VIVEKANAND EDUCATION SOCIETY'S INSTITUTE OF TECHNOLOGY
An Autonomous Institute Affiliated to University of Mumbai
Department of Computer Engineering

Project Report on

Detection of Face Swap in DeepFakes

In partial fulfillment of the Fourth Year, Bachelor of Engineering (B.E.) Degree in Computer Engineering at the University of Mumbai Academic Year 2024-25

Submitted by
Yashneil Ballani (D17 - B , Roll no -02)
Netal Bhansali (D17 - B , Roll no -06)
Rashika Chandwani (D17 - B , Roll no -07)
Bhanu Shamdasani (D17 - B , Roll no -49)

Project Mentor
Dr. Mrs. Priya R L

(2024-25)

VIVEKANAND EDUCATION SOCIETY'S INSTITUTE OF TECHNOLOGY
An Autonomous Institute Affiliated to University of Mumbai
Department of Computer Engineering

Certificate

This is to certify that ***Yashneil Ballani, Netal Bhansali, Rashika Chandwai, Bhanu Shamdasani*** of Fourth Year Computer Engineering studying under the University of Mumbai have satisfactorily completed the project on "***Detection of Face Swap in Deepfakes***" as a part of their coursework of PROJECT-II for Semester-VIII under the guidance of their mentor ***Dr. Priya R L*** in the year 2024-25 .

This thesis/dissertation/project report entitled **Detection of Face Swap in Deepfakes** by ***Yashneil Ballani, Netal Bhansali, Rashika Chandwai, Bhanu Shamdasani*** is approved for the degree of **B. E (Computer Engineering).**

Programme Outcomes	Grade
PO1,PO2,PO3,PO4,PO5,PO6,PO7, PO8, PO9, PO10, PO11, PO12 PSO1, PSO2	

Date:

Project Guide:

Project Report Approval
For
B. E (Computer Engineering)

This thesis/dissertation/project report entitled **Detection of Face Swap in Deepfakes** by ***Yashneil Ballani, Netal Bhansali, Rashika Chandwai, Bhanu Shamdasani*** is approved for the degree of **B. E (Computer Engineering).**

Internal Examiner

External Examiner

Head of the Department

Principal

Date:
Place:

Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea / data / fact / source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

--	--
(Signature)	(Signature)
Yashneil Ballani, 02	Netal Bhansali, 06
--	--
(Signature)	(Signature)
Rashika Chandwani, 07	Bhanu Shamdasani, 49

Date: 28th April, 2025

Acknowledgement

We are thankful to our college Vivekanand Education Society's Institute of Technology for considering our project and extending help at all stages needed during our work of collecting information regarding the project.

It gives us immense pleasure to express our deep and sincere gratitude to Assistant Professor **Priya R L** for her kind help and valuable advice during the development of project synopsis and for her guidance and suggestions.

We are deeply indebted to Head of the Computer Department **Dr.(Mrs.) Nupur Giri** and our Principal **Dr. (Mrs.) J.M. Nair ,** for giving us this valuable opportunity to do this project.

We express our hearty thanks to them for their assistance without which it would have been difficult in finishing this project synopsis and project review successfully.

We convey our deep sense of gratitude to all teaching and non-teaching staff for their constant encouragement, support and selfless help throughout the project work. It is a great pleasure to acknowledge the help and suggestion, which we received from the Department of Computer Engineering.

We wish to express our profound thanks to all those who helped us in gathering information about the project. Our families too have provided moral support and encouragement several times.

Computer Engineering Department

COURSE OUTCOMES FOR B.E. PROJECT

Learners will be able to,

Course Outcome	Description of the Course Outcome
CO1	Able to apply the relevant engineering concepts, knowledge and skills towards the project.
CO2	Able to identify, formulate and interpret the various relevant research papers and to determine the problem.
CO3	Able to apply the engineering concepts towards designing solutions for the problem.
CO4	Able to interpret the data and datasets to be utilized.
CO5	Able to create, select and apply appropriate technologies, techniques, resources and tools for the project.
CO6	Able to apply ethical, professional policies and principles towards societal, environmental, safety and cultural benefit.
CO7	Able to function effectively as an individual, and as a member of a team, allocating roles with clear lines of responsibility and accountability.
CO8	Able to write effective reports, design documents and make effective presentations.
CO9	Able to apply engineering and management principles to the project as a team member.
CO10	Able to apply the project domain knowledge to sharpen one's competency.
CO11	Able to develop a professional, presentational, balanced and structured approach towards project development.
CO12	Able to adopt skills, languages, environment and platforms for creating innovative solutions for the project.

INDEX

A. List of Figures

B. List of Tables

Abstract

With the growing sophistication of face-swapping deepfake technologies, concerns around digital media authenticity have become increasingly significant. Although several detection systems exist, many struggle with limitations such as shallow network architectures, inadequate feature extraction, and poor resilience to high-quality manipulations. These shortcomings reduce their reliability when faced with more advanced synthetic content.

To overcome these challenges, this project introduces a deep learning-based approach using convolutional neural networks and transfer learning to differentiate between real and fake facial images. A curated dataset of authentic and face-swapped images was thoroughly preprocessed and augmented to boost model effectiveness. Several models were trained and assessed—including InceptionV3, EfficientNetB7, and DenseNet201. Among them, a custom sequential model combining EfficientNet with U-Net delivered the best results, achieving a validation accuracy of 79.9% and outperforming the others with more consistent and accurate predictions.

The proposed solution offers several benefits: it generalizes well across different types of deepfakes, supports detailed pixel-level analysis, and is flexible enough to adapt to future enhancements. With improvements like a larger dataset, refined architecture, and ensemble learning, this system has strong potential to serve as a reliable tool in deepfake detection.

Chapter 1: Introduction

This chapter serves as the foundation for understanding the growing concern surrounding deepfake technology and its implications for digital media integrity. It begins by defining deepfakes and exploring their origins, followed by an examination of the technologies that enable their creation. The chapter highlights the potential risks associated with deepfakes, including misinformation, privacy violations, and their impact on public trust. Furthermore, it underscores the need for effective detection mechanisms to combat these threats. The objectives of the research are clearly outlined, emphasizing the aim to develop a robust detection system utilizing advanced deep learning techniques. Finally, the chapter concludes by presenting the structure of the thesis, guiding the reader through the subsequent sections of the research.

1.1 Introduction to the Project

Deepfake technology leverages deep learning algorithms to create highly realistic yet fake images and videos. With advancements in AI, it has become increasingly difficult to distinguish between real and manipulated media, leading to significant concerns across various domains, such as media integrity, security, and public trust. Deepfakes can be used to spread misinformation, defame individuals, and manipulate opinions. As a result, reliable detection methods are essential to ensure the authenticity of digital content.

Despite recent advancements, existing deepfake detection systems still face several challenges. They often struggle to perform effectively in real-world conditions, especially when dealing with low-resolution, compressed, or noisy videos. Many models have limited adaptability, making it difficult to keep pace with the rapidly evolving techniques used in creating deepfakes. Additionally, generalization remains a significant issue, as these models may perform poorly on new, unseen data or across diverse demographics. A common limitation is the neglect of temporal consistency, with a focus on analyzing individual frames rather than considering sequential frame-by-frame relationships. Detecting subtle manipulations presents another challenge, as minor alterations can be easily overlooked. Moreover, the high computational cost associated with many detection methods hinders scalability, particularly for real-time analysis. Ensuring models generalize well to novel deepfake types requires diverse and comprehensive training datasets, which are often lacking.

This project focuses on the detection of face-swap deepfake videos by employing deep learning techniques. The aim is to develop a robust detection system capable of accurately identifying subtle anomalies and artifacts that indicate manipulation in video content. By utilizing sophisticated neural network architectures such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), this system seeks to address the growing challenge of deepfake detection.

1.2 Motivation for the Project

The motivation for this project arises from the increasing misuse of deepfake technology to create manipulated content for malicious purposes. The ability to fabricate realistic videos has made it easier for malicious actors to disseminate fake information and conduct scams, resulting in potential harm to individuals and organizations. The need for reliable detection mechanisms is pressing, as current methods struggle to keep pace with the sophistication of deepfake generation techniques. By addressing this challenge, the project aims to contribute to maintaining the integrity of digital media and enhancing public trust.

1.3 Problem Definition

The proliferation of face-swap deepfake videos created using advanced deep technologies undermines the authenticity of digital media. Current detection methods are inadequate for identifying these sophisticated manipulations, leading to increased risks of misinformation and loss of trust in visual content. In today's digital age, deepfake videos are being used to spread false information, manipulate political opinions, and harm the reputation of individuals, posing significant ethical and security challenges. The ability to create realistic but fake content has fueled the spread of disinformation, incited social unrest, and even facilitated cybercrime, such as identity theft and blackmail. Consequently, the urgent need for reliable and effective detection solutions has become a crucial aspect of preserving public trust in digital communications.

1.4 Existing System

Several existing deepfake detection systems have been developed using different approaches, including convolutional neural networks (CNNs), recurrent neural networks (RNNs), attention mechanisms, and hybrid models. Below are some notable systems and their methodologies:

1. **FaceForensics++ (FF++)**
    - Dataset: A large-scale dataset of manipulated videos using various deepfake generation techniques.
    - Method: Uses EfficientNet and XceptionNet, leveraging transfer learning for deepfake detection.
    - Strengths: High accuracy on known manipulations, pre-trained models available.
    - Limitations: Performance drops on unseen deepfake types.

2. DeepFake Detection Challenge (DFDC)

- Dataset: Facebook's DFDC dataset containing diverse deepfake videos.
- Method: Uses CNN-based architectures like ResNet and EfficientNet, along with transformer-based models for better feature extraction.
- Strengths: Robust to different manipulation types.
- Limitations: Requires large-scale training and computational resources.

3. MesoNet

- Method: A lightweight CNN-based approach designed specifically for detecting deepfake-generated faces.
- Strengths: Computationally efficient and suitable for real-time applications.
- Limitations: Lower accuracy compared to deeper architectures like Xception.

4. Capsule Networks for Deepfake Detection

- Method: Uses capsule networks to capture subtle inconsistencies in deepfake videos, particularly detecting unnatural facial movements.
- Strengths: Good at detecting artifacts in manipulated images.
- Limitations: Computationally expensive and sensitive to adversarial attacks.

5. Hybrid CNN-RNN Approaches

- Method: Combines CNNs for spatial feature extraction and RNNs (LSTMs, GRUs) for temporal analysis across video frames.
- Strengths: Effectively detects inconsistencies in both image texture and temporal motion.
- Limitations: Requires more processing power and training time.

1.5 Drawback of the Existing System

Current deepfake detection systems face several challenges, including:

- **Inability to Detect Sophisticated Manipulations**: As deepfake techniques become more advanced, traditional detection methods often fail to identify subtle inconsistencies in manipulated videos.
- **High False Positive Rates**: Many existing systems produce false positives, leading to misclassification of genuine content as fake.
- **Resource-Intensive**: Some detection algorithms require substantial computational resources, making real-time detection difficult.
- **Limited Generalization**: Detection systems may perform well on specific datasets but fail to generalize across diverse deepfake techniques and scenarios.

1.6 Relevance of the Project

The project is relevant in the context of combating misinformation and ensuring the integrity of digital content in various fields, including journalism, law enforcement, and social media. By developing more accurate and efficient deepfake detection methods, this research contributes to the broader effort to protect digital media from manipulation and to build trust in information shared online.

Chapter 2: Literature Survey

A. Brief Overview of Literature Survey

This chapter consists of a review of existing research and patents related to deepfake detection. It covers the progression from early detection methods based on visual inconsistencies to more advanced approaches using deep learning techniques such as CNNs, attention mechanisms, and hybrid models. The studies reviewed highlight the need for specialized models to detect specific types of manipulations like face and expression swaps. Techniques such as transfer learning, frequency domain analysis, and multimodal inputs (combining audio and video) have shown improved performance. Furthermore, relevant patents demonstrate how these methods are being applied in real-world systems. Overall, the literature emphasizes the importance of adaptable and comprehensive strategies to keep pace with evolving deepfake technologies.

B. Related Works

2.1 Research Papers

[1] A. Kaur et al., "Deepfake video detection: challenges and opportunities," 2024.

- **Abstract:** This paper discusses the challenges and future opportunities for deepfake video detection. It emphasizes real-world applications and provides a review of existing techniques for robust detection in various scenarios.
- **Inference:** Generalization remains a challenge due to the evolving nature of deepfake techniques, but advancements in AI offer potential for significant improvements in detection.

[2] S. Waseem et al., "DeepFake on Face and Expression Swap: A Review," 2023.

- **Abstract:** The paper reviews detection methods specifically targeting face and expression swap deepfakes. It highlights techniques involving machine learning and deep learning, with emphasis on accuracy and detection precision.
- **Inference:** There is a need for methods that are tailored to face and expression swap detection, as these types of deepfakes pose unique challenges.

[3] M. S. Rana et al., “Deepfake detection: A systematic literature review,” 2022.

- **Abstract:** A comprehensive review of the various deepfake detection techniques used in current research, including traditional machine learning, deep learning, and hybrid methods.
- **Inference:** Deep learning approaches show promise in detection, though the rapid evolution of deepfake technology continues to present new challenges.

[4] S. R. Ahmed et al., "Analysis Survey on Deepfake detection and Recognition with Convolutional Neural Networks," 2022.

- **Abstract:** This survey explores the use of Convolutional Neural Networks (CNNs) for deepfake detection, focusing on the performance and limitations of CNN-based approaches.
- **Inference:** While CNNs can effectively detect deepfakes, the models must be updated continuously to keep pace with evolving techniques.

[5] D. Pan et al., "Advanced Deepfake Detection Using Inception-ResNet-v2," 2021.

- **Abstract:** The paper proposes an Inception-ResNet-v2 model for enhanced deepfake detection, using transfer learning to improve model accuracy in challenging scenarios.
- **Inference:** The Inception-ResNet-v2 architecture demonstrates superior performance, particularly in scenarios with subtle deepfake manipulations.

[6] A. Das et al., "Demystifying Attention Mechanism for Deepfake Detection," 2021.

- **Abstract:** This paper examines the use of attention mechanisms to enhance deepfake detection by improving feature extraction and classification accuracy.
- **Inference:** Integrating attention mechanisms into detection models increases their ability to identify subtle manipulations, improving overall accuracy.

[7] A. Kohli et al., "Detecting DeepFake, FaceSwap and Face2Face facial forgeries using frequency CNN," 2021.

- **Abstract:** The paper presents a frequency-based CNN model for detecting different facial forgeries by analyzing manipulated regions.
- **Inference:** Frequency domain analysis enhances detection accuracy by identifying specific artifacts introduced during deep fake manipulation.

[8] Y. Li et al., "Exposing DeepFake Videos By Detecting Face Warping Artifacts," 2019.

- **Abstract:** The authors propose a method to detect deep fake videos by analyzing face warping artifacts using visual inconsistency techniques.
- **Inference:** While effective for identifying warping artifacts, this approach struggles with deepfakes that involve more subtle modifications.

2.2 Patent search

a. European Patents

[1] J. Smith and L. Wang, Methods and Systems for Detecting Fraud During Biometric Identity Verification, European Patent EP4300446A1, Jan. 3, 2024.

- **Abstract:** This system integrates image quality analysis and temporal consistency checks to flag inconsistencies in biometric videos and identify potential deepfakes.
- **Inference:** Combining spatial and temporal data provides a higher likelihood of catching synthetic inconsistencies in deepfake media.

[2] S. M. Mathews, *Methods and Apparatus to Perform Deepfake Detection Using Audio and Video Features*, European Patent EP4050571A1, Aug. 31, 2022.

- **Abstract:** This patent presents methods and apparatuses that use both audio and video features to detect deepfakes. It leverages multimodal data analysis to enhance detection robustness in real-time applications.
- **Inference:** Multimodal analysis combining audio and video streams significantly improves detection accuracy, especially for real-time systems.

[3] A. K. Jain *et al.*, *Robust Spoofing Detection System Using Deep Residual Neural Networks*, European Patent EP4097717A1, Dec. 7, 2022.

- **Abstract:** This invention discloses a spoof detection system using deep residual neural networks to detect deepfakes and other forms of digital forgeries in images and videos.
- **Inference:** Deep residual networks offer increased sensitivity to subtle artifacts, but require high computational efficiency for deployment.

[4] M. Müller *et al.*, *Facial Authentication Anti-Spoofing with Ultrasound Produced Hair Movements Detection*, European Patent EP4016340A1, Jun. 22, 2022.

- **Abstract:** This patent uses ultrasound signals to detect micro hair movements that are not replicable in deepfake or spoofed content, enhancing the authenticity of facial recognition systems.
- **Inference:** Biometric liveness detection through physical micro-movements provides an additional robust defense against deepfakes.

b. US Patents

[1] A. O'Donovan and A. Khan, *Deepfake Detection*, U.S. Patent US20240355337A1, Feb. 1, 2024.

- **Abstract:** The system applies adversarial machine learning techniques to train models that are resistant to deception by generative models like GANs.
- **Inference:** Adversarial training introduces robustness to unseen deepfake formats, but requires constant retraining to handle new threats.

[2] K. S. Kim and S. Kim, *Apparatus and Method for Detecting Deepfake Based on Convolutional Long Short-Term Memory Network*, U.S. Patent US20230281461A1, Sep. 7, 2023.

- **Abstract:** This patent describes a detection framework combining CNNs and LSTM networks to capture spatial and temporal features in videos for deepfake identification.
- **Inference:** CNN-LSTM hybrid models are well-suited for video-based detection as they effectively capture both static and motion-related anomalies.

[3] B. Guo and Y. Li, *Real-Time Face Swapping System and Methods Thereof*, U.S. Patent US20230237778A1, Jul. 27, 2023.

- **Abstract:** This invention outlines a real-time face-swapping mechanism, which can be reverse-engineered to identify common manipulation patterns in face-based deepfakes.
- **Inference:** Understanding generation techniques aids in building more accurate detectors targeting face-swapping artifacts.

[4] N. Moorthy and M. L. Wilson, *Methods and Systems for Detecting Deepfakes*, U.S. Patent US20210142065A1, May 13, 2021.

- **Abstract:** The method focuses on detecting inconsistencies in eye-blinking rates, head movements, and lip-syncing as signs of deepfake media.
- **Inference:** Behavioral biometrics can serve as telltale signs of synthetic content and offer lightweight yet effective detection cues.

[5] J. Morency and H. Kaur, *Audiovisual Deepfake Detection*, U.S. Patent US20220121868A1, Apr. 21, 2022.

- **Abstract:** This patent outlines a method to detect deepfakes by analyzing mismatches between facial expressions and corresponding speech.
- **Inference:** Cross-modal inconsistencies (audio-visual) help uncover manipulations that appear natural when analyzed in isolation.

2.3 Inference Drawn

Existing research on deepfake detection highlights key challenges and advancements in the field. One of the primary issues is the difficulty in generalizing detection models, as deepfake techniques continue to evolve, making it necessary for detection systems to adapt rapidly [1]. Specialized approaches are required for detecting face and expression swaps, as these types of manipulations introduce unique artifacts that traditional models may fail to recognize [2]. Deep learning techniques, particularly CNNs, have shown significant promise in improving detection accuracy; however, continuous model updates are essential to counter emerging deepfake methods [3][4]. Advanced architectures such as Inception-ResNet-v2 have demonstrated superior performance in detecting subtle manipulations, particularly when combined with transfer learning [5]. Additionally, attention mechanisms have been explored to enhance feature extraction, leading to improved classification accuracy [6]. Frequency-based analysis provides another promising avenue, as CNN models trained on frequency domain features have proven effective in identifying forged regions [7]. Furthermore, techniques based on detecting visual inconsistencies, such as face warping artifacts, have been useful for specific deepfake types but struggle against more refined manipulations [8]. These studies collectively indicate that while deep learning continues to improve deepfake detection, future research must focus on enhancing model adaptability, integrating multiple detection strategies, and addressing the limitations posed by evolving deepfake generation techniques. Additionally, multimodal approaches, combining audio and video analysis, have proven particularly effective in real-time detection, allowing for better handling of complex manipulations involving both visual and auditory components [9]. Moreover, behavioral biometrics, such as eye-blinking rates and lip-syncing, have emerged as promising indicators of synthetic content, offering an additional layer of defense against deepfakes [10]. Finally, biometric liveness detection, using innovative methods like ultrasound to detect micro hair movements, provides a physical defense against deepfakes, further strengthening detection models [11]. In conclusion, while significant progress has been made in deepfake detection, the continuous evolution of manipulation techniques necessitates ongoing advancements in detection methods, with multimodal approaches, behavioral biometrics, and liveness detection playing key roles in addressing these challenges.

2.3 Comparison of Existing Systems and Proposed Area of Work

In contrast to existing systems, which often rely on traditional methodologies and are limited in scope, the proposed area of work focuses on developing a comprehensive deepfake detection system that integrates advanced deep learning techniques and a robust training framework. By employing a hybrid model that combines CNNs and RNNs, the proposed system aims to enhance detection accuracy and minimize false positives.

Furthermore, the proposed approach will utilize a more diverse dataset, incorporating a wide range of deepfake types to improve the model's generalization capabilities. This will allow the system to adapt more readily to emerging manipulation techniques. Additionally, the integration of user-friendly interfaces and educational resources is a key focus area, ensuring that end-users are better equipped to understand and combat deepfake threats effectively.

Paper (Year)	**Objective**	**Dataset**	**Methodology**	**Performance Measure**	**Results Achieved**	**Inference**
Ahmed et al. (2022) [4]	Deepfake detection using CNNs	FaceForensics++, Celeb-DF	CNN-based approach	Accuracy, F1-score	Accuracy: 89.6% (FaceForensics++), 86.4% (Celeb-DF)	CNNs effectively detect manipulated media but require frequent updates for new deepfake techniques
Kohli et al. (2021) [7]	Frequency-based CNN for detecting deepfake forgeries	FaceForensics++, DFDC	Frequency CNN	Accuracy, AUC	Accuracy: 91.2%, AUC: 0.94	Targets manipulated regions in frequency domain, limited to facial forgeries, struggles with other deepfake types
Das et al. (2021) [6]	Role of attention mechanisms in deepfake detection	FaceForensics++, Celeb-DF	Attention-based CNN	Accuracy, Precision, Recall	Accuracy: 90.5%, Precision: 89.8%, Recall: 91.3%	Enhances feature extraction and classification, may require high computational resources
Pan et al. (2021) [5]	Deepfake detection using transfer learning	FaceForensics++, DFDC	Inception-ResNet-v2	Accuracy, F1-score	Accuracy: 92.7%, F1-score: 0.91	Transfer learning improves detection accuracy, may not generalize well to unseen deepfake techniques
Li et al. (2019) [3]	Detecting deepfake face-swapping artifacts	Deepfake TIMIT, FaceForensics++	Visual Inconsistency Detection	Accuracy, Precision, Recall	Accuracy: 87.3%, Precision: 86.2%, Recall: 88.1%	Effective in identifying warping artifacts, struggles with subtle modifications

Table 2.3: Comparison of Existing Systems

Chapter 3: Requirement Gathering for the Proposed System

In this chapter, the requirements for the proposed deepfake detection system are outlined. Requirements refer to the essential functionalities, constraints, and specifications that the system must meet to effectively address the challenges posed by deepfake technology. Gathering requirements is a critical step in the development process, ensuring that the system is designed to fulfill its intended purpose. This section discusses the methods used to gather requirements.

3.1 Introduction to requirement gathering

Requirement gathering is the process of identifying and defining the needs of a system before development begins. It ensures that the deepfake detection system meets its intended purpose by outlining essential functionalities, performance criteria, and constraints.
For this project, requirements were gathered using:

- Research and Literature Review: Studying existing deepfake detection techniques and challenges.
- User Needs Analysis: Understanding what users expect from the system, such as accuracy and ease of use.
- Technical Considerations: Identifying the necessary hardware, software, and algorithms for effective detection.

This step helps in building a system that is functional, efficient, and user-friendly.

3.2 Functional Requirements

These are the core features and behaviors that your deepfake detection system should exhibit. Focus on what the system must do:

1. Dataset Integration and Labeling:
The system shall load a total of 4,000 images from two sources—2,000 real images from the FFHQ dataset and 2,000 fake images from the 1 Million Fake Faces dataset—and assign binary labels (0 for real, 1 for fake) to each image for classification.

2. Image Preprocessing and Augmentation:
All images shall be resized to 256×256 pixels, normalized to pixel values between 0 and 1, and augmented using techniques like horizontal flipping, ±20° rotation, Gaussian blur, and brightness adjustments to improve model generalization on the limited 4,000-image dataset.

3. Dataset Splitting:

The system shall automatically divide the dataset into training (70%), validation (15%), and testing (15%) sets, ensuring balanced representation of real and fake images in each subset for unbiased model evaluation.

4. Deep Learning Model Selection and Configuration:

The system shall support the use of pre-trained models—InceptionV3, EfficientNetB7, and DenseNet201—by applying transfer learning and modifying their final layers for binary classification (real vs. fake), reducing training time while improving accuracy on the 4,000-image dataset.

5. Model Training with Optimization:

During training, the system shall use the Adam optimizer and binary cross-entropy loss function, implement early stopping to prevent overfitting, and save the best model checkpoint based on validation accuracy and recall metrics.

6. Model Evaluation and Metrics Reporting:

After training, the system shall evaluate model performance using accuracy, loss, recall, precision, and F1-score, and visualize results using confusion matrices and ROC curves to select the most reliable deepfake detection model.

3.3 Non-Functional Requirements

These are criteria that judge the performance and quality of the system:

1. Performance Requirement:

The system shall achieve a minimum detection accuracy of **90%** and a recall of at least **92%** on the test dataset of 600 images to ensure reliable classification between real and fake images.

2. Scalability Requirement:

The system shall be capable of handling an increase in dataset size (e.g., from 4,000 to 40,000 images) without significant degradation in training or inference performance by using scalable deep learning frameworks like TensorFlow/Keras.

3. Reliability Requirement:

The system shall ensure consistent predictions by utilizing model checkpointing and early stopping during training to avoid overfitting and preserve the best-performing model for deployment.

4. Maintainability Requirement:

The system codebase shall be modular, separating functions for data preprocessing, training, evaluation, and prediction, allowing easy debugging, updates, or replacement of components like models or augmentation techniques.

5. Security Requirement:

The system shall ensure secure handling of input images by processing them locally and not transmitting user data externally, thereby maintaining data confidentiality and protecting against misuse.

3.4 Hardware & Software Requirements

Hardware:

- GPU: NVIDIA Tesla/GeForce GPU for efficient model training and real-time inference.
- RAM: Minimum 16GB of RAM, preferably 32GB, for processing high-definition videos.
- Storage: At least 1TB storage to handle large datasets of video files.
- CPU: Multi-core processor (e.g., Intel i7 or AMD Ryzen 7) for system operations.

Software Requirements:

- Programming Languages: Python (for model implementation), JavaScript (for the front-end)
- Libraries: TensorFlow, PyTorch, NumPy, Scikit-learn
- Frameworks: Flask or Django for web development
- Tools: Google CoLab, Git for version control

3.5 Constraints

Constraints are the limitations or challenges you face during the development of the system:

- Dataset Availability: High-quality datasets with labeled deepfake and real videos may be limited.
- Computational Resources: Training deep learning models, especially for video processing, is resource-intensive and may require high-performance GPUs.
- Time: Given the complexity of the project, time may be a constraint in fully developing the system by the deadline.
- Ethical Considerations: The system must be designed with ethics in mind, ensuring it’s used only for detection and not for any malicious purposes.

Chapter 4: Proposed Design

The design of the proposed system involves a detailed representation of the architecture, modular components, and various diagrams that illustrate the system's functionality and workflow. It outlines how the components interact, from data input to model evaluation. This chapter also highlights the integration of preprocessing, model selection, and evaluation modules in a structured pipeline. To provide a comprehensive understanding, the chapter includes the system's block diagram and modular diagram, offering visual insights into the flow and distribution of tasks across the architecture.

4.1 Block Diagram Representation of the Proposed System

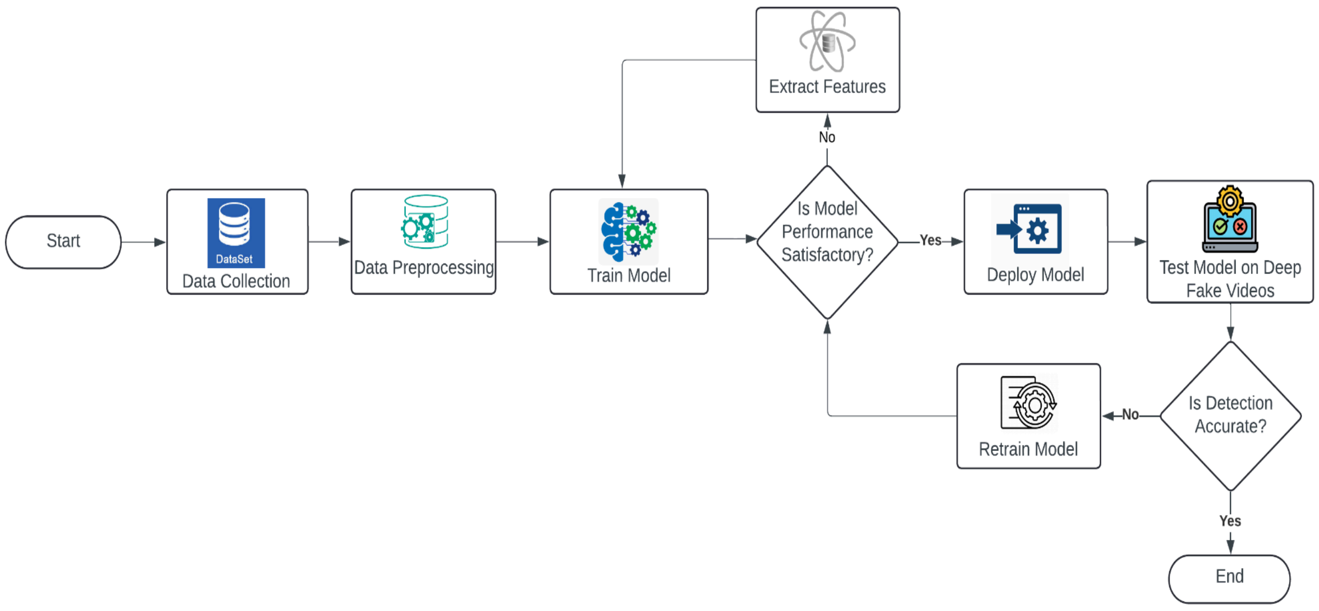


Fig 4.1: Block Diagram of Deepfake System

Figure 4.1 illustrates the deepfake detection pipeline, outlining the systematic process for developing and deploying a deep learning model for deepfake identification.

The pipeline begins with data collection, where real and deepfake video datasets are acquired. These datasets undergo Preprocessing, which includes resizing, normalization, and augmentation to enhance data quality. The preprocessed data is then used to train the model. If the model's performance is unsatisfactory, feature extraction is refined, and the model is retrained. Once the model meets the desired performance criteria, it is deployed and tested on deepfake videos. If the detection accuracy remains inadequate, the model undergoes further retraining. The process concludes once the model achieves the required accuracy threshold, ensuring robust and reliable deepfake detection.

4.2 Modular Diagram Representation of the Proposed System

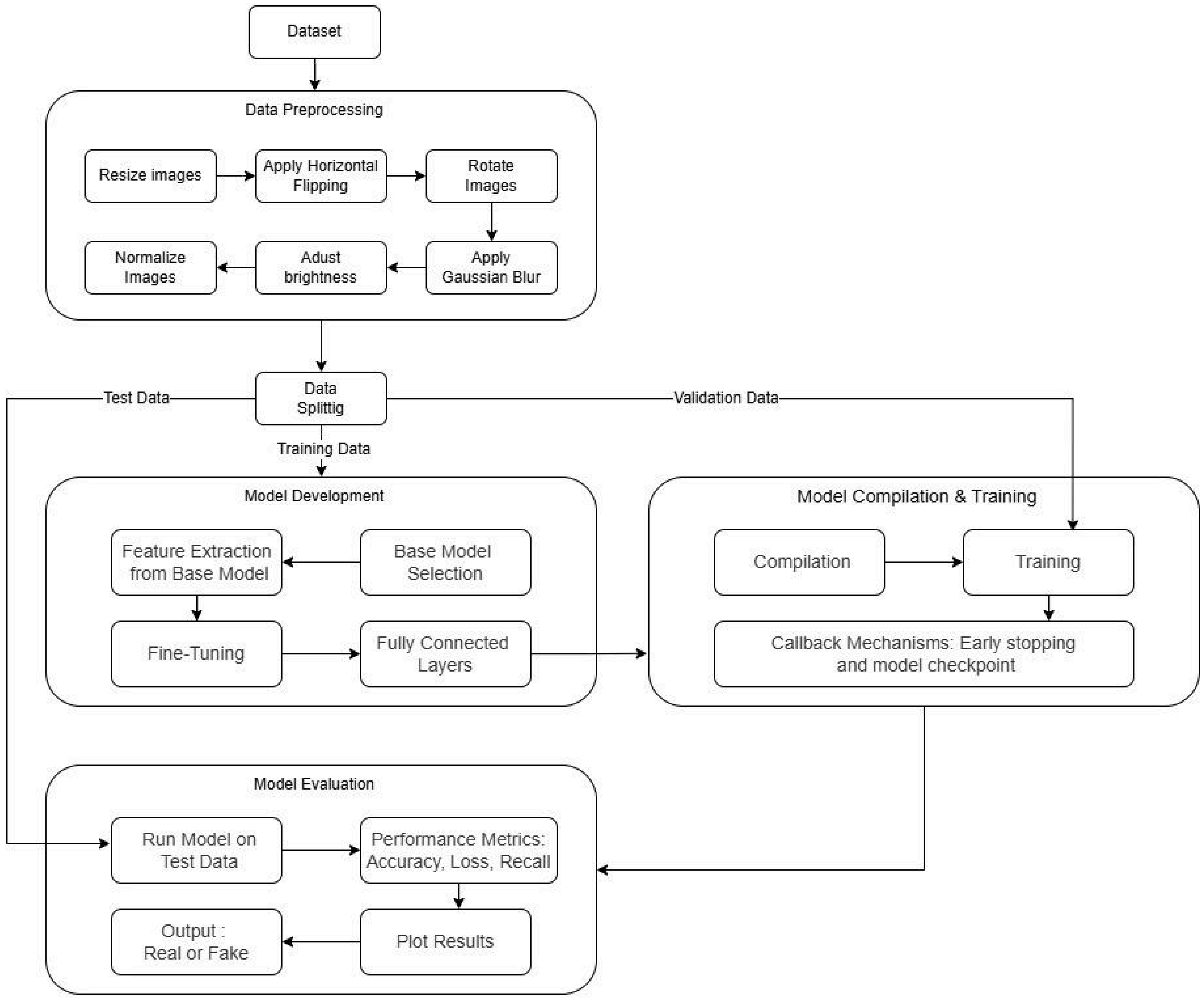


Fig 4.2: Modular Diagram of deepfake System

Figure 4.2 outlines the system workflow for detecting face-swap deepfakes. The process starts with collecting a dataset of real and fake images, which are resized to 224×224 pixels and normalized for uniformity. Data augmentation techniques such as horizontal flipping, rotation, Gaussian blur, and brightness/contrast adjustment are applied to improve model generalization and reduce overfitting. The dataset is then split into training, validation, and test sets. Model development involves training and evaluating multiple architectures—InceptionV3, EfficientNetB7, DenseNet201, and a custom sequential model combining EfficientNet and U-Net. Transfer learning is used for feature extraction, followed by fine-tuning. The models are compiled using Binary Cross-Entropy loss and optimized with Adam. Early Stopping and Model

Checkpoint callbacks help stabilize training. Finally, models are evaluated using metrics like accuracy, recall, and loss, with the system outputting whether an image is real or fake.

4.3 Project Scheduling & Tracking using Timeline / Gantt Chart

Task	**Start Date**	**End Date**	**Duration**
Project Initiation	Aug 9, 2024	Aug 16, 2024	7 days
Literature Review	Aug 17, 2024	Aug 24, 2024	1 week
Dataset Collection and Preparation	Aug 24, 2024	Sep 14, 2024	3 weeks
Data Preprocessing	Sep 15, 2024	Sep 29, 2024	2 weeks
Feature Selection & Extraction	Oct 1, 2024	Oct 8, 2024	1 week
Model Development (Multiple Models)	Oct 8, 2024	Dec 10, 2024	9 weeks
Model Evaluation & Comparison	Dec 12, 2024	Jan 2, 2025	3 weeks
Best Model Selection & Finalization	Jan 2, 2025	Jan 7, 2025	5 days
Testing & Performance Analysis	Jan 7, 2025	Jan 12, 2025	4 weeks
Results Compilation & Visualization	Jan 14, 2025	Jan 28, 2025	3 weeks
Final Implementation Enhancements	Feb 10, 2025	Mar 10, 2025	4 weeks
Documentation & Report Writing	Mar 10, 2025	Apr 10, 2025	4 weeks
Project Review & Final Submission	Apr 10, 2025	Apr 28, 2025	3 weeks

Table 4.3: Project Scheduling

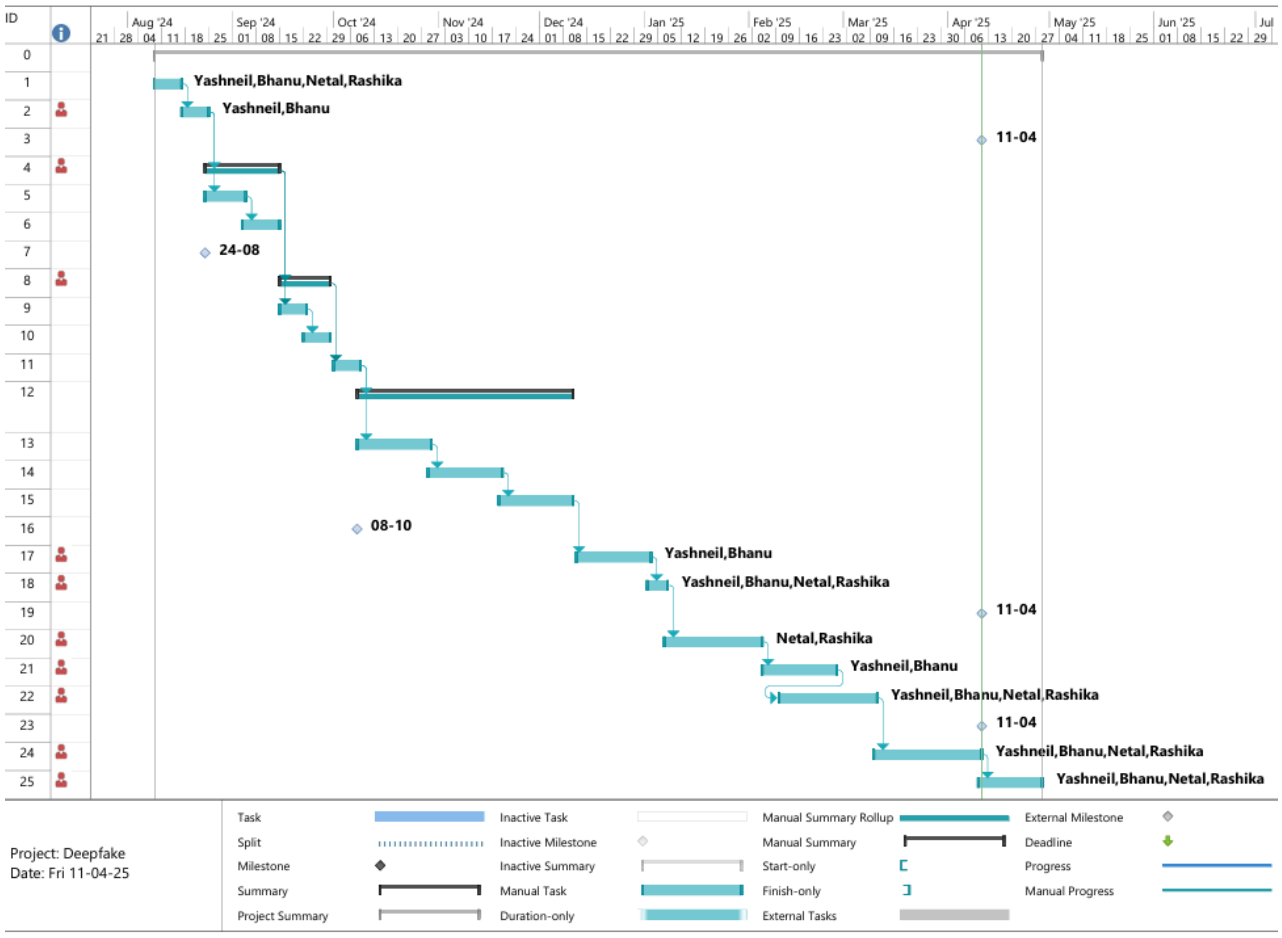


Fig 4.3: Gantt Chart

Chapter 5: Implementation of the Proposed System

5.1 Methodology employed for development

The dataset used for this project comprises 4000 images, consisting of both real and fake face samples. To ensure consistency during model training, all images were standardized to a uniform resolution. During the data preprocessing phase, images were resized to a fixed dimension and subjected to augmentation techniques such as horizontal flipping, rotation, Gaussian blur, and brightness adjustment. These augmentations help the model generalize better across various scenarios. Additionally, pixel values were normalized to stabilize the training process.

For model selection and training, multiple deep learning architectures were explored to identify the most effective one for detecting face-swapped deepfakes. Transfer learning was employed to leverage pre-trained knowledge for efficient feature extraction, while hyperparameters were optimized to improve accuracy and generalization capabilities. During model compilation and training, an appropriate loss function was selected for binary classification, and an optimizer was applied to improve convergence. Callbacks such as early stopping and model checkpointing were implemented to prevent overfitting and retain the best-performing model.

Finally, the models were evaluated using metrics such as accuracy, loss, and recall. This evaluation helped compare the performance of different architectures and identify the most suitable model for face-swap deepfake detection.

5.2 Algorithms and flowcharts for the respective modules developed

A. Data Preprocessing Module

In the data preprocessing phase, the system begins by loading a curated dataset of real and fake face images. Each image is resized to 224×224 pixels to maintain dimensional consistency, which is critical for efficient training across all deep learning models. To enhance generalization, data augmentation techniques are applied, including random horizontal flipping, rotation within ±20 degrees, Gaussian blur, and adjustments in brightness and contrast. These augmentations simulate real-world distortions and help the models become robust to variability in input data. Finally, all pixel values are normalized to the range [0, 1] to stabilize the learning process and accelerate convergence. The preprocessed dataset is then split into training and validation subsets to facilitate performance evaluation.

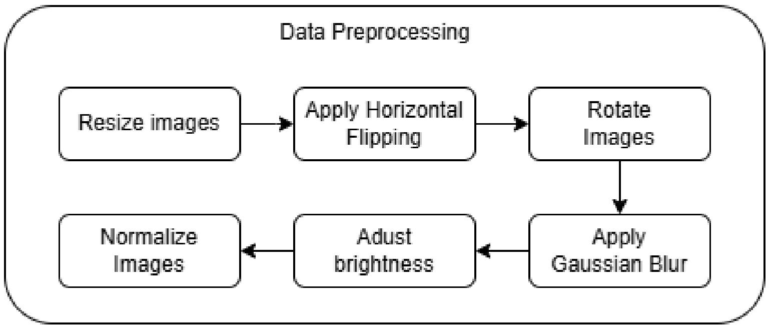


Fig 5.2.1: Data Preprocessing Flowchart

B. Model Development & Training Module

This module involves selecting and configuring multiple deep learning models—namely InceptionV3, EfficientNetB7, and DenseNet201—to identify the best-performing architecture for deepfake detection. Each model is initialized with pre-trained weights to leverage transfer learning, enhancing feature extraction with reduced computational cost. The final classification layer is adapted for binary output (real or fake), and the models are compiled using the binary cross-entropy loss function and the Adam optimizer. Accuracy and recall are monitored throughout training. To avoid overfitting and ensure optimal performance, early stopping and model checkpointing techniques are used. The training process continues until the model reaches optimal validation performance, after which the best version is saved.

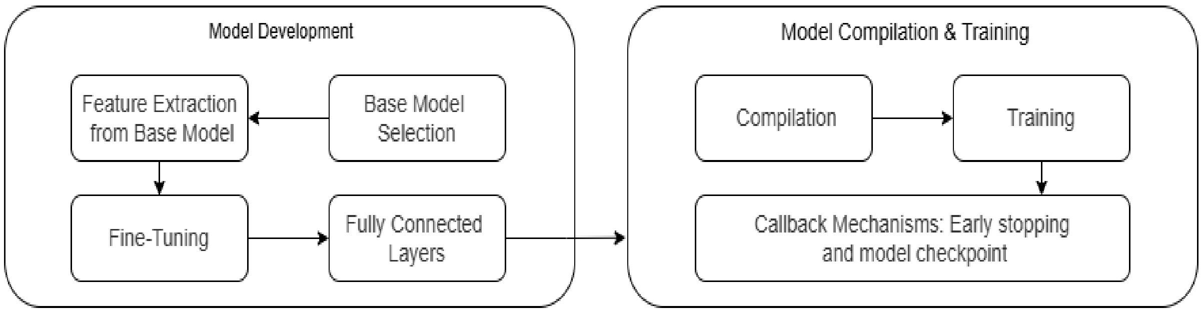

Model Development
Feature Extraction from Base Model
Base Model Selection
Fine-Tuning
Fully Connected Layers
Model Compilation & Training
Compilation
Training
Callback Mechanisms: Early stopping and model checkpoint

Fig 5.2.2: Model Development & Training Flowchart

C. Model Evaluation Module

After training, each model is evaluated using the validation dataset. The system loads the trained models and makes predictions to assess key metrics such as accuracy, loss, and recall. These metrics provide insight into the model's generalization ability and detection capability. Models are compared based on their performance consistency and reliability in identifying fake images. The model that achieves the highest validation accuracy and recall with minimal loss is selected as the final detection system. This approach ensures that the chosen model not only performs well on seen data but also maintains robustness on unseen inputs.

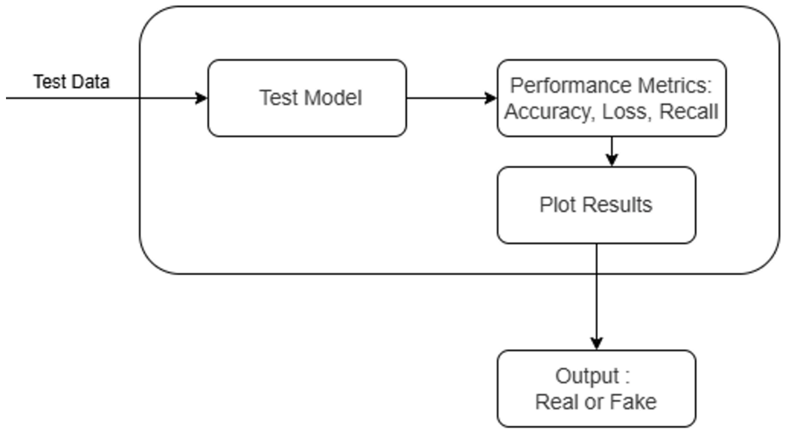


Fig 5.2.3: Model Evaluation Flowchart

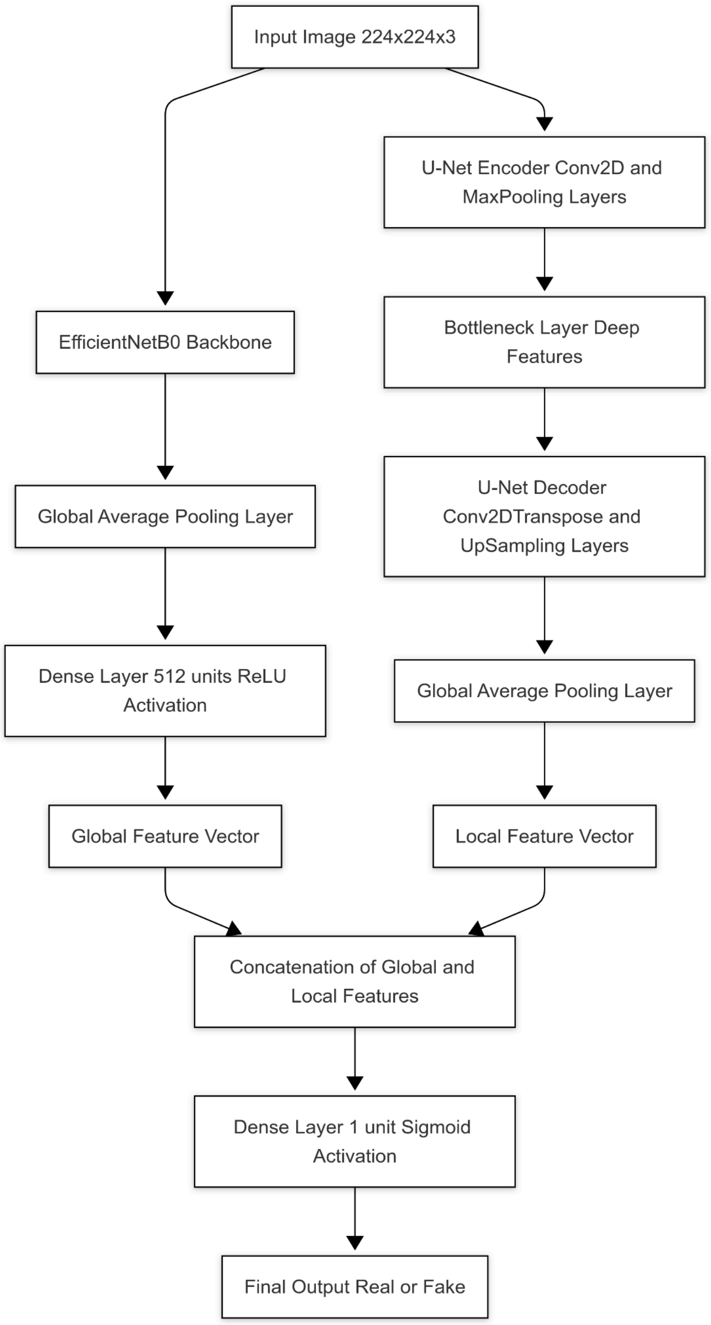
5.3 Datasets source and utilization

For this study, a total of 4,000 face images were carefully selected to create a balanced dataset of real and fake facial images. The real images were chosen from the Flickr-Faces-HQ (FFHQ) dataset, a collection developed by NVIDIA that includes high-quality photographs of human faces. This dataset is known for its wide variety of faces across different age groups, ethnic backgrounds, and facial characteristics, making it a reliable source for authentic samples.

To represent the fake images, samples were taken from the 1 Million Fake Faces dataset, which contains synthetic faces generated using StyleGAN, a well-known generative adversarial network. This dataset, shared by Bojan Komatović, is widely used in research focused on detecting AI-generated content due to the realistic quality of the fake faces it contains.

In order to maintain consistency across the dataset and to support effective model training, all images were resized to 256×256 pixels. Standardizing the image size ensures that the model receives uniform input, which helps improve learning efficiency and reduces any bias that might result from varying image dimensions.

5.4 Proposed system architecture



Input Image 224x224x3
U-Net Encoder Conv2D and MaxPooling Layers
EfficientNetB0 Backbone
Bottleneck Layer Deep Features
Global Average Pooling Layer
U-Net Decoder Conv2DTranspose and UpSampling Layers
Dense Layer 512 units ReLU Activation
Global Average Pooling Layer
Global Feature Vector
Local Feature Vector
Concatenation of Global and Local Features
Dense Layer 1 unit Sigmoid Activation
Final Output Real or Fake

Fig 5.4: Architecture Diagram

1. EfficientNetB0: Global Feature Extractor

EfficientNetB0 is a pre-trained CNN known for its accuracy and efficiency. It uses compound scaling to balance network depth, width, and input resolution. Built using MBConv blocks and Swish activation, it captures hierarchical image features from low-level textures to high-level semantics. The network ends with a Global Average Pooling layer, which compresses spatial information into a 512-dimensional vector.

Role in Deepfake Detection:

Acts as a global observer. It identifies broad inconsistencies in face structure, lighting, and expressions providing semantic understanding crucial for distinguishing real from fake images.

2. Custom U-Net: Local Feature Decoder

The model includes a lightweight, modified U-Net, originally designed for segmentation but adapted here for classification. It uses a basic encoder-decoder structure with Conv2D and MaxPooling layers for encoding, and Conv2DTranspose with UpSampling for decoding. Instead of outputting a segmented map, a Global Average Pooling layer flattens the decoded features into a vector.

Role in Deepfake Detection:

Focuses on pixel-level inconsistencies like tampering edges, micro-expressions, and GAN-induced noise. Acts like a microscope, capturing subtle visual clues that may be missed by standard classification networks.

3. Fusion Strategy: Combining Global and Local Features

Both EfficientNetB0 and the custom U-Net process the same input image. The two resulting feature vectors one encoding global semantic patterns and the other encoding local fine-grained artifacts are concatenated into a unified representation. This merged vector is passed through a final Dense layer with sigmoid activation to predict whether the input is real or fake.

Deepfakes often appear realistic globally but have tell-tale artifacts locally. EfficientNet handles overall visual coherence, while U-Net highlights detailed inconsistencies. The fusion improves robustness, enabling the model to generalize across various deepfake generation techniques.

Chapter 6: Testing of the Proposed System

6.1 Introduction to testing

Testing is a critical phase in the software development lifecycle aimed at evaluating the correctness, performance, and robustness of the proposed deepfake face swap detection system. This phase ensures that the system meets the functional and non-functional requirements, operates reliably across varied inputs, and generalizes well to unseen data. The testing process validates model accuracy, response time, and error handling under different scenarios.

6.2 Types of tests considered

The system underwent multiple levels of testing to ensure reliability and effectiveness. Unit testing was performed on individual components such as data preprocessing functions and model utilities. Integration testing ensured that modules such as data loading, model inference, and evaluation metrics interacted correctly. System testing was carried out to validate the overall pipeline, from input image ingestion to output classification.

6.3 Various test case scenarios considered

Test Case No.	**Test Case Description**	**Expected Output**	**Observed Output**
TC01	System launches successfully	System starts	System starts
TC02	Upload a valid real face image	Real	Real
TC03	Upload a valid fake face image	Fake	Fake
TC04	Upload an unsupported file format (.txt)	Error	Error
TC05	Upload a low-quality real image	Real	Fake *(Misclassified)*
TC06	Upload a high-resolution fake image	Fake	Fake
TC07	Re-upload the same image	Same prediction	Same prediction
TC08	Upload image with no face	Error	Error
TC09	Upload a borderline fake (subtle manipulation)	Fake	Real *(Misclassified)*
TC10	Test resize_image() function	Image is resized to 224×224 pixels	Resizing successful; shape verified

Table 6.3: Test Case Summary

6.4 Inference drawn from the test cases

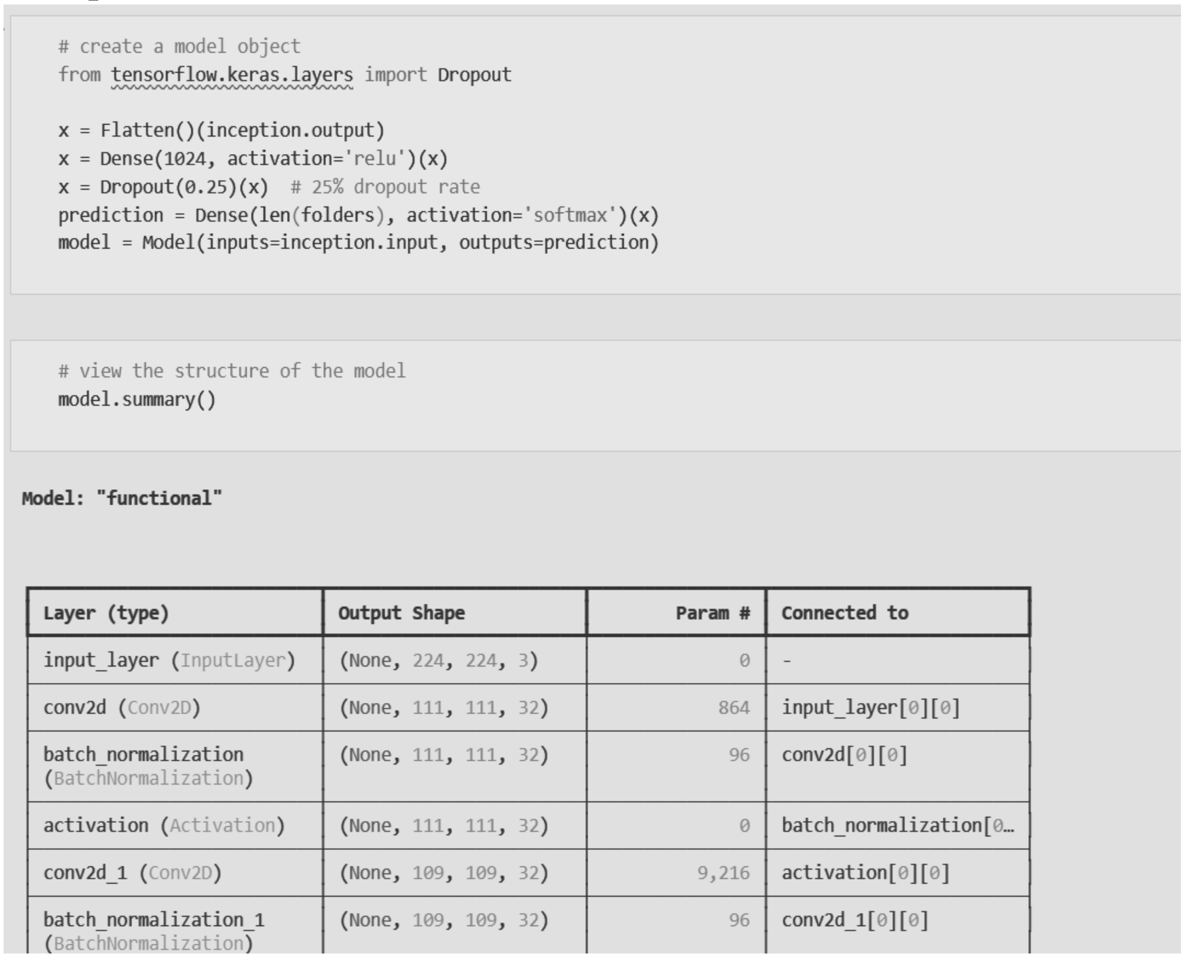
The testing phase revealed that the deepfake detection system is functionally sound and performs as expected under standard conditions. The system correctly classified most real and fake images, and it appropriately handled invalid inputs such as unsupported file types or empty files. However, a few misclassifications were observed, especially in cases involving low-quality real images and subtly manipulated fake images. These misclassifications suggest that the model may benefit from enhanced training on edge-case data or improved robustness to subtle manipulations.

Despite these minor inaccuracies, the system consistently produced reliable results across repeated tests, indicating stability and consistency in its inference pipeline. The performance on basic usability and prediction functionality confirms the system’s readiness for further optimization and deployment.

Chapter 7: Results and Discussion

7.1 Implementation

A. IncptionNetV3



```
# create a model object
from tensorflow.keras.layers import Dropout

x = Flatten()(inception.output)
x = Dense(1024, activation='relu')(x)
x = Dropout(0.25)(x)  # 25% dropout rate
prediction = Dense(len(folders), activation='softmax')(x)
model = Model(inputs=inception.input, outputs=prediction)
```

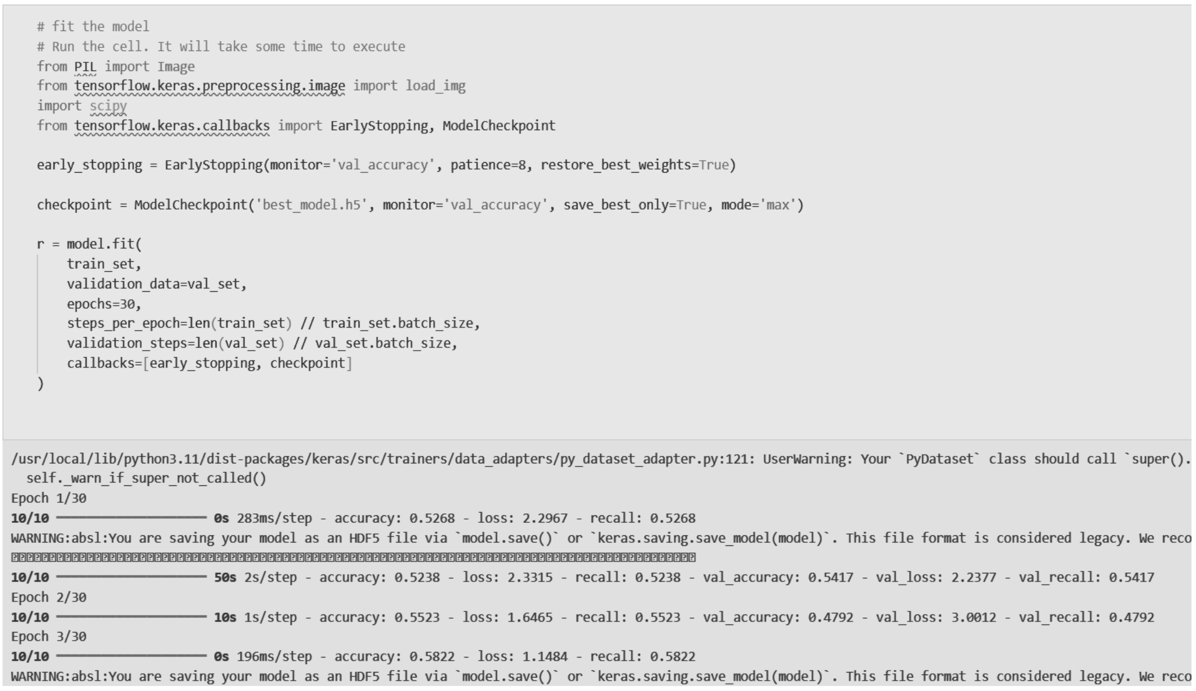
```
# view the structure of the model
model.summary()
```

Model: "functional"

Layer (type)	Output Shape	Param #	Connected to
input_layer (InputLayer)	(None, 224, 224, 3)	0	-
conv2d (Conv2D)	(None, 111, 111, 32)	864	input_layer[0][0]
batch_normalization (BatchNormalization)	(None, 111, 111, 32)	96	conv2d[0][0]
activation (Activation)	(None, 111, 111, 32)	0	batch_normalization[0…
conv2d_1 (Conv2D)	(None, 109, 109, 32)	9,216	activation[0][0]
batch_normalization_1 (BatchNormalization)	(None, 109, 109, 32)	96	conv2d_1[0][0]

Fig 7.1.1:InceptionNetV3: Architectural summary

Fig 7.1.1 illustrates the architectural summary of the InceptionV3-based model. The base InceptionV3 network is extended using additional layers such as Flatten, Dense, and Dropout, which help in classification and regularization. The output layer uses a softmax activation to predict whether the input image is real or fake. Batch normalization and ReLU activation are used after convolutional layers to enhance learning efficiency and reduce overfitting. The architecture showcases a well-structured pipeline for feature extraction and classification tailored for binary deepfake detection.

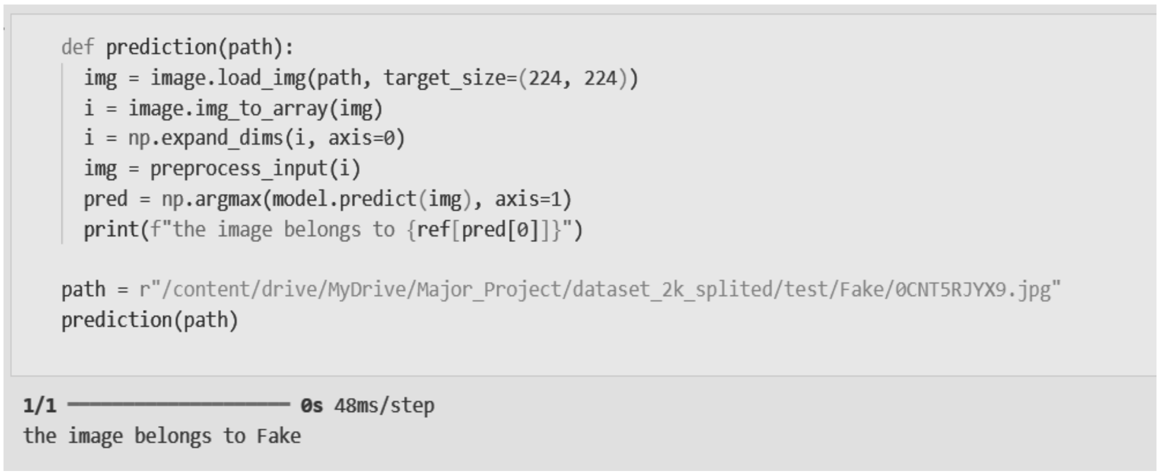


```
# fit the model
# Run the cell. It will take some time to execute
from PIL import Image
from tensorflow.keras.preprocessing.image import load_img
import scipy
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint

early_stopping = EarlyStopping(monitor='val_accuracy', patience=8, restore_best_weights=True)

checkpoint = ModelCheckpoint('best_model.h5', monitor='val_accuracy', save_best_only=True, mode='max')

r = model.fit(
    train_set,
    validation_data=val_set,
    epochs=30,
    steps_per_epoch=len(train_set) // train_set.batch_size,
    validation_steps=len(val_set) // val_set.batch_size,
    callbacks=[early_stopping, checkpoint]
)
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class should call `super().
  self._warn_if_super_not_called()
Epoch 1/30
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 283ms/step - accuracy: 0.5268 - loss: 2.2967 - recall: 0.5268
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We reco
▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯
10/10 ━━━━━━━━━━━━━━━━━━━━ 50s 2s/step - accuracy: 0.5238 - loss: 2.3315 - recall: 0.5238 - val_accuracy: 0.5417 - val_loss: 2.2377 - val_recall: 0.5417
Epoch 2/30
10/10 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.5523 - loss: 1.6465 - recall: 0.5523 - val_accuracy: 0.4792 - val_loss: 3.0012 - val_recall: 0.4792
Epoch 3/30
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 196ms/step - accuracy: 0.5822 - loss: 1.1484 - recall: 0.5822
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We reco
```

Fig 7.1.2: InceptionNetV3: Training

Fig 7.1.2 displays the model training process. It involves fitting the model using a training and validation set, incorporating early stopping and model checkpointing to prevent overfitting and retain the best-performing model. The console output logs the accuracy, loss, and recall metrics for each epoch, demonstrating performance improvements during training. Despite some fluctuation in early epochs, the model shows progressive refinement across iterations.



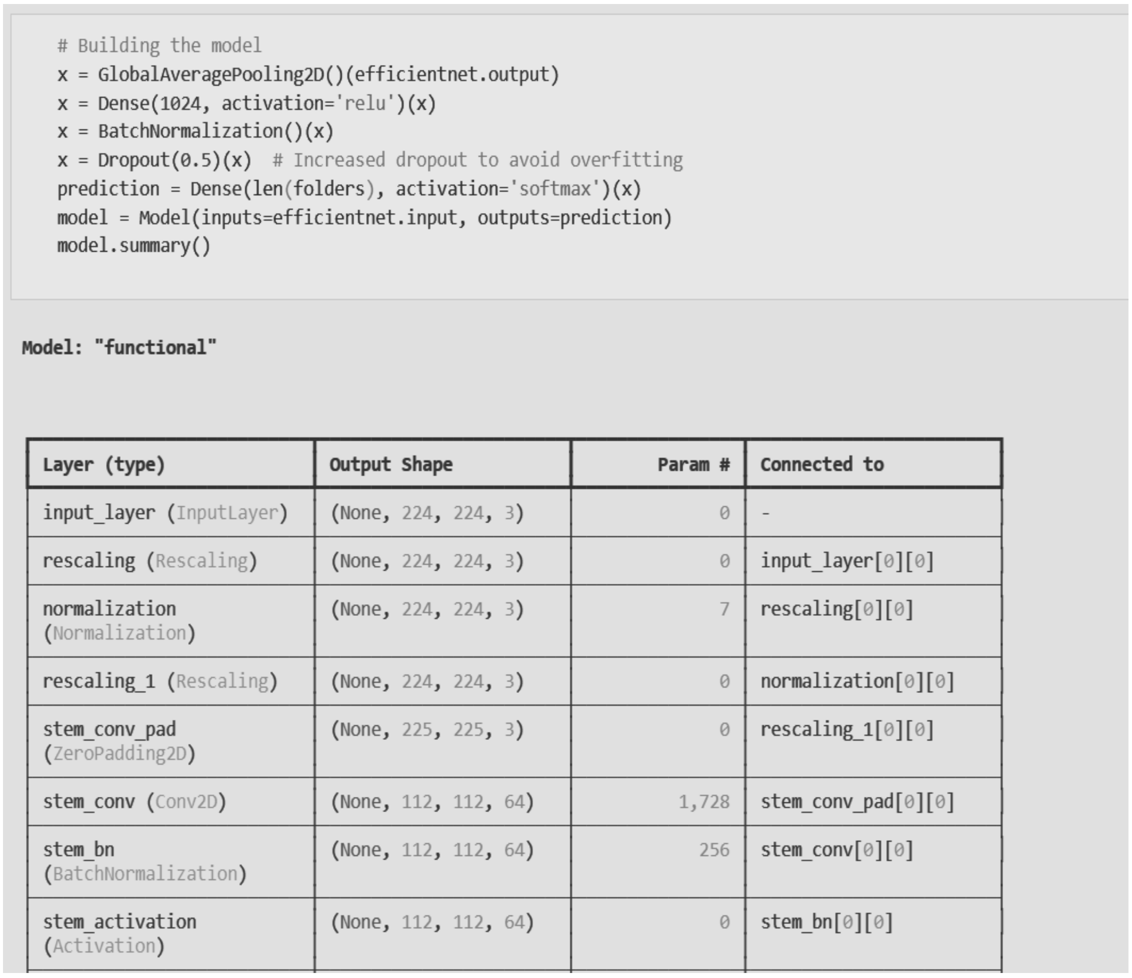
```
def prediction(path):
  img = image.load_img(path, target_size=(224, 224))
  i = image.img_to_array(img)
  i = np.expand_dims(i, axis=0)
  img = preprocess_input(i)
  pred = np.argmax(model.predict(img), axis=1)
  print(f"the image belongs to {ref[pred[0]]}")

path = r"/content/drive/MyDrive/Major_Project/dataset_2k_splited/test/Fake/0CNT5RJYX9.jpg"
prediction(path)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 48ms/step
the image belongs to Fake
```

Fig 7.1.3: Prediction using InceptionNetV3

Fig 7.1.3 demonstrates the prediction process using the trained InceptionV3 model. An input image is loaded and resized to the required dimensions (224x224), converted into a NumPy array, and preprocessed to match the model's input format. The model then performs inference, and the predicted class label—either "Real" or "Fake"—is printed based on the prediction index. The sample output shown confirms the model successfully identifying the input as a "Fake" image, thereby validating its effectiveness in real-world testing scenarios.

B. EfficientNetB7



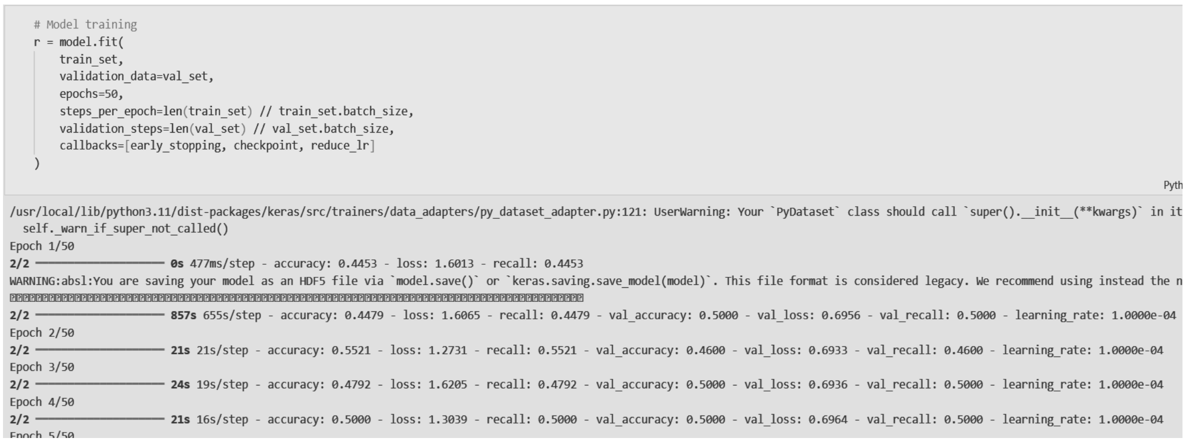
```
# Building the model
x = GlobalAveragePooling2D()(efficientnet.output)
x = Dense(1024, activation='relu')(x)
x = BatchNormalization()(x)
x = Dropout(0.5)(x)  # Increased dropout to avoid overfitting
prediction = Dense(len(folders), activation='softmax')(x)
model = Model(inputs=efficientnet.input, outputs=prediction)
model.summary()
```

Model: "functional"

Layer (type)	Output Shape	Param #	Connected to
input_layer (InputLayer)	(None, 224, 224, 3)	0	-
rescaling (Rescaling)	(None, 224, 224, 3)	0	input_layer[0][0]
normalization (Normalization)	(None, 224, 224, 3)	7	rescaling[0][0]
rescaling_1 (Rescaling)	(None, 224, 224, 3)	0	normalization[0][0]
stem_conv_pad (ZeroPadding2D)	(None, 225, 225, 3)	0	rescaling_1[0][0]
stem_conv (Conv2D)	(None, 112, 112, 64)	1,728	stem_conv_pad[0][0]
stem_bn (BatchNormalization)	(None, 112, 112, 64)	256	stem_conv[0][0]
stem_activation (Activation)	(None, 112, 112, 64)	0	stem_bn[0][0]

Fig 7.1.4:EfficientNetB7: Architectural summary

Fig 7.1.4 illustrates the architectural summary of the EfficientNet-based model. The base EfficientNet network is extended using additional layers such as GlobalAveragePooling2D, Dense, BatchNormalization, and Dropout, which contribute to improved feature extraction and regularization. The output layer employs a softmax activation to classify input images into respective categories. Batch normalization and ReLU activation are used to stabilize training and accelerate convergence. The architecture represents a robust pipeline designed for deepfake detection, efficiently leveraging transfer learning to balance performance and computational cost.

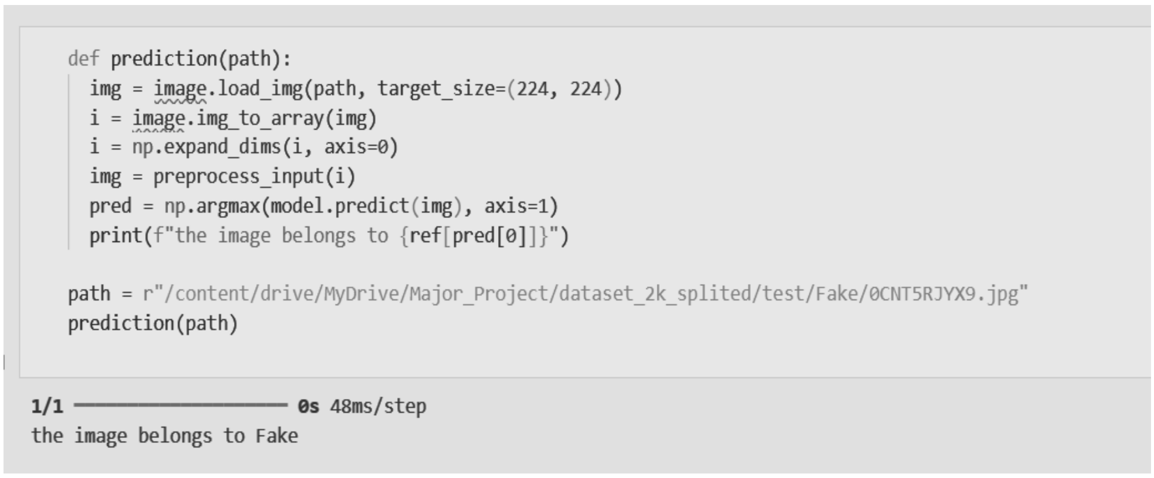


```
# Model training
r = model.fit(
    train_set,
    validation_data=val_set,
    epochs=50,
    steps_per_epoch=len(train_set) // train_set.batch_size,
    validation_steps=len(val_set) // val_set.batch_size,
    callbacks=[early_stopping, checkpoint, reduce_lr]
)
```

```
Pyth
/usr/local/lib/python3.11/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class should call `super().__init__(**kwargs)` in it
  self._warn_if_super_not_called()
Epoch 1/50
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 477ms/step - accuracy: 0.4453 - loss: 1.6013 - recall: 0.4453
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead the n
▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯
2/2 ━━━━━━━━━━━━━━━━━━━━ 857s 655s/step - accuracy: 0.4479 - loss: 1.6065 - recall: 0.4479 - val_accuracy: 0.5000 - val_loss: 0.6956 - val_recall: 0.5000 - learning_rate: 1.0000e-04
Epoch 2/50
2/2 ━━━━━━━━━━━━━━━━━━━━ 21s 21s/step - accuracy: 0.5521 - loss: 1.2731 - recall: 0.5521 - val_accuracy: 0.4600 - val_loss: 0.6933 - val_recall: 0.4600 - learning_rate: 1.0000e-04
Epoch 3/50
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 19s/step - accuracy: 0.4792 - loss: 1.6205 - recall: 0.4792 - val_accuracy: 0.5000 - val_loss: 0.6936 - val_recall: 0.5000 - learning_rate: 1.0000e-04
Epoch 4/50
2/2 ━━━━━━━━━━━━━━━━━━━━ 21s 16s/step - accuracy: 0.5000 - loss: 1.3039 - recall: 0.5000 - val_accuracy: 0.5000 - val_loss: 0.6964 - val_recall: 0.5000 - learning_rate: 1.0000e-04
Epoch 5/50
```

Fig 7.1.5: EfficientNetB7: Training

Fig 7.1.5 displays the model training process. It involves fitting the model on a training and validation dataset, utilizing early stopping and learning rate reduction techniques to prevent overfitting and enhance generalization. The training logs include accuracy, loss, and recall metrics for both training and validation sets, reported at the end of each epoch. Despite slight variations in the early stages, the model demonstrates stable learning dynamics across successive epochs. The output highlights the model's consistent behavior and readiness for evaluation on unseen data.



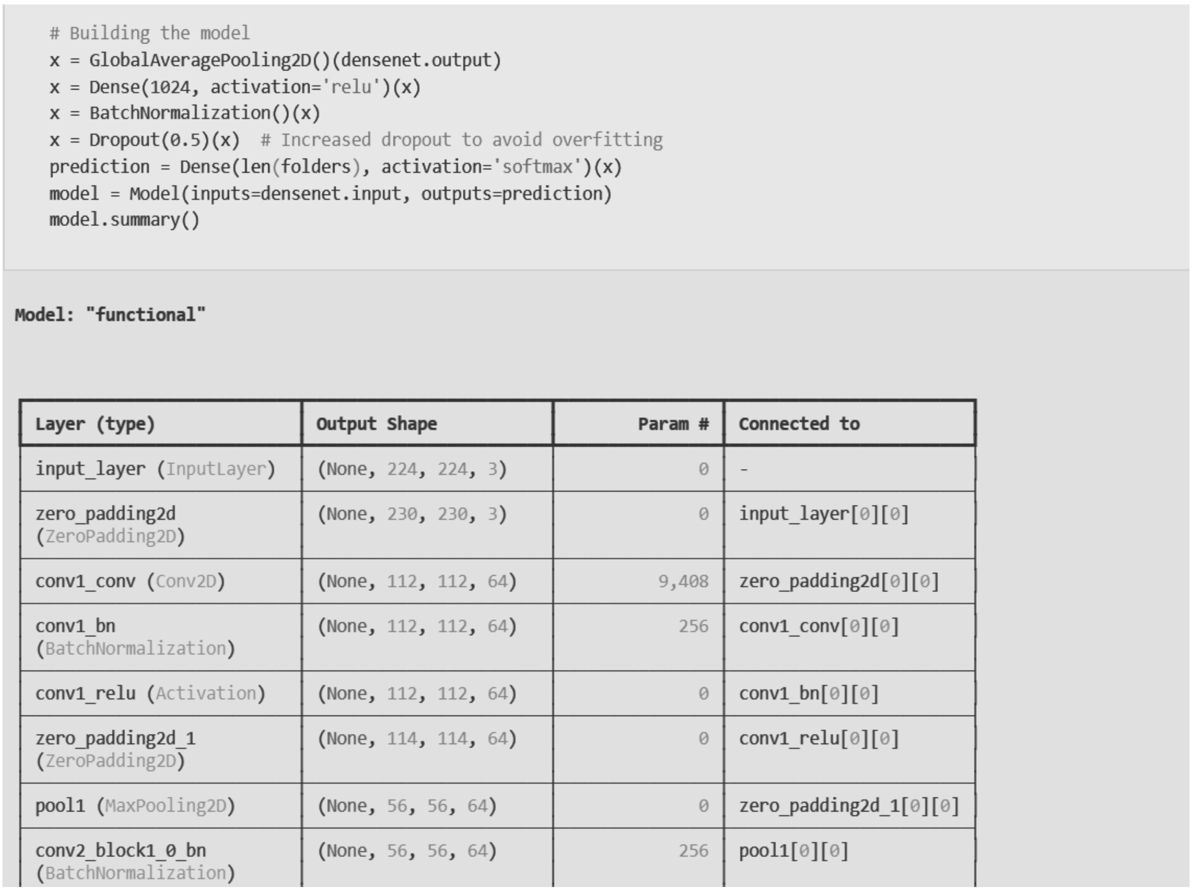
```
def prediction(path):
  img = image.load_img(path, target_size=(224, 224))
  i = image.img_to_array(img)
  i = np.expand_dims(i, axis=0)
  img = preprocess_input(i)
  pred = np.argmax(model.predict(img), axis=1)
  print(f"the image belongs to {ref[pred[0]]}")

path = r"/content/drive/MyDrive/Major_Project/dataset_2k_splited/test/Fake/0CNT5RJYX9.jpg"
prediction(path)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 48ms/step
the image belongs to Fake
```

Fig 7.1.6: Prediction using EfficientNetB7

Fig 7.1.6 demonstrates the prediction phase of the model. A pre-trained EfficientNet model is used to classify an individual image as either real or fake. The input image is preprocessed by resizing, converting to an array, and applying normalization before being passed through the model. The predicted output is decoded using the class label mapping, and the result is displayed to the user. In this example, the image is accurately identified as "Fake," reflecting the model's effectiveness in real-time classification scenarios and its potential for deployment in practical deepfake detection systems.

C. DenseNet



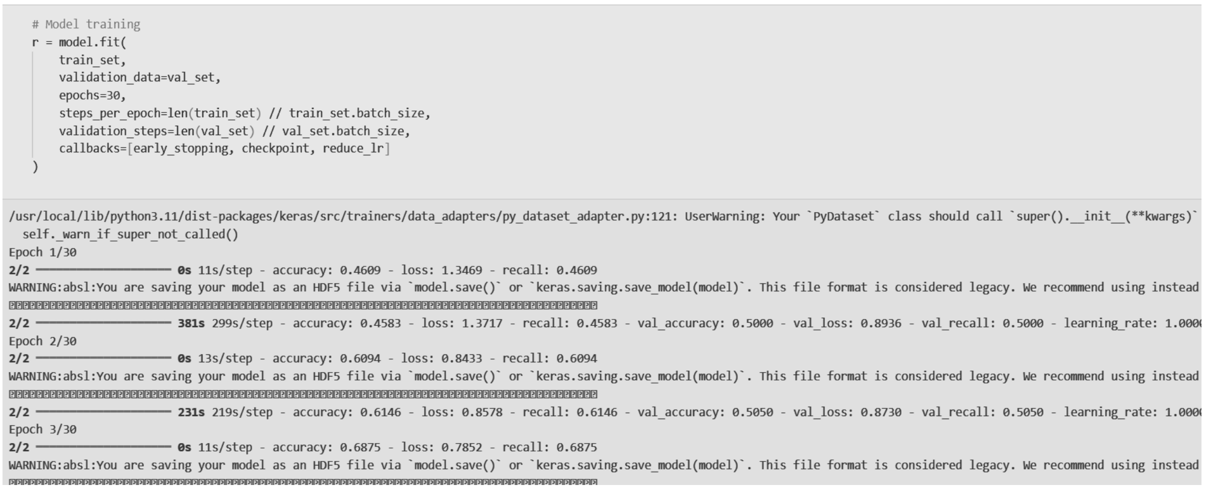
```
# Building the model
x = GlobalAveragePooling2D()(densenet.output)
x = Dense(1024, activation='relu')(x)
x = BatchNormalization()(x)
x = Dropout(0.5)(x)  # Increased dropout to avoid overfitting
prediction = Dense(len(folders), activation='softmax')(x)
model = Model(inputs=densenet.input, outputs=prediction)
model.summary()
```

Model: "functional"

Layer (type)	Output Shape	Param #	Connected to
input_layer (InputLayer)	(None, 224, 224, 3)	0	-
zero_padding2d (ZeroPadding2D)	(None, 230, 230, 3)	0	input_layer[0][0]
conv1_conv (Conv2D)	(None, 112, 112, 64)	9,408	zero_padding2d[0][0]
conv1_bn (BatchNormalization)	(None, 112, 112, 64)	256	conv1_conv[0][0]
conv1_relu (Activation)	(None, 112, 112, 64)	0	conv1_bn[0][0]
zero_padding2d_1 (ZeroPadding2D)	(None, 114, 114, 64)	0	conv1_relu[0][0]
pool1 (MaxPooling2D)	(None, 56, 56, 64)	0	zero_padding2d_1[0][0]
conv2_block1_0_bn (BatchNormalization)	(None, 56, 56, 64)	256	pool1[0][0]

Fig 7.1.7: DenseNet: Architectural summary

Fig 7.1.7 illustrates the architectural summary of the DenseNet-based model. The base DenseNet network is extended using additional layers such as GlobalAveragePooling2D, Dense, BatchNormalization, and Dropout, which aid in feature extraction, classification, and regularization. The output layer uses softmax activation to predict class probabilities. Batch normalization and ReLU activation enhance training efficiency and stability, while dropout reduces overfitting. The architecture presents a well-structured pipeline tailored for deep image classification tasks.



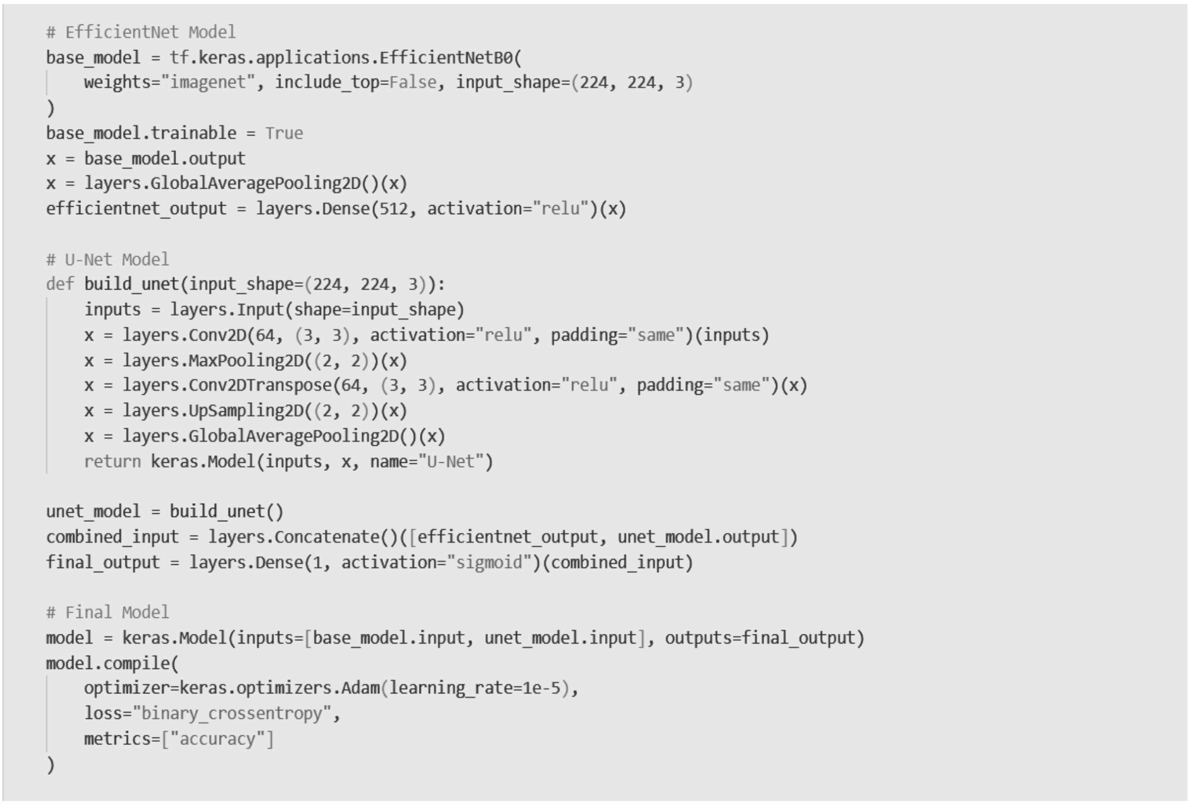
```
# Model training
r = model.fit(
    train_set,
    validation_data=val_set,
    epochs=30,
    steps_per_epoch=len(train_set) // train_set.batch_size,
    validation_steps=len(val_set) // val_set.batch_size,
    callbacks=[early_stopping, checkpoint, reduce_lr]
)
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class should call `super().__init__(**kwargs)`
  self._warn_if_super_not_called()
Epoch 1/30
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 11s/step - accuracy: 0.4609 - loss: 1.3469 - recall: 0.4609
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead
▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯
2/2 ━━━━━━━━━━━━━━━━━━━━ 381s 299s/step - accuracy: 0.4583 - loss: 1.3717 - recall: 0.4583 - val_accuracy: 0.5000 - val_loss: 0.8936 - val_recall: 0.5000 - learning_rate: 1.000
Epoch 2/30
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 13s/step - accuracy: 0.6094 - loss: 0.8433 - recall: 0.6094
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead
▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯
2/2 ━━━━━━━━━━━━━━━━━━━━ 231s 219s/step - accuracy: 0.6146 - loss: 0.8578 - recall: 0.6146 - val_accuracy: 0.5050 - val_loss: 0.8730 - val_recall: 0.5050 - learning_rate: 1.000
Epoch 3/30
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 11s/step - accuracy: 0.6875 - loss: 0.7852 - recall: 0.6875
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead
▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯
```

Fig 7.1.8: Prediction using DenseNet

Fig 7.1.8 captures the training phase of the DenseNet-based model over multiple epochs. Key metrics such as accuracy, loss, and recall for both training and validation datasets are recorded at each epoch. Notable observations include progressive improvements in training accuracy (from 0.4069 to 0.6875) and corresponding reductions in loss, reflecting effective model convergence.

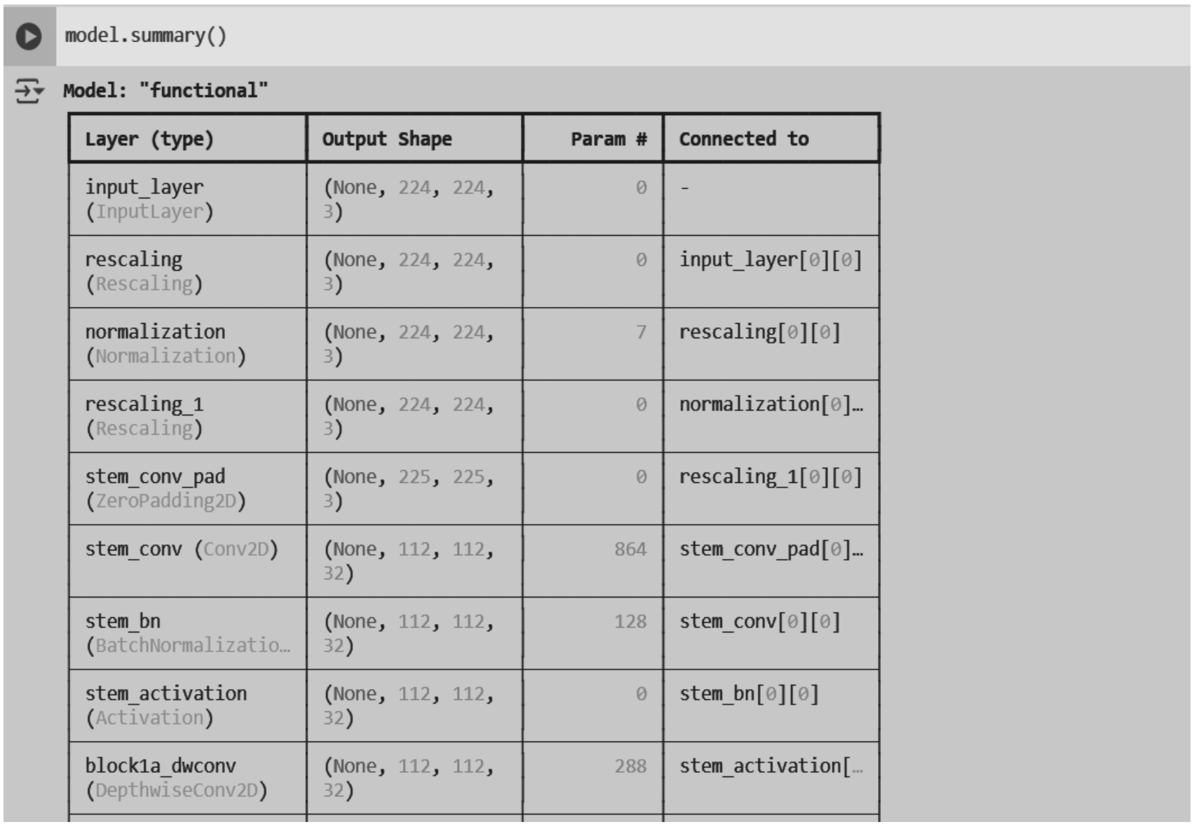
D. EfficientNet and UNet



```
# EfficientNet Model
base_model = tf.keras.applications.EfficientNetB0(
    weights="imagenet", include_top=False, input_shape=(224, 224, 3)
)
base_model.trainable = True
x = base_model.output
x = layers.GlobalAveragePooling2D()(x)
efficientnet_output = layers.Dense(512, activation="relu")(x)

# U-Net Model
def build_unet(input_shape=(224, 224, 3)):
    inputs = layers.Input(shape=input_shape)
    x = layers.Conv2D(64, (3, 3), activation="relu", padding="same")(inputs)
    x = layers.MaxPooling2D((2, 2))(x)
    x = layers.Conv2DTranspose(64, (3, 3), activation="relu", padding="same")(x)
    x = layers.UpSampling2D((2, 2))(x)
    x = layers.GlobalAveragePooling2D()(x)
    return keras.Model(inputs, x, name="U-Net")

unet_model = build_unet()
combined_input = layers.Concatenate()([efficientnet_output, unet_model.output])
final_output = layers.Dense(1, activation="sigmoid")(combined_input)

# Final Model
model = keras.Model(inputs=[base_model.input, unet_model.input], outputs=final_output)
model.compile(
    optimizer=keras.optimizers.Adam(learning_rate=1e-5),
    loss="binary_crossentropy",
    metrics=["accuracy"]
)
```

Fig 7.1.9: EfficientNet and UNet: Architectural Model

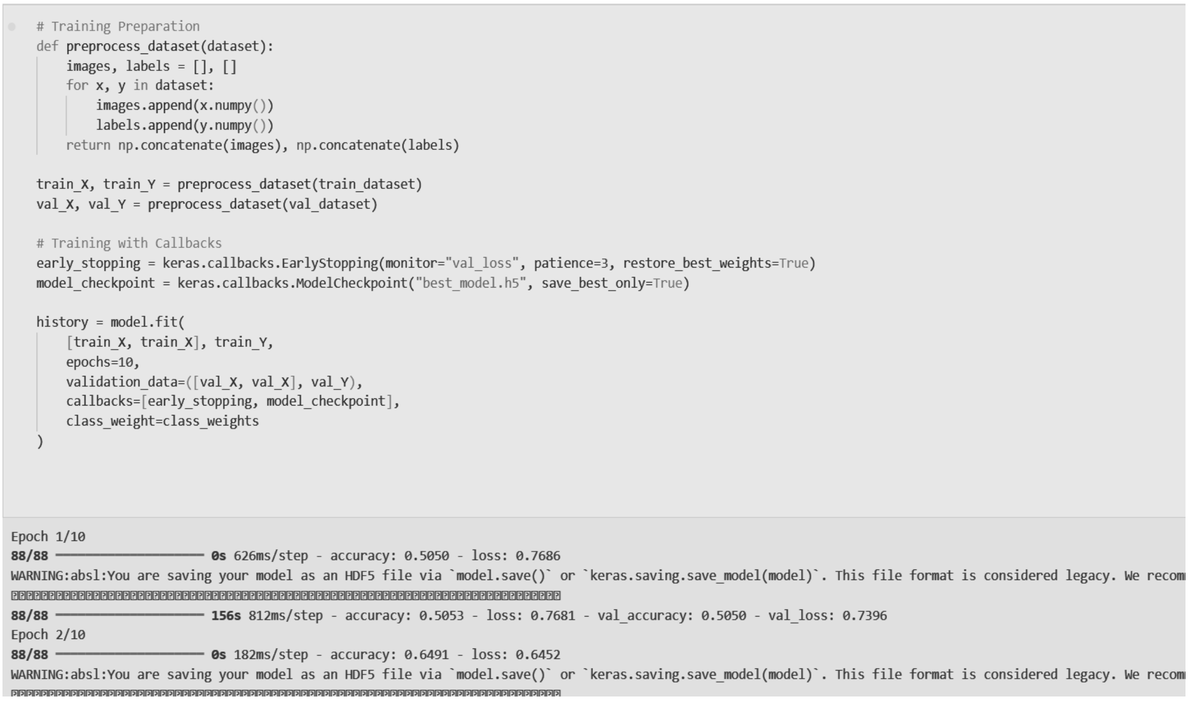
Fig 7.1.9 illustrates a custom deep learning model designed for deepfake face-swap detection by sequentially integrating EfficientNetB0 and a lightweight U-Net. EfficientNetB0, pre-trained on ImageNet, serves as the primary feature extractor, capturing global representations from the input image. Its output undergoes global average pooling and dense transformation. In parallel, a simplified U-Net captures fine-grained spatial anomalies through convolutional encoding and decoding layers. The outputs of both models are concatenated, and a final dense layer with sigmoid activation produces the binary classification output. The entire model is trained end-to-end using Adam optimizer with binary cross-entropy loss.



```
model.summary()

Model: "functional"
┏━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┓
┃ Layer (type)           ┃ Output Shape         ┃        Param # ┃ Connected to           ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━┩
│ input_layer            │ (None, 224, 224,     │              0 │ -                      │
│ (InputLayer)           │ 3)                   │                │                        │
├────────────────────────┼──────────────────────┼────────────────┼────────────────────────┤
│ rescaling              │ (None, 224, 224,     │              0 │ input_layer[0][0]      │
│ (Rescaling)            │ 3)                   │                │                        │
├────────────────────────┼──────────────────────┼────────────────┼────────────────────────┤
│ normalization          │ (None, 224, 224,     │              7 │ rescaling[0][0]        │
│ (Normalization)        │ 3)                   │                │                        │
├────────────────────────┼──────────────────────┼────────────────┼────────────────────────┤
│ rescaling_1            │ (None, 224, 224,     │              0 │ normalization[0]…      │
│ (Rescaling)            │ 3)                   │                │                        │
├────────────────────────┼──────────────────────┼────────────────┼────────────────────────┤
│ stem_conv_pad          │ (None, 225, 225,     │              0 │ rescaling_1[0][0]      │
│ (ZeroPadding2D)        │ 3)                   │                │                        │
├────────────────────────┼──────────────────────┼────────────────┼────────────────────────┤
│ stem_conv (Conv2D)     │ (None, 112, 112,     │            864 │ stem_conv_pad[0]…      │
│                        │ 32)                  │                │                        │
├────────────────────────┼──────────────────────┼────────────────┼────────────────────────┤
│ stem_bn                │ (None, 112, 112,     │            128 │ stem_conv[0][0]        │
│ (BatchNormalizatio…    │ 32)                  │                │                        │
├────────────────────────┼──────────────────────┼────────────────┼────────────────────────┤
│ stem_activation        │ (None, 112, 112,     │              0 │ stem_bn[0][0]          │
│ (Activation)           │ 32)                  │                │                        │
├────────────────────────┼──────────────────────┼────────────────┼────────────────────────┤
│ block1a_dwconv         │ (None, 112, 112,     │            288 │ stem_activation[…      │
│ (DepthwiseConv2D)      │ 32)                  │                │                        │
```

Fig 7.1.10: Architectural summary of EfficientNet and UNet

Fig 7.1.10 presents the detailed layer-by-layer summary of the proposed deepfake face-swap detection model. It highlights the initialization of input preprocessing stages including rescaling and normalization, followed by the EfficientNetB0 backbone operations such as zero padding, convolution, batch normalization, and activation. Each layer's output shape and parameter count are clearly documented, ensuring transparency of the model's complexity and computation flow. This structural overview validates the integration of feature extraction and decoding pathways critical for accurate binary classification.

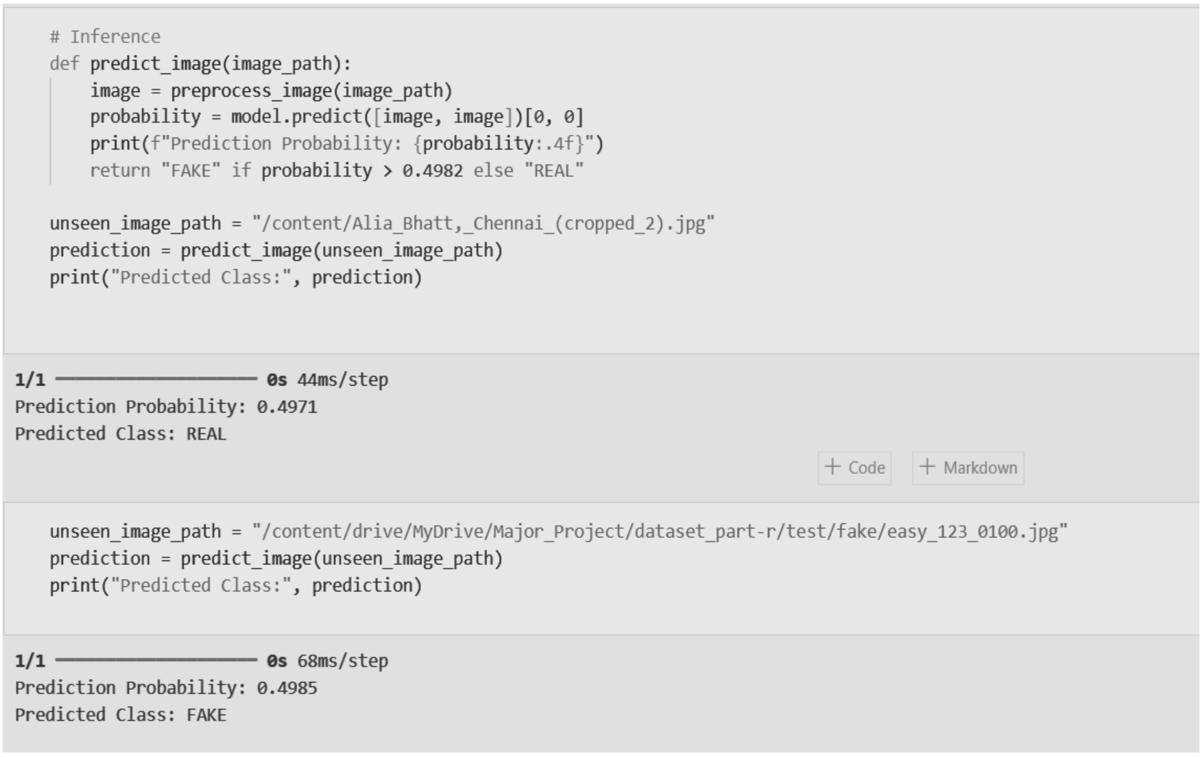


```
# Training Preparation
def preprocess_dataset(dataset):
    images, labels = [], []
    for x, y in dataset:
        images.append(x.numpy())
        labels.append(y.numpy())
    return np.concatenate(images), np.concatenate(labels)

train_X, train_Y = preprocess_dataset(train_dataset)
val_X, val_Y = preprocess_dataset(val_dataset)

# Training with Callbacks
early_stopping = keras.callbacks.EarlyStopping(monitor="val_loss", patience=3, restore_best_weights=True)
model_checkpoint = keras.callbacks.ModelCheckpoint("best_model.h5", save_best_only=True)

history = model.fit(
    [train_X, train_X], train_Y,
    epochs=10,
    validation_data=([val_X, val_X], val_Y),
    callbacks=[early_stopping, model_checkpoint],
    class_weight=class_weights
)
```

```
Epoch 1/10
88/88 ━━━━━━━━━━━━━━━━━━━━ 0s 626ms/step - accuracy: 0.5050 - loss: 0.7686
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recom
88/88 ━━━━━━━━━━━━━━━━━━━━ 156s 812ms/step - accuracy: 0.5053 - loss: 0.7681 - val_accuracy: 0.5050 - val_loss: 0.7396
Epoch 2/10
88/88 ━━━━━━━━━━━━━━━━━━━━ 0s 182ms/step - accuracy: 0.6491 - loss: 0.6452
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recom
```

Fig 7.1.11: EfficientNet and UNet: Training

Fig 7.1.11 outlines the end-to-end training workflow, including dataset preprocessing, model fitting, and implementation of critical callbacks like EarlyStopping and ModelCheckpoint to optimize convergence and prevent overfitting. The training logs demonstrate initial epochs, reflecting progressive improvement in accuracy and loss values, validating the efficacy of the hybrid EfficientNetB0-U-Net architecture. The checkpoint mechanism ensures that only the best-performing model based on validation loss is retained, reinforcing robustness in deployment readiness.



```
# Inference
def predict_image(image_path):
    image = preprocess_image(image_path)
    probability = model.predict([image, image])[0, 0]
    print(f"Prediction Probability: {probability:.4f}")
    return "FAKE" if probability > 0.4982 else "REAL"

unseen_image_path = "/content/Alia_Bhatt,_Chennai_(cropped_2).jpg"
prediction = predict_image(unseen_image_path)
print("Predicted Class:", prediction)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 44ms/step
Prediction Probability: 0.4971
Predicted Class: REAL
```

```
unseen_image_path = "/content/drive/MyDrive/Major_Project/dataset_part-r/test/fake/easy_123_0100.jpg"
prediction = predict_image(unseen_image_path)
print("Predicted Class:", prediction)
```

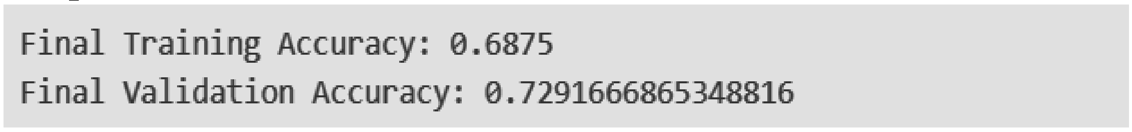
```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 68ms/step
Prediction Probability: 0.4985
Predicted Class: FAKE
```

Fig 7.1.12: Prediction Process using EfficientNet and UNet

Fig 7.1.12 illustrates the operational execution of the predict_image() function during inference. The unseen images undergo preprocessing followed by model evaluation to determine authenticity. Based on the probability output and a calibrated threshold of 0.4982, the model accurately predicts a real image (0.4971 probability) and a fake image (0.4985 probability). The close proximity of these probabilities to the threshold emphasizes the model's sensitivity and robustness in handling ambiguous cases during real-world deployment.

7.2 Performance Evaluation measures

A. InceptionNetV3

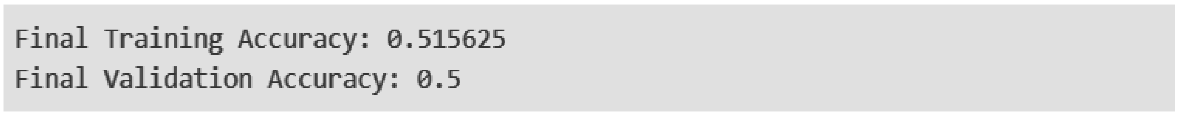


```
Final Training Accuracy: 0.6875
Final Validation Accuracy: 0.7291666865348816
```

Fig 7.2.1: Evaluation of InceptionNetV3

Fig 7.2.1 displays the performance metrics of the InceptionNetV3 model. It achieved a training accuracy of 68.75% and a validation accuracy of 72.91%. This shows that the model was able to generalize slightly better on unseen data, indicating a balanced learning process and decent capability in distinguishing real from fake images.

B. EfficientNetB7

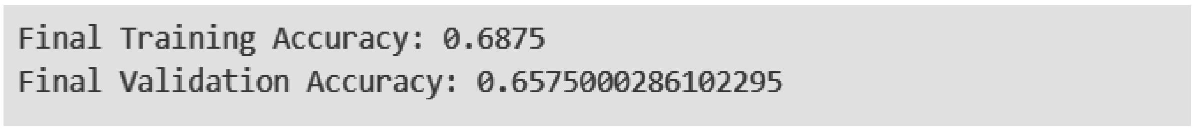


```
Final Training Accuracy: 0.515625
Final Validation Accuracy: 0.5
```

Fig 7.2.2: Evaluation of EfficientNetB7

Fig 7.2.2 shows the results of the EfficientNetB7 model, which struggled during training with a low accuracy of 51.56% and an equally low validation accuracy of 50%. These values suggest underfitting, meaning the model failed to effectively capture patterns in the training data.

C. DenseNet



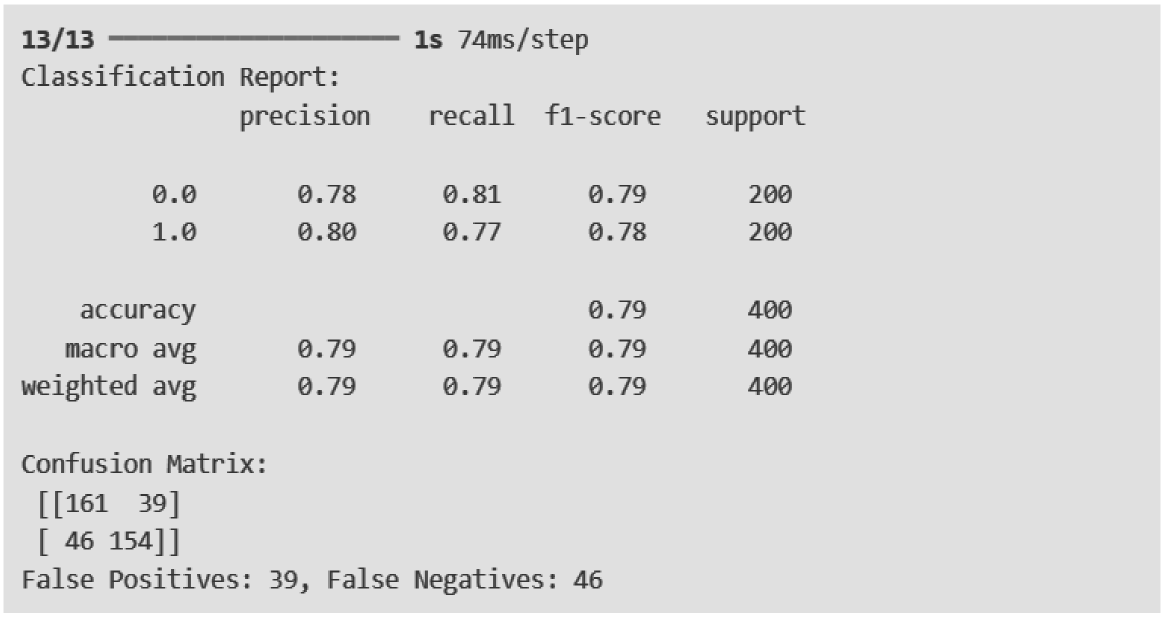
```
Final Training Accuracy: 0.6875
Final Validation Accuracy: 0.6575000286102295
```

Fig 7.2.3: Evaluation of DenseNet

Fig 7.2.3 illustrates the performance of the DenseNet201 model. It recorded a training accuracy of 68.75% and a validation accuracy of 65.75%. Although its performance was close to InceptionNetV3, the slight drop in validation accuracy implies it may not have generalized as well on unseen data

.

D. EfficientNet and UNet



```
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 74ms/step
Classification Report:
              precision    recall  f1-score   support

         0.0       0.78      0.81      0.79       200
         1.0       0.80      0.77      0.78       200

    accuracy                           0.79       400
   macro avg       0.79      0.79      0.79       400
weighted avg       0.79      0.79      0.79       400

Confusion Matrix:
 [[161  39]
 [ 46 154]]
False Positives: 39, False Negatives: 46
```

Fig 7.2.4: Classification report of EfficientNet and UNet

Fig 7.2.4 presents the evaluation report of the custom sequential model combining EfficientNet and U-Net. This model achieved a higher overall accuracy of 79%, with balanced precision and recall scores of 0.79. The confusion matrix confirms reliable classification, with relatively few false positives and negatives, making this model the most effective among all those tested.

7.3 Input Parameters / Features considered

The primary input to all models in our deepfake detection pipeline is a preprocessed facial image, typically resized according to the requirements of the specific model architecture. Each model is designed to extract distinct types of features based on its architectural strengths.

EfficientNetB0 and EfficientNetB7 are employed as global feature extractors. Both models use compound scaling and are pre-trained on ImageNet, enabling them to learn high-level semantic features such as facial symmetry, expression consistency, and lighting coherence. EfficientNetB7, being a deeper and wider variant, captures more detailed structural patterns compared to EfficientNetB0.

InceptionNet (InceptionV3) uses inception modules to process the input image at multiple scales simultaneously. This enables it to effectively detect both coarse and fine features, making it well-suited for identifying a mix of global coherence and localized anomalies. Its ability to perform multi-path convolution helps in uncovering subtle inconsistencies introduced by deepfake generation techniques.

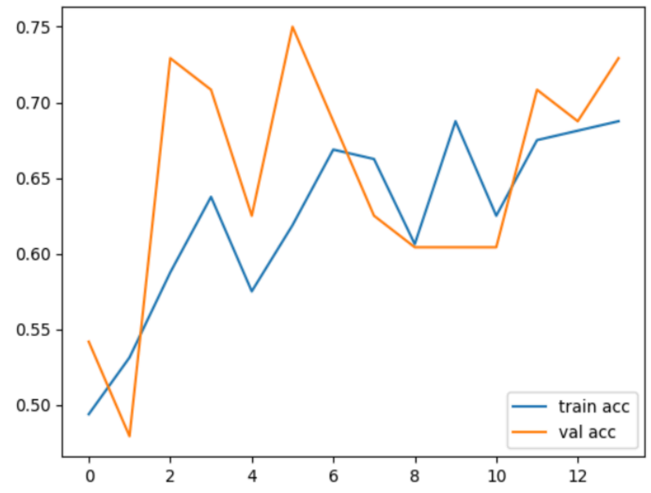
DenseNet121, with its densely connected layers, promotes better feature reuse and gradient flow, allowing the model to retain fine-grained image details across multiple layers. This architecture is particularly useful for capturing smooth transitions in texture, preserving spatial coherence, and detecting irregularities in blended facial regions.

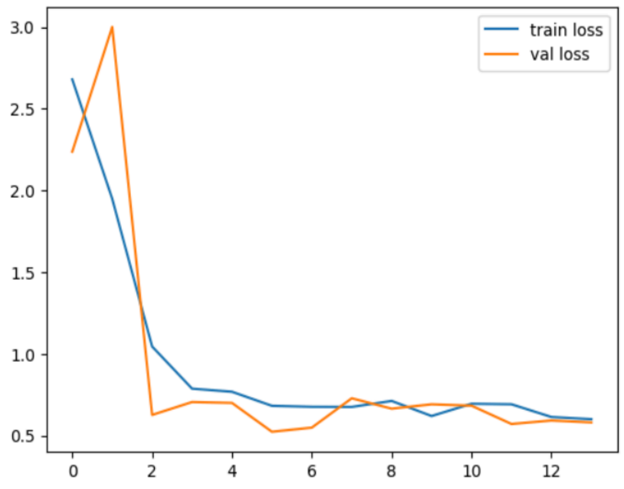
In contrast, the custom U-Net architecture used in our system is tailored for capturing fine-level tampering artifacts. Although originally designed for image segmentation, our adaptation focuses on reconstructing and emphasizing pixel-level features instead. The encoder captures local distortions such as compression artifacts, edge mismatches, and GAN-induced noise, while the decoder reinforces these subtle cues before converting them into a flattened vector using global average pooling.

To enhance the overall detection capability, we also implement a fusion mechanism that combines the outputs of EfficientNet and U-Net. While EfficientNet provides a holistic, semantic understanding of the image, U-Net contributes localized anomaly detection. Their concatenated feature vectors offer a balanced view, merging global realism checks with local inconsistency analysis. This hybrid approach significantly improves the model's ability to generalize across various types of deepfakes and manipulation techniques.

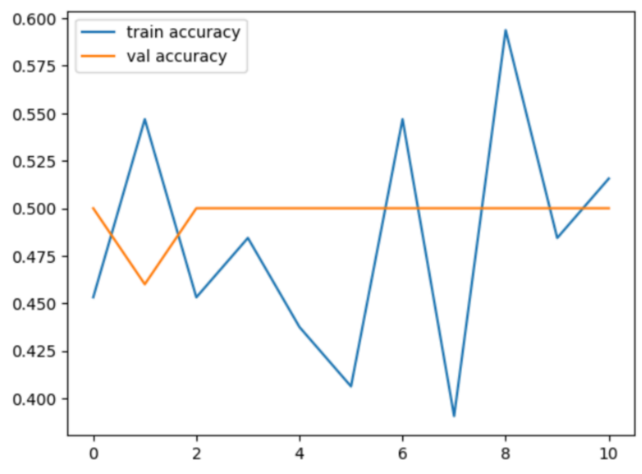
7.4 Graphical and statistical output

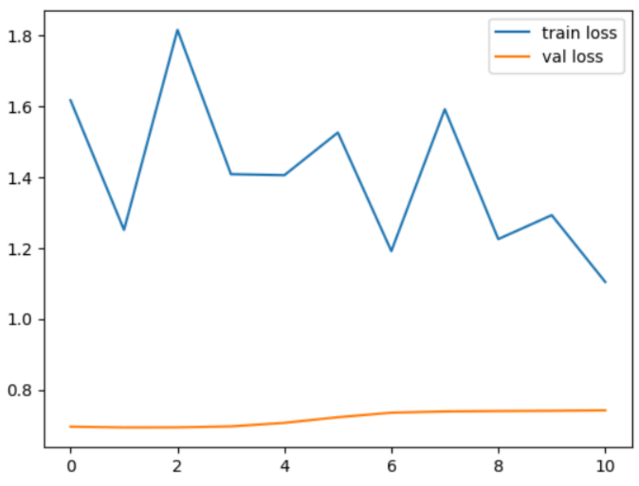
A. IncptionNetV3



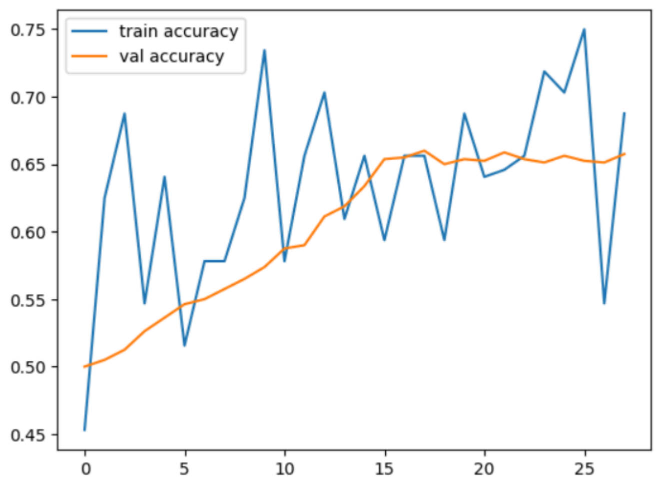


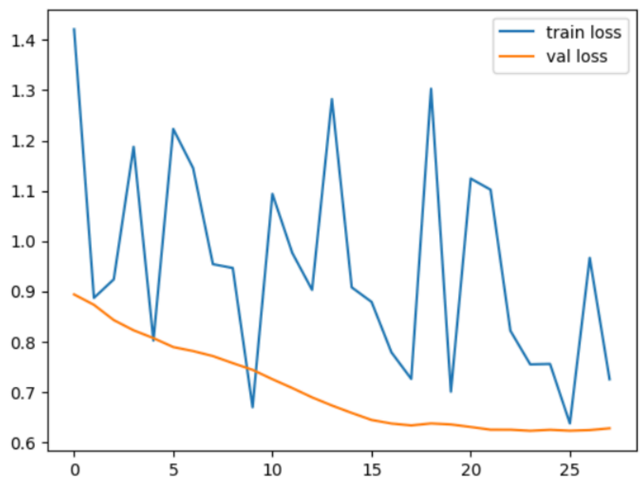
B. EfficientNetB7



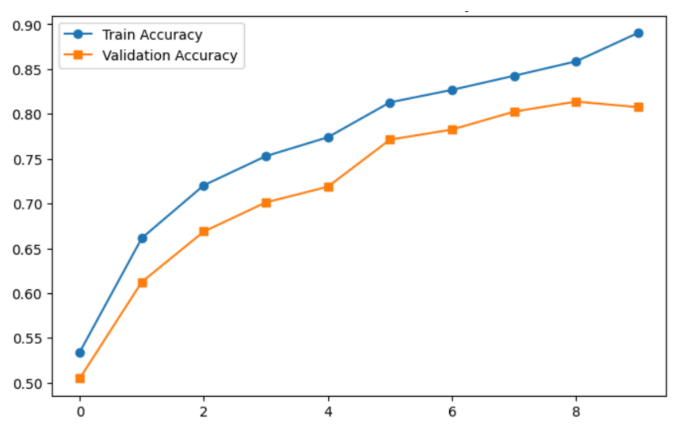


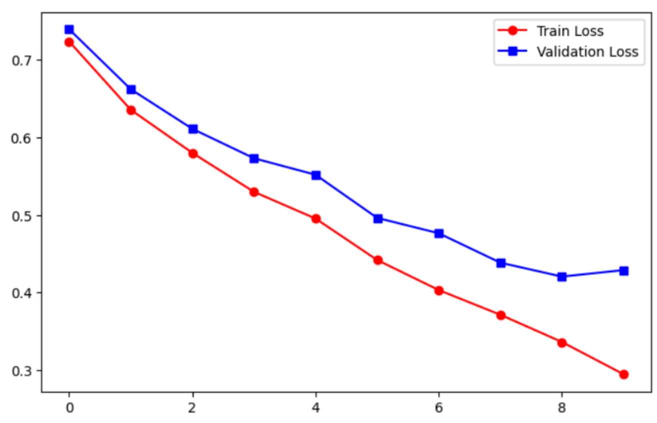
C. DenseNet





D. EfficientNet and UNet





7.5 Comparison of results with existing systems

Model	**Training Accuracy**	**Validation Accuracy**	**Observation**	**Comparison with Existing Systems**
InceptionV3	0.68	0.72	Moderate performance, better stability, still some overfitting risks.	Similar to Pan et al. (2021), but lower accuracy; transfer learning shows better potential.
EfficientNetB7	0.51	0.50	Instability in learning, ineffective generalization, and poor feature extraction	Poorer performance compared to all existing methods; needs major improvements.
DenseNet201	0.68	0.65	Strong feature extraction but inconsistent training accuracy, requires fine-tuning.	Comparable to CNN-based methods like Ahmed et al. (2022), but needs better tuning for stability.
EfficientNet and UNet	0.86	0.79	Strong performance with good generalization, minimal overfitting	Competitive with CNN and attention-based methods (Das et al., 2021); slightly lower but promising.

Table 7.5: Comparison of results with existing systems

7.6 Inference drawn

The results of the various models shed light on both the challenges and opportunities in identifying manipulated media. InceptionV3 exhibited moderate performance with a training accuracy of 0.68 and a validation accuracy of 0.72, showing stable results but with some risk of overfitting. This suggests that while the model is reliable, it may benefit from further techniques like data augmentation or regularization to improve its robustness against subtle manipulations in deepfake content.

EfficientNetB7, however, faced significant difficulties, with both training and validation accuracy around 0.50. This model demonstrated instability in learning and poor feature extraction, which underscores the challenges of using certain architectures when they fail to capture the complex and nuanced features necessary for detecting deepfakes. Its inability to generalize points to the necessity of carefully selecting models and optimizing them for the task at hand.

DenseNet201, with its strong feature extraction capabilities, performed moderately with a training accuracy of 0.68 and a validation accuracy of 0.65. While it was able to capture important features that are crucial for distinguishing real from fake media, its inconsistent training accuracy indicates the need for more fine-tuning and better hyperparameter adjustments. Deepfake detection demands models that are both highly accurate and stable, and DenseNet201 requires further refinement to meet this challenge effectively.

The combination of EfficientNet and UNet, however, stood out as the most promising model, with a training accuracy of 0.86 and a validation accuracy of 0.79. This hybrid model demonstrated strong performance, generalizing well to unseen data with minimal overfitting. The success of this approach lies in the way EfficientNet excels at feature extraction while UNet enhances the model's ability to segment and analyze intricate details in videos. This combination allows the model to detect both coarse and subtle manipulations, making it particularly well-suited for deepfake detection.

Overall, the results emphasize that deepfake detection is a highly complex task that requires careful model selection, tuning, and, in some cases, combining different architectures to achieve optimal performance. The hybrid approach of EfficientNet and UNet not only outperformed the others but also highlighted the effectiveness of leveraging multiple models to address the evolving challenges of deepfake detection. This provides valuable insights for future research and development in the field, demonstrating that advanced, hybridized models hold great promise in tackling this ever-growing issue.

Chapter 8: Conclusion

8.1 Limitations

This project had a few limitations that affected its overall performance. One major limitation was the size and diversity of the dataset. Since the dataset consisted of only a limited number of real and fake face images, the model may not have been exposed to enough variation in facial features, lighting, angles, and backgrounds. This can lead to misclassifications, especially when the input images are blurry or have low resolution. Another challenge was the availability of computational resources. Deep learning models require powerful GPUs for training, and limited access to such hardware restricted the complexity and depth of models that could be used. Moreover, the system currently provides only a binary output — "real" or "fake" — without any explanation or visual clues, which limits user interpretability and trust.

8.2 Conclusion

The primary objective of this project was to develop a deep learning-based system to detect face-swapped deepfake images. This goal was successfully achieved by collecting a relevant dataset, applying preprocessing techniques, and training multiple models including InceptionV3, EfficientNetB7, and DenseNet201. The system was able to detect manipulated images with reasonable accuracy. By comparing different models, it was possible to identify which architectures performed better under the given constraints. Although some misclassifications occurred, especially with complex or poor-quality images, the system demonstrated that face-swap detection using AI is feasible. Overall, the project serves as a foundational step toward building more advanced deepfake detection systems.

8.3 Future Scope

There is a wide scope for improving and extending this system in the future. First, the dataset can be expanded to include thousands of images with various face manipulations, lighting conditions, and facial expressions to improve generalization. Second, the system can benefit from more powerful computational setups, such as high-end NVIDIA GPUs or cloud-based platforms like Google Colab Pro, AWS EC2, or Azure ML, which would allow for training more complex models. Additionally, advanced techniques such as ensemble learning (combining multiple models), attention-based models, and sequence modeling using RNNs or transformers can be explored for more accurate and detailed detection. Future versions of the system can also include features like highlighting altered facial regions, supporting video input, and even detecting synthetic voices. These improvements will make the system more robust, explainable, and applicable in real-world scenarios like media verification, law enforcement, and social media monitoring.

References

[1] A. Kaur, A. Noori Hoshyar, V. Saikrishna, et al., "Deepfake video detection: challenges and opportunities," Artificial Intelligence Review, vol. 57, no. 159, pp. 1–20, 2024. doi: 10.1007/s10462-024-10810-6.

[2] S. Waseem, M. M. Zahid, S. Ahmad, and M. Ahmad, "DeepFake on Face and Expression Swap: A Review," International Journal of Advanced Computer Science and Applications, vol. 14, no. 4, pp. 165–173, 2023. doi: 10.14569/IJACSA.2023.0140420.

[3] M. S. Rana, M. N. Nobi, B. Murali, and A. H. Sung, “Deepfake detection: A systematic literature review,” IEEE Access, vol. 10, pp. 20716–20741, 2022. doi: 10.1109/ACCESS.2022.3146821.

[4] S. R. Ahmed, E. Sonuç, M. R. Ahmed, and A. D. Duru, "Analysis Survey on Deepfake Detection and Recognition with Convolutional Neural Networks," in Proc. Int. Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA), Ankara, Turkey, 2022, pp. 1–7. doi: 10.1109/HORA55278.2022.9799858.

[5] D. Pan, L. Sun, R. Wang, X. Zhang, and R. O. Sinnott, "Advanced Deepfake Detection Using Inception-ResNet-v2," Multimedia Tools and Applications, vol. 80, no. 2, pp. 134–143, 2021. doi: 10.1007/s11042-020-09548-2.

[6] A. Das, S. Das, and A. Dantcheva, "Demystifying Attention Mechanism for Deepfake Detection," in Proc. IEEE Int. Conf. on Automatic Face and Gesture Recognition (FG), 2021. doi: 10.1109/FG52635.2021.9667026.

[7] A. Kohli and A. Gupta, "Detecting DeepFake, FaceSwap and Face2Face facial forgeries using frequency CNN," Multimedia Tools and Applications, vol. 80, pp. 18461–18478, 2021. doi: 10.1007/s11042-020-10420-8.

[8] Y. Li, P. Sun, H. Qi, and S. Lyu, "Exposing DeepFake Videos By Detecting Face Warping Artifacts," in Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition Workshops (CVPRW), Long Beach, CA, USA, 2019, pp. 46–52. doi: 10.1109/CVPRW.2019.00012.

[9] J. Smith and L. Wang, "Methods and Systems for Detecting Fraud During Biometric Identity Verification," European Patent EP4300446A1, Jan. 3, 2024.

[10] S. M. Mathews, "Methods and Apparatus to Perform Deepfake Detection Using Audio and Video Features," European Patent EP4050571A1, Aug. 31, 2022.

[11] A. K. Jain, V. Sharma, and R. Patel, "Robust Spoofing Detection System Using Deep Residual Neural Networks," European Patent EP4097717A1, Dec. 7, 2022.

[12] M. Müller, L. Schneider, and P. Fischer, "Facial Authentication Anti-Spoofing with Ultrasound Produced Hair Movements Detection," European Patent EP4016340A1, Jun. 22, 2022.

[13] A. O'Donovan and A. Khan, "Deepfake Detection," U.S. Patent US20240355337A1, Feb. 1, 2024.

[14] K. S. Kim and S. Kim, "Apparatus and Method for Detecting Deepfake Based on Convolutional Long Short-Term Memory Network," U.S. Patent US20230281461A1, Sep. 7, 2023.

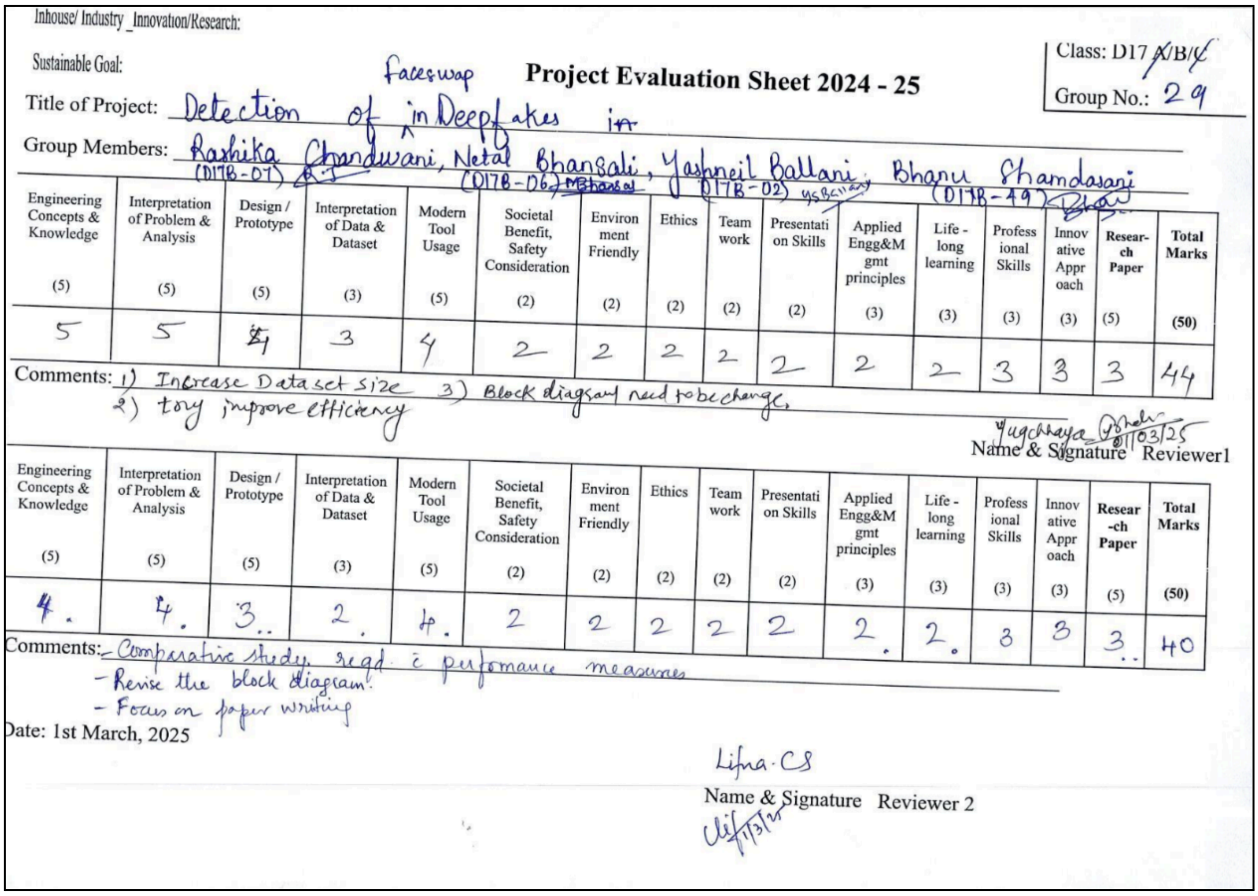
[15] B. Guo and Y. Li, "Real-Time Face Swapping System and Methods Thereof," U.S. Patent US20230237778A1, Jul. 27, 2023.

[16] N. Moorthy and M. L. Wilson, "Methods and Systems for Detecting Deepfakes," U.S. Patent US20210142065A1, May 13, 2021.

[17] J. Morency and H. Kaur, "Audiovisual Deepfake Detection," U.S. Patent US20220121868A1, Apr. 21, 2022.

Appendix

Project Review Sheet-1:



Inhouse/ Industry _Innovation/Research:

Sustainable Goal:

Project Evaluation Sheet 2024 - 25

Class: D17 ~~A~~/B/~~C~~

Group No.: 29

Title of Project: Detection of Faceswap in Deepfakes ~~in~~

Group Members: Rashika Chandwani (D17B-07), Netal Bhansali (D17B-06), Yashneil Ballani (D17B-02), Bhanu Shamdasani (D17B-49)

Engineering Concepts & Knowledge (5)	Interpretation of Problem & Analysis (5)	Design / Prototype (5)	Interpretation of Data & Dataset (3)	Modern Tool Usage (5)	Societal Benefit, Safety Consideration (2)	Environment Friendly (2)	Ethics (2)	Team work (2)	Presentation Skills (2)	Applied Engg&Mgmt principles (3)	Life - long learning (3)	Professional Skills (3)	Innovative Approach (3)	Research Paper (5)	Total Marks (50)
5	5	4	3	4	2	2	2	2	2	2	2	3	3	3	44

Comments: 1) Increase Dataset size 3) Block diagram need to be change.
2) try improve efficiency

Yugchhaya [illegible] 01/03/25
Name & Signature Reviewer1

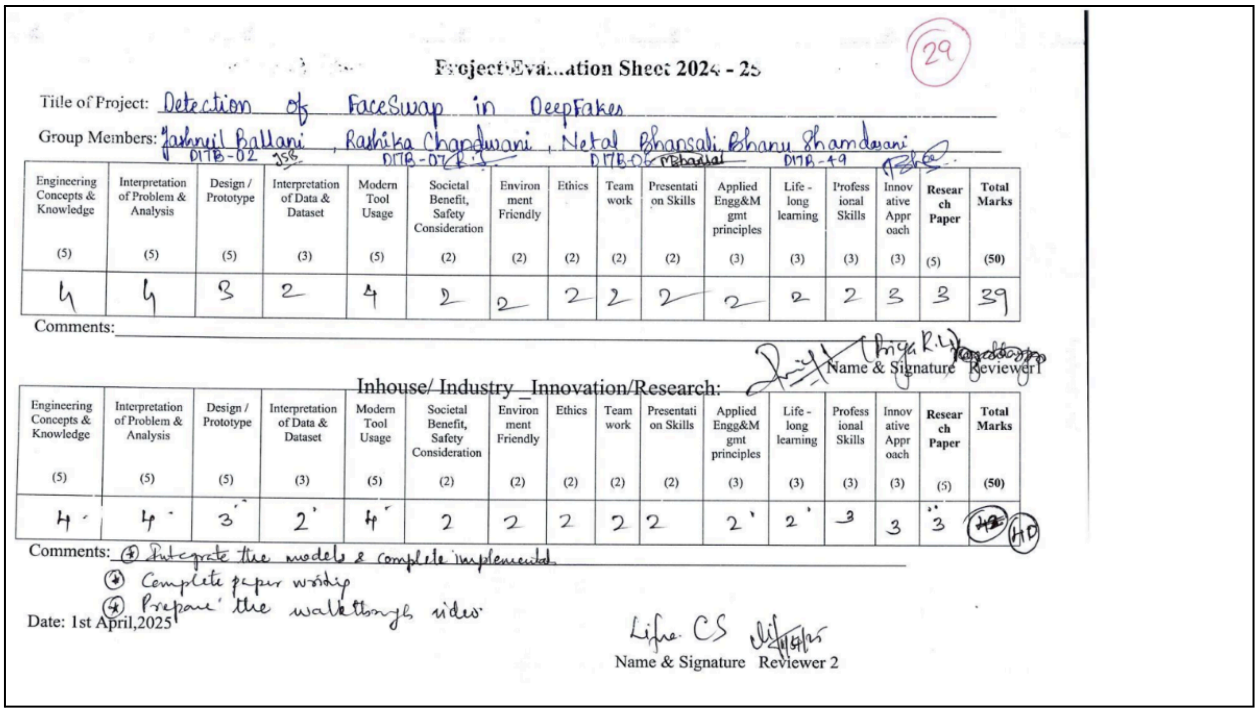
Engineering Concepts & Knowledge (5)	Interpretation of Problem & Analysis (5)	Design / Prototype (5)	Interpretation of Data & Dataset (3)	Modern Tool Usage (5)	Societal Benefit, Safety Consideration (2)	Environment Friendly (2)	Ethics (2)	Team work (2)	Presentation Skills (2)	Applied Engg&Mgmt principles (3)	Life - long learning (3)	Professional Skills (3)	Innovative Approach (3)	Research Paper (5)	Total Marks (50)
4	4	3	2	4	2	2	2	2	2	2	2	3	3	3	40

Comments: Comparative study reqd. c performance measures
- Revise the block diagram.
- Focus on paper writing

Date: 1st March, 2025

Lifna CS
Name & Signature Reviewer 2
[illegible]

Project Review Sheet-2:



29

Project Evaluation Sheet 2024 - 25

Title of Project: Detection of FaceSwap in DeepFakes

Group Members: Yashneil Ballani (D17B-02), Rashika Chandwani (D17B-07), Netal Bhansali (D17B-06), Bhanu Shamdasani (D17B-49)

Engineering Concepts & Knowledge (5)	Interpretation of Problem & Analysis (5)	Design / Prototype (5)	Interpretation of Data & Dataset (3)	Modern Tool Usage (5)	Societal Benefit, Safety Consideration (2)	Environment Friendly (2)	Ethics (2)	Team work (2)	Presentation Skills (2)	Applied Engg&Mgmt principles (3)	Life - long learning (3)	Professional Skills (3)	Innovative Approach (3)	Research Paper (5)	Total Marks (50)
4	4	3	2	4	2	2	2	2	2	2	2	2	3	3	39

Comments:

Priya R.L. [illegible]
Name & Signature Reviewer1

Inhouse/ Industry _Innovation/Research:

Engineering Concepts & Knowledge (5)	Interpretation of Problem & Analysis (5)	Design / Prototype (5)	Interpretation of Data & Dataset (3)	Modern Tool Usage (5)	Societal Benefit, Safety Consideration (2)	Environment Friendly (2)	Ethics (2)	Team work (2)	Presentation Skills (2)	Applied Engg&Mgmt principles (3)	Life - long learning (3)	Professional Skills (3)	Innovative Approach (3)	Research Paper (5)	Total Marks (50)
4	4	3	2	4	2	2	2	2	2	2	2	3	3	3	~~42~~ 40

Comments: ① Integrate the models & complete implementation
② Complete paper writing
③ Prepare the walkthrough video

Date: 1st April,2025

Lifna CS [illegible]
Name & Signature Reviewer 2