COMPARATIVE ANALYSIS OF SPEECH RECOGNITION ON DEEP FAKE AUDIO USING DEEP LEARNING ALGORITHMS

NURUL SAKINAH BINTI SUHAIRI

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

COMPARATIVE ANALYSIS OF SPEECH RECOGNITION ON DEEP FAKE AUDIO USING DEEP LEARNING ALGORITHMS

NURUL SAKINAH BINTI SUHAIRI

This report is submitted in partial fulfillment of the requirements for the Bachelor of Computer Science (Computer Networking) with Honors.

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

2024

DECLARATION

I hereby declare that this project report entitled

[COMPARATIVE ANALYSIS OF SPEECH RECOGNITION ON DEEP FAKE AUDIO USING DEEP LEARNING ALGORITHMS]

is written by me and is my own effort and that no part has been plagiarized without citations.

STUDENT : NURUL SAKINAH BINTI SUHAIRI Date : 2/09/2024

I hereby declare that I have read this project report and found this project report is sufficient in term of the scope and quality for the award of Bachelor of Computer Science (Computer Networking) with Honors.

SUPERVISOR : TS. MOHD HAKIM BIN ABDUL HAMID Date : _____

DEDICATION

This final year project is dedicated to my beloved parents, who never failed to give me moral support, financial support and giving all my needs during my degree. All the encouragement words that push for tenacity ring in my ears help me to never give up until now.

I also dedicated this dissertation to my supervisor, Ts. Mohd Hakim Bin Abdul Hamid, who taught me how does final year project should be sow the seeds. This final year project would not have been possible without guidance from my supervisor and always provide suggestions until my project finishes.

Lastly, this dedication section also needs to be thanked to my friends. They encourage me to keep on track and make sure that I do not neglect my responsibilities as a student.

ACKNOWLEDGMENT

I would like to acknowledge and give my big thanks to my supervisor, Ts. Mohd Hakim Bin Abdul Hamid, who made this work possible and successful. His guidance and advice carried me through all the stages of writing my project. Knowledge based on his experience also helps me to proceed with flow. I am grateful for his assistance while I was conducting my research.

I would also like to thank my members who always inspire me to keep working on this report. I felt gratitude with them who were always willing to assist and share their ideas and knowledge even when they were busy with their own work and studies.

Most importantly, I'd like to express my heartfelt gratitude to my family for their loving support, patience, and encouragement. Their prayers and good wishes helped me stay strong, especially during difficult times. I will be eternally grateful to them.

ABSTRACT

This study presents a comprehensive comparative analysis of speech recognition on deep fake audio using multiple deep learning algorithms. Deep fake technology is becoming more common, so, it is needed to create reliable techniques for identifying and analyzing these deep fake audios. This audio presents serious security, privacy and authenticity verification concerns. In order to accurately distinguish between real and fake audio, this research explores the effectiveness of various deep learning model such as, Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), Generative Adversarial Network (GAN) and Gated Recurrent Units (GRU). Each model is evaluated using a range of performance metrics such as accuracy, F1-score and recall, across the dataset. Furthermore, advanced feature selection methods like Principal Component Analysis (PCA), Recursive Feature Elimination (RFE) and Mutual Information (MI) are used in effort to boost the model's ability for generalization and boost classification performance. This study guarantees a comprehensive and objective evaluation of the model's performance by utilizing Synthetic Minority Over-sampling Technique (SMOTE) to handle data imbalance and Stratified K-fold Cross Validation. The objective of this research are to analyze the process of deep fake audio dataset using various deep learning algorithm, to identify which model produce the best performance through evaluation metrics and to propose and enhanced deep learning method for deep fake audio dataset.

ABSTRAK

Kajian ini membentangkan analisis perbandingan yang lengkap mengenai pengecaman pertuturan pada audio ‘deep fake’ menggunakan pelbagai algoritma ‘Deep Learning’. Teknologi ‘deep fake’ semakin berkembang, oleh itu, terdapat keperluan untuk mencipta teknik yang boleh dipercayai bagi mengenal pasti dan menganalisis audio ‘deep fake’ ini. Audio ini menimbulkan kebimbangan yang serius terhadap keselamatan, privasi, dan pengesahan keaslian. Untuk membezakan dengan tepat antara audio sebenar dan palsu, kajian ini meneroka keberkesanan pelbagai model ‘Deep Learning’ seperti ‘Convolutional Neural Network’ (CNN), ‘Recurrent Neural Network’ (RNN), ‘Long Short-Term Memory’ (LSTM), ‘Generative Adversarial Network’ (GAN), dan ‘Gated Recurrent Unit’ (GRU). Setiap model yang dinilai menggunakan pelbagai metrik prestasi seperti ‘accuracy’, ‘F1-score’, dan ‘recall’ melalui data set. Selain itu, kaedah pemilihan ciri lanjutan seperti ‘Principal Component Analysis’ (PCA), ‘Recursive Feature Elimination’ (RFE), dan ‘Mutual Information’ (MI) digunakan dalam usaha untuk meningkatkan keupayaan model dalam generalisasi dan meningkatkan prestasi klasifikasi. Kajian ini memastikan penilaian prestasi model yang lengkap dan objektif dengan menggunakan Teknik Pengambilan Sampel Berlebihan Minoriti Sintetik (SMOTE) untuk menangani ketidakseimbangan data dan Validasi Silang Stratified K-fold. Objektif kajian ini adalah untuk menganalisis proses set data audio ‘deep fake’ menggunakan pelbagai algoritma pembelajaran mendalam, mengenal pasti model yang menghasilkan prestasi terbaik melalui metrik penilaian, dan mencadangkan kaedah pembelajaran mendalam yang dipertingkatkan untuk set data audio ‘deep fake.

Table of Contents

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
GAN	Generative Adversarial Network
GRU	Gated Recurrent Unit
PCA	Principal Component Analysis
RFE	Recursive Feature Elimination
MI	Mutual Information
PS	Problem Statement
PQ	Project Question
PO	Project Objective
SMOTE	Synthetic Minority Oversampling Technique
DL	Deep Learning
AI	Artificial Intelligence

CHAPTER 1: INTRODUCTION

1.1 Introduction

Speech recognition is an artificial intelligence that has the ability to interpret spoken words and translate them into text by using a computer or machine. A wide range of computer science, linguistics, and computer engineering research is used in speech recognition. Speech recognition features are built into a lot of contemporary gadgets and text-oriented applications to facilitate hands-free or easier device usage. Computer algorithms are used by speech recognition systems to process, interpret, and translate spoken words into text. Through a series of procedures that include audio analysis, segmentation, digitization into a computer-readable format, and algorithmic matching with appropriate text representation, software may convert microphone sound into a written language that can be comprehended by both computers and humans.

Deep learning algorithms, especially those based on neural networks, have completely transformed the field of speech recognition. These algorithms are very good at identifying and transcribing speech because they can learn complex designs from audio data. The growth of deepfake audio, however, introduces a new difficulty. Deepfake audios, high fidelity and subtle details can deceive even the most sophisticated voice recognition algorithms. Consequently, to figure out how successful different speech recognition systems are against such complex audio forgeries, a comparative comparison of such systems is required.

Moreover, there is deep fake audio that can mimic the voice of a specific individual with remarkable accuracy, making it difficult to recognize between actual and manipulated recordings. To meet this obstacle, deep learning algorithms can develop methods to detect and distinguish

between real and fake audio. By exploiting datasets, these algorithms can analyze subtle patterns in the audio channel to find differences that are the sign of manipulation using deep fakes. Based on the research, deep learning can automatically learn features from raw data, such audio waveforms, it is especially well suited for voice recognition applications without the need for explicit feature engineering. Hence, this research aims to analyze deep fake audio dataset using deep learning algorithms and analyze its performance through different evaluation metrics.

1.2 Problem Statement

As technologies keep evolving rapidly, there are some people that misuse them such as violation of deep fake audio technology. Fake audio has been used for malicious purposes and is difficult to classify what is true and false audio during a digital forensic investigation. With the ability to produce truly exact audio copies, there is a need to implement efficient alternatives for detecting and reducing the spread of manipulated audio content. Hence, by using some methods in deep learning, it can help us in finding whether the audio is real or fake (Mvelo Mcuba et al., 2022).

Table 1: Problem Statement

PS	Problem Statement
PS1	Various advantages and disadvantages of each deep learning model give difficulty to detect and analyze deep fake audio (J. Khochare et al., 2021)

1.3 Project Question

According to the problem statement above, the following questions are set up.

Table 2: Project Question

PS	PQ	Project Question (PQ)
PS1	PQ1	How do deep learning algorithms identify the deep fake audio?
	PQ2	What is the best deep learning algorithm to determine deep fake audio?
	PQ3	Would the identification process be better if the best deep learning method combine with feature selection?

1.4 Project Objectives

The objectives listed below will be the main focus of this project in order to achieve research which will compare performance of various types of Deep Learning algorithm.

Table 3: Project Objective

PS	PQ	PO	Project Question (PQ)
PS1	PQ1	PO1	To analyze the process of deep fake audio dataset using deep learning method.
	PQ2	PO2	To identify which method produce the best performance through evaluation metrics.
	PQ3	PO3	To propose an enhanced deep learning method for deep fake audio dataset.

1.5 Project Scope

This research will focus on analyzing the performance of speech recognition on deep fake audio dataset by using a few methods from deep learning techniques:

1. Convolutional Neural Networks (CNN)
2. Recurrent Neural Networks (RNN)
3. Long Short-Term Memory (LSTM)
4. Generative Adversarial Networks (GAN)
5. Gated Recurrent Unit (GRU)

This research will also be using this dataset:

- Deep Fake Audio dataset from Kaggle.com repository (Size: 4 GB)

1.6 Project Contribution

- The identification of the best deep learning method which can be used by speech recognition system to specifically recognize real or fake audio.
- Addressing data imbalance by using SMOTE ensures more accurate and fair evaluation of the models.
- Enhancement of feature selection methods generalize and improve model's performance in classifying real or fake audio.

1.7 Thesis Organization

Chapter 1: Introduction

This chapter is used as a reference for defining the most essential information of the project which includes the problem statement and the objectives that will be focused on. This chapter also discusses the project scope and project outcome which ensures the project is fully understood.

Chapter 2: Literature Review

This chapter will discuss the project details, reading material from internet resources, the previous researcher's results and gather all the information that can be implemented in the current project.

Chapter 3: Research Methodology

In this chapter, the method that is used in this project will be explained and must be done in order to satisfy the project's goal.

Chapter 4: Proposed Method

This chapter focuses on the analysis and design of project assessment techniques with different types of deep learning algorithms that testing with datasets which is Deepfake Audio to fulfils the requirements of throughput and performance tests.

Chapter 5: Implementation

The testing method will be explained in detail in order to complete the task. It will indicate the outcome of each type of deep learning method that is used which is, Convolutional Neural Network (CNN), Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM), Generative Adversarial Networks (GANs) and Gated Recurrent Unit (GRU).

Chapter 6: Testing and Validation

In chapter 6, the analysis result will be compiled and discussed. Suitable comparison tables and graphs will be used to make it easier to understand and differentiate the differences between types of deep learning tested.

Chapter 7: Project Conclusion

A summary of the project and the best deep learning method will be discussed in this chapter. Thus, project limitations throughout this project's progress will be listed out to ensure that another research could make an improvement.

1.8 Conclusion

To summarize this chapter, the purpose of the content is to provide clarification and a deeper comprehension of the projects goals, that compare performance of various types of Deep Learning algorithm. Following that, my research will concentrate on determining the optimum deep learning strategy. The next chapter will also include a study of related work and the current trend in deep learning approaches

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction

This chapter will go through and compile all of the journal, papers, relevant books and related articles. It also will describe the related work and research on the existence of speech recognition. The goal of a literature review is to provide readers with clarification on how research fits into a wider field of study and to give readers a summary of the sources that were used to research a specific topic. Finding research methodologies and strategies that can be used in this project is the major goal of the literature review. To confirm that the study objectives have been satisfied, all data analyses collected through online research will be used.

2.2 Speech Recognition

Human speech is the most favored, efficient, and instinctive way of communication. Consequently, one may assume that speech input is more pleasant for individuals to use when interacting with machines than other outdated forms of communication like keypads and keyboards. Speech Recognition is a technology that enables a computer to interpret speech directly from the microphone or an audio file as input and translate it into text, usually in spoken language script. It states that in order for a speech recognition system to complete a task, it must be able to "perceive" input, "recognize" spoken words, and use that information as input for another machine. Speech recognition will be seen as a communication tool between humans and robots in the future. Figure 1 shows speech recognition representation.

Figure 1: Speech Recognition

(Retrieved from https://developer.nvidia.com/blog/how-to-build-domain-specific-automatic-speech-recognition-models-on-gpus/)

One of the main challenges in developing an autonomous speech recognition system is the wide range of differences in human speech and accent. Compared to those who speak only one language, bilingual or multilingual persons typically exhibit more of these variances in their speech patterns. When it comes to other variables like gender, dialect, and communication speed, the same thing occurs.

2.3 Dataset

A dataset is necessary for speech recognition testing and training. This section will go through the open source that will be utilized in comprehensive detail.

2.3.1 Deepfake Audio

With the speed at which speech recognition technology is developing, Deepfake Audio—scientifically termed logical-access audio spoofing techniques—has become more of a threat to voice interfaces. With the use of automated processes, Deepfake produces misleading material that is more challenging for human analysts to identify. A video of a former President Obama swearing during a public service announcement in 2019 served as a red flag for a significant deepfake incident. (J.Kietzmann et al., 2020).

2.4 Deep Learning

In 2006, Deep Learning (DL) was introduced by Hinton et al. (Hinton GE, 2006), that based on the concept of artificial neural network (ANN). First, deep learning gained popularity, which led to an upgrade and a resurgence of neural network research. Deep learning is now frequently called "new generation neural networks." This is due to the fact that well-trained deep networks can achieve notable success in a wide range of classification and regression tasks. (Karhunen J, 2015).

Nowadays, deep learning is regarded as a fundamental technology of the Fourth Industrial Revolution (4IR, or Industry 4.0), which is a subset of machine learning and artificial intelligence. Deep Learning techniques become black-box devices due to this lack of fundamental understanding, which prevents standard development.

Regarded as one of the hottest subjects in computing, deep learning technology has extensive application in fields such as text analytics, visual identification, healthcare, and cybersecurity, among others. Many companies, including Google, Microsoft, Nokia, and others, investigate it because of its capacity to learn from the provided data and produce meaningful results in a variety of classification and regression issues and datasets. (Karhunen J, 2015).

Deep Learning is viewed as a subset of Artificial Intelligence (AI) and Machine Learning in terms of working domain such as, Deep Learning may be thought of as an AI function that imitates how the human brain processes information. Based on historical data gathered from Google trends, a report (Sarker IH, 2021) illustrates how "Deep Learning" is becoming more and more popular worldwide. DL technology builds computer models by representing data abstractions using several layers.

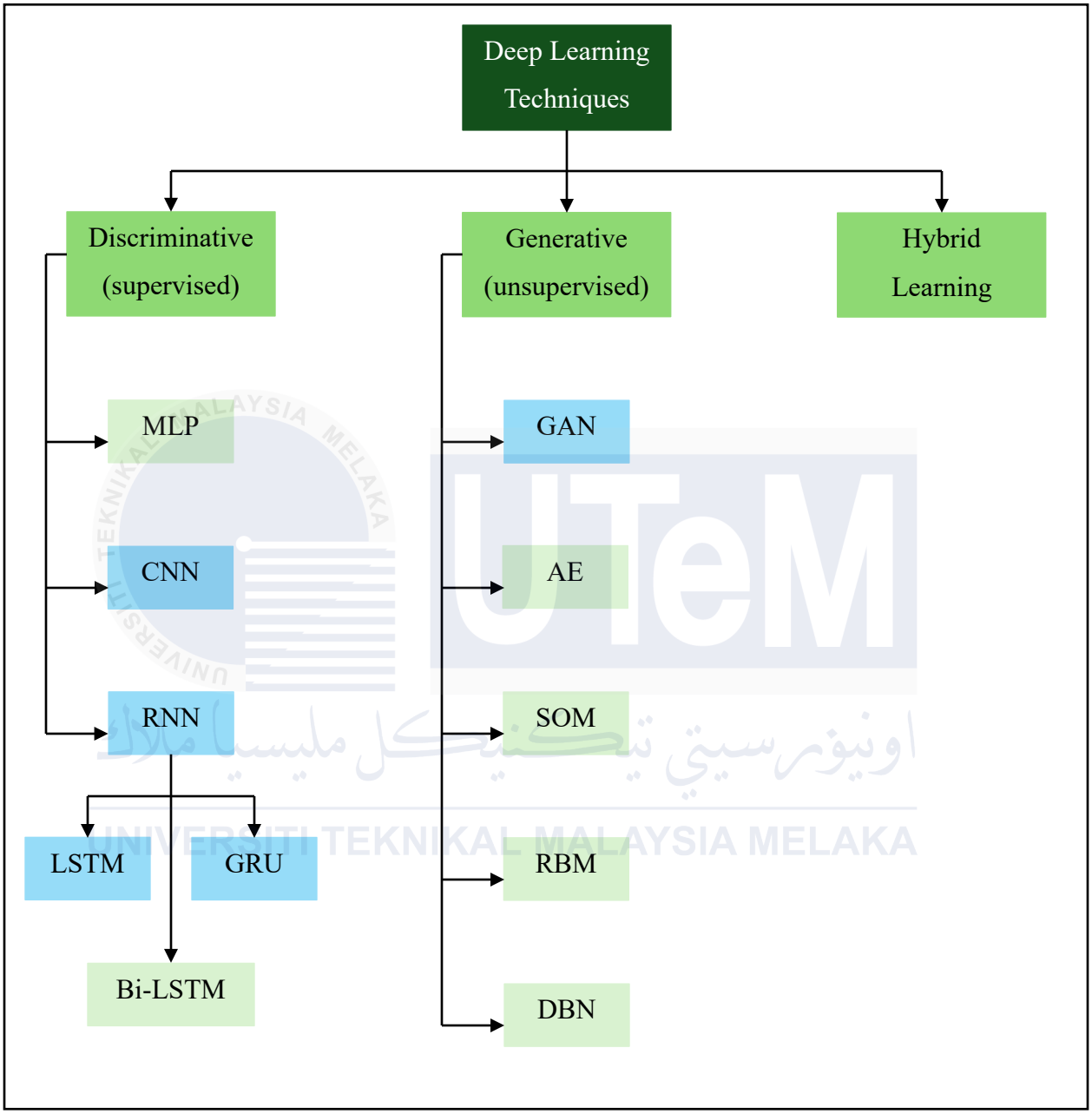

Figure 2: Proposed Deep Learning

Figure 2 above shows the Deep Learning Technique that I proposed in this experiment. The chosen technique is Supervised and Unsupervised Deep Learning.

2.4.1 Convolutional Neural Network (CNN)

Convolutional Neural Networks, or ConvNet for short, are a class of deep learning algorithms that are particularly well-suited for the analysis of visual input. CNN employs convolutional processes, which are based on concepts from linear algebra, to extract features from images and recognize patterns in them. CNNs can be configured to handle audio and other signal data in addition to pictures, even though that is their primary function. It requires a lot less preprocessing than other classification techniques. While filters are manually designed using traditional methods.

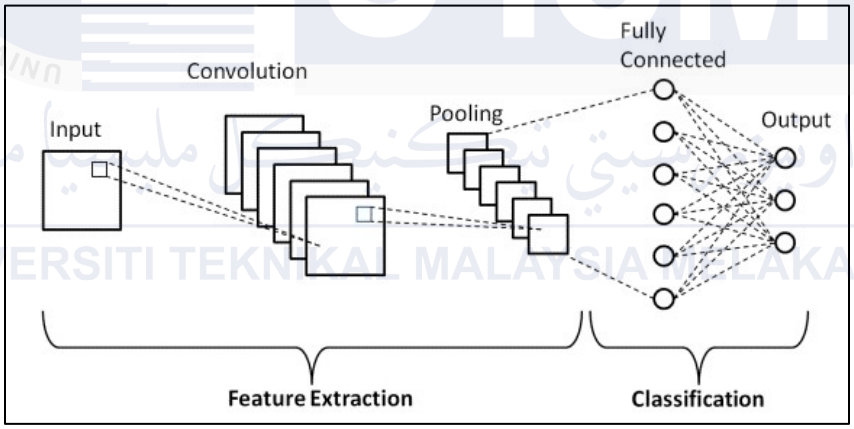

Figure 3: Convolutional Neural Network Architecture

(Retrieved from https://www.upgrad.com/blog/basic-cnn-architecture/)

The architecture of CNN was influenced by the organization of the visual cortex, which is crucial for the perception and processing of visual inputs, as well as the connection patterns of neurons in the human brain. Convolutional layers in CNN design use convolutions to identify features in the input data; pooling layers use feature maps that are downsampled to improve robustness to input fluctuations; and fully connected layers combine learnt features to generate final predictions.

2.4.2 Recurrent Neural Network (RNN)

Recurrent Neural Networks (RNN) is widely used in natural language processing. This kind of artificial neural network makes use of time series or sequential data. For ordinal or temporal issues, like language translation, natural language processing, speech recognition, and picture captioning, these deep learning methods are frequently utilized. They are included in well-known programs like Google Translate, Siri, and voice search. Recurrent neural networks (RNNs), like feedforward and convolutional neural networks (CNNs), learn from training data. Their ability to use information from previous inputs to affect the present input and output sets them apart. Stated differently, they employ the data to forecast the subsequent word in the series.

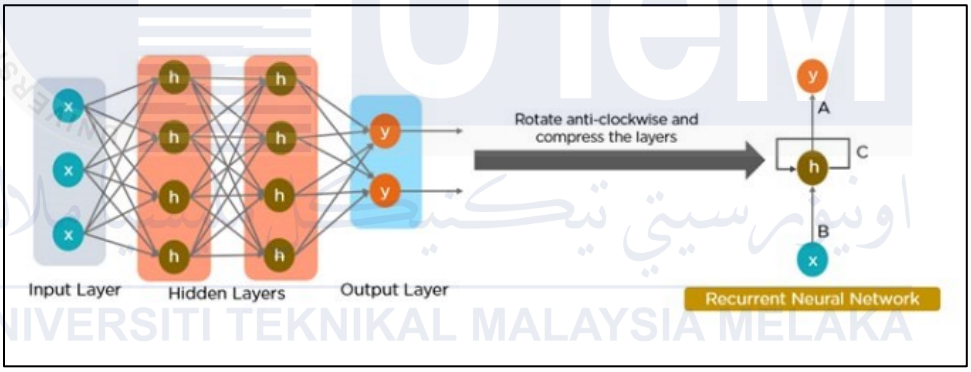

Figure 4: Recurrent Neural Network Architecture

(Retrieved from https://www.analyticsvidhya.com/blog/2022/03/a-brief-overview-of-recurrent-neural-networks-rnn/)

Standard neural networks have independent inputs and outputs; but, under specific conditions, such anticipating a phrases next word, preceding words are required, necessitating the recall of those words. Consequently, RNN was developed, which solved the issue by utilizing a Hidden Layer. The Hidden state, which retains particular details about a sequence, is the most crucial part of an RNN (Analytics Vidhya, 2024).

2.4.3 Long Short-Term Memory (LSTM)

Information can be retained in Long Short-Term Memory (LSTM) Networks, a type of sequential neural network designed for deep learning. It is a unique kind of recurrent neural network that can solve the RNNs vanishing gradient issue. Hochreiter and Schmidh uber created LSTM to address the issue brought about by conventional RNNs and machine learning methods. The Keras library in Python can be used to implement the LSTM Model. By incorporating feedback connections, LSTM is able to interpret data sequences instead of just single data points. It can therefore recognize and forecast patterns in sequential data, such as time series, text, and speech, with great effectiveness. As seen in the illustration below, the LSTM network design is composed of three components, each of which serves a distinct purpose. (Analytics Vidhya, 2024).

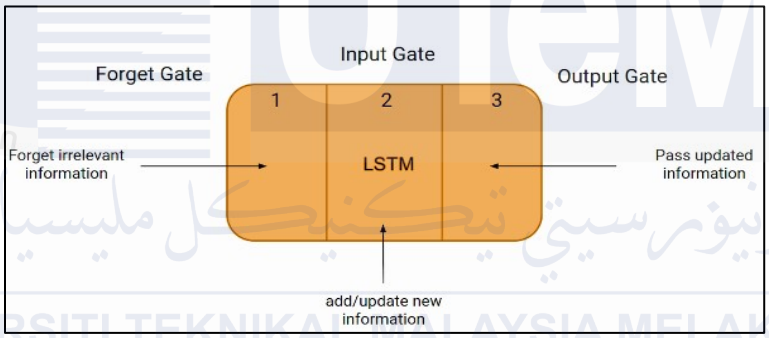

Figure 5: Long Short-Term Memory Architecture

(Retrieved from https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-short-term-memory-lstm/)

The first section determines whether the data from the preceding timestamp should be stored in memory or if it is unimportant and can be ignored. The cell attempts to learn new information from the input to this cell in the second section. Finally, the cell transfers the changed data from the current timestamp to the subsequent timestamp in the third section. A single-time step is this one LSTM cycle. Gates are these three components that make up an LSTM unit. Information enters and exits the memory cell, also known as the LSTM cell, under their supervision. The output gate is the last gate, the input gate is the second, and the forget gate is the first. These three gates, along with an LSTM cell or memory cell, make up an LSTM unit. This

unit can be thought of as a layer of neurons in a conventional feedforward neural network, where each neuron has a current state and a hidden layer.

2.4.4 Generative Adversarial Network (GAN)

Goodfellow introduced Generative Adversarial Networks (GANs), a unique generative model, in 2014. (Goodfellow, 2014). The game theory, which has two networks—a generator and a discriminator—inspired GANs. The generators job is to produce data that is as realistic as feasible in order to trick the discriminator. The discriminators job is to separate phony samples from authentic ones. (D. E. Rumelhart, 1986). Therefore, during the training phase, the discriminator and generator will compete with one another to reach Nash equilibrium (Zhaoqing Pan, 2019).

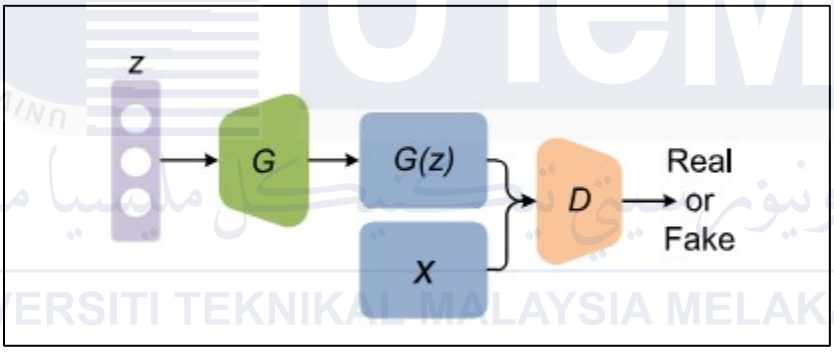

Figure 6: Generative Adversarial Network Architecture

(Retrieved from https://www.researchgate.net/figure/The-architecture-of-generative-adversarial-networks_fig1_331756737

The architecture of GANs is seen in the above figure. While discriminator Ds goal is to accurately discern between genuine and false data, generator Gs goal is to produce fake data as much as possible to match the possible distribution of real data. A random noise vector (z) with a uniform or normal distribution serves as the generators input. Generator G maps the noise to a new data space, producing a multi-dimensional vector known as G(z), or a bogus sample. Additionally, discriminator D is a binary classifier that accepts as inputs the genuine sample from the dataset as well as the fake sample created by generator G. Its output indicates the likelihood

that the sample is real as opposed to fake. The ideal condition is attained when discriminator D is unable to distinguish between data from the generator and the real dataset. At this stage, we have a generator model G that has mastered real data distribution.

2.4.5 Gated Recurrent Unit (GRU)

Specialized memory components called gated recurrent units (GRUs) are used to construct recurrent neural networks. Though they have demonstrated remarkable performance in a variety of tasks, such as obtaining the dynamics beneath brain data, little is known about the precise dynamics that a GRU network may represent. (Ian D. Jordan, 2019). Compared to LSTM, GRU has a simpler design and fewer parameters, which can facilitate training and increase computational efficiency. A "candidate activation vector," which is updated via the reset and update gates, takes the place of the memory cell state in GRU. While the update gate chooses how much of the candidate activation vector to add to the new hidden state, the reset gate chooses how much of the previous hidden state to ignore.

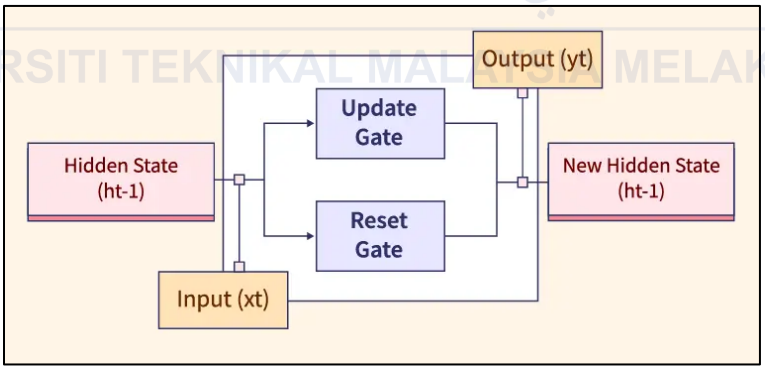

Figure 7: Gated Recurrent Unit Architecture

(Retrieved from https://www.scaler.com/topics/deep-learning/gru-network/)

Rebuilding its hidden state from the previous hidden state and the current input, GRU processes sequential data one element at a time. Combining data from the input and the prior

hidden state, the GRU calculates a "candidate activation vector" at each time step. For the subsequent time step, the concealed state is updated using this candidate vector.

2.5 Related Work

No.	Title & Author	Problem Statement	Method Used	Dataset Used	Contribution of The Paper	Future Research
1.	**Performance Comparison of Various Neural Network for Speech Recognition (2023)** - C. Sridhar - Aniruddha Kanhe	Performance of different Neural Network models for speech recognition	Trained 5 Neural Network models: - CNN - LSTM - Bi-LSTM - GRU - CONV-LSTM	Audio MNIST dataset	Found that the CNN and CONV-LSTM models had the best performance based on MFCC features.	-
2.	**Analysis of Speech Recognition Using Convolutional Neural Network (2020)** - Mohit Bansal - Dr. T. K. Thivakaran	The research aims to compare the performance to identify the best model for speech recognition between the two neural networks: - CNN - Basic Neural	- CNN - Basic Neural Network	Audio dataset collected from the internet, preprocessed by removing noise and cleaning the data.	Trained Speech command dataset for 5 speech command and obtained better performance and accuracy on train, test and validation data using convolutional Neural Network compared to Basic Neural Network.	-

		Network accuracy, validation accuracy, and test accuracy of CNN and basic NN models for speech recognition				
3.	**Performance Evaluation of Deep Neural Networks Applied to Speech Recognition: RNN, LSTM and GRU (2019)** - Apeksha Shewalkar - Deepika Nyavanandi	Comparison of performance of different neural network architectures (RNN, LSTM, GRU) for speech recognition that can match	- RNN - LSTM - GRU	a subset of the TED-LIUM release 2 corpus, which contains audio talks and transcriptions from the TED website and is designed for training acoustic models	- LSTM achieved the best word error rates on the reduced TED-LIUM speech dataset - GRU achieved word error rates close to LSTM, but with faster optimization/training time	- Parameter optimization to investigate the influence of different parameter settings - Experimenting with different learning rates, dropout rates, and higher numbers of

	- Simone A. Ludwig	with Word Error Rates (WER) and optimization speed/running time of the three models				neurons in the hidden layers. - The authors recommend using GRU for the reduced TED-LIUM dataset, as it provides good word error rates within an acceptable training time
4.	**Understanding Deepfakes: A Comprehensive Analysis of Creation, Generation**	In light of difficulties, it is crucial to investigate the	- Meso-4 - Mesolnception-4 - Neural Network technique	VidTIMIT database, which was used to create deepfakes with realistic facial	This paper provides a comprehensive analysis of deepfakes, focusing on their creation,	- Developing more efficient and advanced deep

	and Detection. (2023) - Sami Alanazi - Seemal Asif	use of detection algorithms and important methodologies to address the potential risks associated with deepfake technology. This review research paper aims to provide a thorough analysis of the creation and detection of deepfakes and contribute to a deeper understanding	- GAN	expressions, mouth movement and eye blinks.	generation, detection, and the importance of legislation and techniques for their detection, to address the ethical and legal implications of deepfakes in the digital landscape.	learning methods to distinguish between fake and authentic content. - Incorporating human instincts and cultural context in deepfake detection and combining manual expert analysis with software processing for

		of this concerning technology.				accurate detection.
5.	**The Creation and Detection of Deepfakes (2020)** - Yisroel Mirsky - Wenke Lee	to explore the creation and detection of deepfakes and provide an in-depth view as to show these architectures work.	- RNN - GAN variations like pix2pix and CycleGAN	VGG face dataset	the most effective and threatening deepfakes are those which are (1) the most practical to implement [Training Data, Execution Speed, and Accessibility] and (2) are the most believable to the victim [reality].	- Establishing content provenance and authenticity frameworks and using adversarial machine learning to protect content. - Evaluating the theoretical limits of deepfake attacks such as finding bounds on a model's

						delay and determining the limits of GANs.
6.	**Deepfakes: Threats and Countermeasures Systematic Review (2019)** - M. ALbahar - Jameel Almalki	to provides a systematic review of deepfake technology including its threats and countermeasures.	- Face detection - Multimedia forensics - Watermarking - CNN	Google's Image Search	Compared to detection that is done by humans, convolutional neural networks (CNNs), and other similar approaches, work on the principles of machine learning and have the ability to detect deepfake content via powerful image analysis features.	In order to achieve better and more realistic results, the first set can be further extended with photos from other sources. The second set consists of photos of the face that will be exchanged into the video.
7.	**Comparative Analysis of Deep-Fake Algorithms (2023)** - Nikhil Sontakke - Sejal Utekar	to spur researchers around the world to build innovative new technologies that can help detect deep fakes and manipulated media.	- GAN - CNN - RNN	- VGGFace2 - The Eye-Blinking - DeepfakeTIMIT	CNN have the ability to detect deepfake compared to GAN and RNN.	The authors suggest that future research should focus on addressing the limitations and challenges of current deepfake detection methods,

	- Shriraj Sonawane					to improve the accuracy of detecting deepfake videos as deepfake technologies continue to advance.
8.	**Deepfake Detection: A Systematic Literature Review (2022)** - M. Rana - M. N. Nobi - B. Murali - A. Sung	present a systematic literature review (SLR) on Deepfake detection in the paper and aim to describe and analyze common grounds and the diversity of approaches in current practices on	- Meso-4 - MesoInception-4	- FaceForensics++ - Celeb-DF - DFDC	- In the experiments, the FaceForensics++ dataset occupies the largest proportion. - The deep learning (mainly CNN) models hold a significant percentage of all the models.	Establish a unique framework for fair and consistent assessment of Deepfake detection methods across studies.

		Deepfake detection.				
9.	**Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Direction (2021)** Iqbal H. Sarker	aims to draw a big picture on DL modeling that can be used as a reference guide for both academia and industry professionals.	- Summarize real-world application areas of deep learning. - Identify 10 potential aspects for future deep learning modeling and research directions.	Structured and comprehensive view of deep learning technology were presented, which is considered a core part of artificial intelligence as well as data science.	The sophisticated learning algorithms then need to be trained through the collected data and knowledge related to the target application before the system can assist with intelligent decision-making.	The sophisticated learning algorithms then need to be trained through the collected data and knowledge related to the target application before the system can assist with intelligent decision-making.
10.	**Performance Metrics (Error Measures) in Machine Learning Regression, Forecasting**	Performance metrics develop new topologyto advance knowledge and facilitate	4 categories performance metrics: - Primary - Extended - Composite - Hybrid set	-	- The paper proposed a framework of 4 categories of performance metrics: primary,	- Examine the properties of extended metrics, composite

	and Prognostics: Properties and Typology (2018) A. Botchkarev	the use of metrics in machine learning regression.			extended, composite and hybrid sets. - The paper identified 3 key components that determine the properties of primary metrics: point distance method, normalization method, and aggregation method.	metrics and hybrid sets of metrics. - Conduct an empirical study to find associations between the conceptual properties of primary metrics and their numerical behavior across different data characteristics and

						research requirements
11.	**Effective Treatment of Imbalanced Datasets in Health Care Using Modified SMOTE Coupled with Stacked Deep learning Algorithm (2023)** - Sowjanya A. M. - Mrudula O.	Often times the datasets are quite imbalanced and sampling techniques like Synthetic Minority Oversampling Technique (SMOTE) give only moderate accuracy in such cases.	- Distance-based SMOTE (D-SMOTE) - Bi-phasic SMOTE (BP-SMOTE) - Decision Tree - Naïve Bayes - Random Forest	- Framingham dataset - Breast Cancer dataset - Covid-19 dataset	RF proved to yield high value for evaluation metrics both D-SMOTE and BP-SMOTE.	To increase the accuracy, the researchers can do stacking approach in terms of Stacked CNN and Stacked RNN.
12.	**DeepSMOTE: Fusing Deep Learning and**	Modern advances in deep learning	- SMOTE - AMDO - MC-CCR	- MNIST - FMNIST - CIFAR-10	We propose a carefully designed and thorough	Focus on enhancing DeepSMOTE with

	SMOTE for Imbalanced Data (2023) - Dablain D. - Krawczyk B. - Chawla N. V.	have further magnified the importance of the imbalanced data problem	- MC-RBO	- SVHNs - CelebA	experimental study that compares DeepSMOTE with state-of-the-art oversampling and GAN-based methods. Using five popular image benchmarks and three dedicated skew-insensitive metrics over two different testing protocols, we empirically prove the merits of DeepSMOTE over the reference algorithms. Furthermore, we show that DeepSMOTE	information regarding class-level and instance-level difficulties, which will allow it to better tackle challenging regions of the feature space.

					displays an excellent robustness to increasing imbalance ratios, being able to efficiently handle even extremely skewed problems.	
13.	**A Comprehensive Review of Feature Selection and Feature Selection Stability in Machine Learning (2023)** (Buyukkececi et al., 2023)	Reducing the dataset size by eliminating redundant and irrelevant features plays a pivotal role in increasing the performance of machine learning algorithms, speeding up	Feature Selection method: - Filter - Wrapper - Embedded	LDA dataset	A limited number of studies in literature have analyzed unsupervised and semi-supervised feature selection stability. Thus, these topics constitute an open research field for researchers. This study is mostly based on supervised feature selection and selection stability.	Refer the researcher to encompass other types of feature selection in detail due to the wide scope.

		the learning process, and building simple models.				

2.6 Conclusion

This chapter concluded with a discussion and compilation of the studys literature review. The idea of speech recognition, deep learning and network traffic annotations are explained. In this chapter also introduces in detail about few methods of deep learning that has chosen to process deep fake audio dataset, which is CNN, RNN, LSTM, GAN and GRU. Each deep learning method was explained in detail with related figures to ensure that the delivery of information is relevant and easy to understand. Moreover, the theory of network traffic annotations is also expressed in words sequentially. The overview of previous research also provided to compare its parameters between other researchers that is related to speech recognition analysis.

CHAPTER 3: PROJECT METHODOLOGY

3.1 Introduction

During the methodology stage, the most important element is to outline the steps and procedure that will be followed to conduct the project research effectively. Methodology for research was described as a methodical procedure for gathering significant data and information for use in research projects. It is also characterized as a procedure that enables the researcher to deal with unanswered questions and study how research can be conducted. In the context of research, relevant data and information can be obtained via theoretical techniques, the examination of related experiments or studies, interviews, and other methods. Analysis and interpretation will be done after the data and information are gathered. To guarantee that this studys goals can be achieved and that the research approach is appropriate, it was necessary to adhere to a few rules. Lastly, a project schedule and milestone will be included in this chapter, which will list all the activities involved, and the estimated time taken to complete each activity. The project schedule and milestones are very important because they provide guidelines to follow in each phase of the project.

3.2 Research Methodology

A project methodology is an organized approach to complete a collection of tasks in an efficient and organized way. This section contained the approach, and the comprehensive steps needed to perform the testing.

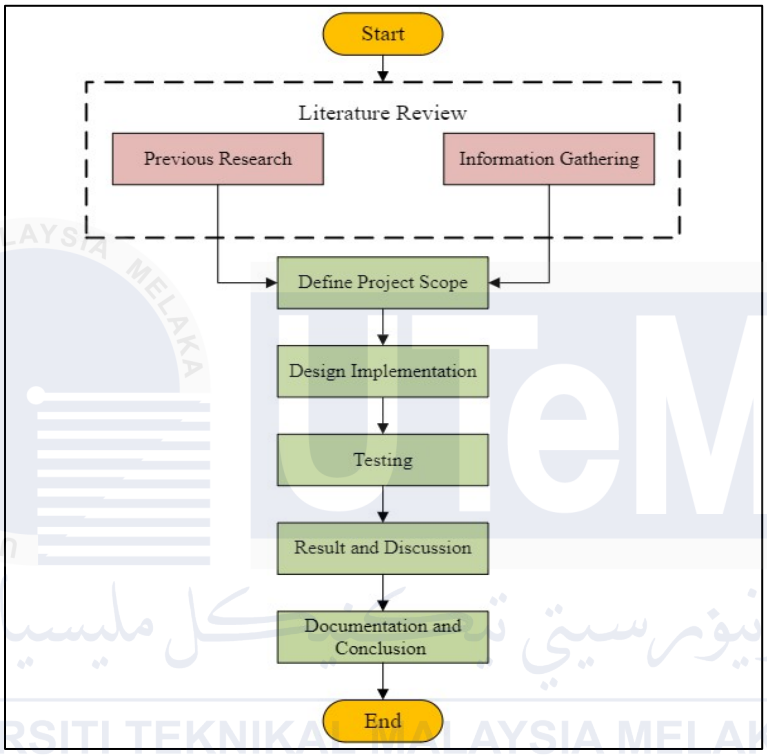

Figure 8: Proposed Research Methodology

Figure 8 shows the flow of my research methodology. To ensure that the testing was completed in accordance with the standard specification and requirement, this study was essentially separated into multiple sections. The overall flow chart of the research process for all of the testing conducted for this study is explained below.

1) Previous Research

In this phase, previous research is being conducted by reading articles and journals that related to the research topic. The selection of articles that have been established and trusted is very important to ensure that the information has been filtered for the next phase is useful and valid.

Among the scopes required to filter articles are CNN, Deep Fake Audio, Speech Recognition and Network Traffic Annotation. As a result, this phase will generate an overview of the research project.

2) Information Gathering

After finding any related study from previously conducted research, the information from each article must be gathered and analyzed. This step provides techniques to further study the articles and related research topics in more depth. Thus, this phase is important for selecting the right hardware, software and operating system that will be used to analysis the project.

3) Define Project Scope

The scope is defined by focusing the performance of speech recognition on deep fake audio by using a few methods from deep learning with network traffic annotations and suitable operating system that are going to be used to succeed in the research project.

4) **Design and Implementation**

The phase will explain the design of deep learning method with network traffic annotation that will give various testing scenarios. Implementation of the project will conduct by using Google Collab and Jupyter Notebook. The language that will be used is Python.

5) Testing

In this phase, the testing on Jupyter Notebook will be conducted for each deep learning method with network traffic annotations. It will come out with the result based on project research's objective on different scenarios.

6) Result and Discussion

This phase will gather the analysis result that was tested by each deep learning method and come out with comparison of accuracy, F1-Score, effectiveness of annotations and precision and recall, according to the objective of the project. The table, graph and chart will be produced in this phase to make the reader easy to read and understand.

7) Documentation and Conclusion

In this phase, all of the tested results will be documented. The objective is to determine the link quality distribution factors, and based on the analyzed data, recommendations are generated.

3.3 Project Schedule and Milestones

This section will be the timeline for this research project. It will allocate specific time for each phase from the start until the project finishes.

3.3.1 Project Milestones

Project Milestones is created and structured the task accordingly with the specific time for each phase from the start until the project finishes.

Table 4: Project Milestone

Phase	Week	Activity	Deliverable
PSM 1			
Planning	1	Meeting with supervisor	Proposal preparation Present proposal to supervisor
		Identify title, problem statement, objectives and scope.	
		Completing proposal	
	2	Submitting proposal to supervisor	
		Correction on proposal	
	3	Meeting with supervisor	
		Proposal accepted by supervisor	
		Submitting proposal	Via portal i@UTeM under ePSM
	4	Starting writing report on chapter 1	Completing chapter 1
		Completing chapter 1	
Analysis	5	Study and research on literature review	Completing chapter 2
		Meeting with supervisor	
		Gather the information needed using Mendeley	
	6	MID-SEM BREAK	
	7	Meeting with supervisor	
		Starting writing report on chapter 2	
		Completing chapter 2	

		Project progress 1	Progress 1 submission
		Submitting chapter 1 and chapter 2 to supervisor and ePSM	
	8	Study on methodology of the project	Completing chapter 3
		Create milestone of the project	
	9	Starting writing report on chapter 3	
		Completing chapter 3	
Design	10	Meeting with supervisor	Completing chapter 4
		Choose suitable tools to be used	
		Design a network environment	
		Set up physical network environment	
		Project progress 2	
	11	Completing chapter 4	
		Submit to supervisor	
	12	PSM 1 draft report preparation	Completing and do correction on report
	13		
	14	Submit draft report to supervisor and evaluator	Presentation to supervisor and evaluator
		Report evaluation	
		Demonstration supervisor and evaluator.	
PSM 2			
Implementation	1	Implement five Deep Learning method to test with Deepfake audio dataset.	Implement the complete progress of testing
Discussion	2	Monitor the performance, analyze and record the result of the project	Completing chapter 5 & 6 Progress presentation 2
		Analyze and record the result.	
	3	Meeting supervisor to discuss the result.	
		Completing chapter 5	Present chapter 5 & 6 with supervisor
		Completing chapter 6	

	4	Submit chapter 5 & chapter 6 to the supervisor and ePSM	
Project Conclusion	5	Highlight summary of the project, constraints and future work.	Present overall report and experiment to supervisor.
		Completing chapter 7	
	6	PSM 2 draft report preparation.	Completing and do correction on both report and experiment.
		Submit draft report PSM 2 to supervisor and evaluator.	Submit completed report to supervisor, evaluator and ePSM.
		Report evaluation.	
		Schedule a presentation date	
	7	Full and final demonstration to supervisor and evaluator.	Final presentation to supervisor and evaluator.

3.3.2 Project Gantt Chart

A Gantt Chart is a visual representation of a timeline for each project activity. If one of the project's activities is delayed, it affects the project's overall durability and expenses, both of which are expected to grow in the future.

Table 5: Gantt Chart

Task/Weeks	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
Preparing proposal																						
Correction on proposal																						
Submitting a proposal to PSM committee																						
Writing report Chapter 1																						
Study and gather information for literature review																						
MID-SEMESTER BREAK																						
Writing report Chapter 2																						
Brief methodology and create milestone																						
Writing report Chapter 3																						
Design and setup network environment																						
Writing report Chapter 4																						
PSM 1 draft report preparation																						

Submit full report to supervisor and evaluator																						
Demonstration with supervisor and evaluator																						
Implement five Deep Learning method to test with Deepfake audio dataset.																						
Monitor the performance, analyze and record the result of the project																						
Writing chapter 5 & 6																						
Completing and submit report chapter 5 & 6																						
Highlight the summary of the project																						
Completing Chapter 7																						
PSM 2 draft report preparation																						
Report evaluation																						
Full and final demonstration supervisor and evaluator																						

3.4 Conclusions

An explanation and defense of the research technique employed in this study are given in this chapter. For the purpose of this investigation, the qualitive approach was selected as the method of inquiry due to the many advantages and reliability it offers. Experimentation with different research methodologies, such as textual analysis and observation, was very important. This chapter also includes a schedule for the project. The analysis and results of the project will be covered in the next chapter.

CHAPTER 4: EXPERIMENTAL DESIGN

4.1 Introduction

It is necessary to design the procedure and structure since it will give a guide for the next chapter, which will cover implementation and testing. This chapter will provide more detail about explanation and analysis that mentioned in the previous chapter. Therefore, this chapter is also necessary for making sure the testing runs efficiently. This will include the elaboration of the design of Deep Learning technique that can be applied to analyze speech recognition on authentication systems by detecting and preventing unauthorized access attempts using deep fake audio.

4.2 Project Requirements Analysis

A project requirement is a specific requirement that is utilized to guarantee the project's seamless execution. This section provides the project specification that will be used when the project is implemented.

4.2.1 Hardware Requirements

Hardware is needed to make sure the project moves without my interruption. The essential hardware of this project is arranged as follows:

Table 6: Hardware Requirement

Specification	Details
Processor	11th Gen Intel® Core™ i5-1135G7 @ 2.40GHz
RAM	8 GB
Storage	SSD with 512 GB storage capacity
Graphic Card	NVIDIA GeForce MX450

4.2.2 Software Requirements

For this project, the following software is necessary to complete the flow of this project:

Table 7: Software Requirement

Specification	Details
Operation System	Window 11
Programming Language	Python 3.11.7
Development Environment	Jupyter Notebook 7.0.8

a) Python

```
[2]: import sys
     print(sys.version)

     3.11.7 | packaged by Anaconda, Inc. | (main, Dec 15 2023, 18:05:47) [MSC v.1916 64 bit (AMD64)]
```

Figure 9: Python Version Used in Jupyter Notebook

Python is a high-level, general-purpose programming language with dynamic semantics. Significant indentation is used in its design philosophy, which prioritizes code readability. For this project, Python version 3.11.7 will be used as a programming language to perform coding.

b) Jupyter Notebook

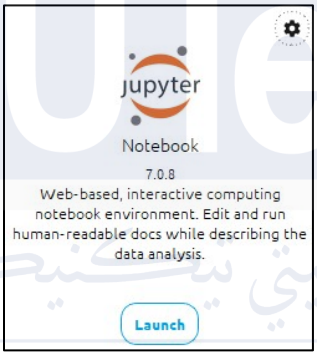

Figure 10: Jupyter Notebook Version Used to Run Code

Jupyter Notebook is to create interactive notebook documents that have the ability to include media, equations, live code, and other computational outputs. Jupyter Notebook is used to document and demonstrate coding workflows or simply experiment with code. Jupyter Notebook version that will be used is 7.0.8.

4.3 Design Overview

Machine Learning Process is the process of designing and constructing an organized and efficient structure that is adapted to the particular requirements of an organization. This takes place in order to satisfy the organizations standards.

It starts with datasets that obtained from Kaggle, the datasets then will go through preprocessing which involves a series of steps and techniques to convert unprocessed data into a format better suited for examination. For the training process, four methods of deep learning algorithms were selected, which are CNN, RNN, LSTM, GAN and GRU. In this process, it will distinguish between real and fake. Lastly, based on all of the process, it will come up with evaluation metrics output such accuracy, recall and F1 score.

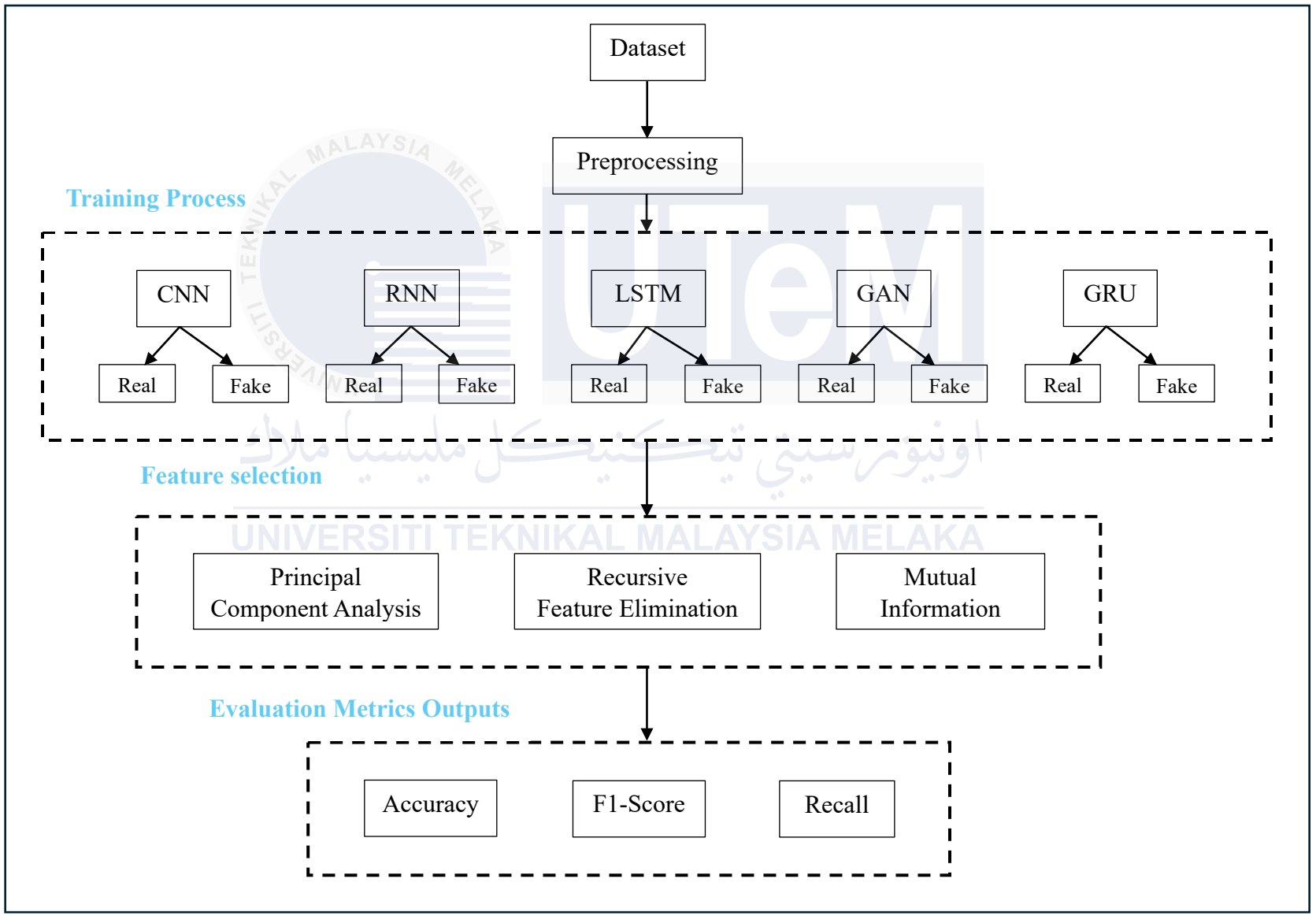

Figure 11: Project Design Overview

4.3.1 Training Process

CNN

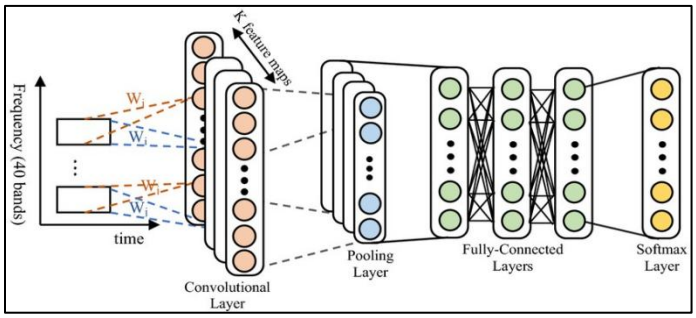

Figure 12: CNN Process

(Retrieved from https://www.researchgate.net/figure/Convolutional-Neural-Network-Architecture-for-Speech-Recognition_fig1_336819865)

Figure 12 shows Convolutional Neural Network (CNN) architecture for speech recognition. CNNs are a kind of artificial neural network that draws inspiration from the anatomy and physiology of the visual cortex in animals. The input layer in this architecture takes is a spectrogram, a picture that depicts speech.

The architecture in the image is made up of multiple layers. The first layer is a convolutional layer which extracts features from the input speech signal. The convolutional layer is followed by a pooling layer which reduces the dimensionality of the data by summarizing the information from small regions. The data then passes through one or more fully connected layers which are to classify them into different speech sounds or words. The final layer is a SoftMax layer which outputs the probabilities of the input speech signal belonging to each of the possible classes (Vishal Passricha et al., 2019).

A spectrogram of the audio signal would represent CNN's input in the context of speech recognition. A spectrogram is a graphic that shows a signals frequency content over time.. CNN would then learn to identify patterns in the spectrogram that correspond to different phonemes which are units of sound in a language. By learning these patterns, CNN can then recognize spoken words.

RNN

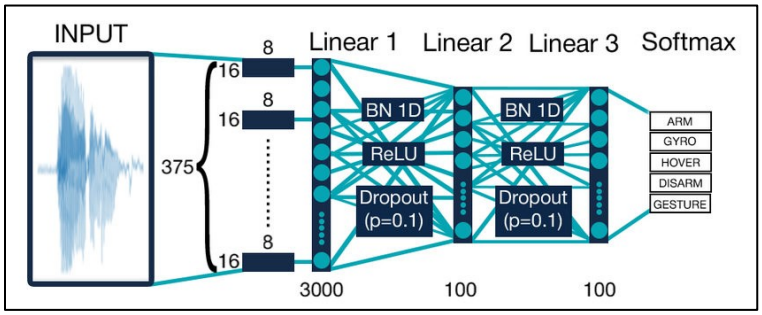

Figure 13: RNN Process

(Retrieved from https://www.researchgate.net/figure/Recurrent-Neural-Network-architecture-used-for-voice-recognition_fig3_338201311)

Figure above shows a Recurrent Neural Network (RNN) architecture used for voice recognition. Speech and text are examples of sequential data that can be processed by an RNN, a kind of artificial neural network. The cutting-edge performance on the assignment of voice recognition is achieved by RNN (D. Coimbra de Andrade et al., 2020) which employs in current work.

The diagram shows the different stages of RNN architecture including input layer, linear layer, ReLU layer, dropout layer and output layer. The input layer receives an input signal which is a vector numbers representing the speech waveform. The linear layers are made up of neurons that perform linear transformations on the input signal. The patterns in the data can be discovered by this transformation. The ReLU layers introduce non-linearity into the network. This is important because It makes it possible for the network to get more precise patterns within the data. Throughout training, a certain percentage of neurons are randomly removed by dropout layers. By doing this, the network is kept from overfitting to the training set. The output layer creates the network's final output, a vector of numbers that represents the probabilities that various words will be pronounced. (Alexander Menshcikov et al., 2019).

This architecture makes it possible to do tasks like voice recognition by systematically processing the speech signal and using the context from previous events to interpret the input at

present. Because of this feature, RNNs are a fundamental component of many machine learning applications, especially those that deal with sequential data (Saba Hesaraki, 2023).

LSTM

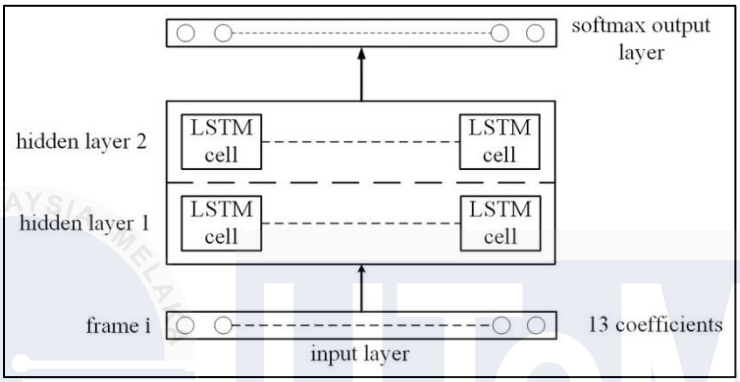

Figure 14: LSTM Process

(Retrieved from https://www.researchgate.net/figure/The-designed-framework-for-the-analysis_fig4_342569619)

Figure above shows the Long Short-Term Memory (LSTM) network, a specific type of RNN architecture that is used for speech recognition tasks. LSTMs are adept at handling long-term dependencies within sequential data that come from their internal gating mechanism. With two layers that include 100 LSTM cells each, this model is a deep RNN. The input layer, which is the lowest layer, has 13 units in it that would house the time frame coefficients. Two LSTM recurrent layer layers make up the following layer. This cell processes the input sequence one step at a time. It is capable of learning and remembering information relevant for the task over long durations. The final layer is a softmax output layer that maps the processed information from the hidden layers (two layers of LSTM) into probabilities corresponding to different words or phonemes in the spoken language (Md Mahadi Hasan Nahid et al., 2020).

To conclude, the stacked LSTM layers process the speech data sequentially. At each step, the LSTM cell considers the current input, along with the information retained from previous steps, to make predictions about the speech signal. Because of this feature, LSTMs can accurately recognize speech by capturing the contextual connections between sounds in speech. Each LSTM

cell analyzes the speech input step-by-step, weighing the preceding moments context in addition to the sound at that moment. This enables the network to learn the complex relationships between sounds in speech, ultimately producing voice recognition that is more accurate compared to traditional neural networks (Zaynab Almutairi, 2022).

GAN

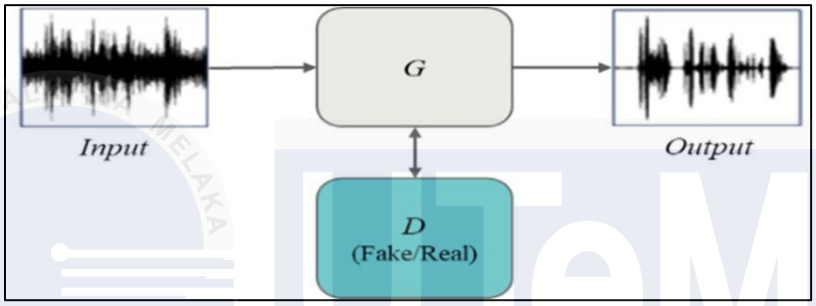

Figure 15: GAN Process

(Retrieved from https://www.researchgate.net/figure/Architecture-of-A-GAN-for-Speech-Enhancement_fig3_354252395)

The application of Generative Adversarial Networks (GAN) in voice recognition systems has also drawn more interest. As seen in the above diagram, a GAN is made up of a discriminator network (D) and a generating network (G). GAN training is often constructed on fully connected or convolutional layers (H. Phan et al., 2019) The fundamental idea underlying GANs is that a generator is trained to produce new data that closely resembles an input dataset. On the other hand, the discriminator gains the ability to distinguish between produced and accurate data. (Yibo He et al., 2023).

The generator aims to produce an output that is comparable to the training data by using random noise as input. The discriminator attempts to discern between true and fraudulent data using both generated and actual data. In a manner akin to playing a game, the discriminator trains to become more adept at spotting bogus input, while the generator trains to provide better, more convincing results. Having a generator that can produce an output that is identical to genuine data

and a discriminator that is unable to discriminate between produced and real data is the ultimate goal of GANs. (Kah Phooi Seng et al., 2023).

GRU

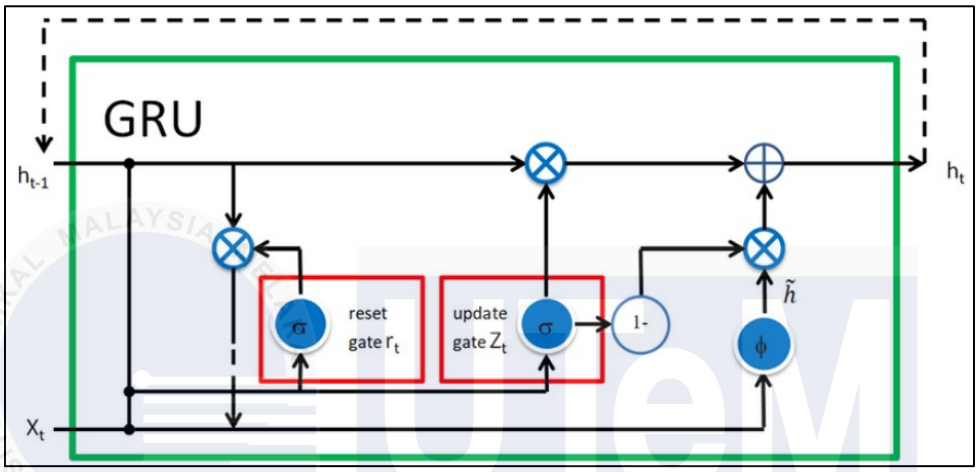

Figure 16: GRU Process

(Retrieved from https://www.researchgate.net/figure/The-diagram-of-a-GRU-cell_fig3_323570767)

The Gated Recurrent Unit (GRU) is a type of recurrent neural network (RNN) designed to handle sequential data. In contrast to conventional RNNs, GRUs have "gates" that control the information flow, which enables them to successfully identify long-term dependencies. Because of this, they are especially well-suited for jobs like machine translation and natural language processing.

The input, the hidden state, and the output are the three main parts of a GRU. The networks memory of previous inputs is represented by the hidden state, whereas the input is the current data point. The output is the prediction or classification made by the GRU based on the input and the hidden state. A GRUs update gate and reset gate control how much of the previous secret state is shown. This mechanism enables GRUs to selectively retain relevant information and discard irrelevant details, improving their ability to model complex sequential patterns.

4.3.2 Evaluation Metrics

In this project, evaluation metrics is important since it is quantitative measures that are employed to evaluate the effectiveness of models, algorithms or systems, particularly in machine learning. These metrics help determine how well a model is performing, guide model improvements and facilitate comparison between different models. It focuses on analyzing different methods to see which methods are the best and can meet the project's objectives. Accuracy, recall and F1 Score are all important considerations. The data that gathered from the analysis gave a full overview of functional deep learning method.

1. **Accuracy**

   A classification metric that calculates the percentage of cases in the dataset that are correctly classified out of all the instances. It is calculated as the number of true positive and true negative predictions divided by the total number of predictions. While accuracy is intuitive and widely used, it can be deceptive in imbalanced datasets when the majority class predominates, potentially masking poor performance on the minority class.

2. **Recall**

   Also known as sensitivity or the true positive rate, is a classification metric that quantifies the percentage of real positive cases that the model correctly identifies. It is computed by dividing the total number of true positive and false negative forecasts by the number of true positive predictions. Recall is crucial in situations where it is important to minimize false negatives, such as in medical diagnoses where missing a positive case could have severe consequences.

3. **F1-Score**

   A classification metric that combines precision and recall into a single measure by taking their harmonic mean. It is particularly useful when there is an uneven class distribution, and there is a need to balance the trade-off between precision (the percentage of positive cases that were accurately detected out of all instances that were anticipated to be positive) and recall. The F1 Score offers a more nuanced view of model performance than accuracy alone, especially in cases where both false positives and false negatives carry significant implications.

4.4 Conclusion

In conclusion, this chapter gives a summary of the projects experimental design. It explains the components required to illustrate the process flow in the project. It aids the research in maintaining an understanding of the implementation procedure that will be carried out.

CHAPTER 5: IMPLEMENTATION

5.1 Introduction

In this chapter, the implementation of Deep Learning process will be shown. The training of each model performs in Jupyter Notebook using python language to get the result. Each model will be tested in the same flow which is model, model combine with SMOTE and model combine with SMOTE and Stratified K-Fold. The final implementation is when the combination applied with selected feature selection for model enhancement.

5.2 Dataset Acquisition

The dataset that used in this project is DEEP-VOICE: DeepFake Voice Recognition, by Jordan J. Bird. This dataset contains examples of real human speech and DeepFake versions of those speeches by using Retrieval-based Voice Conversion. This dataset was obtained from Kaggle, a platform that is widely known for storing dataset and competitions related to data science and machine learning. The dataset that I chose is applicable to the goals of the study which is to analyze the audio whether it is real or fake.

The dataset consists of several elements related to audio processing, sound spectrum and signal's energy. The dataset represents a set of fake audio samples characterized by various audio features. The differences in these features across the samples suggest variability in how these fake audios are generated or manipulated. This dataset used for training a model to detect or classify fake audio based on these features:

- Chroma Short-Time Fourier Transform (chroma_stft)

  Chroma STFT is a feature that captures the distribution of energy across different pitch classes (like C, C#, D, etc.) within an audio signal. It groups the frequencies of a signal into 12 pitch classes, reflecting the harmonic and melodic characteristics of the sound. This is particularly useful for music analysis, as it helps in identifying the tonality and harmony of the audio, making it an important feature in tasks like music genre classification and chord recognition.

- Root Mean Square Energy (rms)

  RMS energy is a measure of the average power of an audio signal over time. It quantifies the loudness of the audio, with higher values indicating louder sounds and lower values suggesting quieter or silent segments. RMS is crucial in identifying the dynamics of audio, helping distinguish between different types of sound events, such as speech, music, or silence. This feature is particularly useful in speech processing and audio event detection, where energy levels are indicative of different acoustic events.

- Spectral Centroid

  Spectral centroid represents the "center of mass" of the spectrum and is often referred to as the brightness of a sound. It is calculated as the weighted mean of the frequencies present in the signal, with the weights being the magnitudes of the corresponding frequencies. A higher spectral centroid indicates a sound with more high-frequency components, often perceived as brighter or sharper. This feature is commonly used in audio analysis to differentiate between different types of sounds, such as distinguishing between percussive and tonal sounds in music.

- Spectral Bandwidth

  Spectral bandwidth measures the width of the frequency spectrum, indicating the range of frequencies present in the audio signal. It captures the spread of frequencies around the spectral centroid, providing insight into the tonal complexity of the sound. Wider bandwidths suggest a richer harmonic content, while narrower bandwidths indicate simpler or more uniform tones. This feature is particularly useful in characterizing the timbre of musical

instruments and distinguishing between different types of audio content based on their frequency distribution.

- Spectral Rolloff

  Spectral roll off is the frequency below which a certain percentage (typically 85%) of the total spectral energy is contained. It effectively separates the harmonic from the noisy parts of the signal, making it useful in distinguishing between sounds with clear tonal characteristics and those that are more noise-like. A lower rolloff frequency suggests that most of the energy is concentrated in the lower frequencies, while a higher roll off indicates a more significant presence of high-frequency components. This feature is valuable in audio signal processing for tasks such as genre classification, where different genres have characteristic frequency distributions.

- Zero Crossing Rate

  The zero-crossing rate is the rate at which the audio signal changes sign, crossing the zero-amplitude line. It is a simple but effective measure of the signals noisiness and percussiveness. Higher zero crossing rates are typically associated with noisier signals or sounds with sharp transients, like percussive instruments, while lower rates are found in smoother, more tonal sounds like singing. This feature is particularly important in speech and music processing for identifying fricatives in speech or the presence of noise and percussive elements in music.

- Mel-Frequency Cepstral Coefficients (MFCC1 to MFCC20)

  MFCCs are a set of features that collectively describe the short-term power spectrum of an audio signal on a mel scale, which mimics the human ears response to different frequencies. Each MFCC represents the amplitude of specific frequency bands, capturing the timbral and tonal qualities of the audio. Lower-order MFCCs capture broad spectral patterns, while higher-order coefficients capture finer details. MFCCs are widely used in speech and audio processing tasks, such as speech recognition and speaker identification, as they provide a compact representation of the audios spectral properties that is highly effective for modeling and classification.

5.3 Data Preprocessing

5.3.1 Feature Extraction

The first step involves loading the audio files and extracting their features. In this code, the Mel-Frequency Cepstral Coefficients (MFCC) are used as features. MFCCs capture the timbral properties of the audio, which are crucial for distinguishing between different sounds, such as real and fake audio. The 'extract_features' function loads an audio file using 'librosa', computes the MFCCs, and ensures that each feature matrix has a consistent shape by padding or truncating the matrix to a fixed length (MAX_LENGTH). This is essential for maintaining uniform input dimensions for subsequent processing steps.

```
# Function to load and extract features from audio files
def extract_features(file_path, max_length=MAX_LENGTH, n_mfcc=N_MFCC):
    y, sr = librosa.load(file_path, sr=None)
    mfcc = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=n_mfcc)

    # Pad or truncate MFCC to the fixed size
    if mfcc.shape[1] > max_length:
        mfcc = mfcc[:, :max_length]
    elif mfcc.shape[1] < max_length:
        pad_width = max_length - mfcc.shape[1]
        mfcc = np.pad(mfcc, ((0, 0), (0, pad_width)), mode='constant')

    return mfcc
```

Figure 17: Feature Extraction

5.3.2 Label Encoding

The labels (real or fake) are converted into numerical format using label encoding. This is a necessary step because machine learning models typically require numerical input. The 'LabelEncoder' from 'sklearn' is used to transform the string labels into integers, which are then used for training and evaluating the model.

```
# Load and prepare data
data, labels = load_data(real_dir, fake_dir)
```

Figure 18: Process of Load and Prepare Data

5.3.3 Data Reduction

Data reduction techniques aim to simplify a dataset while retaining as much relevant information as possible. These techniques are essential for reducing computational complexity, improving model performance, and avoiding overfitting. Data reduction techniques that are used is Principal Component Analysis (PCA), Recursive Feature Elimination (RFE), and Mutual Information (MI).

- Principal Component Analysis (PCA)

  PCA is a dimensionality reduction technique that transforms the data into a set of principal components that capture the maximum variance in the data. By projecting the data onto a lower-dimensional subspace, PCA reduces the number of features while preserving the most important information.

```
# Apply PCA after reshaping
def apply_pca(data, n_components=N_COMPONENTS):
    data_reshaped = data.reshape(data.shape[0], -1)
    pca = PCA(n_components=n_components)
    data_pca = pca.fit_transform(data_reshaped)
    return data_pca
```

Figure 19: Application of PCA

- Recursive Feature Elimination (RFE)

RFE is a feature selection method that recursively removes the least important features, based on model performance, until a desired number of features is reached. This technique helps in identifying the most significant features in the dataset, which can lead to more efficient models with better generalization.

```
# Apply RFE to select top features
rfe = RFE(estimator=LogisticRegression(), n_features_to_select=N_FEATURES)
X_train_rfe = rfe.fit_transform(X_train, y_train)
X_test_rfe = rfe.transform(X_test)
```

Figure 20: Application of RFE

- Mutual Information

Mutual Information (MI) quantifies the dependency between features and the target variable by measuring how much information is shared between them. It helps identify which features provide the most valuable information for predicting the target variable, making it a powerful tool in feature selection. In audio analysis, MI is particularly useful for determining which audio features are most relevant for classification tasks, potentially enhancing the performance of models like Convolutional Neural Networks (CNNs) by focusing on the most informative features.

```
# Apply Mutual Information for feature selection
mi = mutual_info_classif(data_reshaped, labels_encoded)
top_k_indices = np.argsort(mi)[-MAX_LENGTH:]  # Select top features
```

Figure 21: Application of MI

5.3.4 Handling Imbalanced Data

When the dataset is imbalanced, SMOTE is applied to the training data to generate synthetic samples of the minority class. This helps the model learn to recognize the minority class better and prevents it from being biased toward the majority class.

```
# Apply SMOTE to balance the training data
smote = SMOTE(random_state=42, k_neighbors=2)  # Reduce k_neighbors to 2
X_train_res, y_train_res = smote.fit_resample(X_train_flat, np.argmax(y_train, axis=1))
```

Figure 22: Application of SMOTE

5.4 Model Training

Model training plays a crucial role in developing accurate and robust models to distinguish between real and fake audio samples. The process typically involves feeding the Deep Learning algorithm with the extracted audio features like MFCCs, chroma, spectral centroid, spectral bandwidth, spectral roll off and zero crossing rate from dataset.

5.4.1 Model Selection

Model Selection in this research involves choosing the most appropriate deep learning model architecture that can effectively differentiate between real and fake audio samples. This process requires evaluating multiple models such as **CNN, RNN, LSTM, GAN, and GRU** based on their performance metrics like accuracy, F1-score, recall, and precision. Model selection is critical because different architectures may capture various aspects of the audio data more effectively.

5.4.1.1 CNN

```python
# Define the CNN model
def create_model(input_shape):
    model = Sequential()
    model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', padding='same', input_shape=input_shape))
    model.add(BatchNormalization())
    model.add(MaxPooling2D(pool_size=(2, 1)))  # Pool only along the height
    model.add(Dropout(0.25))

    model.add(Conv2D(64, kernel_size=(3, 3), activation='relu', padding='same'))
    model.add(BatchNormalization())
    model.add(MaxPooling2D(pool_size=(2, 1)))  # Pool only along the height
    model.add(Dropout(0.25))

    model.add(Conv2D(128, kernel_size=(3, 3), activation='relu', padding='same'))
    model.add(BatchNormalization())
    model.add(Dropout(0.25))  # Remove pooling layer here

    model.add(Flatten())
    model.add(Dense(256, activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(2, activation='softmax'))

    model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
    return model
```

Figure 23: Application of CNN method

The code defines a convolutional neural network (CNN) model for image classification. It uses convolutional layers to extract features, batch normalization to improve training, max pooling to reduce computational cost, and dropout to prevent overfitting. The final dense layers classify the input images into two categories. The model is compiled with the Adam optimizer and categorical crossentropy loss function.

5.4.1.2 RNN

```
# Define the RNN model with the correct input shape
def create_model(input_shape):
    model = Sequential()
    model.add(SimpleRNN(128, return_sequences=True, input_shape=input_shape))
    model.add(BatchNormalization())
    model.add(Dropout(0.25))

    model.add(SimpleRNN(128, return_sequences=False))
    model.add(BatchNormalization())
    model.add(Dropout(0.25))

    model.add(Dense(256, activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(2, activation='softmax'))

    model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
    return model
```

Figure 24: Application of RNN method

The code defines a recurrent neural network (RNN) model for sequential data. It uses SimpleRNN layers to process the input sequences, with batch normalization and dropout layers to improve training and prevent overfitting. The final dense layers classify the input sequences into two categories. The model is compiled with the Adam optimizer and categorical crossentropy loss function.

5.4.1.3 LSTM

```
# Define the RNN model creation function
def create_model(input_shape):
    model = Sequential()
    model.add(LSTM(128, return_sequences=True, input_shape=input_shape))
    model.add(BatchNormalization())
    model.add(Dropout(0.25))

    model.add(LSTM(128, return_sequences=False))
    model.add(BatchNormalization())
    model.add(Dropout(0.25))

    model.add(Dense(256, activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(2, activation='softmax'))

    model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
    return model
```

Figure 25: Application of LSTM method

The code defines a recurrent neural network (RNN) model for sequential data. It uses LSTM layers to process the input sequences, with batch normalization and dropout layers to improve training and prevent overfitting. The final dense layers classify the input sequences into two categories. The model is compiled with the Adam optimizer and categorical crossentropy loss function.

5.4.1.4 GAN

```
# Define the GAN components
def build_generator(input_shape):
    model = Sequential()
    model.add(Dense(256, input_dim=input_shape))
    model.add(LeakyReLU(alpha=0.2))
    model.add(BatchNormalization())

    model.add(Dense(512))
    model.add(LeakyReLU(alpha=0.2))
    model.add(BatchNormalization())

    model.add(Dense(1024))
    model.add(LeakyReLU(alpha=0.2))
    model.add(BatchNormalization())

    model.add(Dense(np.prod((MAX_LENGTH, N_MFCC, 1)), activation='tanh'))
    model.add(Reshape((MAX_LENGTH, N_MFCC, 1)))

    return model
```

Figure 26: Application of GAN method

The code defines a generator model for a Generative Adversarial Network (GAN). It takes an input shape as a parameter and uses a series of dense layers with LeakyReLU activations and batch normalization to generate a sequence of data. The final layer reshapes the output to match the desired format, which is specified by MAX_LENGTH, N_MFCC, and 1. The output of this generator is then used as input to the discriminator, which is another component of the GAN.

5.4.1.5 GRU

```
# Define the GRU model
def create_model(input_shape):
    model = Sequential()
    model.add(GRU(128, input_shape=input_shape, return_sequences=True, dropout=0.25))
    model.add(BatchNormalization())

    model.add(GRU(128, return_sequences=True, dropout=0.25))
    model.add(BatchNormalization())

    model.add(GRU(128, dropout=0.25))
    model.add(BatchNormalization())

    model.add(Dense(256, activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(2, activation='softmax'))

    model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
    return model
```

Figure 27: Application of GRU method

The code defines a recurrent neural network (RNN) model using Gated Recurrent Units (GRUs). The model takes a sequence as input and processes it through multiple GRU layers with batch normalization and dropout to improve performance and prevent overfitting. The final layers are dense layers with ReLU activation and softmax activation for classification. The model is compiled with the Adam optimizer and categorical crossentropy loss function.

5.4.2 Enhance Interpretability

Enhancing interpretability through feature selection is a vital aspect of this research as it helps in identifying the most relevant features that contribute to distinguishing between real and fake audio. Feature selection methods like Principal Component Analysis (PCA), Recursive Feature Elimination (RFE), and Genetic Algorithms (GA) play a key role in reducing the dimensionality of the data while retaining the most significant information. Moreover, a more interpretable model is easier to explain and validate, which is crucial in fields like deep fake detection, where understanding the model's decision-making process is as important as its accuracy. This approach leads to a more transparent and trustworthy system that not only performs well but also provides insights into why certain audio samples are classified as fake, aiding in the overall goal of research.

5.4.3 Training Process

The training process in this research is central to developing models that can accurately classify real and fake audio. This process involves Deep Learning algorithm with the preprocessed audio features and adjusting the model's parameters to minimize prediction errors. The training process typically includes several key steps:

- Data Preparation

Before training begins, the dataset is split into training and validation sets, ensuring a balanced representation of both real and fake audio samples. Techniques like SMOTE also applied to handle class imbalance.

- Model Initialization

The chosen deep learning models are initialized with specific architectures and hyperparameters, such as learning rates, batch sizes, and the number of layers.

- Loss Calculation

A loss function, such as cross-entropy, is used to measure the difference between the models prediction and the actual label (real or fake). This loss function guides the optimization process.

- Evaluation

After each epoch, the model's performance is evaluated on the validation set to monitor overfitting and generalization. Metrics like accuracy, F1-score, recall, and precision are tracked to assess performance.

- Iteration

The process is repeated for multiple epochs, where the model continuously learns and improves its ability to classify audio samples.

5.5 Model Evaluation

Model evaluation is a phase that determines the success of Deep Learning algorithms in distinguishing between real and fake audio.

5.5.1 Evaluation Metrics

In this research, the choice of evaluation metrics will influence the interpretation of the model performance. Since the dataset is potentially imbalanced, where the number of real audio samples might significantly outweigh the number of fake ones, metrics like accuracy alone might not provide a complete picture. Instead, recall, and the F1-score are also used in the experiment.

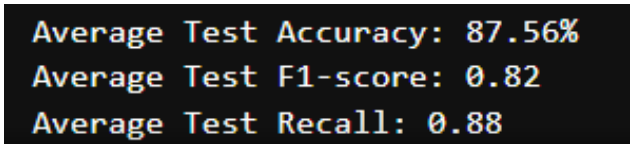

Figure 28: Evaluation Metrics

5.5.2 Cross Validation

Cross validation is another essential component for model evaluation strategies. In this research, applying Stratified K-Fold ensures that each fold maintains the same proportion of real and fake audio samples. By splitting the data into 5 folds and rotating the training and testing phases across these folds, it can significantly reduce the risk of overfitting and obtain a more generalizable model.

```
# Apply Stratified K-Fold Cross-Validation
skf = StratifiedKFold(n_splits=5)
fold_no = 1
accuracies = []
f1_scores = []
recalls = []
```

Figure 29: Application of Stratified K-Fold

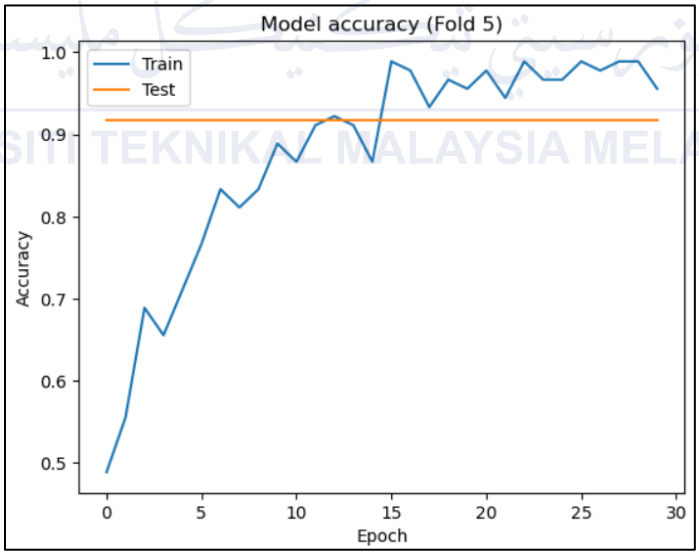

Figure 30: Accuracy Graph of Fold 5

5.6 Conclusion

This section is the most important for understanding the flow of the entire analysis and modifying parts to achieve the goal for this project. This section also discusses the critical function and method applied.

CHAPTER 6: TESTING

6.1 Introduction

This chapter gives a review of the results of implementation that presented in chapter 5. This section elaborates the analysis of classification results and the analysis of the model. The research project mainly analyzes audio dataset using Deep Learning method with some data reduction to enhance the performance of model. This chapter will show the best and highest-performance model results that can analyze whether it is fake or real audio.

6.2 Model Accuracy

Once the training process is complete, the accuracy of the model is being tested. The code extracts MFCC features and uses SMOTE to balance the dataset before processing audio data to distinguish between real and fake files. Then, to guarantee a constant class distribution over folds, it employs stratified K-Fold Cross-Validation. The data is restructured, and labels are encoded in order to train and validate the CNN model over 30 epochs.

Table 8: Accuracy Result of CNN through 30 Epoch

Iteration	CNN		CNN + SMOTE		CNN + SMOTE + Stratified K-Fold	
	Accuracy	Loss	Accuracy	Loss	Accuracy	Loss
Epoch 1	43.63%	3.00	58.41%	2.92	45.93%	4.36
Epoch 2	92.42%	7.09	59.97%	2.02	50.60%	3.84
Epoch 3	90.34%	7.75	71.26%	1.29	55.96%	3.58
Epoch 4	89.30%	6.14	69.55%	0.89	63.98%	1.48
Epoch 5	91.38%	1.23	70.55%	0.85	71.43%	1.11
Epoch 6	62.40%	1.20	70.40%	1.10	62.48%	0.98
Epoch 7	55.62%	1.43	67.14%	0.86	73.10%	0.68
Epoch 8	90.07%	0.47	83.88%	0.39	75.28%	0.68
Epoch 9	90.34%	1.11	74.37%	0.68	86.30%	0.33
Epoch 10	90.34%	1.09	84.66%	0.36	74.89%	0.48
Epoch 11	91.38%	0.54	77.24%	0.56	78.45%	0.36
Epoch 12	87.46%	0.39	89.25%	0.25	86.36%	0.28
Epoch 13	86.15%	0.34	86.75%	0.32	88.98%	0.25
Epoch 14	66.32%	0.58	91.90%	0.18	90.31%	0.21
Epoch 15	86.68%	0.36	92.83%	0.17	87.02%	0.24
Epoch 16	89.30%	0.36	96.03%	0.07	90.48%	0.20
Epoch 17	89.03%	0.31	97.59%	0.07	92.99%	0.17
Epoch 18	90.34%	0.25	97.35%	0.07	97.33%	0.14
Epoch 19	90.60%	0.35	94.70%	0.11	93.32%	0.15
Epoch 20	90.60%	0.23	99.46%	0.03	97.33%	0.09
Epoch 21	90.34%	0.23	97.59%	0.05	96.22%	0.12
Epoch 22	93.46%	0.18	98.52%	0.03	95.60%	0.12
Epoch 23	91.65%	0.21	100%	0.02	95.60%	0.11
Epoch 24	91.65%	0.18	99.07%	0.03	97.33%	0.09
Epoch 25	91.65%	0.19	100%	0.02	97.94%	0.06
Epoch 26	90.60%	0.21	98.28%	0.03	95.60%	0.09
Epoch 27	94.00%	0.19	95.64%	0.09	100%	0.10
Epoch 28	94.00%	0.17	100%	0.01	99.05%	0.03
Epoch 29	92.95%	0.12	98.91%	0.03	98.11%	0.06
Epoch 30	88.26%	0.18	100%	0.00	100%	0.02

Table 9: Accuracy Result of RNN through 30 Epoch

Iteration	RNN		RNN + SMOTE		RNN + SMOTE + Stratified K-Fold	
	Accuracy	Loss	Accuracy	Loss	Accuracy	Loss
Epoch 1	56.15%	0.94	65.66%	0.72	55.33%	0.87
Epoch 2	64.24%	0.74	79.04%	0.46	59.38%	0.87
Epoch 3	85.64%	0.51	88.71%	0.31	81.53%	0.45
Epoch 4	79.64%	0.48	93.70%	0.20	81.89%	0.36
Epoch 5	88.26%	0.37	97.35%	0.17	77.24%	0.47
Epoch 6	91.65%	0.26	97.35%	0.08	90.94%	0.25
Epoch 7	85.64%	0.27	99.46%	0.06	90.77%	0.28
Epoch 8	90.34%	0.18	98.28%	0.04	89.60%	0.24
Epoch 9	91.38%	0.25	97.74%	0.04	95.21%	0.16
Epoch 10	96.34%	0.08	100%	0.01	95.38%	0.12
Epoch 11	98.69%	0.11	100%	0.01	97.69%	0.06
Epoch 12	95.30%	0.12	100%	0.02	95.38%	0.09
Epoch 13	95.04%	0.10	100%	0.01	99.04%	0.69
Epoch 14	100%	0.07	100%	0.02	99.04%	0.69
Epoch 15	97.65%	0.08	100%	0.01	97.30%	0.69
Epoch 16	96.34%	0.09	99.07%	0.01	100%	0.76
Epoch 17	100%	0.05	100%	0.01	96.73%	0.76
Epoch 18	100%	0.05	98.28%	0.03	97.30%	0.76
Epoch 19	100%	0.05	100%	0.00	99.04%	0.04
Epoch 20	100%	0.01	100%	0.00	100%	0.02
Epoch 21	100%	0.02	100%	0.01	98.26%	0.02
Epoch 22	98.69%	0.07	100%	0.01	99.04%	0.02
Epoch 23	100%	0.02	100%	0.00	100%	0.01
Epoch 24	97.65%	0.04	100%	0.00	100%	0.01
Epoch 25	100%	0.02	100%	0.01	100%	0.00
Epoch 26	97.65%	0.04	100%	0.00	100%	0.01
Epoch 27	100%	0.01	100%	0.00	100%	0.01
Epoch 28	100%	0.02	100%	0.00	100%	0.01
Epoch 29	95.30%	0.12	100%	0.00	99.04%	0.02
Epoch 30	100%	0.02	100%	0.00	100%	0.01

Table 10: Accuracy Result of LSTM through 30 Epoch

Iteration	LSTM		LSTM + SMOTE		LSTM + SMOTE + Stratified K-Fold	
	Accuracy	Loss	Accuracy	Loss	Accuracy	Loss
Epoch 1	57.97%	0.64	47.35%	1.12	61.97%	0.73
Epoch 2	7390%	0.56	78.58%	0.39	86.36%	0.31
Epoch 3	81.72%	0.40	91.75%	0.24	84.41%	0.28
Epoch 4	87.99%	0.32	97.20%	0.14	91.20%	0.20
Epoch 5	92.95%	0.22	97.35%	0.08	96.38%	0.18
Epoch 6	94.00%	0.15	95.64%	0.09	98.89%	0.07
Epoch 7	95.30%	0.10	100%	0.03	96.77%	0.07
Epoch 8	94.00%	0.11	100%	0.02	98.89%	0.03
Epoch 9	97.65%	0.06	100%	0.01	99.44%	0.02
Epoch 10	97.65%	0.06	100%	0.02	100%	0.00
Epoch 11	100%	0.02	100%	0.01	100%	0.00
Epoch 12	100%	0.03	100%	0.00	100%	0.00
Epoch 13	97.65%	0.05	100%	0.01	100%	0.00
Epoch 14	100%	0.02	100%	0.01	100%	0.00
Epoch 15	100%	0.01	100%	0.00	100%	0.00
Epoch 16	100%	0.02	100%	0.00	100%	0.00
Epoch 17	100%	0.03	100%	0.00	100%	0.00
Epoch 18	100%	0.01	100%	0.00	100%	0.01
Epoch 19	100%	0.01	100%	0.00	100%	0.00
Epoch 20	100%	0.01	100%	0.00	100%	0.00
Epoch 21	100%	0.00	100%	0.01	100%	0.00
Epoch 22	100%	0.01	100%	0.00	100%	0.00
Epoch 23	100%	0.01	100%	0.01	100%	0.03
Epoch 24	100%	0.01	100%	0.00	100%	0.00
Epoch 25	100%	0.00	100%	0.00	100%	0.00
Epoch 26	100%	0.00	100%	0.00	100%	0.00
Epoch 27	100%	0.00	100%	0.00	100%	0.00
Epoch 28	100%	0.00	100%	0.00	100%	0.00
Epoch 29	100%	0.00	100%	0.00	100%	0.00
Epoch 30	100%	0.00	100%	0.00	100%	0.00

Table 11: Accuracy Result of GAN through 30 Epoch

Iteration	GAN		GAN + SMOTE		GAN + SMOTE + Stratified K-Fold	
	Accuracy	Loss	Accuracy	Loss	Accuracy	Loss
Epoch 1	60.93%	0.99	75.78%	0.50	74.21%	0.40
Epoch 2	63.54%	0.97	69.92%	0.54	62.63%	0.54
Epoch 3	66.40%	0.90	65.78%	0.64	59.63%	0.56
Epoch 4	65.43%	0.90	65.26%	0.62	58.06%	0.58
Epoch 5	65.77%	0.87	63.55%	0.64	56.30%	0.60
Epoch 6	65.28%	0.85	61.47%	0.64	54.84%	0.63
Epoch 7	64.82%	0.84	59.88%	0.64	53.50%	0.63
Epoch 8	62.68%	0.86	59.69%	0.64	52.63%	0.65
Epoch 9	61.47%	0.85	59.04%	0.64	51.99%	0.66
Epoch 10	60.68%	0.85	57.47%	0.65	52.34%	0.63
Epoch 11	60.11%	0.84	56.77%	0.64	52.12%	0.63
Epoch 12	58.10%	0.85	56.53%	0.63	51.61%	0.63
Epoch 13	56.71%	0.84	55.71%	0.63	50.75%	0.63
Epoch 14	55.67%	0.84	54.67%	0.65	49.55%	0.64
Epoch 15	54.67%	0.85	53.82%	0.65	48.95%	0.64
Epoch 16	53.59%	0.86	53.13%	0.64	48.22%	0.65
Epoch 17	53.04%	0.85	52.52%	0.64	47.25%	0.66
Epoch 18	52.00%	0.85	52.03%	0.64	46.97%	0.65
Epoch 19	51.11%	0.84	51.13%	0.64	46.42%	0.65
Epoch 20	49.89%	0.86	50.91%	0.63	45.97%	0.65
Epoch 21	49.53%	0.84	50.45%	0.64	45.60%	0.66
Epoch 22	48.73%	0.85	50.22%	0.63	45.44%	0.65
Epoch 23	48.03%	0.85	49.49%	0.63	44.99%	0.65
Epoch 24	47.32%	0.85	49.01%	0.63	44.51%	0.65
Epoch 25	46.63%	0.85	48.77%	0.63	44.38%	0.65
Epoch 26	46.06%	0.85	48.39%	0.63	44.11%	0.65
Epoch 27	45.58%	0.84	47.75%	0.64	43.98%	0.64
Epoch 28	44.94%	0.85	47.47%	0.64	43.58%	0.65
Epoch 29	44.22%	0.85	47.23%	0.64	43.59%	0.64
Epoch 30	36.46%	0.87	46.79%	0.64	43.33%	0.65

Table 12: Accuracy Result of GRU through 30 Epoch

Iteration	GRU		GRU + SMOTE		GRU + SMOTE + Stratified K-Fold	
	Accuracy	Loss	Accuracy	Loss	Accuracy	Loss
Epoch 1	59.27%	0.79	55.60%	0.89	49.79%	0.89
Epoch 2	71.28%	0.62	58.95%	0.68	58.84%	0.90
Epoch 3	78.33%	0.39	78.43%	0.44	55.61%	0.84
Epoch 4	72.59%	0.59	78.04%	0.42	69.42%	0.66
Epoch 5	83.03%	0.37	78.11%	0.41	73.44%	0.55
Epoch 6	83.29%	0.35	73.59%	0.50	68.08%	0.66
Epoch 7	85.38%	0.29	84.97%	0.38	68.22%	0.57
Epoch 8	89.30%	0.48	89.49%	0.29	70.70%	0.49
Epoch 9	88.26%	0.29	89.18%	0.22	80.61%	0.43
Epoch 10	92.95%	0.23	89.49%	0.22	80.93%	0.48
Epoch 11	87.99%	0.31	90.18%	0.21	77.66%	0.53
Epoch 12	92.69%	0.24	91.51%	0.18	82.88%	0.50
Epoch 13	89.30%	0.24	92.44%	0.20	80.18%	0.42
Epoch 14	93.73%	0.15	92.68%	0.19	95.60%	0.20
Epoch 15	91.65%	0.20	92.92%	0.17	84.06%	0.32
Epoch 16	92.69%	0.27	96.26%	0.08	91.19%	0.18
Epoch 17	95.30%	0.21	95.09%	0.11	89.20%	0.28
Epoch 18	90.60%	0.33	95.48%	0.10	91.51%	0.20
Epoch 19	96.34%	0.13	94.70%	0.13	84.84%	0.27
Epoch 20	89.30%	0.27	96.65%	0.09	90.41%	0.19
Epoch 21	90.34%	0.34	97.20%	0.09	88.07%	0.40
Epoch 22	95.04%	0.13	97.74%	0.09	90.41%	0.33
Epoch 23	91.65%	0.18	94.55%	0.07	86.15%	0.28
Epoch 24	95.04%	0.13	97.59%	0.07	93.64%	0.15
Epoch 25	91.65%	0.18	96.03%	0.09	91.97%	0.18
Epoch 26	96.34%	0.19	97.74%	0.12	89.42%	0.20
Epoch 27	93.73%	0.11	95.33%	0.12	91.37%	0.16
Epoch 28	95.30%	0.10	100%	0.02	95.60%	0.11
Epoch 29	97.65%	0.08	97.74%	0.10	91.73%	0.19
Epoch 30	97.65%	0.08	96.81%	0.05	89.81%	0.17

6.3 Comparison and Analysis of Proposed Method

After all testing and considering have successfully done, the overall comparison is made to ensure which Deep Learning method has high performance. The table below shows the overall result assessment that can be compared among 5 proposed methods which are, CNN, RNN, LSTM, GAN and GRU.

Table 13: Accuracy Result of 5 Proposed Method

METHOD	RESULT (Accuracy)		
	RAW	+ SMOTE	+ STRATIFIED K-FOLD
RNN	76.92%	76.92%	73.33%
CNN	**76.92%**	**84.62%**	**87.56%**
LSTM	76.92%	76.92%	81.03%
GAN	32.81%	51.79%	43.21%
GRU	76.92%	76.92%	64.49%

The table summarizes the results of different machine learning models on a classification task. The accuracy of each model is reported for three different scenarios: without any preprocessing, with SMOTE oversampling, and with stratified k-fold cross-validation. The results show that CNN models consistently outperform other models, with the highest accuracy achieved using SMOTE and stratified k-fold. RNN models, including LSTM and GRU, also perform well, but not as well as CNNs. GANs, on the other hand, have significantly lower accuracy compared to the other models.

Table 14: F1-Score Result of 5 Proposed Method

METHOD	RESULT (F1-Score)		
	RAW	+ SMOTE	+ STRATIFIED K-FOLD
RNN	0.67	0.67	0.74
CNN	**0.67**	**0.81**	**0.81**
LSTM	0.67	0.67	0.78
GAN	0.19	0.12	0.24
GRU	0.67	0.75	0.67

The table summarizes the results of different machine learning models on a classification task using the F1-score metric. The F1-score combines precision and recall evaluating the models performance. The results show that CNN models consistently outperform other models, with the highest F1-score achieved using SMOTE and stratified k-fold. RNN models, including LSTM and GRU, also perform well, but not as well as CNNs. GANs, on the other hand, have significantly lower F1-scores compared to the other models.

Table 15: Recall Result of 5 Proposed Method

METHOD	RESULT (Recall)		
	RAW	+ SMOTE	+ STRATIFIED K-FOLD
RNN	0.77	0.77	0.73
CNN	**0.77**	**0.85**	**0.85**
LSTM	0.77	0.77	0.81
GAN	0.16	0.26	0.22
GRU	0.77	0.77	0.64

The table summarizes the results of different machine learning models on a classification task using the recall metric. Recall measures the models ability to correctly identify positive instances. The results show that CNN models consistently outperform other models, with the highest recall achieved using SMOTE and stratified k-fold. RNN models, including LSTM and GRU, also perform well, but not as well as CNNs. GANs, on the other hand, have significantly lower recall compared to the other models.

Table 16: Result of Best Method + Feature Selection

Method + Feature Selection	Accuracy	F1-Score	Recall
CNN + Recursive Feature Elimination (RFE)	62.44%	0.68	0.62
CNN + Principal Component Analysis (PCA)	**86.03%**	**0.81**	**0.86**
CNN + Mutual Information (MI)	51.15%	0.57	0.51

The table summarizes the results of a CNN model with different feature selection methods on a classification task. The accuracy, F1-score, and recall metrics are reported for models using Recursive Feature Elimination (RFE), Principal Component Analysis (PCA), and Mutual Information (MI). The results show that PCA significantly improves the performance of the CNN model compared to RFE and MI, suggesting that PCA is a more effective feature selection method.

6.4 Conclusion

Finally, the main purpose of this chapter is to present the findings of the analysis and research of the study implement on Deep Learning method, which show that CNN model gives best performance to analyze audio dataset.

CHAPTER 7: CONCLUSION

7.1 Introduction

This chapter would concisely show the entire project including the limitation of the project and clarify the additional development that could be applied in the future in order to have better accuracy and performance.

7.2 Real-World Application of The Proposed Method

The combination of Convolutional Neural Networks (CNN), SMOTE (Synthetic Minority Over-sampling Technique), and Stratified K-Fold Cross-Validation can be applied to a variety of real-world problems beyond deep fake audio detection. In medical image analysis, detecting abnormalities in images such as X-rays, MRIs, or CT scans often involves imbalanced datasets, with fewer abnormal cases compared to normal ones. These techniques can be used to enhance the models ability to identify subtle patterns and impro ve the accuracy of disease detection.

Facial recognition in security systems is another area where these methods are valuable. Security and surveillance systems require accurate identification of individuals, even when datasets have imbalances between different faces. By applying CNNs along with SMOTE and Stratified K-Fold Cross-Validation, the accuracy of facial recognition systems can be improved, particularly in cases with limited data on certain faces.

Object detection in autonomous vehicles also benefits from these techniques. Autonomous vehicles must detect and classify objects such as pedestrians, other vehicles, or road signs in real-time. CNNs can be used to accurately identify and classify these objects, and the application of

SMOTE and Stratified K-Fold Cross-Validation can improve the model's performance, especially in situations where certain objects are underrepresented in the training data.

Sentiment analysis on social media, which involves analyzing text data to detect sentiment, often faces challenges due to imbalanced datasets, where negative sentiments are less common than positive or neutral ones. Although CNNs are primarily used for image data, they can also be applied to text data. When combined with SMOTE and Stratified K-Fold Cross-Validation, the models accuracy in detecting sentiments, particularly negative ones, can be enhanced.

In speech recognition, especially in noisy environments like public places or industrial settings, it is essential to recognize spoken words accurately despite background noise. CNNs can be used to extract and recognize features from audio signals, and the use of SMOTE and Stratified K-Fold Cross-Validation can improve the models robustness and accuracy in challenging conditions.

Anomaly detection in industrial IoT is another application where these techniques are effective. Identifying anomalies in sensor data is crucial for predictive maintenance and fault detection. CNNs can be applied to time-series data from sensors to detect anomalies, and by using SMOTE and Stratified K-Fold Cross-Validation, the detection of rare but critical anomalies can be enhanced.

In environmental monitoring, where data such as air quality, temperature, or pollution levels are monitored, imbalances in datasets are common, as certain conditions are rare. The combination of CNNs, SMOTE, and Stratified K-Fold Cross-Validation can improve the accuracy of detecting and predicting rare environmental conditions, supporting early warnings and better decision-making.

Finally, in fraud detection within financial transactions, analyzing complex patterns in financial data often involves imbalanced datasets. While CNNs are traditionally used for image data, they can also be applied to structured data for pattern recognition. When these techniques are combined, the ability to detect fraudulent transactions can be significantly improved.

7.3 Limitation of The Project

To conduct and complete this research, there are some limitations, and it is listed as below:

1. Results of the proposed method (CNN + SMOTE + Stratified K-Fold + PCA) manage to get lower accuracy and F1-Score due to PCA reduces the dimensionality of the data.
2. Accuracy testing can only be done using one method at a time due to the laptops limited processing power.
3. The large dataset size (4GB) causes the system to take a long time to produce results, especially since it will be analyzed over 30 epochs.
4. The process of loading data into the code and making it understandable is difficult by using audio dataset.

7.4 Future Research

In the future, there will be several upgrades that can be implemented in this project for better performance. Since the combination of CNN with Feature Selection is not compatible, other researchers can improve this research by using other options to reach the necessary target, such as adding ensemble method or treat missing and outlier values . Thus, the researchers also need to try the analyze Deep Learning method with another dataset that is more specific for better accuracy.

7.5 Conclusion

In conclusion, this project successfully achieved its objective of identifying deep learning algorithms that demonstrate high performance in detecting deep fake audio. The findings contribute valuable insights that can be beneficial for forensic teams in enhancing the accuracy and reliability of speech recognition systems when dealing with manipulated audio.

REFERENCES

1. D. Yu and L. Deng, Automatic speech recognition: A deep learning approach. Springer, 2014
2. A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in Acoustics, speech and signal pro-cessing (icassp), 2013 ieee international conference on. IEEE, 2013, pp. 6645–6649
3. M. M. H. Nahid, B. Purkaystha, M. S. Islam, "Bengali speech recognition: A double layered LSTM-RNN approach" ResearchGate, 2020
4. J. Khochare, C. Joshi, B, Yenarkar, S. Suratkar, "A deep learning Framework for Audio Deepfake Detection" ResearchGate, 2021
5. Z. Almutairi, H. Elgibreen, "A review of modern Audio Deepfake Detection methods: Challenges and future directions" 2022
6. Khanjani, Z.; Watson, G.; Janeja, V.P. How deep are the fakes? Focusing on audio deepfake: A survey., 2021
7. Wijethunga R.L., Matheesha D., Noman A., "Deepfake audio detection: A deep learning-based solution for group conversations" 2020
8. M. Malik, M. K. Malik, K. Mehmood, I. Makhdoom, "Automatic speech recognition" 2020
9. Heidari A., Jafri Navimipour N., Dag H., Unal M., "Deepfake detection using deep learning methods" 2023
10. M. Mcuba, A. Singh, R. A. Ikuesan, H. Venter, "Effect of deep learning on deepfake audio" 2020 Heidari
11. Sowjanya A., Mrudula O., "Effective treatment of Imbalanced Datasets in Health Care using Modified SMOTE Coupled with Stacked Deep Learning Algorithms" 2023
12. Dablain D., Krawczyk B., Chawla N. V., "DeepSMOTE: Fusing Deep Learning and SMOTE for Imbalanced Data" 2023
13. Buyukkececi M., Okur M. C., "A Comprehensive Review of Feature Selection and Feature Selection Stability in Machine Learning" 2023