major languages have been used to create a speech corpus. The following are examples of a corpus: EMO-DB (German), SAVEE (English), EMOVO (Italian), MASC (Chinese) and IEMOCAP (English). Lesser-known, smaller databases also exist (Goel & Beigi, 2020).
This thesis aims to compile a South African-based Afrikaans language corpus. Speech clips can be collected by extracting or creating elicited emotional, actor-based, or natural speech. Podcasts, broadcasts, and Creative Commons material are used to minimise ethical issues. The Afrikaans corpus will be the primary dataset for testing. The database will be augmented by using generative adversarial networks (GAN).

- **Objective 2: Pre-processing and hybrid architectures**

SER models have limitations in their overall architecture. Using proper audio pre-processing techniques is another way to optimise the model and increase accuracy. Multiple pre-processing pipelines — defined as ordered sequences of individual techniques — are systematically evaluated, with each variant benchmarked on SER classification accuracy to identify the optimal configuration. The pipelines evaluated include two Kalman filter variants (Kalman_1 and Kalman_2), spectral gating, spectral subtraction, FIR and IIR filtering, pre-emphasis filtering, framing, windowing, and silence removal (Nema & Abdul-Kareem, 2017). From an architectural perspective, hybrid CNN-RNN models are evaluated. Good results have been obtained with hybrid models used for other deep-learning problems (Muaidi, Al-Ahmad, Khdoor, Alqrainy, & Alkoffash, 2014; Shi, Wang, Wang, Liu, & Yan, 2019). The performance of a hybrid architecture is evaluated, together with various pre-processing techniques.

- **Objective 3: Combined Datasets and Multilingual Models**

NN models created with a single corpus tend to have high accuracy (Li & Akagi, 2019; Tamulevičius, et al., 2020). However, multilingual models incorporating more than one language are still challenging. The thesis proposes enhancing the accuracy of multilingual networks by utilising hybrid models. Specifically, developing and implementing novel algorithms and methods, such as multilingual training and language-agnostic feature extraction techniques, have been shown to enhance the performance of these models significantly. These methodological advancements provide a practical solution to the complexities of multilingual processing, demonstrating how targeted algorithmic innovations can improve model accuracy.

1.6 Research Methodology

Methodologies and research paradigms play a crucial role in any research study. A research paradigm can be described as the collectively held convictions and tacit understandings that members of a scientific community settle upon regarding the proper way to frame and tackle the problems within their field (Kuhn, 1970). The research paradigm that will be used is pragmatic. Pragmatic researchers use a method or combination of strategies that advance explicit research (Wilson, 2014).

1.6.1 Research Data

Qualitative research will be used for the theoretical background research and literature review. Qualitative research enables unstructured, exploratory research. Firstly, research will be done on existing speech corpus databases, identifying the most used, largest and most relevant to the study. A strategy will then be devised to determine the most effective method for compiling an Afrikaans corpus for anger, anticipation, joy, trust, fear, surprise, sadness, and disgust. Secondly, research will be conducted to identify existing pre-processing methods and determine how they can be effectively incorporated into the model. Hybrid NN models will be investigated to find the architecture best suited for the research. Thirdly, a review of the literature on multilingual models will be conducted, informing the architecture best suited for the research.

The experimental part of the research will use quantitative methods. An empirical analysis of current methods will provide critical insight into a better way for pre-processing and ultimately compiling a hybrid NN model. Further study will be conducted on current multilingual models, and subsequently, the model will be adjusted to accommodate the scenario.

1.6.2 Model Development

This thesis depends on the Design Science Research Methodology (DSRM) of (Arsalan, Burhan, Rehman, Umer, & Kim, 2023) as the overarching framework for model development. As illustrated in DSRM, a six-phase process — Identify the Problem and Motivate, Define the Objectives of a Solution, followed by Design and Development, then Demonstration, Evaluation, and Communication — is supported by four possible research entry points and an explicit process-iteration loop. DSRM is purpose-built for research that produces engineered artefacts, which, in this research, comprise the Afrikaans speech corpus (Chapter 3) and the novel DCLSTM and DenCaps neural architectures (Chapters 4 and 5). Each phase is applied iteratively across the experimental chapters, ensuring the model is systematically refined and validated using empirical evidence and logical reasoning, and the process is then applied to all simulations.

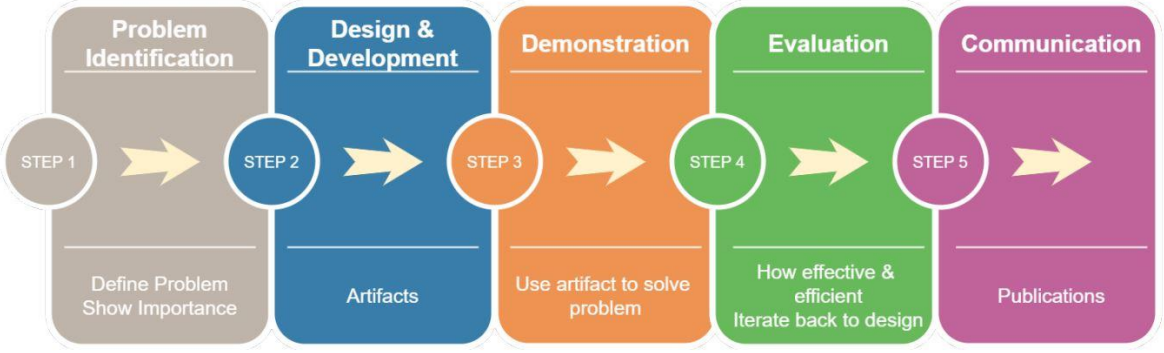

Figure 1.3: The Design Science Research Methodology (DSRM) process model (*Adapted from (Arsalan, Burhan, Rehman, Umer, & Kim, 2023)*)

1.6.2.1 Data collections

The data used in the research consists of two components: an Afrikaans dataset and a multilingual dataset. The Afrikaans dataset was compiled exclusively from open-source Creative Commons audio sources, as detailed in Chapter 3. Audio was collected from publicly available speech material (e.g., podcasts and recordings) and segmented into short utterances (2–5 seconds).

All clips were manually verified and processed using standard audio pre-processing techniques, including resampling, noise reduction, and silence removal. The dataset was organised according to Plutchik's Wheel of Emotions, comprising eight emotion classes: anger, anticipation, disgust, fear, joy, sadness, surprise, and trust. The dataset was balanced as far as possible, with multiple samples per emotion class to ensure adequate representation for training.

The multilingual dataset was compiled from established corpora, including EmoDB and other benchmark datasets. In total, approximately 23,000 samples across 7 languages and 8 emotion classes were used.

All audio samples were transformed into acoustic features (MFCCs, chroma, tonnetz). The combined dataset was split into 80% training and 20% testing sets, with multiple randomised runs to ensure robustness. Ethical requirements, including anonymisation and informed consent (for Afrikaans recordings), were strictly followed.

1.6.2.2 Design Science Research Methodology

The Design Science Research Methodology includes six fundamental phases: identifying the issue and establishing motivation, defining the goals of the solution, designing and creating the solution, demonstrating how it works, assessing its effectiveness, and sharing the results (Arsalan, Burhan, Rehman, Umer, & Kim, 2023). DSRM is purpose-built for research that produces engineered artefacts, and supports iterative refinement through feedback loops between phases. Herewith is a brief description of each phase and how it is applied in the research.

- **Identify Problem and Motivate:** The phase forms the core of the literature review, investigating current Speech Emotion Recognition methods, their limitations for multilingual settings, and the absence of dedicated SER resources for Afrikaans. The motivation is grounded in both academic gaps (low-resource African languages) and practical applications (call centres, healthcare, intelligent assistants).
- **Define Objectives of a Solution:** From the identified gaps, three solution objectives are defined: (i) compile a dedicated Afrikaans speech corpus, (ii) develop hybrid neural network architectures suited to the temporal–spectral nature of emotional speech, and (iii) extend the proposed architectures to multilingual settings.
- **Design and Development:** The phase produces the engineered artefacts of the research: the Afrikaans corpus (Chapter 3), the DCLSTM hybrid architecture (Chapter 4), and the DenCaps architecture (Chapter 5). Audio preprocessing pipelines (Kalman filters, spectral

ating, silence removal) are designed and compared. Initial models are built and tested using MATLAB and Python, with the EmoDB corpus serving as a baseline reference dataset.

- **Demonstration:** Each artefact is demonstrated against suitable problem instances. The Afrikaans corpus is augmented using GANs and Tacotron 2 / WaveNet synthesis. The DCLSTM is demonstrated on the Afrikaans corpus, and the DenCaps architecture is demonstrated on a multilingual mix of seven languages.
- **Evaluation:** The trained models are assessed against the research hypothesis using a consistent set of objective performance metrics applied to held-out test data. Classification performance is measured using accuracy, precision, recall, macro-F1 score, and the confusion matrix across the eight canonical emotion classes (Anger, Anticipation, Disgust, Fear, Joy, Sadness, Surprise, Trust), with macro-F1 given particular weight to account for class imbalance. Model robustness is further evaluated using speaker-independent train/validation/test splits and sensitivity analyses across key hyperparameters to verify that the observed gains are not artefacts of a particular partition or configuration. The hybrid DCLSTM and DenCaps architectures are benchmarked against established baselines (standard CNN, LSTM, and CLSTM models) on both the Afrikaans corpus and the multilingual test conditions. Outcomes feed back into the design and development phase to guide further refinement.
- **Communication:** The findings are disseminated through the seven peer-reviewed publications listed in §1.8 and the present thesis. Communication closes the iteration loop and informs subsequent design refinements.

  The DSRM is applied iteratively rather than as a single linear sequence. The development and analysis of the Afrikaans speech corpus in Chapter 3 constitute the primary Design and Development output for Objective 1. Insights gained from this phase informed the design and demonstration of hybrid neural architectures and preprocessing techniques in Chapter 4 (Objective 2). These refinements were then extended and evaluated in the multilingual context of Chapter 5 (Objective 3). The iterative application within the Design Science Research framework ensures that each chapter builds directly upon the findings of the previous one, transforming the methodological plan into a coherent, executed research process.

1.7 Delineation

Speech, and especially SER, is a very complex topic. A significant amount of time and resources have been devoted to studying the topic. The thesis focuses on creating an Afrikaans corpus, enhancing accuracy through pre-processing and hybrid architectures, and improving multilingual deep learning models. The focus ensures that the research is done within the stipulated timeframe.

1.8 Contributions to Knowledge

i. Creation of Afrikaans speech corpus.
ii. Improving SER accuracy using pre-processing and hybrid models.
iii. Increase multilingual model accuracy.

Parts of the research have been previously published in peer-reviewed journal articles and conference proceedings. The relationship between these publications and the chapters of the thesis is as follows:

• Chapter 3 (Compilation of Afrikaans Speech Corpus):

Based on (Norval & Wang, 2022) and informed by (Norval, Wang, & Sun, Synthetic Speech Data Generation Using Generative Adversarial Networks, 2024), which describes the creation of an Afrikaans speech corpus and the use of generative adversarial networks (GANs) for synthetic speech data augmentation.

• Chapter 4 (Pre-processing and Hybrid Architectures):

Derived from (Norval & Wang, Speech Emotion Recognition using Hybrid Architectures, 2024), which investigates hybrid CNN–LSTM architectures and preprocessing techniques for SER.

• Chapter 5 (Multilingual Detection Systems):

Derived from (Norval & Wang, Hybrid Architecture and pre-processing for Speech Emotion Recognition, 2024), which explores advanced architectures, including capsule-based and dendritic models, to improve multilingual SER performance.

All previously published work has been substantially extended, integrated, and restructured within the thesis to form a coherent and unified contribution.

1.9 Hypothesis

This research hypothesises that hybrid neural architectures combining convolutional and recurrent components — specifically the Dendritic Convolutional Long Short-Term Memory (DCLSTM) and Dendritic Capsule (DenCaps) models — together with advanced audio pre-processing techniques, result in statistically significant improvements in Speech Emotion Recognition (SER) performance for both Afrikaans and multilingual datasets when compared to conventional CNN, LSTM, and CLSTM models.

The hypothesis is evaluated through the following testable propositions:

H1: The inclusion of dendritic layers within hybrid architectures improves classification performance (accuracy and macro F1-score) over standard CLSTM models on the Afrikaans speech corpus.

H2: The application of advanced audio pre-processing techniques improves model robustness and overall classification accuracy.

H3: The proposed hybrid architectures generalise across languages, maintaining strong performance on multilingual datasets (e.g., Afrikaans, English, German).

H4: The fuzzy classification framework provides an improved representation of overlapping

emotional states compared to strict single-label classification.

The **null hypothesis (H_0)** asserts that there is no meaningful difference in performance between the proposed models and the baseline architectures. Statistical significance is determined using repeated experimental runs and hypothesis testing, as presented in Chapters 4 and 5.

1.10 Thesis Structure

The thesis is organised into six Chapters. The key findings of the research are detailed in Chapters three to five. Two of these topics were published as standalone papers, yet both addressed the overall aim of the research.

Chapter 1: Introduction.
Chapter 2: A Literature Review.
Chapter 3: The Compilation of Afrikaans Speech Corpus.
Chapter 4: Investigating Speech Sample Pre-Processing and Hybrid NN Architectures.
Chapter 5: Evaluating Multilingual Detection Systems.
Chapter 6: The Objectives Set Out are Evaluated and Discussed.

CHAPTER 2: Literature Review

2.1 Introduction

Over the last decade, tremendous progress has been made in terms of technology. Significant progress has been made with HCI devices, including home automation systems, mobile phones, gaming consoles, and computers. Nowadays, all HCI devices incorporate speech recognition. Examples include Apple Siri, Amazon Alexa, Google Nest voice assistants, Samsung Bixby, and modern conversational AI systems such as OpenAI ChatGPT voice integration and Google Gemini. A natural evolution for enhanced interaction is the recognition of emotions. Several technology firms and academic institutions are currently researching it. Speech is much less intrusive than, for instance, video recording. Speech emotion recognition has considerable commercial potential. Industries that would benefit from the technology would be call centres and gaming. Emotion is a complex phenomenon and has been studied by numerous researchers. There are eight primary identified emotions: Joy, Sadness, Surprise, Anger, Anticipation, Disgust, Fear and Trust (Carvalhais & Magalhães, 2018; Deschamps-Berger, Lamel, & Devillers, 2021).

In the chapter, human emotion and speech will be reviewed. The primary emotions in the play are dissected and examined. From a speech perspective, all aspects of speech are looked at. Speech processing, speech physiology and speech generation. Following this, digital audio is examined and reviewed. Digital audio makeup and processing are investigated. Following the introduction is a thorough review of artificial intelligence and machine learning. Aspects such as architecture, education, metrics, and validation are examined. Subsequently, the MFCC process and other audio processing techniques, such as Chroma and Tonnetz, are discussed in detail. Finally, a speech corpus and hybrid NN models are reviewed.

2.2 Human Emotion

Emotion is a complex and dynamic phenomenon that has evolved across cultures and civilisations, playing an indispensable role in human communication (Firth-Godbehere, 2023). Without emotion, effective interpersonal interaction would be severely impaired (Laith, Daniel, Kelly, & Buss., 2015; Al-Shawaf & Lewis).

Plutchik's model was adopted in the research as a practical categorical framework for harmonising emotion labels across heterogeneous speech datasets. Its structured representation of core affective states provides a usable basis for mapping labels from datasets that were originally constructed under different annotation schemes. However, the choice also introduces limitations. Plutchik's taxonomy originated within a Western psychological tradition and was not developed specifically for multilingual speech emotion recognition. Its Western-centric theoretical framing reflects assumptions that may not transfer uniformly across linguistic and cultural contexts. Emotional expression and interpretation vary across languages, speaker communities, and recording settings. Consequently, the mappings used in the thesis should be understood as an operational standardisation strategy rather than a claim of universal cross-cultural equivalence across all speech datasets. The limitation is particularly relevant for non-Western datasets such as the Japanese OGVC, the Greek AESDD, and EMOV-DB, where some original labels do not align perfectly with the selected categories. Accordingly, the study

prioritises consistency across speech databases in modelling, while acknowledging that some culturally specific nuance may be lost during label harmonisation. The trailblazing psychological researcher Paul Ekman proposed a foundational set of six core affective states: anger, surprise, disgust, enjoyment, fear, and sadness.

Building on the foundation, Plutchik's Wheel of Emotions expands the framework to eight primary emotions: anger, anticipation, disgust, fear, joy, sadness, surprise, and trust, as illustrated in Figure 2.1. Each of these emotions varies in intensity, with higher-intensity forms such as rage (anger), ecstasy (joy), and terror (fear) represented toward the centre of the wheel. The model also illustrates relationships between emotions, including opposing pairs and combinations (dyads), providing a structured framework for analysing emotional states.

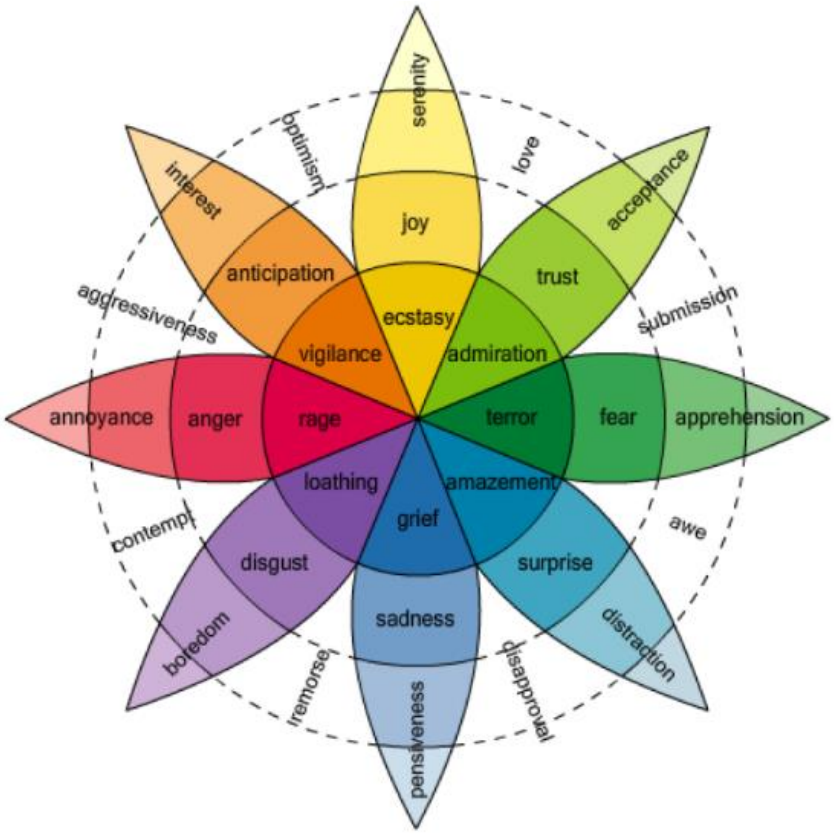

Figure 2.1: Plutchik's wheel of emotions *(Suttles & Ide, 2013)*

Opposing pairs are illustrated in

Table 2.*1*. Emotions are inherently intricate, rarely occurring in isolation, and lack rigid boundaries, as illustrated in Table 2.2 (Mohammad & Turney, 2013).

Table 2.1: Opposing paired Emotions.

Emotion 1	**Emotion 2**
Joy	Sadness
Trust	Disgust
Fear	Anger

Anger	Anticipation

Table 2.2: Primary Dyads

Emotion 1	Emotion 2	Sum / Overlap
Anger	Anticipation	Aggressiveness
Anticipation	Joy	Optimism
Disgust	Anger	Contempt
Fear	Surprise	Awe
Joy	Trust	Love
Sadness	Disgust	Remorse
Surprise	Sadness	Disapproval
Trust	Fear	Submission

2.3 Speech Physiology

Speech production originates from air pressure generated by the lungs, which is modulated and filtered by the vocal tract. On the receiving end, the ear captures these sound waves, transmitting them to the brain for interpretation and understanding.

2.3.1 Human Speech Processing

The decoding of emotional content in speech begins with sensory input routed to the thalamus for initial processing. The information is subsequently relayed to the cortex for advanced auditory and emotional analysis, as illustrated in Figure 2.2.

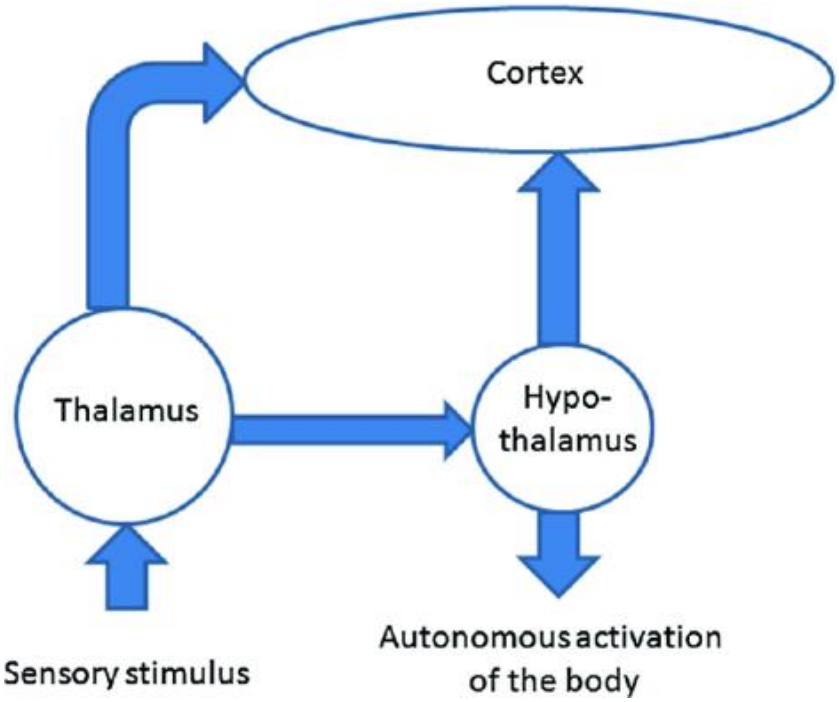

Figure 2.2: Emotion processing in the brain *(Chakravarthy, 2019)*.

2.3.2 Human Speech Generation

Speech emerges from three primary physiological components, depicted in Figure 2.3

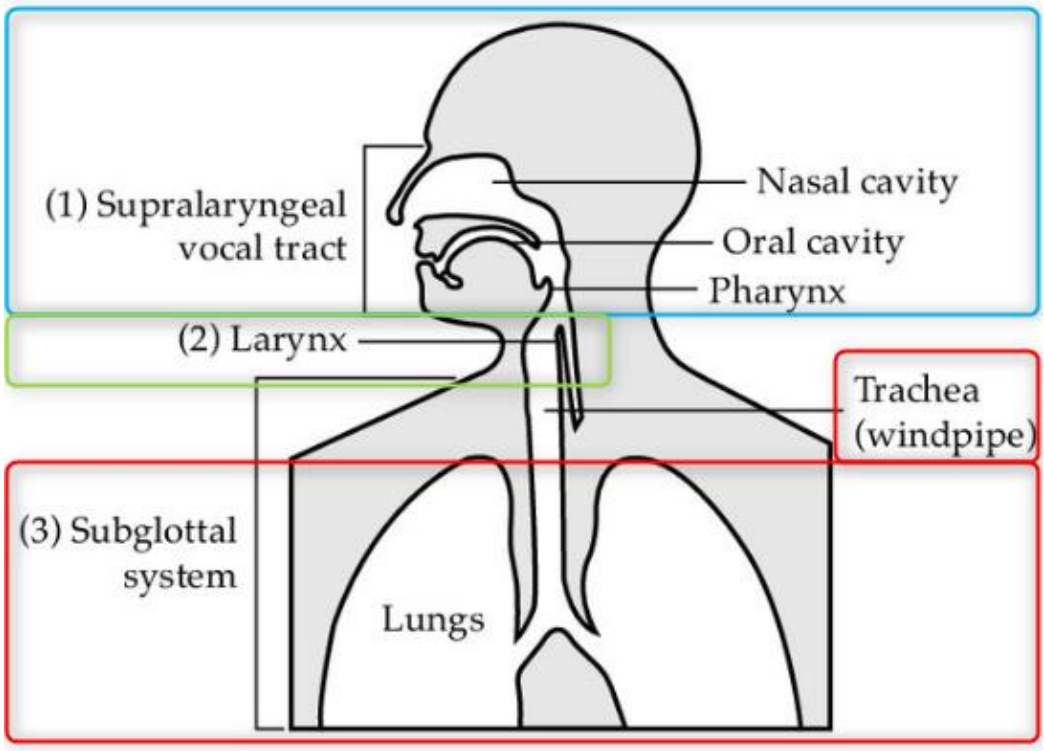

Figure 2.3: Physiologic Components of Human Speech Production (Adapted from *(Lieberman, 1984)*).

The subglottal system (highlighted in red), the larynx (highlighted in green), and the supralaryngeal system (highlighted in blue). Their functions are as follows:

- Subglottal System: The lungs expel air through the trachea, providing the airflow foundation.
- Larynx: The structure converts airflow into periodic puffs, regulated by vocal fold vibration.
- Supralaryngeal System: Acting as an acoustic filter, it shapes sound through the vocal tract's configuration.

Sounds detectable by the human ear generally fall within a span stretching from roughly 20 Hz at the low end up to about 20 kHz at the upper limit, while the bulk of spoken communication tends to occupy a narrower band situated between approximately 100 Hz and 5 kHz, as illustrated in Figure 2.4.

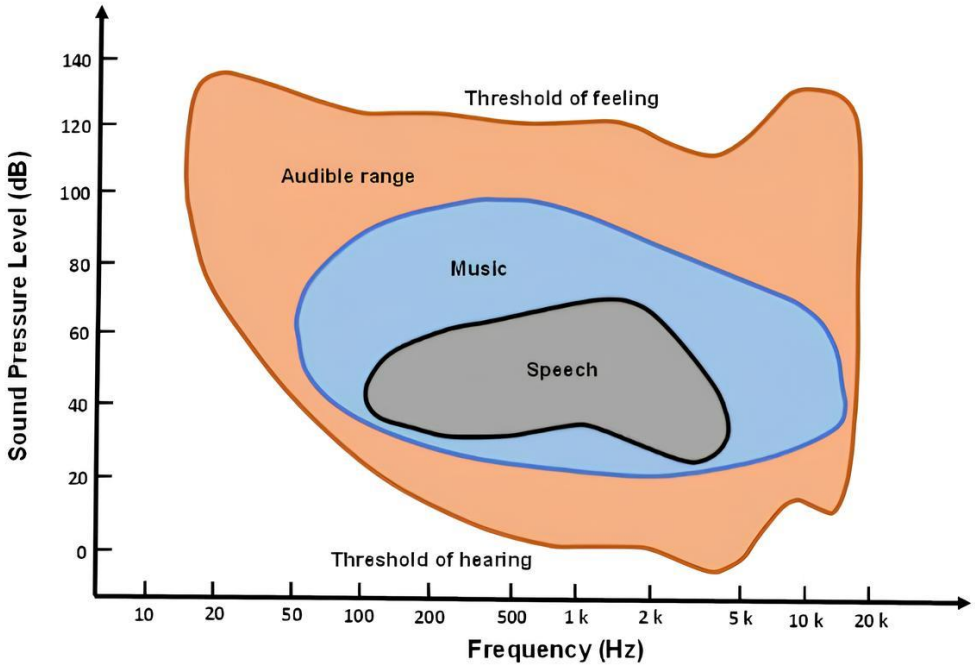

Figure 2.4: Frequency range of human hearing *(Acoustics, 2020-2021)*.

2.3.3 Human Speech Emotion Characteristics

Table 2.3 outlines the acoustic and physiological characteristics associated with specific emotions, focusing on speech rate, pitch, intensity, and related features.

Table 2.3: Voice and Emotion *(Brave & Nass, 2003)*.

Characteristic	Fear	Anger	Sadness	Happiness	Disgust
Speech Rate	Significantly Faster	Slightly Faster	Slightly Slower	Faster or Slower	Significantly Slower
Pitch Average	Significantly Higher	Significantly Higher	Slightly Lower	Significantly Higher	Significantly Lower
Pitch Range	Significantly Wider	Significantly Wider	Slightly Narrower	Wider	Slightly Wider
Intensity	Normal	Higher	Lower	Higher	Lower
Voice Quality	Irregular	Breathy	Resonant	Blaring	Grumbled
Pitch Changes	Normal	Abrupt	Downward Inflections	Upward Inflections	Downward Inflections
Articulation	Precise	Tense	Slurring	Normal	Normal

Speech articulation results from coordinated movements of the jaw, teeth, lips, and tongue. Acoustically, pitch corresponds closely to frequency, while intensity aligns with amplitude. These attributes collectively underpin the expression of emotional nuances in speech.

2.4 Audio Review

2.4.1 Input Formats

The conversion of analogue audio to digital form involves several techniques, governed by two primary parameters: sampling rate and bit depth. The sampling rate, defined as the number of samples captured per second, determines temporal resolution, while bit depth, the number of bits per sample, dictates amplitude precision.

This study uses a sampling rate of 44.1 kHz with 16-bit depth, a configuration standard for high-quality audio and consistent with the native format of the Afrikaans corpus and the public benchmark datasets (EmoDB, RAVDESS) used for comparative evaluation. According to the Nyquist–Shannon sampling theorem, a 44.1 kHz rate faithfully captures frequency content up to 22.05 kHz, comfortably exceeding the upper bound of the human voice and the spectral range relevant to emotional cues such as pitch, formant structure, and prosodic variation. Retaining the native 44.1 kHz rate — rather than down-sampling to 16 kHz as is common in speech recognition pipelines — preserves the high-frequency energy and fine spectral detail that carry paralinguistic information, which is particularly important for emotion recognition. A 16-bit depth provides a dynamic range of approximately 96 dB, sufficient to represent both quiet and loud emotional utterances without perceptible quantisation noise. An example of a signal sampled under these parameters is shown in Figure 2.5.

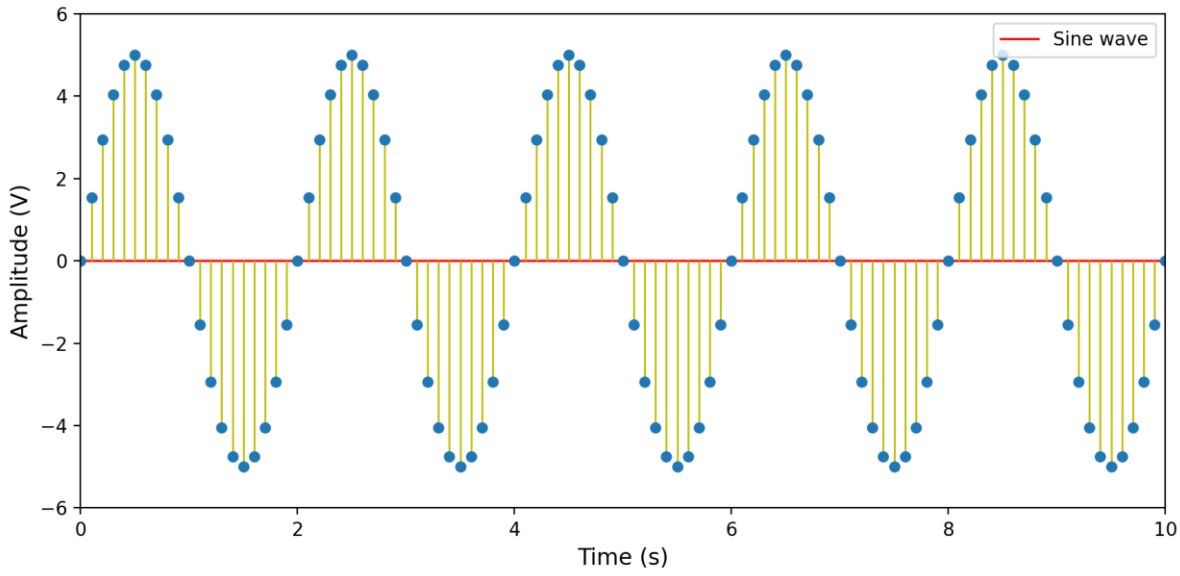

Figure 2.5: Sampled Signal (*Generated using Python*).

During digitisation, analogue signal values are approximated to the nearest discrete level, a process known as quantisation, illustrated in Figure 2.6.

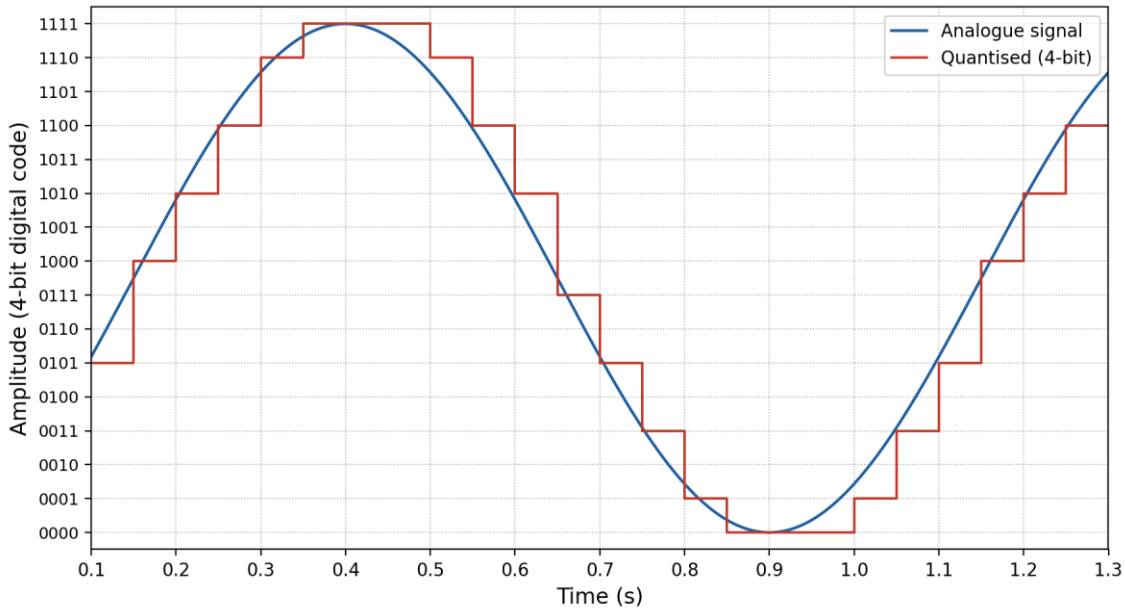

Figure 2.6: Quantisation Signal (*Generated using Python*).

Equally key is shifting the signal from the time domain to the frequency domain via the Fourier transform. Illustrated in Figure 2.7 and Figure 2.8.

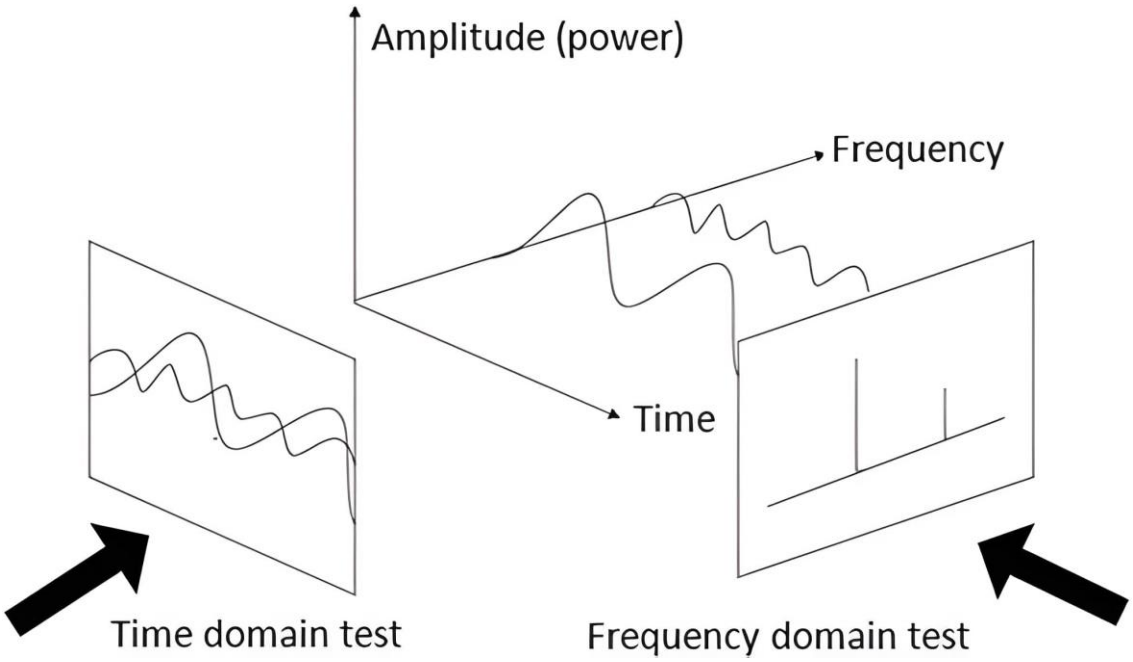

Figure 2.7: Time vs Frequency Domain *(Sun, 2019)*.

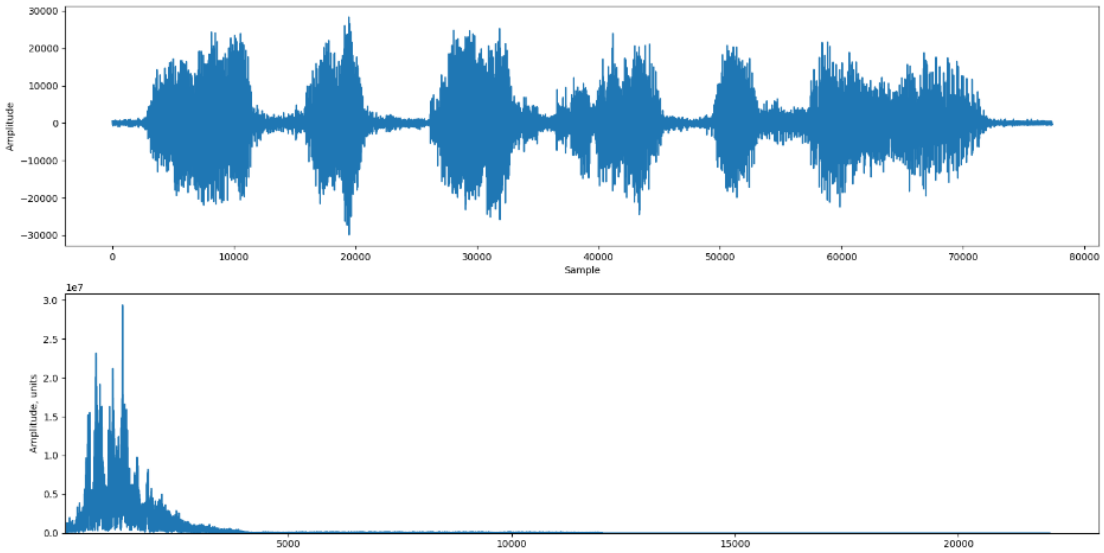

Figure 2.8: Time domain Fourier Spectrum (*Generated using Python*).

2.4.2 Audio Processing

Digital Signal Processing (DSP) involves the mathematical manipulation of signals from diverse sources, including voice, video, temperature, pressure, position, and audio. The thesis emphasises audio DSP, as it constitutes the primary input for neural network (NN) models. Digitised audio signals undergo operations such as addition, subtraction, multiplication, or division to enhance quality (Pohlmann, 2011). The section explores techniques designed to enhance input data clarity, with a focus on noise reduction strategies.

2.4.2.1 Noise Reduction

Noise refers to random, unwanted fluctuations that degrade signal quality and integrity (Vaseghi, 1996). An example of noise superimposed on a signal is shown in Figure 2.9.

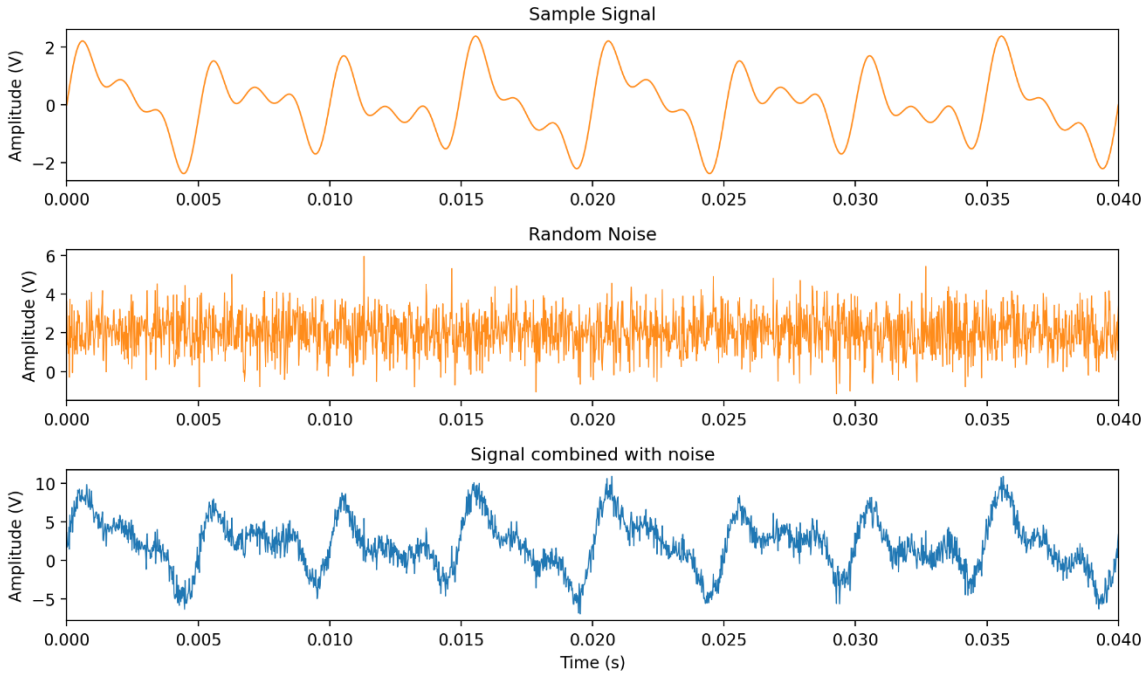

Figure 2.9: Random Noise Added to a Signal (*Generated using Python*).

Unwanted signal contamination comes in many guises, ranging from white and coloured varieties through to impulsive bursts, short-lived transient pulses, thermal and shot components, electromagnetic pickup, distortions introduced by the transmission channel, echo artefacts, reflections arising from multipath propagation, and residual modelling errors (Vaseghi, 1996). Figure 2.10 Illustrates a signal after noise reduction.

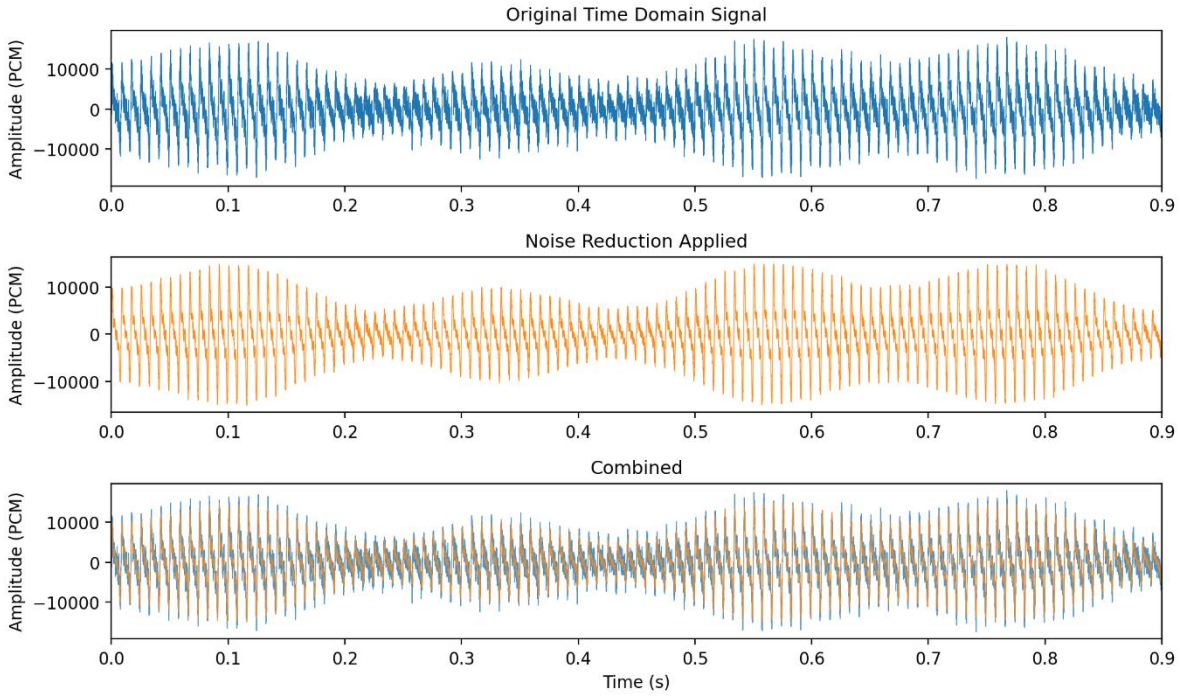

Figure 2.10: Noise Reduction in Signal (*Generated using Python*).

Contemporary noise removal techniques include the Kalman Filter, Least Mean Square (LMS) Adaptive Filter, Signal-Dependent Rank Order Mean (SDROM) algorithm, and Spectral Gating algorithm, each offering distinct approaches to enhancing audio clarity (Naik, Bhatikar, & Gaude, 2021; Haque & Bhattacharyya, 2018).

2.4.2.2 Kalman Filter

Introduced by R.E. Kalman in 1960, the Kalman Filter is a discrete-data linear filtering algorithm initially developed for state estimation in control systems, notably autonomous navigation (Welch & Bishop, 1995). Its utility extends to audio signal processing, where it excels in noise reduction. The filter addresses the estimation of a system's state within a discrete-time controlled process, modelled by the linear stochastic equation (Equation 2.1). When incorporating measurements, the model adjusts as per Equation 2.2.

State Transition Model

$$x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1} \tag{2.1}$$

Measurement Model

$$z_k = Hx_k + v_k \tag{2.2}$$

Here w_k and v_k denotes process and measurement noise, respectively, assumed to be independent with Gaussian distributions:

Process Noise Distribution

$$p(w) \sim \mathcal{N}(0, Q) \tag{2.3}$$

Measurement Noise Distribution

$$p(v) \sim \mathcal{N}(0, R) \tag{2.4}$$

Figure 2.11 demonstrates the Kalman Filter's application to a noisy audio signal, highlighting its effectiveness in smoothing perturbations.

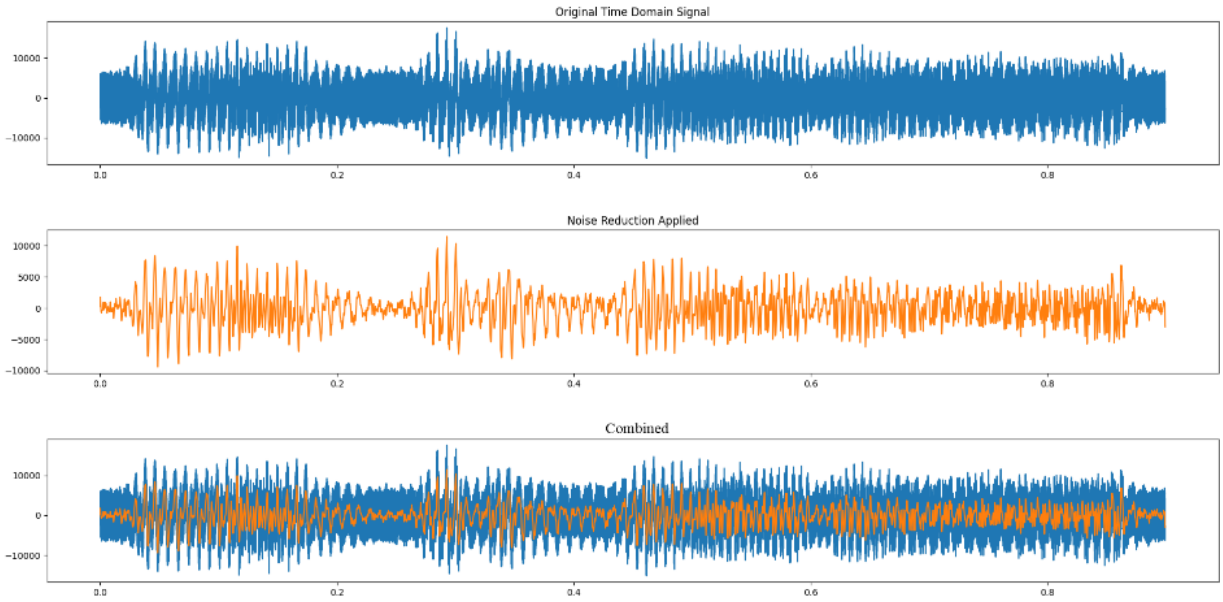

Figure 2.11: Kalman Filter applied to a noisy signal (*Generated using Python*).

- Least Mean Square (LMS) Adaptive Filter

The Least Mean Square (LMS) Adaptive Filter is an iterative algorithm for noise reduction, expressed as:

LMS Filter Output

$$y(k) = w_1 \cdot x_1(k) + \cdots + w_n \cdot x_n(k) \tag{2.5}$$

where k is the discrete time index, (.)T denotes transposition, y(k) is the filtered output, w is the vector of adaptive filter coefficients, and $x(k)$ is the input signal vector. Stability is maintained under the condition:

LMS Stability Criterion:

$$0 < \mu < 2/\lambda_m ax \tag{2.6}$$

where μ is the step size and $\lambda_{\max}$ is the maximum eigenvalue of the input correlation matrix. Figure 2.12 illustrates its noise-reduction capability.

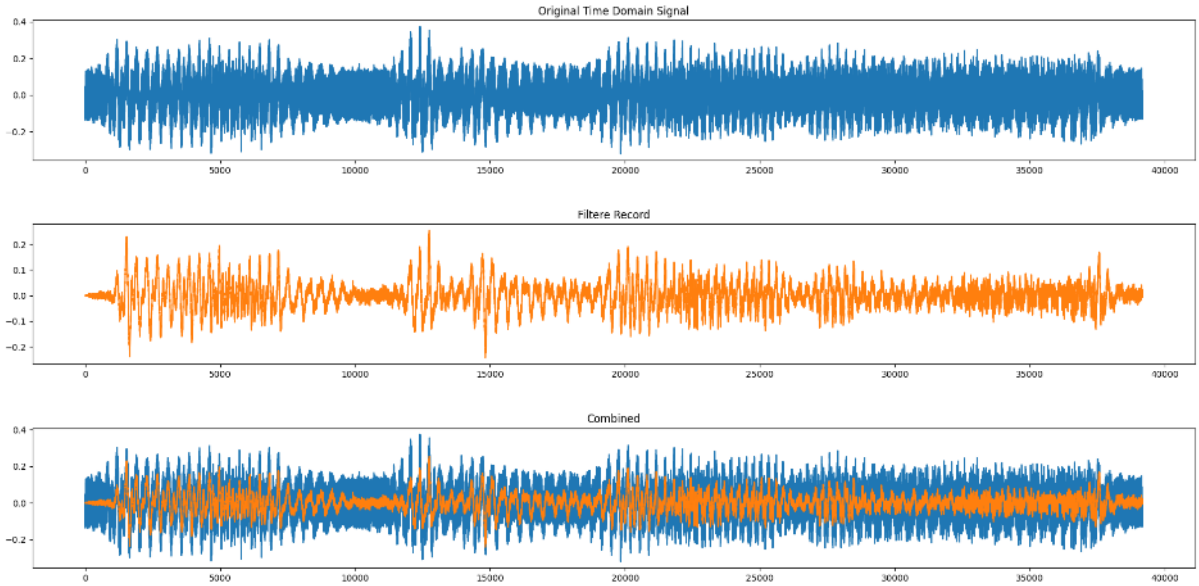

Figure 2.12: Noise reduction by using LMS (*Generated using Python*).

2.4.2.3 Spectral Gating Algorithm

Spectral gating is a robust algorithm used extensively in audio signal processing. It is beneficial for noise reduction in audio signals. The algorithm identifies and isolates the 'spectral envelope', a signal's frequency characteristic. In its essential operation, the spectral gating algorithm applies a gate in the frequency domain, allowing frequencies within the spectral envelope to pass while suppressing the others. The process results in the reduction of noise, especially broadband noise, while preserving the essential characteristics of the original signal. The effectiveness of the spectral gating algorithm, however, can depend on the nature of the noise and the signal. Figure 2.13 illustrates the concept. For instance, it may be less effective when the noise and signal characteristics overlap significantly in the frequency domain. Despite the limitation, spectral gating is an indispensable tool in audio signal processing due to its relative simplicity and robust performance in many scenarios.

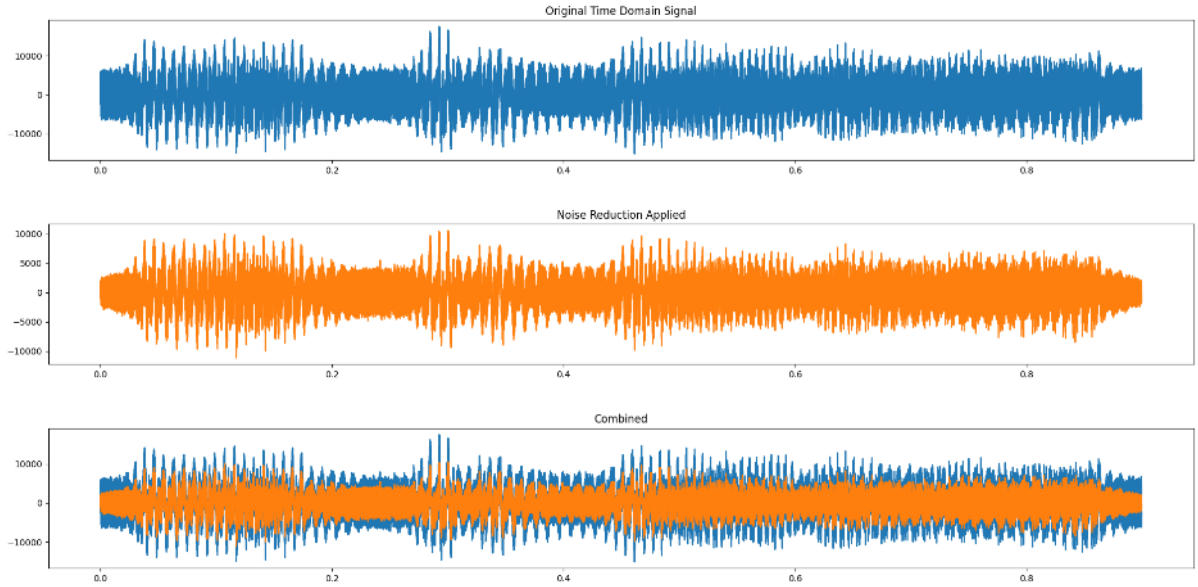

Figure 2.13: Noise reduction by using Spectral Gating (*Generated using Python*).

2.4.2.4 Spectral Subtraction

Spectral subtraction removes noise by subtracting the magnitude spectrum of a noise signal from the original signal. The formula for the method is below.

$$\left|\hat{X}(f)\right|^b = |Y(f)|^b - \alpha|N(f)|^b \tag{2.7}$$

$\left|\hat{X}(f)\right|^b$ approximates the original signal spectrum $|X(f)|^b$. $|N(f)|^b$ are the time-averaged noise spectra. The $\propto$ parameter controls the amount of noise subtracted (Vaseghi, 1996). Spectral subtraction is illustrated in Figure 2.14.

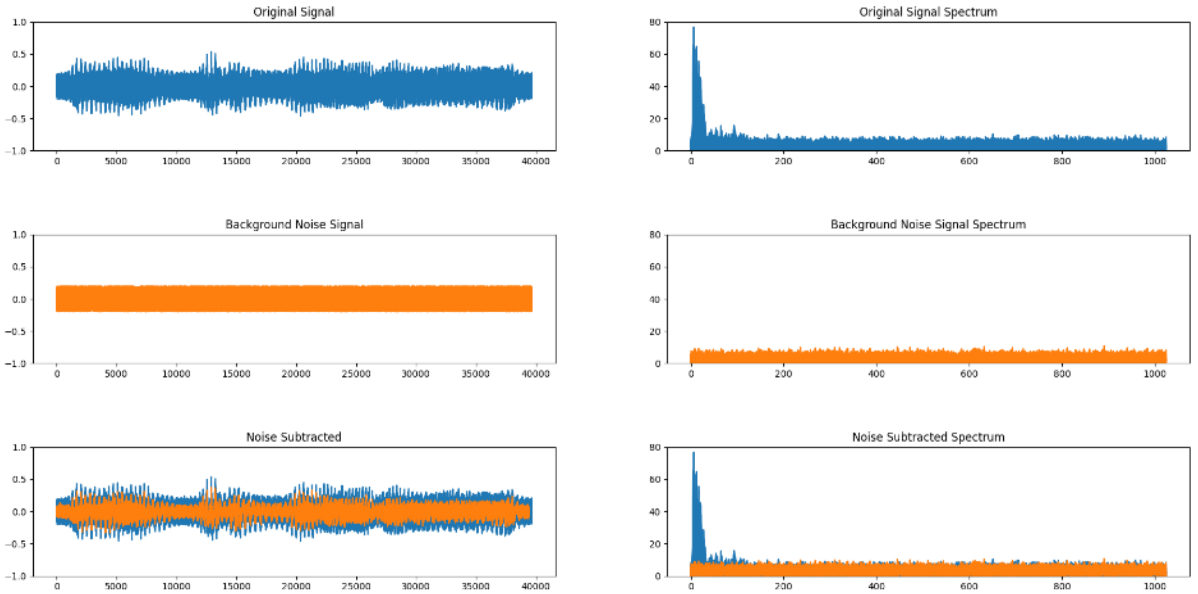

Figure 2.14: Spectral Subtraction Example (*Generated using Python*).

A variation of the method is called the Boll Spectral Subtraction (Naik, Bhatikar, & Gaude, 2021). The method implements spectral averaging and residual noise reduction.

2.4.2.5 FIR & IIR Filters

Filtering can be done on an analogue signal or a digitised discrete signal. The generic name for a linear time-invariant system is a filter. Discrete-time LTI systems are also referred to as digital filters. Two types of digital filters are used. The Finite Impulse Response (FIR) and the Infinite-duration Impulse Response (IIR) (Ingle & Proakis, 2012). The primary difference between FIR and IIR is that FIR is more straightforward and lacks a feedback mechanism. Standard filtering methods include high-pass, low-pass, band-pass, and notch filters. A low-pass filter is illustrated in Figure 2.15. A bandpass filter and response are illustrated in Figure 2.16 and Figure 2.17. The filter order and frequency response are also visible.

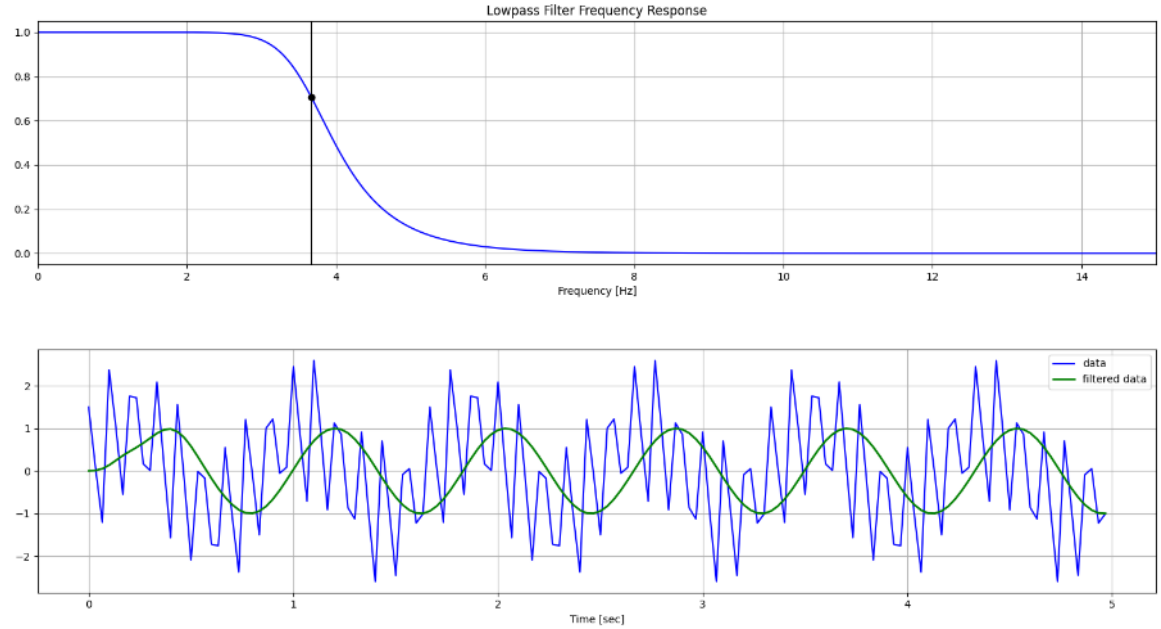

Figure 2.15: Lowpass Filter (*Generated using Python*).

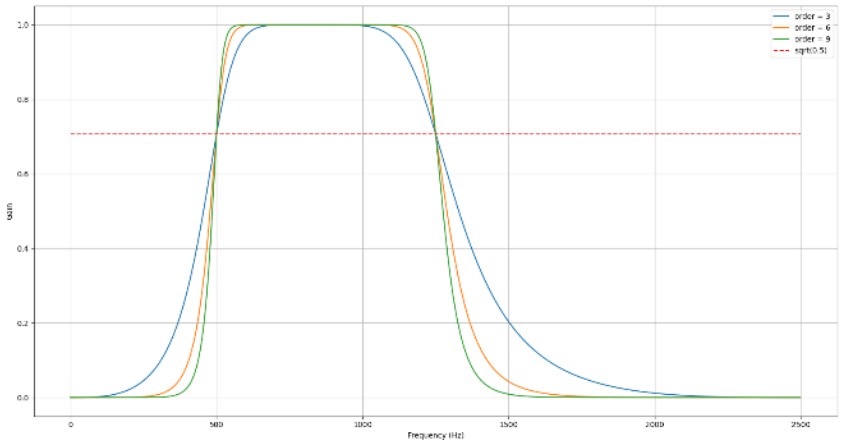

Figure 2.16: Bandpass Filter Responses (*Generated using Python*).

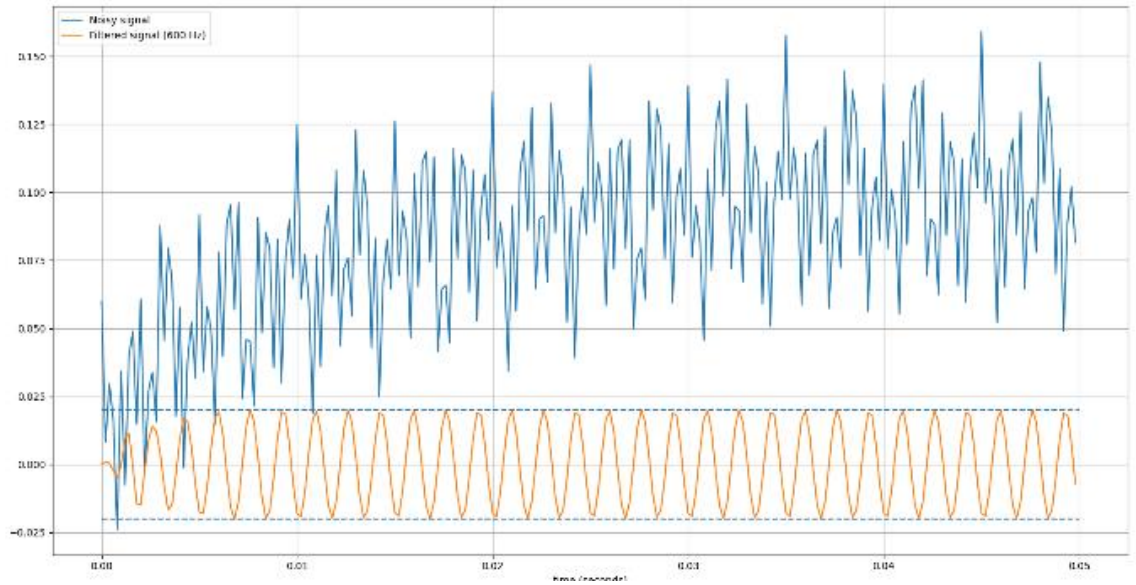

Figure 2.17: Bandpass Filter (*Generated using Python*).

For the research, the input sounds will be processed via a bandpass filter to filter out unwanted frequencies (Rehman, Liu, Wu, Cao, & Jiang, 2022).

2.4.2.6 Silence Removal

For the research, silent audio segments will be removed from voice clips and processed (Harár, Burget, & Dutta, 2017). To remove silence, one needs to take the following steps:

- Load data.
- Split the file into segments, keeping the data above a specific dB threshold.
- Recombine the split sections into the new file.
- An illustrative implementation of the procedure using the Librosa library for Python is illustrated in Figure 2.18 which compares the original waveform with the silence-trimmed output.

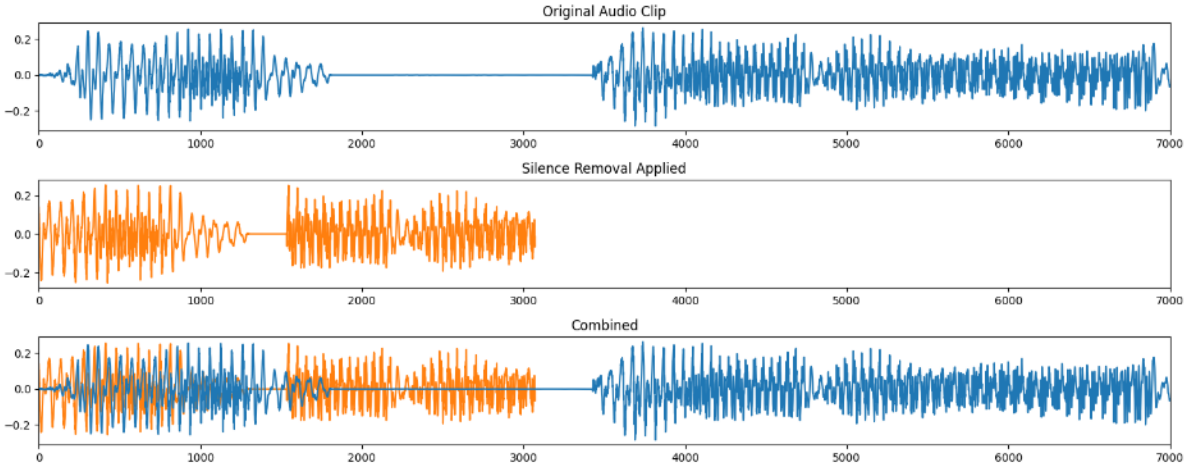

Figure 2.18: Silence removal applied to a speech signal (*Generated using Python*).

2.4.2.7 Comparative Evaluation of Audio Preprocessing Techniques

While the preceding sections describe several noise reduction techniques, a critical comparison is required to evaluate their suitability for Speech Emotion Recognition (SER). Each method offers distinct advantages and limitations depending on the characteristics of the noise and the need to preserve emotionally relevant speech features. The Kalman Filter provides a probabilistic framework for estimating clean speech signals under noisy conditions and is particularly effective in scenarios with structured or partially known noise characteristics. Its ability to model temporal dependencies makes it well-suited for speech signals where continuity is important. However, its reliance on assumptions such as linearity and Gaussian noise can limit performance in highly complex or unpredictable acoustic environments. The Least Mean Square (LMS) adaptive filter offers a computationally efficient and flexible solution that adapts dynamically to changing noise conditions. The approach makes it suitable for non-stationary noise environments. However, its performance is sensitive to parameter selection, particularly the step size, and it may converge slowly or introduce distortions that can affect prosodic features such as pitch and intensity, which are critical for SER. Spectral Gating operates in the frequency domain and is effective for suppressing broadband noise without requiring an explicit noise model. The approach makes it practical for real-world applications where noise characteristics are unknown. However, when noise overlaps significantly with speech frequencies, the method may suppress important emotional cues, particularly for low-energy emotions such as sadness or fear.

Spectral Subtraction is widely used due to its simplicity and computational efficiency, particularly for stationary noise. However, it is prone to introducing artefacts such as musical noise, which can distort temporal and spectral features essential for emotion recognition, thereby reducing classification performance.

From a comparative perspective, spectral methods (gating and subtraction) are generally effective for stationary noise but risk introducing artefacts or removing subtle emotional features. Adaptive filtering methods such as LMS improve robustness to changing noise conditions but may struggle with stability and parameter sensitivity. The Kalman Filter provides a more balanced approach, offering improved signal reconstruction and preservation of temporal dynamics, albeit at increased computational cost.

For Speech Emotion Recognition, preserving subtle prosodic and spectral features is critical, as over-aggressive noise reduction can remove emotionally salient information. Therefore, preprocessing techniques must balance noise suppression with feature preservation.

In the study, Kalman filtering and spectral gating were selected for empirical evaluation (see Chapter 4). The second Kalman filter variant (Kalman_2) achieved the highest performance, indicating that structured noise modelling provides a more effective balance between noise reduction and preservation of emotional signal characteristics in controlled recording conditions.

2.5 AI & NN Architectures

AI has been around since the 1950s. With advances in computing power and technologies such as big data, data analytics, and supercomputing, there is considerable interest. Recent developments in AI include deep learning, image recognition, machine learning, and natural language processing. AI will eventually impact most aspects of everyday life. AI will also transform business activities and social media (Gbadegeshin, et al., 2021). In Figure 2.19 it

can be observed that Deep learning is a subcategory of general artificial intelligence. In Figure 2.20 the difference between machine learning and deep learning can be observed.

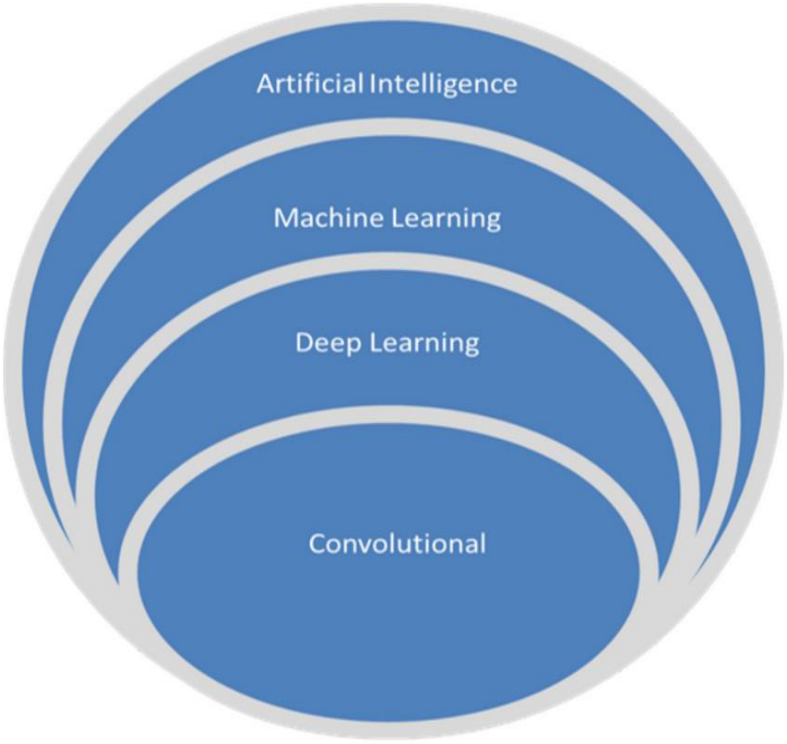

Figure 2.19: Artificial Intelligence branches *(Abid & Munir, 2025)*.

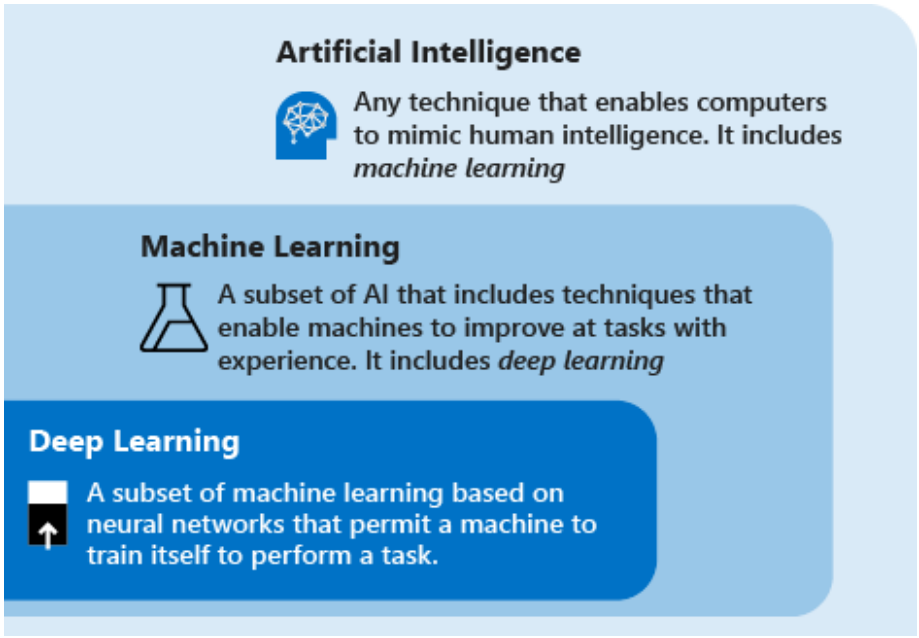

Figure 2.20: Machine Learning vs Deep Learning *(Deep learning vs. machine learning in Azure Machine Learning, 2026)*.

Machine learning used to be more popular, but with the improvements in deep learning coupled with computing power, deep learning has become the method of choice. The critical difference is that no manual classification is required for Deep Learning.

2.5.1 ANN

A Deep Learning network is built on the basic building block called an Artificial Neural Network (ANN). Natural neurons are illustrated in Figure 2.21 and artificial neurons are illustrated in Figure 2.22 (Gershenson, 2003).

2.5.1.1 Neurons

The artificial neuron is a computational model. Natural neurons inspired it. For natural neurons, signals are received through synapses located on the dendrites. As illustrated in Figure 2.21. When strong signals surpass a certain threshold, the neuron is activated. Subsequently, a signal is emitted through the axon. The signal can now be sent to other synapses, which in turn would activate other neurons (Gershenson, 2003).

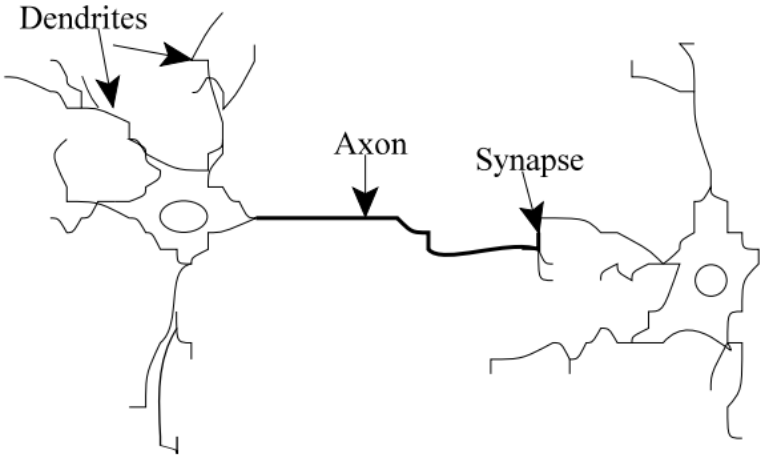

Figure 2.21: Natural neuron *(Gershenson, 2003)*.

When modelling artificial neurons, the design is based on natural neurons. An artificial neuron is illustrated in Figure 2.22. Artificial neurons consist of inputs multiplied by weights and then computed by a mathematical function. Inputs are the same as dendrites, and weights are the same as synapses. A mathematical function determines the activation of the neuron (Gershenson, 2003).

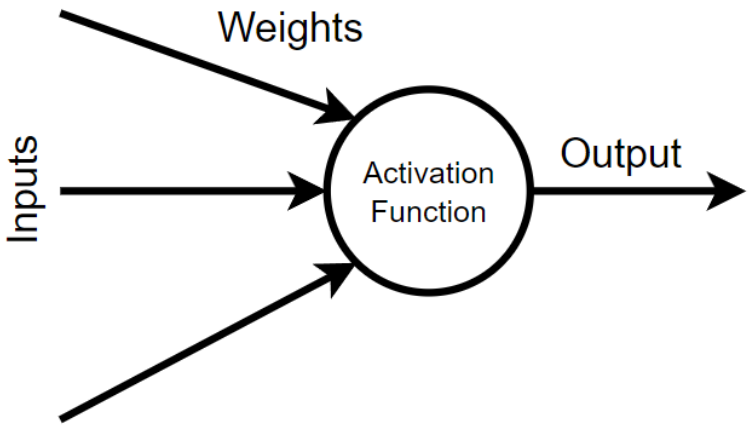

Figure 2.22: Artificial neuron.

When the weight of an artificial neuron is high, the input multiplied by it will be more substantial. Weights can be negative as well. If the weight is negative, the input signal is inhibited. Weight determines the computation of the neuron. By adjusting weights, the correct output is achieved for a given input signal. ANNs can have hundreds of thousands of neurons. Various algorithms do weight calculations (Gershenson, 2003).

2.5.1.2 Back Propagation

Backpropagation is a commonly employed method for training feed-forward neural networks. Backpropagation computes the gradient of the loss with respect to the weights. Weights are constantly updated whilst training to minimise gradient loss. Other variants, like the stochastic gradient, are also used. Gradients are calculated using the chain rule, layer by layer. Backward iteration from the last layer is used to avoid redundant calculations (Drucker & Cun, 1992).

2.5.1.3 Layers

Neural Network layers are illustrated in Figure 2.23. The primary layers that can be found in a Deep Learning Neural Network are:

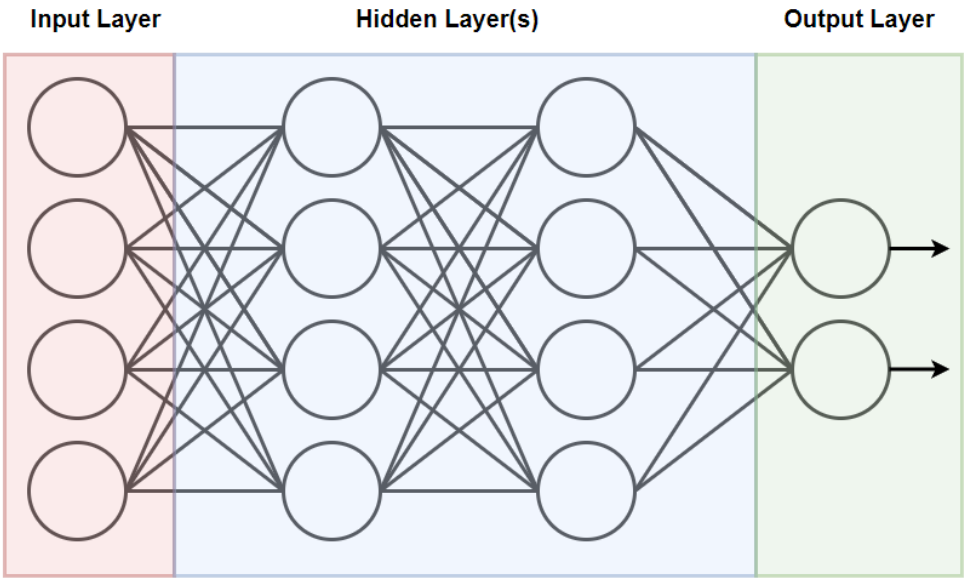

Figure 2.23: Neural Network Layers *(Mishra, Reddy, Sandesh, & Pathak, 2021)*

- **Basic layer**

A model can be described as one or more basic layers of neurons. Each layer contains neurons that, in turn, apply transformations on elements of the input tensors. In Deep learning, the primary layer is known as the dense layer.

- **Convolution Layer.**

For computer vision and more complex tasks, the convolution layer is used. The convolution layer is used to analyse an image piece by piece. The image is segmented into blocks, and each block is processed individually through the layer. Convolutional layers are very effective at classifying image objects. The concept is illustrated in Figure 2.24.

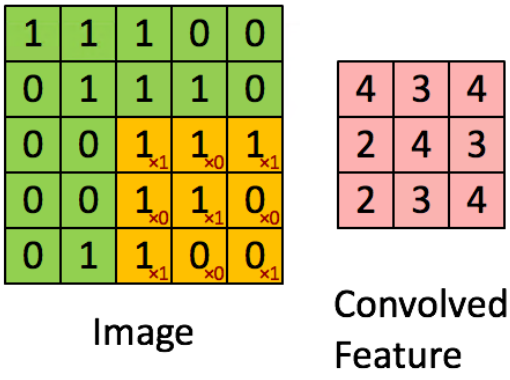

Figure 2.24: Convolution in action *(Saha, 2018)*.

- **Pooling layer**

Pooling is used to reduce the size of the data. There are quite a few methods to accomplish

this, but the well-known pooling methods are Max Pooling and Average Pooling (Murray & Perronnin, 2014). The concept is illustrated in Figure 2.25.

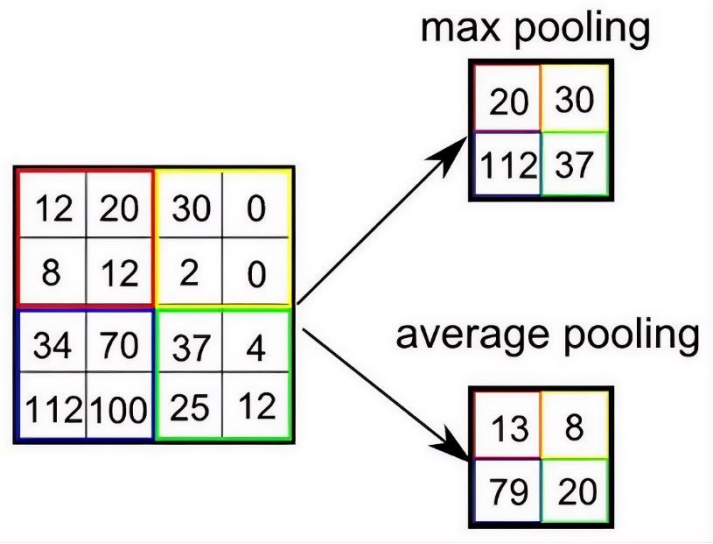

Figure 2.25: Types of pooling *(Saha, 2018)*.

- **Recurrent Layer**

A recurrent layer is mainly used for Natural Language Processing (NLP) and text processing. A recurring layer allows the Deep learning model to have memory. In more technical terms, the input will be analysed several times. The looping mechanism enables the capture of sequential information in the input data. The difference between CNN and RNN is illustrated in Figure 2.26.

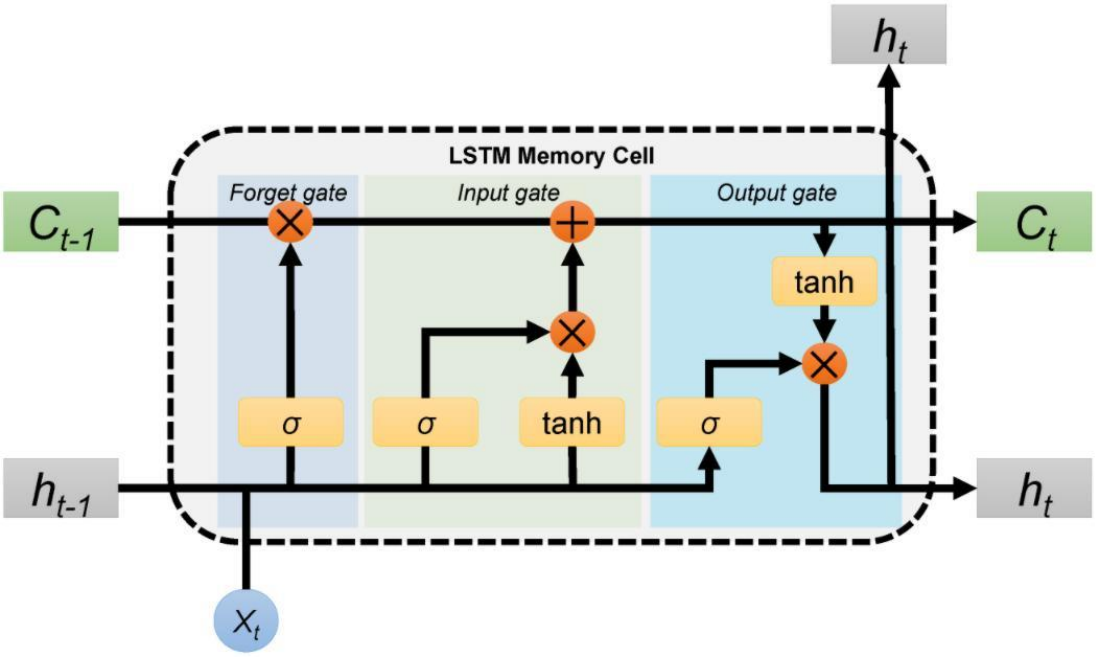

Figure 2.26: LSTM Recurrent Layer *(Hochreiter & Schmidhuber, 1997)*.

- **Normalisation Layer:**

The output from a deep learning network is often scattered. Normalisation normalises the data. The layer ensures that the output data has the same distribution as the input data. Data normalisation illustrated in Figure 2.27.

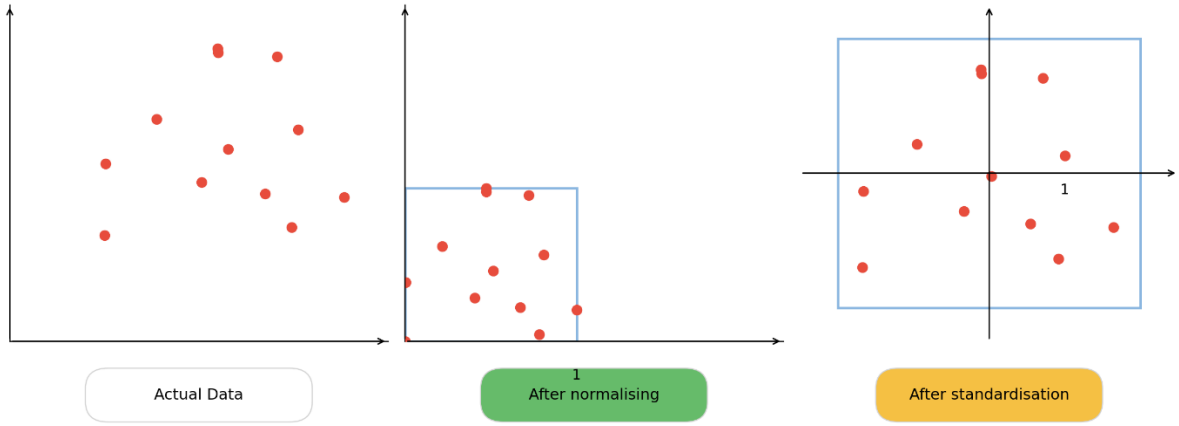

Figure 2.27: Data normalisation (*Generated using Python*).

- **Regularisation layer:**

The regularisation layer improves the model's learning. Examples of regularisation layers are the Dropout layer, Spatial Dropout1D layer, Spatial Dropout2D layer, Spatial Dropout3D layer, Gaussian Dropout layer, Gaussian Noise layer, Activity Regularisation layer, and the Alpha Dropout layer. The drop layer changes some tensor values to 0, increasing the model's complexity and efficiency.

- **Attention layer:**

The most advanced layer is the attention layer. The primary task of an attention layer is to enhance the performance of the RNN network. Generally, in a RNN, the already analysed data is kept in memory to compare with new values. Usually, the latest data is compared to the memory in one direction but not in another direction. Memory and new data are also not compared in terms of their relationship with each other. Using the concept of the attention layer establishes the relationship between new data and data in memory. Doing so means that more data is extracted, resulting in better accuracy.

2.5.1.4 Activation Functions

The function that decides whether a neuron fires is called an Activation Function (AF). AF are functions used to compute the weighted sum of inputs and biases. (Szandała, 2020; Nwankpa, Ijomah, Gachagan, & Marshall, 2018). Some standard Activation functions are described in more detail in the section below.

- **Step Function**

The step function showing a neuron activation is illustrated in Figure 2.28.

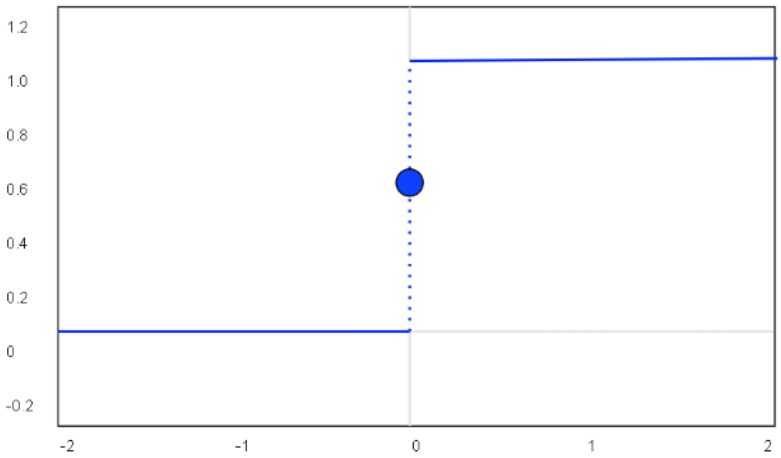

Figure 2.28: Step Function. Adapted from *(Szandała, 2020)*.

$$f(x) = \{0 \text{ for } x \leq T,\ 1 \text{ for } x > T\} \tag{2.8}$$

The formula above shows the step function. For the function F(x)=1, with conditions: for x < T and x>T. Primitive NN implements the step function. These primitive NNs usually have one hidden layer and a single output.

- **Linear Activation Function**

One of the problems and limitations of the step function is the presence of zero gradients. The gradient is not updated with backpropagation. The result of the gradient not being updated is an unprogressive gradient or slope. Applying a linear function instead of a step function overcomes the obstacle. The most straightforward linear equation is one that has an equivalent input-output relationship. The relationship is evident in the equation below.

$$f(x) = ax, \text{ where } a \in \mathbb{R} \tag{2.9}$$

$\mathbb{R}$ denotes the set of real numbers.

Multiple neurons can be activated when using linear activation. For linear activation functions, each layer in a multi-layered network is activated. The activations are fed into the next layer as inputs. The linear activation function is illustrated in Figure 2.29.

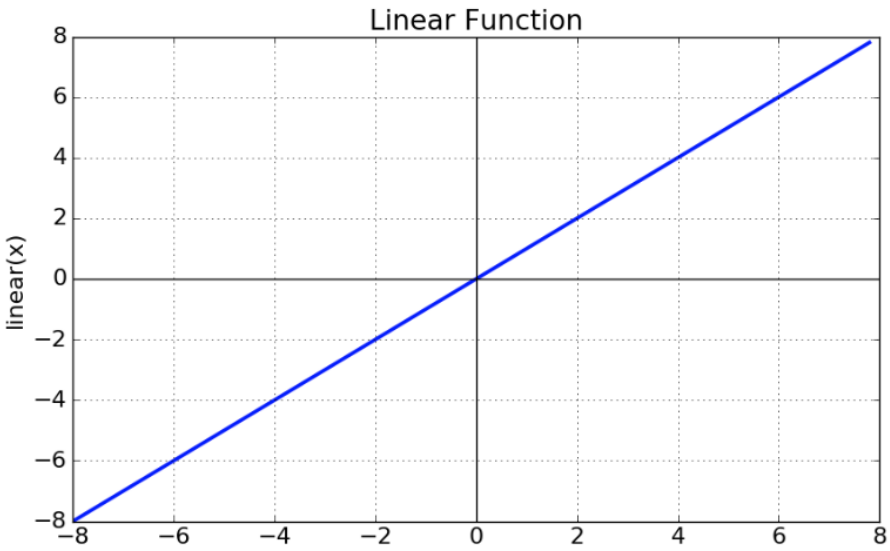

Figure 2.29: Linear activation *(SHARMA, 2017)*.

- "S"-shaped Activation Functions

The most common activation functions in NN today are the “S”-shaped functions. These include the Sigmoid function. These activation functions have a nonlinear output.

- Sigmoid Function

The sigmoid curve is representative of the sigmoid function. The sigmoid function is illustrated in Figure 2.30.

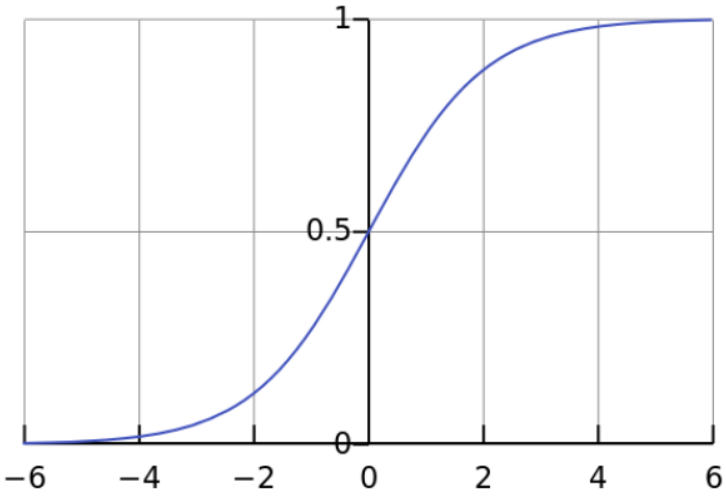

Figure 2.30: Sigmoid *(Szandała, 2020)*.

The sigmoid activation function provides for a smooth gradient. The characteristic makes it suitable for use in shallow neural networks. The equation below describes the sigmoid function.

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{2.10}$$

- Hyperbolic Tangent Activation Function

A limitation of the Sigmoid activation is that the activation value might get stuck on the edge. To address the limitation, a hyperbolic function is used as the activation function. The hyperbolic activation function is illustrated in Figure 2.31.

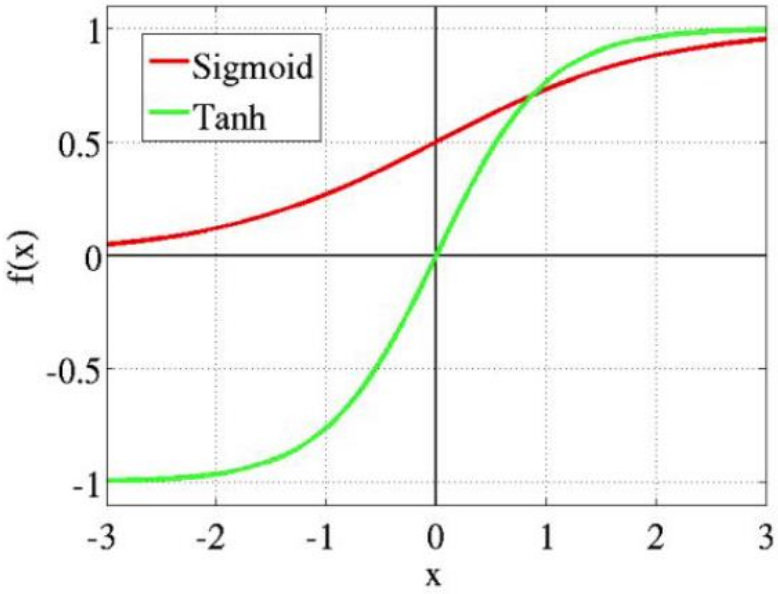

Figure 2.31: Hyperbolic Tangent *(Szandała, 2020)*.

The mathematical formula describing the function is presented below.

$$f(x) = \tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{2.11}$$

A key reason for choosing the Hyperbolic activation function over the Sigmoid activation function is that the derivatives of tanh are noticeably more extensive than those of the Sigmoid function.

- **Soft Sign Activation Function**

The Soft sign activation function is one of the most essential functions. It is much smoother than the tanh activation function. The function is illustrated in Figure 2.32.

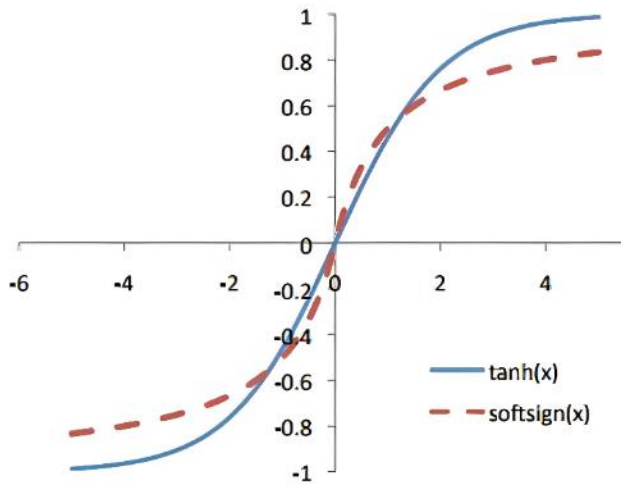

Figure 2.32: Soft sign vs tanh function *(Szandała, 2020)*.

The mathematical representation of the soft sign activation function is shown below.

$$f(x) = \frac{x}{1+|x|} \tag{2.12}$$

The Soft sign activation function grows poly-nominally rather than exponentially. The soft sign activation also prevented neurons from becoming saturated, resulting in improved learning.

- **Rectified Linear Unit Activation Function**

The Rectified Linear Unit Activation Function (ReLU) sends the input directly to the output if the value is positive or zero. If the input value is not positive, the output is zero. The ReLU function is illustrated in Figure 2.33.

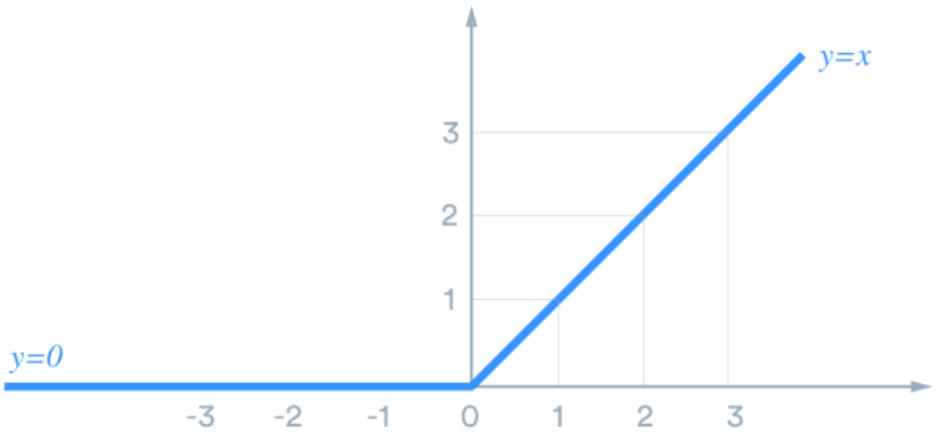

Figure 2.33: Rectifier Linear Unit Activation *(Szandała, 2020)*.

The mathematical function describing the activation function is shown below.

$$f(x) = x^+ = \max(0, x) \tag{2.13}$$

One of the benefits of using ReLU is its simplicity, which in turn requires less computing power. The vanishing gradient problem is avoided in ReLU. It must be mentioned that the ReLU function is sparsely activated because it outputs a zero for negative values (DeepAI, 2019).

While the activation functions described above are widely used, their suitability varies across application domains, particularly in Speech Emotion Recognition (SER). Conventional activation functions like the sigmoid and hyperbolic tangent (tanh) are susceptible to the vanishing gradient issue, where gradients decrease during backpropagation in deep networks, which restricts the capacity to learn intricate hierarchical representations. The sigmoid function is further constrained by its output range of 0 to 1, which can lead to neuron saturation and slower convergence. Although tanh partially mitigates the vanishing-gradient problem by producing zero-centred outputs, it remains susceptible to saturation at extreme values, reducing its effectiveness in deep architectures for sequential audio processing.

Compared to other activation functions, the Rectified Linear Unit (ReLU) has emerged as the favored option for hidden layers in deep learning architectures. This is mainly because it is computationally efficient and effectively preserves gradient flow for positive inputs. ReLU introduces sparsity by deactivating neurons for negative inputs, which can improve generalisation and reduce overfitting. The property is particularly advantageous in SER tasks, where models must learn discriminative features from high-dimensional spectral representations such as MFCCs. However, ReLU also introduces the "dying ReLU" problem, where neurons may become permanently inactive if they consistently receive negative inputs. The consideration is especially relevant in SER, where low-energy emotional expressions, such as sadness or fear, may produce weaker activations that risk being suppressed.

Empirical studies in speech emotion recognition have demonstrated that ReLU-based architectures generally outperform sigmoid-based networks in both convergence speed and classification accuracy, particularly in CNN and hybrid CLSTM models. However, these studies also highlight the need for careful weight initialisation and regularisation techniques, such as dropout, to mitigate the risk of inactive neurons and preserve subtle emotional features.

Based on these considerations, the study adopts ReLU as the primary activation function in the convolutional, dense, and dendritic layers of the CNN, CLSTM, DCLSTM, and DenCaps architectures. Tanh is utilised in the LSTM components, as its zero-centred activations help ensure stable sequential modelling. In contrast, softmax is applied in the output layer for classifying multiple emotions. The combination provides a balance between efficient training, stable gradient propagation, and effective modelling of emotional speech characteristics.

2.5.1.5 Dropout

One of the most commonly used Regularisation layer functions is the Dropout layer. When an NN is trained on a small dataset, it tends to overfit the data. The effect of overfitting is noise in the training data (Brownlee, 2018). At each training step, the dropout layer sets input units to 0. The concept is illustrated in Figure 2.34. The units are set so that the sum over all inputs is unchanged. Dropout makes the training noisy, which in turn forces nodes to take responsibility (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014).

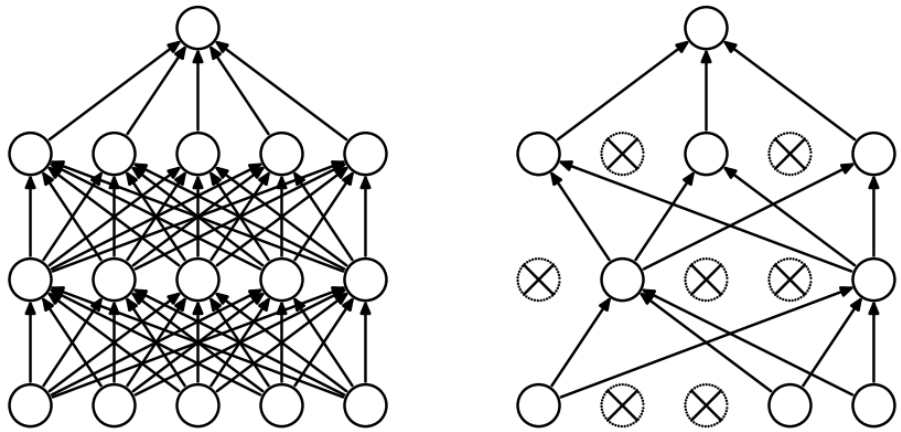

Figure 2.34: Dropout *(Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014)*.

2.5.2 CNN

A type of feed-forward network containing a convolutional input layer is called a convolutional neural network (CNN). These types of networks are primarily used in image analysis (Mishra, Reddy, Sandesh, & Pathak, 2021). The CNN architecture is shown in Figure 2.35. A standard Convolutional Neural Network (CNN) includes several key layers: the Convolutional Layer, the Pooling Layer, the Fully Connected Layer, Dropout, and Activation layers. Illustrated in Figure 2.36. The convolution layer takes the input and convolves or slides over it, processing the input data piece by piece. The output is then passed onto the next layer.

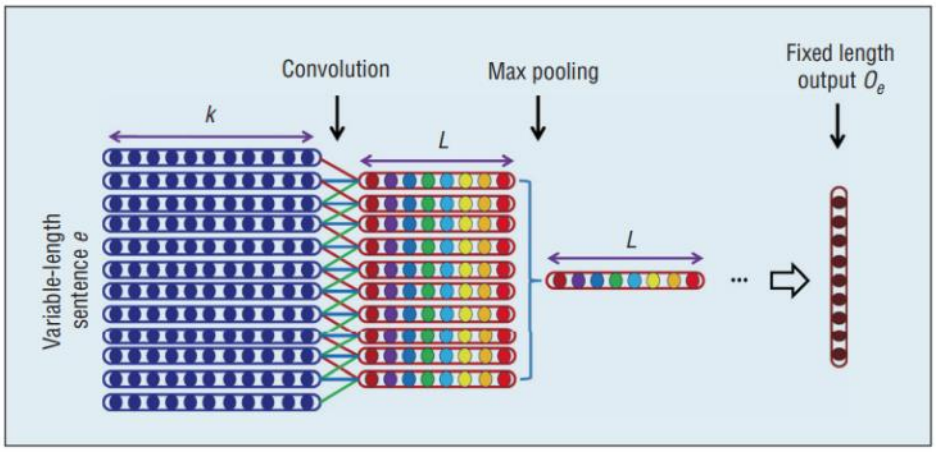

Figure 2.35: CNN Architecture *(Zhang & Zong, 2015)*.

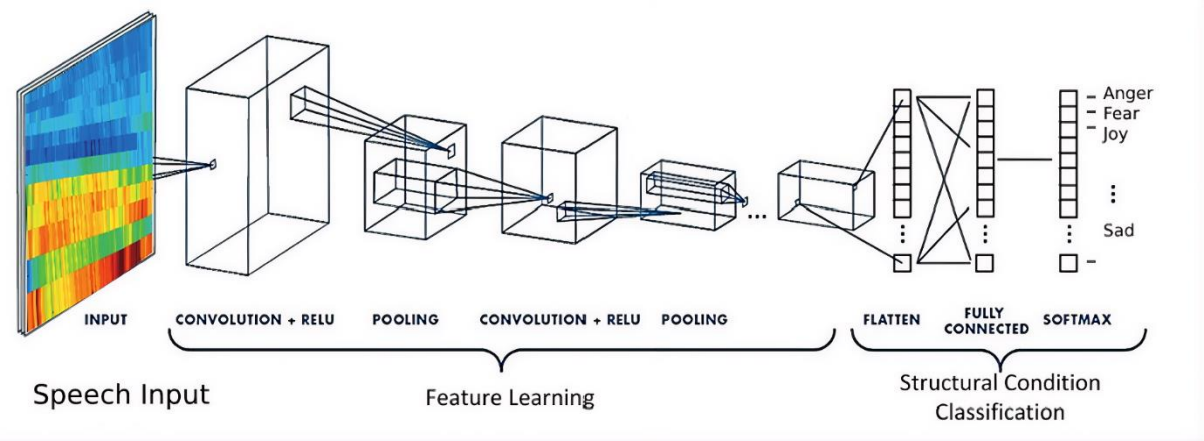

Figure 2.36: CNN layers. Adapted from *(Zhang & Zong, 2015)*.

2.5.3 RNN & LSTM

2.5.3.1 RNN

Humans, when reading through a passage, can grasp the meaning of each new word by drawing on the words encountered earlier — a contextual stitching the brain handles naturally. Recurrent neural networks (RNNs) work on a comparable principle. RNNs belong to the family of neural models designed to process ordered data streams, primarily for sequential pattern recognition. An incoming stream of audio or image data is split into successive segments and dispatched to the network one segment at a time. Because RNNs retain a form of memory across steps, they are well-suited to applications such as share-price forecasting, language translation, speech transcription, and image recognition. A representative RNN is depicted in Figure 2.37. During the forward pass, the network ingests one word at a time and preserves the ordering by looping prior activations back in alongside each newly arriving word. The output of that operation is then routed as input into the hidden layers. As a concrete example, take a sentence containing four words supplied to the network. In the first step, the opening word is converted into a vector and dispatched to a designated hidden layer comprising a number of neurons, where the vector is combined with the layer's assigned weights. The

weighted contributions are passed through an activation function such as ReLU, yielding the hidden-layer output at that step. That output is recirculated back into the same hidden layer as part of the recurrent loop. In the second step, the second word is fed into the same hidden layer using the same weight set previously applied to the first word. Separately, the output produced for the first word is multiplied by its own distinct weights before being routed back into the same hidden-layer neurons alongside the new input as part of the loop. The combined contributions — the weighted input for word two plus the weighted recurrent signal carried over from word one — are pushed through the activation operator to produce the step-two output. As a result, the output for the second word reflects both the second word's own input and the residue of the first word's output. This mechanism is what allows the network to keep the sequence intact through to the final word of the sentence. In the same fashion, the weighted input of the third word, combined with the output of the second word, is sent through the activation operator to produce the third word's output at step three. The procedure continues in this manner until the final word is reached. After the fourth word has passed through, its output is multiplied by a new weight set and pushed through an output-stage activation operator, such as SoftMax, which delivers the predicted value as the network's final output. Once processing is complete, a loss term is computed between the produced output and the model's predicted value. With the loss in hand, the weights are updated via backpropagation. To carry this out, the derivative of the loss with respect to the output is computed and used to update the weights, with the chain rule providing the required derivatives. The cycle repeats until the loss curve flattens at its global minimum, at which point training is terminated. RNNs come with their own difficulties and limitations, most notably the vanishing gradient and exploding gradient phenomena. As an example, when the network is backpropagating, sigmoid activations remain bounded between 0 and 1. With repeated application of the chain rule, the derivative shrinks to a negligible magnitude after successive differentiation. Consequently, the weight updates become minimal, which produces the vanishing-gradient problem. As a counter-example, swap the sigmoid for an alternative such as ReLU. Under that arrangement, the derivative will be greater than zero. Consequently, the weights can be updated by a large amount, leading to the exploding gradient problem. Architectures such as the LSTM were introduced specifically to address the vanishing- and exploding-gradient issues. (Mishra, Reddy, Sandesh, & Pathak, 2021).

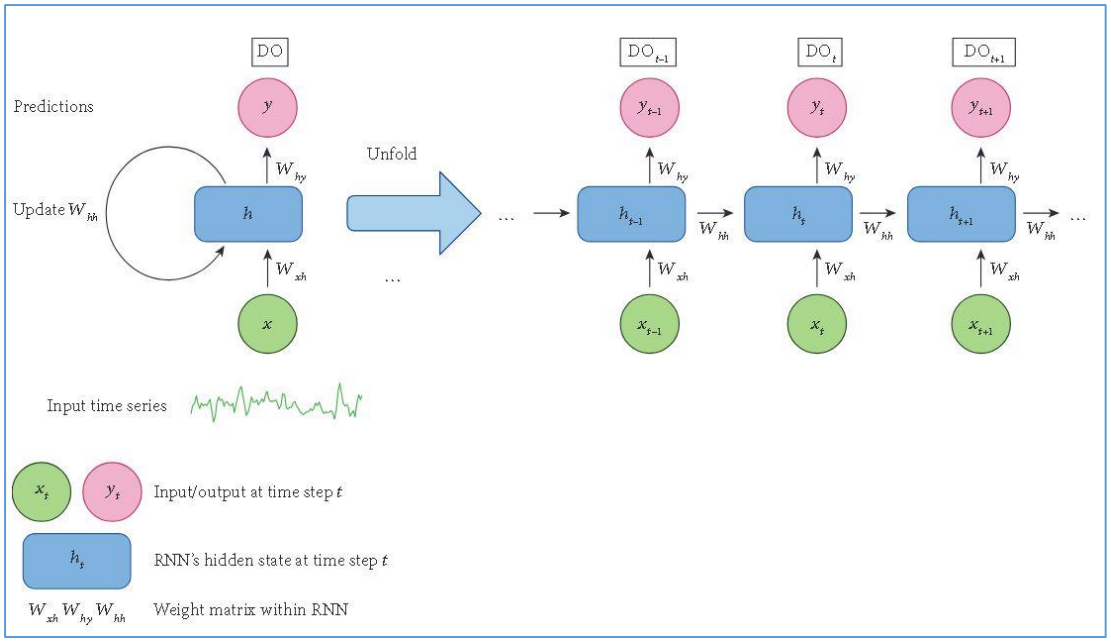

Figure 2.37: RNN *(Mishra, Reddy, Sandesh, & Pathak, 2021)*.

2.5.3.2 LSTM

A technique known as long short-term memory (LSTM) was developed to tackle the issue of vanishing gradients. The LSTM architecture can preserve error signals by utilising constant error carousels (CECs). The CEC solution consists of a continuous error flow within specific cells. Access to the CEC cells is allowed via the multiplicative gate units. The multiplicative gate units also protect memory content from perturbation by irrelevant inputs. The units together form a memory cell. (Hochreiter & Schmidhuber, 1997; Staudemeyer & Morris, 2019).

2.5.3.2.1 Vanishing Gradient Problem

In a standard Recurrent Neural Network (RNN), learning long-term dependencies is difficult because the backpropagated gradient may either grow or decay exponentially as it is propagated over multiple time steps. As a result, the gradient may either explode or vanish. Exploding gradients can lead to unstable learning and oscillating weight updates, whereas vanishing gradients can make learning extremely slow or prevent the network from capturing long-range temporal dependencies. The gradient-based update of the weight ($\mathrm{W_{uv}}$) over the interval from time ($\mathrm{t'}$) to time (t).

$$\Delta \mathrm{W_{uv}} = -\eta \frac{\partial \mathrm{E_{total}(t)}}{\partial \mathrm{W_{uv}}} \tag{2.14}$$

where the gradient of the total error with respect to the weight (W_{uv}) over the interval ($[t', t]$) can be expressed as

$$\frac{\partial E_{\mathrm{total}}(t',t)}{\partial W_{uv}} = \sum_{\tau=t'}^{t} \vartheta_u(\tau)\ x_v(\tau) \tag{2.15}$$

where ($\vartheta_u(\tau)$) denotes the backpropagated error signal of unit (u) at time (τ), for ($t' \leq \tau \leq t$), given by

$$\vartheta_u(\tau) = f_u'!\left(z_u(\tau)\right)\left(\sum_{v\in U} W_{vu}\ \vartheta_v(\tau+1)\right) \tag{2.16}$$

Consider an RNN with a set (U) of non-input units. The error signal generated at an output neuron ($o \in O$) is propagated backwards through time (t) to an earlier time (t'), where ($t' < t$). The corresponding error scaling factor for an arbitrary unit (v) is defined as:

$$\frac{\partial\vartheta_v(t')}{\partial\vartheta_o(t)} = f_v'\left(z_v(t')\right)\begin{cases} W_{ov} & t-t'=1 \\ \sum_u \frac{\partial\vartheta_u(t'+1)}{\partial\vartheta_o(t)} W_{uv} & t-t'>1 \end{cases} \tag{2.17}$$

To evaluate the expression in (2.17), the network is unrolled over time. For each time step (τ), where ($t' \leq \tau \leq t$), let (u_τ) denote a non-input unit in the unrolled network. By setting ($u_{t'} = v$) and ($u_t = o$), the error scaling factor can be written as

$$\frac{\partial\vartheta_v(t')}{\partial\vartheta_o(t)} = \sum_{u_{t'+1}\in U} \cdots \sum_{u_{t-1}\in U}\left(\prod_{\tau=t'+1}^{t} f'_{u_{\tau-1}}\left(z_{u_{\tau-1}}(\tau-1)\right) W_{u_\tau u_{\tau-1}}\right) \tag{2.18}$$

It follows from Equation (2.18) that if

$$\left|f'_{u_{\tau-1}}\left(z_{u_{\tau-1}}(\tau-1)\right) W_{u_\tau u_{\tau-1}}\right| > 1 \tag{2.19}$$

for all (τ) , the product term grows exponentially with increasing temporal distance. Consequently, the gradient may explode, resulting in unstable learning behaviour and oscillating weight updates due to large and conflicting error signals.

Conversely, if,

$$\left|f'_{u_{\tau-1}}\left(z_{u_{\tau-1}}(\tau-1)\right) W_{u_\tau u_{\tau-1}}\right| < 1 \tag{2.20}$$

for all(τ), the product term decreases exponentially as the temporal distance increases. In such cases, the gradient vanishes, preventing the network from effectively learning long-range dependencies.

Finally, consider the expression:

$$\sum_{o\in O} \frac{\partial\vartheta_v(t')}{\partial\vartheta_o(t)} \tag{2.21}$$

which represents the cumulative contribution of the backpropagated error signals from the output units. If these individual contributions become vanishingly small, their combined effect diminishes, thereby impairing effective learning across extended time intervals.

The behaviour of the LSTM cell is captured by the following equations, where x_t is the input at time step t, h_t is the hidden state, c_t is the cell state, σ denotes the sigmoid function, and ⊙ denotes element-wise multiplication.

Forget gate:

$$f_t = \sigma\big(W_f \cdot [h_{t-1}, x_t] + b_f\big) \tag{2.22}$$

Input gate:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2.23}$$

Candidate cell state:

$$\tilde{c}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \tag{2.24}$$

Cell state update:

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \tag{2.25}$$

Output gate:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{2.26}$$

Hidden state:

$$h_t = o_t \odot \tanh(c_t) \tag{2.27}$$

The forget gate f_t controls how much of the previous cell state is retained; the input gate i_t regulates how much new candidate information $\tilde{c}_t$ is written into the cell state; and the output gate o_t determines how much of the cell state is exposed in the hidden state h_t. The gating mechanism allows the LSTM to selectively retain or discard information across long sequences, addressing the vanishing-gradient problem characteristic of standard RNNs.

2.5.3.2.2 CEC

The Constant Error Carousel (CEC) is designed to preserve a constant error flow through the memory cell. Multiplicative gating mechanisms regulate access to the cell state and are trained to determine when information should be written to, retained in, or read from the cell. Consider a unit (u) with a single recurrent self-connection:

$$\vartheta_u(\tau) = f_u'\big(z_u(\tau)\big) W_{uu} \vartheta_u(\tau + 1) \tag{2.28}$$

From Equations (2.19) and (2.20), constant error flow is obtained when the multiplicative factor affecting the backpropagated error is equal to unity:

$$f_u'\big(z_u(\tau)\big) W_{uu} = 1 \tag{2.29}$$

Equation (2.29) implies that:

$$f_u'\big(z_u(\tau)\big) = \frac{1}{W_{uu}} \tag{2.30}$$

Since (W_{uu}) is constant, $(f_u'\big(z_u(\tau)\big))$ must also be constant. The condition implies that (f_u) must be a linear activation function:

$$y_u(\tau+1) = f_u\big(z_u(\tau+1)\big) = f_u(y_u(\tau)W_{uu}) = y_u(\tau) \tag{2.31}$$

By setting ($W_{uu} = 1$) and using the identity activation function, ($f_u(x) = x$), the activation remains constant over time. The self-recurrent structure forms the Constant Error Carousel at the core of the LSTM architecture and enables the preservation of error signals across extended time intervals.

However, in order for the memory cell to remain useful, the influence of external inputs and outputs must still be controlled. The mechanism is achieved through the input, output, and forget gates of the LSTM architecture.

2.5.3.2.3 Memory Blocks

The CEC is connected to itself and to other units in the neural network, which can result in conflicting weight-update signals being transmitted to a given neuron u. To address this, the standard LSTM unit is extended into a more complex structure known as a memory block, the architecture of which is illustrated in Figure 2.38.

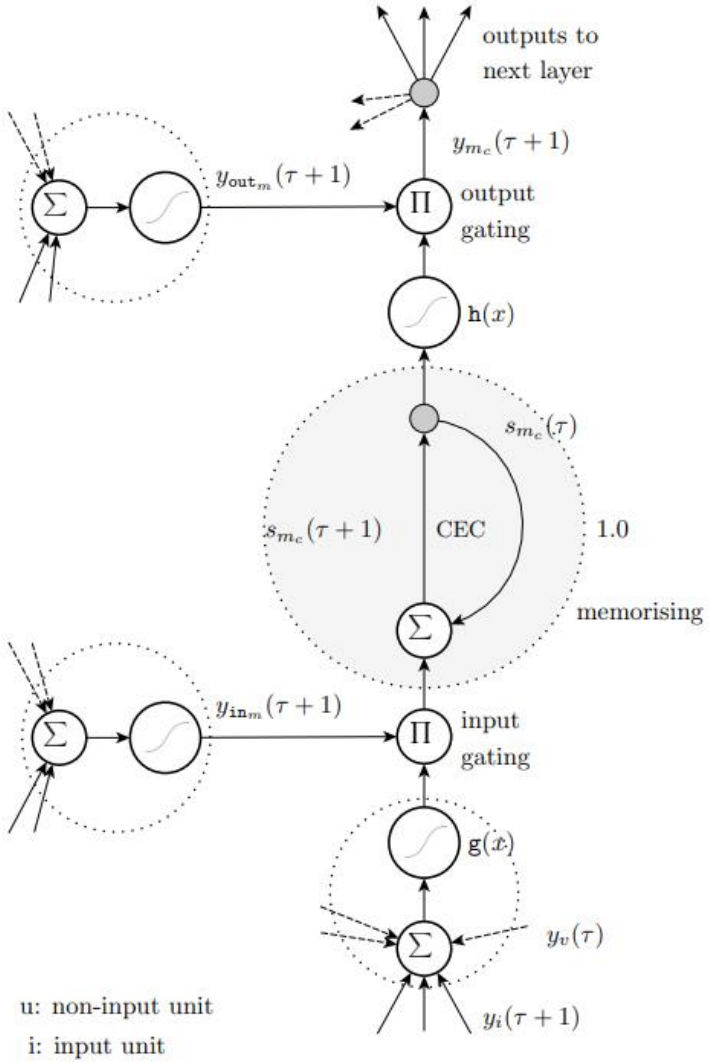

Figure 2.38: A standard LSTM memory block *(Staudemeyer & Morris, 2019)*.

The memory cell input is a simple sigmoid threshold unit with an activation range of [0, 1]. The input controls the signals from the NN to the memory cell by scaling the input signal. When the gate is closed, the activation is close to zero. The gates can also be implemented to protect contents from disturbance by irrelevant signals. The CEC input gate activation is also known as the cell state. The access to the output gates is regulated to prevent and protect other memory cells from disturbance (Staudemeyer & Morris, 2019).

2.5.4 GAN

In recent years, GANs have become an area of interest for many artificial intelligence institutions. A two-player zero-sum game initially inspired them. A GAN includes a generator and a discriminator. Estimating potential distribution and generating new samples lies at the core of GANs' operation. GANs are widely studied due to their extensive applications in image, vision, speech, and language processing.

The basic architecture is illustrated in Figure 2.39. A GAN (Generative Adversarial Network) consists of two main components: a Generator (G) and a Discriminator (D). The Generator is designed to transform a simple random distribution, denoted as $p_z(z)$, into a more complex distribution, represented as $\mathrm{p_g(x)}$. Here, (z) symbolises random noise, and (x) refers to the target data samples. Through training, the goal is for the generated sample distribution $\mathrm{P_g(x)}$ to become indistinguishable from the real data distribution $(\mathrm{P_d(x)}$. Meanwhile, the Discriminator is trained to distinguish genuine samples (real data) from those generated by the Generator (fake samples) using a minimax approach (Tian, Wan, & Liu, 2018).

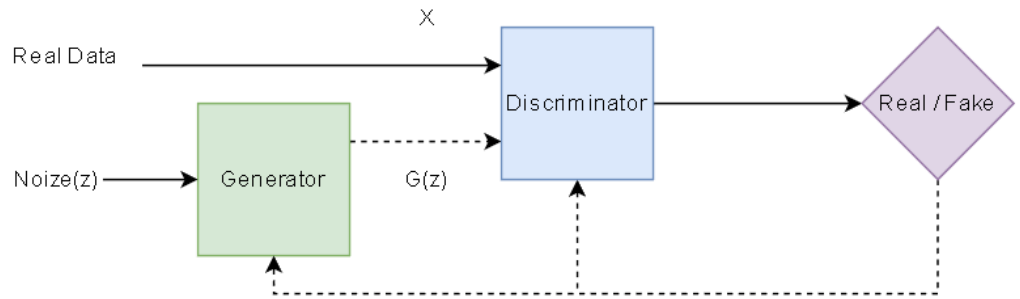

Figure 2.39: Basic GAN architecture.

This study utilises a GAN model, specifically Google Tacotron, to generate synthetic audio.

2.5.4.1 Tacotron 2

The progression from Google WaveNet to Tacotron 2 is demonstrated in Figure 2.40 (Shen, et al., 2017; Yamamoto, Song, & Kim, 2020).

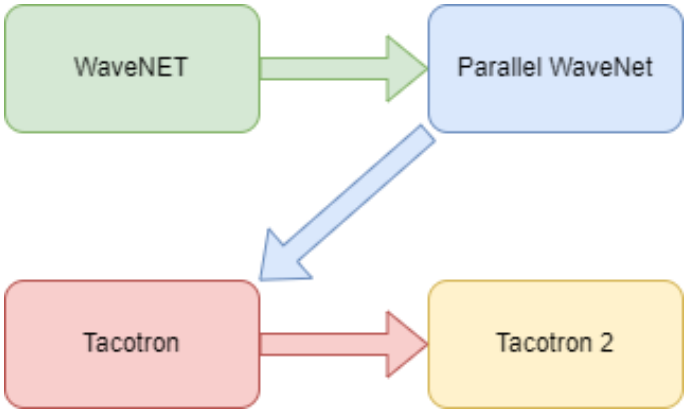

Figure 2.40: Tacotron 2 evolution.

The Tacotron 2 architecture is depicted in Figures 2.41 and 2.42. The Tacotron 2 architecture is based on an LSTM framework and utilises an Encoder-Attention-Decoder model. It transforms text into Mel spectrograms by embedding characters and phonemes through an encoder network. This embedding is then processed by a stack of convolutional layers. Following this, the output is fed into a bi-directional LSTM. The decoder, which is an autoregressive LSTM, generates one slice of the Mel spectrogram at a time. The attention model plays a crucial role by linking the encoder and decoder, guiding the decoder on which sections of the encoded text to focus on during each call (Shen, et al., 2017; Shen, et al., 2017; Goodfellow, et al., 2014). Tacotron 2 is primarily an RNN model that incorporates attention mechanisms to convert input text into a spectrogram. This spectrogram can then be transformed into speech using a vocoder. Common vocoders used for this purpose include WaveNet and the traditional Griffin-Lim algorithm.

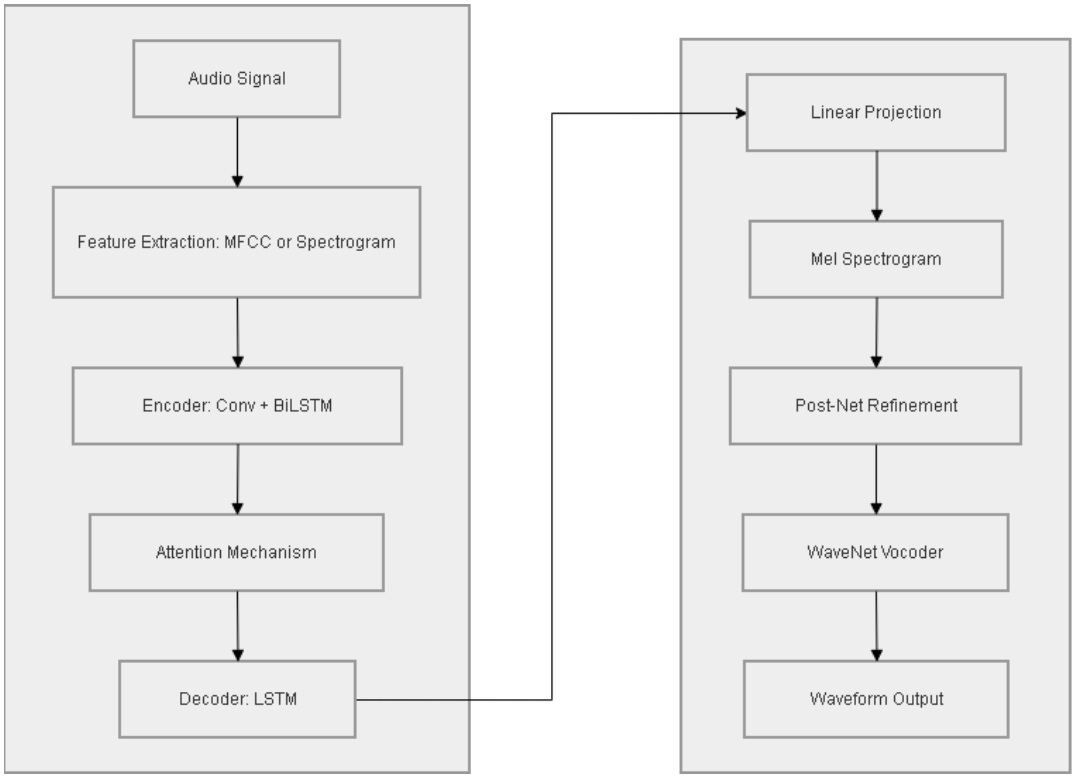

Figure 2.41: The Tacotron 2 framework combined with a WaveNet vocoder *(Shen, et al., 2017)*.

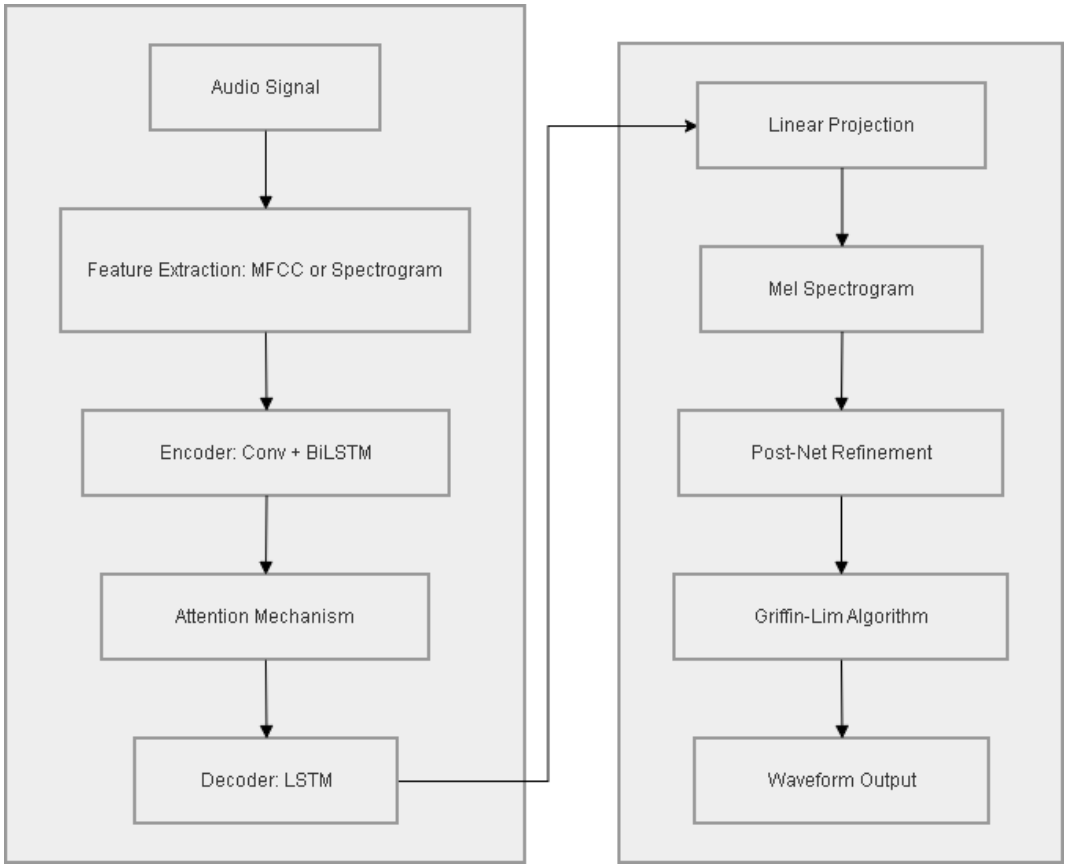

Figure 2.42: Tacotron 2 system design utilising Griffin-Lim.

A range of optimisers was evaluated for different model variations, with the Adam optimiser yielding the most effective results and enabling rapid model convergence. Before backpropagation in each mini-batch, the gradient was reset to zero to prevent unnecessary buildup of gradients. Both WaveNet and Griffin-Lim were assessed for speech synthesis.

2.5.5 Dendritic NN Layers

In recent years, there has been a notable surge in interest towards developing artificial neural network architectures inspired by biological processes. One notable advancement in the field is the incorporation of dendritic computations into artificial neuron models, leading to the development of "dendritic neuron layers." Unlike standard artificial neurons, which are mainly based on point neurons and cannot replicate the intricate dynamics of dendritic trees found in biological neurons, dendritic trees process synaptic inputs in a highly sophisticated, often nonlinear way. This architectural innovation allows for more advanced input-output transformations, going beyond simple summation. By integrating dendritic neuron layers into neural networks, we can significantly enhance computational capabilities, enabling more subtle and efficient pattern recognition and data processing. These layers aim to connect traditional deep learning models with the complex dynamics typical of biological neural circuits. However, the practical use of dendritic neuron layers, their benefits compared to existing architectures, and strategies for effective training are still under investigation, highlighting the need for further research (Ji, Tang, Zhao, Tang, & Todo, 2022).

A dendritic neuron model captures the complex processing capabilities of dendritic trees found in real neurons. In these models, dendritic trees—branched extensions that receive signals from other neurons—can carry out independent calculations before sending their outputs to the neuron's soma (or cell body). The soma then compiles the results from these dendritic computations to determine the neuron's overall response. This approach is different from the perceptron model, which typically uses a straightforward summation followed by an activation function. The dendritic neuron model allows for multiple layers of computation within a single neuron, where each branch performs its own computation and integrates the results back at the soma. As a result, learning in dendritic models can be more complex than in perceptrons, as it may involve adjusting weights not only along the dendritic branches but also at the soma. The concept is illustrated in Figure 2.43. The model of dendritic neurons more accurately represents the complexity of biological neurons, enabling intricate computations within individual neurons and allowing for the modelling of nonlinear computations that a single perceptron cannot achieve (Egrioglu, Ba¸s, & Chen, 2022).

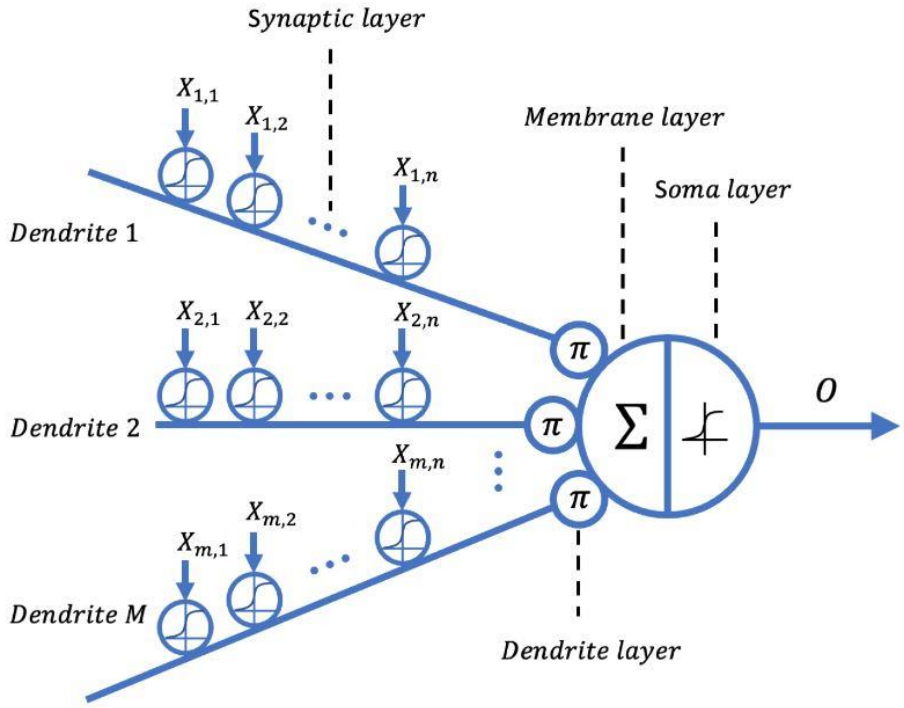

Figure 2.43: Dendritic Neuron Model *(Tao, et al., 2022)*.

When comparing the two models developed to mimic neural computation, the perceptron stands out for its simple, linear approach. On the other hand, the dendritic neuron model delves into the complex nonlinear and hierarchical processing characteristics found in biological neurons. This distinction is grounded in how each model operates: the perceptron utilises a step function or threshold activation function to establish a piecewise-linear decision boundary. In contrast, dendritic models utilise sophisticated mathematical functions to account for nonlinearities and hierarchical processing. These models are capable of both spatial and temporal integration, enabling nonlinear aggregation of inputs across different dendritic branches. This capability for nonlinear summation is especially beneficial for Speech Emotion Recognition (SER), as it effectively captures intricate relationships, supports hierarchical processing, integrates features spatially, interprets temporal dynamics, handles nonlinear acoustic patterns, and improves model expressiveness. Conversely, the perceptron's linear design limits its effectiveness in these areas. The fundamental formula of the basic perceptron is (Chen, et al., 2024):

$$y = f(w \cdot x + b) \tag{2.32}$$

where:

y is the output.

f is an activation function can be a simple step function for basic perceptrons, but in more advanced networks, it might be a sigmoid, tanh, ReLU, or other types.

w is the weight vector.

x is the input vector.

b is the bias.

The dendritic layer utilizes a multiplicative approach to process the outputs generated by various synapses in the synaptic layer. In this model, the synaptic layer takes in signals from adjacent neurons and individually processes each signal using a sigmoid function. After this initial processing, the dendritic layer manipulates the outputs from the synaptic layer using a multiplicative technique. The membrane layer then aggregates the processed signals from each dendritic branch through a summation function. Finally, the somatic layer applies another sigmoid function to the combined signals from the membrane layer, resulting in the final output signal that defines the behavior of the dendritic neuron model. Although modeling dendritic computations can be complex and is subject to change as the field evolves, a simplified overview might resemble this structure (Ji, Tang, Zhao, Tang, & Todo, 2022):

$$y = f\sum_i g(w_i.x_i) + b) \tag{2.33}$$

$f(\cdot)$ indicates the activation function applied to the sum of the weighted inputs along with the bias, representing the activation function of the soma.

$\sum_i$ represents the total of all the inputs (i).

g represents the activation function applied to each weighted input.

The selection of an activation function can vary depending on factors such as the nature of the data and the structure of the network. Two commonly utilised activation functions are:

1) Sigmoid Function for Activation:

$$\text{sigm a}(z) = \frac{1}{1+e^{-z}} \tag{2.34}$$

2) Rectified Linear Activation Function (ReLU):

$$\text{ReL U}(z) = max(0, z) \tag{2.35}$$

wi are the weights linked to the i-th dendritic branch.

xi is input to the i-th dendritic branch. The output of a neuron is calculated as the weighted sum of its inputs plus a bias term. Neuron Output:

$$\sigma(\sum_i w_i \cdot x_i + b) \tag{2.36}$$

The formula $y = \sigma(\sum_i g(w_i \cdot x_i) + b)$ correctness requires the following. Demonstrating that the desired computation is accurately represented. In particular for neural networks. Here's a summary of the proof:

1) Temporal and Spatial Summation:
   The summation expression $\sum_i g(w_i \cdot x_i)$ captures how signals are pooled across the dendritic structure. An optionally applied transfer function $g(\cdot)$ can inject non-linear behaviour, allowing the network to model richer dependencies present in the incoming data:
2) Offset Parameter (b):
   The offset parameter (b) shifts the unit's baseline firing level, accommodating influences that lie beyond the direct contribution of synaptic connections.
3) Squashing Function (σ):
   The logistic transfer function $\sigma(z) = \frac{1}{1+e^{-z}}$ imposes non-linearity and constrains its output to the interval $(0, 1)$, which lends itself naturally to two-class decision tasks.
4) Input Admissibility:
   The expression remains well-behaved provided the argument fed into the logistic function $\sigma(\cdot)$ stays within a sensible numerical interval. Suitable initialisation and updating of the connection strengths w_i and offset b throughout learning preserve this condition.
5) Learning and Refinement:
   How well the model performs hinges on its ability to refine itself as training progresses. This entails iterative tuning of the connection strengths w_i and offset b to shrink the gap separating the network's predictions $\hat{y}$ from the ground-truth labels y.
6) Iterative Parameter Optimisation:
   The learning procedure generally relies on first-order gradient-based solvers or comparable iterative schemes. Reliable convergence behaviour is what gives this approach its practical value, ensuring that the tunable coefficients of the network settle at configurations that meaningfully minimise the chosen objective $\mathcal{L}(\theta)$. The pooling stage spans every branch of the dendritic tree, highlighting a key distinction: whereas a single perceptron evaluates one weighted combination $\sum_i w_i x_i$ followed by a single nonlinearity $\sigma(\cdot)$, each dendritic branch instead carries out its own localised weighted summation paired with its own nonlinear mapping. The branch-level results are then combined and further integrated at the cell body before the unit's output is produced. These formulations should be read as simplifications: the underlying mathematics, especially within the dendritic compartment, can be considerably richer in practice. Such richness may take the form of varied nonlinear operators, alternative integration rules, and additional structural subtleties that mirror our progressively deepening grasp of how dendrites contribute to neural computation. It is also worth noting that the precise functional and computational role of real dendrites in biological tissue remains under active investigation. As a result, today's artificial neural network constructions can only be expected to reflect a narrow subset of the full range of phenomena exhibited by their biological counterparts.

2.5.6 Capsule Networks

Capsule Networks (CapsNets) emerged from the deep learning community as a fresh architectural proposal aimed at addressing several shortcomings inherent in standard convolutional models. The approach, originally put forward by Geoffrey Hinton together with

his research collaborators, centres on the notion of a "capsule"—a small grouping of units that jointly identify visual entities within a structured perceptual hierarchy and encode each entity's positional and part-whole properties as vector-valued activations. Unlike conventional convolutional architectures, whose representations tend to falter under variations in viewpoint, rotation, or object size, CapsNets retain the underlying part-whole structure, which translates into stronger results on problems where reasoning about spatial layout is essential (Patrick, Adekoya, Mighty, & Edward, 2022).

A defining characteristic of CapsNets is their dynamic routing protocol — often termed "routing-by-agreement" — through which lower-tier capsules selectively dispatch their outputs to the most compatible higher-tier capsules in the layer above. This scheme stands in for the pooling operations relied upon by traditional convolutional pipelines and enables information flow to adapt itself in response to the specific input being processed, departing in a meaningful way from the rigid, predetermined operations that defined earlier neural designs (Pawan & Rajan, 2022). Owing to this adaptability, CapsNets have produced promising results across a spectrum of vision tasks, from straightforward image categorisation to the interpretation of richly populated scenes. They consequently offer a credible avenue toward neural systems that are simultaneously more transparent in their reasoning and more robust under perturbation, qualities that remain highly sought after as the broader machine learning field continues to evolve (Wu, et al., 2019; Shahin, 2022).

2.5.7 Hybrid Architectures

A hybrid NN is a mixture of CNN & RNN networks. A CNN is a typical feed-forward network where an RNN feeds data back onto itself. It is worth noting that RNN networks are best suited for sequential data, whereas CNNs are more suitable for spatial data. CNNs & RNNs both have strengths and shortcomings. The aim is to use the best features for a specific purpose.

2.5.8 Convolution Neural Network Layers

2.5.8.1 Convolution layer

The layer where data interaction happens is the convolution layer. Examples of input data are 1D time series and images. A sliding window is applied to the input, known as filtering. The resulting output is the multiplication result of the sliding window elements.

2.5.8.2 Activation Layer

Activation functions can be divided into two categories, namely linear and non-linear. Activation functions are usually placed at a network's end or between convolution layers. Nonlinear activation functions are typically placed between successive convolutional layers.

2.5.8.3 Pooling layer

The pooling layer's main aim is to downsample the data features. The approach leads fewer training parameters. By having fewer parameters, overfitting is prevented. The operation involves sliding a two-dimensional filter over the data, thus reducing the dimensions. The

types of pooling mainly used are max-pooling and average pooling.

2.5.8.4 Fully Connected Layer

A fully connected layer is introduced to flatten the output. The fully connected layer is mainly used for classification. A higher-dimensional dataset is flattened into a one-dimensional data representation.

2.5.8.5 Batch Normalisation Layer

If instability is detected in any layer, batch normalisation is applied. Usually, the output of activation functions is normalised. In layman's terms, values are rescaled to be all in the same range.

2.5.9 LSTM layer

An RNN is an NN where the output is fed back into the input for the current step. RNN has the limitation of only being able to look back a few steps. A subcategory of RNN, namely LSTM, is introduced to solve the issue. LSTM also addresses the vanishing gradient and exploding gradient problem. LSTM is perfect for speech-related tasks because speech is continuous in the time domain. As each frame is processed, LSTM increases information between frames and assists in recognising temporal continuity.

2.5.10 Fusion of CNN and LSTM

An example of a hybrid CNN and LSTM network is shown in Figure 2.44. The input is a CNN with a convolution layer, an activation layer, and a pooling layer. The output of the pooling layer is fed into the LSTM input. The fully connected layer finally classifies the LSTM result. The layer configuration can be optimised and changed depending on the requirement. (Qazi & Kaushik, 2020; Zhao, Mao, & Chen, 2019).

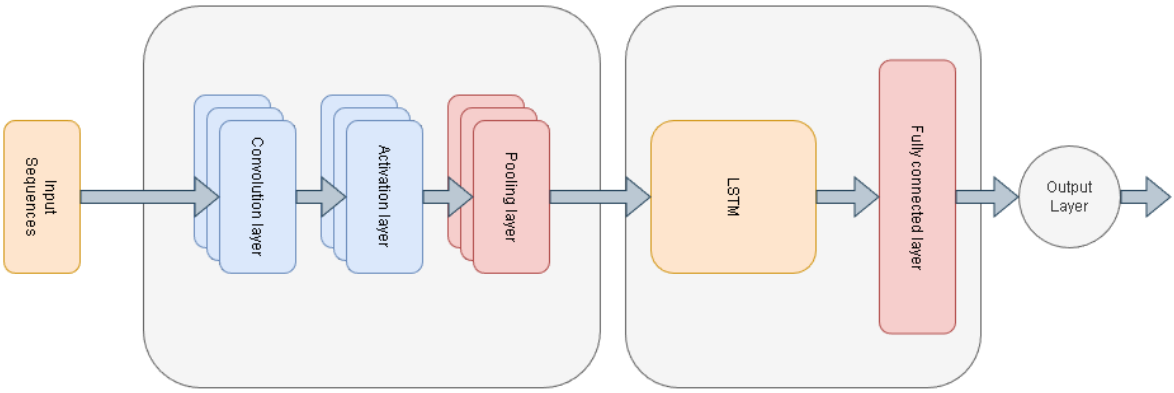

Figure 2.44: Typical CLSTM Hybrid architecture.

2.5.11 Probability / Fuzzy Outputs

In terms of probability prediction, the TensorFlow component is used. More specifically, the Keras "model. predict" function. A network is trained with a specific number of classes; in the configuration, eight are used. The types are the emotions from the Plutchik wheel of emotions. These emotions are Anger, Anticipation, Disgust, Fear, Joy, Sadness, Surprise and Trust

(Looney & Dascalu, 2007).

Consider the following output illustrated in Table 2.4.

Table 2.4: Example Keras model emotion probability outputs (illustrative sample data).

Emotion	Result
Anger	55%
Anticipation	45%
Disgust	2%
Fear	2%
Joy	2%
Sadness	2%
Surprise	1%
Trust	1%

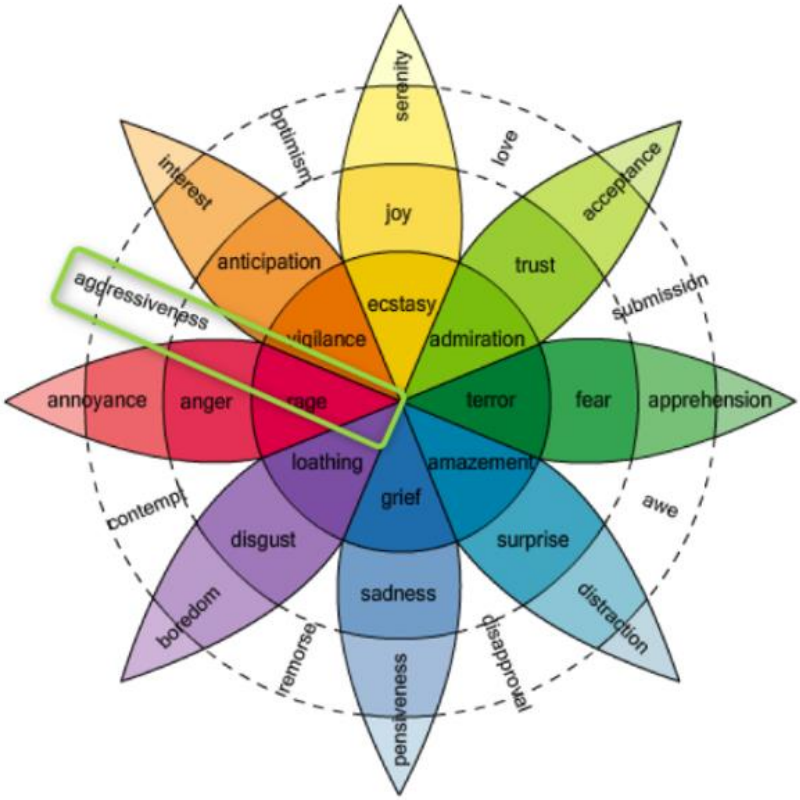

Figure 2.45: Plutchik's Wheel of Emotions (*The image file is taken from Wikimedia Commons.*)

From this, one can infer that the emotions of anger and anticipation are involved, and the combination of the two is aggressiveness. The concept is illustrated in Figure 2.45. With a well-trained model, it tends to lean much more toward one emotion.

In essence, the probabilities relate directly to the values of the given activation function. In the configuration, Softmax was used.

2.5.12 Supervised and Unsupervised Learning

The main difference between the two approaches is the use of labelled datasets. Supervised learning uses labelled input and output data, while unsupervised learning does not.

When a network is trained iteratively, knowing what the output should be is known as supervised learning. Generally, supervised models tend to be more accurate than unsupervised models. The drawback of supervised learning is the requirement for intervention to label the data (Bojja, 2025).

In contrast to supervised learning, unsupervised models discover the structure of the unlabelled data on their own. Human intervention is still required to validate the detected variables.

2.5.12.1 Supervised Learning

The use of labelled datasets is known as supervised learning. When training on these datasets, the algorithm is supervised to accurately classify the data outcomes. The model can measure its accuracy over time using input and output labels (Obaido, et al., 2024).

There are two problems with supervised learning, namely classification and regression:

2.5.12.1.1 Classification

An algorithm is used to assign test data into specific categories, known as classification. Examples of classification algorithms are linear classifiers, support vector machines, decision trees and random forests (Kinasih, Handayani, Ardiansah, & Damanhuri, 2024).

2.5.12.1.2 Regression

Another type of supervised learning is regression. Regression is an algorithm used to understand the relationship between dependent and independent variables. Regression models help predict numerical values based on different data points. Examples of regression algorithms are linear, logistic, and polynomial regression (Lu, et al., 2025).

2.5.12.1.3 Unsupervised Learning

When using unsupervised learning, machine learning algorithms cluster unlabeled datasets. Such algorithms uncover hidden patterns in data without requiring human intervention.

Unsupervised learning performs three main tasks: clustering, association and dimensionality reduction (Chaudhry, et al., 2023).

2.5.12.1.4 Clustering

A data mining technique called clustering groups unlabeled data based on similarities or differences. K-means clustering algorithms assign similar data points into groups. In K-means clustering, K represents the number of clusters and the granularity of the grouping. Clustering is helpful for image compression and market segmentation (Atif, et al., 2023).

2.5.12.1.5 Association

Another type of unsupervised learning is association. The association task is used to find the

relationship between variables in each dataset. The association task is primarily used for recommendation engines and market basket analysis. In layman's terms, the concept equates to "Customers Who Bought The Item Also Bought" recommendations (Salminen, Mustak, Sufyan, & Jansen, 2023).

2.5.12.1.6 Dimensionality Reduction

When the number of features or dimensions is too high, dimensionality reduction is used. The method reduces the number of data inputs to a manageable size whilst preserving data integrity. The technique is used for data preprocessing. An example includes the usage of autoencoders to remove noise from visual data (Mishra, Reddy, Sandesh, & Pathak, 2021; Delua, 2021).

2.5.13 Underfitting and Overfitting

Overfitting can be described as good performance on the training data and poor generalisation to other data. Underfitting can be defined as poor performance on the training data and poor generalisation to other data.

Overfitting is caused by insufficient training data, a training data set that is too small, training data that contains irrelevant noise, a model that is trained too long on specific data, or a model that is too complex.

Underfitting is caused by the following: Data has garbage noise values, the model has a high bias, and the model is too simple (Pothuganti, 2018; Ying, 2019).

2.5.14 Overfitting

Overfitting can be categorised into three kinds:

2.5.14.1 Training Data Set Too Small

With minimal training data sets, the NN tends to overfit, retaining all training examples. Smaller training sets also have sparse sampling points in high-dimensional space, which can lead to overfitting. Adding random noise to the dataset is one approach to improving the mapping structure (Rather, Kumar, & Gandomi, 2024).

2.5.14.2 Irrelevant, Noisy Training Data

When a model becomes too focused on the specific details and noise present in the training data, it can lead to a decline in its performance on new data. This phenomenon is known as overfitting. As a result, the model learns the random fluctuations and noise in the training set as if they were actual concepts (Wei, et al., 2024).

2.5.14.3 Single Sample Training

Only when training is done with a dataset containing too many similar characteristics are such patterns learned, while other relevant information is disregarded (Wang, Duentsch, Guo, & Khan, 2023).

2.5.14.4 Model Complexity

Model complexity can result in a model that is overly complicated, which may hinder its ability to generalise to new data. Similarly, overfitting can lead to a model that performs well on the training data but poorly when faced with new data (Hu, Chu, Pei, Liu, & Bian, 2021).
Overfitting can be prevented by:

2.5.14.4.1 Early Stopping

In the case of early stopping, the training is paused before the machine learning model has learned to account for the noise in the data. It is essential to get the timing right. If paused too soon, the model will not give accurate results (Bartlett, et al., 2023).

2.5.14.4.2 Pruning

Pruning or feature selection identifies the most critical features within the training set and eliminates those that are irrelevant. Feature selection can be manually adjusted according to the situation. For example, you can examine input parameters such as body structure, face shape, and ear position to predict whether an image depicts a human or an animal. You may prioritise face shape and ignore the shape of the eyes (Cheng, Zhang, & Shi, 2024).

2.5.14.4.3 Regularisation

A collection of techniques that prevent overfitting is called Regularisation. Factors that do not impact prediction outcomes are eliminated and reduced. Regularisation would give a lower penalty value to essential features and a higher penalty value to less essential features (Salehin & Kang, 2023).

2.5.14.4.4 Ensembling

Combining predictions from multiple separate machine learning algorithms is known as an ensemble. Ensemble methods include bagging and boosting. It is called boosting when different machine learning models train one after the other in series. With bagging, the models are trained in parallel (Mienye & Sun, 2022).

2.5.14.4.5 Data Augmentation

Data augmentation involves modifying data slightly for each learning pass. Data can be augmented by simple image manipulation such as rotation or flipping (Services, n.d.).

2.5.15 Underfitting

Underfitting occurs when a model fails to learn the patterns in the training data effectively and cannot generalise to new data well. Underfit models perform poorly on training data and have unreliable prediction outcomes. High bias and low variance are the root causes of underfitting (Aliferis & Simon, 2024).

Underfitting can be categorised into four kinds:

2.5.15.1 Data Noise

A significant reason for underfitting is that the model cannot extract patterns from the dataset due to noise. In essence, it trains on incorrect features. Noise causes the model to exhibit a high bias due to its inability to accurately capture the relationship between the input examples and the target values (Madmoni, Tibor, Nelken, & Rafaely, 2021).

2.5.15.2 The Model Has A High Bias

A high level of bias causes underfitting. The issue occurs when the algorithm fails to capture relevant relationships between features and target outputs. Models with high bias typically include more assumptions about the target function.

2.5.15.3 The Size of The Training Dataset Used Is Not Enough

The model performs poorly with too little data in the training dataset. The reason is that the model cannot capture the relationship between the input and the target values.

2.5.15.4 The Model Is Too Simple

When the model is too simple, then underfitting occurs. In such cases, there are too few or too many features. It can also be a case of the model being regularised too much. If regularised too much, the model cannot learn from the dataset. Simple models tend to have less variance in their predictions but more bias towards wrong outcomes (Belkin, Hsu, Ma, & Mandal, 2019).

Overcoming underfitting:

2.5.15.4.1 Increase The Number of Features In The Dataset

If the number of features in the model is too small, then the model will not be able to detect relevant patterns. The problem can be overcome by adding features and complexity to your data, which can help overcome underfitting (Theng & Bhoyar, 2024).

2.5.15.4.2 Increase Model Complexity

Your model may be underfitting simply because it is not complex enough to capture patterns in the data. Using a more complex model, such as switching from a linear to a non-linear model or adding hidden layers to your neural network, will often help solve underfitting.

2.5.15.4.3 Reduce Noise In The Data

Various noise reduction techniques exist, depending on the type of input data. Image or audio filtering can help remove the noise from the dataset before it is used to train the NN (Healy, Johnson, Pandey, & Wang, 2023).

2.5.15.4.4 Increase The Duration of Training The Data

There is a delicate balance between stopping training too early or too late. It is imperative to

get the timing right. Too much training can lead to overfitting, while too little training can result in underfitting (Hussein & Shareef, 2024).

2.5.15.4.5 Good Fit

To find the best fit for a given model, one needs to look at the performance of a machine-learning model over time using the training data. As training progresses, the error for the model decreases, but if left for too long, unnecessary details and noise are learnt (Li, Rajbahadur, Lin, Bezemer, & Jiang, 2024). A good rule of thumb is to stop training when the error increases. The concept is illustrated in Figure 2.46.

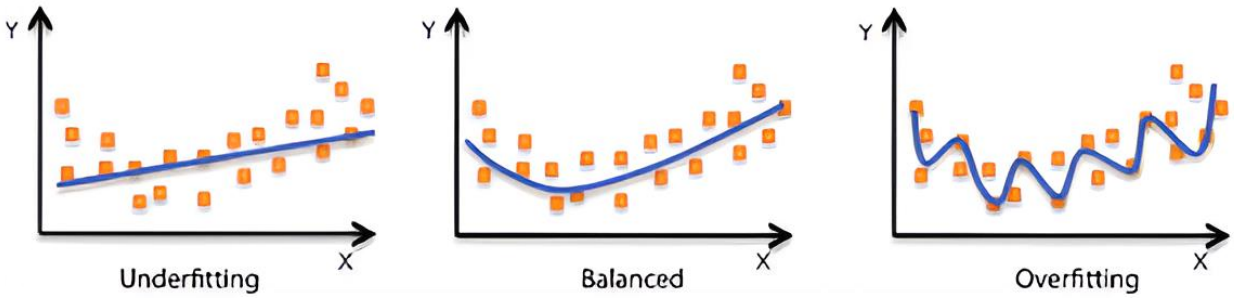

Figure 2.46: Good fit *(docs.aws.amazon.com, 2023)*.

2.5.16 Validation and Performance

From a validation point of view, the aim would be to use 10% of the dataset in question to validate the accuracy. The following evaluation metrics are available (Mishra 2018).

2.5.16.1 Classification Accuracy

Accuracy can be defined by the following:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} \tag{2.37}$$

If there is an equal number of samples belonging to each class, then it works well.

2.5.16.2 Logarithmic Loss

Log loss or logarithmic loss is the process of penalising false classifications.
If N samples belong to M classes, then:

$$Log\ Loss = \frac{-1}{N}\sum_{i=1}^{N}\ \sum_{j=1}^{M} y_{ij} * \log\left(p_{ij}\right) \tag{2.38}$$

where $y_{ij} \in \{0,1\}$represents the true label indicator, and p_{ij}represents the predicted probability that the sample $i$belongs to the class j.

2.5.16.3 Confusion Matrix

A confusion matrix describes the complete performance of the model (Visa, Ramsay, Ralescu, & Knaap, 2011). An illustrative confusion matrix example for 180 samples is presented in Table 2.5. The formula is presented in the equation below.

$$Accuracy = \frac{True\ Positive + True\ Negative}{Total\ sample} \quad (2.39)$$

Table 2.5: Illustrative confusion matrix (hypothetical example, $n = 180$).

n = 180	**Predicted: NO**	**Predicted: YES**
Actual: NO	50	20
Actual: YES	10	100

2.5.16.4 The area under the Curve

One of the most widely used metrics for evaluation is Area Under Curve (AUC).
The concept is illustrated in Figure 2.47. The basic terms are:

True Positive Rate (Sensitivity): The True Positive Rate is TP/ (FN+TP).
True Positive rates refer to the percentage of positive data points that are correctly identified as such.
The formula is presented in the equation below.

$$\text{True Positive Rate} = \frac{\text{True Positive}}{\text{TruePositive+ FalseNegative}} \quad (2.40)$$

True Negative Rate (Specificity): True Negative Rate is TN / (FP+TN).
The formula is presented in the equation below.

$$\text{True Negative Rate} = \frac{\text{True Negative}}{\text{TrueNegative+FalsePositive}} \quad (2.41)$$

False Positive Rate: False Positive Rate is defined as FP / (FP+TN).
The formula is presented in the equation below.

$$\text{False Positive Rate} = \frac{\text{False Positive}}{\text{True Negative+False Positive}} \quad (2.42)$$

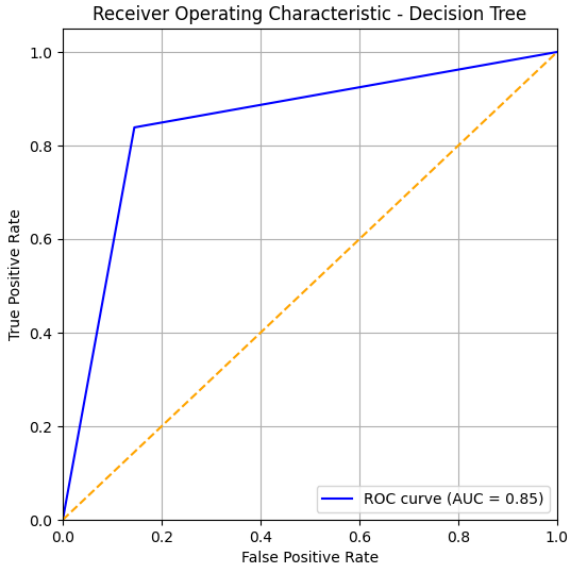

Figure 2.47: AUC (*Area Under the Curve*).

2.5.16.5 F1 Score

The harmonic mean between precision and recall is called the F1 score. The formula is shown in the equation below.

$$F1 = 2 * \frac{1}{{}^{1}/_{precision} + {}^{1}/_{recall}} \tag{2.43}$$

2.5.16.6 Mean Absolute Error

The Mean Absolute Error is the average difference between the Predicted Values and the Original Values. The formula is shown in the equation below.

$$Mean\ Absolute\ Error = \frac{1}{N}\sum_{j=1}^{N}|y_j - \hat{y}_j| \tag{2.44}$$

2.5.16.7 Mean Squared Error

Mean Squared Error is very similar to Mean Absolute Error. The difference is that MSE takes the average of the squares of the differences between the original and predicted values.
The formula is presented in the equation below.

$$Mean\ Squared\ Error = \frac{1}{N}\sum_{j=1}^{N}(y_j - \hat{y}_j)^2 \tag{2.45}$$

2.5.17 Comparative Analysis of Related SER Studies

Convolutional Neural Networks (CNNs) are highly effective for extracting spatial features from structured representations of audio signals, such as spectrograms and Mel-frequency cepstral coefficient (MFCC) maps. Their convolutional layers enable the detection of local patterns, such as frequency bands and formant structures, which are important for identifying emotion-related characteristics in speech. However, CNNs operate primarily on fixed input

windows and do not inherently model temporal dependencies across sequences. As a result, they may fail to capture the dynamic evolution of emotional cues over time, such as changes in pitch, intensity, and speaking rate, which are critical for accurate speech emotion recognition (SER).

In contrast, Long Short-Term Memory (LSTM) networks are specifically designed to model temporal dependencies in sequential data. By maintaining memory cells and gating mechanisms, LSTMs can capture long-range temporal relationships and contextual information within speech signals. These characteristics make them well-suited for analysing the progression of emotional states over time. However, LSTMs process input sequences in a largely sequential manner and do not explicitly exploit local spatial correlations within the feature space. Consequently, they may overlook fine-grained spectral patterns that are effectively captured by convolutional operations.

Hybrid architectures, such as Convolutional LSTM (CLSTM) models, address these limitations by integrating the complementary strengths of CNNs and LSTMs. In such architectures, CNN layers are typically employed as front-end feature extractors to learn spatial representations from input features, while LSTM layers are used to model temporal dependencies within the extracted feature sequences. This combination enables the model to simultaneously capture local spectral characteristics and long-term temporal dynamics, thereby improving performance in SER tasks. The integration of spatial and temporal modelling is particularly beneficial in complex and multilingual settings, where emotional expression varies across both time and frequency domains.

To contextualise these architectures within the broader SER literature, recent studies have reported strong performance using both hybrid and transformer-based approaches. Table 2.6 presents a comparative analysis of representative SER studies, including their datasets, architectures, reported performance metrics, and identified limitations.

Table 2.6: Comparative analysis of recent SER studies

Study	Dataset	Language(s)	Architecture / Method	Reported Accuracy	Reported F1 / UAR	Key Contribution
(Pepino, Riera, & Ferrer, 2021)	IEMOCAP	EN	wav2vec 2.0	72.10%	UAR	Self-supervised SER
(Wang, Boumadane	IEMOCAP	EN	HuBERT (fine-	73.00%	WA	Transformer-based SER

, & Heba, 2021)			tuned)			
(Zhao, Mao, & Chen, 2019)	EmoDB	DE	CLSTM	95.90%	–	Hybrid CNN-LSTM
(T., Mustaqeem , & S., 2020)	EMO-DB / IEMOCAP	DE / EN	Deep-Net CNN	92.0 / 77.0%	–	Efficient CNN model
(Naderi & Nasersharif , 2023)	EMO-DB / IEMOCAP	DE / EN	Wav2Vec2 + Attn	95.4 / 76.9%	–	Attention + SSL fusion
Norval and Wang (this study)	Multilingual	Multi	DenCaps	71.91%	Macro F1: 0.69	Multilingual SER (Afrikaans)

The studies presented in Table 2.6 indicate that high SER accuracy is often achieved on benchmark datasets such as IEMOCAP and Berlin EmoDB. Transformer-based models, including wav2vec 2.0 and HuBERT, benefit from large-scale pre-training and demonstrate strong performance in single-language settings. Similarly, hybrid CNN-LSTM architectures achieve high accuracy by combining spatial and temporal modelling capabilities. However, these results are typically obtained under controlled conditions, with limited variability in language, speaker diversity, and recording environments.

A key limitation across the reviewed literature is the reliance on single-language corpora and relatively small or controlled datasets. Although these studies demonstrate strong performance, they do not adequately address multilingual generalisation or low-resource language scenarios. In addition, performance metrics are not consistently reported across studies: some report accuracy alone, others include weighted accuracy (WA) or unweighted average recall (UAR), and fewer report macro-F1 scores. This inconsistency complicates direct comparison between models and highlights the need for more comprehensive evaluation frameworks.

This study addresses these gaps by focusing on a multilingual SER configuration that includes Afrikaans as a low-resource language. Unlike prior work that primarily evaluates models on single-language datasets, the proposed approach evaluates CNN, CLSTM, and DenCaps architectures across seven languages and eight emotion classes. The DenCaps model achieves a best accuracy of 71.91%, with a mean accuracy of 0.7100 and a macro-F1 score of 0.6917, demonstrating robust performance under more challenging and realistic multilingual

conditions.

A more detailed empirical comparison between the proposed DenCaps model and published SER approaches is presented in Chapter 5 Table 5.27.

2.6 MFCC

The Mel-frequency cepstrum (MFFC) represents the short-term power spectrum of a sound based on a linear cosine transform of a log power spectrum on a nonlinear Mel scale of frequency. The concept is illustrated in Figure 2.48. The coefficients that make up an MFC are called the Mel-frequency cepstral coefficients (MFCCs). For the MFCC, the frequency bands are equally spaced on the mel scale. The mel scale, also known as the melodic scale, is a scale that approximates the human auditory system's response. A standard scale is more linearly spaced. By utilising the frequency-warping mechanism, the audio is better represented (Hight & Eljhani, 2013; Prabakaran & Sriuppili, 2021; Fahmy, 2010).

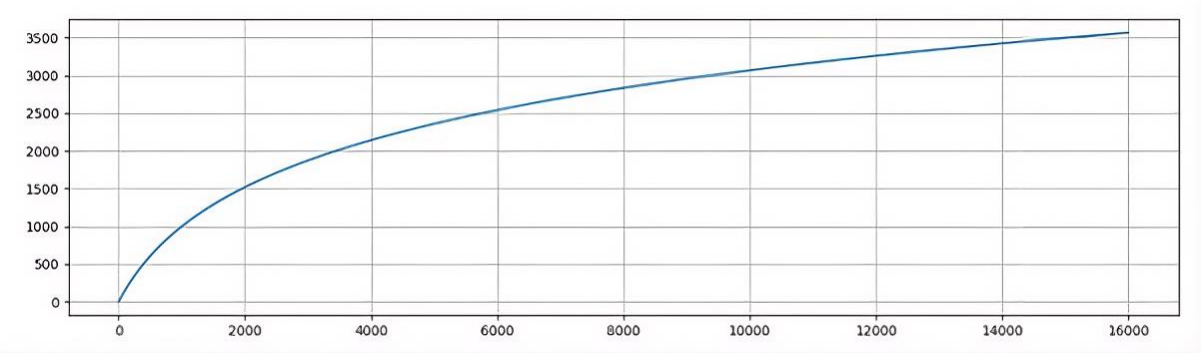

Figure 2.48: Mel Filter Scale (*Generated using Python*).

2.6.1 Steps

The following steps need to be applied to produce an MFCC. The full process is illustrated in Figure 2.49. The digital input is processed through various steps, resulting in an MFCC that is used to train the NN.

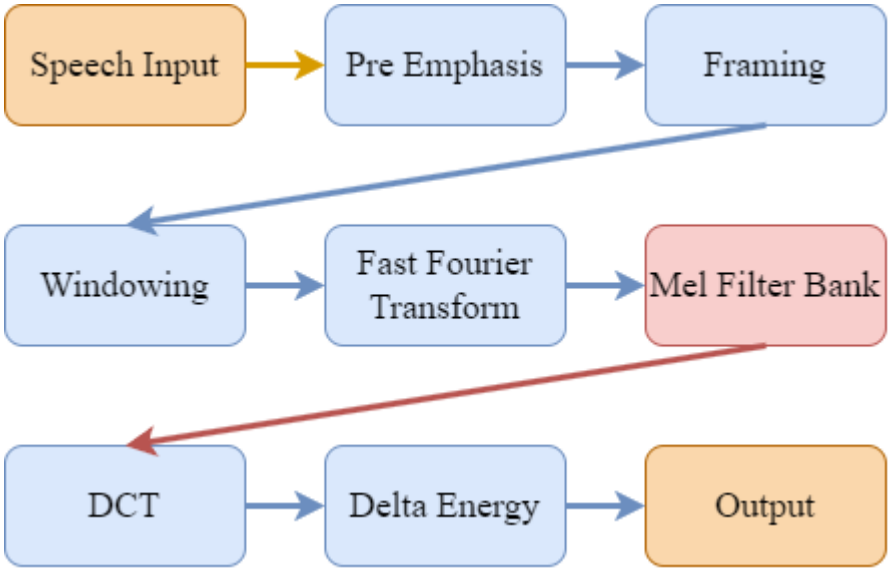

Figure 2.49: MFCC Steps.

2.6.1.1 Input

Analogue sound clips are sampled and stored in the 16-bit format at 44.1 kHz. A typical digitised sound clip is illustrated in Figure 2.50.

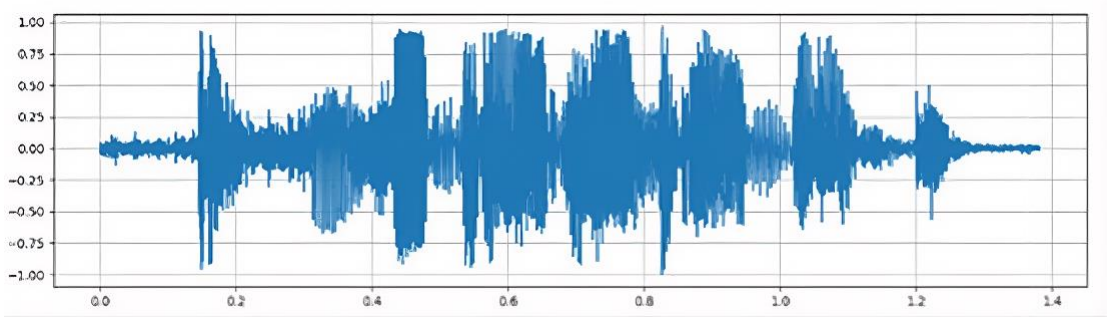

Figure 2.50: Digitised Sound Clip (*Generated Using Python*).

2.6.1.2 Pre-emphasis

The digital audio is processed by a mechanism called pre-emphasis. The digitised speech signal s[n] is positioned through a low-order bypass filter to flatten the signal spectrally. The process is illustrated in Figure 2.51. The goal of the filter is to emphasise higher frequencies.

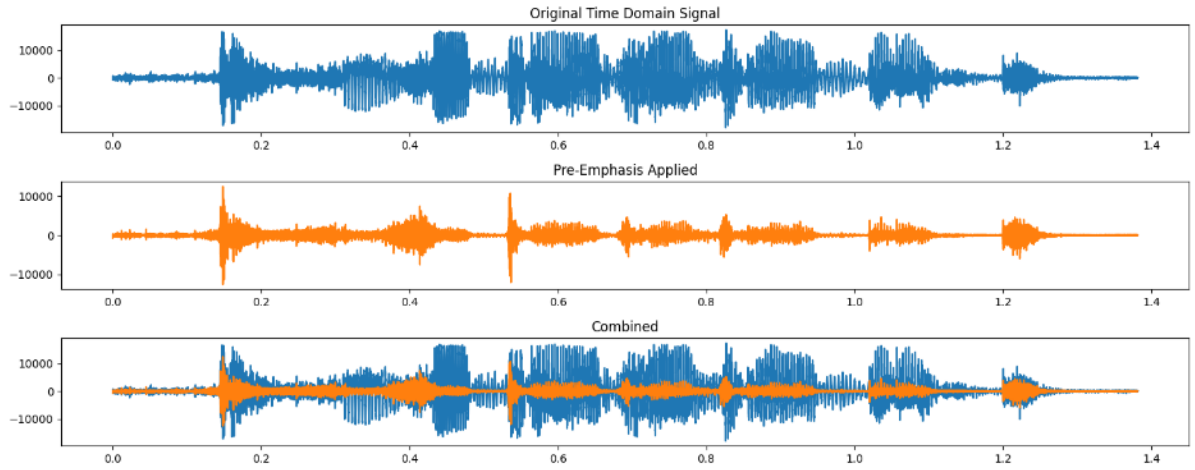

Figure 2.51: Pre-emphasis (*Generated using Python*).

2.6.1.3 Framing

Next, signal framing or segmentation is applied to the signal. Essentially, the process involves segmenting an audio signal into smaller blocks. The typical duration of a frame is 20-30ms. There is not enough information when frames are too short; when frames are too long, there is too much information. The framing process is illustrated in Figure 2.52.

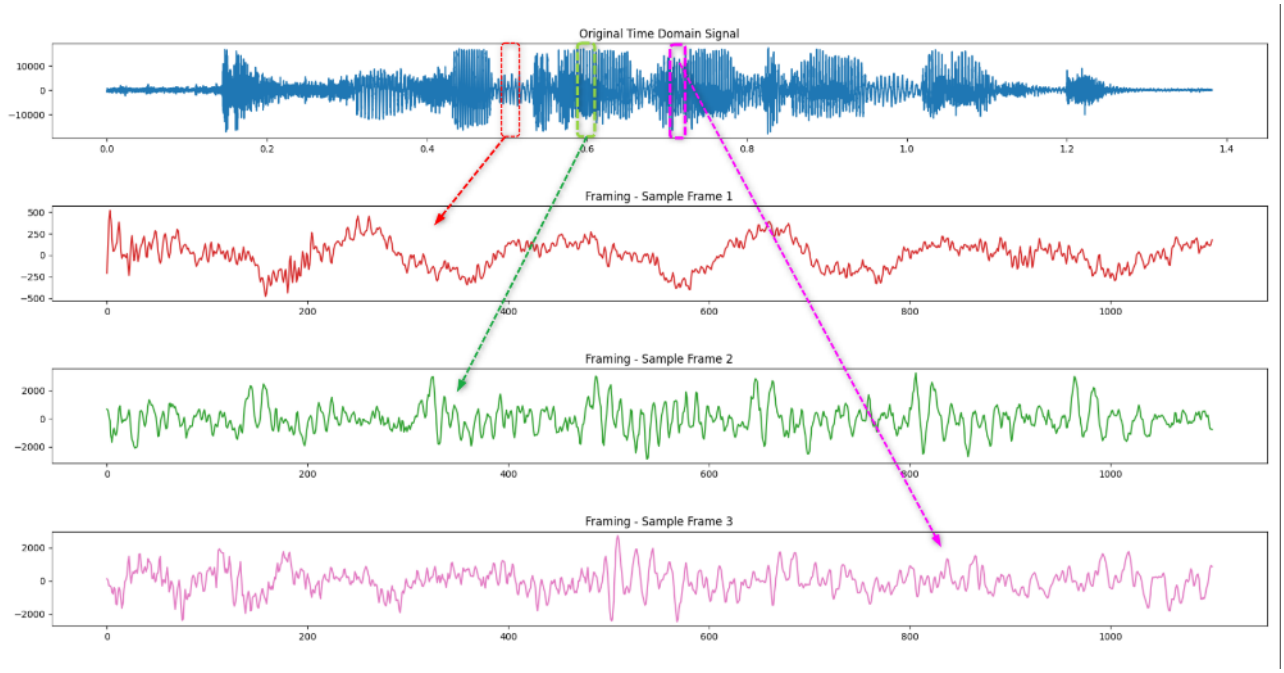

Figure 2.52: Framing (*Generated using Python*).

2.6.1.4 Windowing

The subsequent step is to window each frame. Essentially, windowing is the point-wise

multiplication of a signal in the time domain. Some standard windowing functions include the rectangular window, the Hamming window, the Blackman window, and the flattop window. A characteristic of a good window function is that the resulting signal has a narrow main lobe and low side lobe levels. One of the most used window functions is the Hamming window. The generalised window and filter is illustrated in Figure 2.53. The applied windowing is illustrated in Figure 2.54. The Hamming window formula is presented below.

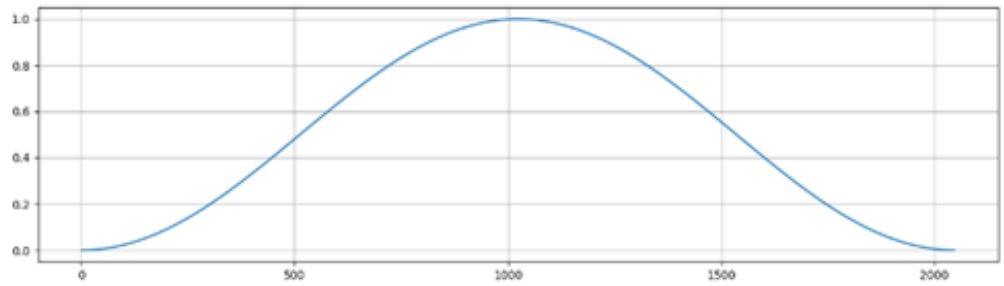

Figure 2.53: Generalised Hamming window (*Generated using Python*).

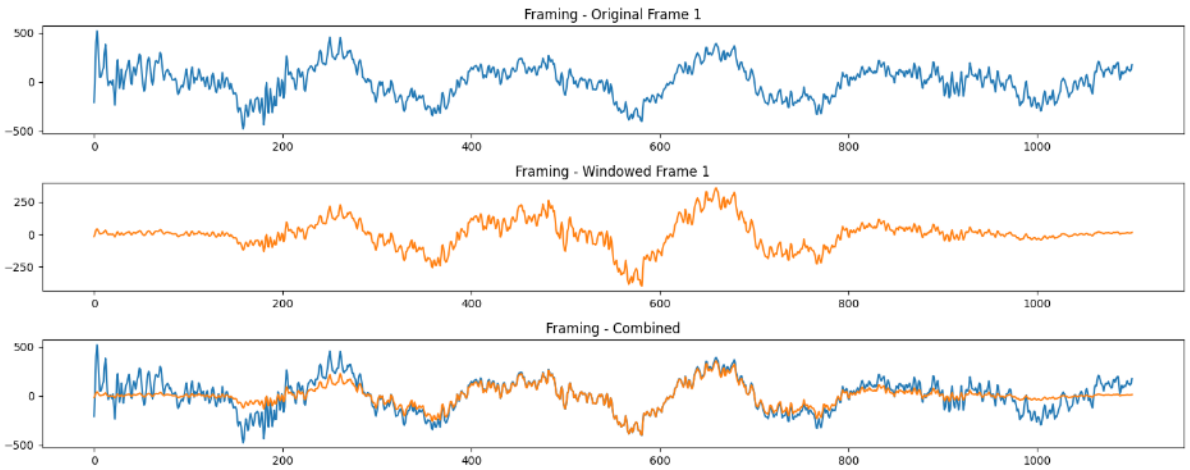

Figure 2.54: Windowing (*Generated using Python*).

$$\omega(n) = 0.5\left(1 - \cos\left(2\pi\frac{n}{N}\right)\right), 0 \leq n \leq \mathrm{N} \tag{2.46}$$

2.6.1.5 Fast Fourier Transform

A transform called the Fast Fourier Transform (FFT) is used to map the data from the time domain to the frequency domain. Illustrated in Figure 2.55.

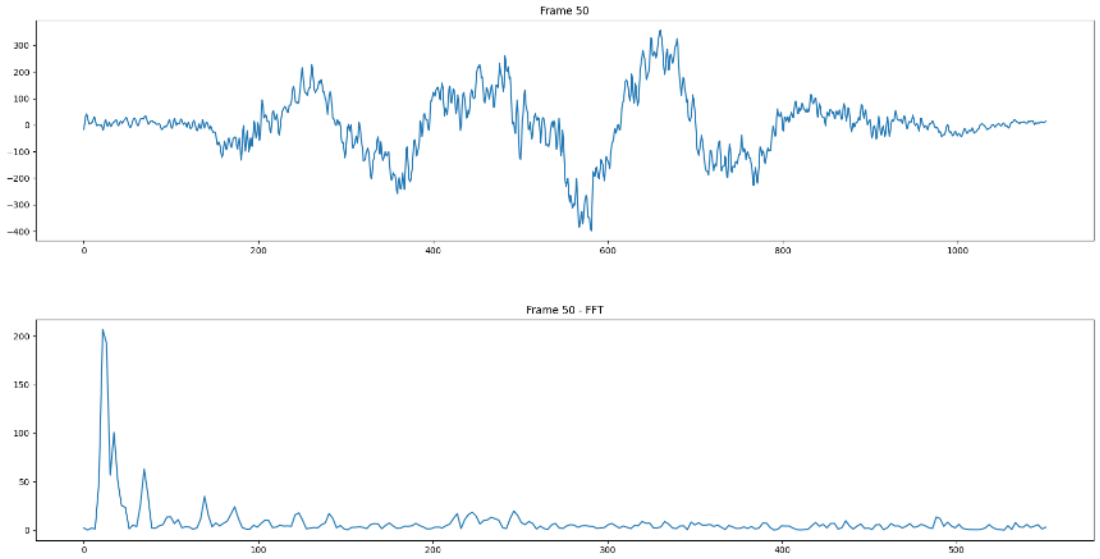

Figure 2.55: Fast Fourier Transform (*Generated using Python*).

The formula for the FFT can be seen below (Oberst, 2007).

$$x[k] = \sum_{n=0}^{N-1} x[n]\, e^{\frac{-j2\pi kn}{N}} \tag{2.47}$$

When comparing the Direct Fourier Transform (DFT) to the FFT, note that the FFT requires substantially fewer computational steps. The conventional method for computing the DFT for $N = 1024$ requires computational effort proportional to N^2, which equals 1,048,576 operations. The complexity is more than 50 times the computational effort required by the Fast Fourier Transform (FFT). The FFT is a technique that sequentially combines progressively larger weighted sums of data samples to efficiently compute the DFT coefficients (Gasmi, 2022; Gan, 2020).

2.6.1.6 Mel Filter Bank

The resulting FFT spectrum contains the relevant spectral information on a linear scale. However, human hearing and perception are less sensitive to frequencies above 1000 Hz. In light of this, the spectrum is warped using a logarithmic Mel scale. To achieve this, a bank of filters distributed equally below 1000 Hz and spaced logarithmically above 1000 Hz is applied to the signal. The frequencies and filter banks are illustrated in Figure 2.56 and Figure 2.57 respectively.

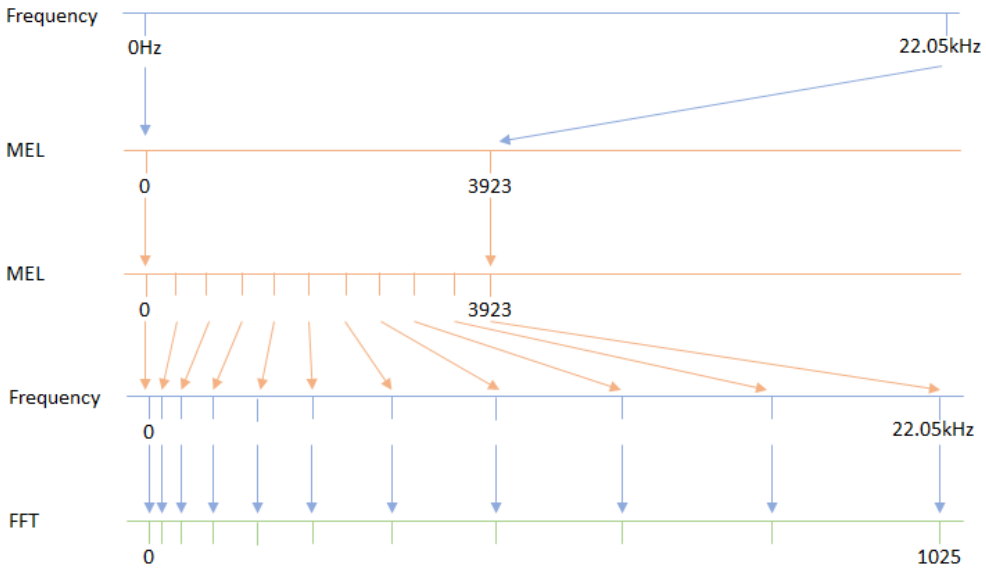

Figure 2.56: Mel Frequencies and Filter Banks *(Ilm, 2018)*.

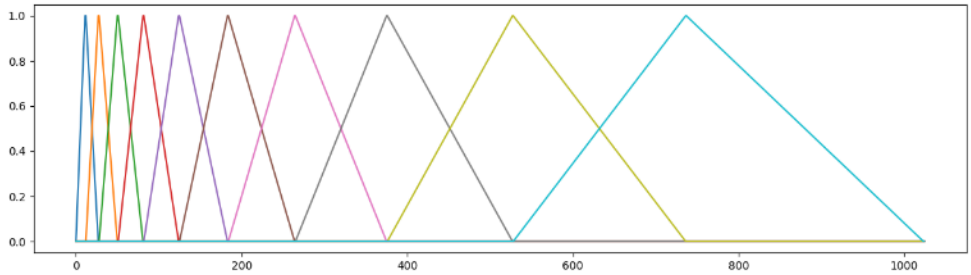

Figure 2.57: Mel Filter Bank (*Generated using Python*).

The Mel filter bank is further normalised by utilising the formula below. The now normalised filter bank is illustrated in Figure 2.58.

$$Mel(f) = 1125 * \ln(1 + f/700) \tag{2.48}$$

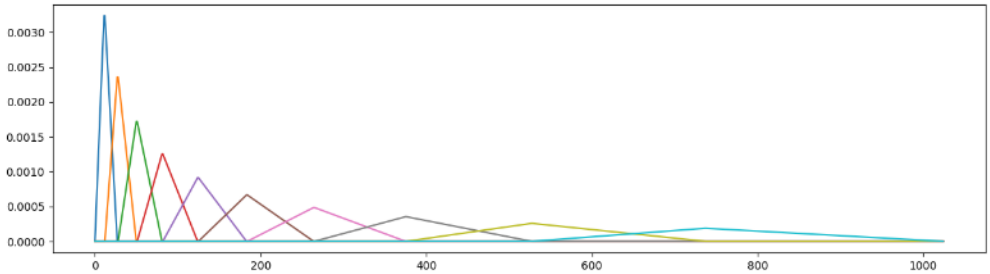

Figure 2.58: Normalised Mel Filter Bank (*Generated using Python*).

2.6.1.7 DCT

Nasir Ahmed first proposed the Discrete Cosine Transform (DCT). The DCT transform is used in various video and audio standards. These include JPEG (Joint Photographic Experts Group) and MPEG (Moving Pictures Experts Group). DCT works by transforming data from the spatial domain to the frequency domain. Data is represented as a sum of sinusoids of varying magnitude. DCT is a high-energy compact. The most significant information is concentrated in a few DCT coefficients. Less important frequencies are discarded (AbuBaker, Eshtay, & AkhoZahia, 2016; Poda, Tamtaoui, & Bouyakhf, 2009). For a 2D N by M image, 2D DCT is defined as:

$$F(u,v) = \left(\frac{2}{N}\right)^{\frac{1}{2}} \cdot \left(\frac{2}{M}\right)^{\frac{1}{2}} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} \Lambda(u) \cdot \Lambda(v) \times \cos \frac{\pi u}{2\mathrm{N}} (2\mathrm{i}+1) \cos \frac{\pi v}{2\mathrm{M}} (2\mathrm{j}+1) \cdot f(i,j) \tag{2.49}$$

The typical compression and decompression steps are illustrated in Figure 2.59.

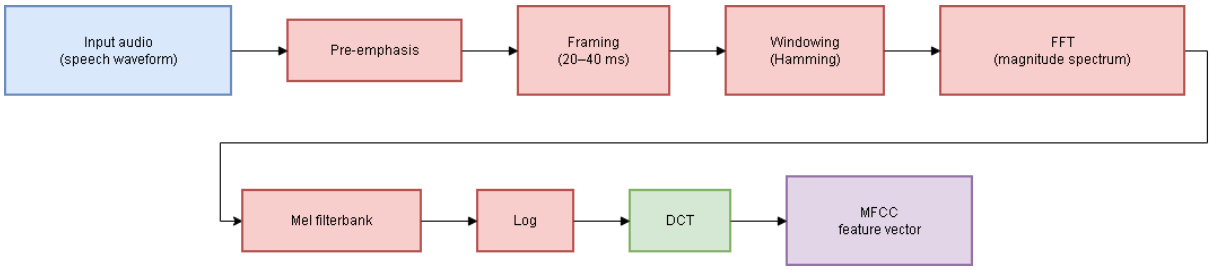

Figure 2.59: DCT compression steps. Adapted from *(AbuBaker, Eshtay, & AkhoZahia, 2016)*.

2.6.1.8 Delta Energy

The delta coefficients are computed using the following formula.

$$d_t = \frac{\sum_{n=1}^{N} n(c_{t+n} - c_{t-n})}{2\sum_{n=1}^{N} n^2} \tag{2.50}$$

where dt is a delta coefficient from frame t. It is computed in terms of the static coefficients ct−n to ct+n. n is usually taken to be 2 (Singh & Ghangas, 2016).

Two variations of delta coefficients are used. These are the delta differential and the delta-delta acceleration.

2.6.1.9 Delta MFCC

The Delta MFCC features can be extracted by taking the first derivative of the MFCC features. The related Delta features represent the change of the cepstral features over time. The advantage of Delta features over MFCC features is that they represent temporal information. A common technique used to differentiate crossing trajectories is the use of delta features.

2.6.1.10 Delta Delta MFCC

If one takes the derivative of the Delta feature, one ends up with the Delta-Delta features. The technique shows the change between frames in the corresponding delta features. Delta-delta features are also referred to as acceleration coefficients. Delta-delta contains more extended temporal context information. Peaks or valleys are better detected and identified using Delta-Delta data.

2.7 Chroma & Tonnetz

2.7.1 Chroma

A Chromagram is a representation of a chroma vector. There are 12 pitch classes in an audio signal—a chroma vector attempts to determine each pitch class's energy amount. A time window is moved across the audio signal to determine pitch energy across the 12 classes. The result is a magnitude spectrum that captures the harmonic and melodic characteristics of the audio. An example of a chromagram are illustrated in Figure 2.60.

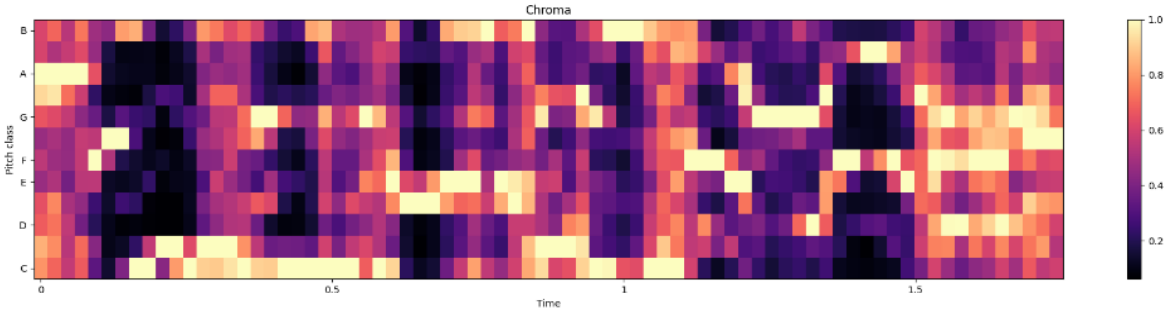

Figure 2.60: Chroma (*Generated using Python*).

2.7.2 Tonnetz

The Tonnetz (German for 'tone network') is a conceptual lattice diagram. The diagram represents tonal space. The phenomenon was first described by Leonhard Euler in 1739 (Calinger, 1999). Tonnetz represents the spatial representations of tonal distance and relationship in layperson's terms. Tonal distribution is evident in Figures 2.61 and 2.62. A sample tonnetz representation is shown in Figure 2.63.

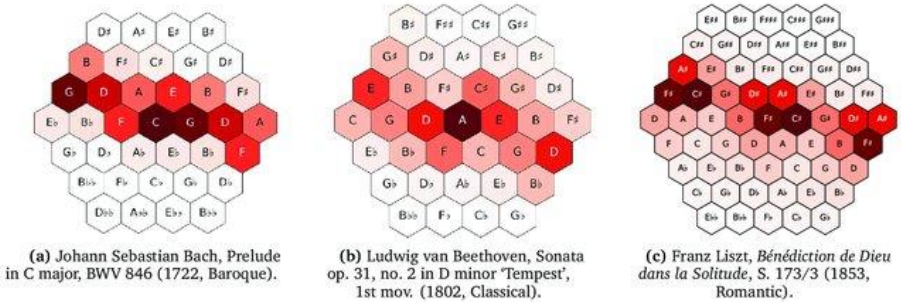

Figure 2.61: Tonal pitch-class distributions plotted as heatmaps on the Tonnetz *(Lieck, Moss, & Rohrmeier, 2020)*.

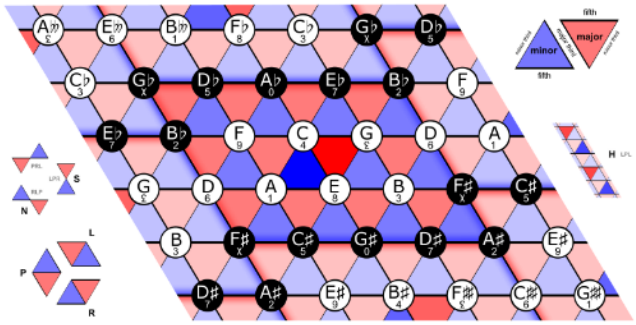

Figure 2.62: Neo-Riemannian (*Wikimedia Commons*).

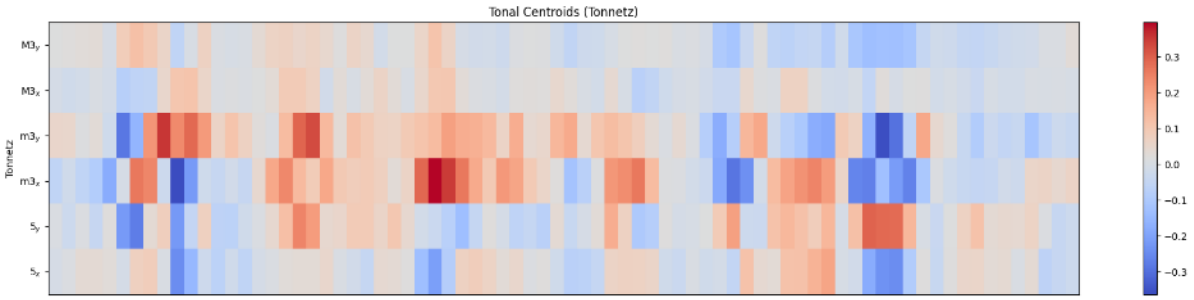

Figure 2.63: Tonnetz (*Generated using Python*).

2.7.3 Comparative Evaluation of MFCC, Chroma, and Tonnetz Features for SER

Mel-Frequency Cepstral Coefficients (MFCC), Chroma features, and Tonnetz representations are widely used feature extraction techniques in audio processing, each capturing different aspects of the speech signal. While these methods are often described independently, their suitability for Speech Emotion Recognition (SER) varies significantly depending on the nature of the task and the characteristics of the data.

MFCC features are the most commonly used representation in SER due to their ability to model the short-term power spectrum of speech in a manner aligned with human auditory perception. By compressing spectral information into a compact set of coefficients, MFCCs effectively capture timbral characteristics such as pitch, tone, and intensity variations, which are strongly correlated with emotional expression. However, MFCCs primarily represent short-time spectral information and may not fully capture long-term temporal dynamics or harmonic relationships inherent in emotional speech.

Chroma features, in contrast, represent the distribution of spectral energy across pitch classes, independent of octave. They are particularly effective in music information retrieval tasks where harmonic structure is dominant. In the context of SER, however, their utility is more limited, as speech signals do not exhibit the stable harmonic patterns found in musical signals. While Chroma features may capture certain pitch-related emotional cues, they are generally less discriminative for speech-based emotion classification compared to MFCCs.

Tonnetz features provide a tonal representation based on harmonic relationships between pitch classes, capturing transitions and relationships within a tonal space. Similar to Chroma, Tonnetz is highly effective in music analysis but less directly applicable to speech, where tonal relationships are less structured and more variable. Consequently, Tonnetz features tend to contribute less to SER performance unless combined with other feature representations.

From a comparative perspective, MFCCs offer the most robust and widely validated feature set for SER, balancing dimensionality reduction with the preservation of perceptually relevant information. Chroma and Tonnetz features may provide complementary information, particularly in capturing pitch-related characteristics, but are insufficient as standalone features for accurate emotion recognition in speech. In the study, MFCC features were selected as the primary input representation due to their proven effectiveness in SER and their ability to generalise across different languages and recording conditions. Chroma and Tonnetz features were considered as supplementary representations; however, their contribution was found to be less significant in preliminary evaluations. The reliance on MFCC aligns with established literature, where it remains the dominant feature extraction technique for speech-based emotion classification tasks.

2.8 Speech Corpus

A number of vocal recording collections have been assembled to support academic investigation in this area. The linguistic coverage of these resources spans a variety of languages, including Mandarin, Russian, English, Italian, French, and German. Some of the most extensively cited collections include AIBO, EMODB, RAVDESS, ENTERFACE, RML, IEMOCAP, AFEW, and MELD. An aggregated resource known as EMOSET, formed by consolidating several individual datasets into a single unified collection, has likewise been developed. Sampling efforts have additionally extended to certain Ethiopian tongues as well as Sepedi, one of the indigenous languages spoken in South Africa. Given that South Africa recognises eleven languages at the official level, the present work targets the construction of a vocal recording collection for one of those languages — specifically Afrikaans.

2.9 Conclusion

This literature review examined the intersection of human emotion, speech processing, and artificial intelligence (AI), with a focus on the commercial potential and technological advancements in SER. The review began by analysing the complexity of human emotions based on Plutchik's wheel of emotions and their critical role in communication. The physiological aspects of speech were also examined, including how various emotional states are generated, processed, and influence speech. The understanding of speech production and emotional modulation provides essential context for studying SER.

The review continued with a comprehensive analysis of audio and digital signal processing (DSP) techniques, emphasising how audio signals are digitised, processed, and filtered. Methods such as the Kalman filter, LMS adaptive filter, and spectral gating were highlighted for their effectiveness in reducing noise and enhancing signal clarity. These methods, paired with advanced audio features like Mel-frequency cepstral coefficients (MFCCs) and delta

features, form the foundational input for SER systems.

Much of the review focused on the various AI and neural network (NN) architectures, particularly hybrid models like CLSTM, which are instrumental in classifying emotions from speech data. The distinctions between supervised and unsupervised learning methods, as well as the challenges of overfitting and underfitting, were also addressed, providing insights into the model validation process.

In conclusion, integrating speech physiology, digital signal processing, and advanced AI architectures forms the core of SER systems. The technological landscape is rapidly evolving, with significant advancements in machine learning and neural networks that enhance emotion recognition capabilities. The review underscores the potential for SER to transform industries such as call centres, gaming, and healthcare, providing a robust foundation for future research in the field.

CHAPTER 3: Compilation of Afrikaans Speech Corpus

A key objective of the thesis is the development of an Afrikaans speech corpus, addressing a significant gap in speech emotion recognition (SER) research. Globally, major languages such as German (EMO-DB), English (SAVEE, IEMOCAP), Italian (EMOVO), and Chinese (MASC) have well-established speech corpus yet resources for African languages, including Afrikaans, remain limited despite the existence of some smaller, lesser-known databases. The compilation of a speech corpus is essential for advancing SER, enabling the study of emotional expression through audio. The corpus can be derived from natural, acted, or elicited speech, sourced from platforms such as podcasts, television series, and radio broadcasts. To circumvent ethical concerns, the study employs CC-licensed material. The resulting Afrikaans corpus serves as the primary dataset for evaluation within the research, with its utility enhanced through data augmentation using Generative Adversarial Networks (GANs) trained on existing samples.

3.1 Introduction

In recent years, speech emotion recognition (SER) has emerged as a pivotal area of research in both academic and commercial sectors. As human-computer interaction (HCI) technologies advance, the integration of emotion recognition into speech systems can transform industries such as call centres, gaming, and healthcare (Yu & Li, 2015). However, despite its vast commercial potential, SER research has mainly been confined to widely spoken languages, such as English, leaving a significant gap in resources and understanding for underrepresented languages, including those spoken in Africa (Abate, Tachbelie, & Schultz, 2020) created (Norval & Wang, 2022).

In South Africa, a nation rich in linguistic diversity, Afrikaans is among the 11 official languages millions speak. Despite its widespread use, there is a conspicuous absence of robust datasets tailored for Afrikaans, especially in the domain of speech emotion recognition. The gap in data hinders the development of accurate SER models for the language, limiting its inclusion in cutting-edge HCI systems and technologies.

The motivation behind the research stems from the pressing need to develop and enhance speech emotion recognition models for African languages, with a specific focus on Afrikaans. The research aims to create a high-quality Afrikaans speech corpus categorised according to Plutchik's wheel of emotions, which identifies eight primary emotions: Anger, Anticipation, Disgust, Fear, Joy, Sadness, Surprise, and Trust. In addition, the study aims to overcome the data scarcity challenge by employing Generative Adversarial Networks (GANs) to generate synthetic speech samples, augmenting the existing dataset (Barnard, Davel, Heerden, Wet, & Badenhorst., 2014).

This research aims to significantly contribute to the field of SER for African languages by developing a publicly accessible, high-quality Afrikaans speech corpus. The corpus will facilitate further research and model development, enabling the creation of more inclusive and

accurate SER systems that reflect the linguistic and cultural diversity of South Africa. By addressing the current data limitations and expanding the scope of SER research, the study seeks to unlock new opportunities for emotion-driven applications in various industries, from customer service to mental health monitoring.

3.2 Related Works

3.2.1 South African Languages

South Africa recognises eleven official languages, reflecting its rich linguistic diversity. These are detailed in Table 3.1 based on the 2001 Census (Probyn, 2006), with their geographical distribution illustrated in Figure 3.1.

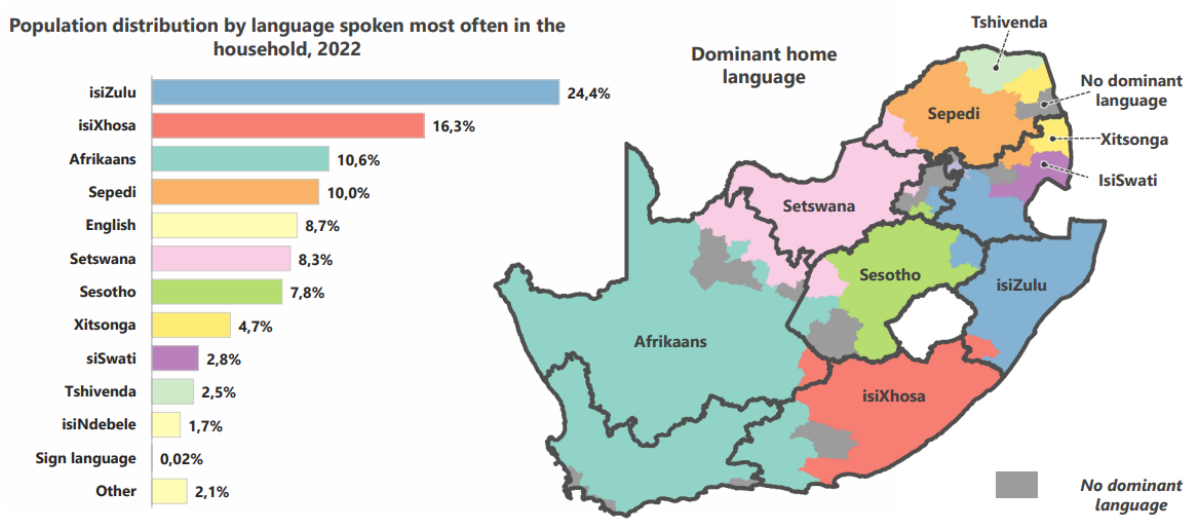

Figure 3.1: South African languages distribution *(Africa, 2023)*.

Table 3.1: South African Languages *(Africa, 2023)*.

Language	% of population	Number of speakers
isiZulu	24.4%	15.13 million
isiXhosa	16.3%	10.11 million
Afrikaans	10.6%	6.57 million
Sepedi	10.0%	6.20 million
English	8.7%	5.40 million
Setswana	8.3%	5.15 million
Sesotho	7.8%	4.84 million
Xitsonga	4.7%	2.92 million
siSwati	2.8%	1.74 million
Tshivenda	2.5%	1.55 million
isiNdebele	1.7%	1.05 million

3.2.2 Afrikaans

Afrikaans is classified as a West Germanic language with historical roots in the Dutch dialects spoken in the Netherlands during the seventeenth century. Its linguistic development began with the arrival of Dutch settlers at the Cape of Good Hope in 1652, led by Jan van Riebeeck. Over time, the language evolved through sustained interaction between early Dutch settlers and indigenous Khoi and San communities, resulting in significant linguistic adaptation and creolisation. Additional lexical and grammatical influences were introduced through immigration from France and Germany, as well as through contact with enslaved populations brought to the region from various parts of Africa and Asia. Contemporary Afrikaans is predominantly spoken in South Africa and Namibia, with smaller speech communities also present in Botswana and Zimbabwe (Hamans, 2021).

Afrikaans has its origins in the Dutch settlers who came to the Cape of Good Hope during the mid-17th century. Afrikaans gradually developed from Dutch, incorporating vocabulary and syntax from other languages in contact with it, like the Khoi, San, Portuguese, Malay, and Bantu languages (Giliomee, 2020)

3.2.3 Afrikaans Speech Corpus

One of the primary goals and objectives of the thesis is to create an Afrikaans speech corpus

3.2.3.1 Collection of Samples

Numerous speech emotion recognition (SER) databases currently exist, predominantly in English. Developing a bespoke South African Afrikaans corpus is a significant advancement toward facilitating additional research in African languages. Plans are in place to continually enhance the Afrikaans corpus by incorporating more audio clips.
It is crucial to note that the Afrikaans corpus is modelled on Plutchik's eight primary emotions, while the EmoDB corpus focuses on seven emotions. Future endeavours will explore potential improvements, including more sophisticated models and enhanced audio samples. The compiled Afrikaans corpus will be publicly accessible, enabling researchers to utilise and build upon the resource.

Multiple digital media channels operating under Creative Commons Attribution licenses allow the lawful reuse of audio content for academic and research purposes. These platforms provide accessible speech material that can be utilised for corpus development and emotion-based speech analysis. The concept is illustrated in Figure 3.2.

The selected audio and video segments are archived locally to enable efficient retrieval and processing during corpus construction. The collection procedure focuses on identifying speech excerpts that clearly convey a target emotional state, for example, scenes in which a speaker exhibits emotions such as suprise, sadness, or joy.

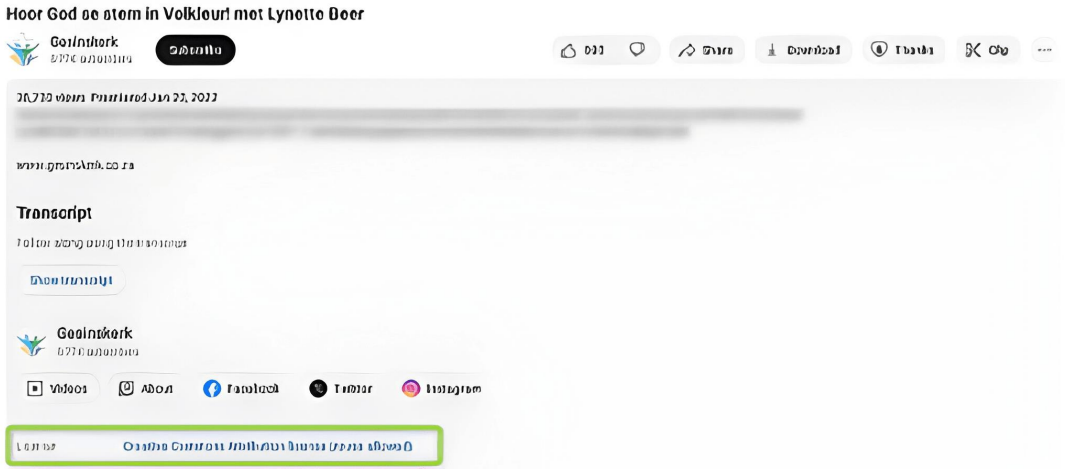

Figure 3.2: Example samples.

3.2.3.2 Settings and Software

Audio extraction and editing were performed using the Audacity digital audio platform (https://www.audacityteam.org). Once an appropriate speech segment containing a clearly identifiable emotional expression had been identified, the corresponding audio excerpt was recorded and processed. To maintain consistency across the corpus, all speech samples were standardised to durations between 1 and 5 seconds, stored in mono format, and sampled at 44.1 kHz. The concept is illustrated in Figure 3.3

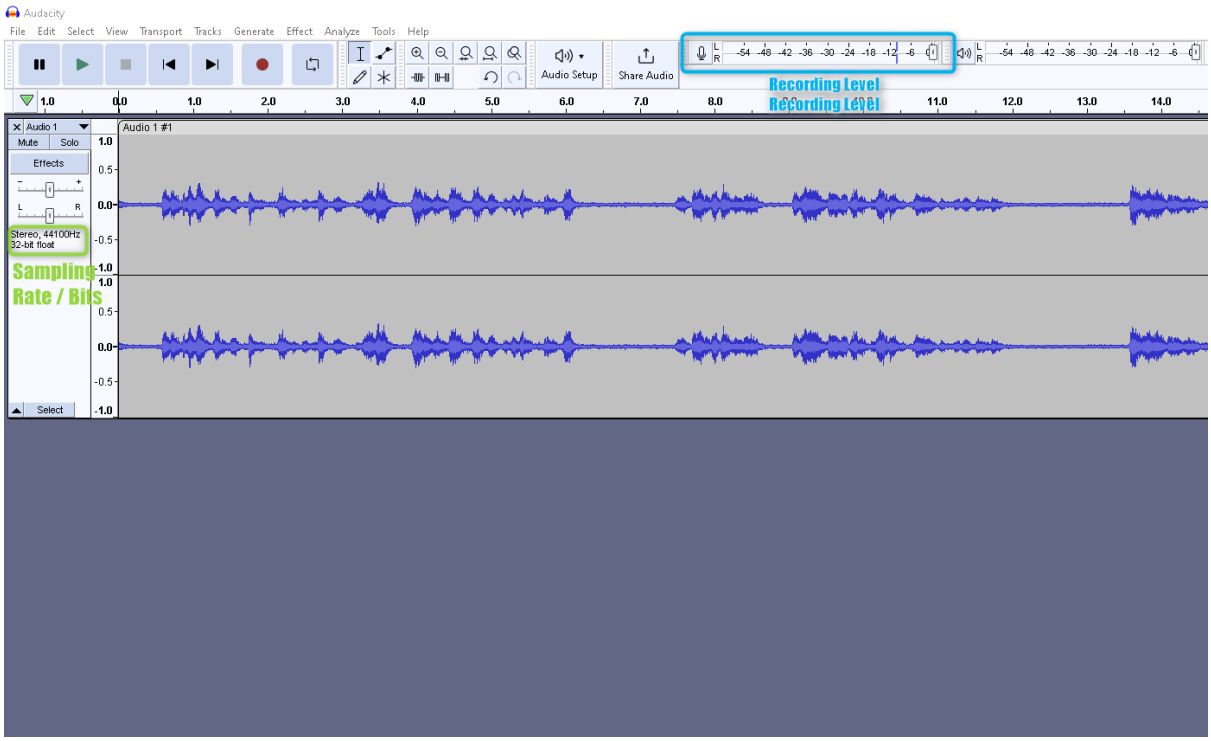

Figure 3.3: Audacity.

All speech samples were preserved in a lossless audio format to maintain acoustic fidelity during preprocessing and feature extraction. A structured directory hierarchy and systematic

file-labelling approach were implemented to support efficient dataset organisation, retrieval, and emotion-category management. Illustrated in Figure 3.4 and Figure 3.5.

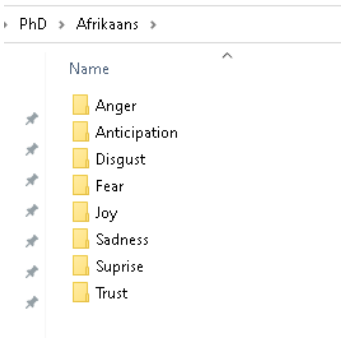

Figure 3.4: Folder structure.

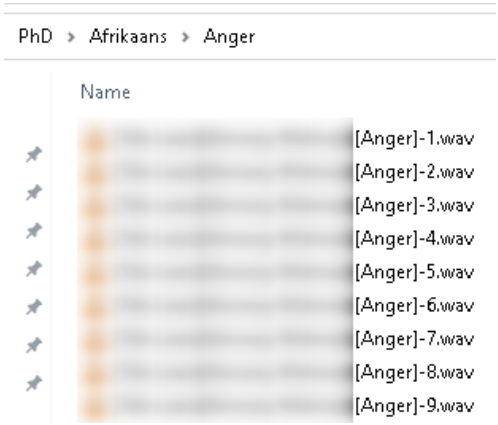

Figure 3.5: Naming convention of files.

3.2.3.3 Processing

Because the sourced recordings exhibited varying amplitude levels and acoustic intensity, audio normalisation and selective gain adjustment were applied where necessary prior to final storage in the corpus. The concept is illustrated in Figure 3.6.

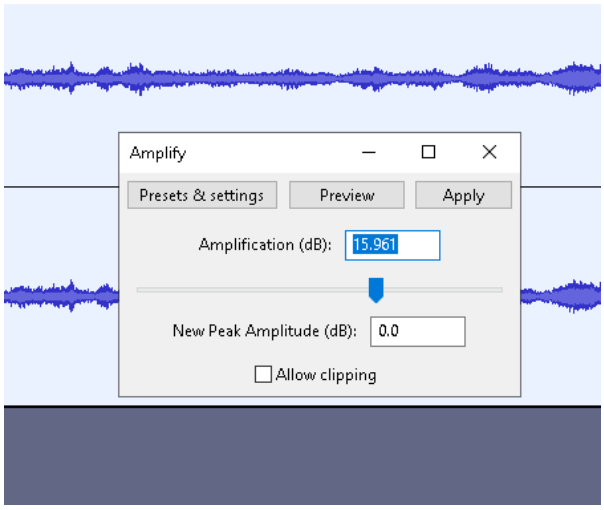

Figure 3.6: Audio post-processing.

3.2.3.4 Verification

Each collected audio sample underwent a structured verification process to ensure consistency in recording quality, signal amplitude, clip duration, and emotional classification. The annotation procedure involved assessment by the primary researcher together with two additional independent reviewers to improve the reliability and consistency of the assigned emotion labels.

In situations where the reviewers could not reach consensus on either the assigned affective label or the technical adequacy of a given clip, that clip was returned for additional inspection, re-editing, and possible relabelling. Recordings that suffered from intrusive ambient noise, weak or unconvincing affective delivery, or any comparable shortcoming were discarded outright so as to safeguard the overall reliability of the collection.

The sampling plan targeted roughly one hundred verified utterances per affective class, with eight such classes defined in advance. Following this scheme to completion produced an Afrikaans affective speech resource totalling close to 800 individual recordings.

3.2.4 Synthetic Speech

Generative Adversarial Networks (GANs), an architectural paradigm of profound intrigue, occupy a leading position in technological innovation. These networks find their applications in various domains, highlighting their multifaceted utility. Some salient examples encompass text-to-image translation and image manipulation/editing. Furthermore, they are extensively utilised to enhance the resolution of images and facilitate the creation of three-dimensional objects.

Delving into audio, distinct frameworks, including Google's WaveNet and Parallel WaveNet and their successors, Tacotron 1 and 2, have emerged as the preferred choices. These frameworks engender synthetic-based audio, underlining their unique capacity in the auditory field.

Moreover, synthetic data generation can bridge the gap in deficient training data scenarios. The approach augments the available data volume, fostering further research and training. GANs and audio frameworks are progressively shaping the technology landscape, presenting promising prospects for future developments.

3.2.4.1 Text-To-Speech Synthesis

The machine-driven production of voice signals that resemble natural human speech took root as a field of inquiry around the middle of the previous century, and it has remained noteworthy ever since for the particular range of capabilities it enables (Story, 2019). The state of the art has, however, moved on from earlier paradigms, with current efforts converging on far more capable techniques — chiefly those grounded in deep neural architectures alongside conditional adversarial generative pipelines. The vocal data underpinning the present study is sourced from the NCHLT Afrikaans repository, an archive that aggregates approximately fifty-six hours of spoken audio drawn from a pool of eight individual speakers. Distribution of this

archive falls under a Creative Commons arrangement, which permits unrestricted reuse by interested parties (Heerden, Barnard, Badenhorst, Davel, & Waal, 2014). Adopting a resource of this scale and breadth is expected to substantially strengthen the investigation by furnishing it with a richly populated body of audio material to draw upon.

Legacy Synthesis

The lineage of machine-generated speech reaches back to the final stretch of the 1950s. The earliest realisations of such systems relied heavily on parametric source-filter modelling, which became known as Linear Predictive Coding (LPC). As the underlying signal-processing toolkit matured, LPC progressively yielded ground to an alternative spectral representation based on line spectral pairs (LSPs). The early 1980s brought a particularly noteworthy development with the appearance of a purpose-built integrated circuit implementing the LSP scheme, which constituted a meaningful advance along the longer arc of synthetic-voice research.

It is also worth observing that the timbral character of the first generations of artificial voices leaned overwhelmingly toward the masculine register, a pattern that persisted until AT&T succeeded in producing a credible feminine synthetic voice. With that milestone in place, the path was cleared for a substantially wider and more inclusive palette of synthetic vocal identities to take shape over subsequent decades.

Across the years, the discipline has accumulated a varied catalogue of generative strategies. Concatenation-based stitching of recorded fragments, optimal-unit selection, diphone assembly, narrowly scoped domain pipelines, formant-resonance modelling, vocal-tract articulatory simulation, hidden-Markov-style statistical parametric methods, and reduced sinusoidal approximations all sit within this catalogue. Each technique introduces its own distinctive flavour to the resulting acoustic output, and together they have steadily extended the practical reach of artificial speech generation systems (Kumar, Koul, & Singh, 2022).

3.2.4.1.1 Neural Network Speech Synthesis

Modern practice in this area centres almost entirely on neural-network-driven pipelines, which are typically arranged to produce an intermediate spectral or vocoder representation that is subsequently rendered into an audible waveform corresponding to the input text. Among the architectural families that have gained particular traction within these pipelines, the Generative Adversarial Network occupies an especially prominent position. Its sustained popularity reflects both its computational tractability and its demonstrated capacity to handle demanding generative problems, qualities that have together driven much of the recent forward momentum in the synthetic-voice space.

3.2.4.2 GAN Networks

A schematic depiction of the standard GAN configuration is provided in Figure 3.7. Structurally, a GAN is built around a pair of cooperating yet adversarial sub-modules: a Generator component, denoted G, and a Discriminator component, denoted D. The role assigned to the generator is to learn a transformation that takes samples drawn from an easily sampled latent distribution $p_z(z)$— typically a Gaussian or uniform noise source represented by the variable z— and pushes them onto the considerably more intricate data manifold

described by $P_g(x)$, where x refers to a target observation drawn from the real-world distribution $P_d(x)$. Through iterative optimisation, the generator is gradually steered toward producing samples whose induced distribution $P_g(x)$ becomes statistically indistinguishable from that of genuine data $P_d(x)$. Meanwhile, the discriminator is concurrently trained to draw a reliable boundary between authentic data instances and the synthetic counterparts emitted by the generator, with the entire two-player optimisation cast in the form of a minimax objective (Tian, Wan, & Liu, 2018).

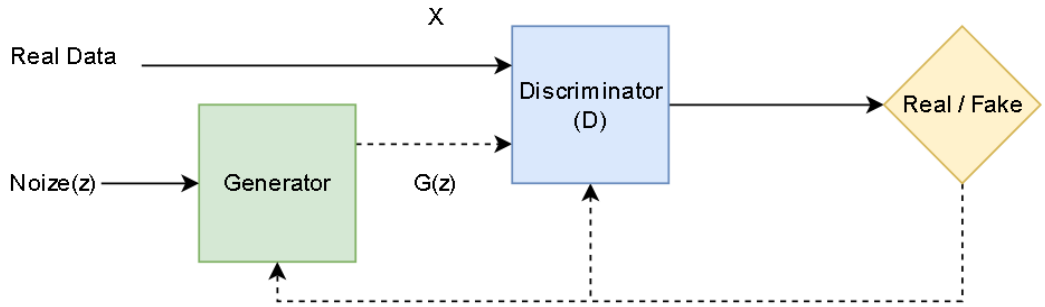

Figure 3.7: GAN Architecture.

Tacotron 2 is used to generate the samples. Figure 3.8 illustrates the evolution from Google WaveNet to Tacotron 2.

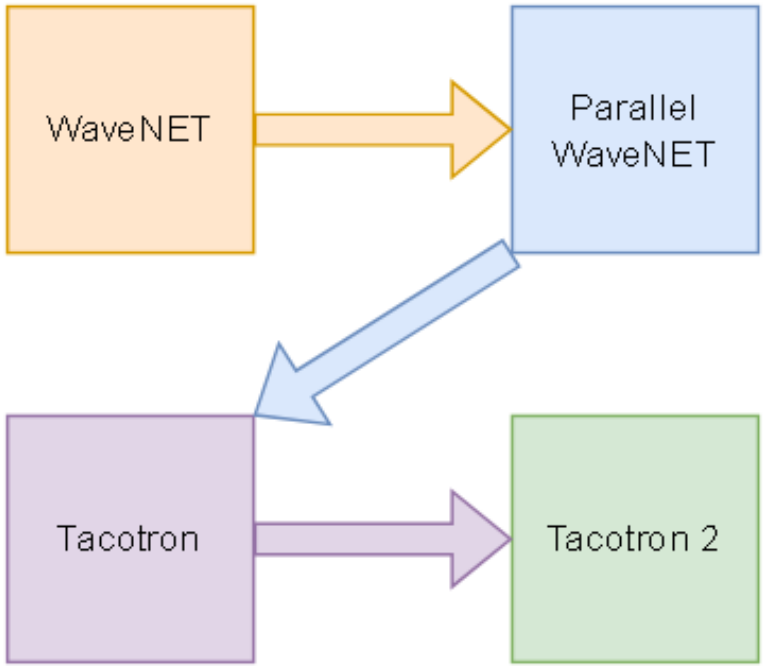

Figure 3.8: Tacotron 2 evolution.

At its core, the Tacotron 2 design rests on a recurrent backbone built from long short-term memory (LSTM) units, organised within an encoder–attention–decoder pipeline. The system's

role is to map written input into Mel-scale spectrogram representations. Within the encoder stage, individual characters or phoneme symbols are first projected into a dense vector representation, which is then passed through a sequence of convolutional layers to further extract features.

The output of that convolutional block is then forwarded to a bidirectional LSTM module. Generation on the decoder side, by contrast, is handled by an autoregressive LSTM that emits one Mel-spectrogram frame per step. Connecting the two halves of the architecture, an attention mechanism functions as a learned alignment, signalling to the decoder which segments of the encoded textual representation should be drawn upon at each generation step (Shen & Pang, Tacotron 2: Generating Human-like Speech from Text, 2017; Goodfellow, et al., 2014).

In essence, Tacotron 2 can be viewed as an attention-equipped recurrent neural network (RNN) that takes a text sequence as input and outputs an associated spectrogram. Translating that intermediate spectrogram back into an audible waveform is then handled by a separate vocoder stage, with WaveNet and the well-established Griffin–Lim reconstruction procedure being two representative options for this purpose. The overall arrangement is depicted in Figure 3.9.

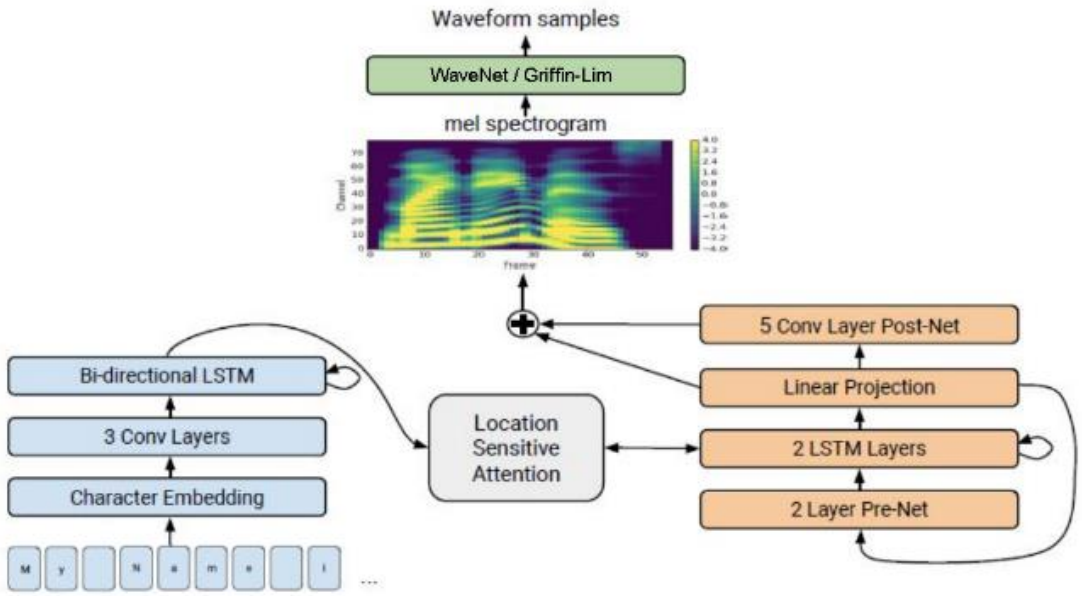

Figure 3.9: Tacotron 2 System architecture using WaveNet vocoder.

A range of optimiser configurations was trialled across different model variants, with Adam ultimately standing out as the strongest performer in terms of both achieved quality and the speed at which the network reached convergence. As a routine step prior to executing the backward pass on each mini-batch, the accumulated gradient buffers were cleared back to zero so as to avoid unintended carry-over from previous updates. On the waveform-reconstruction side, two alternatives were placed under comparison — namely the neural WaveNet vocoder and the classical Griffin–Lim reconstruction procedure.

3.2.4.3 Database and Settings

Training of the Tacotron 2 network was conducted on the NCHLT Afrikaans collection. This

resource takes the form of a broadband audio archive accompanied by full orthographic transcriptions and totals approximately fifty-six hours of recorded speech. A held-out evaluation portion drawn from eight separate voices is also bundled with the dataset.

On the compute side, model fitting was performed across Google Colab and Kaggle, both of which provide access to capable accelerator hardware in the form of NVIDIA P100, V100, and T4 GPUs. TensorFlow and PyTorch were the two deep-learning libraries used throughout this part of the work.

3.2.4.4 Training and Synthesising

This concept is illustrated in Figure 3.10. Each training epoch takes approximately one hour to complete. The duration is consistent with the computational demands of autoregressive sequence-to-sequence models trained on tens of thousands of audio samples. Contributing factors include the high frame-rate mel-spectrogram representation (~86 frames per second), the sequential nature of the decoder, the ~28 million trainable parameters in Tacotron 2, the attention mechanism's quadratic cost in sequence length, and the constraints of shared cloud GPU environments (Google Colab / Kaggle) where batch size and I/O throughput are limited.

Figure 3.10: Epoch training.

Source audio in WAV format is first converted into NumPy .npy containers that hold the corresponding Mel-spectrogram representations, and these arrays then serve as the input examples used by the Tacotron 2 network during fitting. The model is regarded as sufficiently trained — and thus ready to be deployed for inference — once the validation loss settles in the 0.15 region. A graphical view of the overall training process is provided in Figure 3.11.

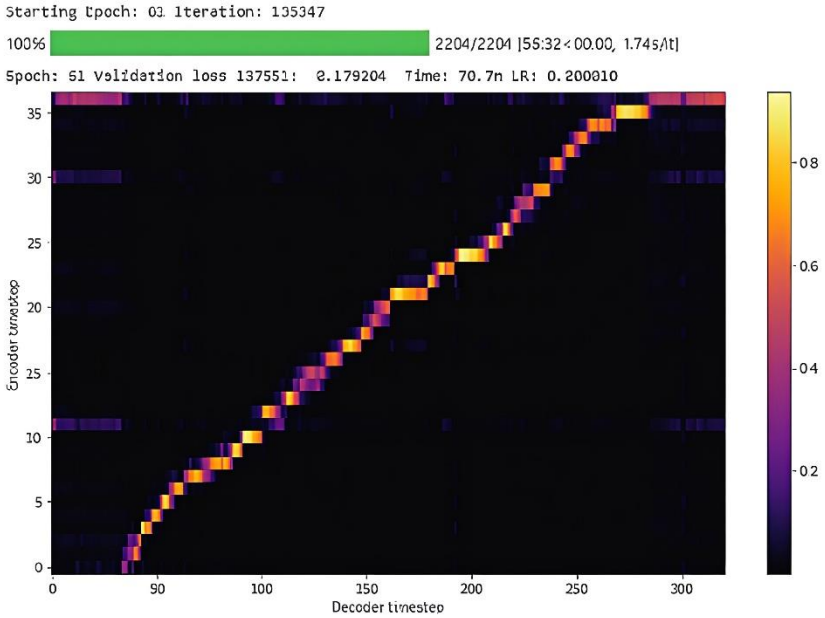

Figure 3.11: Tacotron 2 Network Training.

The implementation on which this work is based was originally derived from the NVIDIA

Tacotron 2 reference codebase. That said, a series of significant adjustments and customisations were applied to it in order to align the pipeline with the particular characteristics of the Afrikaans spoken-language dataset. Once fitting was complete, the resulting model was put to work producing synthetic speech utterances, as depicted in Figure 3.12.

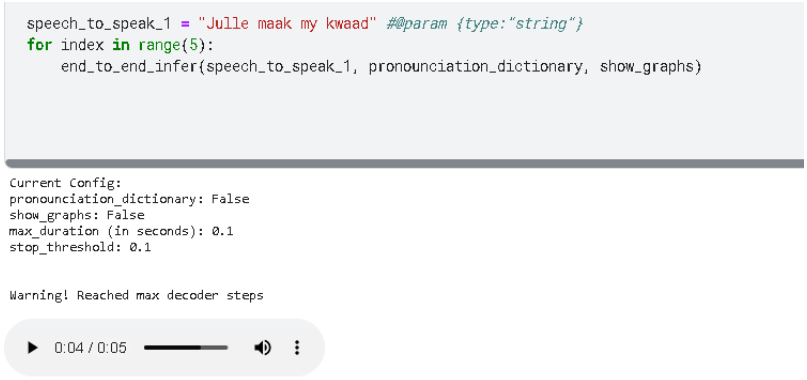

Figure 3.12: Speech generation carried out using a fully trained Tacotron 2 instance.

The synthesised audio fragments were obtained from a diverse set of input sentences and subsequently grouped into the eight target affective labels under consideration in this study, namely Anger, Anticipation, Disgust, Fear, Joy, Sadness, Surprise, and Trust. With respect to perceptual fidelity of the rendered output, the neural WaveNet vocoder consistently delivered superior results when compared against the Griffin–Lim reconstruction procedure, yielding clips that were noticeably cleaner and more natural-sounding overall.

3.3 Results

3.3.1 Afrikaans speech corpus

Evaluation of the curated audio set was performed within the Python ecosystem using the Librosa toolkit, paired with an LSTM recurrent architecture. Prior to model ingestion, each utterance was transformed into its corresponding Mel-Frequency Cepstral Coefficient (MFCC) representation, which then served as the input feature stream presented to the network. The performance figures obtained on this corpus were subsequently logged and benchmarked against equivalent measurements drawn from comparable affective speech resources reported in the literature. The neural-network back end relied on TensorFlow as its underlying computational framework.

Predictive performance was quantified using a combination of the accuracy_score routine from scikit-learn's metrics module and Keras' built-in model evaluation interface. Within a multi-label evaluation regime, the accuracy_score routine returns what is commonly described as subset (or exact-match) accuracy. The numerical assessment relies on the standard tally of correctly identified positives and negatives, alongside misclassifications and recall values. From these primary counts, derived indicators including the F1 metric, mean absolute and mean squared error, and the receiver-operating-characteristic AUC can be assembled. A side-by-side inspection illustrated in Figure 3.13 highlights how the on-disk layout and per-class file totals differ between the EmoDB collection and the Afrikaans build, with the former defining seven

affective classes and the latter expanding the schema to encompass eight.

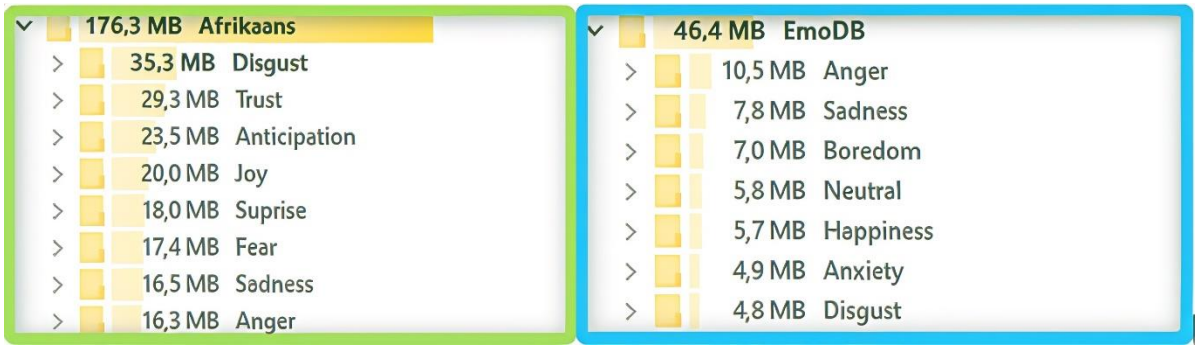

Figure 3.13: On-disk folder layout.

Subtle distinctions in affective labelling, sample counts per class, and resulting accuracy figures are surfaced in Table 3.2.

Table 3.2: Side-by-side Corpus comparison (The full LSTM calculation results are provided in Appendix A.2.1 and A.2.2).

Database	**EmoDB**	**Afrikaans**
Emotion	Anger	Anger
	Anxiety	Anticipation
	Boredom	Fear
	Disgust	Disgust
	Happiness	Joy
	Neutral	Suprise
	Sadness	Sadness
		Trust
LSTM Mean Accuracy (10 runs)	**0.6848**	**0.5725**

The LSTM accuracy reported in Table 3.2 is computed as the proportion of correctly classified samples over the total number of samples in the dataset.

In addition to accuracy, other evaluation metrics, such as precision, recall, macro F1-score, weighted F1-score, and balanced accuracy, are utilised in the later experimental chapters (Chapters 4 and 5), where class imbalance is more pronounced. These metrics provide a more comprehensive evaluation of model performance across all emotion classes.

Regression-based metrics such as mean absolute error (MAE) and mean squared error (MSE) are not applicable in the study, as the task is a multi-class classification problem rather than a regression problem.

Similarly, the area under the curve (AUC) metric is not reported, as AUC is primarily suited for binary classification tasks. In multi-class emotion recognition, macro F1-score and balanced

accuracy are more appropriate and widely adopted evaluation measures.

For completeness, full experimental results, including per-run values, mean, and standard deviation across multiple runs, are provided in the Appendices.

3.3.2 Synthetic Speech

Set against alternative hosted offerings such as Google Cloud's Text-to-Speech service and Narakeet, the audio output from the model developed in this study holds its own in perceptual fidelity and produces a credibly natural-sounding voice. It is, however, worth noting that the marketplace for commercial Afrikaans speech-synthesis products remains noticeably underserved compared with offerings in many other widely spoken languages.

For quantitative quality assessment, the customary instrument has long been the Mean Opinion Score (MOS) procedure formalised under ITU-T Recommendation P.85 in 1994. This evaluation approach gathers listener judgements along seven separate five-point dimensions, covering perceived overall audio quality, the cognitive effort required to follow the material, intelligibility difficulties, clarity of articulation, accuracy of pronunciation, the perceived pace of delivery, and overall pleasantness of the listening experience. The interpretation of each individual five-point dimension follows the standard verbal anchors: a score of 1 corresponds to "Bad", 2 to "Poor", 3 to "Fair", 4 to "Good", and 5 to "Excellent".

A side-by-side comparison of how the Afrikaans Tacotron 2 implementation fares relative to Google's Cloud TTS offering is presented in Table 3.3.

Table 3.3: MOS Scores.

Model	Afrikaans (Tacotron Griffin-Lim)	Afrikaans (Tacotrons WaveNet)	Google Cloud TTS
Global Impression	2	4	5
Listening Effort	3	4	5
Comprehension Problems	2	5	5
Speech Sound Articulation	3	5	5
Pronunciation	2	5	5
Speaking Rate	2	4	5
Voice Pleasantness	1	4	5

3.4 Discussion

3.4.1 Afrikaans speech corpus

A reasonable number of speech-emotion recognition (SER) datasets are already in circulation

within the research community, though most are dominated by English-language content. By contrast, the construction of a purpose-built South African Afrikaans resource creates new opportunities for affective-computing research in the broader landscape of African tongues. Going forward, the Afrikaans collection assembled here is expected to undergo ongoing curation, with additional voice samples progressively added to it. Routine augmentation techniques together with Conditional Generative Adversarial Network (C-GAN) pipelines — exemplified by tools such as Google's Tacotron and WaveNet — are anticipated to play a meaningful role in expanding and strengthening the resource over time.
Future work will also examine avenues to upgrade both the modelling components and the underlying audio material to higher quality. The completed corpus is to be released openly, allowing any interested researcher to use it directly or build further extensions on top of it.

3.4.2 Synthetic Speech

The work undertaken here resulted in the production of artificially generated voice recordings specifically targeted at the Afrikaans language. A number of competing architectures were reviewed during this exercise, with Tacotron 2 ultimately standing out as both the most capable and the most practical choice for the task at hand. Several waveform-reconstruction back ends, encompassing Griffin–Lim alongside WaveNet, were also subjected to detailed examination. The resulting audio fragments allowed key facets of voice generation to be probed, including how lifelike the output sounds, the rhythmic and intonational contour, the apparent spontaneity of delivery, and the handling of phonetically ambiguous passages.
Quality benchmarking across these variants was conducted using the Mean Opinion Score (MOS) protocol as the chosen yardstick. Out of the vocoder options trialled, WaveNet emerged as the strongest performer. It should be noted, nonetheless, that the MOS rating achieved by the Afrikaans system developed in this work falls a little short of that observed for established commercial alternatives. The flip side of this observation is that the commercial market for Afrikaans-specific voice synthesis is itself rather thinly populated at present. To help address this gap, the model produced here will be released into the public domain so that other researchers can adopt it directly or build further refinements on top of it.

3.5 Conclusion

The development and analysis of the Afrikaans speech corpus in the chapter constitute the principal Design and Development output for Objective 1 within the Design Science Research Methodology. Key insights emerged regarding variability in recording conditions, the limited size of the dataset, which required synthetic augmentation using GANs and Tacotron 2, and the constraints of conventional acoustic features (MFCC, Chroma, and Tonnetz) in fully capturing the temporal and contextual aspects of emotional expression. These observations directly informed the iterative refinement of hybrid neural network architectures and preprocessing techniques examined in Chapter 4, addressing Objective 2.

The Afrikaans speech corpus developed in the chapter also provides the foundation for the experimental work that follows. Chapter 4 uses the corpus in a controlled single-language setting to evaluate hybrid neural architectures and preprocessing techniques, enabling clear

isolation of model and preprocessing effects. The architectural insight gained — particularly the performance of the DCLSTM model and its dendritic non-linear feature integration — informs the multilingual extension in Chapter 5, where the dendritic-layer contribution is further developed within the proposed DenCaps model and evaluated on a significantly larger dataset comprising approximately 23,500 samples from multiple languages. The Kalman_2 preprocessing finding from Chapter 4 is specific to the recording conditions of the Afrikaans corpus and is not directly carried into the multilingual setting; cross-corpus application of Kalman_2 is identified as future work in §6.2. The progression aligns with the iterative nature of the Design Science Research framework and establishes a clear transition from observation to single-language model development, then to multilingual evaluation.

CHAPTER 4: Hybrid Architectures and Speech Data Pre-Processing

Speech Emotion Recognition (SER) systems face inherent challenges due to the complexity of extracting emotional cues from speech (Nema & Abdul-Kareem, 2017). The performance of these models is often constrained by architectural limitations and the presence of noise or irrelevant acoustic features. To address these challenges, the chapter explores advanced audio pre-processing techniques, including noise reduction, spectral subtraction, pre-emphasis filtering, framing, windowing, and silence removal, to improve input data quality. Furthermore, the study evaluates the effectiveness of hybrid neural network architectures, specifically CNN-RNN models, which have shown promising results in deep learning applications (Shi, Wang, Wang, Liu, & Yan, 2019; Muaidi, Al-Ahmad, Khdoor, Alqrainy, & Alkoffash, 2014). By integrating these techniques, the research aims to optimise SER accuracy, enhance model robustness, and contribute to the development of more reliable emotion detection systems.

4.1 Introduction

This chapter builds directly on the Afrikaans corpus developed in Chapter 3. Using the single-language dataset as a controlled testbed, hybrid neural architectures and preprocessing techniques are evaluated in isolation. The resulting optimal configurations—particularly the DCLSTM model and Kalman_2 filter—provide the methodological foundation extended in Chapter 5. In recent years, advanced neural network architectures, such as CNNs, LSTMs, and hybrid models such as CLSTMs, have revolutionised various domains, including natural language processing, image recognition, and time-series forecasting. CNNs, typically applied to tasks with grid-like structures like images, extract spatial features through convolutional and pooling layers. LSTMs, on the other hand, excel in processing sequential data, making them adept at capturing temporal dependencies. However, both architectures face limitations when handling tasks that require a nuanced understanding of spatial and temporal dynamics.

Building on the Afrikaans corpus developed in Chapter 3, the chapter advances the Design and Development and Demonstration phases of the Design Science Research Methodology by systematically designing, implementing, and evaluating hybrid neural architectures (CNN, LSTM, CLSTM, and DCLSTM) and optimised audio preprocessing techniques. The architectures and preprocessing pipelines produced in this chapter constitute the engineered artefacts addressing Objective 2 of the research.

To address these challenges, the hybrid architecture, CLSTM, combines the strengths of CNNs in spatial feature extraction and LSTMs in managing temporal dependencies. The integration has proven successful in applications such as video classification and speech emotion recognition (SER), where understanding both spatial and temporal features is essential.

- **Strengths of CNN**

Within the proposed speech emotion recognition (SER) framework, CNNs perform spatial feature extraction on acoustic representations such as MFCCs and spectrograms. CNNs

effectively capture localised spectral patterns, including pitch variations, energy distributions, and formant structures associated with emotional expression. The use of convolutional filters enables translation invariance, allowing features to be detected regardless of their position in the time-frequency domain. Additionally, weight sharing reduces the number of trainable parameters, improving computational efficiency and generalisation. CNNs also provide robustness to noise by learning hierarchical feature representations, making them well-suited for processing real-world speech signals.

- **Strengths of LSTM**

Within the proposed framework, LSTMs are responsible for modelling temporal dependencies in speech signals. Emotional cues in speech evolve over time, and LSTMs effectively capture these sequential dynamics. The memory cell and gating mechanisms (input, forget, and output gates) allow LSTMs to retain relevant information while mitigating the vanishing gradient problem. The architecture enables the modelling of long-term dependencies across speech frames. LSTMs also handle variable-length input sequences, making them suitable for speech recordings of different durations. When combined with CNN-extracted features, LSTMs enhance performance by learning temporal relationships between spatial features.

Building on the foundation, recent innovations have introduced Dendritic Convolutional Long Short-Term Memory (DCLSTM) models, inspired by biological dendritic neurons. DCLSTMs enhance the traditional CLSTM by incorporating non-linear dendritic structures, allowing for more complex, localised computations. The architecture excels in spatiotemporal tasks, offering improved accuracy over conventional models, particularly in domains such as speech recognition and emotion detection.

In the field of SER, integrating these advanced models with robust preprocessing techniques like noise reduction, Kalman filtering, and spectral gating has significantly improved the clarity of audio inputs. These methods ensure that neural networks receive cleaner, more precise data, further boosting the performance of emotion recognition systems.

The continual evolution of these architectures, alongside innovative preprocessing techniques, promises to push the boundaries of human-computer interaction, enabling more accurate and emotionally intelligent systems.

4.2 Related Works

4.2.1 CNN & LSTM

Convolutional Neural Networks (CNNs) constitute a category within deep learning frameworks, primarily utilised in computer vision, though their application extends to audio and natural language processing tasks. The architecture of CNNs is intrinsically designed to interpret data exhibiting a grid-like structure, exemplified by grid-like representations such as images or time-frequency representations of audio signals (e.g., spectrograms). The core components of CNNs include convolutional layers, pooling layers, and fully connected layers. Convolutional layers utilise a collection of trainable filters that execute convolutions over the

input volume's width and height, calculating the dot product between the filter's values and the input, thereby generating a two-dimensional activation map. Pooling layers diminish the data's dimensionality, thus mitigating the risk of overfitting (Alzubaidi, et al., 2021). Subsequently, fully connected layers are employed to classify the attributes delineated by the convolutional and pooling layers. Renowned CNN architectures, such as LeNet-5, AlexNet, VGGNet, GoogLeNet, and ResNet, have substantially contributed to image recognition and have been successfully adapted for audio-based tasks. These architectures have progressively evolved to become more profound, to extract increasingly complex features and enhance accuracy. Innovations such as skip connections, introduced in ResNet, address issues endemic to deeper networks, including the vanishing-gradient problem. In the study, audio signals are transformed into spectrogram or MFCC representations, enabling CNNs to extract spatial features relevant to speech emotion recognition. The CNN employed in the research is depicted in Figure 4.1.

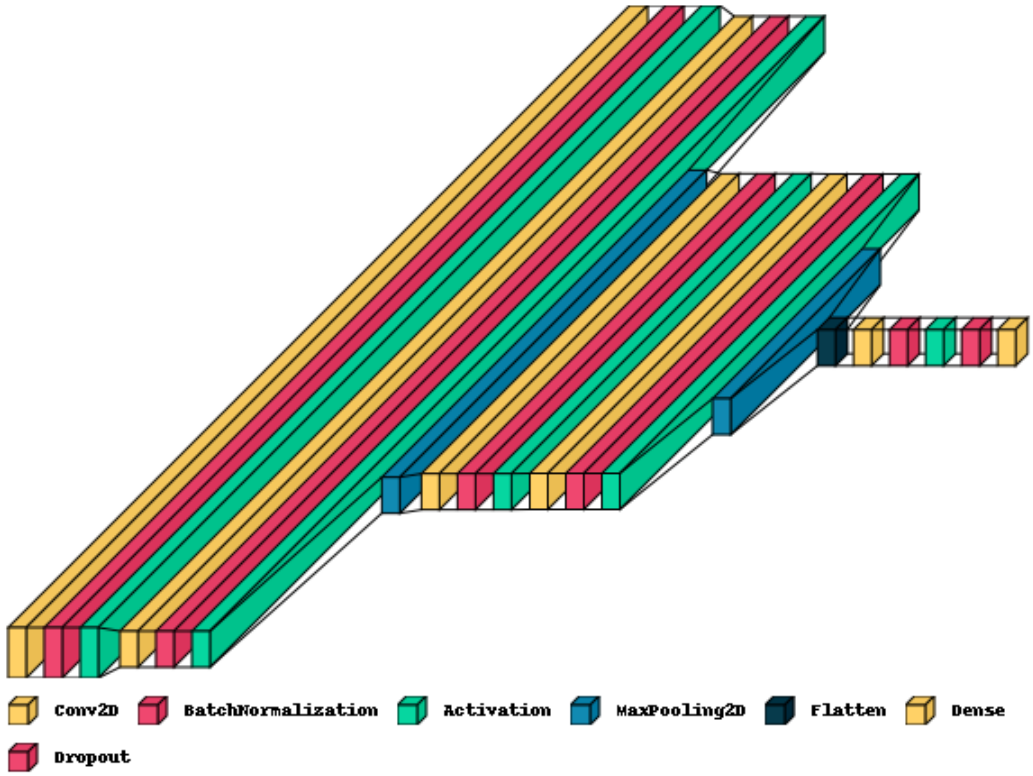

Figure 4.1: CNN (*generated by VisualKeras*).

Long Short-Term Memory (LSTM) networks signify a pivotal advancement in deep learning, particularly in addressing temporal dynamics and dependencies inherent in sequential data. As a specialised variant of Recurrent Neural Networks (RNNs), LSTMs are meticulously engineered to circumvent the vanishing gradient dilemma—an issue prevalent in conventional RNNs that obstructs the acquisition of long-term dependencies—through their distinct cellular architecture, which comprises forget, input, and output gates (Gers, 2001). These mechanisms collectively afford LSTMs the capacity to modulate the information flow, discerning between the data to be preserved and discarded over sequential time steps. As a result, LSTMs have demonstrated remarkable proficiency across various domains, including natural language processing, time-series forecasting, and speech recognition, where the comprehension of

temporal correlations is paramount. Despite their impressive attributes, LSTMs are not without challenges, encompassing computational demands and optimisation complexities, thereby catalysing continuous scholarly efforts to refine and enhance the utility of the architecture (Le, Ho, Lee, & Jung, 2019). An LSTM network is illustrated in Figure 4.2.

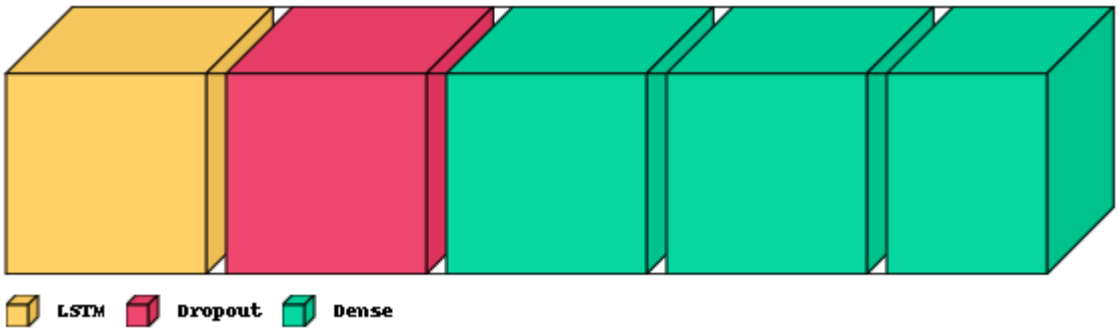

Figure 4.2: LSTM (*generated by VisualKeras*).

4.2.2 CLSTM

Convolutional LSTM (CLSTM) constitutes a composite neural architecture that fuses the complementary strengths of Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) units. CNNs are widely recognised for their effectiveness in capturing localised, translation-invariant patterns, particularly when applied to acoustic inputs represented as spectrograms or MFCCs. LSTMs, by comparison, are adept at modelling dependencies that span extended temporal ranges, which makes them a natural choice for handling ordered sequences and time-series signals. In a standard CLSTM pipeline, the convolutional stage acts first, sweeping over the raw input to distil a broad set of localised descriptors. The resulting high-level feature maps are then reinterpreted as a temporally ordered stream and passed forward into the LSTM stage as its sequential input. The layer meticulously evaluates the sequences to identify temporal relationships among the features (Zhou, Sun, Liu, & Lau, 2015). CLSTM-based models have delivered strong performance across a range of problems where joint reasoning over space and time is essential, including affective speech analysis and forecasting tasks that rely on sequential measurements. The design draws on the localised pattern-detection abilities of convolutional layers in tandem with the temporal-reasoning machinery of recurrent units, offering a coherent end-to-end mechanism for jointly learning spatial and temporal regularities. The introduction of LSTM units into the broader deep-learning toolkit was itself a landmark development, largely because it provided a workable answer to long-standing difficulties in modelling depth and long-range temporal structure simultaneously. The CLSTM utilised in the research is depicted in Figure 4.3 (Kim & Cho, 2018).

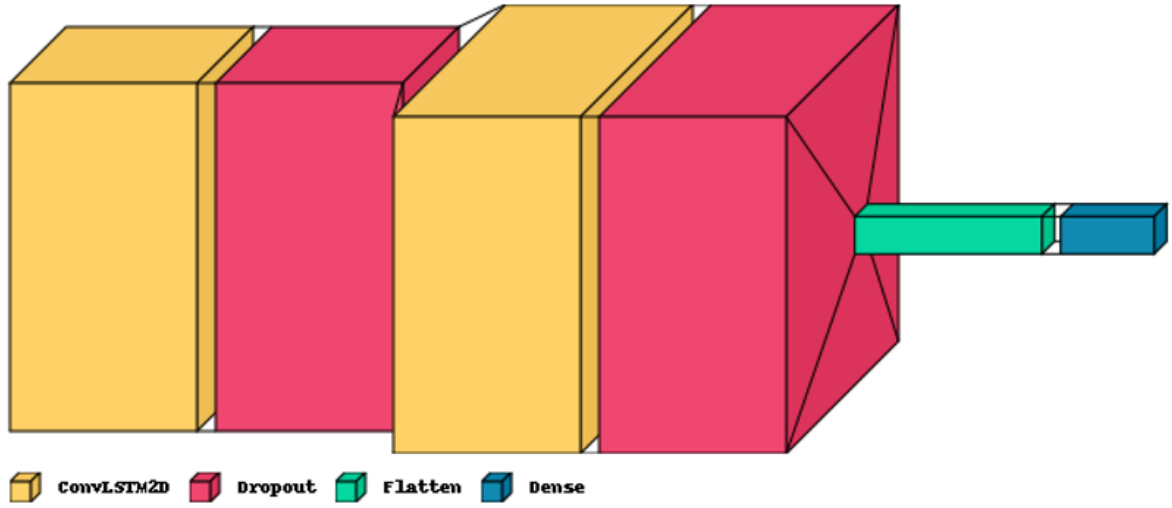

Figure 4.3: CLSTM (*generated by VisualKeras*).

4.2.3 Dendritic Convolutional Long Short-Term Memory.

Recent advancements within the realm of deep learning have culminated in the development of innovative architectural designs aiming to enhance the capabilities of neural networks, particularly concerning the analysis of time-series data and spatial processing tasks. Among these developments, the Dendritic Convolutional Long Short-Term Memory (DCLSTM) emerges as a noteworthy innovation (Ji, Tang, Zhao, Tang, & Todo, 2022). The architecture synergises the dendritic integration mechanisms—mirroring the functionality observed in biological neurons—with the convolutional and recurrent attributes inherent to CLSTM (Gul, 2023). Unlike conventional LSTMs, which linearly process information and often struggle with complex spatial-temporal patterns, the dendritic element of DCLSTM facilitates a non-linear integration of spatial data, thus offering a refined approach to local computation (Yamamoto, Song, & Kim, 2020). The inclusion of convolutional layers grants the model the proficiency to discern spatial patterns, whereas the LSTM component equips it with the capability to proficiently manage temporal sequences. Preliminary empirical evaluations suggest that DCLSTM exhibits commendable efficacy in tasks such as speech emotion recognition and spatiotemporal forecasting of audio signals, highlighting its potential to significantly alter the conventional methodologies employed in neural processing within intricate domains. In the context of the study, acoustic features derived from speech signals (e.g., MFCCs and spectrograms) are used as input, enabling the DCLSTM architecture to model both spatial and temporal characteristics relevant to emotion recognition.

4.2.4 Audio Preprocessing and Noise Reduction

Digital Signal Processing (DSP) takes signals from various sources and mathematically manipulates the data. Signal types include voice, video, temperature, pressure, position and, of course, audio. Audio is the primary input to the neural network in the research. Audio signals are digitised and then mathematically added, subtracted, multiplied or divided (Pohlmann, 2011). The section looks at various techniques for better and cleaner input data.

4.2.4.1 Noise Reduction

Noise in a signal is random, unwanted variations or fluctuations that interfere with the signal data and quality. Below is an example of noise added to a signal (Vaseghi, 1996).

In the vanguard of audio noise mitigation techniques, several methodologies have emerged, notable among which are the Kalman Filter, the Least Mean Square (LMS) Adaptive Filter, the Signal-Dependent Rank Order Mean algorithm (SDROM), and the Spectral Gating Algorithm. Figure 4.4 demonstrates the application of noise reduction utilising the "noisereduce" Python package, exemplifying the efficacy of contemporary noise removal techniques.

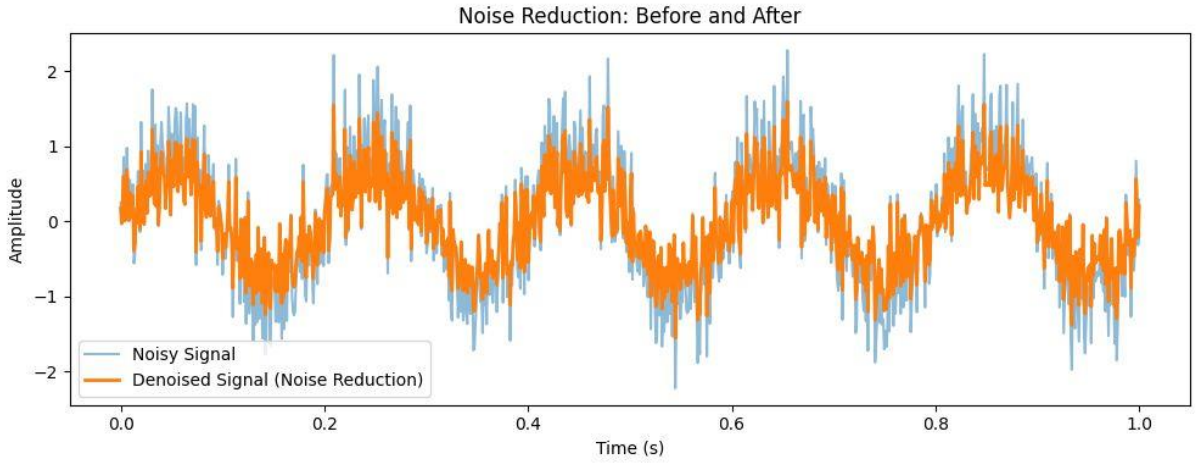

Figure 4.4: Noise Reduction (*Generated using Python*)

4.2.4.2 Kalman Filter

In 1960, R.E. Kalman published a seminal paper on his discrete-data linear filter (Welch & Bishop, 1995). While much of the ensuing research has focused on its applications in autonomous navigation, the Kalman filter also offers substantial utility in audio signal processing. The concept is illustrated in Figure 4.5.

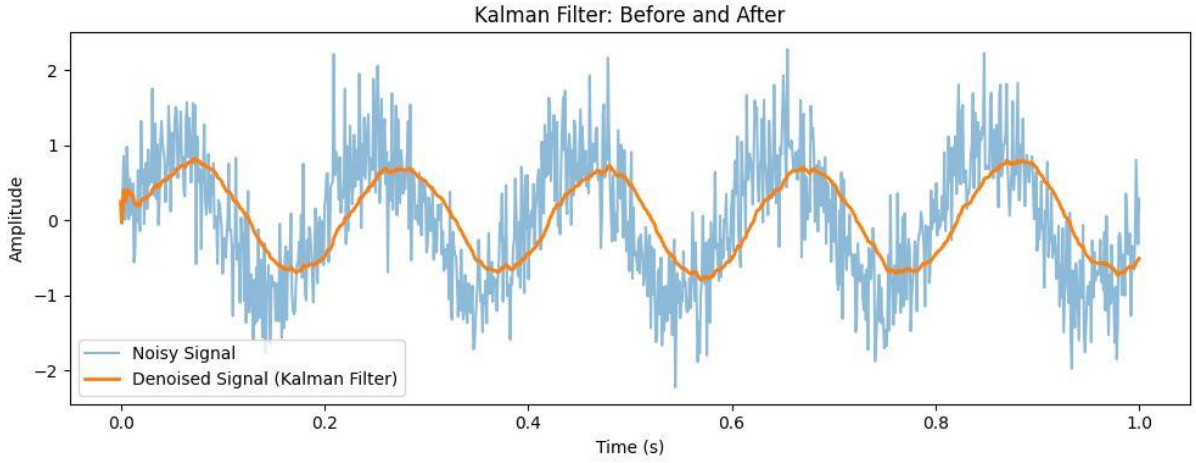

Figure 4.5: Kalman Filter (*Generated using Python*)

4.2.4.3 Spectral Gating Algorithm

Spectral gating is a popular method in audio signal processing, especially for reducing noise. It works by focusing on the 'spectral envelope,' which describes the primary frequency characteristics of a signal. The algorithm acts like a gatekeeper in the frequency domain, allowing only the frequencies within the spectral envelope to pass through while blocking out the rest. The approach helps to lower the noise levels, particularly from broad-spectrum sources, without distorting the core aspects of the original sound. However, the success of spectral gating can vary depending on the type of noise and the audio signal itself. For example, if the noise shares a lot of frequencies with the signal, the algorithm might struggle to separate the two effectively. The concept is illustrated in Figure 4.6.

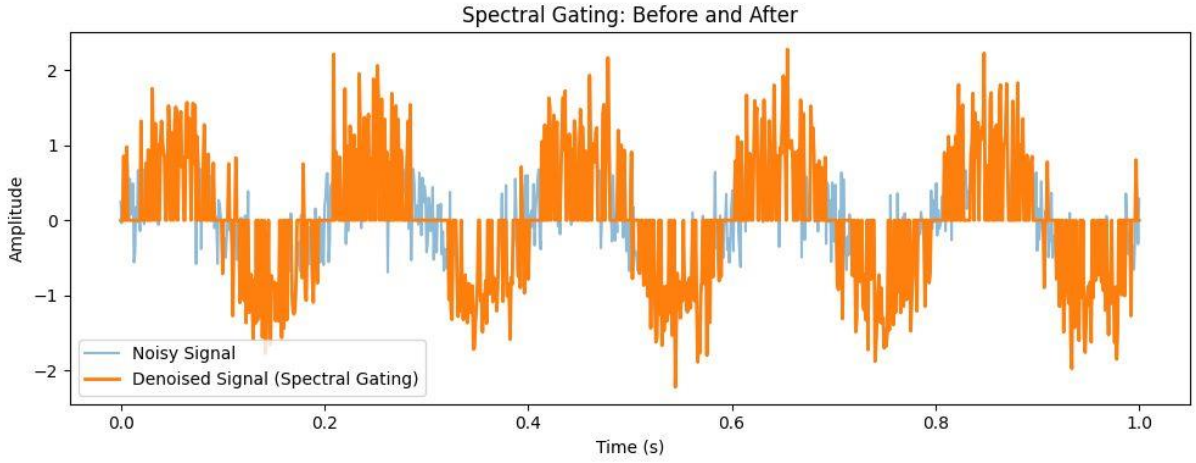

Figure 4.6: Spectral Gating (*Generated using Python*)

4.2.4.4 Spectral Subtraction

Spectral subtraction removes noise by subtracting the magnitude spectrum of a noise signal from the original signal. A variation of the method is called the Boll Spectral Subtraction **(Naik, Bhatikar, & Gaude, 2021)**. The method implements spectral averaging and residual noise reduction. The concept is illustrated in Figure 4.7.

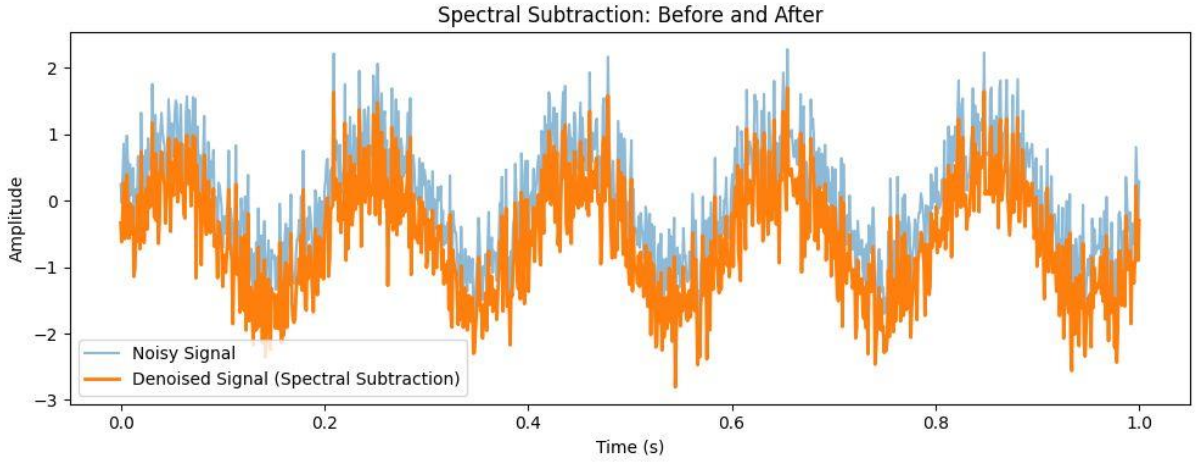

Figure 4.7: Spectral Subtraction (*Generated using Python*)

4.2.4.5 FIR & IIR Filters

Filtering can be performed on either an analogue signal or a discrete signal that has been digitised. Linear time-invariant systems, often referred to as filters, are the generic terms for these systems. Digital filters, another name for discrete-time LTI systems, come in two varieties: Finite Impulse Response (FIR) and Infinite-duration Impulse Response (IIR) (Ingle & Proakis, 2012). The principal distinction between FIR and IIR is the absence of a feedback mechanism in FIR, making it more straightforward. Various types of filtering, including High Pass, Low Pass, Bandpass, and Notch, can be implemented. The illustration below demonstrates a lowpass filter and a bandpass filter. Additionally, the filter order and frequency response can be observed. For the research, the input sounds will be processed via a bandpass filter to filter out unwanted frequencies (Rehman, Liu, Wu, Cao, & Jiang, 2022). The concept is illustrated in Figure 4.8.

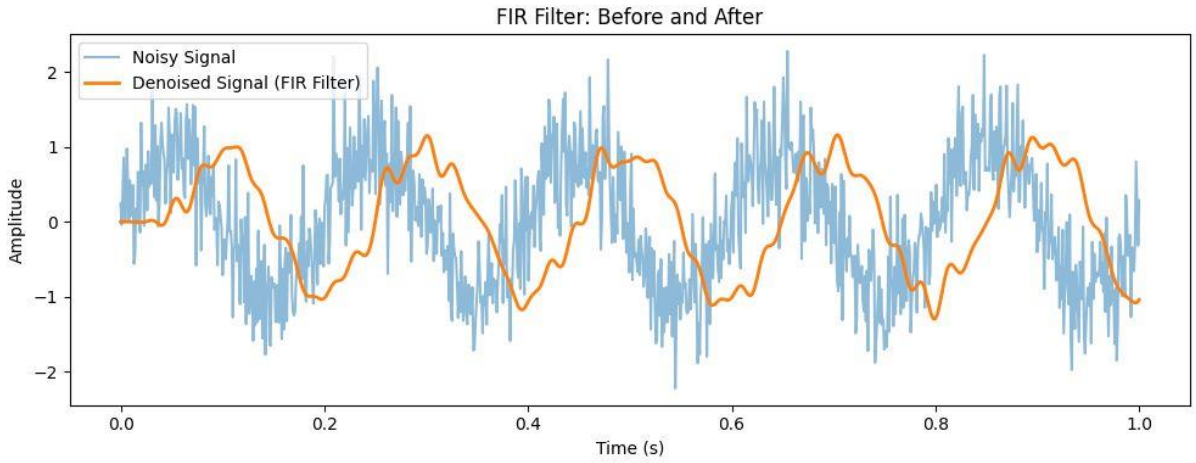

Figure 4.8: FIR Filter (*Generated using Python*)

4.2.4.6 Silence Removal

For the research, silent audio segments will be removed from voice clips and processed. To remove silence, one needs to take the following steps:

- Load data.
- Split the file into segments, keeping the data above a specific dB threshold.
- Recombine the split sections into the new file.

The concept is illustrated in Figure 4.9.

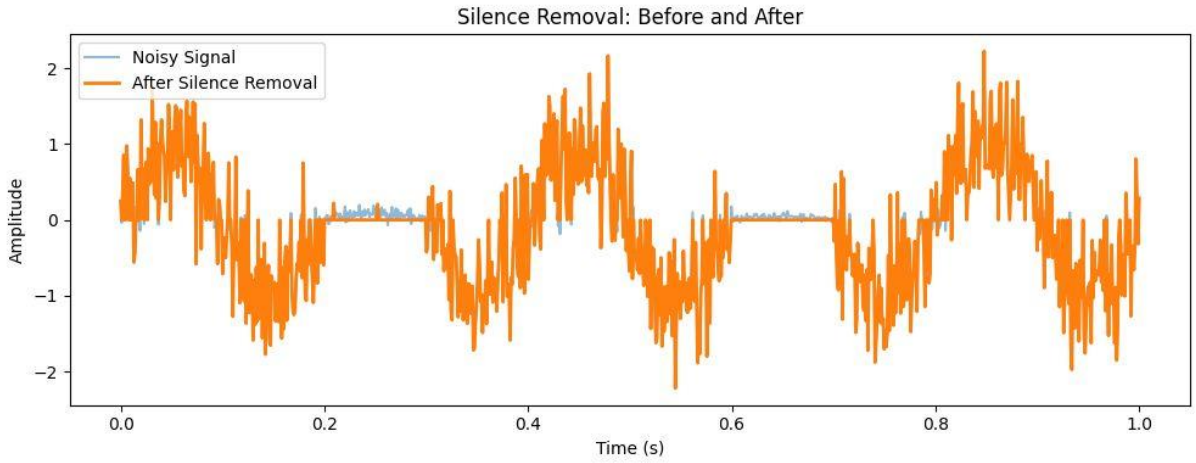

Figure 4.9: Silence Removal (*Generated using Python*)

4.3 Materials and Methods

4.3.1 Justification for Current Research

First put forward by the psychologist Robert Plutchik in 1980, the so-called Emotion Wheel offers a structured account of how affective states in humans can be organised and classified. Its layout rests on four bipolar axes spanning a total of eight foundational affective states — joy paired against sadness, trust against disgust, fear against anger, and surprise against anticipation. Richer or more nuanced affective experiences can then be derived as blends of these foundational categories in varying proportions (Mondal & Gokhale, 2020). An illustration of a single training instance, accompanied by the features pulled from it, is illustrated inFigure 4.10 below.

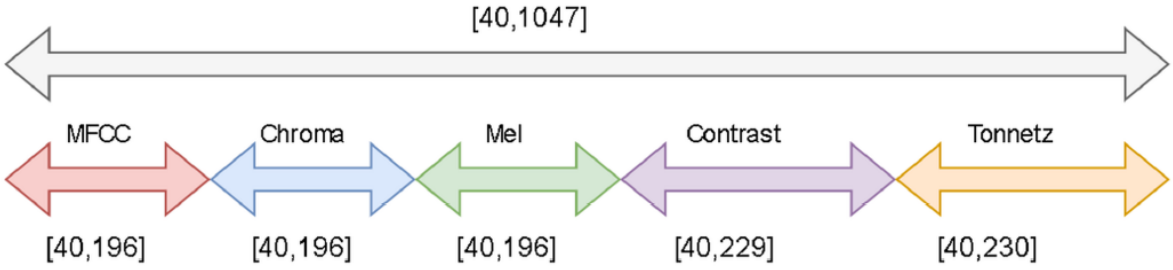

Figure 4.10: Data Frame (*Generated using DrawIO*).

These features were selected based on their widespread use and established effectiveness in speech and audio analysis tasks. MFCCs capture the spectral envelope of speech signals and approximate the human auditory system's response, while Chroma and Tonnetz features provide complementary representations of harmonic and tonal characteristics.

However, the use of these features implicitly assumes that they are sufficiently expressive to capture emotional cues across different languages and speech contexts. The assumption requires qualification. Emotional expression in speech is influenced not only by universal acoustic correlates such as pitch, intensity, and timbre, but also by language-specific phonetic

structures, prosodic patterns, and cultural factors. Consequently, no fixed handcrafted feature set can be considered universally complete or language-independent.

In the study, these features are treated as a practical, widely adopted baseline representation rather than as an exhaustive encoding of emotional information. The hybrid neural network architectures utilised (CNN, LSTM, and CLSTM) enable the model to learn higher-level representations and temporal dependencies beyond the handcrafted features. Nevertheless, the potential limitations of feature generalisability across multilingual datasets should be considered when interpreting the results.

4.3.2 Proposed System

A review of the prior literature surfaces several shortcomings that arise when affect detection is approached through stand-alone baseline designs built around CNN-only or LSTM-only backbones. Such pipelines typically operate on the input signal one fixed-length segment at a time, an arrangement that can struggle to surface the slower-evolving relationships and finer affective shifts that unfold over longer stretches of audio. Affective cues in spoken material tend to be highly contextual in nature, and rigidly windowed processing is poorly suited to representing that kind of context. A second weakness flagged in the literature concerns the quality of the underlying audio itself, given that recordings from realistic settings often contain a mix of background noise, signal distortions, and inconsistencies arising from varying capture conditions. To address these gaps, the present chapter introduces a fresh architectural proposal that combines a Dendritic Neuron Model with an upgraded audio preprocessing front end. A schematic overview of the proposed pipeline is presented in n Figure 4.11. The pipeline put forward in this work draws on a purpose-built Afrikaans spoken-language collection assembled specifically for the study (Norval & Wang, 2022). From each recording, five distinct acoustic descriptors are derived, namely MFCC, Chroma, Mel, Contrast, and Tonnetz. Simulations are first executed using CNN and LSTM architectures as baseline models for comparison. These architectures are included to establish baseline performance and to highlight the improvements achieved by the proposed hybrid and DCLSTM models. Subsequently, a hybrid architecture CLSTM is used, and finally, the DCLSTM network is evaluated. The input features are first processed through a convolutional layer to extract spatial representations, after which the resulting feature maps are passed to the dendritic layer for enhanced non-linear feature integration. The corpus is processed using the pre-processing pipeline to improve the result, and then each pre-processing method is evaluated.

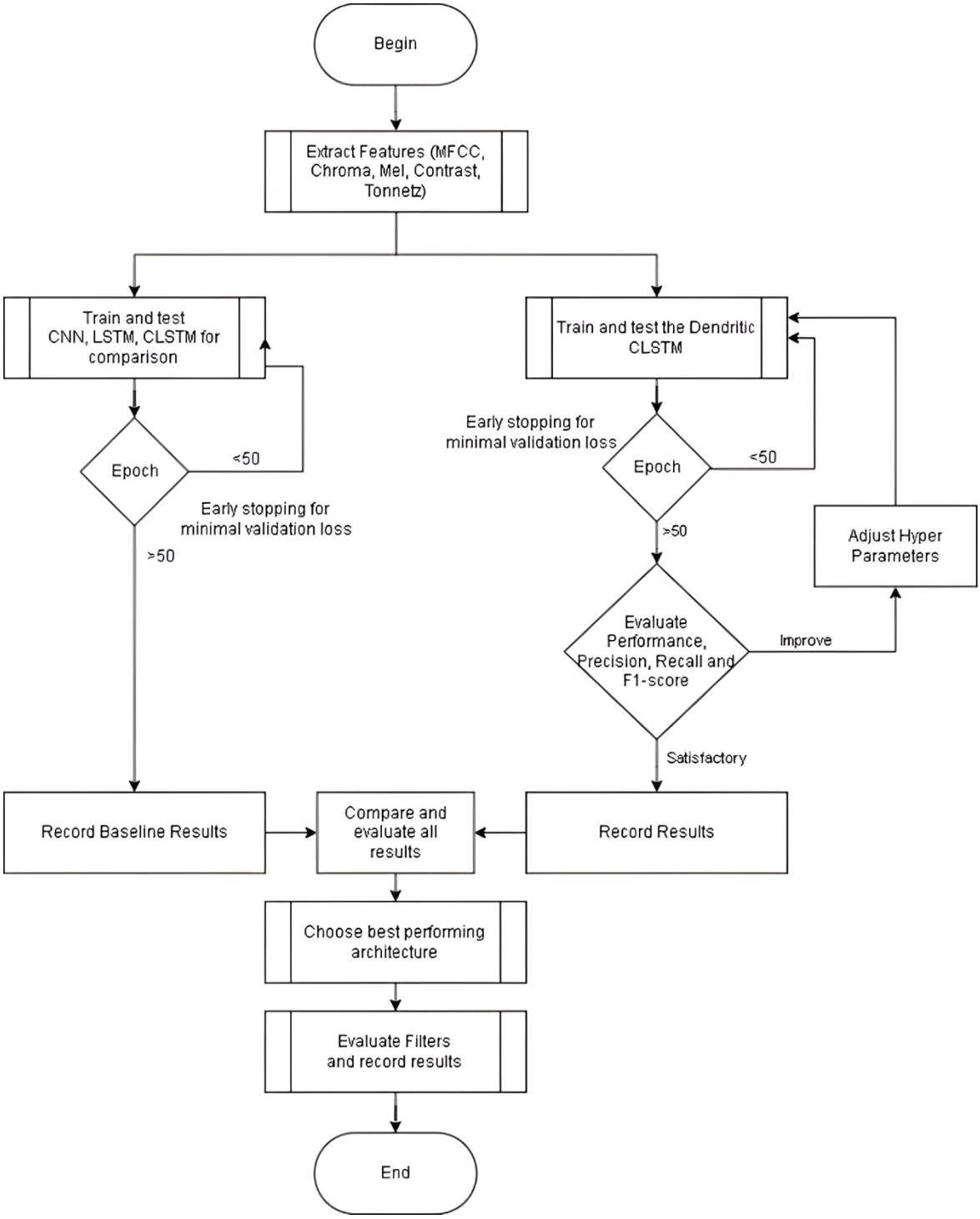

Figure 4.11: Process flowchart of the Proposed System.

The maximum number of epochs was set to 50 based on preliminary experimental analysis, which indicated that model convergence typically occurred within the specified range. An early stopping mechanism based on validation loss was employed to terminate training when no further improvement was observed, thereby preventing overfitting and unnecessary computation.

4.3.2.1 CNN

The CNN network is implemented as a Sequential model in the Keras framework, designed to process structured time-frequency representations of audio signals, such as spectrograms and MFCCs.

Step 1: Input Layer

The network begins with a Conv2D layer with eight filters and a kernel size of 13×13. The input shape is defined as (H, W, 1), where H and W represent the dimensions of the time-frequency representation, and the final dimension indicates a single-channel input derived from audio signals.

Step 2: Batch Normalisation

A BatchNormalisation layer is applied after the convolutional layer. The layer normalises the activations of the previous layer across batches, improving training stability and convergence.

Step 3: Activation Function (ReLU)

The Rectified Linear Unit (ReLU) activation function is applied to introduce non-linearity, enabling the model to learn complex patterns from the input data.

Step 4: Additional Convolutional Layers

The model continues with additional Conv2D layers, each with eight filters. The first two layers use a kernel size of 13×13, followed by a final convolutional layer with a smaller kernel size of 2×2. Each convolutional layer is followed by batch normalisation and ReLU activation.

Step 5: Max Pooling

Two MaxPooling2D layers with pool sizes of (2,1) are used to reduce the spatial dimensions of the feature maps. The operation reduces computational complexity and helps control overfitting.

Step 6: Flattening

A Flatten layer converts 2D feature maps into a one-dimensional vector for processing by the fully connected layers.

Step 7: Fully Connected Layer

A Dense layer with 64 units is applied to interpret the extracted features. The layer is followed by batch normalisation and ReLU activation.

Step 8: Dropout Regularisation

A Dropout layer with a rate of 0.2 is used to prevent overfitting by randomly deactivating a fraction of neurons during training.

Step 9: Output Layer

The final Dense layer contains num_labels units, corresponding to the number of emotion classes. A softmax activation function is used to produce probability distributions for multi-class classification.

Step 10: Model Compilation
The model is compiled using the Adam optimiser and categorical cross-entropy loss function. Accuracy is used as the evaluation metric.

Summary
This architecture, consisting of repeated convolutional layers with batch normalisation and ReLU activation, interspersed with max pooling and followed by fully connected layers, is designed to extract hierarchical features from time-frequency representations of audio signals, making it well-suited for speech emotion recognition tasks.

4.3.2.2 LSTM

The LSTM network is implemented as a Sequential model using Long Short-Term Memory (LSTM) units, designed to process sequential data derived from audio features such as MFCCs and related representations.

Step 1: Input Sequence
The model processes input sequences of shape (num_timesteps, num_features), where num_timesteps denotes the number of time steps in the sequence, and num_features denotes the number of extracted audio features at each time step.

Step 2: LSTM Layer
The network begins with an LSTM layer comprising 128 units. The layer captures temporal dependencies in the sequential data, enabling the model to learn how emotional patterns evolve over time. LSTM units are particularly effective at modelling long-range dependencies in speech signals.

Step 3: Dropout Regularisation
A Dropout layer with a rate of 0.5 is applied after the LSTM layer. The approach reduces overfitting by randomly setting a portion of the input units to zero during training, improving generalisation.

Step 4: Fully Connected Layer (Dense 1)
A Dense layer with 32 neurons and a ReLU activation function is applied. The approach introduces nonlinearity and enables the model to learn complex feature representations from the LSTM layer's temporal outputs.

Step 5: Fully Connected Layer (Dense 2)
A second Dense layer with 16 neurons and a tanh activation function is used. The tanh function outputs values in the range (-1, 1), providing an alternative non-linear transformation of the learned features.

Step 6: Output Layer
The final Dense layer contains num_labels neurons, corresponding to the number of emotion classes. A softmax activation function is applied to generate probability distributions for multi-class classification.

Step 7: Model Compilation
The model is compiled using the Adam optimiser. For multi-class classification tasks, the categorical cross-entropy loss function is used, with accuracy as the evaluation metric.

Summary
This LSTM architecture is designed to model temporal dependencies in sequential audio data, enabling effective learning of how emotional characteristics evolve over time, making it well-suited for speech emotion recognition tasks.

4.3.2.3 CLSTM

The Convolutional LSTM (CLSTM) model serves as the baseline hybrid architecture for speech emotion classification, combining convolutional feature extraction with recurrent temporal modelling.

Step 1: Input Representation
The model accepts input tensors of shape (num_timesteps, num_features), where num_timesteps denotes the temporal dimension and num_features the extracted audio features.

Step 2: Convolutional Layer
A one-dimensional convolutional (Conv1D) layer is applied to extract local patterns from the speech signal representation, capturing short-term spectral and temporal features.

Step 3: Max Pooling
A MaxPooling layer is used to reduce the dimensionality of the feature maps, lowering computational complexity and helping to prevent overfitting.

Step 4: Dropout Regularisation (1)
A Dropout layer with a rate of 0.5 is applied to improve generalisation by randomly deactivating neurons during training.

Step 5: LSTM Layer
The processed features are passed to an LSTM layer, which captures long-term temporal dependencies and models how emotional characteristics evolve over time.

Step 6: Fully Connected Layer
The LSTM output is fed into a Dense layer with ReLU activation, enabling the model to learn higher-level feature representations.

Step 7: Dropout Regularisation (2)
A second Dropout layer with a rate of 0.5 is applied to further reduce overfitting.

Step 8: Output Layer
The final Dense layer contains num_labels units, corresponding to the number of emotion classes. A softmax activation function is used to produce probability distributions for multi-class classification.

Step 9: Model Compilation

The model is compiled using the Adam optimiser and categorical cross-entropy loss function, with accuracy used as the evaluation metric.

Step 10: Training Configuration
Training is performed with a batch size of 32/64 for 50-80 epochs, using early stopping based on validation loss.

Step 11: Model Saving
The trained model is saved in TensorFlow (.tf) format to facilitate reproducibility and comparative analysis. Where required, the get_config() method is overridden to ensure proper saving and loading of custom components.

Summary
The CLSTM architecture integrates spatial feature extraction and temporal sequence modelling, enabling effective learning of both local and sequential patterns in speech signals for emotion recognition.

4.3.2.4 DCLSTM

The proposed Dendritic Convolutional Long Short-Term Memory (DCLSTM) model extends the CLSTM architecture by incorporating a biologically inspired dendritic processing layer to enhance non-linear feature integration for speech emotion recognition.

Step 1: Input Representation
The model accepts input tensors of shape (num_timesteps, num_features, 1), where num_timesteps denotes the number of temporal frames, num_features denotes the number of extracted audio features, and the final dimension indicates a single-channel input.

Step 2: Convolutional Feature Extraction
Spatial feature extraction is performed using two stacked Conv2D layers with ReLU activation and same padding. The first layer uses 128 filters with a 3×3 kernel, followed by a second layer with 32 filters and a 3×3 kernel. These layers capture local spatial patterns in time-frequency representations of audio signals.

Step 3: Max Pooling
A MaxPooling2D layer with a pool size of 2×2 reduces the spatial dimensions of the feature maps, lowering computational complexity.

Step 4: Dropout Regularisation (1)
A Dropout layer with a rate of 0.5 is applied to reduce overfitting and improve generalisation.

Step 5: Reshaping for Sequential Processing
The resulting feature maps are reshaped into a sequential representation, where one spatial dimension is treated as the temporal sequence, and the remaining dimensions are flattened into feature vectors.

Step 6: LSTM Layer

An LSTM layer with 64 units (return_sequences = True) is applied to model temporal dependencies and capture how emotional features evolve over time.

Step 7: Flattening
The output of the LSTM layer is flattened into a one-dimensional vector to prepare it for integration with higher-level features.

Step 8: Dendritic Layer
A custom DendriticLayer with 64 units and two dendritic segments per neuron is applied. The layer performs non-linear feature integration by processing inputs through independent dendritic segments before aggregation, mimicking the behaviour of biological neurons and enhancing representational capacity.

Step 9: Fully Connected Layer
The dendritic output is passed through a Dense layer with 32 units and ReLU activation to further refine the learned features.

Step 10: Dropout Regularisation (2)
A second Dropout layer with a rate of 0.5 is applied to further reduce overfitting.

Step 11: Output Layer
The final Dense layer contains num_classes units with softmax activation, producing probability distributions for multi-class emotion classification.

Step 12: Model Compilation
The model is compiled using the Adam optimiser and categorical cross-entropy loss function, with classification accuracy used as the primary evaluation metric.

Step 13: Implementation and Saving
The architecture is implemented using the TensorFlow/Keras functional API. To ensure proper saving and loading of the custom DendriticLayer, the get_config() method is overridden.

Summary
The DCLSTM architecture integrates convolutional, recurrent, and dendritic processing mechanisms, enabling enhanced modelling of complex spatial-temporal patterns in speech signals and improving emotion recognition performance.

4.4 Results

Each experiment was repeated 10 times to account for stochastic variability inherent in deep learning model training.

4.4.1 Training Data

First put forward by the psychologist Robert Plutchik in 1980, the so-called Emotion Wheel offers a structured account of how affective states in humans can be organised and classified. Its layout rests on four bipolar axes spanning a total of eight foundational affective states — joy paired against sadness, trust against disgust, fear against anger, and surprise against

anticipation. Richer or more nuanced affective experiences can then be derived by blending these foundational categories in varying proportions (Mondal & Gokhale, 2020). The label inventory adopted for the present investigation maps directly onto these eight categories drawn from Plutchik's scheme. The audio material itself is sourced from a purpose-built Afrikaans spoken-language collection assembled specifically for this work (Norval & Wang, 2022). Each effective class contributes in the region of one hundred recordings, bringing the overall sample count to approximately eight hundred. From each recording, five distinct acoustic descriptors are extracted and provided to the network during fitting: Chroma, Contrast, Mel, MFCC and Tonnetz.

In the study, these features are therefore treated as a robust baseline representation rather than an exhaustive encoding of emotional information. The use of hybrid neural network architectures (CNN, LSTM, and CLSTM) enables the model to learn higher-level representations and temporal dependencies beyond the handcrafted features. Nevertheless, the potential limitations of feature generalisability across multilingual datasets should be considered when interpreting the results.

4.4.2 Audio Pre-Processing

Table 4.1 shows the two architectures, CNN and LSTM. The number of samples tested was also increased from 40 to 128. Every architecture is tested 10 times.

Table 4.1: CNN Architecture Simulation Results.

Run	Accuracy	Macro F1	Weighted F1	Balanced Acc	Loss
1	0.5550	0.5393	0.5561	0.5550	3.111
2	0.4650	0.4511	0.4623	0.465	3.4603
3	0.5850	0.5698	0.5870	0.5850	2.8535
4	0.5900	0.5629	0.5826	0.5900	3.0158
5	0.5800	0.5608	0.5773	0.5800	2.4431
6	0.5600	0.5421	0.5580	0.5600	2.8391
7	0.5300	0.5165	0.5303	0.5300	3.275
8	0.5550	0.5566	0.5638	0.5550	3.0308
9	0.5550	0.5409	0.5543	0.5550	2.8113
10	0.5150	0.5041	0.5069	0.5150	3.5307

Table 4.2: CNN Summary Table (Mean ± Std)

Metric	**Mean**	**Std Dev**
Accuracy	0.5490	± 0.0376
Macro F1	0.5344	± 0.0358
Weighted F1	0.5479	± 0.0386
Balanced Accuracy	0.5490	± 0.0376
Loss	3.0371	± 0.3269
Best Val Accuracy	0.5975	± 0.0210

One example of the summary metric calculation is provided for the CNN accuracy values reported in Table 4.1. The mean accuracy is calculated as the arithmetic average of the 10 runs:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{4.1}$$

Substituting the 10 CNN accuracy values:

$$\bar{x} = \frac{0.5550+0.4650+0.5850+0.5900+0.5800+0.5600+0.5300+0.5550+0.5550+0.5150}{10} = 0.5490 \tag{4.2}$$

The standard deviation is calculated as:

$$s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2} \tag{4.3}$$

which gives:

$$s = 0.0376 \tag{4.4}$$

Thus, the CNN accuracy in Table 4.2 is reported as 0.5490 ± 0.0376.

The confusion Matrix for LSTM is illustrated in Figure 4.12.

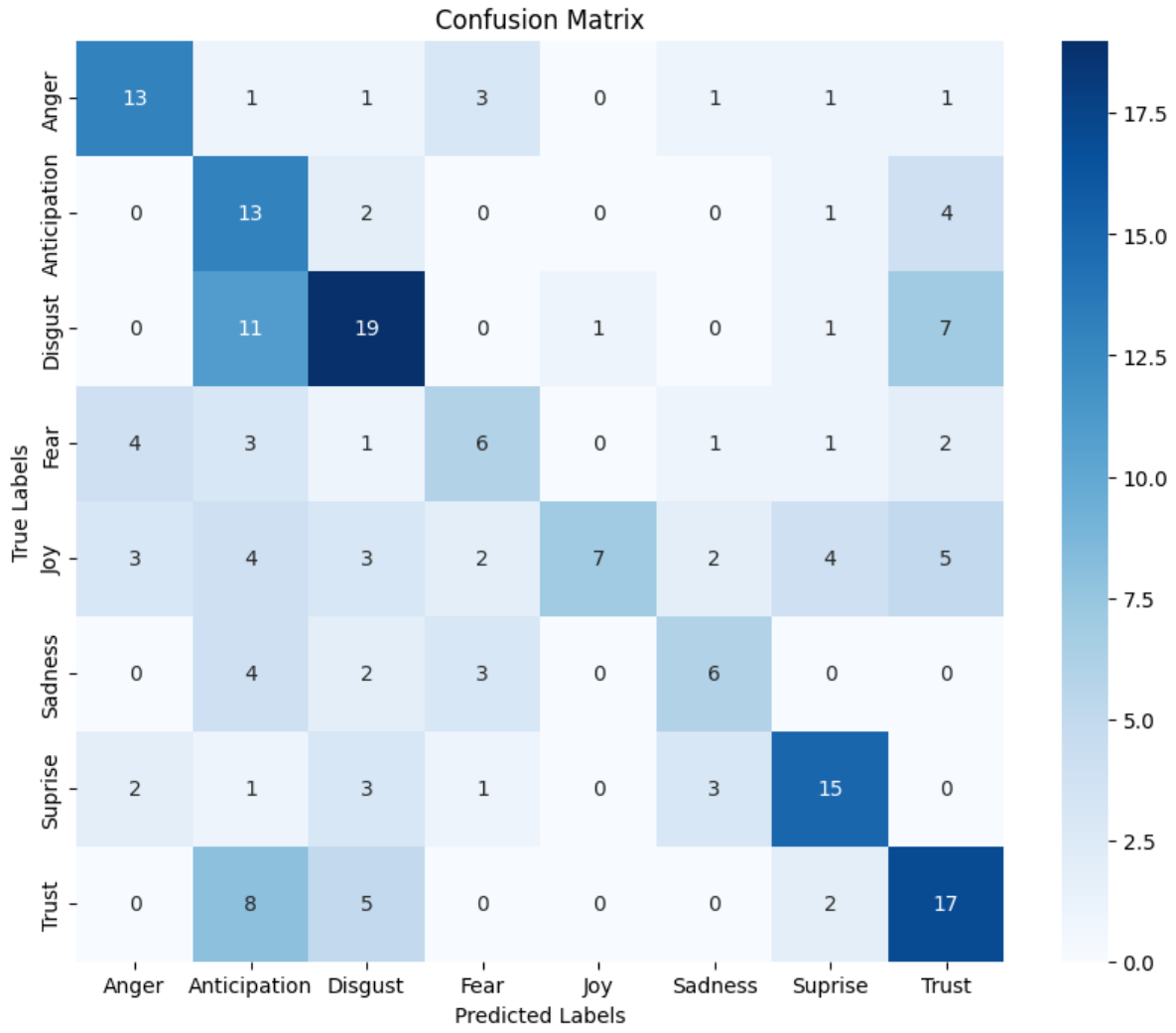

Figure 4.12: LSTM Confusion Matrix.

Table 4.3 show the LSTM architecture performance for each run for Accuracy, Precision, F1Score, Recall and Loss.

Table 4.3: LSTM Architecture Simulation Results

Run	Accuracy	Macro F1	Weighted F1	Balanced Acc	Loss
1	0.5350	0.5291	0.5417	0.5367	1.7160
2	0.5850	0.5916	0.5817	0.5895	1.6503
3	0.5750	0.5649	0.5626	0.5865	1.5898
4	0.5700	0.5566	0.5663	0.5654	1.6536
5	0.5600	0.5446	0.5510	0.5564	1.6109
6	0.5850	0.5716	0.5748	0.5918	1.6690
7	0.5550	0.5365	0.5483	0.5517	1.5597
8	0.5350	0.5275	0.5324	0.5321	1.6928
9	0.5350	0.5279	0.5221	0.5502	1.6363
10	0.5700	0.5493	0.5615	0.5608	1.6061

Table 4.4: LSTM Summary Table (Mean ± Std)

Metric	Mean	Std Dev
Accuracy	0.5605	0.0189
Macro F1	0.5500	0.0203
Weighted F1	0.5542	0.0178
Balanced Accuracy	0.5621	0.0202
Loss	1.6384	0.0454
Best Val Accuracy	0.5605	0.0189

The confusion Matrix for CLSTM is illustrated in Figure 4.13.

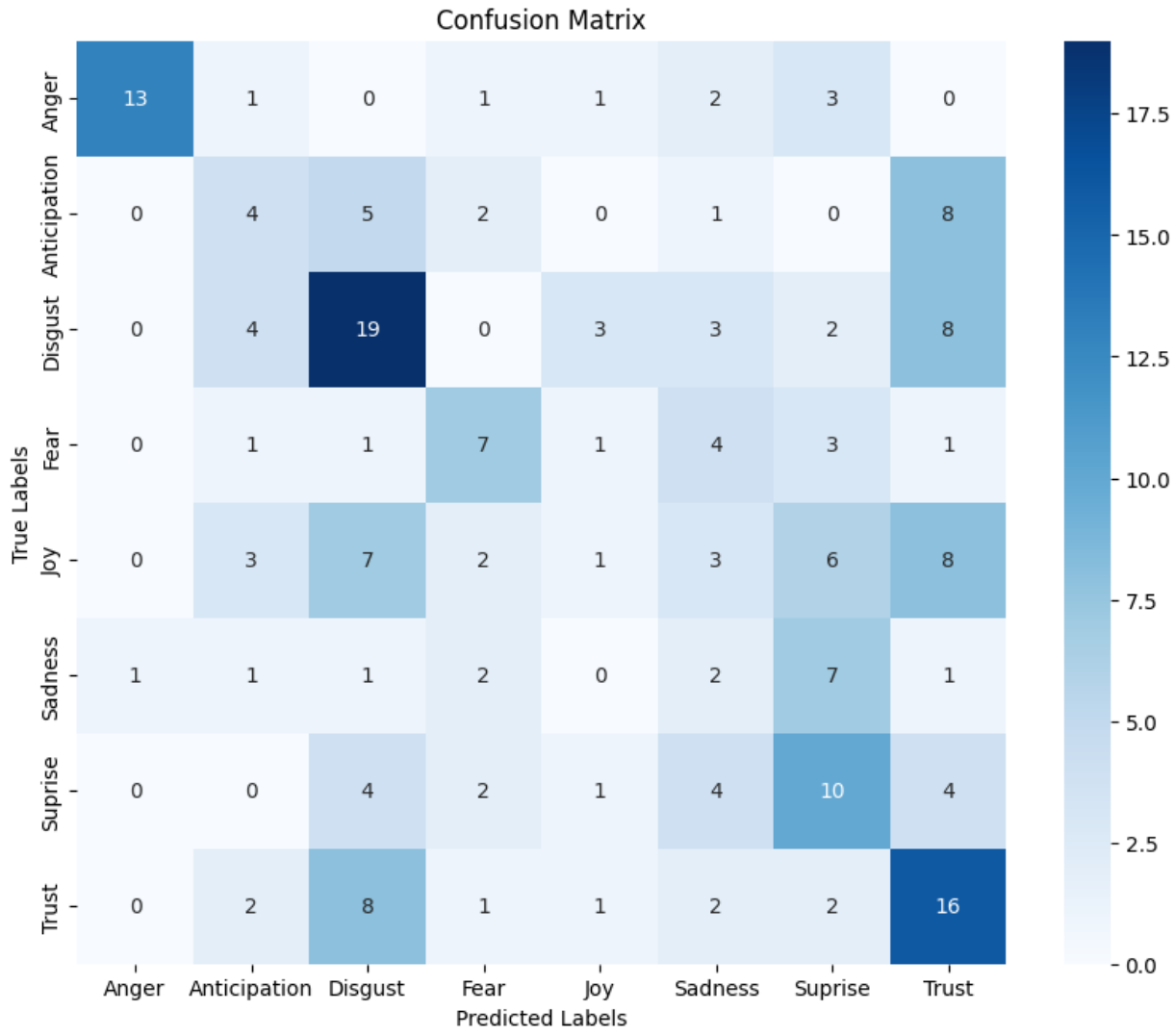

Figure 4.13: CLSTM Confusion Matrix.

The results below in Table 4.5 and Table 4.7 show the two architectures, CLSTM and DCLSTM, with Dendritic Neurons. The number of samples tested was also increased from 40 to 128. Every architecture is tested ten times.

Table 4.5: CLSTM Architecture Simulation Results.

Run	Accuracy	Macro F1	Weighted F1	Balanced Acc	Loss
1	0.6300	0.6260	0.6351	0.6300	1.1209
2	0.6150	0.5977	0.6154	0.6150	1.0774
3	0.6350	0.6095	0.6264	0.6350	1.0656
4	0.6750	0.661	0.6762	0.6750	1.0294
5	0.6400	0.6194	0.6382	0.6400	1.0541
6	0.6750	0.6669	0.6789	0.6750	1.0658
7	0.6400	0.6311	0.6428	0.6400	1.0994
8	0.6550	0.6508	0.6592	0.6550	1.0538
9	0.6550	0.6314	0.6578	0.6550	1.0560
10	0.6800	0.6710	0.6833	0.6800	1.0199

Table 4.6: CLSTM Summary Table (Mean ± Std)

Metric	Mean	Std Dev
Accuracy	0.6500	± 0.0217
Macro F1	0.6365	± 0.0250
Weighted F1	0.6513	± 0.0234
Balanced Accuracy	0.6500	± 0.0217
Loss	1.0642	± 0.0300
Best Val Accuracy	0.6690	± 0.0115

The confusion Matrix for DCLSTM is illustrated in Figure 4.14.

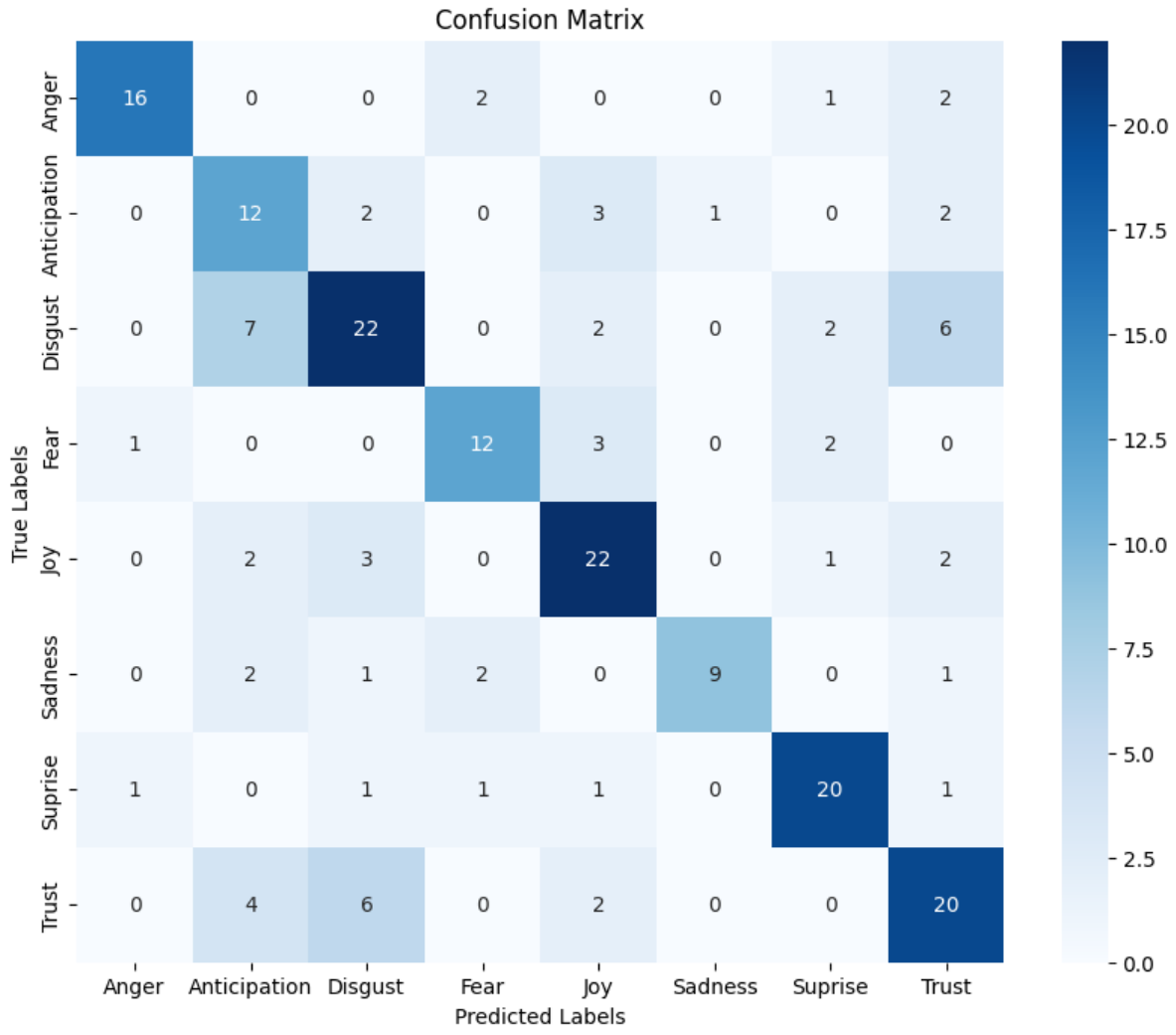

Figure 4.14: DCLSTM Confusion Matrix.

Table 4.7: DCLSTM Architecture Simulation Results.

Run	Accuracy	Macro F1	Weighted F1	Balanced Acc	Loss
1	0.7050	0.6837	0.7047	0.7050	0.8779
2	0.6450	0.6375	0.6465	0.6450	0.9093
3	0.7200	0.7180	0.7210	0.7200	0.8141
4	0.7000	0.6874	0.7064	0.7000	0.8855
5	0.7000	0.6924	0.7059	0.7000	0.7835
6	0.7050	0.6937	0.7026	0.7050	0.8909
7	0.7750	0.7674	0.7814	0.7750	0.8512
8	0.6650	0.6578	0.6683	0.6650	0.9551
9	0.7250	0.7117	0.7267	0.7250	0.8229
10	0.7350	0.7046	0.7339	0.7350	0.9427

Table 4.8: DCLSTM Summary Table (Mean ± Std)

Metric	Mean	Std Dev
Accuracy	0.7075	± 0.0359
Macro F1	0.6954	± 0.0350
Weighted F1	0.7097	± 0.0364
Balanced Accuracy	0.7075	± 0.0359
Loss	0.8733	± 0.0557
Best Val Accuracy	0.7505	± 0.0222

To ensure that the observed performance differences between models are not attributable to random variation, each model was evaluated over 10 independent runs. The mean and standard deviation for all evaluation metrics were computed, and 95% confidence intervals were derived to assess the stability of the results. A paired t-test was conducted on model pairs, with accuracy as the primary evaluation metric. The results indicate that the proposed DCLSTM model significantly outperforms the baseline CLSTM model ($p < 0.001$). In addition to

statistical significance, effect sizes were calculated using Cohen's d. The results demonstrate large to very large effect sizes across all major comparisons, confirming that the observed improvements are not only statistically significant but also practically meaningful.

Table 4.9: Comparative Performance of CNN, LSTM, CLSTM, and DCLSTM Models (Mean ± Standard Deviation with 95% Confidence Intervals)

Model	**Mean**	**Std Dev**	**N**	**SE**	**95% CI Low**	**95% CI High**
CNN	0.5490	0.0376	10	0.0119	0.5221	0.5759
LSTM	0.5605	0.0189	10	0.006	0.547	0.574
CLSTM	0.6500	0.0217	10	0.0069	0.6345	0.6655
DCLSTM	0.7075	0.0359	10	0.0114	0.6818	0.7332

Table 4.10: Statistical Significance Results for Model Comparisons (Paired t-test and Cohen's d)

Comparison	**Mean Diff**	**p-value**	**t-stat**	**Cohen's d**	**Significant**	**Effect Size**
DCLSTM vs CLSTM	0.0575	0.0008	4.9667	1.5706	Yes	Very Large
DCLSTM vs CNN	0.1585	<0.0001	10.9513	3.4631	Yes	Very Large
DCLSTM vs LSTM	0.147	<0.0001	~11.5	~5.15	Yes	Very Large
CLSTM vs CNN	0.101	<0.0001	8.7505	2.7672	Yes	Very Large
CLSTM vs LSTM	0.0895	<0.0001	~9.8	~4.40	Yes	Very Large
CNN vs LSTM	0.0115	0.4	0.86	0.39	No	Small

The DCLSTM model achieved the highest performance among the evaluated architectures,

with a mean accuracy of approximately 70.75% across multiple runs. Statistical analysis confirms that the improvement is significant ($p < 0.001$) and is supported by large effect sizes, indicating that the performance gains are both statistically and practically meaningful.

To ensure transparency and reproducibility, the full experimental results for all preprocessing filter configurations are provided in the Appendices. Each filter configuration was evaluated over 10 independent runs, and the complete set of results, including all performance metrics, is included for reference. The summary statistics presented in the chapter reflect the aggregated outcomes of these runs, while only the most relevant and best-performing configurations are discussed in the main text for clarity.

Table 4.11: Pre-Processing Results (The full calculation results can be viewed in the Appendices, A.1.1 through A.1.7)

Filter	**Acc Mean**	**Acc Std**	**Best Acc**	**Macro F1 Mean**	**Loss Mean**
Kalman_2	0.7340	0.0223	0.7800	0.7169	0.7720
Silence	0.6839	0.0291	0.7236	0.6751	0.9367
Kalman_1	0.6255	0.0331	0.6900	0.6075	1.0920
Generic NR	0.6087	0.0160	0.6304	0.5765	1.1302
Spectral G	0.4795	0.0382	0.5450	0.4559	1.4470
Spectral Sub	0.4690	0.1009	0.5450	0.4303	1.4323
Bandpass	0.4505	0.1104	0.5500	0.4229	1.4784

The confusion Matrix for DCLSTM pre-processing is illustrated in Figure 4.15.

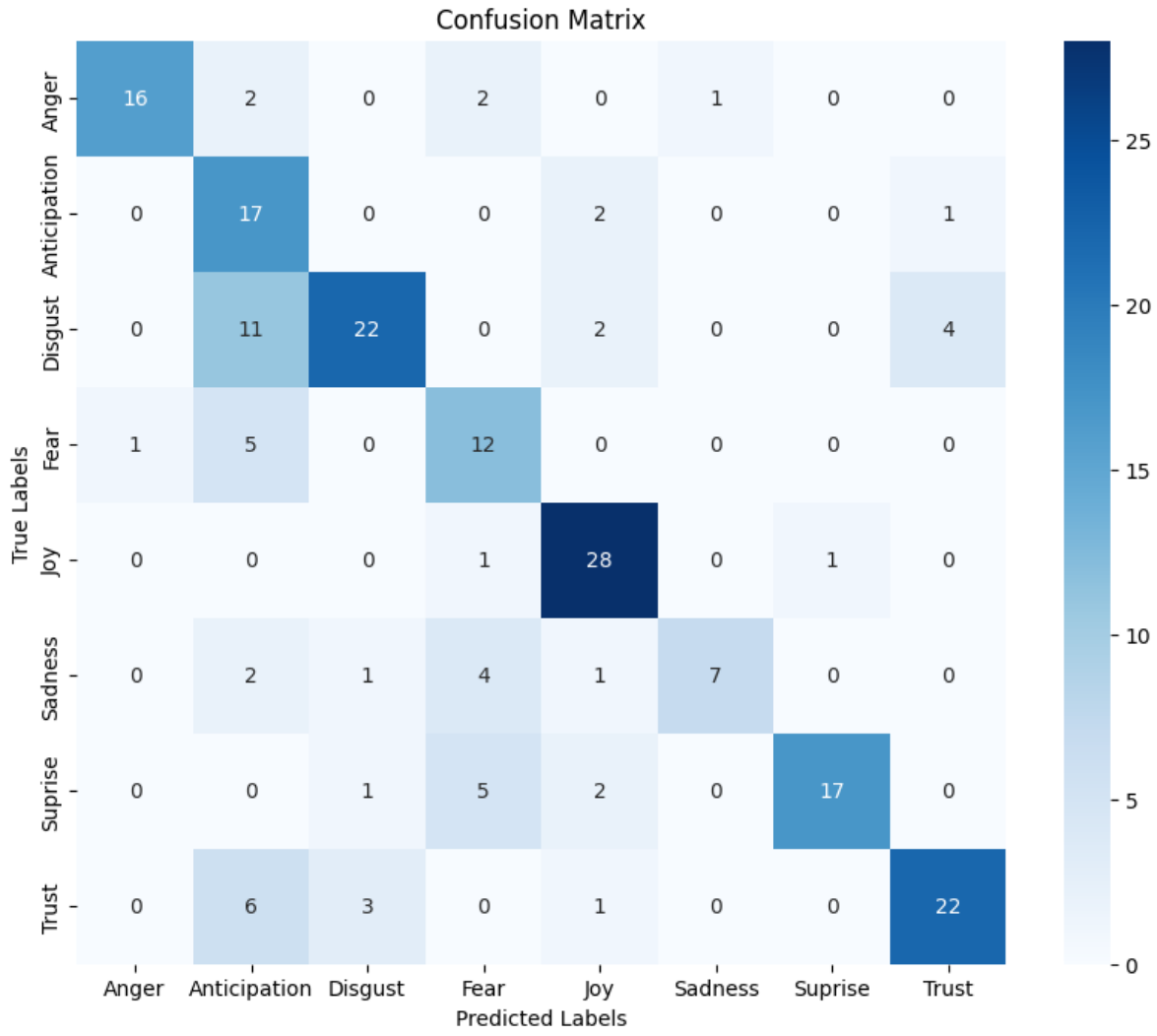

Figure 4.15: Pre Processing CLSTM Confusion Matrix.

As delineated in Table 4.7 the Dendritic Convolutional Long Short-Term Memory (DCLSTM) is the most efficacious architecture. The network's preeminence is attributed to its capacity for hierarchical processing; dendritic computations enable the integration of a hierarchical computational approach, enriching the model's depth without increasing the number of layers. The amalgamation of dendritic structures, convolutional operations, and Long Short-Term Memory units allows the DCLSTM to apprehend a broad spectrum of characteristics, encompassing immediate subtleties and extensive temporal dependencies.

The second Kalman filter variant (Kalman_2) consistently achieved the highest accuracy across the evaluated configurations, with an average accuracy of approximately 78.50% over the ten test runs using the DCLSTM architecture (As shown in Table 4.5). The performance advantage was consistently observed compared to alternative preprocessing techniques, including spectral gating and generic noise reduction methods, which yielded lower accuracy. The reported results reflect performance under the specific experimental conditions and dataset used in the research. The selection of the Kalman_2 filter was based on comparative empirical performance across the evaluated preprocessing configurations Table 4.11. However, a formal

sensitivity analysis examining performance variability across differing noise levels, recording environments, speaker demographics, or speech length was beyond the scope of the research. The limitation represents a recognised limitation and is recommended as a direction for future work.

4.4.3 Computational Complexity Analysis

Beyond classification accuracy, the computational cost of each architecture was quantified to evaluate practical feasibility. For each model, six measures are reported: trainable parameter count, the number of floating-point operations (FLOPs) per forward pass, and mean per-epoch training time on the Afrikaans corpus. Parameters were extracted directly from model.summary(); FLOPs were estimated using TensorFlow's static graph profiler; epoch times were averaged over six independent training epochs on the Afrikaans corpus with batch size 32 on the same hardware platform. Table 4.12 reports the resulting figures.

* LSTM uses the 2-D MFCC matrix without a channel dimension; this is standard for sequence-only models.

Table 4.12: Computational Complexity.

Model	Input Shape	Parameters	FLOPs (forward pass)	Mean epoch time (s)	Std (s)
CNN	(40, 1047, 1)	42,919,816	6.37 G	0.72	0.03
LSTM	(40, 1047)*	449,096	0.01 G	0.31	0.02
CLSTM	(40, 1047, 1)	8,743,048	6.28 G	0.9	0.03
DCLSTM	(40, 1047, 1)	8,923,528	6.28 G	0.93	0.03

4.5 Model Training, Convergence, and Hyperparameter Configuration

The reported performance metrics reflect results obtained under the defined experimental conditions on the Afrikaans corpus (and the multilingual dataset for pre-processing comparisons). They do not include systematic sensitivity analysis across variables such as noise levels, speaker characteristics, recording quality, or speech segment length.

All models in this chapter (CNN, LSTM, CLSTM, and DCLSTM) were trained using the Adam optimiser (learning rate = 0.001) with categorical cross-entropy loss and accuracy as the primary evaluation metric. The DCLSTM architecture integrates convolutional feature extraction, LSTM temporal modelling, and a custom dendritic layer to capture more complex non-linear dependencies. Convergence behaviour was assessed by analysing the training and validation performance across epochs. Early stopping (patience = 12 on validation loss, restoring best weights) was employed to prevent overfitting. The training history demonstrated a consistent decrease in loss and stabilisation of validation accuracy in later epochs, indicating

that the models converged without significant overfitting. The highest validation accuracy achieved during training was recorded and used as an indicator of optimal model performance.

In addition to standard training, the DCLSTM experiments incorporated light data augmentation (Gaussian noise + small time shifting) and softened class weights ($\alpha = 0.5$) to improve robustness on the modestly imbalanced Afrikaans corpus. The hyperparameters used across all models were selected based on empirical testing and established practices in deep learning for speech processing. Although a comprehensive sensitivity analysis was beyond the scope of this study, the selected configuration yielded stable and reproducible results across the ten independent experimental runs. Table 4.13 summarises the key hyperparameters used for the models in this chapter.

Table 4.13: Hybrid Model Hyperparameters

Component	**Parameter**	**Value**
Input	Input shape	(num_timesteps, num_features, 1)
Conv2D Layer 1	Filters	128
Conv2D Layer 1	Kernel size	(3 × 3)
Conv2D Layer 1	Activation	ReLU
Conv2D Layer 2	Filters	64
Conv2D Layer 2	Kernel size	(3 × 3)
Conv2D Layer 2	Activation	ReLU
MaxPooling2D	Pool size	(2 × 2)
Dropout (Conv)	Dropout rate	0.5
LSTM Layer	Units	64
LSTM Layer	Return sequences	TRUE
Dendritic Layer (DCLSTM)	Units	64
Dendritic Layer (DCLSTM)	Dendritic segments	3
Dense Layer	Units	64
Dense Layer	Activation	ReLU
Dropout (Dense)	Dropout rate	0.5
Output Layer	Units	num_classes (= 8)
Output Layer	Activation	Softmax
Optimiser	Type	Adam (lr = 0.001)
Loss Function	—	Categorical Cross-Entropy
Batch Size	—	32
Max Epochs	—	100
Early Stopping	Patience	12 (on validation loss)
Data Handling	Shuffle	TRUE
Augmentation	Noise factor / shift	0.03 / 0.1
Class Weighting	Softened α	0.5
Validation	—	Held-out test set

4.6 Discussion

In conclusion, exploring advanced neural network architectures and preprocessing techniques in Speech Emotion Recognition (SER) presents a compelling narrative of technological

evolution aimed at overcoming inherent limitations and significantly enhancing accuracy. The empirical analysis, as evidenced by the results detailed in Table 4.1, Table 4.2, Table 4.3, Table 4.4, Table 4.5, Table 4.6, Table 4.7, Table 4.8, Table 4.9, Table 4.10 and Table 4.11 reveals a comparative performance evaluation of CNN, LSTM, CLSTM, and Dendritic Convolutional Long Short-Term Memory (DCLSTM) models. Notably, the augmentation of the sample size from 40 to 128 has provided invaluable insights into the scalability and robustness of each architecture under varying data volumes. The limitations highlighted, including the reduced efficiency for temporally varying input data, susceptibility to overlearning, and constraints imposed by large-layer-internal architectures, underscore the critical challenges in current SER methodologies. However, the advent of DCLSTM, particularly when enhanced with dendritic computations, marks a significant milestone in addressing these challenges. The hierarchical processing capability inherent in the DCLSTM architecture, as evidenced by its stronger performance relative to the baseline architectures evaluated in the research

Table 4.7. It demonstrates its potential to more adeptly capture a wide array of characteristics, ranging from nuanced to temporally extended dependencies. Moreover, the impact of preprocessing techniques on SER accuracy is profoundly illustrated in Table 4.11, where various methods, including noise removal and Kalman filters, were evaluated. The second variant of the Kalman filter emerged as the optimal preprocessing technique, attributing its success to effectively handling noisy data and smoothly transitioning emotional states. The findings underscore the nuanced complexity of emotional expressions within audio data and the necessity for sophisticated preprocessing to unravel these intricacies. The CNN and LSTM models provide a solid foundation for understanding spatial and temporal features within data, respectively. However, their integration into hybrid models such as CLSTM and further into DCLSTM with dendritic neurons introduces an architectural sophistication that transcends the capabilities of their standalone counterparts. The evolution signifies a deliberate shift towards models that learn and integrate features that mimic the biological intricacies of human cognitive processes. Future directions in SER research should continue to focus on refining hybrid models and exploring preprocessing techniques that align with the dynamic nature of audio data. The advancements in DCLSTM models, mainly through incorporating dendritic layers, offer a promising avenue for achieving greater accuracy and robustness in emotion recognition. Ultimately, the quest for enhancing SER accuracy through innovative architectural designs and preprocessing methodologies remains a pivotal endeavour in the broader context of human-computer interaction, with the potential to redefine the boundaries of machine empathy and emotional intelligence.

4.7 Conclusion

In the chapter, an approach to speech emotion recognition is presented that integrates advanced preprocessing techniques with hybrid neural network architectures. Models such as CNNs, LSTMs, and the proposed DCLSTM architecture are explored. Models such as CNNs, LSTMs, and the proposed DCLSTM architecture are evaluated for their ability to capture emotional characteristics in complex audio signals. The DCLSTM architecture, inspired by biological dendritic neurons, combines convolutional layers with LSTMs to integrate spatial and temporal

features. In addition, an audio preprocessing pipeline incorporating noise reduction, spectral gating, and Kalman filtering techniques was implemented to improve the quality of the input data. Experimental results indicate that the integrated system achieved improved performance relative to the baseline CNN and LSTM models evaluated in the study. These findings highlight the potential of hybrid architectures for enhancing speech emotion recognition within the constraints of the available dataset and experimental setup.

While transformer-based architectures such as wav2vec 2.0 and HuBERT have demonstrated strong performance in recent speech emotion recognition research, they were not included in the evaluation. These models typically require substantially larger training datasets and significantly higher computational resources than were available for the study. The present work focuses on hybrid CLSTM architectures with dendritic enhancements as a computationally efficient alternative suitable for the relatively modest-sized Afrikaans corpus.

In summary, the chapter demonstrated that the DCLSTM architecture, incorporating dendritic processing, outperforms standard CNN and LSTM models on the Afrikaans corpus. These findings establish the value of dendritic layers for capturing nuanced temporal and spectral dependencies in emotional speech and provide a clear architectural direction. The design principles of dendritic enhancement proven effective here are extended in Chapter 5 through the DenCaps model, enabling evaluation of scalability and cross-lingual generalisation on a significantly larger multilingual dataset.

CHAPTER 5: Multilingual Detection Systems

Evaluating Speech Emotion Recognition (SER) across multiple speech datasets presents unique challenges. The chapter focuses on a multilingual dataset combining languages such as Afrikaans, English, German, French, Greek, Mongolian, and Japanese, comprising approximately 23,000 clips. A novel neural network architecture, DenCaps, is introduced to improve accuracy. The architecture leverages Dendritic Layers inspired by biological neurons and is coupled with Capsule Networks. Comparatively, DenCaps is evaluated against CLSTM architectures, showing improved performance. The research highlights the potential of DenCaps for multilingual models, offering a biologically inspired approach that demonstrates improved performance relative to the baseline models evaluated in the research. Training multilingual models for SER requires robust architectures and advanced methodologies. To address the complexities of multilingual data processing, hybrid models like DenCaps and CLSTM are explored. The incorporation of Capsule Networks and dendritic layers is proposed to enhance accuracy across diverse languages, thereby addressing key challenges in multilingual SER.

5.1 Introduction

In recent years, advanced NN architectures such as Dendritic Neural Networks (DNNs) and Capsule Networks (CapsNets) have revolutionised the field of Speech Emotion Recognition (SER), significantly improving the ability of machines to understand human emotions from speech. Traditional models like CNNs and LSTMs have achieved moderate success, but their limitations in capturing emotional speech's dynamic and nuanced nature are becoming increasingly apparent. To address the gap, the proposed hybrid model, DenCaps, combines the advanced feature extraction capabilities of DNNs with the hierarchical pattern recognition of CapsNets, offering a novel approach to SER (Patrick, Adekoya, Mighty, & Edward, 2022).

Inspired by the complex processes in biological neurons, Dendritic Neural Networks (Ding, Yu, Gu, Gao, & Zhang, 2024) incorporate dendritic layers that enhance the network's ability to integrate synaptic inputs. The approach provides a more refined interpretation of temporal dynamics in audio signals, which is crucial for accurately capturing emotional states. On the other hand, Capsule Networks, introduced by Geoffrey Hinton, maintain spatial hierarchies and relationships within data, making them highly resilient to variations in input orientation, scale, and pose. The property is particularly beneficial in SER, where variations in acoustic environments and speaker nuances must be consistently handled.

Drawing on the model refinements and preprocessing insights developed in Chapter 4, the chapter extends the Demonstration and Evaluation phases of the Design Science Research Methodology to multilingual datasets, introducing and assessing the novel DenCaps architecture for cross-language speech emotion recognition. The DenCaps architecture and the multilingual evaluation constitute the engineered artefacts addressing Objective 3 of the research.

By integrating these two advanced architectures, the DenCaps model addresses the

shortcomings of traditional CNN and CLSTM models in SER, providing a more robust framework for emotion detection. With dendritic layers enhancing feature extraction and CapsNets preserving hierarchical relationships, the hybrid DenCaps model improves accuracy, particularly in diverse, real-world acoustic conditions. Building on the single-language optimisation conducted in Chapter 4, the chapter extends the investigation to a multilingual context. The architectural insights gained from DCLSTM — particularly dendritic non-linear feature integration — are further developed in the proposed DenCaps model to address the increased complexity and variability of cross-lingual emotional speech. The multilingual corpora used in this chapter are aggregated from existing publicly available datasets, each released in pre-cleaned form by their original curators; a unified low-level noise-reduction pipeline (such as the Kalman_2 filter validated on the Afrikaans corpus in Chapter 4) was therefore not re-applied at the multilingual scale, since the source corpora differ in recording conditions and prior preprocessing. Direct application of Kalman_2 to the multilingual pipeline is identified as future work in §6.2.

5.2 Related Works

5.2.1 Advanced Network Models

The exploration of "Advanced Network Models" in the context of Speech Emotion Recognition (SER) underscores the potential of leveraging the distinct characteristics of Dendritic Neural Networks (Ding, Yu, Gu, Gao, & Zhang, 2024) and Capsule Networks (CapsNets) to address the inherent challenges of the field (Zhao, Li, Zeng, Wang, & Zhang, 2022). Integrating dendritic layers into neural network architectures facilitates the emulation of biological neural processes, particularly in the sophisticated integration of synaptic inputs, allowing for a nuanced interpretation of complex input patterns. Dendritic layers enhance the network's ability to process the temporal dynamics of audio signals, which is crucial for capturing the fleeting nature of speech in emotional states. CapsNets, on the other hand, offer an architecture designed to recognise hierarchical patterns within data, which is especially beneficial for tasks that require spatial understanding, such as SER. By encoding spatial hierarchies and relationships within their capsules, these networks are adept at maintaining consistency of recognition despite variations in orientation, scale, and pose of the input data—attributes that can be translated into the audio domain as consistency across varied acoustic environments and speaker nuances. The CLSTM network, a hybrid model, combines the spatial feature recognition capabilities of CNNs with the sequential data processing strengths of LSTMs. In the realm of SER, the CLSTM model stands out for its ability to process and integrate features across time and space, offering a robust framework for detecting and classifying emotions in speech. Combining these cutting-edge models—Dendritic Neural Networks, CapsNets, and CLSTM—into an ensemble approach for audio emotion detection offers a compelling strategy to enhance the robustness and interpretability of SER systems. Such an ensemble capitalises on the complementary strengths of each model: the temporal sequence learning of CLSTM, the advanced feature integration of dendritic layers, and the strong spatial processing capabilities of CapsNets. Together, they form an advanced network model framework that holds promise for improving the accuracy and reliability of emotion detection from audio signals, paving the

way for more empathetic and contextually aware human-computer interaction.

5.2.2 Dendritic Neural Network Layer

The artificial neural network landscape has, over the last few years, witnessed a noticeable pivot in the direction of architectures whose inner workings draw inspiration from the way biological neural tissue actually behaves (Ji, Tang, Zhao, Tang, & Todo, 2022). One particularly consequential strand of this trend involves grafting dendrite-like computational mechanisms onto conventional artificial neuron formulations, an idea that has given rise to what are now called dendritic neuron layers. The motivation here is straightforward enough: classical artificial neurons trace their lineage back to the point-neuron abstraction, a simplification that effectively collapses the entire neuronal body into a single summing junction and consequently leaves no room to mimic the elaborate processing carried out across a real dendritic tree. In actual biological neurons, by contrast, the dendritic arbour combines incoming synaptic signals through a rich web of nonlinear interactions, supporting input-to-output mappings that are considerably more expressive than a flat weighted sum could ever be.

The advent of dendritic neuron layers within neural networks heralds a significant enhancement in computational power, enabling more refined and efficient recognition and processing of patterns in datasets. The proposed approach aims to reduce the gap between the existing deep learning models and the complex dynamical systems observed in biological neural circuits. The concept is illustrated in Figure 5.1. However, the effective deployment of dendritic neuron layers, their superiority relative to current neural architectures, and methodologies for their efficient training are still under active exploration, indicating a need for extensive research in these areas (Gul, 2023).

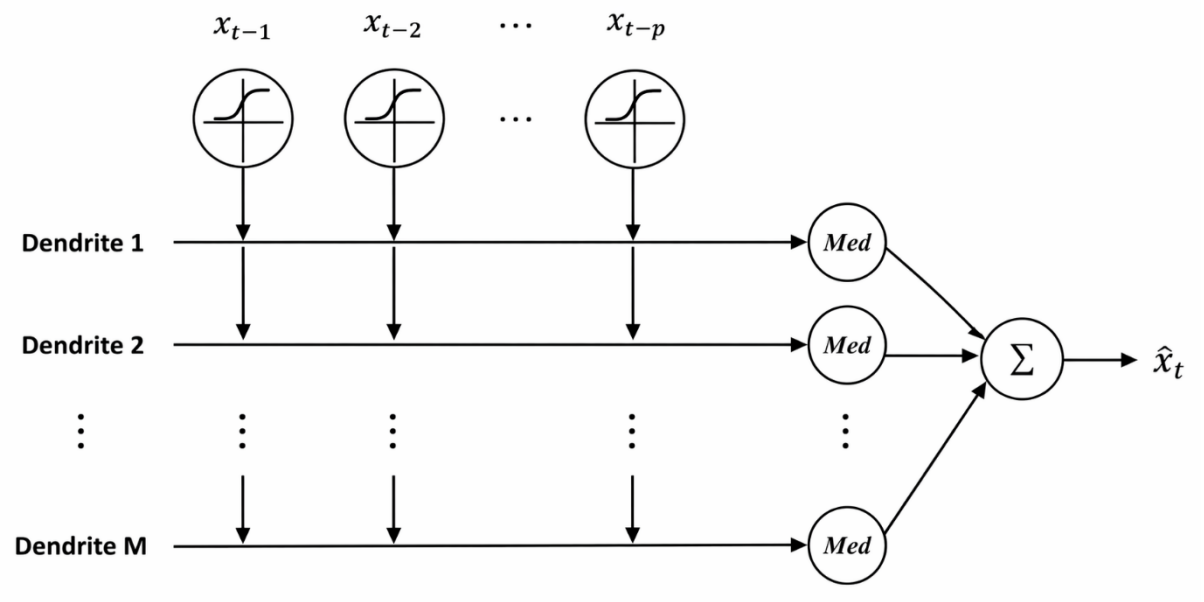

Figure 5.1: The architecture of DNM-ANN *(Bas, Egrioglu, & Cansu, 2024)*.

5.2.2.1 Capsule Networks

One particularly significant arrival within the contemporary deep learning landscape has been the Capsule Network — frequently abbreviated to CapsNet — which constitutes a conscious break from the long-standing assumptions that have shaped conventional convolutional model design. The framework was first formulated by Geoffrey Hinton and colleagues in his research circle, and its central architectural innovation is a building block known as a "capsule." Instead of representing each processing unit as a single scalar-valued response, a capsule bundles a compact group of neurons whose collective activation pattern simultaneously signals that a particular perceptual element is present within a layered visual scene and conveys, in vector form, the geometric and hierarchical positioning of that element with respect to its neighbours (Patrick, Adekoya, Mighty, & Edward, 2022). The notion is illustrated graphically in Figure 5.2. Standard convolutional architectures are well-documented to degrade performance when the visual input is rotated, resized, or otherwise repositioned. The capsule formulation, by contrast, holds onto the part-whole organisation of the scene throughout processing, and this property carries over into measurably better outcomes on tasks that depend on grasping how the constituent visual elements are spatially configured.

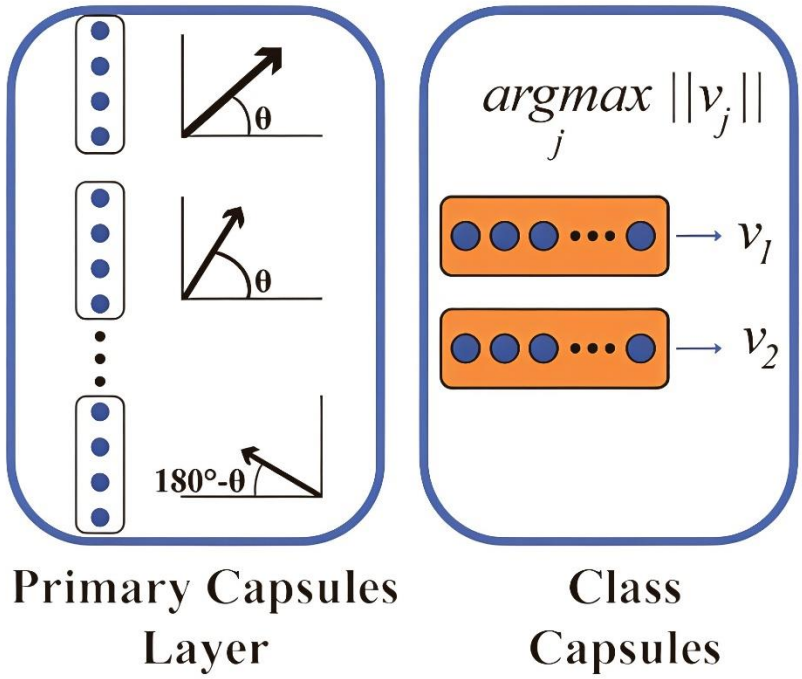

Figure 5.2: Example Capsule NN Layer *(Ramirez-Quintana, Macias-Macias, Ramirez-Alonso, Chacon-Murguia, & Corral-Martinez, 2023)*.

5.2.2.2 Fuzzy classification

Fuzzy/Soft/Probabilistic classification is used when predicting multiple classes, not just one class, but the percentage of each class. The model outputs a probability distribution over all possible classes instead of assigning a single definitive label. For instance, when classifying emotions, the network might output a distribution such as 80% anger, 10% disgust, etc., indicating the model's confidence in each emotion category. The approach enables more

nuanced predictions, capturing the class's uncertainty or overlap (Sung, Kim, & Kang, 2023).

5.2.2.3 DenCaps

The fusion of Dendritic Neural Layers with Capsule Network architecture represents a novel approach to deep learning, particularly for Speech Emotion Recognition (SER). A dendritic layer, inspired by biological dendrites' complex structure and functionality, enables the network to perform sophisticated, non-linear integrations of synaptic inputs. The approach allows for processing nuanced features within speech, which indicate emotional states. When such a dendritic layer feeds into a Capsule Network, the system benefits from the dendritic layer's adept feature extraction capabilities. With their dynamic routing and ability to preserve hierarchical relationships, Capsule Networks can further enhance the model's sensitivity to spatial and temporal information within speech data. The synergy can enhance emotion recognition by capturing subtle inflexions, intonation, and dynamics in a speech often lost in traditional CNNs. In the context of SER, the combined strengths of dendritic layers and Capsule Networks offer several advantages: Enhanced Feature Extraction: Dendritic layers can pre-process and refine input data, emphasising salient emotional features that improve the Capsule Network's learning efficiency. Robustness to Variability: Capsule Networks can maintain performance despite variations in speech patterns, such as speed and accent, making the system more robust and widely applicable. Contextual Sensitivity: Integrating dendritic computations can provide a deeper understanding of the context, a critical factor in accurately interpreting emotions in speech. Improved Generalisation: The combined model could better generalise new, unseen data, thanks to the more complex and biologically-inspired processing of dendritic layers. Computational Efficiency: While Capsule Networks are typically more computationally intensive than other architectures, the preliminary processing by dendritic layers can reduce redundancy and enhance efficiency. Reduced Data Requirements: By capturing more information from each input sample, the dendritic-capsule system may reduce the need for vast training data typically required for deep learning models. In SER, detecting and classifying emotions from audio signals are highly complex tasks that can significantly benefit from models that can capture the intricate patterns in the data. The combination of dendritic neural layers and Capsule Networks holds promise for creating more sophisticated, accurate, and reliable SER systems that push the boundaries of artificial emotional intelligence.

5.2.3 Multilingual Speech Corpus

5.2.3.1 ASVP-ESD

ASVP-ESD (The Audio, Speech, and Vision Processing Lab Emotional Sound database) (Dejoli, He, & Xie, 2021) is a comprehensive database designed for the analysis of speech and non-speech emotional sounds, encompassing over 12,000 audio files that capture a wide range of emotional states from various sources without language restriction. It stands out for its realistic, non-scripted content, including diverse emotions and intensities, making it a valuable resource for emotion recognition research. A pool of roughly 13,285 audio recordings was assembled by drawing material from cinematic releases, broadcast television programmes, and content hosted on YouTube, encompassing both vocal and non-vocal segments. The collection

captures twelve distinct naturally occurring affective states — namely boredom, neutral, happiness, sadness, anger, fear, surprise, disgust, excitement, pleasure, pain, and disappointment — each rendered at two separate intensity gradations. The dataset includes multiple languages, including Chinese, English, French, Russian, and others. 11 of the 12 emotions fit into Plutchik's Wheel of Emotions. Illustrated in Figure 5.3. The emotions that are an exact match are marked in green. The ones that are not precisely matched are marked with a red outline.

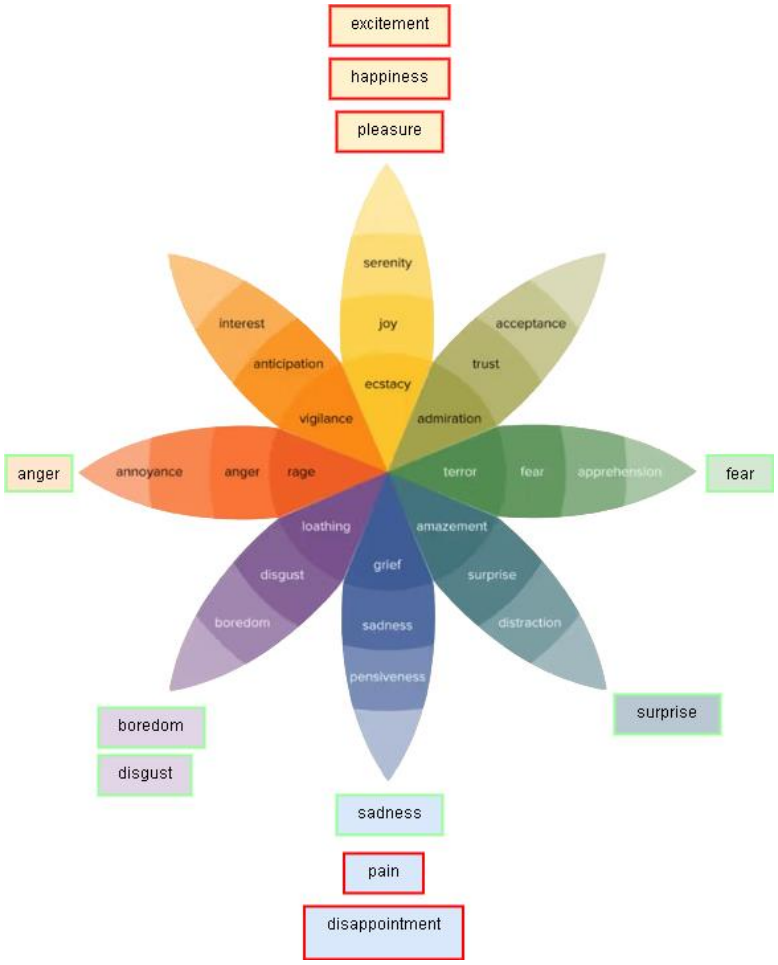

Figure 5.3: ASVP-ESD Plutchik Mapping.

Pain: Plutchik's Wheel of Emotions is a framework for understanding human emotions and their relationships, including contrasts and combinations. The scheme arranges affective states into four opposing pairs drawn from a set of eight foundational emotions: joy set against sadness, anger set against fear, trust set against disgust, and surprise set against anticipation. These primary emotions can be mixed or intensified to create a broad spectrum of human emotional experiences. Pain, as an emotional and physical experience, is not explicitly listed in Plutchik's Wheel of Emotions. However, it can be associated with several emotions in the model depending on the context and nature of the pain. For example, Physical pain might closely relate to the primary emotion of fear, as it can be a response to bodily harm or a threat to one's physical well-being. The experience of pain can also lead to sadness due to loss, suffering, or helplessness. Pain might be associated with disgust in some contexts, mainly if it

results from something repulsive or disturbing. Emotions are complex and can be experienced and expressed in multifaceted ways. So, while pain is not a category within Plutchik's model, the emotions that arise from pain can be mapped onto the wheel. Understanding where these emotional responses fit within Plutchik's framework can help recognise the complexity of emotional reactions to pain and the interconnectedness of different emotional states (Lumley, et al., 2011).

Disappointment: In Plutchik's Wheel of Emotions, disappointment is understood as a blend of surprise and sadness. The conceptual framework posits that emotions can be combined to form more complex feelings, with disappointment arising from the juxtaposition of unexpected outcomes and a sense of loss or dissatisfaction (van Dijk & Zeelenberg, 2002).

Excitement: In Plutchik's Wheel of Emotions, excitement is often associated with higher-intensity feelings derived from primary emotions such as joy and anticipation. While Plutchik's model outlines eight primary emotions—anger, anticipation, joy, trust, fear, surprise, sadness, and disgust—it also considers how these can combine or intensify to form more complex emotions. Excitement, therefore, can be understood as a complex emotion stemming from the anticipation of something positively anticipated or the intense experience of joy (Johnson, 2020).

Happiness: In Plutchik's Wheel of Emotions, happiness aligns with the emotion of joy, one of the eight primary emotions identified by Plutchik. Joy, along with trust, fear, surprise, sadness, anticipation, anger, and disgust, forms the foundation of the emotional model, which illustrates the complex interrelationships and intensities of human emotions (Fredrickson, 1998).

Pleasure: In Plutchik's Wheel of Emotions, "pleasure" is not explicitly labelled; however, emotions related to pleasure, such as joy or trust, could be seen as encompassing the feeling of pleasure. Plutchik's model includes eight primary emotions—joy, trust, fear, surprise, sadness, anticipation, anger, and disgust—each having a polar opposite. Pleasure would likely fall under "joy," which is opposite to "sadness" in the model, emphasising positive, pleasurable feelings of happiness and contentment (Berridge & Kringelbach., 2015).

5.2.3.2 CaFÉ

The Canadian French Emotional Speech Dataset (CaFÉ) comprises 936 utterances, encapsulating six distinct sentences pronounced by 12 male and female actors. The dataset, designed to address the absence of emotional speech data in Canadian French, categorises utterances into six basic emotions plus one neutral emotion, each expressed with two different intensity levels: mild and robust.

A Canadian French Emotional Speech Dataset (CaFÉ) (Gournay, Lahaie, & Lefebvre, 2018). 6 different sentences by 12 speakers (6 females + 6 males). 7 emotions: happy, sad, angry, fearful, surprised, disgusted and neutral. Each emotion is acted in 2 different intensities.

6 of the seven emotions in the corpus are used. Also, there is an exact 1 to 1 mapping to Plutchik's emotions. Illustrated in Figure 5.4.

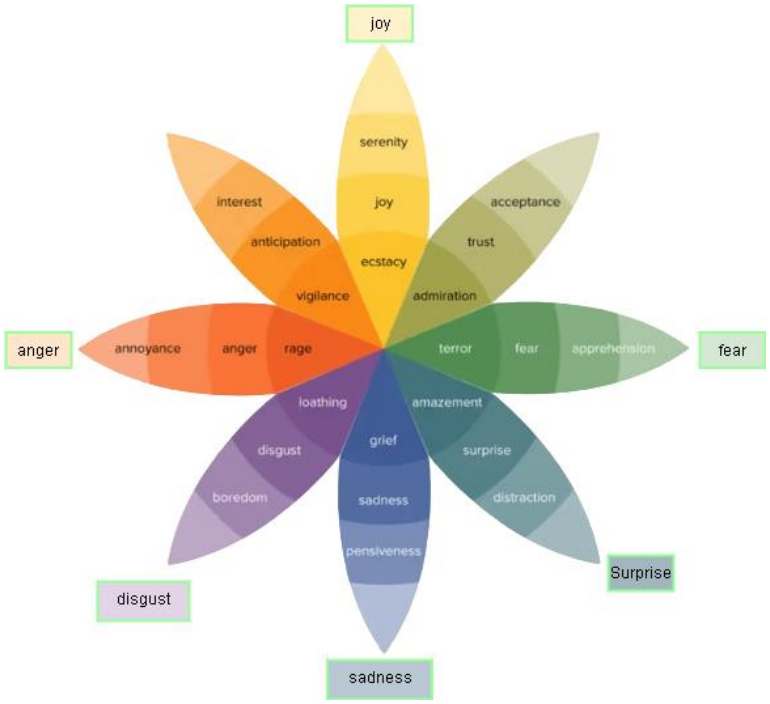

Figure 5.4: CaFÉ Plutchik Mapping *(Gournay, Lahaie, & Lefebvre, 2018)*.

5.2.3.3 EmodDB

EmodDB (Berlin Database of Emotional Speech) (Burkhardt, Paeschke, Rolfes, Sendlmeier, & Weiss, 2005). The Berlin Database of Emotional Speech (EMO-DB) is actually a well-established corpus that contains emotional speech recordings in German, not Italian. It comprises recordings of ten actors conveying seven different emotions (anger, boredom, disgust, anxiety/fear, happiness, sadness, and neutral) through ten sentences, making it a fundamental resource for research in speech emotion recognition and related fields.

Six of the seven emotions in the corpus are used. Also, there is a one-to-one mapping onto Plutchik's emotions. Figure 5.5 illustrates this.

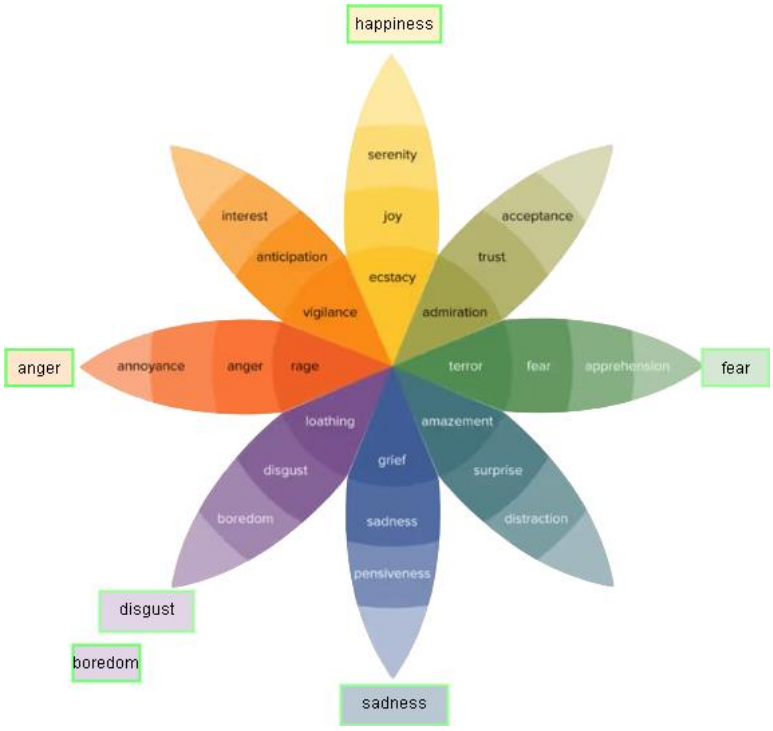

Figure 5.5: EMODB Plutchik Mapping.

5.2.3.4 AESDD

The Acted Emotional Speech Dynamic Database (AESDD) (Vryzas, Kotsakis, Liatsou, Dimoulas, & Kalliris, 2018). The Acted Emotional Speech Dynamic Database (AESDD) is a publicly available dataset that provides a foundation for Speech Emotion Recognition (SER) research, mainly focusing on the Greek language. The dataset was developed to fill the gap of a high-quality SER database in Greek and includes utterances of acted emotional speech, encapsulating a range of emotions such as anger, disgust, fear, happiness, and sadness. The creation involved collaboration with professional actors and was motivated by research on an emotion-triggered lighting framework for theatrical performances. AESDD is designed to be dynamic, with the intention of ongoing expansion through contributions from actors and performers. It is structured into five folders corresponding to five emotion classes and file names, including the emotion's initial, utterance, and speaker numbers. It has been subject to subjective evaluation experiments, which estimated human accuracy in recognising intended emotions in utterances at around 74%. The database has been utilised in machine learning experiments, testing various classifiers like Multi-Class Logistic Regression, Multi-Layer Perceptrons, Support Vector Machines with polynomial kernels, Logistic Model Trees, and Subspace Ensemble Learning classifiers. These experiments were conducted using tools such as the WEKA environment and MATLAB's Classification Learner Toolbox to understand and improve the classification of emotional states from speech data.

All five emotions in the corpus are used, with an exact one-to-one mapping to Plutchik's emotion categories. Figure 5.6 illustrates the mapping.

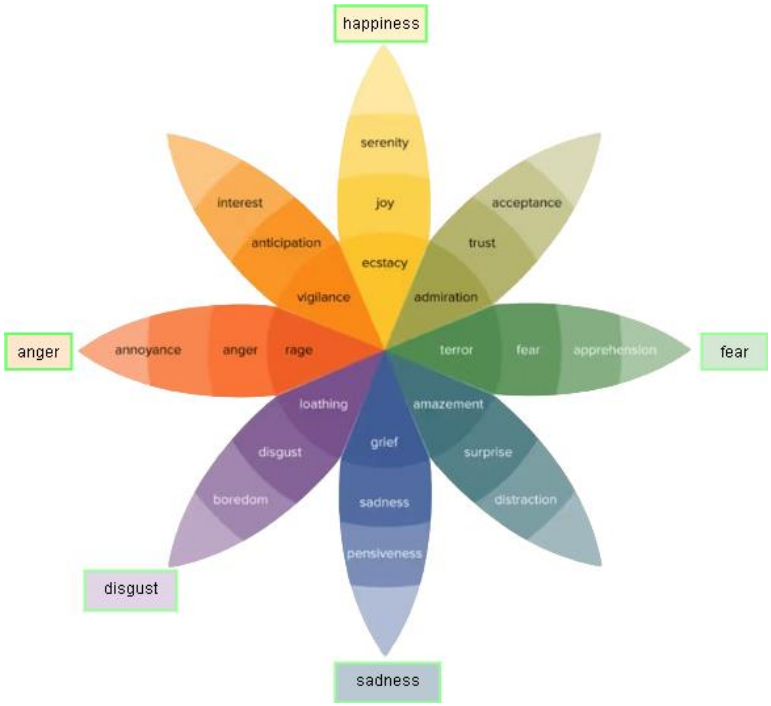

Figure 5.6: AESDD Plutchik Mapping.

5.2.3.5 AFRIKAANS

The Afrikaans corpus is the one that was mentioned in the previous chapter (Norval & Wang, 2022).

5.2.3.6 EMOV-DB

The Emov-DB (The emotional voices database) (Adigwe, Tits, Haddad, Ostadabbas, & Dutoit, 2018) The English dataset is designed for Speech Emotion Recognition (SER) and contains recordings from four speakers, two males and two females. It showcases a variety of emotional styles, including neutral, sleepiness, anger, disgust, and amusement. The extensive dataset, totalling 5.88 GB, is a valuable resource for developing and testing SER systems.

Two of the four emotions in the corpus are used. Also, two emotions are exact 1-to-1 matches with Plutchik's emotions. 2 emotions are not exact matches. Illustrated in Figure 5.7.

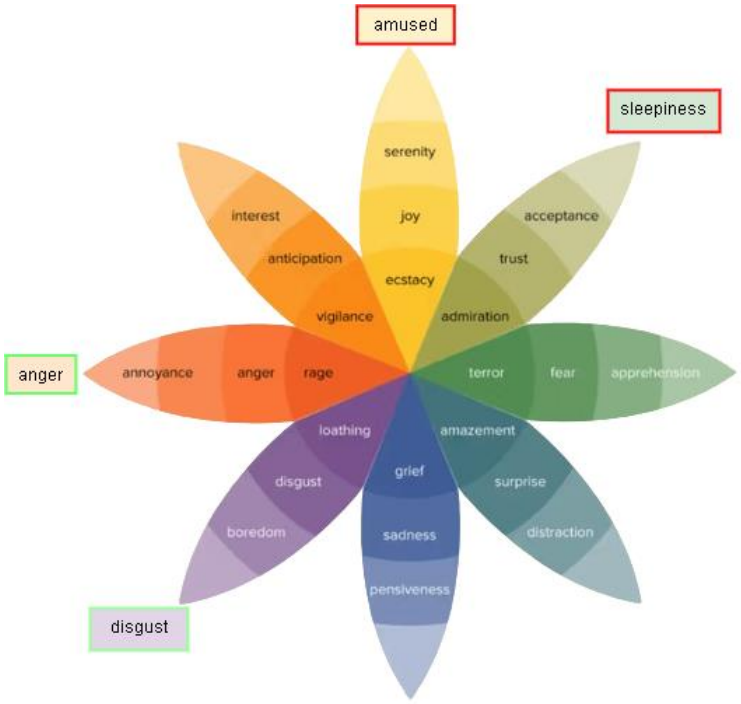

Figure 5.7: Emov-DB Plutchik Mapping.

Sleepiness does not directly correspond to the eight primary emotions (joy, trust, fear, surprise, sadness, anticipation, anger, and disgust) on Plutchik's Wheel of Emotions. Plutchik's model focuses on more active emotions related to evolutionary survival strategies. Sleepiness, being more of a physiological state than an emotion per se, doesn't fit neatly into Plutchik's emotional categories. However, it could be indirectly related to feelings of relaxation or calmness, which might be associated with the lower intensity of emotions like trust and anticipation in Plutchik's model. It's important to note that physiological states like sleepiness can influence emotional experiences, but they are not classified as core emotions in Plutchik's theory.

Plutchik's Wheel of Emotions doesn't explicitly include "amused" as one of the primary or secondary emotions. However, considering the wheel's structure and the emotions it encompasses, "amused" could likely fall under or be closely related to "joy," one of the eight primary emotions identified by Plutchik. The wheel illustrates the relationships and intensities of primary emotional responses, where "amused" might be seen as a less intense form of joy or a combination of pleasure with other emotions.

5.2.3.7 OGVC

The OGVC corpus, developed by Yoshiko Arimoto and Hiromi Kawatsu from Tokyo University of Technology, is a specialised database designed to study and develop voice conversion technologies (Arimoto, Kawatsu, Ohno, & Iida, 2012). The corpus consists of meticulously recorded voice data from multiple speakers, providing a rich resource for researchers focusing on voice transformation and synthesis. The data includes various speech samples capturing different emotions, intonations, and linguistic content, enabling detailed analysis and application in voice conversion systems. The OGVC corpus is particularly noted

for its diversity in speaker demographics, which aids in creating more versatile and adaptive voice conversion models. The collection serves as a resource for investigators developing and evaluating algorithmic methods to modify a speaker's vocal traits without compromising the naturalness or comprehibility of the resulting speech. As voice conversion technology continues to evolve, the OGVC corpus remains a valuable asset for advancing research and developing applications in communication aids, entertainment, and personalised voice interfaces. The concept is illustrated in Figure 5.8.

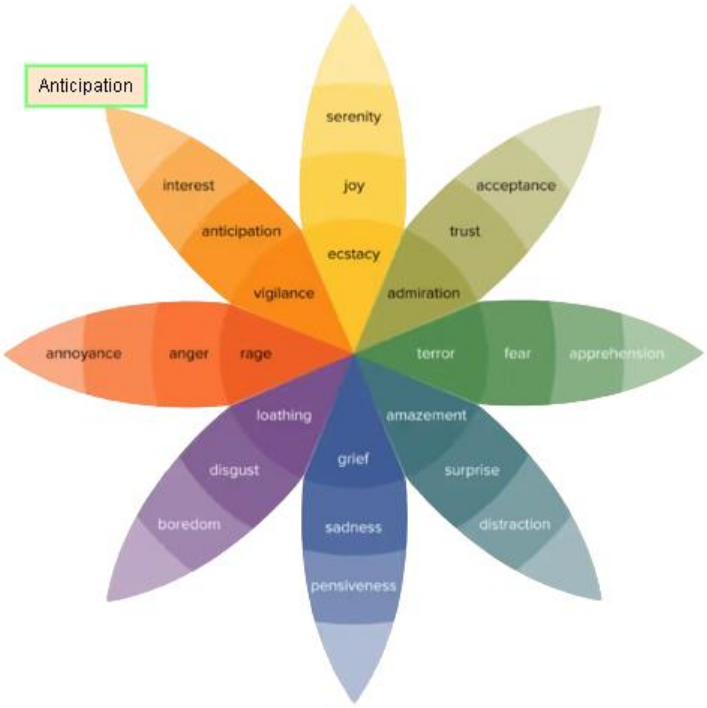

Figure 5.8: OGVC Plutchik Mapping.

This library is primarily used to supplement the Anticipation count of samples, which was lacking.

5.2.3.1 Data Augmentation

Audio processing within the deep learning workflow involves reshaping and adjusting acoustic signals to make them suitable for model fitting and subsequent analysis. The associated steps span tasks such as feature extraction, where informative descriptors (for instance, Mel-frequency cepstral coefficients or spectrograms) are distilled from raw waveforms to serve as model inputs, alongside preparatory operations like normalisation that keep amplitude levels uniform across the dataset. This preparatory work is critical for improving model performance and enabling reliable pattern recognition in sound data, with downstream applications spanning speech transcription, music categorisation, and environmental sound identification.

Data augmentation plays a central role in deep learning models, especially in the audio domain. It artificially enlarges the training pool by synthesising altered variants of the existing material. Common manipulations include pitch shifting, time stretching, the injection of controlled noise, and adjustments to playback speed. Such alterations sharpen the models' capacity to generalise to unseen examples by exposing them to a broader spread of conditions during fitting. In an audio context, where elements such as recording environment, speaker traits, and ambient noise can heavily sway model behaviour, augmentation is particularly important. By widening the diversity of the training material, the resulting models become more resilient and considerably less prone to overfitting.

.

5.2.3.1.1 Pitch Shifting

Pitch shifting is a significant technique in audio processing, allowing for the alteration of an audio signal's pitch without affecting its tempo. The effect has a wide range of applications, from creating harmonies and tuning vocal tracks to changing the key of a music piece. Pitch shifting is achieved by modifying the frequency components of the audio signal, and it's a common technique in music production and sound design.

5.2.3.1.2 Time Stretching

Time stretching is changing the speed or duration of an audio signal without affecting its pitch. The approach can be useful for fitting tracks into a specific length, creating slow-motion effects, or synchronising audio clips in post-production. Time stretching works by spreading out or compressing the waveform temporally, often using sophisticated algorithms to maintain sound quality.

5.2.3.1.3 Adding Noise

Adding noise involves intentionally introducing a random signal to an audio recording. The technique can enhance the realism of a synthesised sound, mask unwanted artefacts, or be used as an artistic effect. Depending on the application, the noise can vary in type, such as white noise, pink noise, or custom-designed noise profiles.

5.2.3.1.4 Reversing Audio

Reversing audio flips the audio signal in time, making it play backwards. The effect can create unique sounds and textures not found in forward-playing audio. It's used for artistic purposes, special effects in films, or to uncover hidden messages in music. Reversing can reveal interesting sound characteristics not apparent in the original recording.

5.2.3.1.5 Harmonic and Percussive Components Separation

Separating harmonic and percussive components of an audio signal involves dividing the audio into melodic and rhythm elements. The separation is proper for audio analysis, mixing, and enhancing music tracks, allowing engineers and producers to process the melody and rhythm independently. Techniques for such separation analyse the audio's temporal and spectral characteristics to distinguish between sustained tones and transient attacks.

5.3 Materials and Methods

It should be noted that the approach adopted in the research involves multilingual training on a combined dataset rather than explicit cross-lingual transfer learning. Transfer learning typically involves training a model on one language and subsequently adapting it to another. In contrast, the models in the research are trained simultaneously on multiple languages to learn language-agnostic representations. The objective is therefore to achieve cross-lingual generalisation rather than transfer learning in the strict sense.

This chapter builds upon the dendritic layer approach developed and validated in Chapter 4. The architectural insights gained from the DCLSTM model—particularly the effectiveness of dendritic processing for capturing complex temporal and hierarchical dependencies—are extended here through the proposed DenCaps architecture, which integrates dendritic layers with Capsule Networks to address the increased complexity and variability of multilingual emotional speech.

5.3.1 Justification For Current Research

In SER, remarkable progress has been achieved in formulating sophisticated models that adeptly differentiate between a spectrum of emotional states expressed through verbal communication. Despite these advancements, ongoing challenges and shortcomings necessitate continuous academic exploration. A principal challenge in the domain of speech emotion recognition is the optimisation of feature selection — extracting pertinent features that accurately convey the emotional undertones embedded within audio signals remains a complex endeavour. Moreover, the issue of high dimensionality is prevalent; speech signals are often represented in an extensive dimensional space, which renders the computational processing and modelling of these signals particularly resource-intensive. Additionally, context awareness is frequently overlooked — emotional expressions are intrinsically linked to specific contexts, yet numerous models fail to account for the contextual dimension of variability. In response to the difficulties outlined above, an attractive avenue forward is to bring together dendrite-inspired processing layers, capsule-based architectures, and ensemble-style modelling — a combination that offers genuine potential to enhance both the resilience and the interpretive transparency of SER pipelines. In particular, fusing Capsule Networks (CapsNets) with

Dendritic Neural Networks is particularly compelling when the target application is to infer affective state from acoustic input. These advanced network models exhibit distinct properties potentially advantageous for discerning complex emotional cues within speech, given their proficiency in capturing hierarchical data structures and contextual dependencies. The interdisciplinary approach, blending innovations from neural computation and advanced machine learning, signifies a strategic direction towards more refined and contextually aware emotion recognition technologies.

5.3.2 Proposed System

The scholarly review indicates that traditional deep learning architectures such as CNNs and LSTMs exhibit limitations in audio emotion detection. Specifically, these limitations pertain to the processing of audio signals in uniform, fixed-sized segments, which may not sufficiently encapsulate the long-term dependencies or the nuanced temporal variations indicative of emotional states. Emotions manifested in audio are intrinsically linked to contextual factors, which present a challenge for detection within the confines of these fixed-sized windows.
Furthermore, the quality of audio data is a significant factor impacting the efficacy of emotion detection models. In practical scenarios, audio data is frequently compromised by distortions, ambient noise, and inconsistencies in recording conditions, all of which can obfuscate the emotional cues within the signal.

This thesis introduces a meticulous methodology for training two distinct neural network architectures. The initial model is a Convolutional Long-Short-Term Memory (CLSTM) network established to provide baseline results. Subsequently, a novel hybrid network incorporating both Dendritic and Capsule network mechanisms (DenCaps) is introduced. The DenCaps network is trained with meticulously adjusted hyperparameters to optimise accuracy. In terms of training, early stopping is implemented as well, to keep the validation loss to a minimum.

Finally, the results are compared using accuracy, precision, F1 score, recall and loss metrics. Subsequently, fuzzy detection is done on random samples and compared to the Plutchik wheel of emotions. The concept is illustrated in Figure 5.9.

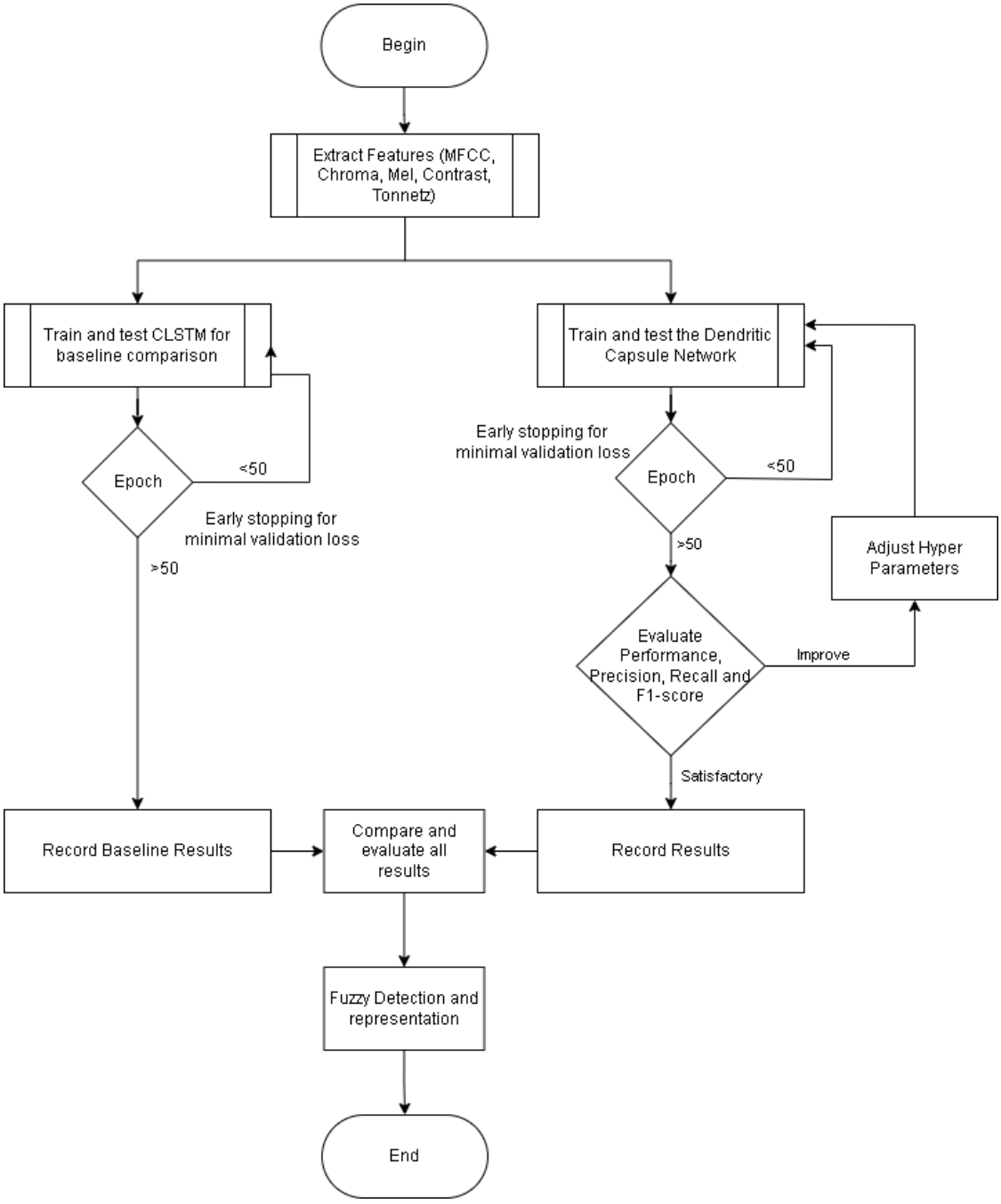

Figure 5.9: Training and evaluation Flowchart.

5.3.3 Dataset and Training

The training data comprises the combined speech corpus mentioned above and augmented data. Utilisation of Google Colab with a high-speed V100 GPU environment was employed to meet the substantial memory and processing demands of the training procedure. For convenience and rapid retrieval, both the previously fitted model checkpoints and the underlying training material were uploaded to Google Drive. Because building the requisite implementations for the Dendritic Layer module, the Capsule Neural Network component, and the Ensemble combination scheme required a substantial amount of code, several reference implementations sourced from public GitHub projects were pulled in and adapted as starting points during the experimental phase. The primary libraries employed included TensorFlow Keras, NumPy, and scikit-learn. The sample breakdown is illustrated in Figure 5.10. The total number of audio samples across the combined corpus is 23,508. Joy (4,406) and Sadness (4,314) are the most represented emotions, followed by Disgust (3,727) and Anger (3,212). Anticipation has a significant count (3,206) primarily due to OGVC and the Augmented corpus. Trust (1,870), Fear (1,612), and Surprise (1,161) have lower but still substantial representations. The distribution highlights the diversity and richness of the datasets, providing a robust foundation for emotion recognition research. The significant variations in sample sizes and emotion representation across the combined corpus offer opportunities for comprehensive analysis and model training across a wide range of emotional expressions.

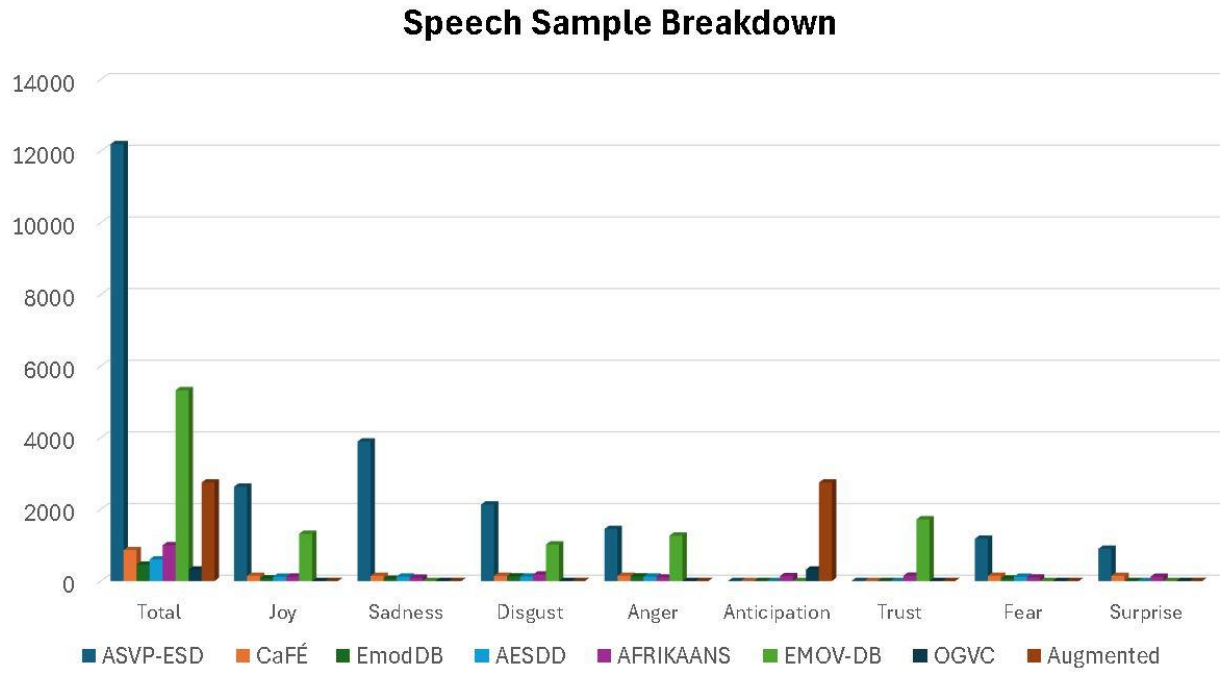

Figure 5.10: Speech sample breakdown chart (*Generated using Python*)

5.4 Results

5.4.1 Test Data

For the testing phase, audio examples were drawn at random from the larger reservoir of 23,508 available recordings, an arrangement intended to provide a credible measure of how well each model carries over to previously unseen material. Roughly 4,700 of these clips were carved off and dedicated to the validation and held-out test partitions. Performance on this material was then quantified using four standard indicators: Accuracy, Precision, Recall, and F1 Score.

5.4.2 Class Imbalance

The multilingual dataset presented in Figure 5.10 exhibits a clear class imbalance across the eight emotion categories (Anger, Anticipation, Disgust, Fear, Joy, Sadness, Surprise, and Trust). The class imbalance results in certain emotions (e.g., Joy and Sadness) being overrepresented, while others (e.g., Fear and Surprise) are significantly underrepresented. Such skewed distributions are common in speech emotion recognition (SER) datasets but introduce a systematic bias during model training. Class imbalance affects learning by biasing the optimisation process toward majority classes, leading to inflated overall accuracy while degrading performance on minority classes. The problem is particularly pronounced in multilingual SER, where emotional expression already varies across languages, compounding the difficulty of learning robust representations. To address this, class balancing was incorporated directly into the training process using weighted categorical loss. Raw class weights were computed inversely proportional to class frequencies and subsequently softened using a scaling factor $\alpha=0.5$ to prevent instability caused by extreme weighting. The softened weights ranged from 0.9004 (Joy) to 1.4518 (Surprise), ensuring that minority classes contributed more significantly to the loss function without dominating training. In addition to weighted loss, evaluation metrics were extended beyond overall accuracy to include macro F1-score and balanced accuracy. These metrics provide a more reliable assessment under class imbalance by assigning equal importance to each class. Despite these mitigation strategies, residual imbalance effects remain evident in the results, particularly for minority classes such as Fear and Surprise, which consistently exhibit lower recall. Therefore, while class weighting improves robustness, class imbalance remains a limiting factor in multilingual SER performance and is acknowledged as an inherent dataset constraint.

5.4.3 CNN

The CNN model was evaluated under the class-balanced training regime described in Section 5.4.2. Incorporating class weights led to a more equitable contribution from minority classes during optimisation, reducing bias toward dominant emotions. Across ten experimental runs, the CNN achieved moderate performance, with improvements observed in macro-level metrics compared to unbalanced training. However, discrepancies between overall accuracy and balanced accuracy indicate that class imbalance effects persist. Specifically, the majority classes, such as Anticipation and Trust, achieved higher recall, while the minority classes—particularly Fear and Surprise—remained difficult to classify.

The confusion matrix in Figure 5.11 further illustrates the class imbalance, with stronger diagonal dominance for majority classes and weaker representation for minority categories. The findings confirm that, while class weighting mitigates bias, the CNN architecture remains sensitive to class imbalance in multilingual settings.

Overall, the CNN serves as a baseline model, demonstrating that conventional convolutional architectures can partially adapt to imbalance but lack the representational capacity to fully resolve class disparity in complex multilingual emotional data. tions.

Table 5.14: Multilingual Data - CNN

Run	**Accuracy**	**Macro F1**	**Weighted F1**	**Balanced Acc**	**Loss**
1	0.5168	0.4625	0.4898	0.4805	1.5100
2	0.5151	0.4292	0.4793	0.4572	1.5459
3	0.5225	0.4808	0.5064	0.4851	1.4039
4	0.5128	0.4142	0.4679	0.4478	1.4497
5	0.5106	0.4417	0.4756	0.4572	1.5161
6	0.5296	0.4814	0.5143	0.4827	1.4636
7	0.5121	0.4266	0.4797	0.4484	1.5050
8	0.4662	0.3649	0.4082	0.4038	1.5696
9	0.5332	0.4758	0.5053	0.4917	1.3912
10	0.5379	0.4859	0.5142	0.5004	1.4713

Table 5.15: CNN Summary Table (Mean ± Std)

Metric	**Mean**	**Std Dev**
Accuracy	51.57%	± 1.88%
Macro F1	0.4463	± 0.0367
Weighted F1	0.4841	± 0.0299
Balanced Accuracy	0.4655	± 0.0271
Loss	1.4826	± 0.0548
Best Val Accuracy	51.57%	± 1.88%

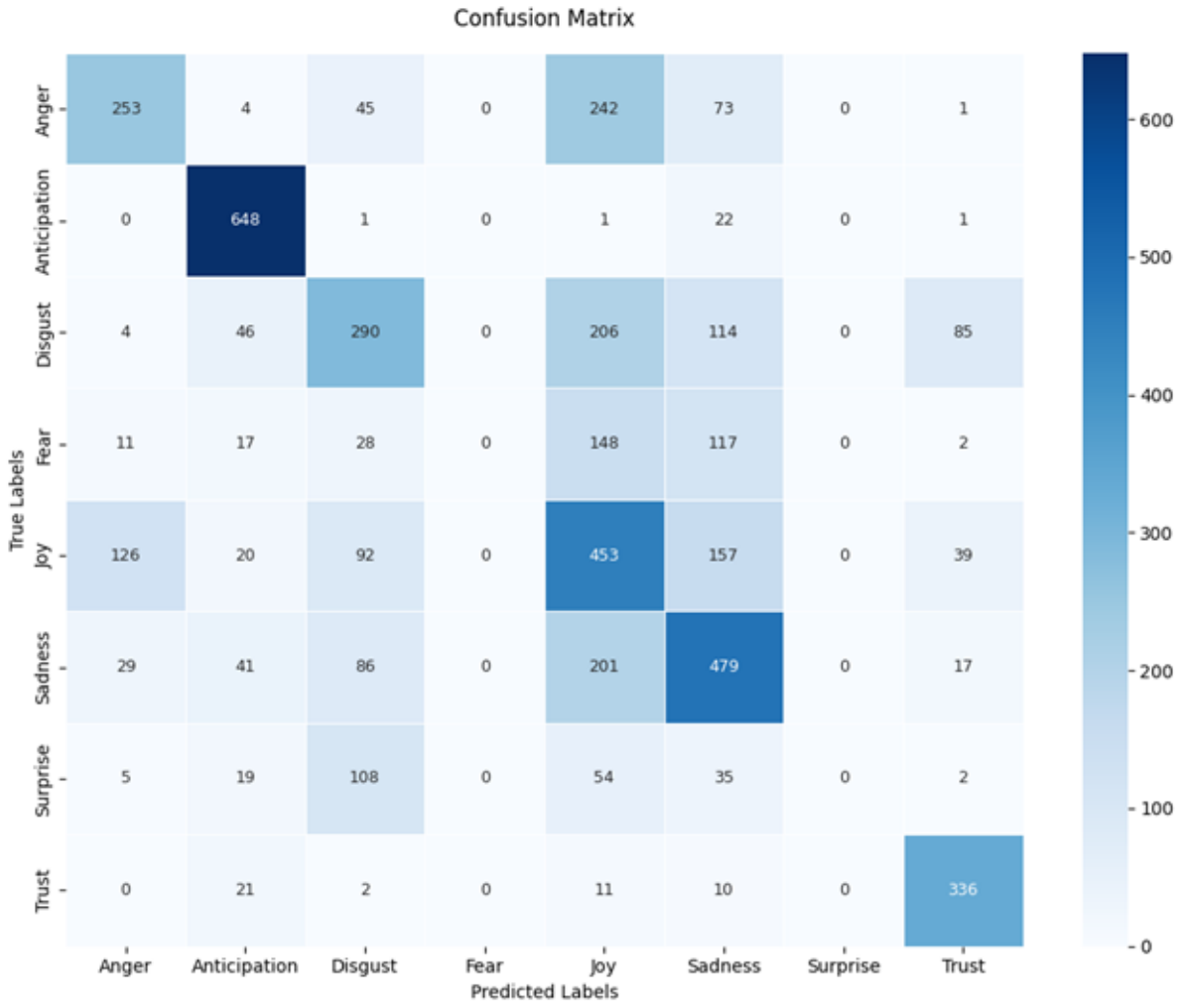

Figure 5.11: CNN Confusion Matrix.

5.4.4 CLSTM

The CLSTM model was evaluated using the same class-balanced training approach as described in Section 5.4.2. Across ten runs, the model achieved a mean accuracy of 57.84% ± 1.29%, a macro F1-score of 0.5168 ± 0.0143, and a balanced accuracy of 0.5269 ± 0.0158. The confusion matrix Figure 5.12 shows improved performance on minority classes compared to the CNN, with Fear no longer completely unclassified. However, Surprise remains largely unrecognised, indicating that class imbalance is still not fully resolved. Overall, the CLSTM reduces bias toward dominant classes and provides more balanced predictions than CNN, but variability across runs and poor minority-class recall confirm that class imbalance remains a key limitation.

Table 5.16: Multilingual Data - CLSTM

Run	**Accuracy**	**Macro F1**	**Weighted F1**	**Balanced Acc**	**Loss**
1	0.5868	0.5280	0.5689	0.5402	1.1537
2	0.6036	0.5415	0.5824	0.5593	1.1158
3	0.5921	0.5350	0.5756	0.5416	1.1321
4	0.5659	0.5010	0.5430	0.5172	1.1850
5	0.5681	0.5123	0.5525	0.5177	1.1832
6	0.5630	0.5015	0.5456	0.5090	1.1939
7	0.5853	0.5201	0.5638	0.5292	1.1378
8	0.5825	0.5234	0.5625	0.5328	1.1496
9	0.5710	0.5056	0.5506	0.5138	1.1855
10	0.5657	0.4995	0.5445	0.5083	1.2016

Table 5.17: CLSTM Summary Table (Mean ± Std)

Metric	**Mean**	**Std Dev**
Accuracy	57.84%	1.29%
Macro F1	0.5168	0.0143
Weighted F1	0.5589	0.0131
Balanced Accuracy	0.5269	0.0158
Loss	1.1638	0.0281
Best Val Accuracy	57.84%	1.29%

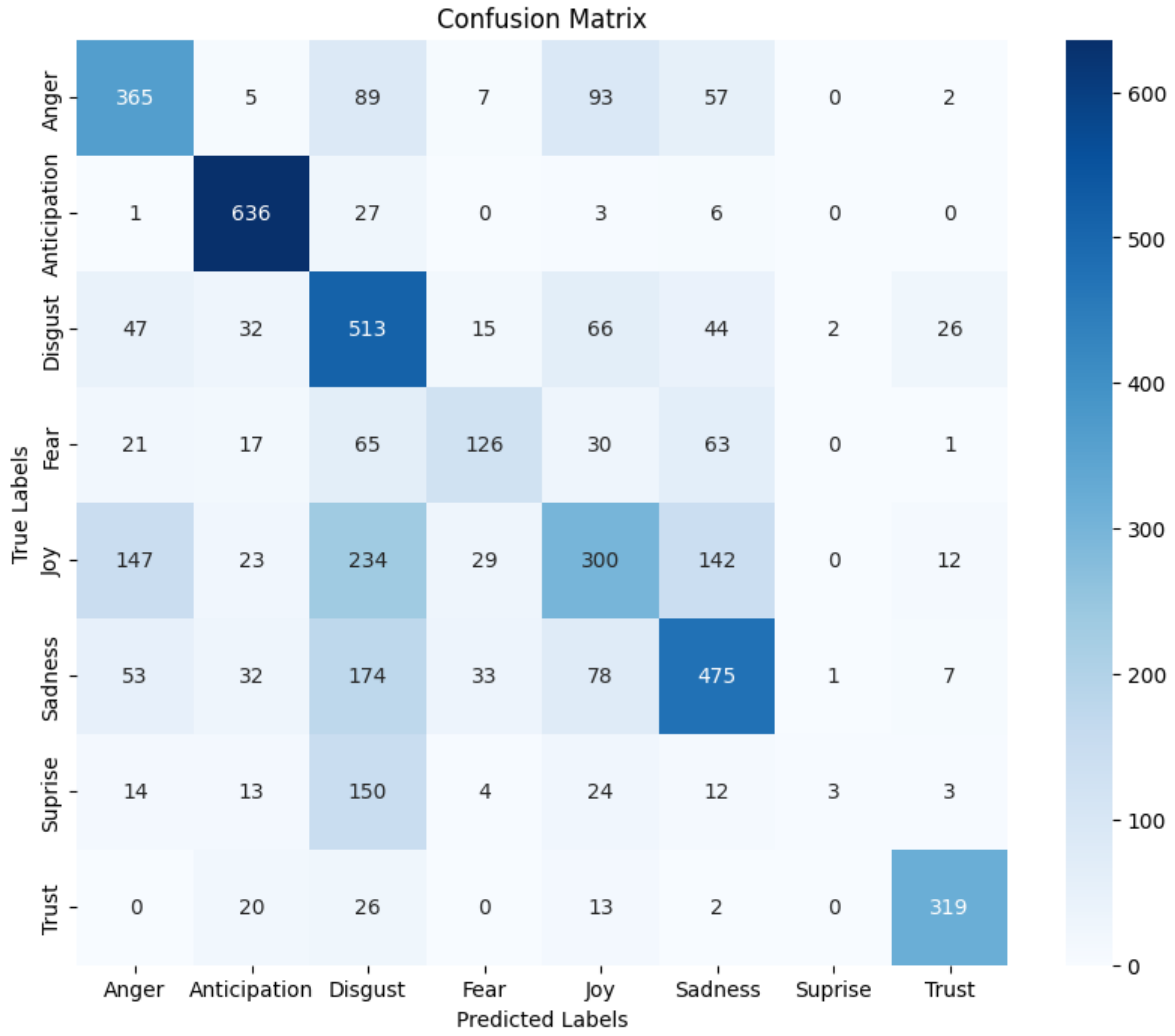

Figure 5.12: CLSTM Confusion Matrix.

5.4.5 DenCaps

The Dendritic Capsule Network (DenCaps) integrates dendritic-inspired nonlinear processing with Capsule Networks to better model hierarchical and context-dependent representations in speech signals. Unlike CNN and CLSTM architectures, DenCaps captures both local feature interactions and part–whole relationships through dynamic routing, which is particularly beneficial for emotionally rich and highly variable multilingual speech data.

To address the class imbalance identified in Section 5.4.2, the DenCaps model was trained with softened class weights ($\alpha = 0.5$), ensuring that minority classes such as Fear, Surprise, and Trust were not underrepresented during optimisation. The proposed approach resulted in more stable learning and improved generalisation across all emotion classes.

The DenCaps model consistently outperformed the CNN and CLSTM models across all evaluation metrics. anticipation is observed in the 95%+ percentile, indicating a clear and unambiguous emotional classification. Model accuracy is shown in Figure 5.14. The model loss is shown in Figure 5.15. Results are in the tables below.

Table 5.18: Multilingual Data - DenCaps

Run	**Accuracy**	**Macro F1**	**Weighted F1**	**Balanced Acc**	**Loss**
1	0.7103	0.6957	0.7092	0.7013	0.9649
2	0.7099	0.6909	0.7083	0.6944	0.9611
3	0.7116	0.6926	0.7100	0.6937	1.0068
4	0.7040	0.6865	0.7019	0.6918	0.9008
5	0.7084	0.6894	0.7071	0.6895	0.8626
6	0.7191	0.6978	0.7160	0.7000	1.0112
7	0.7074	0.6865	0.7047	0.6931	1.0161
8	0.7135	0.6951	0.7116	0.6983	1.0028
9	0.7169	0.6995	0.7152	0.7033	0.9539
10	0.7044	0.6832	0.7027	0.6859	0.9754

Table 5.19: DenCaps Summary Table (Mean ± Std)

Metric	**Mean**	**Std Dev**
Accuracy	0.7100	±0.0047
Macro F1	0.6917	±0.0053
Weighted F1	0.7081	±0.0049
Balanced Accuracy	0.6951	±0.0056
Loss	0.9656	±0.0532
Best Val Accuracy	0.7100	±0.0047

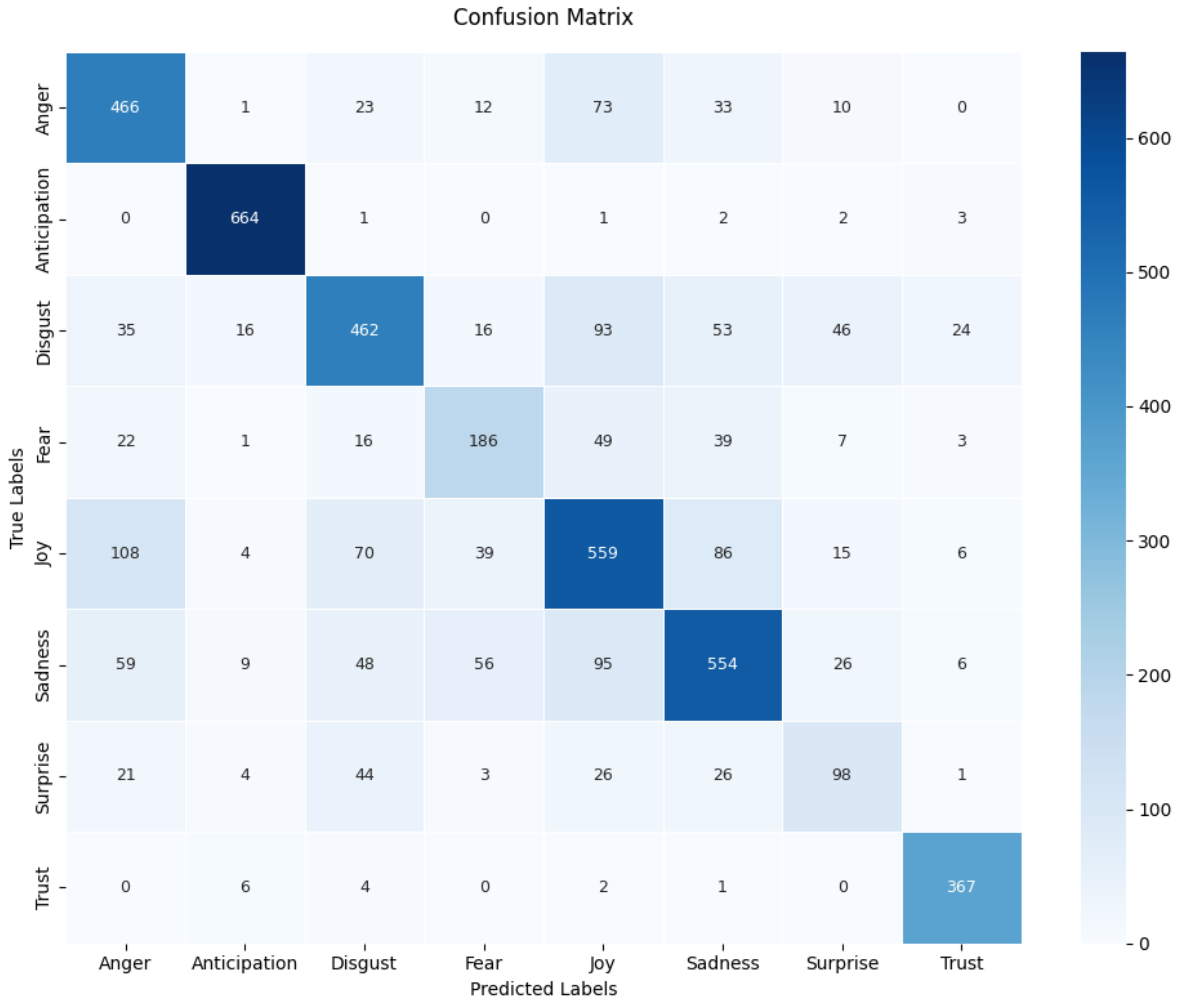

Figure 5.13: DenCaps Confusion Matrix.

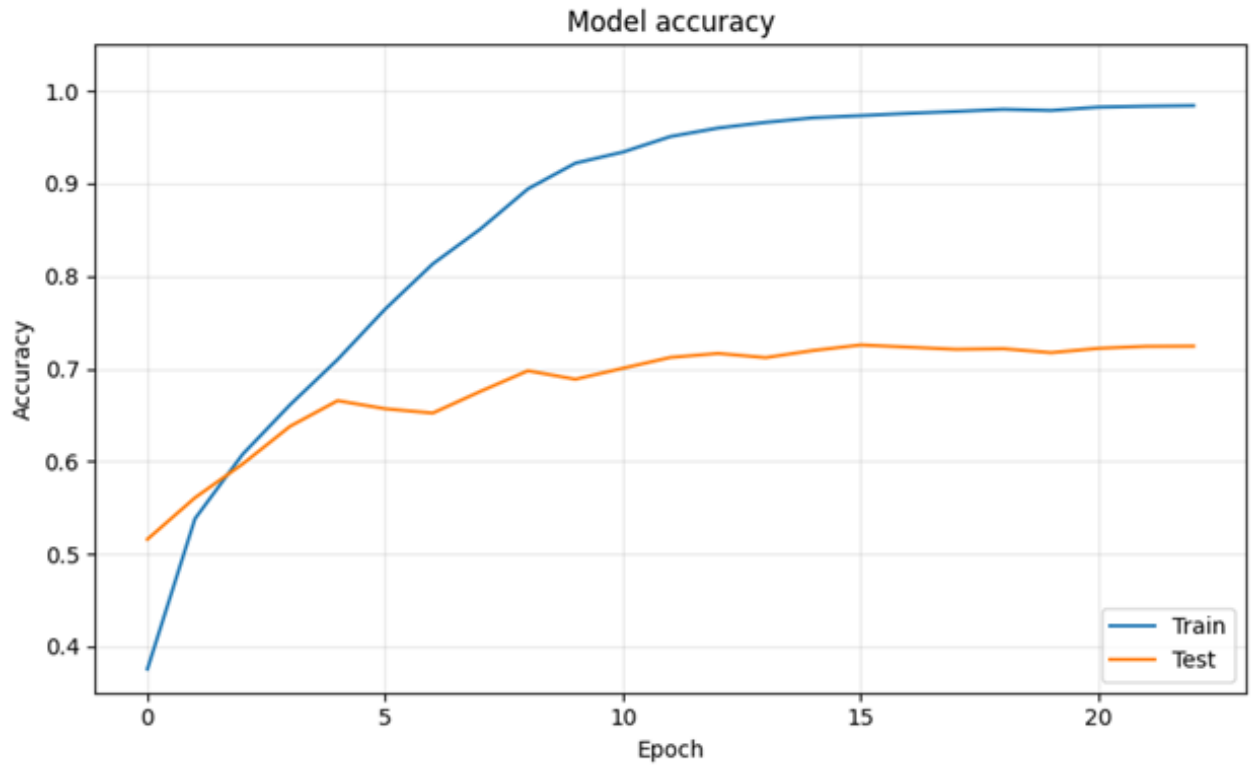

Figure 5.14: Model accuracy.

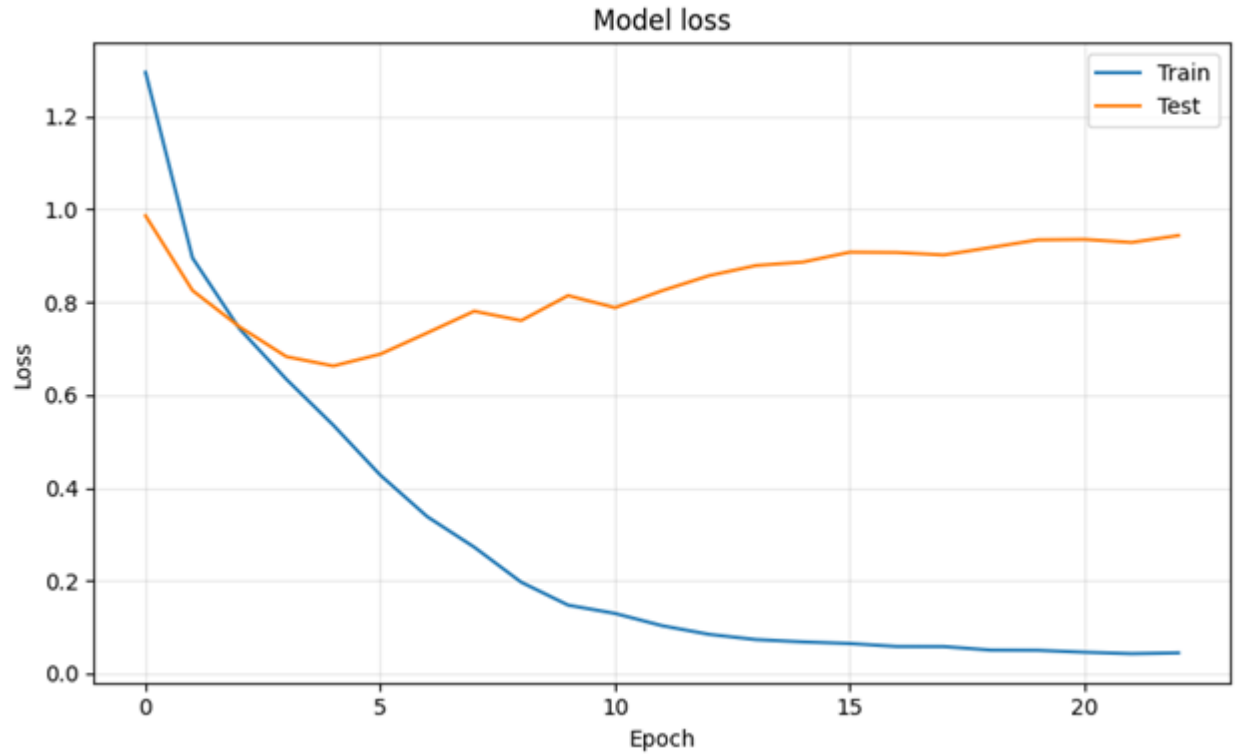

Figure 5.15: Model loss.

5.4.6 EmoDB DenCaps

Table 5.20: EmoDB - DenCaps

Run	Accuracy	Macro F1	Weighted F1	Balanced Acc	Loss
1	0.6413	0.4436	0.5564	0.5077	0.7408
2	0.6957	0.5625	0.6411	0.5683	0.5221
3	0.5870	0.4074	0.5130	0.4415	0.8703
4	0.6522	0.4560	0.5655	0.5012	0.6860
5	0.5000	0.2561	0.3739	0.3625	0.9591
6	0.4674	0.2462	0.3670	0.3400	0.8822
7	0.5326	0.2657	0.3894	0.3700	0.8712
8	0.5217	0.5093	0.5291	0.5413	0.6575
9	0.7065	0.5722	0.6644	0.6020	0.5790
10	0.5761	0.4397	0.5351	0.4768	0.7508

Table 5.21: EmoDB DenCaps Summary Table (Mean ± Std)

Metric	Mean	Std Dev
Accuracy	56.30%	4.93%
Macro F1	0.3671	0.098
Weighted F1	0.4738	0.0864
Balanced Accuracy	0.4289	0.073
Loss	0.8511	0.082
Best Val Accuracy	56.30%	4.93%

Table 5.22: Comparative Summary of CNN, CLSTM, and DenCaps Models (Mean ± Std Dev)

Model	Mean	Std Dev	95% CI Low	95% CI High
CNN	0.5157	0.0198	0.5015	0.5299
CLSTM	0.5784	0.0136	0.5686	0.5882
DenCaps	0.7106	0.0049	0.7070	0.7141
EmoDB DenCaps	0.5881	0.0833	0.5284	0.6477

Table 5.23: Cross-Dataset Statistical Comparison (DenCaps Multilingual vs EmoDB)

Comparison	Mean Diff	p-value	t-stat	Cohen's d	Significant
DenCaps vs CLSTM	0.1322	0.0000	28.6652	9.0647	Yes
DenCaps vs CNN	0.1949	0.0000	30.1298	9.5279	Yes
CLSTM vs CNN	0.0627	0.0000	7.3434	2.3222	Yes
DenCaps vs EmoDB	0.1225	0.0002	4.6008	1.4549	Yes

5.4.7 Statistical Analysis of Multilingual and Cross-Dataset Results

To ensure that the observed performance differences between models are not attributable to random variation, each architecture was evaluated over ten independent runs. Summary statistics, including mean and standard deviation, were computed for all evaluation metrics. A paired t-test was conducted to compare model performance using accuracy as the primary evaluation metric. The results illustrated in Table 5.22 and Table 5.23 indicate that the proposed DenCaps model significantly outperforms both CNN and CLSTM architectures ($p < 0.001$). In addition to statistical significance, effect sizes were calculated using Cohen's d. The results demonstrate large effect sizes across all comparisons, confirming that the observed improvements are both statistically significant and practically meaningful. To further evaluate generalisation, the DenCaps model was tested on the EmoDB dataset. A paired t-test between multilingual and EmoDB results confirmed statistically significant differences ($p < 0.001$), indicating that while the model generalises well, dataset-specific characteristics still influence performance.

5.4.8 Fuzzy Probabilistic Classification

The classification procedure is defined as follows: a random sample is selected from the test dataset. The selected sample is passed through a trained model to predict the probabilities of different emotions. The predicted probabilities for each emotion are normalised to 100% by dividing each probability by the total sum and multiplying by 100, giving the percentage representation of each emotion for that sample.

In Figure 5.16, anticipation is observed in the 95%+ percentile, indicating a clear and unambiguous emotional classification.

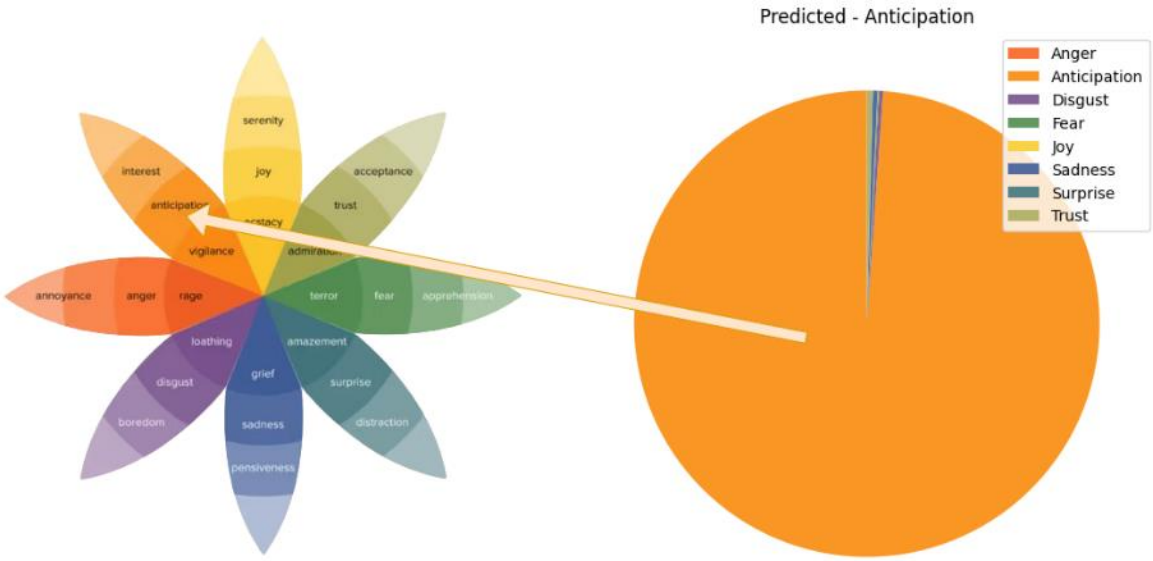

Figure 5.16: Fuzzy detection mapped to Plutchik - Example 1 - Anticipation.

In Figure 5.17 The highest percentage is for disgust; second place is for joy, sadness and surprise. Sadness pulls the resulting vector down, joy upwards, and surprise pulls it rightwards and slightly downwards. The other emotions are neglected because they do not have significant representation.

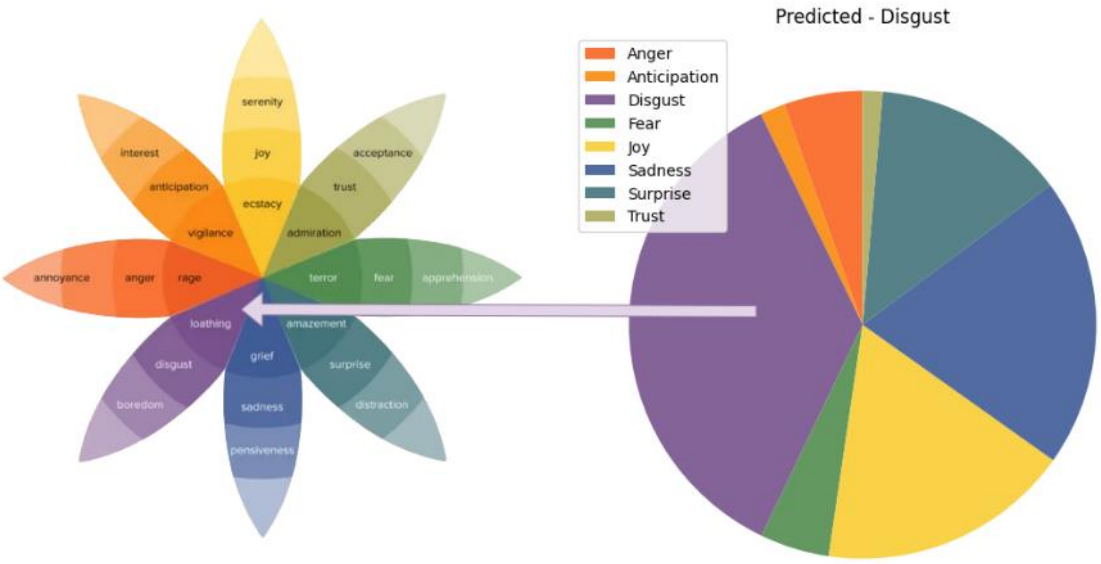

Figure 5.17: Fuzzy detection mapped to Plutchik - Example 2 - Loathing.

In Figure 5.18 It can be seen that the highest percentage is for joy. Secondly, anger. Disgust and sadness are in third place. The resulting vector is pulled left and down towards anger, then pulled slightly downwards by disgust and sadness.

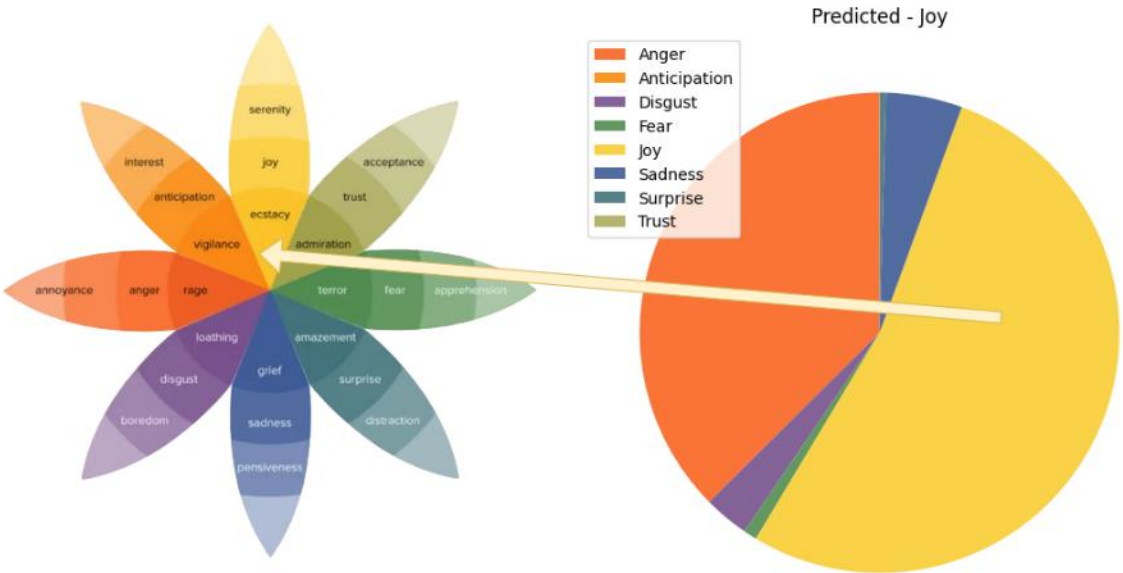

Figure 5.18: Fuzzy detection mapped to Plutchik - Example 3 - Ecstasy.

5.4.9 Class Imbalance Analysis and Mitigation

The multilingual dataset (MULTI_LANGUAGE_Final) exhibits significant class imbalance, as evidenced by both the number of audio samples per emotion class and the storage size of

the raw audio files, as shown in the dataset directory analysis illustrated in Table 5.24 below.

Table 5.24: Class Imbalance Analysis

Emotion	Size	Number of Files	Percentage of
Trust	1.8 GB	1,871	22.5
Joy	1.6 GB	4,407	19.2
Disgust	1.4 GB	3,728	17.2
Anger	1.3 GB	3,213	15.9
Sadness	936 MB	4,315	11.2
Anticipation	784 MB	3,207	9.4
Fear	261.2 MB	1,613	3.1
Surprise	129.5 MB	1,162	1.6
Total	8.2 GB	23,517	100.0

This distribution is typical in a speech emotion recognition corpus, where emotions such as Joy, Sadness, and Trust are more frequently represented than rarer emotions such as Surprise and Fear. Without mitigation, models tend to become biased toward majority classes, producing inflated overall accuracy while performing poorly on minority classes. The effect is evident when comparing accuracy with the macro-averaged F1-score.

To address the imbalance, softened class weights were applied during training. Raw balanced weights were first computed using sklearn.utils.class_weight.compute_class_weight('balanced') and then softened with $\alpha = 0.5$ using the formula:

$$wc' = 1 + \alpha(wc - 1) \quad (5.1)$$

5.5 Model Training, Convergence, and Hyperparameter Configuration

The reported performance metrics reflect results obtained under the defined experimental conditions and multilingual datasets. They do not include systematic sensitivity analysis across variables such as noise levels, speaker characteristics, recording quality, or speech segment length.

The DenCaps model was trained using the Adam optimiser with categorical cross-entropy loss and accuracy as the primary evaluation metric. The architecture integrates convolutional, dendritic, and capsule-based components to capture both spatial and hierarchical relationships within speech features. Convergence behaviour was assessed by analysing the training and validation performance across epochs. The training history demonstrated a consistent decrease in loss and stabilisation of validation accuracy in later epochs, indicating that the model converged without significant overfitting. The highest validation accuracy achieved during training was recorded and used as an indicator of optimal model performance. The hyperparameters used in the DenCaps model were selected based on empirical testing and

established practices in deep learning for speech processing. Although a comprehensive sensitivity analysis was beyond the scope, the selected configuration yielded stable and reproducible results across multiple experimental runs. Table 5.25 summarises the key hyperparameters used in the DenCaps model.

Table 5.25: DenCaps Hyperparameters

Component	**Parameter**	**Value**
Input	Input shape	(num_timesteps, num_features, 1)
Conv2D Layer 1	Filters	128
Conv2D Layer 1	Kernel size	(3 x 3)
Conv2D Layer 1	Activation	ReLU
Conv2D Layer 2	Filters	32
Conv2D Layer 2	Kernel size	(3 x 3)
Conv2D Layer 2	Activation	ReLU
MaxPooling2D	Pool size	(2 x 2)
Dropout Layer 1	Dropout rate	0.5
Dendritic Layer	Units	64
Dendritic Layer	Dendritic segments	2
Capsule Layer	Number of capsules	10
Capsule Layer	Capsule dimension	32
Capsule Layer	Routing iterations	3
Dense Layer	Units	32
Dense Layer	Activation	ReLU
Dropout Layer 2	Dropout rate	0.5
Output Layer	Units	num_classes
Output Layer	Activation	Softmax
Optimiser	Type	Adam
Loss Function		Categorical Cross-Entropy
Batch Size		32
Epochs		50
Data Handling	Shuffle	True
Validation		Held-out test set

5.5.1 Computational Complexity of DenCaps

For consistency with the analysis presented in Section 0, the computational complexity of the DenCaps architecture is reported in Table 5.26. The same profiling protocol was applied: parameters from model.summary(), FLOPs from TensorFlow's static graph profiler, and per-epoch training time averaged over six epochs on the same hardware.

Table 5.26: Computational complexity of the DenCaps architecture (multilingual corpus, batch size 32, mean of 3 epochs).

Model	Input Shape	Parameters	FLOPs (forward pass)	Mean epoch time (s)	Std (s)
DenCaps	(40, 1047, 1)	38,480,648	2.94 G	12.28	16.44

5.6 Discussion and Conclusion

The statistical analysis presented in Table 5.22 and Table 5.23 confirms that the DenCaps architecture demonstrates a statistically significant improvement over both CNN and CLSTM models within the multilingual experimental setting. The observed improvements are not due to random variation, as demonstrated by the paired t-test results and large effect sizes. Furthermore, the comparison with the EmoDB dataset highlights the model's ability to generalise across datasets. Although performance decreases on EmoDB, the model maintains competitive accuracy, indicating robust feature learning and adaptability to different linguistic conditions. These findings validate the effectiveness of combining dendritic processing with Capsule Networks, thereby improving the representation of hierarchical and contextual emotional features in multilingual speech data.

It is important to contextualise these results relative to the findings in Chapter 4, where the DCLSTM model achieved a higher accuracy of approximately 78.50% on the Afrikaans single-language corpus. The DenCaps model addresses a more complex problem, namely multilingual speech emotion recognition across heterogeneous datasets. The reduction in accuracy is therefore expected, as multilingual SER introduces additional variability in phonetics, prosody, recording conditions, and emotional expression across languages.

Consequently, the contribution of the DenCaps architecture should not be interpreted as an improvement in absolute accuracy over DCLSTM, but rather as an extension of the model's capability to generalise across languages while maintaining competitive performance. The results align with Hypothesis H3, which emphasises cross-language generalisation rather than single-language optimisation. The distinction illustrates the trade-off between optimisation for controlled single-language datasets and generalisation for real-world multilingual applications.

5.6.1 CNN Performance

As shown in Table 5.15, the CNN model achieved a mean accuracy of 51.57% ± 1.88% across ten independent runs, with a macro F1-score of 0.4463 ± 0.0367 and balanced accuracy of 0.4655 ± 0.0271. These results highlight the limitations of traditional CNN architectures in handling the complex temporal dynamics and contextual nuances inherent in emotional speech. While CNNs effectively capture local spatial features, they struggle with long-term dependencies and the subtleties of emotional expressions over time. The results are consistent with related work highlighting the limitations of CNNs in processing non-uniform, fixed-size audio segments, which are critical for accurate emotion detection (Ding et al., 2024).

5.6.2 CLSTM Performance

Table 5.17 illustrates the performance of the CLSTM model, which achieved a mean accuracy of 57.84% ± 1.29% across ten independent runs, with a macro F1-score of 0.5168 ± 0.0143 and balanced accuracy of 0.5269 ± 0.0158. The CLSTM architecture, which integrates the spatial feature extraction capabilities of CNNs with the sequential processing strengths of LSTMs, shows an improvement over the CNN models. The observed improvement indicates that combining convolutional layers with LSTM units enhances the modelling of temporal dependencies and contextual factors, which are essential for recognising emotions in speech. The results align with the advantages of CLSTM models discussed in the literature, where they are noted for their ability to integrate features across both time and space, enhancing the detection and classification of emotions in speech (Zhao et al., 2022).

5.6.3 DenCaps Performance

The DenCaps model in Table 5.18 achieved the highest accuracy. The improved performance of the DenCaps architecture can be attributed to its combination of dendritic layers and Capsule Networks. Dendritic layers enhance the model's ability to process complex input patterns through sophisticated, non-linear integration of synaptic inputs, mimicking biological neural processes. Capsule Networks contribute by recognising hierarchical patterns and maintaining spatial relationships within the data. The integration of dendritic neural layers with Capsule Networks enables DenCaps to capture the temporal and spatial characteristics of emotional speech, thereby improving emotion recognition accuracy. These findings are supported by recent advancements in integrating dendritic computations and Capsule Networks, which have shown promise in improving the robustness and interpretability of deep learning models for speech emotion recognition (Ji, Tang, Zhao, Tang, & Todo, 2022).

To contextualise the DenCaps performance against state-of-the-art approaches, summarises representative published results for transformer-based and hybrid models on benchmark SER datasets. Direct numerical comparison is constrained by differences in dataset composition, language coverage, emotion taxonomies, and evaluation protocols; the figures are therefore reported for contextual reference rather than as a like-for-like benchmark.

Table 5.27: Comparative reference of published SER results and the DenCaps model

Model	**Architecture type**	**Dataset**	**Languages**	**Emotions**	**Reported accuracy**
wav2vec 2.0 (Pepino, Riera, & Ferrer, 2021)	Transformer (frozen, pre-trained)	IEMOCAP	English	4	72.1% (UAR)
HuBERT-large, fine-tuned (Wang, Boumadane, & Heba, 2021)	Transformer (fine-tuned)	IEMOCAP	English	4	73.0% (WA, speaker-independent)

2D CLSTM (Zhao, Mao, & Chen, 2019)	Hybrid CNN + LSTM	Berlin EMO-DB	German	7	95.9% (speaker-independent)
Deep-Net (T., Mustaqeem, & S., 2020)	Lightweight CNN	EMO-DB / IEMOCAP	German / English	7 / 4	92.0% / 77.0%
Wav2Vec2 + attention fusion (Naderi & Nasersharif, 2023)	Transformer hybrid	EMO-DB / IEMOCAP	German / English	7 / 4	95.4% / 76.9%
DenCaps (this work)	Hybrid (Dendritic + Capsule)	Multilingual	7	8	71.91%

The published transformer-based results — wav2vec 2.0 at 72.1% UAR (Pepino, Riera, & Ferrer, 2021) and fine-tuned HuBERT-large at 73.0% WA in the speaker-independent setting (Wang, Boumadane, & Heba, 2021)— were obtained on the four-class IEMOCAP corpus in a single language (English) and benefit from large-scale unsupervised pre-training on thousands of hours of speech. Hybrid CLSTM and lightweight CNN models report higher absolute accuracies (92–96%) on EMO-DB, but EMO-DB is a single-language seven-class corpus of approximately 500 utterances recorded under controlled conditions. The DenCaps result, by contrast, was obtained under a substantially harder evaluation regime: eight emotion classes, seven languages, and approximately 23,500 samples, without large-scale pretraining. The 71.91% accuracy is therefore competitive given the increased class cardinality, linguistic diversity, and absence of transfer from a foundation model. Direct comparison with published work is otherwise limited by differences in corpus composition, emotion taxonomies, and evaluation protocols. Accordingly, the DenCaps result of 71.91% should be interpreted as a competitive outcome within the defined experimental framework and resource constraints, rather than as a claim of state-of-the-art performance. A direct head-to-head benchmark against wav2vec 2.0 and HuBERT on an identical multilingual eight-class evaluation protocol is identified as a priority for future work in the section 6.2.

5.6.4 Conclusion

A hybrid speech emotion recognition (SER) model, DenCaps, was developed by integrating dendritic neural layers with Capsule Networks. The study explored the theoretical foundations of both approaches and applied them to multilingual speech data to evaluate their effectiveness in capturing complex emotional patterns.

The DenCaps model demonstrated improved performance compared with the baseline CNN and CLSTM models, achieving an accuracy of up to 71.91%. These results suggest that combining dendritic processing with capsule-based architectures can enhance the modelling of both temporal and spatial characteristics of emotional speech.

While the results are promising, the evaluation was limited to the architectures and datasets

considered. Direct comparison with recent transformer-based models was beyond the scope. Future research may extend the approach by benchmarking DenCaps against large pre-trained models such as wav2vec 2.0 and HuBERT, particularly as larger and more diverse multilingual datasets become available.

CHAPTER 6: Conclusion, Research Summary and Future Recommendations

This thesis follows a coherent and progressive research pipeline aligned with its three objectives. Chapter 3 addressed the scarcity of resources for African languages by developing a dedicated Afrikaans speech corpus. In Chapter 4, the corpus was used to optimise hybrid neural architectures and preprocessing techniques in a controlled setting, with the DCLSTM model identified as the most effective architecture and the Kalman_2 filter as the most effective noise-reduction technique on Afrikaans audio. Chapter 5 then extended the architectural findings — specifically the dendritic-layer contribution — to a multilingual context through the DenCaps model. The Kalman_2 preprocessing component was not transferred to the multilingual setting because the constituent corpora are released pre-cleaned by their original curators; cross-corpus application of Kalman_2 is identified as future work.

This progression—resource creation, targeted optimisation, and multilingual extension—forms the central thread of the thesis, demonstrating how each stage builds on the previous to advance robust, generalisable speech emotion recognition systems.

6.1 Research summary and discussion

Chapters 3 to 5 follow a progressive structure: corpus development (Chapter 3), architecture optimisation in a controlled single-language setting (Chapter 4), and multilingual scaling and generalisation (Chapter 5).

From the research objectives, the following was achieved:

- Compilation of a high-quality Afrikaans speech corpus tailored for Speech Emotion Recognition (SER). Audio samples were collected and processed from various Creative Commons sources, focusing on the primary emotions identified in Plutchik's Wheel of Emotions. To overcome data scarcity, Generative Adversarial Networks (GANs) were employed to generate synthetic speech, further enriching the dataset. The newly developed corpus addresses a significant gap in African language resources for SER research and is publicly accessible to support future developments.
- Evaluation of advanced audio pre-processing techniques to optimise SER model performance. Techniques such as noise reduction, spectral subtraction, pre-emphasis filtering, framing, windowing, and silence removal were systematically tested to improve input data quality. The pre-processed audio data enabled better clarity and increased accuracy when fed into neural network models, providing a foundation for enhanced SER systems.
- Development and assessment of hybrid neural network architectures combining Convolutional Neural Networks (CNNs) with Recurrent Neural Networks (RNNs) and Dendritic Capsule Networks (DenCaps). The hybrid architectures significantly improved in recognising emotional speech, particularly in a multilingual context. By integrating advanced models such as DenCaps, which mimic biological dendritic processing. Performance exceeding that of traditional CNN and RNN models was achieved, demonstrating the effectiveness of these architectures for SER.

A structured research methodology was applied to explore and address critical challenges in Speech Emotion Recognition (SER), focusing specifically on the Afrikaans language and multilingual models. By compiling a novel Afrikaans speech corpus, applying advanced pre-processing techniques, and using hybrid CLSTM architectures, the research aims to enhance SER accuracy and usability in multilingual contexts. The solutions developed include the creation of a comprehensive Afrikaans speech corpus, optimising speech pre-processing pipelines, and applying multilingual training to improve model performance across languages. Unlike previous studies focusing primarily on single-language models or data collection methods, the research integrates advanced technical innovations and data augmentation strategies to provide a more practical and adaptable framework for SER. Chapter 1 outlined the research questions and objectives, highlighting the need for better SER systems for underrepresented languages like Afrikaans. Chapter 2 presents a detailed literature review covering foundational concepts in human emotion, speech processing, and neural networks. The background provides a foundation for the contributions, including emotion detection using speech signals and the importance of developing culturally and linguistically diverse SER models. Chapter 3 discusses the compilation of the Afrikaans speech corpus, an essential resource that fills a gap in available SER datasets. The collection process, data validation procedures, and ethical considerations involved in creating the corpus are described. Chapter 4 presents the design of a hybrid neural network model that integrates CNN and LSTM layers to capture spatial and temporal speech features. Pre-processing techniques such as noise reduction, silence removal, and spectral subtraction were optimised to enhance the accuracy of SER systems, particularly for multilingual tasks. Chapter 5 evaluates the overall performance of the multilingual SER system by testing it on various single corpus datasets, including Afrikaans. The hybrid models demonstrate significant improvements in detecting emotions across multiple languages. The results of the optimised techniques are compared with those of established models, demonstrating improved performance, particularly in multilingual settings.

The research contributes to the field of SER by providing a comprehensive and scalable solution for underrepresented languages, enriching the body of knowledge on multilingual speech emotion detection to practical applications, and offering a refined pre-processing and hybrid neural network architecture for businesses and industries seeking to integrate emotion-aware technologies, and to future research by paving the way for further innovations in multilingual SER systems.

The achieved accuracy of 71.91% in the multilingual setting represents a meaningful improvement over earlier results and demonstrates the effectiveness of the proposed approach. While this performance is slightly lower than the single-language Afrikaans model (78.50%), this is expected due to the increased complexity and variability inherent in multilingual datasets. Therefore, the achieved accuracy is considered satisfactory within the context of the research objectives, while still allowing room for further improvement.

6.2 Future recommendations

Future studies should also conduct sensitivity and robustness analyses of the proposed architectures and preprocessing techniques (particularly Kalman_2) across varying acoustic conditions, speaker characteristics (e.g., gender, age), recording quality, and utterance length to better quantify performance variability and generalisability. In particular, applying the

Kalman_2 preprocessing pipeline to the multilingual corpora — which was not done in the present work due to differences in source-corpus preparation — is a priority for follow-up evaluation, to determine whether the noise-reduction gains observed on Afrikaans transfer to the cross-lingual setting. The research outcomes indicate several areas for future exploration. Further studies are recommended to expand and enrich the Afrikaans speech corpus by incorporating additional South African languages, thereby promoting linguistic diversity and improving the robustness of multilingual models. In line with this, future work could expand on advanced preprocessing techniques, investigating more sophisticated noise reduction algorithms and real-time audio processing methods to improve system efficiency and accuracy in practical applications. Furthermore, refining hybrid neural network architectures should be a key focus. Future research could explore integrating attention mechanisms and transformer models to improve the system's context understanding capabilities. Additionally, researchers should evaluate the scalability of models like the DenCaps across more extensive and varied datasets, ensuring that these models can perform reliably in real-world scenarios. Lastly, cross-modal emotional recognition warrants further investigation, allowing for the detection of emotions across multiple modalities beyond speech to create more comprehensive and accurate systems. Future studies will contribute significantly to advancing Speech Emotion Recognition systems and their practical deployment in diverse, multilingual, and multi-modal environments by addressing these areas. The research questions, objectives and answers are illustrated in Figure 6.1. Future work should include benchmarking the proposed architectures against state-of-the-art models such as wav2vec 2.0 and HuBERT on standardised datasets to enable direct comparison with existing literature.

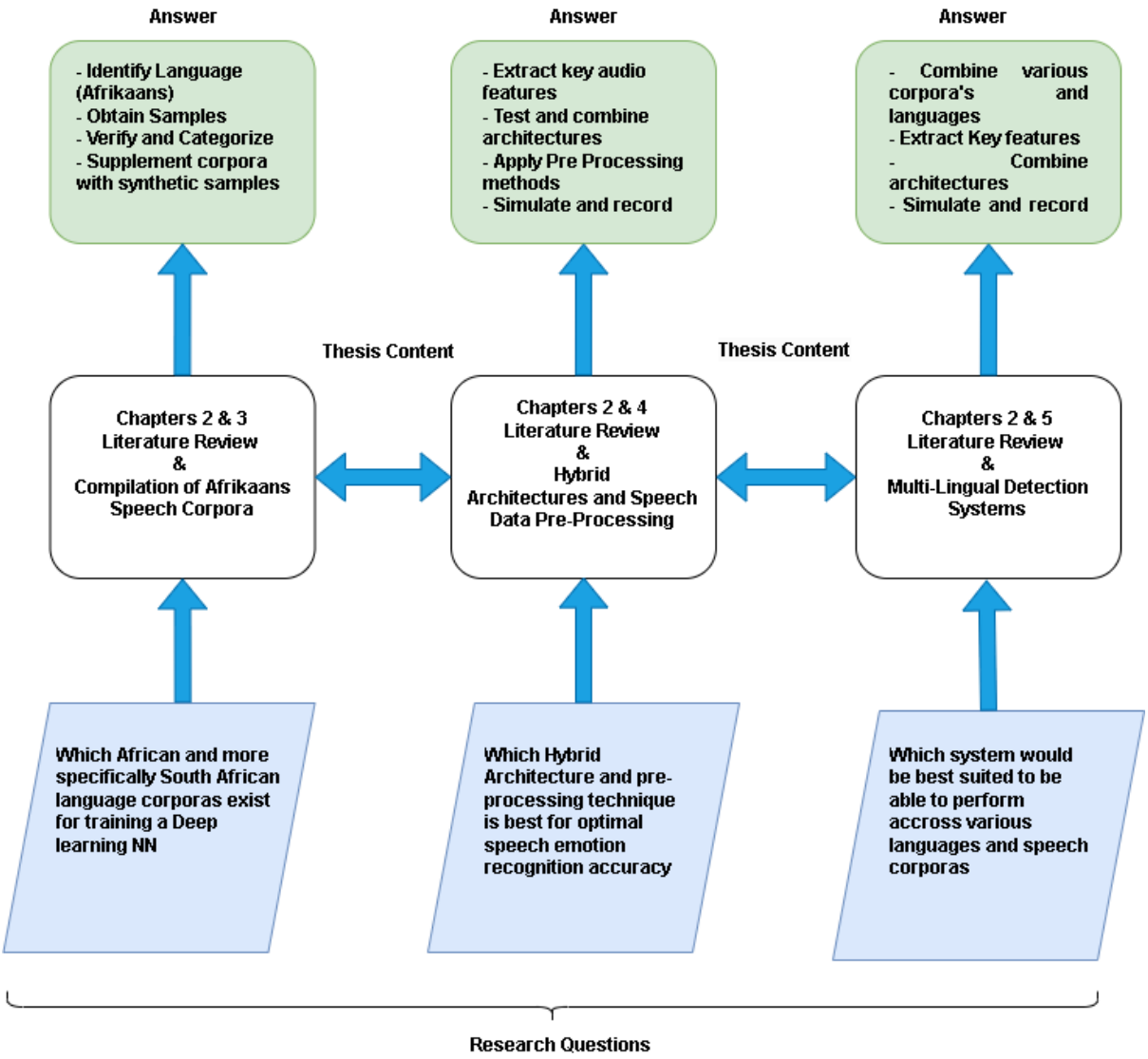

Figure 6.1: Research questions, objectives and answers.

The research did not address integrating the proposed Speech Emotion Recognition (SER) model with existing commercial speech-recognition systems. Future studies should investigate efficient methods for integrating legacy systems, such as current voice assistants and existing speech processing frameworks, with advanced SER techniques. Such integration would enable a smoother transition for industries relying on traditional speech recognition technologies while facilitating the progressive replacement of outdated components.

REFERENCES

Abate, S. T., Tachbelie, M. Y., & Schultz, T. (2020). Multilingual Acoustic and Language Modeling for Ethio-Semitic Languages. *Proceedings of Interspeech*, 1047–1051. doi:10.21437/Interspeech.2020-2856

Abid, M. A., & Munir, K. (2025). A systematic review on deep learning implementation in brain tumor segmentation, classification and prediction. *Multimedia Tools and Applications, 84*, 38203–38242. Retrieved from https://doi.org/10.1007/s11042-025-20706-4

AbuBaker, A., Eshtay, M., & AkhoZahia, M. (2016). Comparison Study of Different Lossy Compression Techniques Applied on Digital Mammogram Images. *International Journal of Advanced Computer Science and Applications, 7*(12), 149–155. Retrieved from https://doi.org/10.14569/ijacsa.2016.071220

Acoustics, I. C. (2020-2021). *International Year of Sound*. Retrieved April 30, 2026, from https://sound2020.org/: https://www.sound2020.org/sound/audible/

Adeel, M., Tao, Z.-Y., Jin, S.-Y., Guo, C.-J., & Alsuhaibani, M. (2026). Assessing random forest performance in low resource speech emotion recognition. *Scientific Reports, 16*(1), 854. Retrieved from https://doi.org/10.1038/s41598-025-30511-6

Adigwe, A., Tits, N., Haddad, K. E., Ostadabbas, S., & Dutoit, T. (2018). The emotional voices database: Towards controlling the emotion dimension in voice generation systems. *arXiv preprint*. Retrieved from https://arxiv.org/pdf/1806.09514.pdf

Africa, S. S. (2023). *Census 2022: Statistician-General's presentation*. Retrieved from https://census.statssa.gov.za/assets/documents/2022/Census_2022_SG_Presentation_10102023.pdf

Aliferis, C., & Simon, G. (2024). Overfitting, Underfitting and General Model Overconfidence and Under-Performance Pitfalls and Best Practices in Machine Learning and AI. *Artificial Intelligence and Machine Learning in Health Care and Medical Sciences: Best Practices and Pitfalls*, 477-524. Retrieved from https://doi.org/10.1007/978-3-031-39355-6_10

Al-Shawaf, L., & Lewis, D. (n.d.). Evolutionary Psychology and the Emotions. *Emotion Review, 8*(2), 173-186. doi:10.1007/978-3-319-28099-8_516-1

Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., . . . Farhan, L. (2021). Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. *Journal of Big Data, 8*, Article 53. Retrieved from https://doi.org/10.1186/s40537-021-00444-8

Arimoto, Y., Kawatsu, H., Ohno, S., & Iida, H. (2012). Naturalistic emotional speech collection paradigm with online game and its psychological and acoustical assessment. *Acoustical Science and Technology, 33*(6), 359-369. Retrieved from https://doi.org/10.1250/ast.33.359

Arsalan, A., Burhan, M., Rehman, R. A., Umer, T., & Kim, B.-S. (2023). An Efficient Data Retrieval and Forwarding Technique for Named Data Network Based Wireless Multimedia Sensor Networks. *IEEE Access, 11*, 15315-15328. doi:10.1109/ACCESS.2023.3244247

Atif, M., Shafiq, M., Farooq, M., Ayub, G., Leisch, F., Ilyas, M., & Ahsan, M. (2023). Monitoring Changes in Clustering Solutions: A Review of Models and Applications. *Journal of Probability and Statistics, 2023*(1), 7493623. Retrieved from https://doi.org/10.1155/2023/7493623

Barhoumi, C., & BenAyed, Y. (2025). Real-time speech emotion recognition using deep learning and data augmentation. *Artificial Intelligence Review, 58*(2), 49. doi:https://doi.org/10.1007/s10462-024-11065-x

Barnard, E., Davel, M. H., Heerden, C. v., Wet, F. d., & Badenhorst., J. (2014). The NCHLT Speech Corpus of the South African Languages. *4th International Workshop on Spoken Language Technologies for Under-Resourced Languages*, (pp. 194-200). St Petersburg, Russia. Retrieved from http://hdl.handle.net/10204/7549

Bartlett, C. W., Bossenbroek, J., Ueyama, Y., McCallinhart, P., Peters, O. A., Santillan, D. A., . . . Ray, W. C. (2023). Invasive or More Direct Measurements Can Provide an Objective Early-Stopping Ceiling for Training Deep Neural Networks on Non-invasive or Less-Direct Biomedical Data. *SN Computer Science, 4*(2), 161. Retrieved from https://doi.org/10.1007/s42979-022-01553-8

Bas, E., Egrioglu, E., & Cansu, T. (2024). Robust training of median dendritic artificial neural networks for time series forecasting. *Expert Systems with Applications, 238*, Article 122080. Retrieved from https://doi.org/10.1016/j.eswa.2023.122080

Basu, S., Chakraborty, J., Bag, A., & Aftabuddin, M. (2017). A Review on Emotion Recognition using Speech. *International Conference on Inventive Communication and Computational Technologies (ICICCT)*, *IEEE*, pp. 109-114. Coimbatore, India. Retrieved from https://ieeexplore.ieee.org/document/7975169/

Belkin, M., Hsu, D., Ma, S., & Mandal, S. (2019). Reconciling modern machine-learning practice and the classical bias–variance trade-off. *PNAS, 116*(32), 15849–15854. Retrieved from https://doi.org/10.1073/pnas.1903070116

Berridge, K. C., & Kringelbach., M. L. (2015). Pleasure Systems in the Brain. *Neuron, 86*(3), 646-664. Retrieved from https://doi.org/10.1016/j.neuron.2015.02.018

Bojja, R. R. (2025). A Comparative Study of Supervised and Unsupervised Learning Approaches. *IMRJR*, 80-88. doi:10.17148/IMRJR.2025.020411

Brave, S., & Nass, C. (2003). Emotion in human-computer interaction. *The human-computer interaction handbook: Fundamentals, evolving technologies and emerging applications*, 81-96. doi:10.1201/9781410615862-13

Brownlee, J. (2018). *A Gentle Introduction to Dropout for Regularizing Deep Neural Networks*. Retrieved April 30, 2026, from Machine Learning Mastery: https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/

Burkhardt, F., Paeschke, A., Rolfes, M., Sendlmeier, W. F., & Weiss, B. P. (2005). A Database of German Emotional Speech., (pp. 1517-1520). Rotterdam, Netherlands. Retrieved from https://doi.org/10.21437/interspeech.2005-446

Calinger, R. (1999). *A Contextual History of Mathematics to Euler.* Upper Saddle River, NJ : Prentice Hall.

Carvalhais, T., & Magalhães, L. (2018, November). Recognition and Use of Emotions in Games., (pp. 1-8). Lisbon, Portugal. Retrieved from https://doi.org/10.1109/itcgi.2018.8602898

Casale, S., Russo, A., G. Scebba, & Serrano, S. (2008, August). Speech Emotion Classification Using Machine Learning Algorithms., (pp. 158-165). Santa Clara, California. doi:10.1109/ICSC.2008.43

Chakravarthy, V. (2019). Circuits of Emotion. *Demystifying the Brain*, 285–319. Retrieved from https://doi.org/10.1007/978-981-13-3320-0_10

Chaudhry, M., Shafi, I., Mahnoor, M., Vargas, D. L., Thompson, E. B., & Ashraf, I. (2023). A Systematic Literature Review on Identifying Patterns Using Unsupervised Clustering Algorithms: A Data Mining Perspective. *Symmetry, 15*, 1679. doi:10.3390/sym15091679

Chen, K., Yue, G., Yu, F., Shen, Y., & Zhu, A. (2007). Research on Speech Emotion Recognition System in E-Learning. *ICCS 2007* (pp. 555-558). Beijing: ICCS 2007, 7th International Conference. Retrieved from https://doi.org/10.1007/978-3-540-72588-6_91

Chen, Y., Li, L., Li, W., Guo, Q., Du, Z., & Xu, Z. (2024). Fundamentals of neural networks. *AI Computing Systems*, 17–51. Retrieved from http://dx.doi.org/10.1016/b978-0-32-395399-3.00008-1

Cheng, H., Zhang, M., & Shi, J. Q. (2024). A Survey on Deep Neural Network Pruning: Taxonomy, Comparison, Analysis, and Recommendations. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 46*(12), 10558-10578. doi:10.1109/TPAMI.2024.3447085

Deep learning vs. machine learning in Azure Machine Learning. (2026). Retrieved April 30, 2026, from Microsoft Build 2026: https://learn.microsoft.com/en-us/azure/machine-learning/concept-deep-learning-vs-machine-learning?view=azureml-api-2

DeepAI. (2019, 17 March). *Rectified Linear Units*. Retrieved from DeepAI: https://deepai.org/machine-learning-glossary-and-terms/rectified-linear-units

Dejoli, T. T., He, Q., & Xie, W. (2021). Audio,Speech and Vision Processing Lab Emotional Sound database (ASVP-ESD). Retrieved from https://doi.org/10.5281/ZENODO.4782712

Delua, J. (2021, March 12). *Supervised vs. Unsupervised Learning: What's the Difference?* Retrieved from Www.ibm.com: https://www.ibm.com/cloud/blog/supervised-vs-unsupervised-learning

Deschamps-Berger, T., Lamel, L., & Devillers, L. (2021, September). Emotion Recognition in Emergency Call Centers: The Challenge of Real-Life Emotions., (pp. 1-5). Nara, Japan. Retrieved from https://doi.org/10.1109/aciiw52867.2021.9666308

Ding, Y., Yu, J., Gu, C., Gao, S., & Zhang, C. (2024). A multi-in and multi-out dendritic neuron model and its optimization. *Knowledge-Based Systems, 286*, 111442. Retrieved from https://doi.org/10.1016/j.knosys.2024.111442

docs.aws.amazon.com. (2023, Mar 23). *Model Fit: Underfitting vs. Overfitting*. Retrieved from Amazon Machine Learning: https://docs.aws.amazon.com/machine-learning/latest/dg/model-fit-underfitting-vs-overfitting.html

Drucker, H., & Cun, Y. L. (1992). Improving generalization performance using double backpropagation. *IEEE Transactions on Neural Networks, 3*(6), 991-997. doi:10.1109/72.165600

Egrioglu, E., Ba¸s, E., & Chen, M.-Y. (2022). Recurrent dendritic neuron model artificial neural network for time series forecasting. *Information Sciences, 607*, 572–584. Retrieved from https://www.sciencedirect.com/science/article/pii/S0020025522005941

Fahmy, M. M. (2010). Palmprint Recognition Based on Mel Frequency Cepstral Coefficients Feature Extraction. *Ain Shams Engineering Journal, 1*(1), 39-47. Retrieved from https://doi.org/10.1016/j.asej.2010.09.005

Firth-Godbehere, R. (2023). *A HUMAN HISTORY OF EMOTION : how the way we feel built the world we know.* London: Fourth Estate Ltd. Retrieved from https://www.harpercollins.co.uk/9780008393755/a-human-history-of-emotion/

Fredrickson, B. L. (1998). What Good Are Positive Emotions? *Review of General Psychology, 2*(3), 300-319. Retrieved from https://doi.org/10.1037/1089-2680.2.3.300

Gan, W. S. (2020). Fast Fourier Transform. *Signal Processing and Image Processing for Acoustical Imaging*, 17–20. Retrieved from https://doi.org/10.1007/978-981-10-5550-8_5

Gasmi, A. (2022). What is Fast Fourier Transform? *Société Francophone de Nu-trithérapie et de Nutrigénétique Appliquée.*, hal-03741810. Retrieved from https://hal.science/hal-03741810v1

Gbadegeshin, S. A., Natsheh, A. A., Ghafel, K., Tikkanen, J., Gray, A., Rimpiläinen, A., . . . Hirvonen, N. (2021, November). WHAT IS an ARTIFICIAL INTELLIGENCE (AI): A SIMPLE BUZZWORD or a WORTHWHILE INEVITABILITY? Online Conference: ICERI2021 Proceedings. doi:10.21125/iceri.2021.0171

George, S. M., & P. Muhamed Ilyas. (2024). A review on speech emotion recognition: A survey, recent advances, challenges, and the influence of noise. *Neurocomputing, 568*. Retrieved from https://doi.org/10.1016/j.neucom.2023.127015

Gers, F. A. (2001). LSTM recurrent networks learn simple context-free and context-sensitive languages. *IEEE TNN, 12*(6), 1333-1340. doi:https://doi.org/10.1109/72.963769

Gershenson, C. (2003). Artificial Neural Networks for Beginners. *arXiv: Neural and Evolutionary Computing*. Retrieved 1 31, 2023, from https://arxiv.org/pdf/cs/0308031

Giliomee, H. (2020). *The Afrikaners: A Concise History.* Cape Town, Cape Town: Tafelberg.

Goel, S., & Beigi, H. (2020). Cross Lingual Cross Corpus Speech Emotion Recognition. *arXiv*. Retrieved from https://arxiv.org/abs/2003.07996

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., . . . Bengio, Y. (2014). Generative Adversarial Networks. *Computer and information sciences*. Retrieved from https://doi.org/10.48550/arXiv.1406.2661

Gournay, P., Lahaie, O., & Lefebvre, R. (2018). A canadian french emotional speech dataset. 399–402. Retrieved from https://doi.org/10.1145/3204949.3208121

Gul, H. E. (2023). Statistical learning algorithms for dendritic neuron model artificial neural network based on sine cosine algorithm. *Information Sciences, 629*, 398-412. Retrieved from https://doi.org/10.1016/j.ins.2023.02.008

Hamans, C. (2021, December). Afrikaans: A Language Where Ideology and Linguistics Meet. *Scripta Neophilologica Posnaniensia, 21*, 15-92. doi:10.14746/snp.2021.21.02

Haque, M., & Bhattacharyya, K. (2018). Speech Background Noise Removal Using Different Linear Filtering Techniques. *Lecture Notes in Electrical Engineering, 475*, 297–307. Retrieved from https://doi.org/10.1007/978-981-10-8240-5_33

Harár, P., Burget, R., & Dutta, M. (2017, Feb). Speech emotion recognition with deep learning. *4th International Conference on Signal Processing and Integrated Networks (SPIN)*, (pp. 137-140). Noida, India. doi:10.1109/SPIN.2017.8049931

Healy, E. W., Johnson, E. M., Pandey, A., & Wang, D. (2023). Progress made in the efficacy and viability of deep-learning-based noise reduction. *The Journal of the Acoustical Society of America, 153*(5), 2751-2751. Retrieved from https://doi.org/10.1121/10.0019341

Heerden, C. v., Barnard, E., Badenhorst, J., Davel, M., & Waal, A. d. (2014). *NCHLT Afrikaans Speech Corpus*. Retrieved 4 20, 2023, from https://repo.sadilar.org/handle/20.500.12185/280

Hight, B. H., & Eljhani, M. M. (2013). Front-End of Wake-Up-Word Speech Recognition System Design on FPGA. *Journal of Telecommunications System & Management, 2*(1). Retrieved from https://doi.org/10.4172/2167-0919.1000108

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation, 9*(8), 1735–1780. Retrieved from https://doi.org/10.1162/neco.1997.9.8.1735

Houser, K. (2022, Feb). *Robot kid successfully conveys six emotions on its face*. Retrieved from https://www.freethink.com/technology/companion-robots

Hu, X., Chu, L., Pei, J., Liu, W., & Bian, J. (2021). Model Complexity of Deep Learning: A Survey. *Knowledge and Information Systems, 63*, 2585-2619. Retrieved from https://api.semanticscholar.org/CorpusID:232168493

Hussein, B. M., & Shareef, S. M. (2024). An Empirical Study on the Correlation between Early Stopping Patience and Epochs in Deep Learning. *ITM Web Conf*, 01003. doi:https://doi.org/10.1051/itmconf/20246401003

Ilm, I. (2018). *MFCC implementation and tutorial*. Retrieved from https://www.kaggle.com/code/ilyamich/mfcc-implementation-and-tutorial/notebook

Ingle, V. K., & Proakis, J. G. (2012). *Digital Signal Processing Using MATLAB.* Stamford: Cengage Learning.

Ji, J., Tang, C., Zhao, J., Tang, Z., & Todo, Y. (2022). A survey on dendritic neuron model: Mechanisms, algorithms and practical applications. *Neurocomputing, 489*, 390–406. Retrieved from http://dx.doi.org/10.1016/j.neucom.2021.08.153

Johnson, M. K. (2020). Joy: a review of the literature and suggestions for future directions. *The Journal of Positive Psychology, 15*(1), 5-24. Retrieved from https://doi.org/10.1080/17439760.2019.1685581

Khalil, R. A., Jones, E., Babar, M. I., Jan, T., Zafar, M. H., & Alhussain, T. (2019). Speech Emotion Recognition Using Deep Learning Techniques: A Review. *IEEE Access, 7*, 117327-117345. doi:10.1109/ACCESS.2019.2936124

Kilimci, Z. H., Bayraktar, Ü., & Küçükmanisa, A. (2025). Evaluating raw waveforms with deep learning frameworks for speech emotion recognition. *Multimedia Tools and Applications, 84*(36). Retrieved from https://doi.org/10.1007/s11042-025-20930-y

Kim, T.-Y., & Cho, S.-B. (2018). Web traffic anomaly detection using C-LSTM neural networks. *Expert Systems with Applications, 106*, 66-76. Retrieved from https://doi.org/10.1016/j.eswa.2018.04.004

Kinasih, A., Handayani, A., Ardiansah, J., & Damanhuri, N. (2024). Comparative analysis of decision tree and random forest classifiers for structured data classification in machine learning. *Science in Information Technology Letters, 5*. doi:10.31763/sitech.v5i2.1746

Kuhn, T. S. (1970). *The Structure of Scientific Revolutions.* Chicago, London: University of Chicago Press.

Kumar, Y., Koul, A., & Singh, C. (2022). A Deep Learning Approaches in Text-To-Speech System: A Systematic Review and Recent Research Perspective. *Multimedia Tools and Applications*, Online only (no range, est. 40 pp.). Retrieved from https://doi.org/10.1007/s11042-022-13943-4

Laith, A.-S., Daniel, C.-B., Kelly, A., & Buss., D. M. (2015). Human Emotions: An Evolutionary Psychological Perspective. *Emotion Review, 8*(2), 173–186. Retrieved from https://doi.org/10.1177/1754073914565518

Le, X.-H., Ho, H., Lee, G., & Jung, S. (2019). Application of Long Short-Term Memory (LSTM) Neural Network for Flood Forecasting. *Water, 1387*, Article 1387 (11 pp.). Retrieved from https://doi.org/10.3390/w11071387

Li, H., Rajbahadur, G. K., Lin, D., Bezemer, C. -P., & Jiang, Z. M. (2024). Keeping Deep Learning Models in Check: A History-Based Approach to Mitigate Overfitting. *IEEE Access, 12*, 70676-70689. doi:10.1109/ACCESS.2024.3402543

Li, X., & Akagi, M. (2019). Improving multilingual speech emotion recognition. *JAIST Repository (thesis/report), 110*, 1-12. Retrieved from http://hdl.handle.net/10119/17071

Lieberman, P. (1984). *The Biology and Evolution of Language.* Cambridge, Mass.: Harvard University Press.

Lieck, R., Moss, F. C., & Rohrmeier, M. (2020). The Tonal Diffusion Model. *Transactions of the International Society for Music Information Retrieval, 1*(3), 153–166. Retrieved from https://doi.org/10.5334/tismir.46

Looney, C. G., & Dascalu, S. (2007). A Simple Fuzzy Neural Network. *Conference: Proceedings of the ISCA 20th International Conference on Computer Applications in Industry and Engineering*, (pp. 12-16). San Francisco, California, USA.

Lu, J., Lu, X., Wang, Y., Zhang, H., Han, L., Zhu, B., & Wang, B. (2025). Comparison between logistic regression and machine learning algorithms on prediction of noise-induced hearing loss and investigation of SNP loci. *Scientific Reports, 15*(1), 15361. Retrieved from https://doi.org/10.1038/s41598-025-00050-1

Lumley, M. A., Cohen, J. L., Borszcz, G. S., Cano, A., Radcliffe, A. M., Porter, L. S., . . . Keefe, F. J. (2011). Pain and emotion: a biopsychosocial review of recent research. *Journal of Clinical Psychology, 67*(9), 942-968. Retrieved from https://doi.org/10.1002/jclp.20816

Madmoni, L., Tibor, S., Nelken, I., & Rafaely, B. (2021). The Effect of Partial Time-Frequency Masking of the Direct Sound on the Perception of Reverberant Speech. *IEEE/ACM Transactions on Audio, Speech, and Language Processing, 29*, 2037-2047. doi:10.1109/TASLP.2021.3084742

Manamela, P., Manamela, M., Modipa, T., Joseph, T. S., & Billson, T. M. (2018). The Automatic Recognition of Sepedi Speech Emotions Based on Machine Learning Algorithms. *International Conference on Advances in Big Data, Computing and Data Communication Systems (icABCD 2018)*, (pp. 1-7). Durban, South Africa. doi:10.1109/ICABCD.2018.8465403

Mienye, I. D., & Sun, Y. (2022). A Survey of Ensemble Learning: Concepts, Algorithms, Applications, and Prospects. *IEEE Access, 10*, 99129-99149. doi:10.1109/ACCESS.2022.3207287

Mishra, R., Reddy, G., Sandesh, Y., & Pathak, H. (2021, April). The Understanding of Deep Learning: A Comprehensive Review. *Mathematical Problems in Engineering*, 1-15. Retrieved from https://doi.org/10.1155/2021/5548884

Mohammad, S. M., & Turney, P. D. (2013). Crowdsourcing a Word-Emotion Association Lexicon. *Computational Intelligence, 3*(29), 435-465. Retrieved from https://doi.org/10.1111/j.1467-8640.2012.00460.x

Mondal, A., & Gokhale, S. S. (2020). Mining Emotions on Plutchik's Wheel. *IEEE Xplore*. Retrieved from https://doi.org/10.1109/SNAMS52053.2020.9336534

Muaidi, H., Al-Ahmad, A., Khdoor, T., Alqrainy, S., & Alkoffash, M. (2014). Arabic Audio News Retrieval System Using Dependent Speaker Mode, Mel Frequency Cepstral Coefficient and Dynamic Time Warping Techniques. *Research Journal of Applied Sciences, Engineering and Technology, 7*(24), 5082–5097. Retrieved from https://doi.org/10.19026/rjaset.7.903

Murray, N., & Perronnin, F. (2014). Generalized Max Pooling. *IEEE Computer Society*, 2473 - 2480. doi:10.1109/CVPR.2014.317

Naderi, N., & Nasersharif, B. (2023). Cross Corpus Speech Emotion Recognition using transfer learning and attention-based fusion of Wav2Vec2 and prosody features. *Knowledge-Based Systems, 277*. Retrieved from https://doi.org/10.1016/j.knosys.2023.110814

Naik, S. S., Bhatikar, G., & Gaude, U. (2021, June). Analysis of Best Algorithm for Noise Reduction in Podcasting. *International Journal of Scientific Research in Science and Technology, 8*(3), 246-250. Retrieved from https://doi.org/10.32628/IJSRST218342

Nema, B. M., & Abdul-Kareem, A. A. (2017). Preprocessing Signal for Speech Emotion Recognition. *Al-Mustansiriyah Journal of Science, 3*(28), 157. Retrieved from https://doi.org/10.23851/mjs.v28i3.48

Norval, M., & Wang, Z. (2022, December). Creation of an Afrikaans Speech Corpus for Speech Emotion Recognition. *2nd International Conference on Robotics, Automation and Artificial Intelligence (RAAI)*, 193-196. doi:10.1109/RAAI56146.2022.10092988

Norval, M., & Wang, Z. (2024). Hybrid Architecture and pre-processing for Speech Emotion Recognition. *2024 International Conference on Advanced Mechatronic Systems (ICAMechS)*, (pp. 31-36). Shiga, Japan. doi:10.1109/ICAMechS63130.2024.10818817

Norval, M., & Wang, Z. (2024). Speech Emotion Recognition using Hybrid Architectures. *International Journal of Computing*, 1-10. Retrieved from https://doi.org/10.47839/ijc.23.1.3430

Norval, M., Wang, Z., & Sun, Y. (2024). Synthetic Speech Data Generation Using Generative Adversarial Networks. *International Conference on Cloud Computing and Computer Networks.* (pp. 117-126). Singapore, Singapore: Springer Nature Switzerland. doi:10.1007/978-3-031-47100-1_11

Nwankpa, C., Ijomah, W., Gachagan, A., & Marshall, S. (2018). Activation Functions: Comparison of trends in Practice and Research for Deep Learning. *arXiv*. Retrieved from https://doi.org/10.48550/arxiv.1811.03378

Obaido, G., Mienye, I. D., Egbelowo, O. F., Emmanuel, I. D., Ogunleye, A., Ogbuokiri, B., . . . Aruleba, K. (2024). Supervised machine learning in drug discovery and development: Algorithms, applications, challenges, and prospects,. *Machine Learning with Applications, 17*, 100576. Retrieved from https://doi.org/10.1016/j.mlwa.2024.100576

Oberst, U. (2007). The Fast Fourier Transform. *SIAM Journal on Control and Optimization, 46*(2), 496–540. Retrieved from https://doi.org/10.1137/060658242

Papakostas, M., Spyrou, E., Giannakopoulos, T., Siantikos, G., Sgouropoulos, D., Mylonas, P., & Makedon, F. (2017). Deep Visual Attributes vs. Hand-Crafted Audio Features on Multidomain Speech Emotion Recognition. *Computation, 5*(2), 16.

Patrick, M. K., Adekoya, A. F., Mighty, A. A., & Edward, B. Y. (2022). Capsule networks – a survey. *Journal of King Saud University, 34*, 1295–1310. Retrieved from https://www.sciencedirect.com/science/article/pii/S1319157819309322

Pawan, S. J., & Rajan, J. (2022). Capsule networks for image classification: A review. *Neurocomputing, 509*, 102-120. Retrieved from https://doi.org/10.1016/j.neucom.2022.08.073

Pepino, L., Riera, P., & Ferrer, L. (2021). Emotion Recognition from Speech Using Wav2vec 2.0 Embeddings. *CoRR, abs/2104.03502*. Retrieved from https://arxiv.org/abs/2104.03502

Petrushin, V. (2000). Emotion in Speech: Recognition and Application to Call Centers. *Proceedings of Artificial Neural Networks in Engineering*, 151–158.

Poda, P., Tamtaoui, A., & Bouyakhf, E. H. (2009). Designing DCT Source Significance Information for Efficient and Robust Transmission. *Contemporary Engineering Sciences, V, 2*(2), 67-84. Retrieved from http://www.m-hikari.com/ces/ces2009/ces1-4-2009/podaCES1-4-2009.pdf

Pohlmann, K. (2011). *Principles of Digital Audio, Sixth Edition (Digital Video/Audio) 6th Edition.* New York: McGraw-Hill.

Pothuganti, S. (2018). Review on over-fitting and under-fitting. *nternational Journal of Advanced Research in Electrical Electronics and Instrumentation Engineering, 7*(9), 3692-3695. doi:10.15662/IJAREEIE.2018.0709015.

Prabakaran, D., & Sriuppili, S. (2021). Speech Processing: MFCC Based Feature Extraction Techniques- an Investigation. *Journal of Physics: Conference Series, 1717*, 012009. doi:10.1088/1742-6596/1717/1/012009

Probyn, M. (2006). Language and Learning Science in South Africa. *Language and Education, 20*(5), 391–414. Retrieved from https://doi.org/10.2167/le554.0

Qazi, H., & Kaushik, B. N. (2020, Jun). A Hybrid Technique Using CNN+LSTM for Speech Emotion Recognition. *International Journal of Engineering and Advanced Technology, 9*(5), 343–347. Retrieved from https://doi.org/10.35940/ijeat.e1027.069520

Ramirez-Quintana, J. A., Macias-Macias, J. M., Ramirez-Alonso, G., Chacon-Murguia, M. I., & Corral-Martinez, L. F. (2023). A novel Deep Capsule Neural Network for Vowel Imagery patterns from EEG signals. *Biomedical Signal Processing and Control, 81*, 104500. Retrieved from https://doi.org/10.1016/j.bspc.2022.104500

Rather, I. H., Kumar, S., & Gandomi, A. H. (2024). Breaking the data barrier: a review of deep learning techniques for democratizing AI with small datasets. *Artificial Intelligence Review, 57*(9), 226. Retrieved from https://doi.org/10.1007/s10462-024-10859-3

Rehman, A., Liu, Z., Wu, M., Cao, W., & Jiang, C. (2022). Real-time Speech Emotion Recognition Based on Syllable-Level Feature Extraction. Retrieved from https://doi.org/10.48550/arXiv.2204.11382

Saha, S. (2018, Dec 15). *A Comprehensive Guide to Convolutional Neural Networks—the ELI5 Way*. Retrieved from Towards Data Science: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Salehin, I., & Kang, D.-K. (2023). A Review on Dropout Regularization Approaches for Deep Neural Networks within the Scholarly Domain. *Electronics, 12*(14), 3106. doi:10.3390/electronics12143106

Salminen, J., Mustak, M., Sufyan, M., & Jansen, B. J. (2023). How can algorithms help in segmenting users and customers? A systematic review and research agenda for algorithmic customer segmentation. *Journal of Marketing Analytics, 11*(4), 677-692. doi:https://doi.org/10.1057/s41270-023-00235-5

Services, A. W. (n.d.). *What Is Overfitting? - Overfitting in Machine Learning Explained - AWS*. Retrieved April 29, 2026, from Amazon Web Services: https://aws.amazon.com/what-is/overfitting

Shahin, I. H. (2022). Novel dual-channel long short-term memory compressed capsule networks for emotion recognition. *Expert Systems with Applications, 188*, 116080. Retrieved from https://www.sciencedirect.com/science/article/pii/S0957417421014172

Shakil, M., FA, F., NK, P., SMHS, I., ASM, M., MA, R., & HA., K. (2025). Speech Emotion Recognition Using Deep Learning-Based Ensemble Learning and Feature Fusion. *Journal of Imaging*, 273. doi:https://doi.org/10.3390/jimaging11080273

SHARMA, S. (2017, Sept 6). *Activation Functions in Neural Networks*. Retrieved from Medium. Towards Data Science: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

Shen, J., & Pang, R. (2017, 12 19). *Tacotron 2: Generating Human-like Speech from Text*. Retrieved from https://ai.googleblog.com/2017/12/tacotron-2-generating-human-like-speech.html

Shen, J., Pang, R., Weiss, R. J., Schuster, M., Jaitly, N., Yang, Z., . . . Wu, Y. (2017). Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions. *arXiv:1712.05884*. Retrieved from https://doi.org/10.48550/arXiv.1712.05884

Shi, X., Wang, T., Wang, L., Liu, H., & Yan, N. (2019). Hybrid Convolutional Recurrent Neural Networks Outperform CNN and RNN in Task-state EEG Detection for Parkinson's Disease., (pp. 939-944). Lanzhou, China. doi:10.1109/APSIPAASC47483.2019.9023190

Singh, A., & Ghangas, S. (2016, Aug). SPEAKER RECOGNITION USING MFCC AND DELTADELTA MFCC AND CLASSIFICATION USING ARTIFICIAL. *International Journal of Advance Research in Science and Engineering, 5*(8). Retrieved from http://ijarse.com/images/fullpdf/1472553022_P518-819.pdf

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *The Journal of Machine Learning Research, 15*(1), 1929−1958. Retrieved from https://dl.acm.org/doi/10.5555/2627435.2670313

Staudemeyer, R. C., & Morris, E. R. (2019). Understanding LSTM -- a tutorial into Long Short-Term Memory Recurrent Neural Networks. *arXiv*. Retrieved from https://arxiv.org/abs/1909.09586

Story, B. H. (2019). History of Speech Synthesis. *The Routledge Handbook of Phonetics*, 9–33. Retrieved from https://doi.org/10.4324/9780429056253-2

Sun, J. (2019). Research on Vocal Sounding Based on Spectrum Image Analysis. *EURASIP Journal on Image and Video Processing*. Retrieved from https://doi.org/10.1186/s13640-018-0397-0

Sung, M., Kim, J., & Kang, J.-M. (2023). Probabilistic Classification Method of Spiking Neural Network Based on Multi-Labeling of Neurons. *Mathematics, 11*(5), 1224. Retrieved from https://doi.org/10.3390/math11051224

Suttles, J., & Ide, N. (2013). Distant Supervision for Emotion Classification with Discrete Binary Values. (pp. 121–136). Springer, Berlin, Heidelberg. Retrieved from https://doi.org/10.1007/978-3-642-37256-8_11

Szandała, T. (2020). Review and Comparison of Commonly Used Activation Functions for Deep Neural Networks. *arXiv*, 12 pp. (est.).

Szwoch, M., & Szwoch, W. (2015). Emotion Recognition for Affect Aware Video Games. *Image Processing & Communications Challenges 6, 313*, 227–36. Retrieved from https://doi.org/10.1007/978-3-319-10662-5_28

T., A., Mustaqeem, & S., K. (2020). Deep-Net: A Lightweight CNN-Based Speech Emotion Recognition System Using Deep Frequency Features. *Sensors, 20*(18), 5212. Retrieved from https://doi.org/10.3390/s20185212

Tamulevičius, G., Korvel, G., Yayak, A. B., Treigys, P., Bernatavičienė, J., & Kostek, B. (2020). A Study of Cross-Linguistic Speech Emotion Recognition Based on 2D Feature Spaces. *Electronics, 9*(10), 1725. Retrieved from https://doi.org/10.3390/electronics9101725

Tao, S., Todo, Y., Zheng, T., Li, B., Zhang, Z., & Inoue, R. (2022). A novel artificial visual system for motion direction detection in grayscale images. *Mathematics, 10*, 2975. Retrieved from https://doi.org/10.3390/math10162975

Thanapattheerakul, T., Mao, K., Amoranto, J., & Chan, J. H. (2018). Emotion in a Century: A Review of Emotion Recognition. *Proceedings of the 10th International Conference on Advances in Information Technology*, *17*, pp. 1-8. New York, United States. Retrieved from https://doi.org/10.1145/3291280.3291788

Theng, D., & Bhoyar, K. K. (2024). Feature selection techniques for machine learning: a survey of more than two decades of research. *Knowledge and Information Systems, 66*(3), 1575-1637. Retrieved from https://doi.org/10.1007/s10115-023-02010-5

Tian, Q., Wan, X., & Liu, S. (2018). Generative Adversarial Network based Speaker Adaptation for High Fidelity WaveNet Vocoder. *Computer and information sciences*. Retrieved from https://doi.org/10.48550/arxiv.1812.02339

van Dijk, W. W., & Zeelenberg, M. (2002). *26*(4), 321-331. doi:https://doi.org/10.1023/A:1022823221146

Vaseghi, S. V. (1996). *Advanced Digital Signal Processing and Noise Reduction, Second Edition.* John Wiley & Sons Incorporated. Retrieved from https://doi.org/10.1002/0470841621.ch11

Visa, S., Ramsay, B., Ralescu, A., & Knaap, E. v. (2011, April 16-17). Confusion Matrix-based Feature Selection. *Artificial Intelligence and Cognitive Science*. Retrieved from https://www.researchgate.net/publication/220833270_Confusion_Matrix-based_Feature_Selection

Vryzas, N., Kotsakis, R., Liatsou, A., Dimoulas, C. A., & Kalliris, G. (2018). Speech emotion recognition for performance interaction. *Journal of the Audio Engineering Society, 6*(66), 457-467. Retrieved from https://m3c.web.auth.gr/research/aesdd-speech-emotion-recognition/

Wang, H., Duentsch, I., Guo, G., & Khan, S. A. (2023). Special issue on small data analytics. *International Journal of Machine Learning and Cybernetics, 14*(1), 1-2. Retrieved from https://doi.org/10.1007/s13042-022-01699-0

Wang, Y., Boumadane, A., & Heba, A. (2021). A fine-tuned wav2vec 2.0. *CoRR, abs/2111.02735*. Retrieved from https://arxiv.org/abs/2111.02735

Wei, Y., Deng, Y., Sun, C., Lin, M., Jiang, H., & Peng, Y. (2024). Deep learning with noisy labels in medical prediction problems: a scoping review. *Journal of the American Medical Informatics Association, 31*(7), 1596-1607. Retrieved from https://doi.org/10.1093/jamia/ocae108

Welch, G., & Bishop, G. (1995). *An Introduction to the Kalman Filter*. Retrieved 11 4, 2022, from https://cs.unc.edu/~welch/media/pdf/kalman_intro.pdf

Wilson, V. (2014). Research Methods: Triangulation. *Evidence Based Library and Information Practice, 9*(1), 74. Retrieved from https://doi.org/10.18438/b8ww3x

Wu, X., Liu, S., Cao, Y., Li, X., Yu, J., dai, D., . . . S. Hu. (2019). Speech Emotion Recognition Using Capsule. Retrieved from https://doi.org/10.1109/icassp.2019.8683163

Yamamoto, R., Song, E., & Kim, J. M. (2020). Parallel Wavegan: A Fast Waveform Generation Model Based on Generative Adversarial Networks with Multi-Resolution Spectrogram., (pp. 6199-6203). Barcelona, Spain. doi:10.1109/ICASSP40776.2020.9053795

Ying, X. (2019, Feb). An Overview of Overfitting and Its Solutions. *Journal of Physics: Conference Series, 1168*(2), 022022. doi:10.1088/1742-6596/1168/2/022022

Yu, D., & Li, D. (2015). *Automatic Speech Recognition.* London: Springer London.

Zehra, W., Javed, A. R., Jalil, Z., Gadekallu, T. R., & Kahn, H. U. (2021). Cross corpus multi-lingual speech emotion recognition using ensemble learning. *Complex & Intelligent Systems, 7*, 1845–1854. Retrieved from https://doi.org/10.1007/s40747-020-00250-4

Zhang, J., & Zong, C. (2015). Deep Neural Networks in Machine Translation: An Overview. *IEEE Intelligent Systems, 30*(5), 16-25. Retrieved from https://doi.org/10.1109/mis.2015.69

Zhao, D., Li, Y., Zeng, Y., Wang, J., & Zhang, Q. (2022). Spiking CapsNet: A spiking neural network with a biologically plausible routing rule between capsules. *Information Sciences, 610*, 1-13. Retrieved from https://doi.org/10.1016/j.ins.2022.07.152

Zhao, J., Mao, X., & Chen, L. (2019). Speech emotion recognition using deep 1D & 2D CNN LSTM networks. *Biomedical Signal Processing and Control, 47*, 312-323. Retrieved 2 2, 2023, from https://doi.org/10.1016/j.bspc.2018.08.035

Zhou, C., Sun, C., Liu, Z., & Lau, F. C. (2015). A C-LSTM Neural Network for Text Classification. *arXiv*. Retrieved from https://doi.org/10.48550/arXiv.1511.08630

APPENDICES

Appendix A — Full Experimental Results

This appendix presents the complete results from the 10-run experimental protocol described in Sections 4.4 and 3.3. Each table reports per-run performance metrics (accuracy, macro F1, weighted F1, balanced accuracy, and validation loss) followed by the mean and standard deviation across all ten runs. The filter comparison results in Section A.1 evaluate seven pre-processing strategies on the multilingual dataset, while Section A.2 reports baseline performance on the Afrikaans corpus and the EmoDB corpus, respectively. All experiments used identical model architecture, hyperparameters, and random seeds across the ten runs to isolate the effect of the variable under investigation.

A.1 Chapter 4: Filter Comparison Results

A.1.1 Generic Noise Removal (noise_remove_generic)

Run	Best Accuracy	Macro F1	Weighted F1	Balanced Acc	Loss
1	0.6130	0.5977	0.6186	0.6130	1.1123
2	0.6130	0.5700	0.5988	0.6130	1.1894
3	0.5957	0.5693	0.5985	0.5957	1.1270
4	0.6261	0.5997	0.6233	0.6261	1.1087
5	0.6087	0.5936	0.6200	0.6087	1.1243
6	**0.6304**	**0.6056**	**0.6284**	**0.6304**	**1.0854**
7	0.5870	0.5248	0.5677	0.5870	1.1020
8	0.6000	0.5578	0.5919	0.6000	1.1839
9	0.5870	0.5564	0.5873	0.5870	1.1742
10	0.6261	0.5900	0.6263	0.6261	1.0952
Mean	**0.6087**	**0.5765**	**0.6061**	**0.6087**	**1.1302**
Std Dev	0.0160	0.0255	0.0203	0.0160	0.0383

A.1.2 Bandpass Noise Removal (noise_remove_bandpass)

Run	Accuracy	Macro F1	Weighted F1	Balanced Acc	Loss
1	0.4950	0.4950	0.5131	0.4950	1.2988
2	**0.5500**	**0.5281**	**0.5437**	**0.5500**	**1.4623**
3	0.5100	0.4875	0.5045	0.5100	1.3606
4	0.4600	0.4427	0.4459	0.4600	1.4633
5	0.4350	0.4065	0.4082	0.4350	1.4516
6	0.1500	0.0326	0.0391	0.1500	2.0584
7	0.5000	0.4994	0.5031	0.5000	1.4073
8	0.4800	0.4441	0.4684	0.4800	1.4124
9	0.4600	0.4522	0.4615	0.4600	1.4457
10	0.4650	0.4409	0.4448	0.4650	1.4235
Mean	**0.4505**	**0.4229**	**0.4332**	**0.4505**	**1.4784**
Std Dev	0.1104	0.1418	0.1441	0.1104	0.2101

A.1.3 Silence Removal (silence_remove)

Run	Accuracy	Macro F1	Weighted F1	Balanced Acc	Loss
1	0.7035	0.7037	0.7050	0.7035	0.8520
2	0.6985	0.6954	0.6944	0.6985	0.9561
3	0.6784	0.6760	0.6767	0.6784	1.0079
4	0.6784	0.6524	0.6661	0.6784	0.8674
5	0.6935	0.6898	0.6922	0.6935	1.1038
6	0.6482	0.6492	0.6524	0.6482	0.9716
7	**0.7236**	**0.7240**	**0.7266**	**0.7236**	**0.8313**
8	0.6231	0.5947	0.6164	0.6231	1.0007
9	0.6985	0.6842	0.6936	0.6985	0.8826
10	0.6935	0.6814	0.6945	0.6935	0.8932
Mean	**0.6839**	**0.6751**	**0.6818**	**0.6839**	**0.9367**
Std Dev	0.0291	0.0359	0.0308	0.0291	0.0860

A.1.4 Spectral Gating (spectral_gating)

Run	Accuracy	Macro F1	Weighted F1	Balanced Acc	Loss
1	0.4450	0.4115	0.4274	0.4450	1.4870
2	0.5150	0.4951	0.5002	0.5150	1.3990
3	0.4450	0.4590	0.4510	0.4450	1.4490
4	0.4750	0.4047	0.4136	0.4750	1.4052
5	0.5050	0.5063	0.5015	0.5050	1.4865
6	0.4700	0.4681	0.4706	0.4700	1.4200
7	0.4150	0.3479	0.3696	0.4150	1.4948
8	0.4950	0.4749	0.4747	0.4950	1.4761
9	**0.5450**	**0.5100**	**0.5207**	**0.5450**	**1.4280**
10	0.4850	0.4815	0.4831	0.4850	1.4248
Mean	**0.4795**	**0.4559**	**0.4612**	**0.4795**	**1.4470**
Std Dev	0.0382	0.0521	0.0463	0.0382	0.0364

A.1.5 Spectral Subtraction (spectral_subtract)

Run	Accuracy	Macro F1	Weighted F1	Balanced Acc	Loss
1	0.4950	0.4642	0.4777	0.4950	1.3650
2	0.4450	0.4054	0.3986	0.4450	1.4405
3	0.1950	0.0408	0.0636	0.1950	2.0538
4	0.5050	0.4755	0.4992	0.5050	1.3657
5	0.5100	0.4776	0.4872	0.5100	1.3227
6	**0.5450**	**0.5246**	**0.5319**	**0.5450**	**1.2771**
7	0.5150	0.4879	0.4960	0.5150	1.4096
8	0.4700	0.4684	0.4744	0.4700	1.2879
9	0.4750	0.4597	0.4493	0.4750	1.3918
10	0.5350	0.4989	0.5220	0.5350	1.4090
Mean	**0.4690**	**0.4303**	**0.4400**	**0.4690**	**1.4323**
Std Dev	0.1009	0.1402	0.1375	0.1009	0.2248

A.1.6 Kalman Filter (Variant 1)(kalman_1)

Run	Accuracy	Macro F1	Weighted F1	Balanced Acc	Loss
1	0.6100	0.5938	0.6079	0.6100	1.1899
2	0.5600	0.5571	0.5666	0.5600	1.1031
3	0.6150	0.5980	0.6102	0.6150	1.0444
4	0.6150	0.6082	0.6157	0.6150	1.0560
5	**0.6900**	**0.6639**	**0.6847**	**0.6900**	**1.0268**
6	0.6200	0.6052	0.6181	0.6200	1.1554
7	0.6200	0.5887	0.6113	0.6200	1.0668
8	0.6350	0.6190	0.6365	0.6350	1.1276
9	0.6400	0.6234	0.6368	0.6400	1.1110
10	0.6500	0.6175	0.6429	0.6500	1.0388
Mean	**0.6255**	**0.6075**	**0.6231**	**0.6255**	**1.0920**
Std Dev	0.0331	0.0276	0.0305	0.0331	0.0544

A.1.7 Kalman Filter (Variant 2)(kalman_2)

Run	Accuracy	Macro F1	Weighted F1	Balanced Acc	Loss
1	0.7400	0.7326	0.7408	0.7400	0.7897
2	0.7450	0.7381	0.7473	0.7450	0.8474
3	0.7000	0.6767	0.6949	0.7000	0.8795
4	**0.7850**	**0.7658**	**0.7803**	**0.7850**	**0.7199**
5	0.7150	0.7057	0.7168	0.7150	0.7379
6	0.7400	0.7248	0.7368	0.7400	0.7146
7	0.7400	0.7138	0.7399	0.7400	0.7532
8	0.7300	0.7171	0.7303	0.7300	0.7898
9	0.7400	0.7157	0.7339	0.7400	0.7315
10	0.7100	0.6831	0.7073	0.7100	0.7512
Mean	**0.7345**	**0.7173**	**0.7328**	**0.7345**	**0.7715**
Std Dev	0.0235	0.0259	0.0234	0.0235	0.0552

A.2 Chapter 3: Corpus Baseline Results

A.2.1 Afrikaans Corpus

Run	Accuracy	Macro F1	Weighted F1	Balanced Acc	Loss
1	0.5500	0.5415	0.5419	0.5421	1.5959
2	**0.6100**	**0.6019**	**0.6040**	**0.6102**	**1.6468**
3	0.5950	0.5849	0.5919	0.6008	1.5708
4	0.5450	0.5372	0.5387	0.5405	1.5925
5	0.5400	0.5185	0.5175	0.5545	1.6400
6	0.5900	0.5816	0.5822	0.6007	1.5492
7	0.5550	0.5354	0.5485	0.5441	1.6092
8	0.5800	0.5689	0.5749	0.5791	1.5803
9	0.5950	0.5809	0.5947	0.5846	1.6293
10	0.5650	0.5486	0.5635	0.5529	1.6159
Mean	**0.5725**	**0.5599**	**0.5658**	**0.5709**	**1.6030**
Std Dev	0.0234	0.0258	0.0270	0.0258	0.0297

A.2.2 EmoDB

Run	Accuracy	Macro F1	Weighted F1	Balanced Acc	Loss
1	**0.7174**	**0.6319**	**0.6988**	**0.6208**	**1.0435**
2	0.6522	0.5867	0.6449	0.5835	1.1030
3	0.6957	0.6637	0.6981	0.6599	0.9606
4	0.7174	0.6137	0.6894	0.6218	1.2902
5	0.6848	0.5484	0.6387	0.5647	1.0587
6	0.6413	0.5703	0.6382	0.5756	1.2037
7	0.6848	0.6177	0.6794	0.6122	1.1004
8	0.7174	0.6544	0.7051	0.6512	1.0417
9	0.6630	0.5381	0.6285	0.5431	1.0083
10	0.6739	0.6225	0.6716	0.6160	1.0987
Mean	**0.6848**	**0.6048**	**0.6693**	**0.6049**	**1.0909**
Std Dev	0.0262	0.0405	0.0277	0.0356	0.0907

Appendix B — Source Code

B.1 Feature Extraction

Listing B.1: Feature extraction from WAV audio files. Loads cached training data via joblib if available; otherwise, extracts MFCC, chroma, mel-spectrogram, spectral contrast, and tonnetz features using librosa, then stratifies an 80/20 train/test split.

```
from .settings import _CLASS_LABELS, _DATA_PATH, mean_signal_length

import os
import sys
import joblib
import numpy as np
import librosa
import soundfile
from typing import Tuple
from sklearn.model_selection import train_test_split


def load_data(load_from_file, name):
    """Load cached features"""
    save_dir = os.path.join(os.getcwd(), 'training_data')
    os.makedirs(save_dir, exist_ok=True)

    if load_from_file:
        x_train = joblib.load(os.path.join(save_dir, f'x_train_{name}.joblib'))
        x_test = joblib.load(os.path.join(save_dir, f'x_test_{name}.joblib'))
        y_train = joblib.load(os.path.join(save_dir, f'y_train_{name}.joblib'))
        y_test = joblib.load(os.path.join(save_dir, f'y_test_{name}.joblib'))
        num_labels = joblib.load(os.path.join(save_dir, f'num_labels_{name}.joblib'))
    else:
        x_train, x_test, y_train, y_test, num_labels = extract_data()
        joblib.dump(x_train, os.path.join(save_dir, f'x_train_{name}.joblib'))
        joblib.dump(x_test, os.path.join(save_dir, f'x_test_{name}.joblib'))
        joblib.dump(y_train, os.path.join(save_dir, f'y_train_{name}.joblib'))
        joblib.dump(y_test, os.path.join(save_dir, f'y_test_{name}.joblib'))
        joblib.dump(num_labels, os.path.join(save_dir, f'num_labels_{name}.joblib'))

    return x_train, x_test, y_train, y_test, num_labels


def initialise():
    """Suppress TensorFlow logging and clear the console."""
    os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'
    os.environ['TF_FORCE_GPU_ALLOW_GROWTH'] = 'true'
    os.environ['TF_ENABLE_GPU_GARBAGE_COLLECTION'] = 'false'
    os.system('cls' if os.name == 'nt' else 'clear')


def extract_data():
    """Read all WAV files under _DATA_PATH and return an 80/20 split."""
    data, labels = get_data(_DATA_PATH, class_labels=_CLASS_LABELS)
    x_train, x_test, y_train, y_test = train_test_split(
        data, labels, test_size=0.2, random_state=42)
    return (np.array(x_train), np.array(x_test),
            np.array(y_train), np.array(y_test),
            len(_CLASS_LABELS))


def get_data(data_path: str,
             flatten: bool = True,
```

```
             mfcc_len: int = 40,
             class_labels: Tuple = _CLASS_LABELS):
    """Walk each emotion folder under data_path and extract feature vectors."""
    data, labels, names = [], [], []
    cur_dir = os.getcwd()
    os.chdir(data_path)
    file_count = sum(len(files) for _, _, files in os.walk(data_path))
    tempcount = 0

    for i, directory in enumerate(class_labels):
        sys.stderr.write(f"started reading folder {directory}\n")
        os.chdir(directory)
        for filename in os.listdir('.'):
            tempcount += 1
            filepath = os.path.join(os.getcwd(), filename)
            print(f"** {tempcount} / {file_count} *** -> {filepath}")
            feature_vector = extract_feature(
                filepath,
                mfcc=True, chroma=True, mel=True,
                contrast=True, tonnetz=True,
                num_features=40)
            data.append(feature_vector)
            labels.append(i)
            names.append(filename)
        sys.stderr.write(f"ended reading folder {directory}\n")
        os.chdir('..')

    os.chdir(cur_dir)
    return np.array(data), np.array(labels)


def extract_feature(file_name, **kwargs):
    """Extract a concatenated feature vector (MFCC, chroma, mel, contrast, tonnetz)."""
    use_mfcc = kwargs.get("mfcc")
    use_chroma = kwargs.get("chroma")
    use_mel = kwargs.get("mel")
    use_contrast = kwargs.get("contrast")
    use_tonnetz = kwargs.get("tonnetz")
    num_features = kwargs.get("num_features")

    with soundfile.SoundFile(file_name) as sound_file:
        X = sound_file.read(dtype="float32")
        sample_rate = sound_file.samplerate
        s_len = len(X)

        # Pad or trim to mean_signal_length
        if s_len < mean_signal_length:
            pad_len = mean_signal_length - s_len
            pad_rem = pad_len % 2
            pad_len //= 2
            X = np.pad(X, (pad_len, pad_len + pad_rem),
                       'constant', constant_values=0)
        else:
            pad_len = (s_len - mean_signal_length) // 2
            X = X[pad_len:pad_len + mean_signal_length]

        result = np.array([]).reshape(num_features, 0)
        stft = np.abs(librosa.stft(X))

        if use_mfcc:
            mfcc_feat = np.array(librosa.feature.mfcc(
                y=X, sr=sample_rate, n_mfcc=num_features))
            result = np.hstack((result, mfcc_feat))

        if use_chroma:
            chroma_feat = np.array(librosa.feature.chroma_stft(
                S=stft, sr=sample_rate, n_chroma=num_features))
            result = np.hstack((result, chroma_feat))

        if use_mel:
            mel_feat = np.array(librosa.feature.melspectrogram(
                y=X, sr=sample_rate, n_mels=num_features))
```

```
            result = np.hstack((result, mel_feat))

        if use_contrast:
            contrast_feat = np.array(librosa.feature.spectral_contrast(
                S=stft, sr=sample_rate))
            result = np.hstack((result, np.pad(
                contrast_feat, ((0, num_features - 7), (0, 0)),
                'constant', constant_values=0)))

        if use_tonnetz:
            tonnetz_feat = np.array(librosa.feature.tonnetz(
                y=librosa.effects.harmonic(X), sr=sample_rate))
            result = np.hstack((result, np.pad(
                tonnetz_feat, ((0, num_features - 6), (0, 0)),
                'constant', constant_values=0)))

    return result
```

B.2 CLSTM Model

Listing B.2: CLSTM model definition and training. Implements a hybrid Conv1D–LSTM classifier for speech emotion recognition: a 1D convolutional feature extractor feeds a 64-unit LSTM, followed by dense layers with dropout regularisation. Training uses the Adam optimiser with categorical cross-entropy loss and early stopping on validation loss.

```
import matplotlib.pyplot as plt
from tensorflow.keras import Sequential, utils
from tensorflow.keras.layers import (Conv1D, MaxPooling1D, LSTM,
                                     Flatten, Dense, Dropout)
from tensorflow.keras.callbacks import EarlyStopping

MODEL_PATH = "../MULTI_LANGUAGE_JOBLIB/CLSTM"


def conv_lstm_example():
    """Build, train, and evaluate the CLSTM model on cached features."""
    # Load cached features
    x_train, x_test, y_train, y_test, num_labels = load_data(
        load_from_file=True, name='multi_large')

    y_train = utils.to_categorical(y_train)
    y_test = utils.to_categorical(y_test)

    num_timesteps = x_train.shape[1]
    num_features = x_train.shape[2]
    num_classes = y_train.shape[1]

    # Build CLSTM architecture
    model = Sequential([
        Conv1D(128, kernel_size=5, activation='relu',
               input_shape=(num_timesteps, num_features)),
        MaxPooling1D(pool_size=2),
        Dropout(0.5),
        LSTM(64, return_sequences=True),
        Flatten(),
        Dense(32, activation='relu'),
        Dropout(0.5),
        Dense(num_classes, activation='softmax')
    ])

    model.compile(loss='categorical_crossentropy',
                  optimizer='adam',
                  metrics=['accuracy'])
    model.summary()

    # Early stopping to prevent overfitting (see Section 2.5.14.4.1)
    early_stopping = EarlyStopping(
        monitor='val_loss',
        patience=8,
        restore_best_weights=True,
        mode='min')

    # Train the model
    history = model.fit(
        x_train, y_train,
```

```
        batch_size=32,
        epochs=100,
        shuffle=True,
        validation_data=(x_test, y_test),
        callbacks=[early_stopping],
        verbose=1)

    # Evaluate
    loss, acc = model.evaluate(x_test, y_test, verbose=0)
    print(f"Test Accuracy:    {acc * 100:.2f}%")
    print(f"Test Loss:        {loss * 100:.2f}%")
    print(f"Highest Val Acc:  {max(history.history['val_accuracy']) * 100:.2f}%")

    # Plot training curves (once, after training)
    plt.figure(figsize=(12, 4))

    plt.subplot(1, 2, 1)
    plt.plot(history.history['loss'], label='Train')
    plt.plot(history.history['val_loss'], label='Validation')
    plt.title('Loss vs. Epoch')
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
    plt.legend()

    plt.subplot(1, 2, 2)
    plt.plot(history.history['accuracy'], label='Train')
    plt.plot(history.history['val_accuracy'], label='Validation')
    plt.title('Accuracy vs. Epoch')
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy')
    plt.legend()

    plt.tight_layout()
    plt.show()

    print('conv_lstm Done')
```

B.3 DenCaps Model

Listing B.3: DenCaps model with custom Dendritic and Capsule layers, including the full training and evaluation pipeline. The dendritic layer applies multiple segment-wise transformations before a soma-stage aggregation, mimicking biological dendritic computation. The capsule layer applies dynamic routing-by-agreement to encode part-whole relationships. Training uses early stopping on validation loss, followed by precision/recall/F1 evaluation and a confusion matrix over the eight emotion classes.

```
# ================================================================================
# B.3 DenCaps Model (Multilingual - With Class Balancing)
# ================================================================================

import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import tensorflow as tf
from tensorflow.keras import layers, utils
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.callbacks import EarlyStopping
from sklearn.metrics import (precision_score, recall_score, f1_score,
                             confusion_matrix, classification_report)
from sklearn.utils.class_weight import compute_class_weight

MODEL_PATH = "../MULTI_LANGUAGE_JOBLIB/Large"

CLASS_LABELS = ["Anger", "Anticipation", "Disgust", "Fear",
                "Joy", "Sadness", "Surprise", "Trust"]

ALPHA = 0.5   # Softened class weighting factor (consistent with Chapter 5)

# ----------------------------------------------------------------
# Squash activation (Sabour et al., 2017)
# ----------------------------------------------------------------
def squash(x, axis=-1):
    s_squared_norm = tf.reduce_sum(tf.square(x), axis=axis, keepdims=True)
    scale = s_squared_norm / (1.0 + s_squared_norm) / tf.sqrt(s_squared_norm + 1e-7)
    return scale * x

# ----------------------------------------------------------------
# Capsule layer with dynamic routing-by-agreement
# ----------------------------------------------------------------
class CapsuleLayer(layers.Layer):
    def __init__(self, num_capsules, dim_capsule, routings=3, **kwargs):
        super().__init__(**kwargs)
        self.num_capsules = num_capsules
        self.dim_capsule = dim_capsule
        self.routings = routings

    def build(self, input_shape):
        self.kernel = self.add_weight(
            name='capsule_kernel',
            shape=(input_shape[-1], self.num_capsules * self.dim_capsule),
            initializer='glorot_uniform',
            trainable=True)

    def call(self, inputs):
        inputs = tf.reshape(inputs, (-1, inputs.shape[-1]))
```

```
        u = tf.matmul(inputs, self.kernel)
        u = tf.reshape(u, (-1, self.num_capsules, self.dim_capsule))

        b = tf.zeros_like(u[:, :, 0])
        for _ in range(self.routings):
            c = tf.nn.softmax(b, axis=1)
            s = tf.reduce_sum(c[:, :, tf.newaxis] * u, axis=1)
            v = squash(s)
            b = b + tf.reduce_sum(u * v[:, tf.newaxis, :], axis=2)
        return v

    def get_config(self):
        config = super().get_config()
        config.update({
            'num_capsules': self.num_capsules,
            'dim_capsule': self.dim_capsule,
            'routings': self.routings
        })
        return config

# ------------------------------------------------------------------
# Custom Dendritic Layer
# ------------------------------------------------------------------
class DendriticLayer(layers.Layer):
    def __init__(self, units, segments, **kwargs):
        super().__init__(**kwargs)
        self.units = units
        self.segments = segments

    def build(self, input_shape):
        self.segment_weights = []
        self.segment_biases = []
        for i in range(self.segments):
            self.segment_weights.append(self.add_weight(
                name=f'segment_weight_{i}',
                shape=(input_shape[-1], self.units),
                initializer='random_normal',
                trainable=True))
            self.segment_biases.append(self.add_weight(
                name=f'segment_bias_{i}',
                shape=(self.units,),
                initializer='zeros',
                trainable=True))

        self.soma_weight = self.add_weight(
            name='soma_weight',
            shape=(self.units * self.segments, self.units),
            initializer='random_normal',
            trainable=True)
        self.soma_bias = self.add_weight(
            name='soma_bias',
            shape=(self.units,),
            initializer='zeros',
            trainable=True)

    def call(self, inputs):
        outputs = []
        for w, b in zip(self.segment_weights, self.segment_biases):
            x = tf.nn.relu(tf.matmul(inputs, w) + b)
            outputs.append(x)
        concat = tf.concat(outputs, axis=-1)
        soma_output = tf.nn.relu(tf.matmul(concat, self.soma_weight) + self.soma_bias)
        return soma_output
```

```
    def get_config(self):
        config = super().get_config()
        config.update({
            'units': self.units,
            'segments': self.segments
        })
        return config

# ------------------------------------------------------------
# Main DenCaps training function (with class balancing)
# ------------------------------------------------------------
def DenCaps_example():
    """Train and evaluate the DenCaps architecture on cached features (with class balancing)."""
    # Load cached features
    x_train, x_test, y_train_raw, y_test_raw, _ = load_data(
        load_from_file=True, name='multi_large')

    # ====================== CLASS BALANCING ======================
    classes = np.unique(y_train_raw)
    raw_weights = compute_class_weight('balanced', classes=classes, y=y_train_raw)
    class_weight_dict = {cls: 1.0 + ALPHA * (wt - 1.0)
                         for cls, wt in zip(classes, raw_weights)}

    print("Softened class weights (α=0.5):")
    for cls, wt in class_weight_dict.items():
        print(f"  Class {cls} ({CLASS_LABELS[cls]}): {wt:.4f}")

    # One-hot encoding
    y_train = utils.to_categorical(y_train_raw)
    y_test = utils.to_categorical(y_test_raw)

    # Reshape for Conv2D
    in_shape = x_train[0].shape
    x_train = x_train.reshape(x_train.shape[0], in_shape[1], in_shape[0], 1)
    x_test = x_test.reshape(x_test.shape[0], in_shape[1], in_shape[0], 1)

    num_timesteps = x_train.shape[1]
    num_features = x_train.shape[2]
    num_classes = len(CLASS_LABELS)
    input_shape = (num_timesteps, num_features, 1)

    # ====================== MODEL ARCHITECTURE ======================
    inputs = tf.keras.Input(shape=input_shape)
    x = layers.Conv2D(128, (3, 3), activation='relu')(inputs)
    x = layers.Conv2D(32, (3, 3), activation='relu')(x)
    x = layers.MaxPooling2D(pool_size=2)(x)
    x = layers.Dropout(0.5)(x)
    x = layers.Flatten()(x)
    x = DendriticLayer(units=64, segments=2)(x)
    x = Capsule_Layer(num_capsules=10, dim_capsule=32, routings=3)(x)
    x = Dense(32, activation='relu')(x)
    x = Dropout(0.5)(x)
    outputs = Dense(num_classes, activation='softmax')(x)

    model = tf.keras.Model(inputs=inputs, outputs=outputs)
    model.compile(optimizer='adam',
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])
    model.summary()

    # ====================== TRAINING ======================
    early_stopping = EarlyStopping(
```

```
        monitor='val_loss',
        patience=8,
        restore_best_weights=True,
        mode='min')

    history = model.fit(
        x_train, y_train,
        batch_size=32,
        epochs=30,
        shuffle=True,
        validation_data=(x_test, y_test),
        callbacks=[early_stopping],
        class_weight=class_weight_dict,   # ← CLASS BALANCING ENABLED
        verbose=1)

    # ====================== EVALUATION ======================
    loss, acc = model.evaluate(x_test, y_test, verbose=0)
    print(f"Test Accuracy:    {acc * 100:.2f}%")
    print(f"Test Loss:        {loss * 100:.2f}%")
    print(f"Highest Val Acc:  {max(history.history['val_accuracy']) * 100:.2f}%")

    # Training curves, classification report, and confusion matrix (same as before)
    # ... [keep your existing plotting and metrics code here] ...

    y_pred = model.predict(x_test)
    y_pred_classes = np.argmax(y_pred, axis=1)
    y_true = np.argmax(y_test, axis=1)

    print("\nClassification Report:\n")
    print(classification_report(y_true, y_pred_classes, target_names=CLASS_LABELS))

    conf_matrix = confusion_matrix(y_true, y_pred_classes)
    print("\nConfusion Matrix:\n", conf_matrix)

    plt.figure(figsize=(10, 8))
    sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',
                xticklabels=CLASS_LABELS, yticklabels=CLASS_LABELS)
    plt.xlabel('Predicted Labels')
    plt.ylabel('True Labels')
    plt.title('Confusion Matrix')
    plt.tight_layout()
    plt.show()
```