

DEEPFAKE GUARDIAN: A SOLUTION FOR DETECTING MEDIA MANIPULATION

Dissertation-II

Submitted in partial fulfilment of the requirements for the degree of

Master of Computer Applications

in

Department of Computer Applications

by

RESHMAA S

23MCA0326

Under the Guidance of

Dr Selva Rani B

Associate Professor, Senior SCORE

School of Computer Science Engineering and Information Systems VIT, Vellore.



VIT[®]
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

April 2025

DEEPFAKE GUARDIAN: A SOLUTION FOR DETECTING MEDIA MANIPULATION

Dissertation-II

Submitted in partial fulfilment of the requirements for the degree of

Master of Computer Applications

in

Department of Computer Applications

by

RESHMAA S

23MCA0326

Under the Guidance of

Dr Selva Rani B

Associate Professor, Senior SCORE

School of Computer Science Engineering and Information Systems VIT, Vellore.



VIT[®]

Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

April 2025

DECLARATION

I hereby declare that the **PMCA699J - Dissertation-II** entitled “**Deepfake Guardian: A Solution for Detecting Media Manipulation**” submitted by me, for the award of the degree of Master of Computer Applications in Department of Computer Applications, School of Computer Science Engineering and Information Systems to VIT is a record of bonafide work carried out by me under the supervision of Dr Selva Rani B, Associate Professor Senior, SCORE, VIT, Vellore.

I further declare that the work reported in this dissertation has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place: Vellore

Date: 16.04.2025



Signature of the Candidate

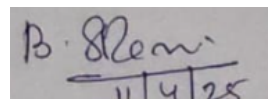
CERTIFICATE

This is to certify that the **PMCA699J - Dissertation-II** entitled “**Deepfake Guardian: A Solution for Detecting Media Manipulation**” submitted by RESHMAA S (23MCA0326), SCORE, VIT, for the award of the degree of Master of Computer Applications in Department of Computer Applications, is a record of Bonafide work carried out by her under my supervision during the period, 13. 12. 2024 to 16.04.2025, as per the VIT code of academic and research ethics.

The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university. The dissertation fulfils the requirements and regulations of the University and in my opinion meets the necessary standards for submission.

Place: Vellore

Date: 16.04.2025

A handwritten signature in dark ink, appearing to read 'B. S. Ram', with the date '11/4/25' written below it.

Signature of the VIT-SCORE – Guide

Internal Examiner

External Examiner

Head of the Department
Department of Computer Application

ACKNOWLEDGEMENTS

It is my pleasure to express with a deep sense of gratitude to my **Dissertation-II** guide **Dr. Selva Rani B, Associate Professor Senior, School of Computer Science Engineering and Information Systems**, Vellore Institute of Technology, Vellore for her constant guidance, continual encouragement, in my endeavor. My association with her is not confined to academics only, but it is a great opportunity on my part to work with an intellectual and an expert in the field of **Deep Learning**.

"I would like to express my heartfelt gratitude to Honorable Chancellor **Dr. G Viswanathan**; respected Vice Presidents **Mr. Sankar Viswanathan, Dr. Sekar Viswanathan**, Vice Chancellor **Dr. V. S. Kanchana Bhaaskaran**; Pro-Vice Chancellor **Dr. Partha Sharathi Mallick**; and Registrar **Dr. Jayabarathi T.**

My whole-hearted thanks to Dean **Dr. Daphne Lopez**, School of Computer Science Engineering and Information Systems, Head, Department of Computer Applications **Dr. E Vijayan**, MCA Project Coordinator **Dr. Karthikeyan P**, SCORE School Project Coordinator **Dr. Thandeeswaran R**, all faculty, staff and members working as limbs of our university for their continuous guidance throughout my course of study in unlimited ways.

It is indeed a pleasure to thank my parents and friends who persuaded and encouraged me to take up and complete my dissertation successfully. Last, but not least, I express my gratitude and appreciation to all those who have helped me directly or indirectly towards the successful completion of the dissertation.

Place: Vellore

Date: 16.04.2025



RESHMAA S

EXECUTIVE SUMMARY

Embarking on this dissertation journey has been both exciting and enriching, offering a unique opportunity to bridge academic theory with real-world applications of deep learning. The rapid advancement of AI-generated media has raised serious concerns about the authenticity of digital content. Deepfakes, highly realistic yet artificially generated images, videos, and audio pose significant threats to personal privacy, digital trust, and online security. To address this, the project introduces a backend-powered deepfake detection system designed to identify manipulated media using deep learning techniques across various formats. The detection framework adopts a modular architecture suited for different media types. For image detection, EfficientNet is used to identify subtle visual anomalies. Video analysis involves frame extraction followed by processing with the Xception model to detect tampering artifacts. In the case of audio, speech signals are converted into Mel spectrograms and analysed through a convolutional neural network enhanced with an attention mechanism, enabling the detection of nuanced inconsistencies in vocal patterns.

This work contributes to the broader mission of promoting digital content authenticity and combating misinformation. The model training process utilized categorical cross-entropy as the loss function, the Adam optimizer, and a dynamic learning rate scheduler. Performance was evaluated using key metrics such as accuracy, precision, recall, and F1-score.

	CONTENTS	PAGE NO.
	Declaration	i
	Certificate	ii
	Acknowledgements	iii
	Executive Summary	iv
	Table of Contents	v
	List of Figures	viii
	List of Tables	ix
1	INTRODUCTION	1
	1.1 Motivation for the project	2
2	LITERATURE REVIEW	3
	2.1 Literature Survey	3
3	PROBLEM DEFINITION AND SCOPE	11
	3.1 Problem Statement	11
	3.1.1 Goals and Objectives	12
	3.2 Scope of Project	13
	3.3 Proposed Work	13
	3.4 Gap Identification	14
	3.5 Hardware Resource Required	15
	3.6 Software Resource Required	16
4	BACKGROUND ON DEEPFAKE TECHNOLOGY	17
	4.1 Definition and Evolution	17
	4.1.1 Definition of Deepfake	17
	4.1.2 Evolution of Deepfake	18
	4.2 Tools used for Deepfake Creation	19
	4.3 Deepfake Scenario Examples and their Implications	20
5	SYSTEM REQUIREMENTS SPECIFICATIONS	24
	5.1 Introduction	24

5.1.1	Purpose and Scope of Document	24
5.1.2	Use Case View	24
5.2	System Specifications	25
5.2.1	Functional Requirements	25
5.2.2	Non-Functional Requirements	25
6	PROJECT PLAN	26
6.1	Project Schedule	26
7	DETAILED DESIGN DOCUMENT	29
7.1	System Architecture	29
7.2	Model Architecture	30
7.2.1	Audio Model Architecture	30
7.2.2	Image Model Architecture	31
7.2.3	Video Model Architecture	32
8	PROJECT IMPLEMENTATION	33
8.1	Tools and Technologies used	33
8.1.1	Research Study	33
8.1.2	UML Tools	34
8.1.3	Programming Tools	35
8.1.4	Dataset	35
8.1.5	IDE	36
8.1.6	Libraries	37
8.2	Algorithm Details	39
8.2.1	Dataset Details	39
8.2.2	Preprocessing Details	40
8.2.3	Model Details	41
8.2.4	Model Training Details	42
8.2.5	Model Prediction Details	43
8.3	Performance Metrics	44

9	RESULT AND DICUSSION	48
9.1	Screenshots	48
9.1.1	Audio Deepfake Detection	48
9.1.2	Image Deepfake Detection	55
9.1.3	Video Deepfake Detection	61
10	CONCLUSION AND FUTURE WORK	67
10.1	Conclusion	67
10.2	Future Work	67
11	REFERENCES	69

List of Figures

Figure No.	Title	Page No.
3.1	Current deepfake media types and detection techniques	11
4.1	Timeline of Deepfakes	18
4.2	An example for Style-GAN images	19
4.3	Deepfake images for political manipulation	20
4.4	Climatical change misinformation	21
4.5	Identity Fraud	23
5.1	Use Case Diagram	24
6.1	Project Plan	26
7.1	System Architecture	29
7.2	Audio Model Architecture	30
7.3	Image Model Architecture	31
7.4	Video Model Architecture	32
8.1	Google Scholar	33
8.2	StarUML	34
8.3	Luchidchart	35
8.4	Kaggle Dataset	35
8.5	Kaggle Notebook	36
8.6	(a)original Mel-freq audio, (b) cloned Mel-freq audio	40
8.7	Data Augmentation Techniques	40
8.8	Extracted frames from video	41
8.9	Confusion Matrix	46
8.10	ROC Curve	46
8.11	AUC Curve	46

List of Tables

Table No.	Title	Page No.
2.1	Literature Survey	3
3.1	Hardware Requirements	15

CHAPTER 1

INTRODUCTION

The rapid expansion of social media platforms and the increasing accessibility of smart devices have significantly transformed how individuals communicate, share information, and engage with digital content. Platforms like Facebook, Instagram, Twitter, and Snapchat have become essential for global connectivity, enabling users to express their thoughts, opinions, and personal experiences instantly. The widespread use of smartphones, tablets, and internet-connected devices has further simplified the process of capturing and sharing media in real time. This digital transformation has empowered individuals to become content creators, fostering a culture of online engagement and self-expression.

However, along with these advancements, the emergence of deepfake technology has introduced significant challenges. Deepfakes are AI-generated synthetic media that manipulate images, videos, audio, and text using sophisticated deep learning models. Techniques such as convolutional neural networks (CNNs), recurrent neural networks (RNNs), generative adversarial networks (GANs), variational autoencoders (VAEs), and diffusion models (DMs) enable the creation of hyper-realistic content that can be difficult to distinguish from authentic media. While deepfake technology has legitimate applications in entertainment, accessibility, and education, its misuse poses serious threats, including misinformation, identity theft, fraud, and cyberattacks. Malicious actors can exploit deepfake techniques such as face-swapping, voice cloning, and lip-syncing for deceptive purposes, undermining trust in digital media.

Recognizing the risks associated with deepfakes, it is crucial to develop effective detection mechanisms to mitigate potential harm. Governments must address security concerns, industries need to safeguard their operations, and individuals should be aware of the implications of AI-generated content. Cybercriminals increasingly leverage deepfakes for phishing attacks, financial fraud, and social engineering, making the need for advanced detection systems more urgent. This project aims to develop an advanced deepfake detection system using deep learning models to identify manipulated content across multiple modalities, including images, audio, and video. The system employs a CNN-based approach to detect facial inconsistencies in images, while speech signals are transformed into Mel spectrograms and analyzed using a CNN with an attention mechanism to identify artificial voices. For video frame detection, a fine-tuned Xception model is applied, which captures spatial features and patterns to detect deepfake manipulations. The detection system is trained on diverse datasets

and evaluated through performance metrics such as accuracy, confusion matrices, and classification reports to ensure reliability. By combining multiple detection techniques, this project provides a scalable and effective solution to counter deepfake threats, enhance digital media authenticity, and reinforce trust in online content.

1.1 Motivation for the Project:

The evolution of artificial intelligence has led to the widespread creation of deepfake media, including manipulated images, videos, and audio. While these advancements have contributed to innovation in entertainment and creative industries, they also pose serious risks, such as misinformation, identity theft, and cyber threats. The increasing realism of deepfakes makes it difficult to distinguish between authentic and synthetic content, raising concerns about digital security, media credibility, and public trust.

This project is driven by the urgent need for an effective deepfake detection system capable of analysing and verifying media authenticity. Conventional detection methods struggle to keep pace with continuously evolving deepfake techniques, making it necessary to develop more sophisticated solutions. By utilizing deep learning models, including CNNs, Xception model, and attention mechanisms, this research aims to enhance detection accuracy for images, audio, and videos. Adopting this approach, this project seeks to strengthen digital forensics, protect online platforms from manipulated content, and assist in media verification. Through analysis and advanced AI-driven detection, this research contributes to preserving digital integrity, ensuring transparency, and reinforcing trust in online content.

CHAPTER 2

LITERATURE REVIEW

2.1 Literature Survey:

Table 2.1 Literature Survey

S.No	Title	Merits	Demerits
1	Video and Audio Deepfake Datasets and Open Issues in Deepfake Technology: Being Ahead of the Curve	The paper offers a clear overview of deepfake detection datasets for images, videos, and audio, helping researchers stay up to date. It explores challenges and research opportunities for enhancing detection accuracy and robustness. Additionally, it ties deepfake technology to real-world applications, making it valuable for both academic study and practical use.	The paper addresses a wide range of topics, which may limit in-depth exploration of specific areas. It might overlap with existing reviews, offering minimal new insights. The discussion on challenges and future research is broad and lacks detailed examples or practical solutions.
2	Deepfake Detection: A Systematic Literature Review	This presents an up-to-date and thorough review of Deepfake detection by analysing 112 articles from 2018 to 2020. It classifies detection methods into distinct categories, offering useful insights for researchers. It also evaluates the performance of various techniques, highlighting that	It mainly covers research from 2018 to 2020, which may exclude newer developments. The general categorization might miss specific approaches or hybrid methods. Additionally, while it identifies deep learning as the top-performing technique, the paper could provide more detailed

		deep learning-based methods are the most effective.	comparisons and actionable recommendations.
3	Deepfake video detection: challenges and opportunities	The paper offers a thorough examination of the challenges in detecting deepfake videos, including issues related to data, training, and reliability. It emphasizes the leading role of deep learning methods in detection while acknowledging their limitations, providing useful insights for future research. The need for high-quality datasets is stressed, and the paper highlights key research gaps advancements in detection technology.	While the paper addresses the limitations of deep learning methods, it could offer more concrete solutions for overcoming these challenges. It mainly discusses theoretical issues and lacks focus on practical approaches for real-time detection. Additionally, the emphasis on deep learning might overshadow other potentially effective detection methods.
4	DeepFake Detection for Human Face Images and Videos: A Survey	Comprehensive summary of DeepFake detection techniques for face images and videos, detailing their methodologies, results, and performance. It classifies DeepFake creation methods and examines dataset trends and their improvements. Additionally, it explores the challenges involved in both creating and detecting DeepFakes, contributing	Deeper comparisons between the various detection methods to highlight their strengths and weaknesses. It focuses mainly on theoretical aspects, lacking practical solutions for real-world deployment. Furthermore, it may not fully consider newer detection approaches that have not yet been widely implemented.

		valuable insights for developing more robust detection models.	
5	A Survey on the Detection and Impacts of Deepfakes in Visual, Audio, and Textual Formats	The study provides a thorough overview of deepfakes across various media types, exploring their impact and reviewing current detection methods. It emphasizes the need for adaptable, real-time solutions and highlights the importance of combining technology with public awareness and policy.	The paper could benefit from practical case studies and a more focused exploration of specific deepfake types, particularly text-based ones. It also lacks a deeper analysis of emerging detection technologies and their challenges.
6	Audio Deepfake Approaches	Extensive review of techniques used for creating and detecting audio deepfakes. It compares various detection methods, such as SVMs, DTs, CNNs, and RNNs, and provides performance metrics like accuracy, Equal Error Rate (EER), and t-DCF, making it valuable for understanding the effectiveness of different approaches.	Primarily focuses on algorithmic comparisons and does not explore the practical application of these methods in real-world scenarios. Additionally, it may not fully address the influence of external factors, such as dataset quality, on the performance of the detection techniques. Furthermore, the results are mainly based on specific studies, which could limit their broader applicability.

7	An Investigation into the Utilisation of CNN with LSTM for Video Deepfake Detection	Examines the use of hybrid CNN-LSTM models for video deepfake detection, emphasizing the enhanced accuracy achieved by combining spatial and temporal analysis. It provides a detailed review of the latest techniques, compares CNN-LSTM models to other detection methods, and highlights key factors that affect detection performance, offering valuable insights for both researchers and cybersecurity experts.	Despite its strengths, the paper does not offer a detailed analysis of the optimal configurations for real-world implementation. Additionally, it falls short in addressing the limitations of existing datasets and the scalability of CNN-LSTM models, which could affect their generalizability to diverse applications.
8	A Review of Modern Audio Deepfake Detection Methods: Challenges and Future Directions	Detailed review of audio deepfake detection methods, focusing on imitation and synthetic-based deepfakes. It compares detection methods and datasets, highlighting the tradeoff between accuracy and scalability. The paper also identifies key research gaps and points toward future developments in the field.	Primarily focuses on imitation and synthetic-based deepfakes, limiting its scope. Additionally, while it discusses future research directions, it lacks specific solutions to address the existing challenges.

9	Comparison of Deepfake Detection Techniques through Deep Learning	<p>This paper emphasizes the significance of detecting deepfakes, especially for facial recognition systems used in digital forensics. It compares several face-detection classifiers, including Custom CNN, VGG19, and DenseNet-121, using augmented datasets. The findings show that VGG19 achieves the highest accuracy of 95%, and data augmentation enhances model performance while minimizing costs.</p>	<p>The paper primarily focuses on face-detection methods and does not explore deepfake detection for other types of media. The research could be expanded by considering additional models and datasets to validate the results across various deepfake content types.</p>
10	A Survey on Deepfake Video Detection	<p>Potential risks posed by deepfake videos, particularly in the context of manipulating public opinion and targeting public figures. It emphasizes the neutrality of the technology itself but acknowledges the threat its misuse poses to societal stability. The mention of research focusing on detection methods and benchmarks shows that efforts are underway to address this issue, while the call for further advancements highlights the</p>	<p>It lacks specific examples or details about the limitations of current detection methods. It mentions the need for further advancements but does not provide a clear direction for improving these methods or overcoming existing challenges. Additionally, the passage does not discuss potential ethical concerns, such as privacy or bias, related to the detection of deepfakes in real-world scenarios.</p>

		need for more robust solutions for real-world applications.	
11	The DeepFake Detection Challenge (DFDC) Dataset	The DFDC dataset stands out as one of the largest publicly available face-swapped video datasets, consisting of over 100,000 clips from 3,426 actors, showcasing a variety of manipulation methods. Ethical concerns are addressed as all participants consented to the use of their likenesses. Models trained on this dataset have proven effective in detecting deepfakes in real-world scenarios. The Kaggle challenge also promotes innovation, with the paper offering valuable insights from top model submissions.	Despite its size, deepfake detection remains a tough challenge, limiting the reliability of current models. The dataset primarily focuses on face-swapping techniques, excluding other forms of deepfakes like audio or full-body manipulation. While models trained on DFDC show generalization, their ability to handle evolving deepfake methods is uncertain. Additionally, demographic biases may exist due to the paid actors used in the dataset, and the lack of non-manipulated real-world videos complicates model evaluation.
12	A Comparative Study of Deepfake Video Detection Method	This highlights the challenge of detecting deepfake content, which is becoming increasingly difficult to distinguish from real media due to advances in deep learning. It mentions various detection techniques, including visual, local, deep, and	This lacks specific details on the effectiveness or limitations of the mentioned detection approaches. While it addresses the challenges posed by evolving deepfake technology, it does not provide examples or evidence of how current

		temporal feature-based methods, offering a broad view of the approaches used. The evolving nature of deepfake technology is also recognized, emphasizing the need for continuous advancements in detection methods to keep up with more realistic manipulations.	algorithms tackle these issues. Additionally, the abstract does not mention potential biases or limitations in the datasets used to train the detection models, which could affect their real-world performance.
13	The Effect of Deep Learning Methods on Deepfake Audio Detection for Digital Investigation	This paper examines the risks of voice cloning technology and evaluates deepfake audio detection methods. It compares various deep learning techniques using features like MFCC, Mel-spectrum, and spectrogram, with custom architectures showing better results. The findings offer useful insights for distinguishing synthetic and real voices.	The paper lacks details on the limitations of the detection methods and does not address ethical concerns or data biases. It also could provide a more thorough analysis of deep learning techniques in real-world conditions.
14	A Deep Learning Framework for Audio Deepfake Detection	This paper tackles audio deepfake detection using the Fake or Real (FoR) dataset and compares two approaches: feature-based with machine learning algorithms and image-based with deep learning models like TCN and STN. The	The paper doesn't address the limitations of the dataset or challenges in real-world applications. It focuses mainly on TCN, lacking comparison with other deep learning models and offering little insight into scalability or real-time use.

		results show that TCN outperforms traditional machine learning methods, achieving high accuracy, offering a strong solution for classifying audio deepfakes.	
15	Deep Fake Detection using Neural Networks	This paper proposes a deepfake video detection system using neural networks, combining dense and convolutional layers for binary classification. The results show promising accuracy.	The paper does not discuss the limitations of the proposed system, such as its generalization to real-world, noisy video data or the computational requirements.

CHAPTER 3

PROBLEM DEFINITION AND SCOPE

3.1 Problem Statement:

The rise of deepfake technology has made it possible to create extremely realistic but fake images, videos, and audio. These manipulated media files are becoming more difficult to identify, leading to growing concerns about their potential misuse. Deepfakes can be used for harmful purposes such as spreading false information, creating fake political events, and damaging people's reputations. As social media platforms continue to grow, the spread of deepfakes becomes an even more significant issue.

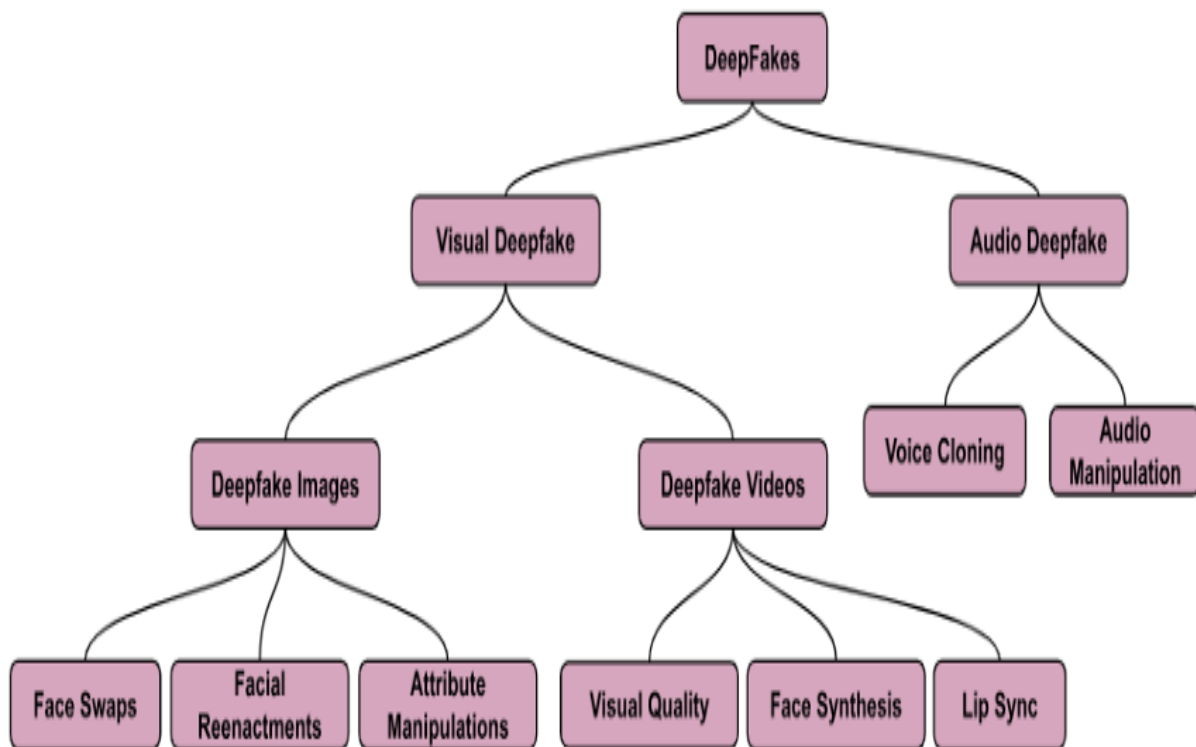


Figure 3.1 Current deepfake media types and detection techniques

With deepfake technology becoming more accessible, the risks associated with these AI-generated media increase. The manipulation of images, videos, and audio has reached a level where it is nearly impossible for people to tell what is real and what is fake. Deepfakes have been used in several harmful ways, including fake news, blackmail, and defamation. The ability to detect these types of content in real-time is crucial to protecting individuals and organizations from potential harm. Developing reliable detection tools is now more important than ever. While creating deepfakes is becoming easier, detecting them remains a major challenge due to the advanced AI techniques involved. This project focuses on creating an effective deepfake detection system that can identify manipulated content in images, videos, and audio files. By using modern AI models, the system aims to reduce the spread of misinformation and help protect digital media integrity.

3.1.1 Goals and Objectives:

The primary objective of this project is to develop an advanced AI-based system capable of detecting deepfakes in images, audio, and video. With deepfake technology advancing rapidly, it has become increasingly difficult to distinguish between genuine and manipulated media. This system aims to enhance the trustworthiness of digital content by applying deep learning methods to analyse different media types. For image detection, a Convolutional Neural Network (CNN) is employed to examine facial features and spot signs of manipulation. In audio detection, speech signals are converted into Mel spectrograms and processed using a CNN and an attention mechanism to differentiate between authentic and synthetic voices. For video analysis using video frames, the project uses a fine-tuned Xception model to capture spatial features and patterns to identify deepfake alterations. The system will be trained using labelled datasets and evaluated based on performance metrics such as accuracy, confusion matrices, and classification reports. Real-time detection capabilities will also be integrated, enabling users to upload media for instant analysis. The goal of this project is to improve detection precision, enhance the system's adaptability to evolving deepfake techniques, and ensure its scalability for applications in fields such as news verification, social media oversight, and forensic investigations.

3.2 Scope of the Project:

There are many tools available for creating the deep fakes, but for deep fake detection there is hardly any tool available. This project covers a broad spectrum of deepfake detection by integrating image, audio, and video analysis into a unified system. It utilizes deep learning models such as CNNs and attention mechanisms to detect tampered content with high accuracy. Additionally, the system's performance is rigorously evaluated using classification metrics such as accuracy, precision, recall, and F1-score to ensure reliable detection. With its adaptability and efficiency, this project offers a robust solution for identifying manipulated digital content and strengthening public trust in media.

3.3 Proposed Work:

This project proposes a deepfake detection system that processes images, audio, and video efficiently, while keeping computational demands low. For image-based detection, the proposed solution uses a pre-trained EfficientNet model, chosen for its lightweight architecture and ability to accurately detect subtle inconsistencies in facial features, textures, and lighting, all with minimal computational overhead. For video analysis, frames are extracted at regular intervals and processed using the proposed fine-tuned Xception model, which identifies anomalies and manipulation traces without requiring heavy computational resources. Temporal inconsistencies across frames are also detected in a way that minimizes processing time. In the case of audio detection, the proposed approach converts speech into Mel spectrograms and analyses them using a CNN with an attention mechanism, enabling the system to focus on crucial variations in pitch, tone, and spectral features, indicative of synthetic speech, all while maintaining efficiency. By combining these optimized models, the proposed system provides an effective solution for deepfake detection with minimal computational effort, ensuring scalability and practical application in real-world scenarios.

3.4 Gap Identification

Although deepfake detection technology has seen significant progress, multiple critical gaps continue to hinder the development of fully robust and efficient systems. Addressing these challenges is essential for creating reliable and practical detection solutions. The major gaps identified are outlined below:

1. Absence of Multi-Modal Deepfake Detection (Image, Video, and Audio)

Current deepfake detection models are generally specialized for a single type of media—either images, videos, or audio rather than handling multiple formats simultaneously. However, in real-world applications, deepfakes can appear across various media types. The lack of integrated, multi-modal detection systems limits the adaptability and effectiveness of existing approaches. Developing a unified framework capable of analysing images, video streams, and audio signals together is crucial for a more comprehensive and practical deepfake detection solution.

2. Challenges in Achieving Computational Efficiency

Many sophisticated deepfake detection models are computationally intensive, demanding high-end hardware for effective training and real-time analysis. This dependency on significant computational power restricts the deployment of detection systems on common consumer devices like smartphones and personal computers. There is an urgent requirement for models that strike a balance between lightweight architecture and high detection accuracy, making them accessible and practical for broader user adoption without compromising performance.

3. Complexity in Video-Based Deepfake Detection and its Optimization

Deepfake detection in videos introduces additional challenges compared to static images, primarily due to the complexity of temporal patterns and increased data volume. Conventional methods that process entire video files result in high computational costs and long training times. An optimized approach involves extracting key frames from videos and treating them as standalone images for analysis. By focusing on individual frames, it becomes possible to

maintain detection accuracy while significantly reducing computational demands, making the system more scalable and efficient for video-based applications.

4. Shortcomings in Existing Audio Attention Mechanisms

In the field of audio deepfake detection, many current strategies treat the spectrogram uniformly, potentially overlooking critical localized features that differentiate real from synthetic audio. Without focused attention mechanisms, subtle but important anomalies in synthetic voices may go undetected. Enhancing detection models with targeted attention strategies can help isolate and prioritize these key audio patterns, improving the model's ability to accurately identify manipulated audio content.

3.5 Hardware Resources Required:

This project requires a computer with sufficient processing capacity to handle image, audio, and video data effectively. The system should ensure smooth processing of multimedia files for accurate analysis.

Table 3.1 Hardware Requirements

Sr. No.	Parameter	Minimum Requirement
1	Intel Core i7-11700K	3.5 GHz
2	RAM	16 GB
3	Hard Disk	512 GB
4	Graphic Card	NVIDIA RTX 3070 or 3080

3.6 Software Resources Required:

Platform:

1. Operating System: Windows 7+
2. Programming Language: Python 3.7+
3. Libraries: TensorFlow, Librosa, OpenCV

CHAPTER 4

BACKGROUND ON DEEPPAKE TECHNOLOGY

4.1 Definition and Evolution:

4.1.1 Definition of Deepfake:

Deepfake technology is an advanced artificial intelligence application that utilizes deep learning to generate highly realistic synthetic media, including images, videos, and audio. The term "deepfake" is formed by combining "deep learning" and "fake," signifying its ability to produce highly realistic but artificially generated digital content. At its core, deepfake models primarily depend on Generative Adversarial Networks (GANs), which consist of two competing neural networks: a generator that fabricates synthetic media and a discriminator that evaluates its authenticity. Through continuous refinement, the generator enhances the quality of its output, making it nearly indistinguishable from real media.

This technology has found applications in various domains. In the entertainment industry, it is used for creating AI-generated movie scenes, visual effects, and realistic voiceovers. It also serves accessibility purposes, helping individuals with speech impairments by producing synthetic voices that closely resemble their natural speech patterns. Additionally, deepfake technology is being explored for educational applications, such as bringing historical figures to life through AI-generated videos. Despite its potential benefits, deepfake technology also raises ethical concerns, including the spread of misinformation, identity theft, and cybersecurity threats. It has been misused for political deception, fraudulent activities, and impersonation. In response, researchers and technology firms are developing deepfake detection algorithms, digital watermarking techniques, and regulatory policies to counter its misuse. The rapid advancement of deepfake technology underscores the importance of responsible AI development, ethical guidelines, and robust detection mechanisms to ensure its positive impact outweighs its potential risks.

4.1.2 Evolution of Deepfake:

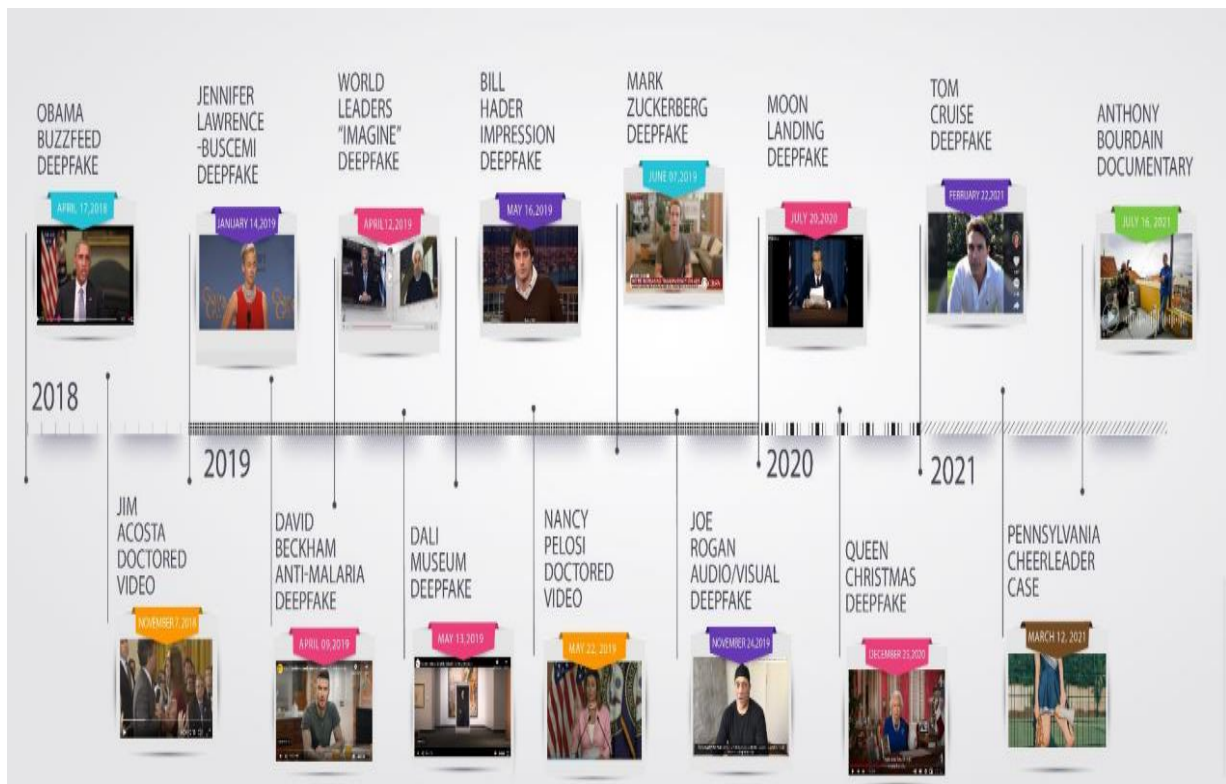


Figure 4.1 Timeline of Deepfakes

Deepfake technology has rapidly evolved from basic digital editing methods to sophisticated AI-driven systems. Initially, media manipulation relied on manual tools like Photoshop and CGI, requiring significant expertise. However, a breakthrough occurred in 2014 with the introduction of Generative Adversarial Networks (GANs) by Ian Goodfellow, allowing AI to generate highly realistic synthetic media through continuous learning. By 2017, deepfake techniques advanced significantly, making face-swapping and voice synthesis more seamless and accessible. The rise of open-source tools like DeepFaceLab and FakeApp further contributed to its widespread use, leading to both creative applications and ethical concerns, such as misinformation and identity fraud. Recent advancements have made deepfake videos and audio nearly indistinguishable from real media, increasing the need for detection mechanisms. AI-based solutions using Convolutional neural networks (CNNs), Recurrent Neural Networks (RNNs), and Attention Mechanisms are being developed to identify manipulated content. Additionally, governments and tech organizations are introducing regulatory frameworks and digital verification methods to curb malicious use. As deepfake technology continues to evolve, ensuring its ethical application while leveraging its benefits remains a key challenge.

4.2 Tools Used for Deepfake Creation:

In recent years, deep learning has made significant advancements in fields like computer vision and robotics. One of the most notable areas of interest is digital face image and video manipulation, which leverages Generative Adversarial Networks (GANs) to create highly realistic outputs. However, GANs still face challenges in generating high-resolution content with precise control over different elements. Achieving disentanglement where distinct features can be modified independently remains complex. Without additional techniques such as regularization, ensuring proper separation of attributes in GAN-generated content is difficult.

Mobile applications like ZAO, Auto FaceSwap, and FaceApp have made it easier for general users to create manipulated media, contributing to the widespread circulation of deepfakes. Many GAN-based face-swapping videos have been shared on platforms like YouTube, demonstrating the growing popularity of this technology. Face swapping, which involves transferring a face from a source image to a target image to create a seamless and natural result, heavily relies on GANs.



Figure 4.2 An example for Style-GAN images

The use of GANs for face synthesis, face-swapping, reenactment, and attribute manipulation is rapidly expanding. Models such as StyleGAN, StyleGAN2, and StyleGAN2-ADA have significantly improved the realism of generated images, making them nearly indistinguishable to the human eye. However, some limitations persist. StyleGAN, for instance, struggles to modify features like skin tone or eye size without affecting other facial attributes. Similarly, BigGAN has difficulty altering specific characteristics, such as a dog's fur colour or length, without unintentionally changing other aspects of the image.

4.3 Deepfakes Scenario Examples and Their Implications:

The rapid advancement of deepfake technology has introduced both opportunities and significant threats across various sectors, including national security, law enforcement, commerce, and society. Deepfakes, which use artificial intelligence to manipulate images, audio, and video, have the potential to mislead the public, influence political and legal proceedings, and facilitate fraudulent activities. The following scenarios illustrate how deepfakes can be misused to create misinformation, incite violence, and manipulate evidence, posing serious ethical and security concerns.

1. Deepfakes and Political Manipulation Fueling Civil Unrest:



Figure 4.3 Deepfake images for political manipulation

One of the most alarming risks posed by deepfake technology is its ability to incite violence and disrupt public order. A fabricated video depicting a high-ranking law enforcement official endorsing police brutality or discriminatory practices could easily spread through social media platforms, provoking mass protests and unrest. Such a scenario could be particularly

dangerous in regions with existing social tensions, where deepfake videos could be strategically released to deepen public mistrust in government institutions. Malicious actors, including political adversaries and extremist groups, could exploit this technology to create convincing yet entirely false narratives. This scenario underscores the potential of deepfakes to erode public trust, destabilize governance, and manipulate democratic processes.

2. Climate Change Misinformation Altering Global Policies:

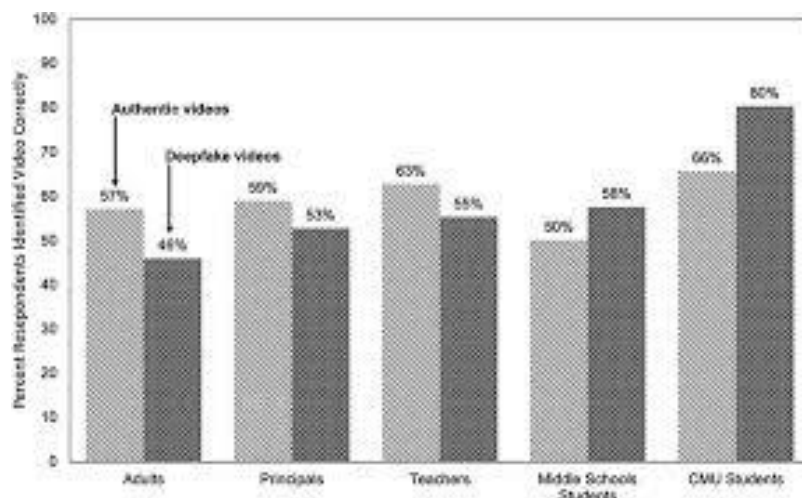


Figure 4.4 Climatical change misinformation

Another area of concern is the use of deepfakes to manipulate scientific data and global policy decisions. Imagine a scenario where AI-generated satellite imagery is used to falsify environmental conditions, such as depicting ice sheet growth in Antarctica to downplay the severity of climate change. Countries or organizations with vested interests in fossil fuels and industrial activities might leverage such synthetic data to challenge international climate agreements, delay policy implementation, or justify their reluctance to adopt sustainable practices. In this case, even though independent scientific institutions may have authentic data to counter these claims, the spread of deepfake-generated misinformation could create confusion, hinder consensus, and weaken the urgency of addressing climate change. This example demonstrates how deepfake technology can be weaponized to manipulate evidence, disrupt policymaking, and influence global economic and environmental strategies.

3. Virtual Kidnapping and Deepfake Ransom Schemes:

The emergence of deepfake technology has also revolutionized cybercrime, particularly in the realm of virtual kidnapping and extortion scams. Criminal organizations can now generate highly realistic deepfake videos of individuals appearing to be kidnapped or in distress, even though the person is completely safe and unaware of the fraudulent scheme. The perpetrators obtain images, videos, and voice recordings of the target from social media or hacked devices, then use AI models to synthesize a convincing "proof of life" ransom video. The victim's family, believing the distressing footage to be real, may be coerced into paying a ransom without verifying the authenticity of the situation. This exploitation of synthetic media demonstrates how deepfake technology can enable non-physical yet highly effective criminal activities, making it increasingly difficult to distinguish between genuine threats and digital deception.

4. Deepfake Evidence Manipulation in Legal Proceedings:

The judicial system relies heavily on video surveillance, forensic analysis, and biometric evidence to determine the guilt or innocence of individuals. However, deepfake technology presents a severe challenge to legal integrity, as it enables criminals to fabricate compelling alibi footage. A defendant accused of a serious crime, such as murder or fraud, could submit a deepfake-generated video as evidence, depicting them at a different location during the time of the offense. By altering timestamps, modifying facial expressions, and integrating synthetic voices, such a deepfake video could create reasonable doubt, ultimately obstructing justice. Furthermore, deepfakes could also be used to frame innocent individuals by generating false confessions or fabricated surveillance footage, leading to wrongful convictions. The rise of deepfake evidence manipulation highlights the urgent need for forensic deepfake detection tools and stricter digital verification protocols in the legal system.

5. Deepfake Identity Fraud and Financial Scams:



Figure 4.5 Identity Fraud

Deepfake technology has also become a powerful tool for identity theft, financial fraud, and corporate espionage. Cybercriminals can use AI to clone the faces and voices of executives, politicians, or public figures, tricking employees or financial institutions into authorizing fraudulent transactions. For instance, a scammer could create a deepfake video of a company's CEO instructing the finance department to wire large sums of money to an offshore account, exploiting employees' trust in their leadership. Similarly, fraudsters could use deepfake technology to impersonate individuals in loan applications, government benefit claims, or social engineering attacks, causing severe financial and reputational damage. The increasing sophistication of deepfake-generated fraud necessitates the development of advanced AI-driven detection systems, enhanced security protocols, and public awareness initiatives to prevent identity-based crimes.

CHAPTER 5

SYSTEM REQUIREMENTS SPECIFICATIONS

5.1 Introduction:

5.1.1 Purpose and Scope of Document:

This document presents a structured plan for the development of a deepfake detection system utilizing deep learning techniques. The document outlines the system's functionality, scope, including use case diagrams, data flow diagrams, functional and non-functional requirements. Additionally, it describes the development approach and the evaluation metrics that will be used to assess the system's performance. The proposed system utilizes EfficientNet for image analysis, Xception for video frame detection, and a CNN with an attention mechanism for audio processing, ensuring precise and computationally efficient deepfake detection across different media types.

5.1.2 Use Case View:

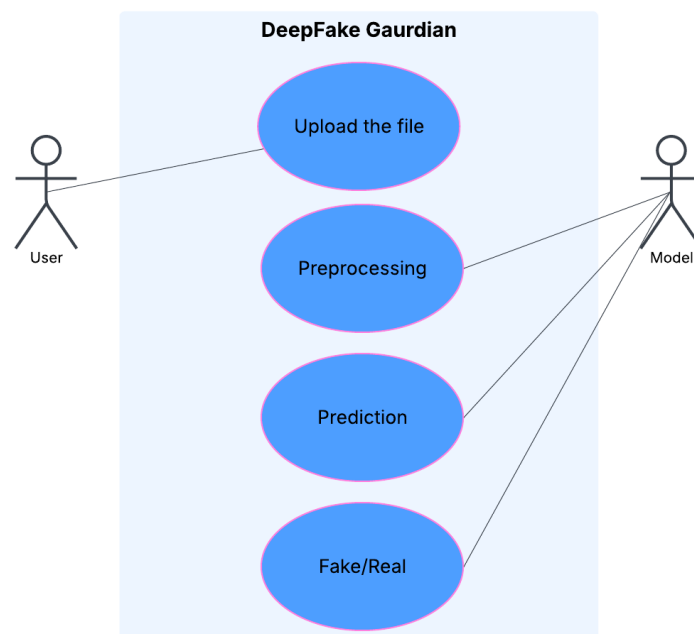


Figure 5.1 Use Case Diagram

5.2 System Specifications:

5.2.1 Functional Requirements:

- **Multi-Format Deepfake Detection:** The system is designed to handle multiple types of media (images, audio, and video) using advanced deep learning models. For image analysis, it utilizes the EfficientNet model, while for audio, it applies a CNN with an attention mechanism. For video, a fine-tuned Xception model is employed for frame-based detection, ensuring high accuracy across all modalities.
- **Video Frame Analysis:** For video content, frames are extracted at regular intervals, and each frame undergoes analysis using the Xception model to identify potential manipulation artifacts and spatial inconsistencies. The system also examines temporal inconsistencies between consecutive frames to improve the overall detection of video deepfakes.
- **Audio Detection via Spectrograms:** Audio content is first converted into Mel spectrograms, which are then processed using a CNN combined with an attention mechanism. This approach focuses on detecting irregularities in pitch, tone, and spectral features indicative of deepfake audio.

5.2.2 Non-Functional Requirements:

- **Scalability:** The system should be able to handle increasing volumes of users and large media files efficiently, ensuring that performance remains stable even as usage grows.
- **Reliability:** The system must perform consistently, ensuring accurate detection of deepfakes with minimal errors over time.
- **Performance Optimization:** The system should balance the need for fast results with maintaining high accuracy in deepfake detection, even for large media files.

CHAPTER 6

PROJECT PLAN

6.1 Project Schedule:

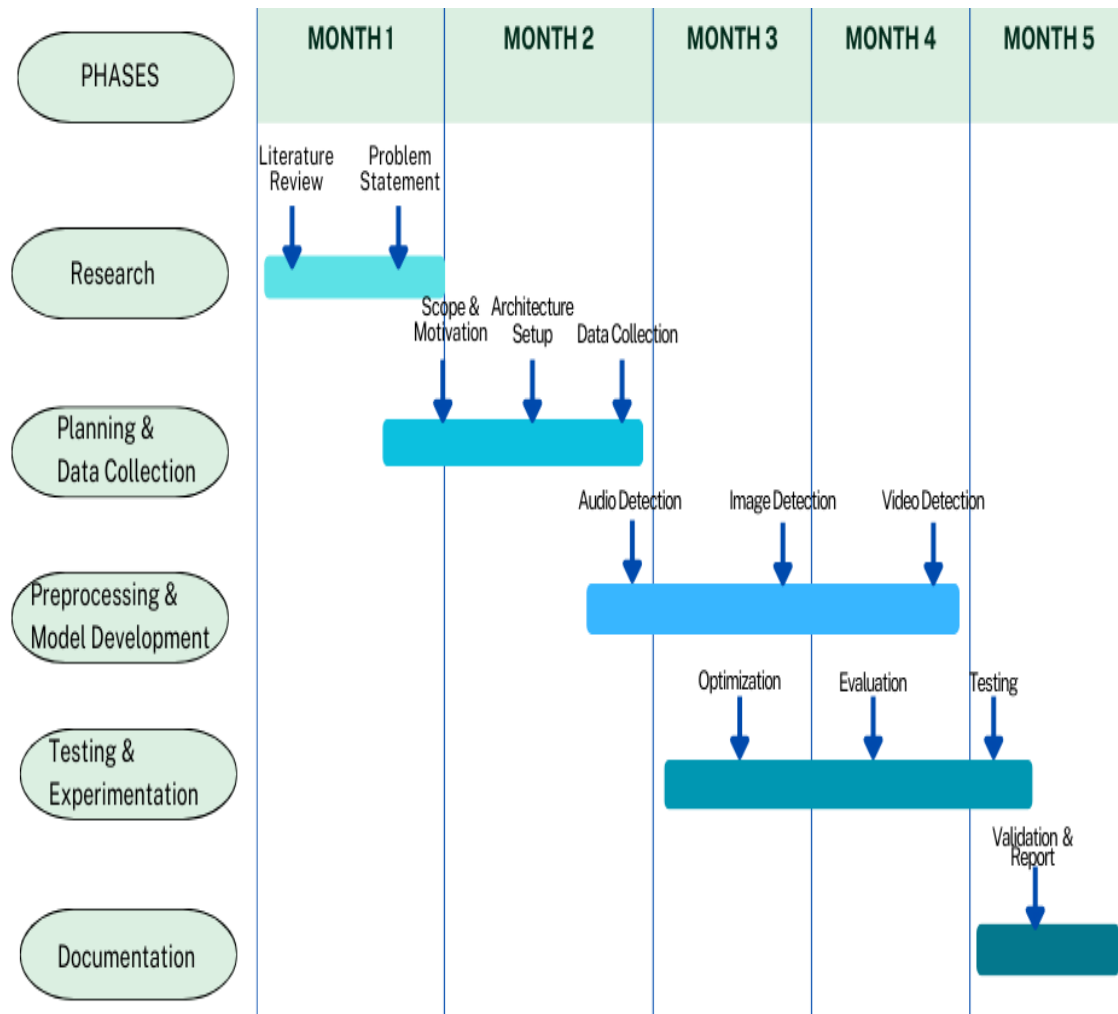


Figure 6.1 Project Plan

Phase 1: Research (December 2024)

The project will begin with an in-depth study of deepfake detection techniques for images, videos, and audio. Existing research papers, methodologies, and detection models will be analyzed to understand their strengths and limitations. The best datasets, including Celeb-DF for videos, will be selected for training and testing. The findings from this phase will help define the project's technical approach and key focus areas.

Phase 2: Planning & Data Collection (December 2024 – January 2025)

A structured plan will be developed for implementing the detection models. The data required for training will be collected, including images, videos, and audio files. Video datasets will be split into individual frames, and audio will be converted into Mel spectrograms to highlight key sound features. The collected data will be reviewed to ensure quality and consistency before moving to preprocessing.

Phase 3: Preprocessing & Model Development (January – March 2025)

In this phase, raw data will be cleaned and processed to improve model performance. Techniques such as data augmentation (flipping, noise addition, time-stretching) will be applied to enhance the dataset. After preprocessing, deep learning models will be developed for each media type. EfficientNet will be used for image detection, Xception for analyzing video frames, and a CNN with Attention for audio detection. The models will be trained with optimized settings to achieve high accuracy.

Phase 4: Testing & Experimentation (March – April 2025)

The trained models will be tested against deepfake samples to assess their detection accuracy. Advanced deepfake techniques will be used to evaluate their robustness. Performance metrics such as accuracy, precision, recall, F1-score, and AUC-ROC will be analyzed. Further refinements, including hyperparameter tuning and optimization techniques, will be applied to enhance detection speed and reliability.

Phase 5: Documentation (April 2025)

The final phase will focus on compiling the project's findings into a structured report. This will include details on the methodology, results, challenges, and conclusions. Any remaining issues will be resolved before finalizing the system. A presentation will be prepared to demonstrate the effectiveness of the deepfake detection models.

CHAPTER 7

DETAILED DESIGN DOCUMENT

7.1 System Architecture:

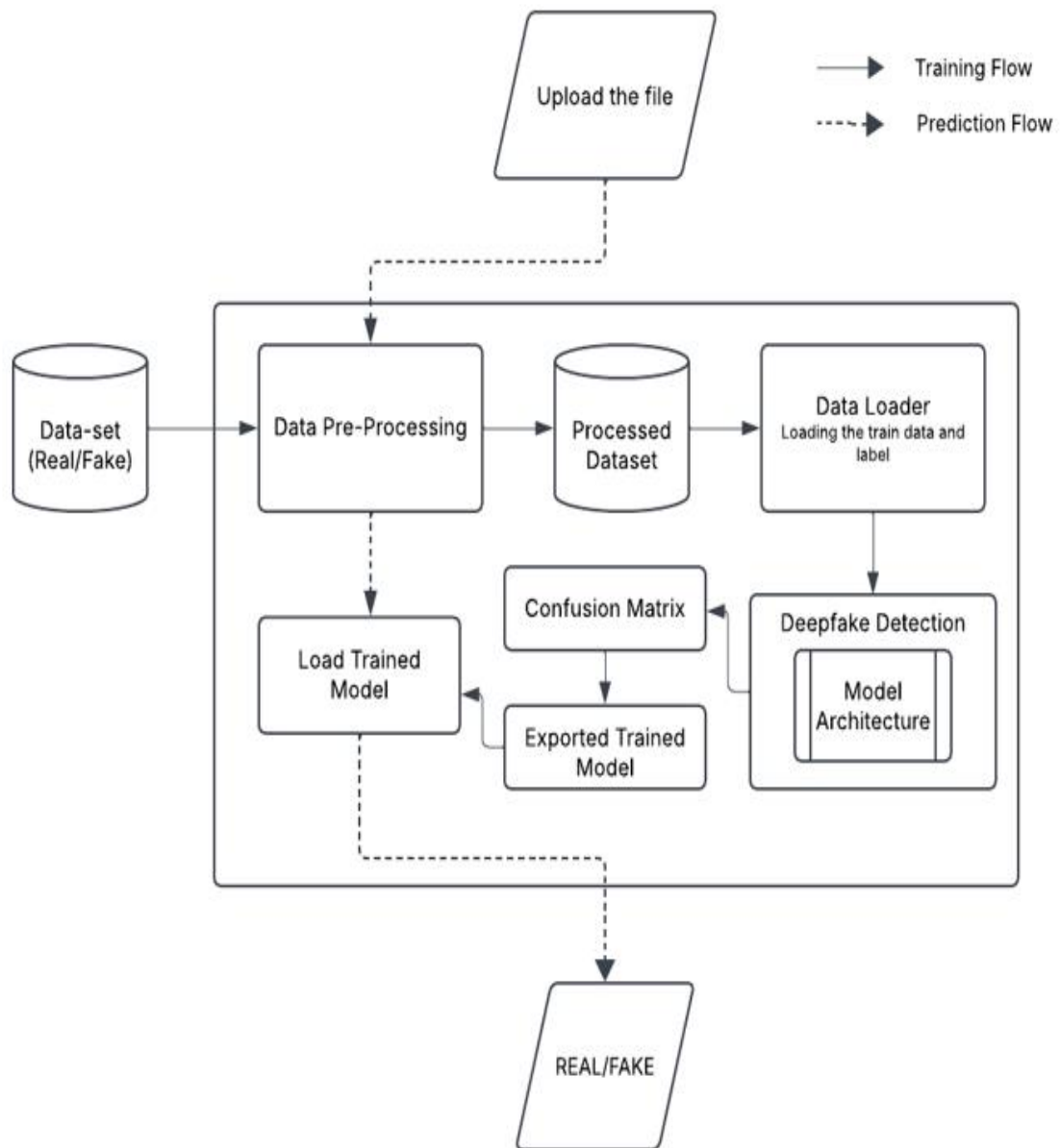


Figure 7.1 System Architecture

The DeepFake Guardian system is structured to efficiently detect manipulated media across images, videos, and audio. It begins with a dataset containing both real and fake samples, which undergoes pre-processing to extract essential features and ensure consistency. The processed dataset is then managed by a data loader, which organizes and labels the data for model training. The system incorporates specialized deepfake detection models: EfficientNet-B0 for image classification, CNN for analyzing video frames, and CNN+Attention for audio spectrograms. During training, model performance is assessed using a confusion matrix, and the trained model is exported for real-world implementation. In the prediction phase, when a user uploads media, it undergoes pre-processing before being analysed by the trained model. The final classification determines whether the input is real or fake, ensuring a structured and efficient deepfake detection workflow. The architecture distinctly separates training and prediction processes, enhancing scalability and adaptability for practical deployment.

7.2 Model Architecture:

7.2.1 Audio Model Architecture:

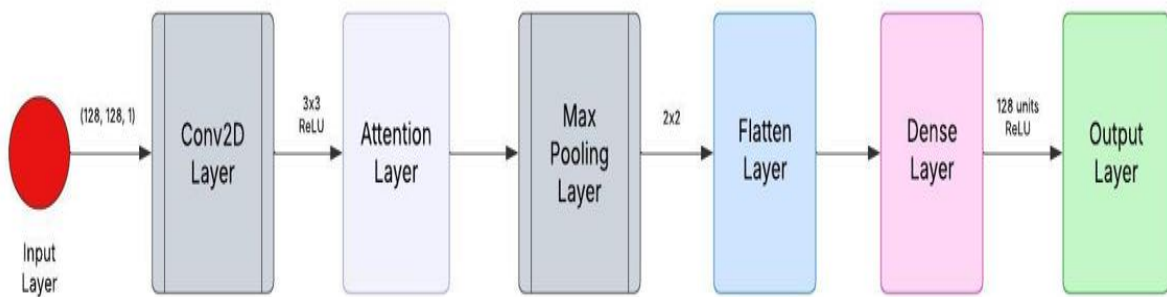


Figure 7.2 Audio Model Architecture

To detect deepfake audio, the process begins by converting raw audio signals into Mel-spectrograms, enabling CNN models to analyse the data in image form. The Mel-spectrograms are generated using Librosa, resized to 128×128 pixels, and stored as images. The next step involves loading the dataset, where real and fake audio files in .wav format are processed, converted into spectrograms, and assigned labels (0 for real, 1 for fake). The data is then reshaped for compatibility with CNN models by adding a single-channel dimension

(128,128,1) and normalized between 0 and 1 for stability during training. A CNN with an Attention Mechanism is used, incorporating three convolutional layers to extract key features, MaxPooling to reduce dimensionality, and attention blocks to highlight critical patterns. Fully connected layers with Dropout (0.3) help prevent overfitting, while a final sigmoid activation function determines whether the audio is real or fake. The model is compiled using Adam optimizer and Binary Cross-Entropy Loss, trained over 20 epochs with a batch size of 32, and validated to monitor performance. Finally, the model is tested on unseen data to evaluate its accuracy in detecting fake audio.

7.2.2 Image Model Architecture:

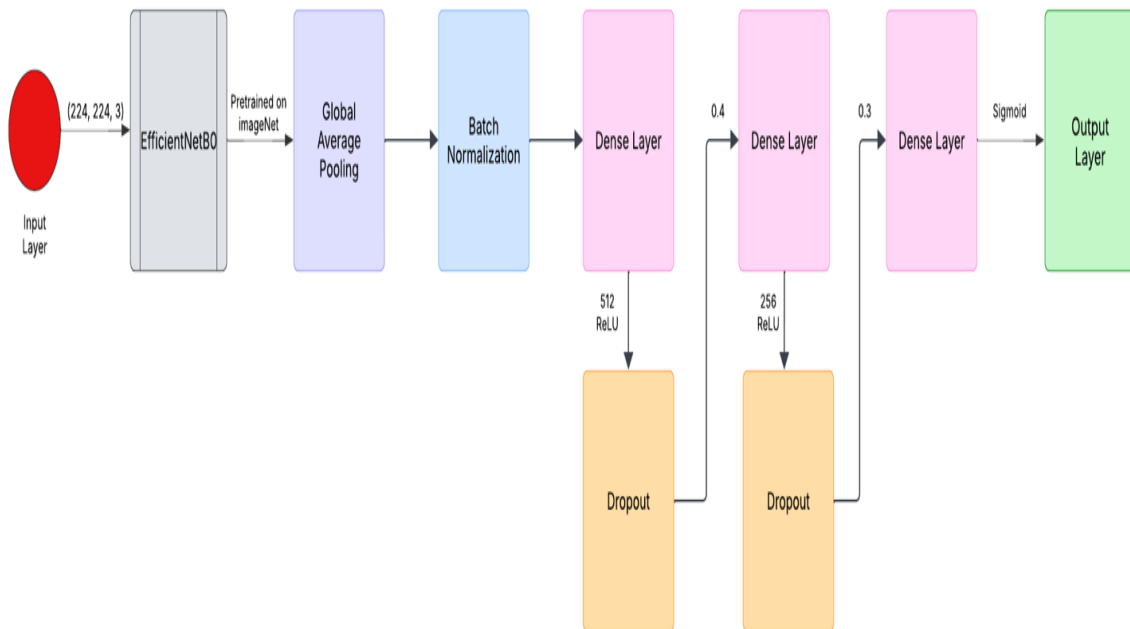


Figure 7.3 Image Model Architecture

The DeepFake detection model follows a systematic approach to ensure accurate classification. It begins with preprocessing, where images are resized to 224×224 pixels, normalized to a [0,1] range, and augmented using transformations like flipping and zooming to enhance model robustness. EfficientNetB0, a pretrained convolutional neural network, extracts essential features such as texture variations, lighting inconsistencies, and facial structures. These features are processed through fully connected layers, utilizing Global Average Pooling for dimensionality reduction, batch normalization for stable learning, and dropout (50%) to minimize overfitting. The final sigmoid activation layer assigns a probability score, with outputs closer to 1 indicating fake and 0 indicating real. The model undergoes

training with Binary Cross-Entropy Loss, using the Adam optimizer for efficient weight adjustments. Its performance is then evaluated using metrics like the Confusion Matrix, Precision, Recall, F1-score, and ROC-AUC Score, ensuring a reliable system that effectively distinguishes real from manipulated media.

7.2.3 Video Model Architecture:

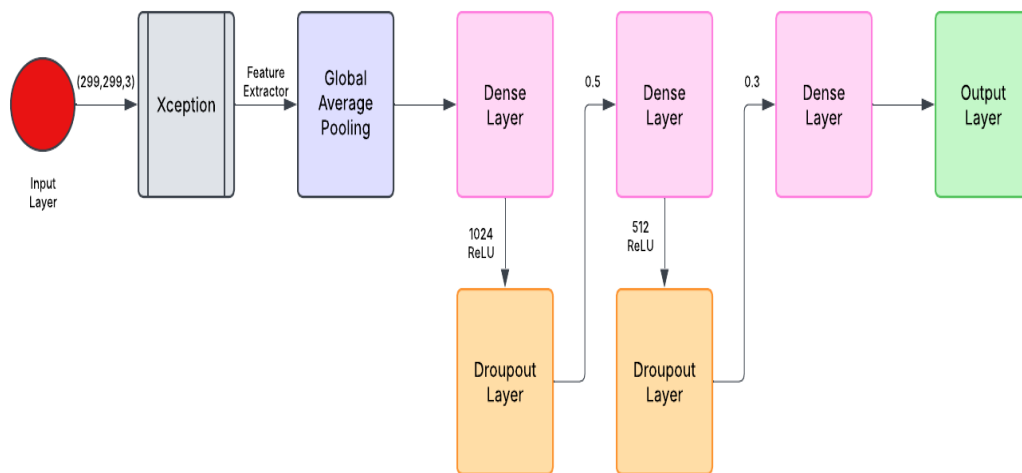


Figure 7.4 Video Model Architecture

The process begins by preprocessing the training, validation, and test datasets using ImageDataGenerator, applying several augmentation techniques such as rotation, zoom, and shifting for the training data. The Xception model, pre-trained on ImageNet, serves as the base model. Following this, a GlobalAveragePooling2D layer is added, followed by two fully connected Dense layers with ReLU activation and Dropout for regularization. The final layer is a Dense layer with a sigmoid activation function for binary classification (Real vs. Fake). The model is compiled using the binary cross-entropy loss function and Adam optimizer. It is trained for 30 epochs with callbacks for early stopping and adaptive learning rate reduction. Finally, the model is tested on unseen video data to evaluate its accuracy by converting into frames to detect fake video or video frame.

CHAPTER 8

PROJECT IMPLEMENTATION

8.1 Tools and Technologies Used:

8.1.1 Research Study:

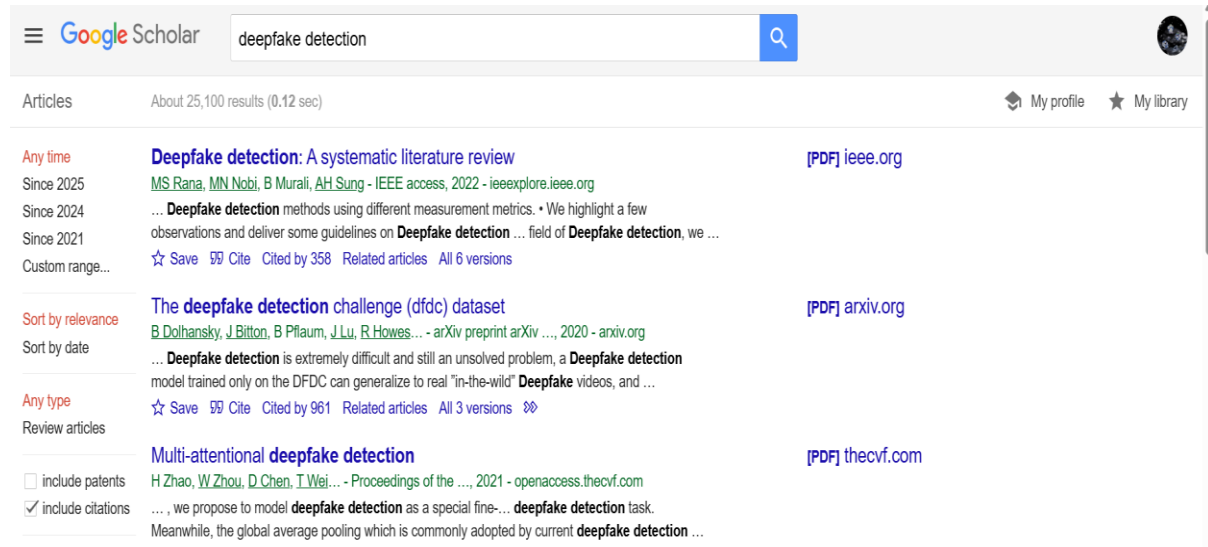


Figure 8.1 Google Scholar

1. Google Scholar:

It was extensively used to perform background research and gather existing knowledge related to deepfake detection. It provided access to a vast collection of peer-reviewed articles, conference papers, technical reports, and theses from around the world. By leveraging Google Scholar, the study maintained a strong academic foundation, ensuring the methods adopted were supported by credible research. It also helped in understanding the strengths and limitations of various existing deepfake detection techniques, guiding the development of an improved solution.

8.1.2 UML Tools:

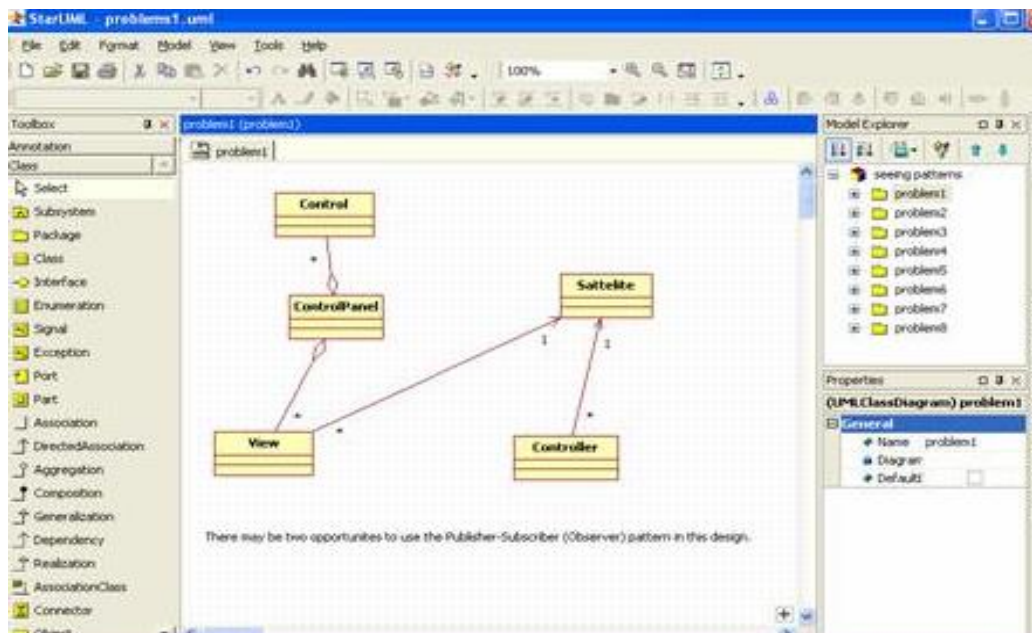


Figure 8.2 StarUML

1. StarUML:

It is a free and flexible diagramming tool, was utilized for creating various Unified Modeling Language (UML) diagrams required during the project. It enabled the design of system architectures, process flows, data flow diagrams, and interaction diagrams. The intuitive drag-and-drop functionality of staruml allowed quick construction of complex diagrams, aiding in better communication of the project structure and workflow across different development stages.

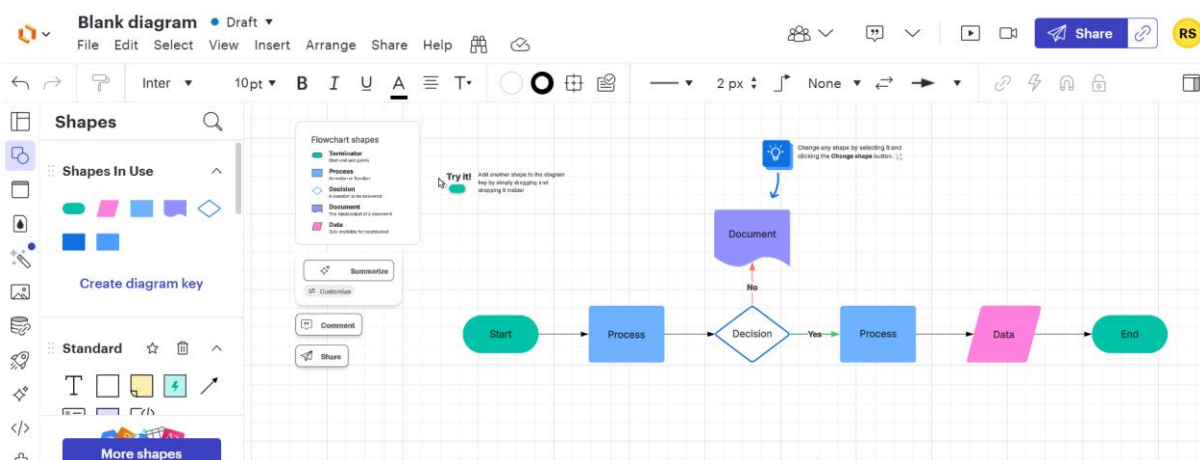


Figure 8.3 Luchidchart

2. Luchidchart:

Served as another vital tool for modeling system designs and data flows. With its professional-grade diagramming capabilities and easy integration with collaborative platforms, Lucidchart was highly useful for generating detailed and clean architectural diagrams. It was particularly helpful for depicting complex relationships between system components, model training workflows, and data pipelines, enhancing the clarity and precision of project documentation.

8.1.3 Programming Tools:

1. Python:

It was chosen as the core programming language for the project due to its simplicity, readability, and extensive support for artificial intelligence and machine learning development. Libraries such as TensorFlow and Keras were used for building deep learning models, OpenCV was applied for video and image processing tasks, and Librosa facilitated audio feature extraction. Python's versatility and large community ecosystem accelerated the model development process, helping efficiently implement, train, and validate deepfake detection models across multiple data types.

8.1.3 Dataset:

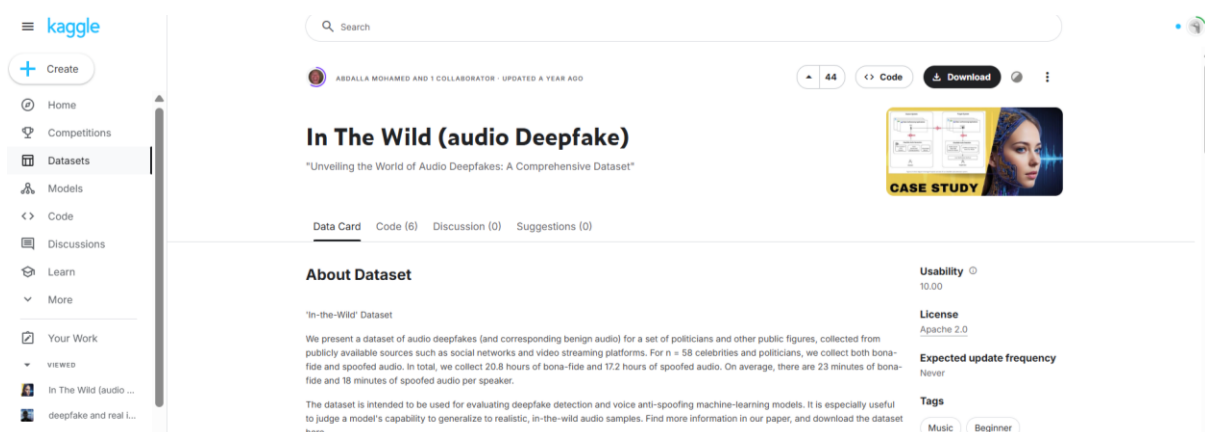


Figure 8.4 Kaggle Dataset

1. Kaggle:

It provided an essential resource for sourcing high-quality datasets required for model training and evaluation. Datasets including manipulated images, synthesized videos, and altered audio clips were retrieved from Kaggle's public repositories. Access to diverse and labeled datasets helped ensure that the models could learn intricate patterns and generalize well to unseen manipulations. Additionally, Kaggle's data competitions and forums offered valuable insights into best practices for handling real-world deepfake datasets.

8.1.4 IDE:



Figure 8.5 Kaggle Notebook

1. Kaggle Notebook:

Used as the primary Integrated Development Environment (IDE) for writing, testing, and executing the code. The platform's built-in support for free GPU and TPU usage significantly reduced model training times and facilitated handling computationally intensive tasks like deep learning. Kaggle Notebooks also allowed seamless sharing, real-time collaboration, and version control, ensuring smooth management of different model versions and experiments throughout the development cycle.

8.1.6 Libraries:

1. os:

The os module is part of Python's standard library and serves as a bridge between Python programs and the operating system. It lets you perform tasks like creating, removing, and moving files and directories. You can also retrieve information about your environment, like the current working directory or environment variables. It's very useful when you need your program to interact with the system's file structure or run system-level commands directly from Python.

2. Numpy:

Numerical Python is a library for scientific computing. It provides a fast and flexible way to work with arrays and matrices of numeric data. It supports a huge range of mathematical operations, like linear algebra, Fourier transforms, and random number generation. numpy arrays are more efficient and powerful than Python's built-in lists, especially for large datasets or heavy mathematical computations.

3. tensorflow:

tensorflow is an end-to-end open-source platform developed by Google that allows you to build and deploy machine learning and deep learning models. It offers various levels of abstraction, so you can quickly build and train models or dive deeper into the model internals if needed. TensorFlow supports both CPU and GPU computation and can scale from a single device to clusters of devices.

4. librosa:

librosa is a Python library specifically created for analyzing and processing audio signals. It provides tools to extract useful information from audio files like tempo, beat tracking, pitch detection, and even music genre classification features. It is widely used in music information retrieval and audio machine learning projects.

5. `cv2` (OpenCV):

`cv2` is the Python interface for OpenCV, a highly efficient library for computer vision tasks. OpenCV (Open-Source Computer Vision Library) enables tasks like image processing, object detection, face recognition, and even real-time video analysis. It supports a variety of programming languages and runs efficiently across different platforms.

6. `matplotlib.pyplot`:

`matplotlib.pyplot` is a state-based interface to the Matplotlib library, designed to make plotting easy. It mimics MATLAB's plotting style and allows users to create static, animated, and interactive visualizations in Python. With `pyplot`, you can generate various types of plots by writing simple code.

7. `seaborn`:

`seaborn` is a data visualization library which is built on top of `matplotlib`. It introduces a higher-level interface for creating attractive and informative statistical graphics. Seaborn is particularly good at creating complex plots easily, such as heatmaps, violin plots, and pair plots. It also integrates well with pandas DataFrames, making it easier to visualize datasets.

8. `tqdm`:

`tqdm` is a very handy library that provides a visual progress bar for Python loops and processes. It's particularly useful when running long loops, data downloads, or model training processes. The progress bar automatically updates itself with information like percentage completed, time elapsed, and estimated remaining time.

9. `sklearn` (scikit-learn):

scikit-learn, often imported as `sklearn`, is one of the most popular Python libraries for machine learning. It provides efficient tools for data analysis and building models. It covers a wide range of tasks like classification, regression, clustering, dimensionality reduction, and model evaluation.

8.2 Algorithm Details:

8.2.1 Dataset Details:

- The Release in the Wild dataset is a carefully compiled collection of audio recordings featuring both authentic (bona fide) and artificially generated (deepfake) speech from 58 well-known individuals, including politicians and celebrities. The dataset is sourced from publicly accessible platforms such as social media and video streaming websites. It comprises a total of 38 hours of audio, with 20.8 hours of real speech and 17.2 hours of synthetically altered speech. On average, each speaker has around 23 minutes of genuine audio and 18 minutes of deepfake content. This dataset is designed to aid in deepfake detection and voice anti-spoofing research, allowing machine learning models to better adapt to real-world, naturally occurring audio samples. By providing a diverse set of speech data, it plays a vital role in evaluating the effectiveness of detection systems against AI-generated voices, contributing to the development of more secure and trustworthy audio verification technologies.
- The dataset comprises 70,000 real face images sourced from the Flickr dataset curated by Nvidia, alongside 70,000 artificially generated faces sampled from a collection of one million fake images created using StyleGAN by Bojan. To facilitate ease of use, both real and fake images have been resized to 256 pixels and organized into training, validation, and test sets. Additionally, CSV files are included for convenience. This dataset serves as a valuable resource for deepfake detection research, providing a balanced collection of authentic and synthetic facial images to train and evaluate machine learning models.
- This dataset consists of 400 videos, including 200 real and 200 deepfake videos, sourced from the FaceForensics++ and Celeb-DF datasets. Each video has been processed to extract face-cropped frames, with 40 frames captured per video. This dataset is designed for training and testing deepfake detection models, providing a balanced set of authentic and manipulated video samples to enhance model performance in identifying synthetic facial manipulations.

8.2.2 Preprocessing Details:

- For audio, the preprocessing stage involves converting audio files into Mel spectrograms, which visually represent sound frequencies over time. Each audio file is sampled at 22,050 Hz, and the spectrogram is computed and converted into a decibel scale to emphasize intensity variations.

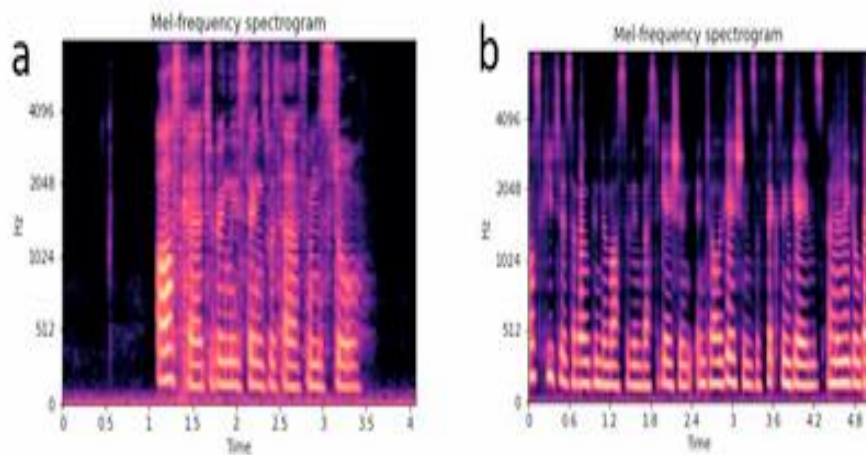


Figure 8.6 (a)original Mel-freq audio, (b) cloned Mel-freq audio

To ensure consistency, spectrogram images are resized to 128×128 pixels and reshaped into $(128, 128, 1)$ to match the expected input format of the model. The dataset consists of real and fake audio samples, with pixel values normalized between 0 and 1 to enhance numerical stability.

- For images, all images are resized to 224×224 pixels and their pixel values are scaled to a range of $[0,1]$ for better numerical stability. To enhance generalization and prevent overfitting, the training dataset is augmented using transformations like rotation, shifting, shearing, zooming, and horizontal flipping.

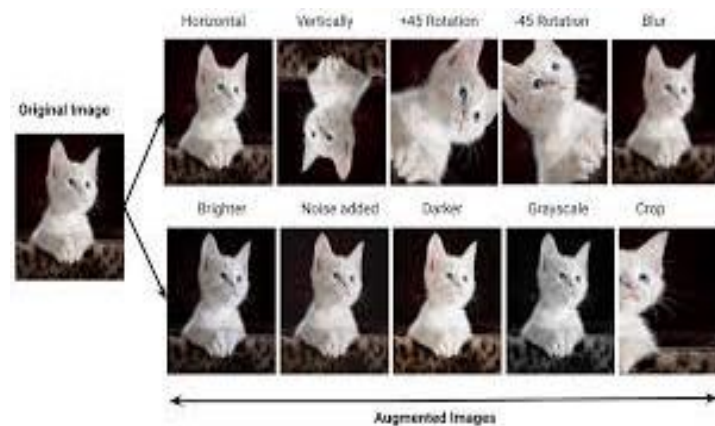


Figure 8.7 Data Augmentation Techniques

The validation and test datasets are only rescaled to preserve image integrity. The images are processed in batches of 64, and since the problem involves two classes.

- For videos, the preprocessing phase ensures that images are properly formatted before being fed into the model. Pixel values are normalized by rescaling them to the range $[0,1]$ to enhance numerical stability.

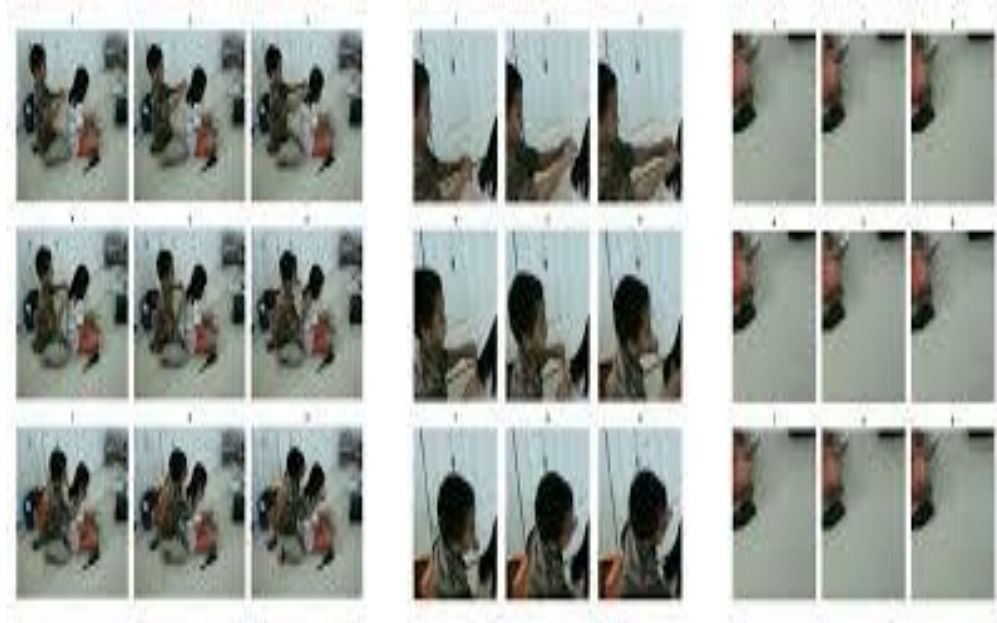


Figure 8.8 Extracted frames from video

To improve model generalization and reduce overfitting, various data augmentation techniques are applied to the training images, including rotation, shifting, shearing, zooming, and horizontal flipping. However, validation and test datasets undergo only rescaling to maintain consistency. All images are resized to 299×299 pixels, which matches the input size required for the Xception model. The batch size is set to 32, and since this is a binary classification task distinguishing real from fake images.

8.2.3 Model Details:

- For audio, the model architecture is based on a Convolutional Neural Network (CNN) with an attention mechanism for improved feature extraction. It comprises three convolutional layers (Conv2D) with ReLU activation, followed by max-pooling layers to reduce dimensionality. Attention modules are added after each convolutional block to emphasize important features while reducing noise. The architecture includes a fully

connected dense layer with 128 neurons and a dropout layer (0.5) to prevent overfitting. The final classification layer is activated using sigmoid, with outputs labeled as real (0) or fake (1). The model is compiled using the Adam optimizer with a learning rate of 0.0001, and binary cross-entropy is used as the loss function.

- For images, model utilizes EfficientNetB0, a pre-trained convolutional neural network designed for computational efficiency and high accuracy. The model is initialized with ImageNet weights, and all layers are set to trainable for fine-tuning. To extract meaningful features, a GlobalAveragePooling2D layer is incorporated, reducing feature map dimensions while retaining important information. A batch normalization layer is used to stabilize training, followed by Dense layers with 512 and 256 neurons, activated by ReLU for better learning. To prevent overfitting, dropout layers with rates of 0.4 and 0.3 are added. The final layer uses a sigmoid activation function to classify images as either real or fake.
- For videos, the deepfake detection model is built upon the Xception architecture, which is pre-trained on ImageNet. Instead of training the entire network from scratch, transfer learning is used to leverage Xception's ability to extract high-level features. To further tailor the model for this specific task, the last 20 layers of Xception are set to be trainable, while the earlier layers remain frozen to retain previously learned representations. Additional layers are added on top of the base model, including a GlobalAveragePooling2D layer, which reduces dimensionality while retaining key features. Two fully connected Dense layers with 1024 and 512 neurons, both using the ReLU activation function, help the model learn complex patterns. Dropout layers with 0.5 and 0.3 dropout rates are incorporated to prevent overfitting, and a sigmoid-activated output layer is used to produce binary predictions.

8.2.4 Model Training Details:

- For audio, the model is trained using techniques such as early stopping and learning rate reduction to optimize performance. EarlyStopping monitors validation loss and stops training if no improvement is observed for three consecutive epochs, while ReduceLROnPlateau lowers the learning rate by 50% if validation loss stagnates for two epochs. The training process runs for 10 epochs with a batch size of 32, and accuracy and loss curves are analyzed to evaluate training progress and stability.

- For images, the model is compiled with the Adam optimizer, using an adaptive learning rate decay to gradually reduce the learning rate for better convergence. The binary cross-entropy loss function is applied, as the classification involves two categories. The model undergoes 10 training epochs, leveraging real-time augmented data. Throughout training, performance is evaluated by monitoring accuracy and loss, ensuring stability and optimal learning. The validation dataset is used to assess model generalization.
- For videos, the model is compiled using the Adam optimizer with a learning rate of 0.0001 and is trained with binary cross-entropy loss, which is ideal for binary classification problems. Accuracy is used as the primary performance metric. The training process spans 30 epochs, but early stopping is implemented to halt training if the validation loss does not improve for five consecutive epochs, ensuring optimal performance. Additionally, a callback is applied to dynamically lower the learning rate if the validation loss stagnates, enhancing model convergence. The model is trained using augmented images from the training dataset, while validation data is used to assess generalization performance.

8.2.5 Model Prediction Details:

- For audio, the function first plays the input audio, extracts the Mel spectrogram, normalizes it, and feeds it into the trained model for classification. The model outputs a probability score and categorizes the input as real or fake with a confidence percentage. Finally, the trained model is saved as an HDF5 file (audio.h5), ensuring that it can be used for future predictions without requiring retraining.
- For images, the model is tested on unseen data to measure its accuracy. Predictions are made by setting a threshold of 0.5, classifying outputs as Real (0) or Fake (1). To evaluate performance, a classification report is generated, detailing metrics like precision, recall, and F1-score. A confusion matrix visually represents correct and incorrect classifications. Additionally, an ROC curve is plotted to assess the trade-off between sensitivity and specificity, with the AUC score indicating overall model performance.
- For videos, once training is complete, the model's performance is evaluated using the test dataset, and the final test accuracy is displayed. Predictions are generated by processing the test images, and outputs are classified based on a 0.5 threshold—values above 0.5 are labelled as Fake, while those below are classified as Real. A classification

report is generated to present precision, recall, and F1-score for both classes, offering insights into the model's effectiveness. This visualization helps identify any biases or misclassifications in the model's decision-making process.

8.3 Performance Metrics:

1. Accuracy:

Accuracy represents the proportion of correctly identified instances, whether real or deepfake, out of the total samples evaluated. In the context of deepfake detection, a higher accuracy value implies that the system is reliably distinguishing between authentic and manipulated media. However, in cases where the dataset is imbalanced, relying solely on accuracy might be misleading. The formula for calculating accuracy is:

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

where TP (True Positive) and TN (True Negative) denote correct predictions, and FP (False Positive) and FN (False Negative) indicate incorrect ones.

2. Precision:

Precision measures how many of the media samples identified as deepfake were truly deepfake. It is a critical metric when the cost of a false positive (wrongly flagging real media as fake) is high. In this project, high precision ensures that flagged deepfakes are manipulated, reducing the risk of mislabelling real content. The formula to compute precision is:

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

where TP is the count of true positives and FP is the count of false positives.

3. Recall (Sensitivity):

Recall indicates the model's ability to detect all actual deepfake instances in the dataset. It answers the question: out of all true deepfakes, how many were correctly detected? In deepfake detection tasks, maintaining a high recall is crucial because missing even a few manipulated samples can lead to serious consequences. The formula for recall is:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

where FN (False Negative) represents the deepfakes that the model failed to detect.

4. F1 Score:

The F1 Score combines both recall and precision into a single metric by calculating their harmonic mean. It is particularly useful when there is a trade-off between precision and recall. In deepfake detection, a high F1 Score implies that the model effectively identifies deepfakes while minimizing incorrect classifications. The F1 Score can be calculated using:

$$\text{F1-Score} = \frac{2 * \text{Recall} * \text{Precision}}{\text{Recall} + \text{Precision}}$$

5. Confusion Matrix:

The confusion matrix is a tabular representation that summarizes the model's performance by displaying the counts of true positives, true negatives, false positives, and false negatives. It provides detailed insights into where the model is making correct or incorrect predictions. For deepfake detection, analyzing the confusion matrix helps to understand how often real or fake media are misclassified, allowing better tuning of the system.

		True Class	
		Positive	Negative
Predicated Class	Positive	TP	FP
	Negative	FN	TN

Figure 8.9 Confusion Matrix

6. ROC Curve:

The Receiver Operating Characteristic (ROC) Curve is a graphical plot that illustrates the model's diagnostic ability by plotting the true positive rate (recall) against the false positive rate at various threshold levels. In this project, the ROC curve helps visualize how well the model differentiates between real and fake samples under different classification thresholds. A curve which is closer to the top-left corner indicates a better-performing model.



Figure 8.10 ROC Curve

7. AUC Score:

The Area Under the ROC Curve (AUC) summarizes the ROC curve's performance into a single value. A higher AUC score means the model has a better capability to distinguish between deepfakes and real media. In deepfake detection, an AUC score close to 1 indicates that the model has an excellent distinction capability, while a score closer to 0.5 implies random guessing.

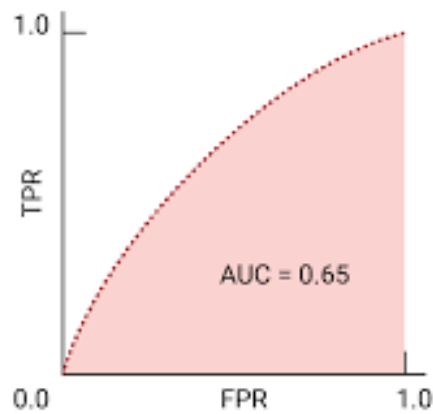


Figure 8.11 AUC Curve

CHAPTER 9

RESULT AND DISCUSSIONS

9.1 Screenshots:

9.1.1 Audio Deepfake Detection:

```
import numpy as np
import librosa
import cv2
import os
import matplotlib.pyplot as plt
import seaborn as sns
from tqdm import tqdm
from tensorflow.keras.models import Model
from tensorflow.keras.layers import (Input, Conv2D, MaxPooling2D, Flatten, Dense, Dropout,
                                     GlobalAveragePooling2D, Reshape, Multiply)
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
from sklearn.metrics import classification_report, confusion_matrix

# Function to extract Mel Spectrogram
def extract_mel_spectrogram(file_path, img_size=(128, 128)):
    y, sr = librosa.load(file_path, sr=22050)
    mel_spec = librosa.feature.melspectrogram(y=y, sr=sr)
    mel_spec_db = librosa.power_to_db(mel_spec, ref=np.max)
    mel_spec_resized = cv2.resize(mel_spec_db, img_size)
    return mel_spec_resized
```

```

# Function to load data
def load_data(directory, img_size=(128, 128)):
    data = []
    labels = []
    classes = {"real": 0, "fake": 1}

    for label in os.listdir(directory):
        class_dir = os.path.join(directory, label)
        if os.path.isdir(class_dir):
            for file in tqdm(os.listdir(class_dir), desc=f"Processing {label}"):
                file_path = os.path.join(class_dir, file)
                if file_path.endswith(".wav"):
                    mel_spec = extract_mel_spectrogram(file_path, img_size)
                    data.append(mel_spec)
                    labels.append(classes[label])

    return np.array(data), np.array(labels)

```

```

# Load data
img_size = (128, 128)
X_train, y_train = load_data(train_dir, img_size)
X_val, y_val = load_data(val_dir, img_size)
X_test, y_test = load_data(test_dir, img_size)

# Reshape for CNN
X_train = X_train.reshape(-1, img_size[0], img_size[1], 1)
X_val = X_val.reshape(-1, img_size[0], img_size[1], 1)
X_test = X_test.reshape(-1, img_size[0], img_size[1], 1)

# Normalize
def normalize(data):
    return (data - np.min(data)) / (np.max(data) - np.min(data) + 1e-8)

X_train, X_val, X_test = map(normalize, [X_train, X_val, X_test])

```

```
# Attention Mechanism
```

```
def attention_block(inputs):  
    x = GlobalAveragePooling2D()(inputs)  
    x = Dense(inputs.shape[-1], activation='relu')(x)  
    x = Dense(inputs.shape[-1], activation='sigmoid')(x)  
    x = Reshape((1, 1, inputs.shape[-1]))(x)  
    x = Multiply()([inputs, x])  
    return x
```

```
# Model with attention
```

```
def create_model():  
    inputs = Input(shape=(128, 128, 1))  
    x = Conv2D(32, (3, 3), activation='relu', padding='same')(inputs)  
    x = MaxPooling2D((2, 2))(x)  
    x = attention_block(x)  
  
    x = Conv2D(64, (3, 3), activation='relu', padding='same')(x)  
    x = MaxPooling2D((2, 2))(x)  
    x = attention_block(x)  
  
    x = Conv2D(128, (3, 3), activation='relu', padding='same')(x)  
    x = MaxPooling2D((2, 2))(x)  
    x = attention_block(x)  
  
    x = Flatten()(x)  
    x = Dense(128, activation='relu')(x)  
    x = Dropout(0.5)(x)  
    outputs = Dense(1, activation='sigmoid')(x)  
  
    model = Model(inputs, outputs)  
    model.compile(optimizer=Adam(learning_rate=0.0001), loss='binary_crossentropy', metrics=['accuracy'])  
    return model  
  
model = create_model()
```

```
# Train
callbacks = [EarlyStopping(monitor='val_loss', patience=3, restore_best_weights=True),
             ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=2)]

history = model.fit(
    X_train, y_train,
    validation_data=(X_val, y_val),
    epochs=10,
    batch_size=32,
    callbacks=callbacks
)
```

```
Epoch 1/10
696/696 ————— 26s 25ms/step - accuracy: 0.7793 - loss: 0.4544 - val_accuracy: 0.9485 - val_loss: 0.1350 - learning_rate: 1.0000e-04
Epoch 2/10
696/696 ————— 12s 17ms/step - accuracy: 0.9565 - loss: 0.1282 - val_accuracy: 0.9732 - val_loss: 0.0739 - learning_rate: 1.0000e-04
Epoch 3/10
696/696 ————— 12s 17ms/step - accuracy: 0.9741 - loss: 0.0814 - val_accuracy: 0.9860 - val_loss: 0.0485 - learning_rate: 1.0000e-04
Epoch 4/10
696/696 ————— 12s 17ms/step - accuracy: 0.9809 - loss: 0.0641 - val_accuracy: 0.9841 - val_loss: 0.0468 - learning_rate: 1.0000e-04
Epoch 5/10
696/696 ————— 12s 17ms/step - accuracy: 0.9838 - loss: 0.0541 - val_accuracy: 0.9866 - val_loss: 0.0383 - learning_rate: 1.0000e-04
Epoch 6/10
696/696 ————— 12s 17ms/step - accuracy: 0.9852 - loss: 0.0492 - val_accuracy: 0.9891 - val_loss: 0.0368 - learning_rate: 1.0000e-04
Epoch 7/10
696/696 ————— 12s 17ms/step - accuracy: 0.9888 - loss: 0.0396 - val_accuracy: 0.9907 - val_loss: 0.0313 - learning_rate: 1.0000e-04
Epoch 8/10
696/696 ————— 12s 17ms/step - accuracy: 0.9896 - loss: 0.0347 - val_accuracy: 0.9910 - val_loss: 0.0302 - learning_rate: 1.0000e-04
Epoch 9/10
696/696 ————— 12s 18ms/step - accuracy: 0.9906 - loss: 0.0315 - val_accuracy: 0.9926 - val_loss: 0.0276 - learning_rate: 1.0000e-04
Epoch 10/10
696/696 ————— 12s 18ms/step - accuracy: 0.9907 - loss: 0.0302 - val_accuracy: 0.9926 - val_loss: 0.0270 - learning_rate: 1.0000e-04
100/100 ————— 1s 14ms/step - accuracy: 0.9820 - loss: 0.0618
```

```

# Classification report
y_pred = (model.predict(X_test) > 0.5).astype("int32")
print("\nClassification Report:\n", classification_report(y_test, y_pred, target_names=["Real", "Fake"]))

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=["Real", "Fake"], yticklabels=["Real", "Fake"])
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()

```

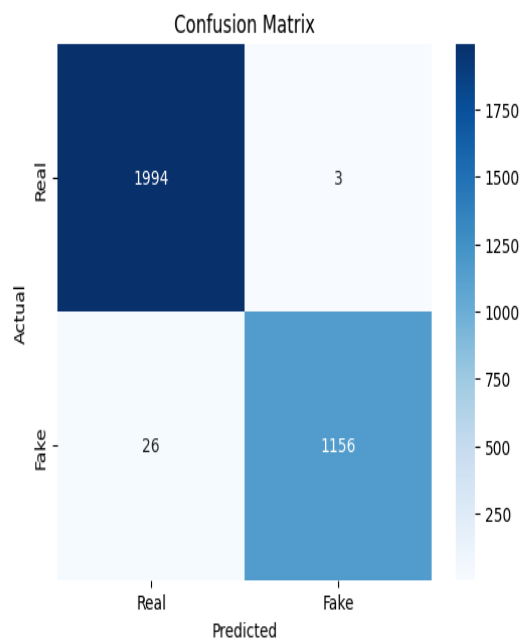
```

Classification Report:
              precision    recall  f1-score   support

     Real       0.99       1.00       0.99       1997
     Fake       1.00       0.98       0.99       1182

 accuracy              0.99       3179
 macro avg       0.99       0.99       0.99       3179
 weighted avg    0.99       0.99       0.99       3179

```



```
# Evaluate
loss, acc = model.evaluate(X_test, y_test)
print(f"Test Accuracy: {acc * 100:.2f}%")
```

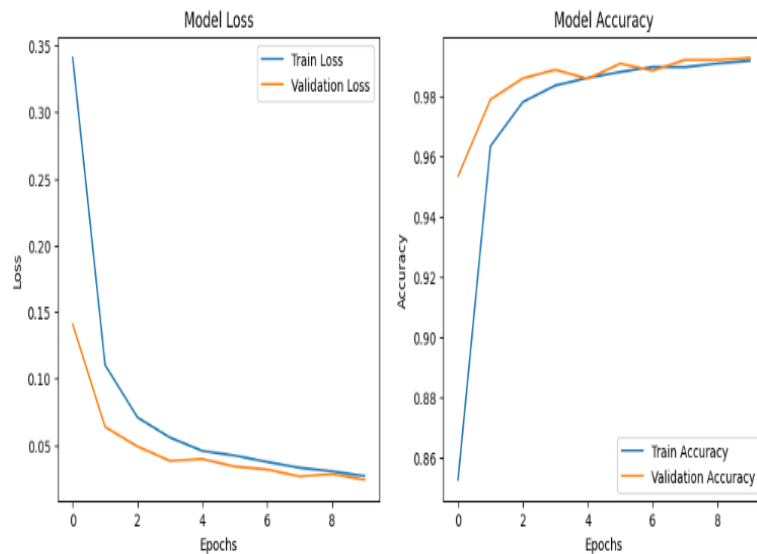
100/100 ————— 1s 5ms/step - accuracy: 0.9855 - loss: 0.0409
Test Accuracy: 99.21%

```
# Plot Model Loss and Accuracy
plt.figure(figsize=(12, 5))

# Loss
plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Model Loss')
plt.legend()

# Accuracy
plt.subplot(1, 2, 2)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.title('Model Accuracy')
plt.legend()

plt.show()
```



Example usage

```
test_audio_path = "/kaggle/input/release-in-the-wild/release_in_the_wild/test/real/10098.wav" # Change path  
predict_audio_with_playback(model, test_audio_path)
```

Playing: /kaggle/input/release-in-the-wild/release_in_the_wild/test/real/10098.wav

▶ 0:00 / 0:00 ————— 🔊 ⋮

1/1 ————— 0s 19ms/step

Prediction: ☒ REAL (0.00)

9.1.2 Image Deepfake Detection:

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import EfficientNetB0
from tensorflow.keras.layers import Dense, Dropout, GlobalAveragePooling2D, BatchNormalization
from tensorflow.keras.models import Model
from sklearn.metrics import classification_report, confusion_matrix, roc_curve, auc
import numpy as np
import os
import matplotlib.pyplot as plt
import seaborn as sns
from PIL import Image
import cv2

# Data preprocessing with optimized augmentation
data_gen = ImageDataGenerator(
    rescale=1.0/255,
    rotation_range=20,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.1,
    zoom_range=0.1,
    horizontal_flip=True,
    fill_mode='nearest'
)

train_data = data_gen.flow_from_directory(train_dir, target_size=(224, 224), batch_size=64, class_mode='binary')
val_data = data_gen.flow_from_directory(val_dir, target_size=(224, 224), batch_size=64, class_mode='binary')
test_data = ImageDataGenerator(rescale=1.0/255).flow_from_directory(test_dir, target_size=(224, 224), batch_size=64, class_mode='binary', shuffle=False)
```

Found 100000 images belonging to 2 classes.

Found 20000 images belonging to 2 classes.

Found 20000 images belonging to 2 classes.

```
# Build model inside strategy scope

base_model = EfficientNetB0(weights='imagenet', include_top=False, input_shape=(224, 224, 3))

for layer in base_model.layers:
    layer.trainable = True # Fine-tune all layers

    x = GlobalAveragePooling2D()(base_model.output)
    x = BatchNormalization()(x)
    x = Dense(512, activation='relu')(x)
    x = Dropout(0.4)(x)
    x = Dense(256, activation='relu')(x)
    x = Dropout(0.3)(x)
    output = Dense(1, activation='sigmoid', dtype='float32')(x)
    model = Model(inputs=base_model.input, outputs=output)

# Learning rate scheduling
lr_schedule = tf.keras.optimizers.schedules.ExponentialDecay(initial_learning_rate=0.001, decay_steps=1000, decay_rate=0.9)
model.compile(optimizer=keras.optimizers.Adam(learning_rate=lr_schedule), loss='binary_crossentropy', metrics=['accuracy'])

# Train model
epochs = 10
history = model.fit(train_data, validation_data=val_data, epochs=epochs)
```

Downloading data from https://storage.googleapis.com/keras-applications/efficientnetb0_notop.h5

16705208/16705208 ————— 0s 0us/step

Epoch 1/10

/usr/local/lib/python3.10/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:122: UserWarning: Your `PyDataset` class should call `super` in its constructor. `**kwargs` can include `workers`, `use_multiprocessing`, `max_queue_size`. Do not pass these arguments to `fit()`, as they will be ignored.
self.warn_if_super_not_called()

1563/1563 ————— 2386s 1s/step - accuracy: 0.8912 - loss: 0.2575 - val_accuracy: 0.9700 - val_loss: 0.0736

Epoch 2/10

1563/1563 ————— 1586s 1s/step - accuracy: 0.9793 - loss: 0.0566 - val_accuracy: 0.8618 - val_loss: 0.3946

Epoch 3/10

1563/1563 ————— 1591s 1s/step - accuracy: 0.9866 - loss: 0.0377 - val_accuracy: 0.9747 - val_loss: 0.0730

Epoch 4/10

1563/1563 ————— 1584s 1s/step - accuracy: 0.9893 - loss: 0.0307 - val_accuracy: 0.9434 - val_loss: 0.1596

Epoch 5/10

1563/1563 ————— 1569s 1000ms/step - accuracy: 0.9923 - loss: 0.0216 - val_accuracy: 0.9938 - val_loss: 0.0179

Epoch 6/10

1563/1563 ————— 1572s 1s/step - accuracy: 0.9939 - loss: 0.0166 - val_accuracy: 0.9197 - val_loss: 0.2938

Epoch 7/10

1563/1563 ————— 1598s 1s/step - accuracy: 0.9963 - loss: 0.0106 - val_accuracy: 0.9279 - val_loss: 0.1989

Epoch 8/10

1563/1563 ————— 1590s 1s/step - accuracy: 0.9973 - loss: 0.0087 - val_accuracy: 0.9693 - val_loss: 0.0981

Epoch 9/10

1563/1563 ————— 1582s 1s/step - accuracy: 0.9974 - loss: 0.0078 - val_accuracy: 0.9871 - val_loss: 0.0409

Epoch 10/10

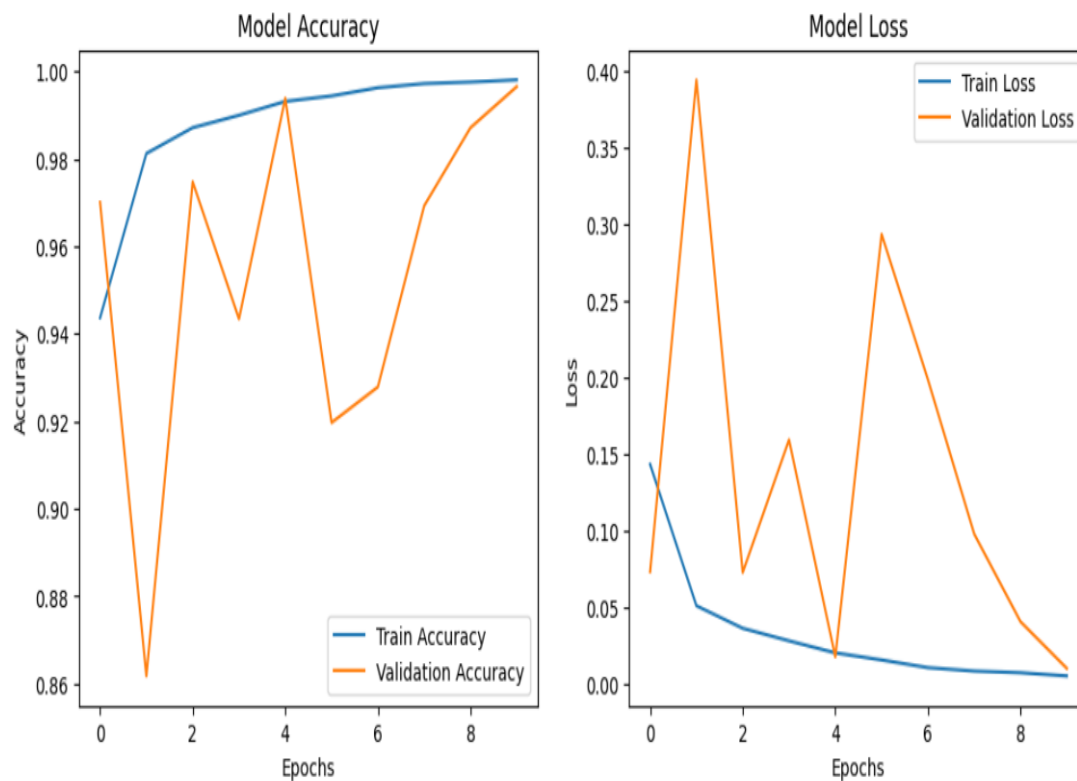
1563/1563 ————— 1608s 1s/step - accuracy: 0.9978 - loss: 0.0058 - val_accuracy: 0.9965 - val_loss: 0.0104

```
# Evaluate model
loss, accuracy = model.evaluate(test_data)
print(f'Test Accuracy: {accuracy:.2f}')
```

313/313 ————— 187s 598ms/step - accuracy: 0.9561 - loss: 0.1479
Test Accuracy: 0.97

```
# Plot accuracy and loss
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.title('Model Accuracy')

plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.title('Model Loss')
plt.show()
```



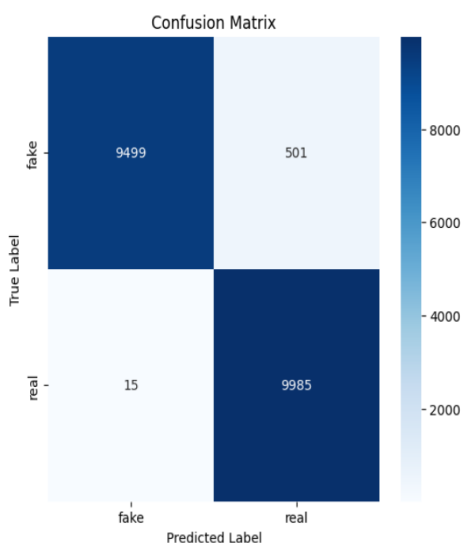
```
# Generate classification report
y_true = test_data.classes
y_pred_probs = model.predict(test_data)
y_pred = (y_pred_probs > 0.5).astype("int32").flatten()
print(classification_report(y_true, y_pred, target_names=test_data.class_indices.keys()))
```

```
313/313 ————— 68s 197ms/step
          precision    recall  f1-score   support

 fake         1.00        0.95        0.97     10000
  real         0.95        1.00        0.97     10000

 accuracy                   0.97     20000
 macro avg         0.98        0.97        0.97     20000
weighted avg         0.98        0.97        0.97     20000
```

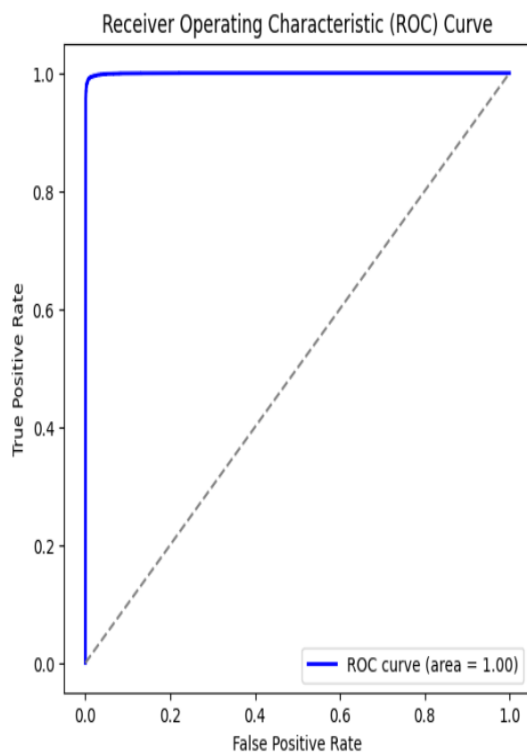
```
# Confusion Matrix
cm = confusion_matrix(y_true, y_pred)
plt.figure(figsize=(6, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=test_data.class_indices.keys(), yticklabels=test_data.class_indices.keys())
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('Confusion Matrix')
plt.show()
```



```

# ROC Curve
fpr, tpr, _ = roc_curve(y_true, y_pred_probs)
roc_auc = auc(fpr, tpr)
plt.figure(figsize=(6, 6))
plt.plot(fpr, tpr, color='blue', lw=2, label=f'ROC curve (area = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='grey', linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend()
plt.show()

```



```

# Display predicted fake images
fake_indices = np.where(y_pred == 1)[0]

plt.figure(figsize=(10, 10))
for i, idx in enumerate(fake_indices[:9]): # Show first 9 fake images
    img_path = os.path.join(test_dir, test_data.filename[idx])
    img = cv2.imread(img_path)
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    plt.subplot(3, 3, i+1)
    plt.imshow(img)
    plt.title(f'Fake (Confidence: {y_pred_probs[idx][0] * 100:.2f}%)')
    plt.axis('off')
plt.tight_layout()
plt.show()

```

Fake (Confidence: 96.73%)



Fake (Confidence: 86.08%)



Fake (Confidence: 75.13%)



9.1.3 Video Deepfake Detection:

```
import os
import numpy as np
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import Xception
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D, Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import ReduceLROnPlateau, EarlyStopping
```

```
# Image Preprocessing
img_size = (299, 299)
batch_size = 32

train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=40,
    width_shift_range=0.3,
    height_shift_range=0.3,
    shear_range=0.3,
    zoom_range=0.3,
    horizontal_flip=True,
    fill_mode='nearest')

val_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)
```



```
train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=img_size,
    batch_size=batch_size,
    class_mode='binary')
```

```
val_generator = val_datagen.flow_from_directory(
    val_dir,
    target_size=img_size,
    batch_size=batch_size,
    class_mode='binary')
```

```
test_generator = test_datagen.flow_from_directory(
    test_dir,
    target_size=img_size,
    batch_size=batch_size,
    class_mode='binary',
    shuffle=False)
```

Load Xception Model

```
base_model = Xception(weights='imagenet', include_top=False, input_shape=(299, 299, 3))
```

```
for layer in base_model.layers[:-20]: # Unfreeze last 20 layers for fine-tuning
    layer.trainable = True
```

Add Custom Layers

```
x = base_model.output
```

```
x = GlobalAveragePooling2D()(x)
```

```
x = Dense(1024, activation='relu')(x)
```

```
x = Dropout(0.5)(x)
```

```
x = Dense(512, activation='relu')(x)
```

```
x = Dropout(0.3)(x)
```

```
output_layer = Dense(1, activation='sigmoid')(x)
```

```
model = Model(inputs=base_model.input, outputs=output_layer)
```

```

# Compile Model
model.compile(optimizer=Adam(learning_rate=0.0001), loss='binary_crossentropy', metrics=['accuracy'])

# Callbacks
lr_scheduler = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=3, min_lr=1e-6)
early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)

# Train Model
epochs = 30
history = model.fit(
    train_generator,
    validation_data=val_generator,
    epochs=epochs,
    callbacks=[lr_scheduler, early_stopping],
    verbose=1)

```

```

Epoch 1/30
364/364 ————— 351s 859ms/step - accuracy: 0.7060 - loss: 0.5156 - val_accuracy: 0.9004 - val_loss: 0.2813 - learning_rate: 1.0000e-04
Epoch 2/30
364/364 ————— 303s 819ms/step - accuracy: 0.9303 - loss: 0.1739 - val_accuracy: 0.8338 - val_loss: 0.5794 - learning_rate: 1.0000e-04
Epoch 3/30
364/364 ————— 304s 822ms/step - accuracy: 0.9522 - loss: 0.1177 - val_accuracy: 0.9192 - val_loss: 0.2694 - learning_rate: 1.0000e-04
Epoch 4/30
364/364 ————— 303s 819ms/step - accuracy: 0.9667 - loss: 0.0793 - val_accuracy: 0.9296 - val_loss: 0.2322 - learning_rate: 1.0000e-04
Epoch 5/30
364/364 ————— 304s 822ms/step - accuracy: 0.9684 - loss: 0.0790 - val_accuracy: 0.9225 - val_loss: 0.2158 - learning_rate: 1.0000e-04
Epoch 6/30
364/364 ————— 303s 819ms/step - accuracy: 0.9724 - loss: 0.0628 - val_accuracy: 0.9237 - val_loss: 0.2845 - learning_rate: 1.0000e-04
Epoch 7/30
364/364 ————— 303s 819ms/step - accuracy: 0.9769 - loss: 0.0594 - val_accuracy: 0.9112 - val_loss: 0.3619 - learning_rate: 1.0000e-04
Epoch 8/30
364/364 ————— 303s 819ms/step - accuracy: 0.9759 - loss: 0.0587 - val_accuracy: 0.9075 - val_loss: 0.4634 - learning_rate: 1.0000e-04
Epoch 9/30
364/364 ————— 303s 818ms/step - accuracy: 0.9818 - loss: 0.0369 - val_accuracy: 0.9404 - val_loss: 0.2328 - learning_rate: 2.0000e-05
Epoch 10/30
364/364 ————— 304s 822ms/step - accuracy: 0.9853 - loss: 0.0286 - val_accuracy: 0.9383 - val_loss: 0.2360 - learning_rate: 2.0000e-05
75/75 ————— 17s 218ms/step - accuracy: 0.9166 - loss: 0.3683

```

```
# Evaluate Model
test_loss, test_acc = model.evaluate(test_generator, verbose=1)
print(f"Test Accuracy: {test_acc * 100:.2f}%")
```

Test Accuracy: 92.87%

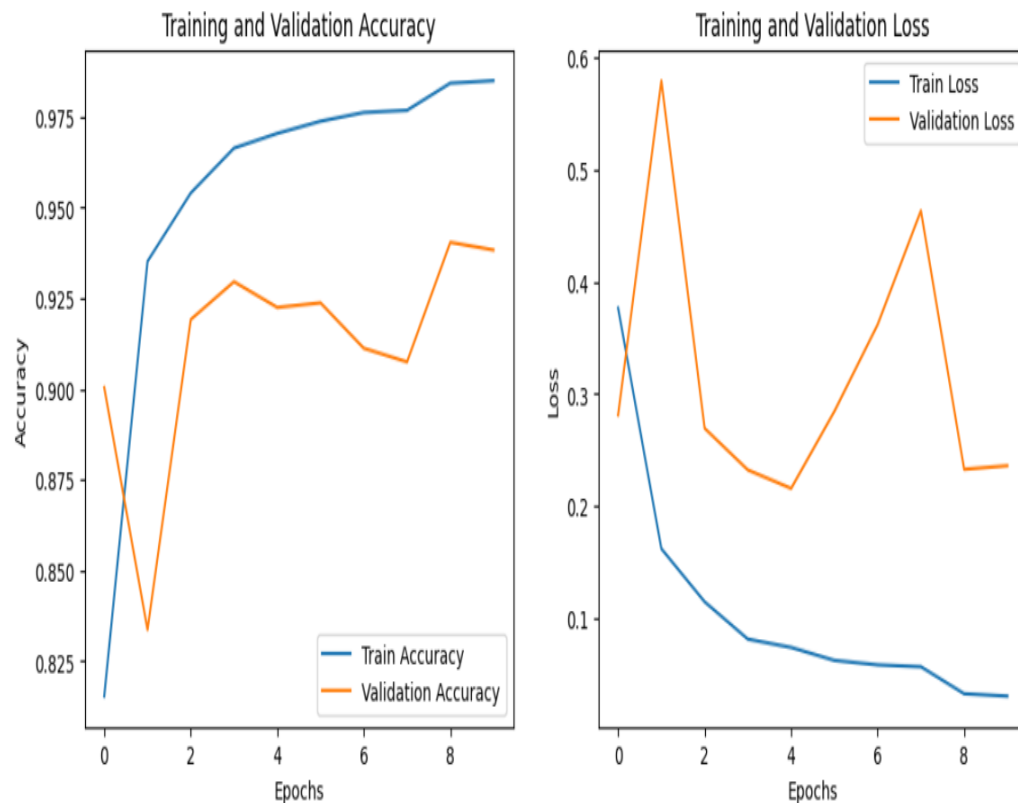
```
# Plot Training Accuracy and Loss
def plot_training_history(history):
    plt.figure(figsize=(12, 5))

    # Accuracy Plot
    plt.subplot(1, 2, 1)
    plt.plot(history.history['accuracy'], label='Train Accuracy')
    plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
    plt.xlabel('Epochs')
    plt.ylabel('Accuracy')
    plt.legend()
    plt.title('Training and Validation Accuracy')

    # Loss Plot
    plt.subplot(1, 2, 2)
    plt.plot(history.history['loss'], label='Train Loss')
    plt.plot(history.history['val_loss'], label='Validation Loss')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.legend()
    plt.title('Training and Validation Loss')

    plt.show()

plot_training_history(history)
```



```
# Classification Report & Confusion Matrix
y_true = test_generator.classes
y_pred = (model.predict(test_generator) > 0.5).astype(int).flatten()

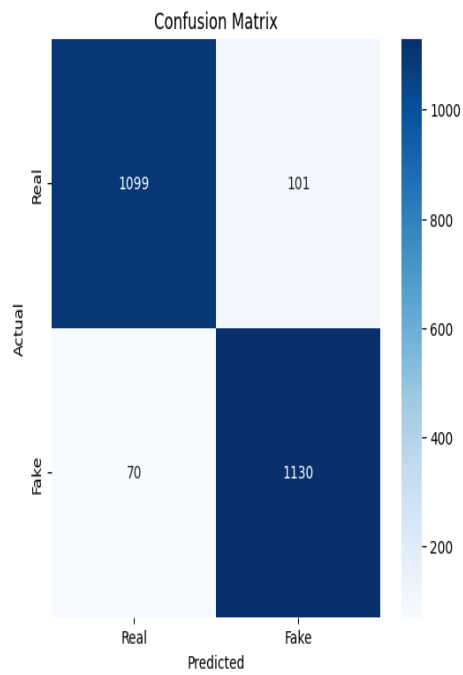
print("Classification Report:")
print(classification_report(y_true, y_pred, target_names=['Real', 'Fake']))

# Plot Confusion Matrix
conf_matrix = confusion_matrix(y_true, y_pred)
plt.figure(figsize=(6, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=['Real', 'Fake'], yticklabels=['Real', 'Fake'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

```
Classification Report:
              precision    recall  f1-score   support

     Real       0.94       0.92       0.93       1200
     Fake       0.92       0.94       0.93       1200

 accuracy              0.93       2400
  macro avg       0.93       0.93       0.93       2400
 weighted avg       0.93       0.93       0.93       2400
```



```
# Example Usage
# /kaggle/input/deep-fake-detection-dfd-entire-original-dataset
video_path = "/kaggle/input/deep-fake-detection-dfd-entire-original-dataset/DFD_manipulated_sequences/DFD_manipulated_sequences/01_04__hugging_happ
predict_video_frames(video_path, model)
```

Model loaded
1/1 ————— 1s 1s/step



1/1 ————— 0s 260ms/step



CHAPTER 10

CONCLUSION AND FUTURE WORK

10.1 Conclusion:

To conclude, this project introduces a powerful and deepfake detection system that utilizes deep learning techniques to detect manipulated media in images, videos, and audio formats. By incorporating pre-trained models like EfficientNet for image detection, a fine-tuned Xception model for analyzing video frames, and CNNs with attention mechanisms for audio analysis, the system provides an effective and scalable solution for deepfake identification. The implementation of cutting-edge methods ensures both high accuracy and computational efficiency, offering a valuable resource to tackle the rising challenge of misinformation and uphold the authenticity of digital media. This approach not only advances the field of deepfake detection but also plays a crucial role in preserving the credibility of digital content in a time of widespread media manipulation.

10.2 Future Work:

Technological solutions, especially those based on evolving fields like artificial intelligence, always have room for further growth and refinement. Deepfake Guardian has laid a strong foundation, but it holds potential for even broader impact through future developments. One promising direction is the creation of a browser extension. By integrating deepfake detection directly into users' web browsing experience, this extension would allow for instant identification of manipulated content without requiring users to switch platforms. Such real-time detection would greatly improve convenience and accessibility.

Currently, Deepfake Guardian primarily focuses on detecting facial deepfakes. A logical next step would be to extend the system's capabilities to detect full body deepfakes. This enhancement would expand its relevance across fields like digital forensics, content regulation, and online media authentication, where sophisticated deepfakes are increasingly posing challenges.

As deepfake generation techniques continue to advance, future iterations of this system must adopt more powerful AI models and faster, real-time detection pipelines to stay ahead. This evolution will not only enhance detection accuracy but also ensure timely and reliable responses against new and complex threats. Furthermore, to support wider public use, the development of a comprehensive, user-friendly, and visually appealing interface should be prioritized. Offering this improved platform free of charge would promote greater community engagement and make effective deepfake detection tools accessible to a larger audience, ultimately contributing to a safer digital environment.

CHAPTER 11

REFERENCES

- [1] Kaur, A., Noori Hoshyar, A., Saikrishna, V., Firmin, S., & Xia, F. (2024). Deepfake video detection: challenges and opportunities. *Artificial Intelligence Review*, 57(6), 1-47.
- [2] Malik, A., Kuribayashi, M., Abdullahi, S. M., & Khan, A. N. (2022). DeepFake detection for human face images and videos: A survey. *Ieee Access*, 10, 18757-18775.
- [3] Rana, M. S., Nobil, M. N., Murali, B., & Sung, A. H. (2022). Deepfake detection: A systematic literature review. *IEEE access*, 10, 25494-25513.
- [4] Shaaban, O. A., Yildirim, R., & Alguttar, A. A. (2023). Audio Deepfake Approaches. *IEEE Access*, 11, 132652-132682.
- [5] Raza, A., Munir, K., & Almutairi, M. (2022). A novel deep learning approach for deepfake image detection. *Applied Sciences*, 12(19), 9820.
- [6] Mubarak, R., Alsabou, T., Alshaikh, O., Inuwa-Dutse, I., Khan, S., & Parkinson, S. (2023). A survey on the detection and impacts of deepfakes in visual, audio, and textual formats. *Ieee Access*, 11, 144497-144529.
- [7] Akhtar, Z., Pendyala, T. L., & Athmakuri, V. S. (2024). Video and audio deepfake datasets and open issues in deepfake technology: being ahead of the curve. *Forensic Sciences*, 4(3), 289-377.
- [8] Almars, A. M. (2021). Deepfakes detection techniques using deep learning: a survey. *Journal of Computer and Communications*, 9(05), 20-35.
- [9] Heidari, A., Jafari Navimipour, N., Dag, H., & Unal, M. (2024). Deepfake detection using deep learning methods: A systematic and comprehensive review. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 14(2), e1520.
- [10] Yu, P., Xia, Z., Fei, J., & Lu, Y. (2021). A survey on deepfake video detection. *Iet Biometrics*, 10(6), 607-624.
- [11] Dolhansky, B., Bitton, J., Pflaum, B., Lu, J., Howes, R., Wang, M., & Ferrer, C. C. (2020). The deepfake detection challenge (dfdc) dataset. *arXiv preprint arXiv:2006.07397*.

- [12] Almutairi, Z., & Elgibreen, H. (2022). A review of modern audio deepfake detection methods: challenges and future directions. *Algorithms*, 15(5), 155.
- [13] Khochare, J., Joshi, C., Yenarkar, B., Suratkar, S., & Kazi, F. (2021). A deep learning framework for audio deepfake detection. *Arabian Journal for Science and Engineering*, 1-12.
- [14] Mcuba, M., Singh, A., Ikuesan, R. A., & Venter, H. (2023). The effect of deep learning methods on deepfake audio detection for digital investigation. *Procedia Computer Science*, 219, 211-219.
- [15] Badale, A., Castelino, L., Darekar, C., & Gomes, J. (2018). Deep fake detection using neural networks. In *15th IEEE international conference on advanced video and signal based surveillance (AVSS)* (Vol. 2).