VIVEKANAND EDUCATION SOCIETY'S INSTITUTE OF TECHNOLOGY

(An Autonomous Institute Affiliated to University of Mumbai)

Department of Computer Engineering

V.E.S.

Since 1962

Project Report on

Detection of Face Swap in Deepfakes

Submitted in partial fulfillment of the requirements of the degree

BACHELOR OF ENGINEERING IN COMPUTER ENGINEERING

By

Yashneil Ballani D17B/02

Netal Bhansali D17B/06

Rashika Chandwani D17B/07

Bhanu Shamdasani D17B/49

Project Mentor

Prof. Priya R.L

University of Mumbai

(AY 2024 - 25)

VIVEKANAND EDUCATION SOCIETY'S INSTITUTE OF TECHNOLOGY

(An Autonomous Institute Affiliated to University of Mumbai)

Department of Computer Engineering

V.E.S.
Since 1962

Certificate

This is to certify that **Yashneil Ballani (D17B - 02), Netal Bhansali (D17B - 06), Rashika Chandwani (D17B - 07) and Bhanu Shamdasani(D17B - 49)** of Fourth Year Computer Engineering studying under the University of Mumbai has satisfactorily presented the project on " **Detection of Face-Swap based Deep Fake Videos**" as a part of the coursework of PROJECT-I for Semester-VII under the guidance of ***Mrs. Priya R L*** in the year 2024-2025.

Date

Internal Examiner

External Examiner

Project Mentor

Head of the Department
Dr. Mrs. Nupur Giri

Principal
Dr. J. M. Nair

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** (Project Guide) 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
CO 1	Do literature survey/industrial visit and identify the problem of the selected project topic.
CO2	Apply basic engineering fundamental in the domain of practical applications FORproblem identification, formulation and solution
CO 3	Attempt & Design a problem solution in a right approach to complex problems
CO 4	Cultivate the habit of working in a team
CO 5	Correlate the theoretical and experimental/simulations results and draw the proper inferences
CO 6	Demonstrate the knowledge, skills and attitudes of a professional engineer & Prepare report as per the standard guidelines.

INDEX

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 machine 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 AI and machine learning (ML) 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 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.

1.4 Problem Definition

The proliferation of face-swap deepfake videos created using advanced AI 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.

1.5 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.

1.6 Methodology Used

The proposed methodology includes:

1. **Data Collection**: Gather a dataset containing both authentic and manipulated videos using various face-swap techniques.
2. **Data Preprocessing**: Standardize the collected data by resizing, normalization, and augmentation to enhance the robustness of the dataset.
3. **Feature Extraction**: Utilize advanced techniques to extract significant attributes indicative of face swapping or deepfake manipulations.
4. **Model Training**: Apply CNNs for spatial pattern recognition and RNNs for temporal dependency analysis across video frames.

5. **Evaluation**: Assess the model's performance using metrics such as accuracy, precision, recall, F1 score, and detection latency.
6. **Real-Time Processing**: Implement the system for real-time analysis to dynamically classify deepfake and genuine content.

Chapter 2: Literature Survey

In this chapter, a comprehensive review of existing literature on deepfake technology and detection methods is presented. It begins by surveying historical and recent advancements in deepfake creation techniques, detailing how they have evolved over time. The chapter further discusses various approaches to deepfake detection, highlighting their strengths and weaknesses. Key studies and methodologies from the field are analyzed to identify gaps in current research, providing context for the proposed study. Additionally, this chapter sets the stage for understanding the theoretical frameworks that inform the detection techniques explored in later chapters.

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 Existing Systems

In recent years, various systems have been developed to detect deep fake content, leveraging advancements in machine learning and computer vision. Prominent solutions include deep learning-based models that utilize convolutional neural networks (CNNs) for image analysis and recurrent neural networks (RNNs) for video content. Tools like TensorFlow and PyTorch have enabled researchers to build and train sophisticated models capable of identifying manipulated media with varying degrees of success.

However, despite these advancements, existing systems face several limitations. Many models are effective only against specific types of deep fakes, particularly those that utilize traditional editing techniques. As the sophistication of deepfake creation increases, these systems struggle to maintain accuracy, leading to higher false positive rates and missed detections. Furthermore, many of these solutions lack the robustness needed for real-time applications, making them impractical for widespread use.

2.3 Lacuna in the Existing Systems

The current landscape of deepfake detection systems reveals significant lacunae that hinder their effectiveness. One major issue is the performance inconsistency across different types of deep fakes. Many systems are trained on limited datasets, resulting in models that cannot generalize effectively to new or unseen manipulation techniques. This leads to vulnerabilities, particularly as new deepfake creation methods continue to evolve.

Additionally, existing systems often demonstrate high false positive rates, where legitimate content is incorrectly flagged as manipulated, causing unnecessary alarm. This issue is exacerbated by the lack of comprehensive and diverse datasets for training, which can introduce biases and limit the model's applicability in real-world scenarios. Moreover, there is a notable gap in user awareness and education regarding deepfakes, as many users lack the tools and knowledge to identify manipulated media effectively.

2.4 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 machine 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.

2.5 Focus Area

The main focus area that this project will be aimed at is creating a scalable and adaptable deepfake detection system that not only improves accuracy but enhances user engagement through intuitive design and educational tools, such as:

- **Innovative Algorithms**: Algorithm development using the best machine learning techniques available to enhance detection rates across the much wider range of deepfakes.
- **User Education:** An educational component that could create better awareness about deepfakes would educate users of how to recognize a manipulated content and the impact of deepfake technology.

Chapter 3: Requirements 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, including stakeholder interviews, surveys, and literature reviews.

3.1 Functional Requirements

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

- Input/Output Handling: It should accept video or image files as input and provide a binary classification (real or fake) as output.
- Frame Analysis: The system should analyze individual frames using CNNs to detect spatial inconsistencies in faces (e.g., unnatural facial features, artifacts).
- Comprehensive Logging: The system should log all detection activities and results, providing evidence in cases where a deepfake is detected.

3.2 Non-Functional Requirements

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

- Performance: The system must be able to process videos quickly enough to support real-time detection, maintaining a latency below a set threshold.
- Scalability: The system should handle various resolutions and frame rates without significant degradation in performance.
- Accuracy: It should consistently achieve high accuracy in detecting manipulated videos, with a target of at least 95% classification accuracy.
- Usability: The system interface should be user-friendly, providing clear indications of video authenticity.
- Robustness: It should be resilient against noise or minor distortions in videos to avoid false positives or negatives.

3.3 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.

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 Techniques Utilized till Date for the Proposed System

- Data Augmentation: ImageDataGenerator was employed to enhance the training dataset through various transformations, including rescaling, zooming, and shearing. This technique helps improve model robustness by introducing variability in the training data, which can mitigate overfitting.
- Convolutional Neural Networks (CNNs): The model uses convolutional layers (Conv2D) for feature extraction from images. Convolutional layers are crucial for identifying patterns and structures within image data, making them ideal for tasks like image classification.
- Transfer Learning with ResNet50: The ResNet50 architecture was utilized as a pre-trained model for feature extraction. This technique leverages the knowledge gained from training on the ImageNet dataset, allowing for improved performance and reduced training time for our specific task by using the weights of a well-established model.
- Recurrent Neural Networks (RNNs): LSTM (Long Short-Term Memory) layers were integrated to capture sequential dependencies in the data. While traditionally used for time-series data, LSTMs can also help in modeling the spatial hierarchies and temporal dependencies in image sequences.
- Model Ensemble: The architecture combines CNN and RNN elements, utilizing both spatial and sequential feature learning. This ensemble approach allows the model to take advantage of different aspects of the data, potentially improving predictive accuracy.

3.6 Tools Utilized till Date for the Proposed System

- TensorFlow and Keras: The primary libraries used for building and training the deep learning model. TensorFlow provides a flexible framework for constructing complex neural networks, while Keras offers an intuitive API for easy model development.
- PIL (Python Imaging Library): Used for image processing tasks such as loading and resizing images. This library facilitates the manipulation of image data to prepare it for input into the neural network.

- Matplotlib: Employed for visualizing training and validation metrics. This tool helps in plotting loss and accuracy graphs, allowing for an easy assessment of model performance over epochs.
- Google Colab: The development environment used for coding and running the model. Google Colab provides access to GPUs, facilitating faster training of deep learning models.
- NumPy: Utilized for numerical operations and data manipulation, particularly in handling arrays and matrices essential for image data processing and model predictions.

3.7 Project Proposal

The proposed system aims to deliver a reliable deepfake detection tool that combines spatial and temporal analysis using CNNs and RNNs. It will process videos frame by frame, analyzing inconsistencies in facial features and transitions. The system will be implemented to support real-time detection with high accuracy and user-friendly interface features.

Chapter 4: Proposed Design

The design of the proposed system "Deepfake Video Detection using AI/ML Techniques" involves a detailed representation of the architecture, modular components, and different diagrams to explain the system's functionality.

4.1 Block Diagram Representation of the Proposed System

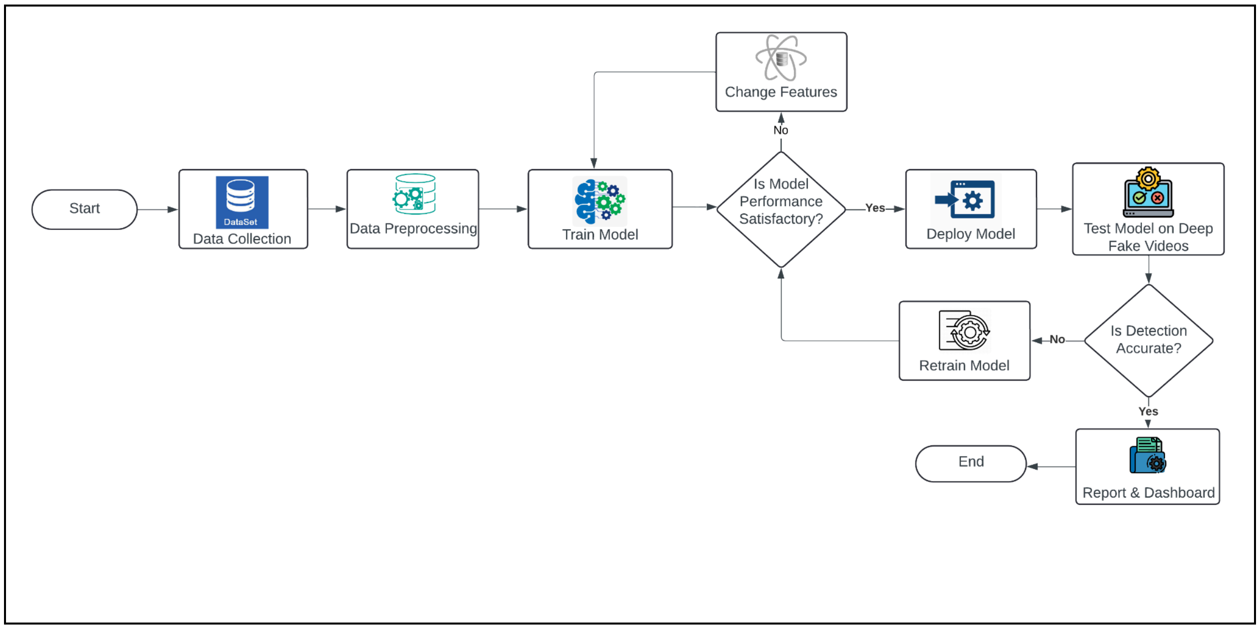


Fig 1: Block Diagram

- **Start**: The process begins with collecting video data.
- **Data Collection**: The dataset containing deepfake and real videos is gathered. These videos serve as the input data for the model training and testing phases.
- **Data Preprocessing**: Videos are preprocessed to extract frames, resize, normalize, and apply other transformations to prepare the data for training.
- **Train Model**: The preprocessed data is used to train the AI/ML model. Techniques such as CNNs for spatial analysis and RNNs for temporal analysis may be employed here.

- **Is Model Performance Satisfactory?**: The model's performance is evaluated. If the results are not satisfactory, features can be modified or adjusted to improve the model.
- **Change Features** (if required): Modifications to features or model hyperparameters are made to enhance model performance.
- **Deploy Model**: Once the model's performance is satisfactory, it is deployed for real-world testing.
- **Test Model on Deepfake Videos**: The deployed model is tested using deepfake videos to evaluate its detection accuracy.
- **Is Detection Accurate?**: If the detection accuracy is not sufficient, the model is retrained to improve the results.
- **Retrain Model** (if required): The model undergoes further training to address any shortcomings and boost detection performance.
- **Report & Dashboard**: If detection is accurate, a report is generated, and results are displayed on a dashboard.
- **End**: The process concludes after generating the report.

4.2 Modular Diagram Representation of the Proposed System

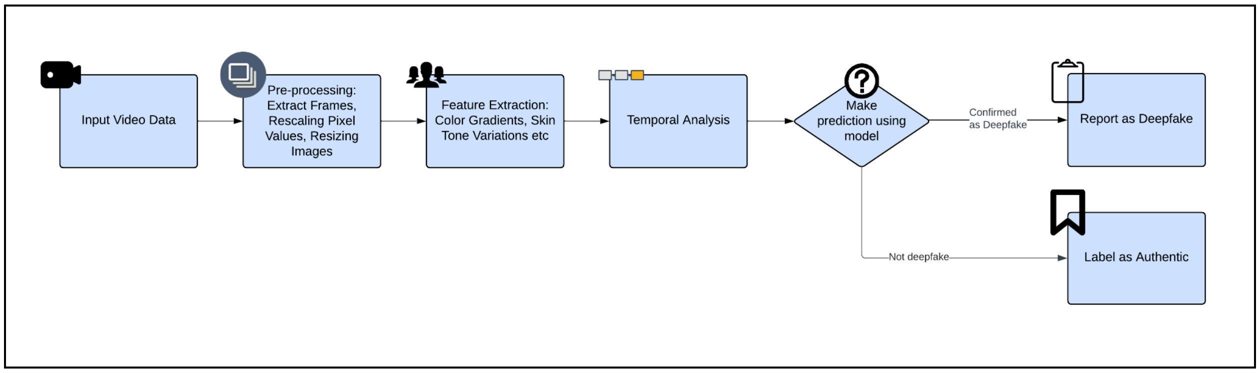

Input Video Data
Pre-processing: Extract Frames, Rescaling Pixel Values, Resizing Images
Feature Extraction: Color Gradients, Skin Tone Variations etc
Temporal Analysis
Make prediction using model
Confirmed as Deepfake
Report as Deepfake
Not deepfake
Label as Authentic

Fig 2: Modular Diagram

- **Input Video Data**: Raw video is provided.
- **Pre-processing**: Extract frames, resize images, and rescale pixel values.
- **Feature Extraction**: Analyze color gradients and skin tone variations for anomalies.
- **Temporal Analysis**: Examine frame sequences for inconsistencies over time.
- **Prediction Using Model**: The model classifies the video as either deepfake or authentic.

- **Output**:
    - **Report as Deepfake** (if manipulation detected).
    - **Label as Authentic** (if no manipulation detected).

4.3 Design of the Proposed System:

a. Flowchart for the Proposed System

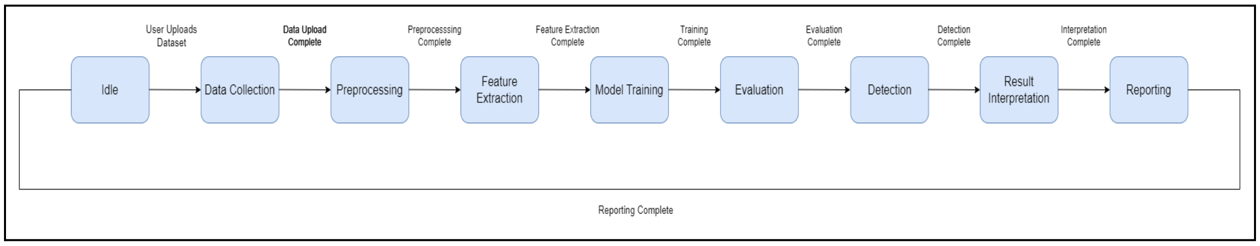


Fig 3: Flowchart

Illustrates the flow of activities from video upload to output display.

- **Idle**: System waits for user input (dataset upload).
- **Data Collection**: User uploads real and deepfake video datasets.
- **Preprocessing**: Prepares data through tasks like frame extraction and normalization.
- **Feature Extraction**: Extracts distinguishing features from the preprocessed data.
- **Model Training**: Trains a machine learning model on the extracted features, tuning hyperparameters.
- **Evaluation**: Assesses model performance using metrics like accuracy and F1 score.
- **Detection**: Applies the trained model to detect deepfakes in new data.
- **Results Interpretation**: Analyzes detection outcomes and patterns.
- **Reporting**: Generates a report summarizing findings and metrics, returning to the idle state.

b. Activity Diagram

The activity diagram outlines the sequential process of detecting deepfake videos in the proposed system.

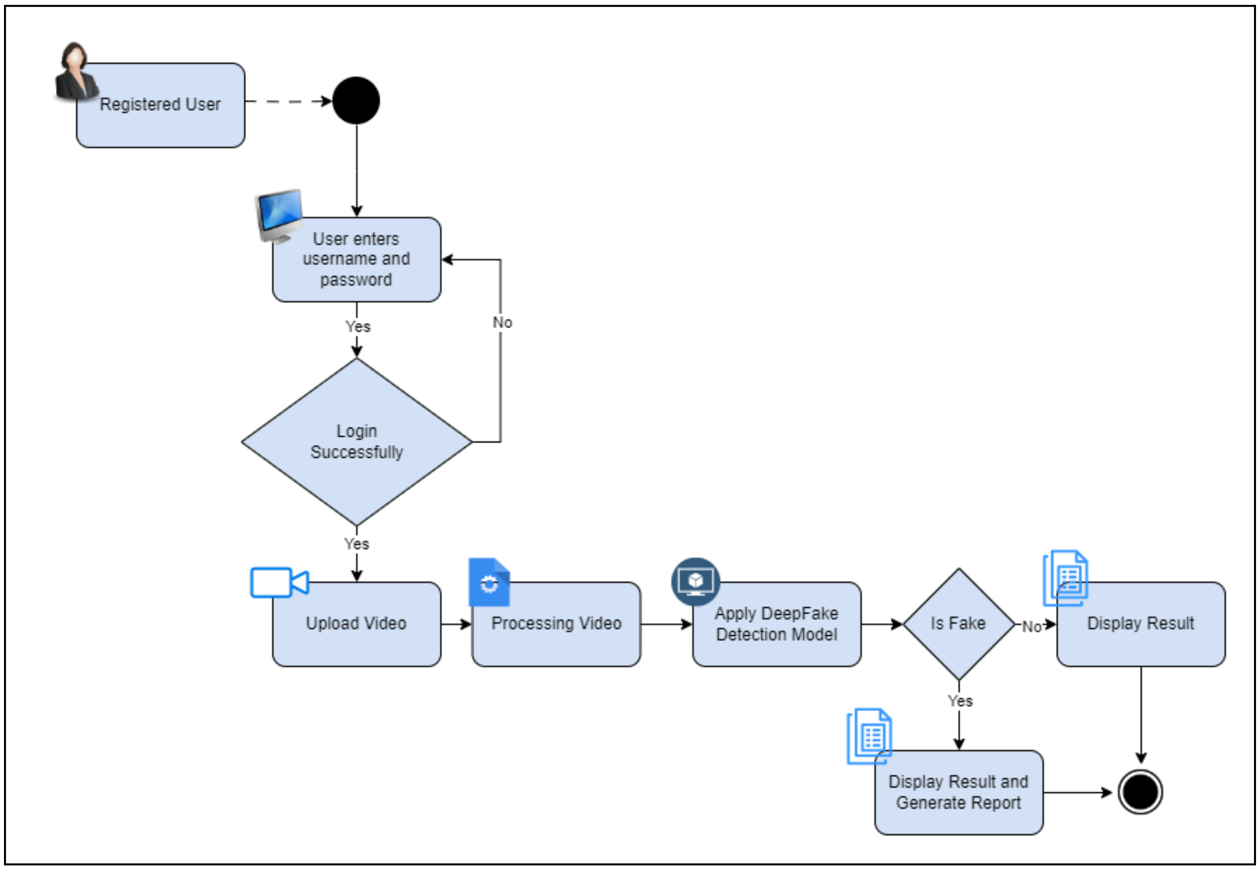

Registered User
User enters username and password
Yes
No
Login Successfully
Yes
Upload Video
Processing Video
Apply DeepFake Detection Model
Is Fake
No
Display Result
Yes
Display Result and Generate Report

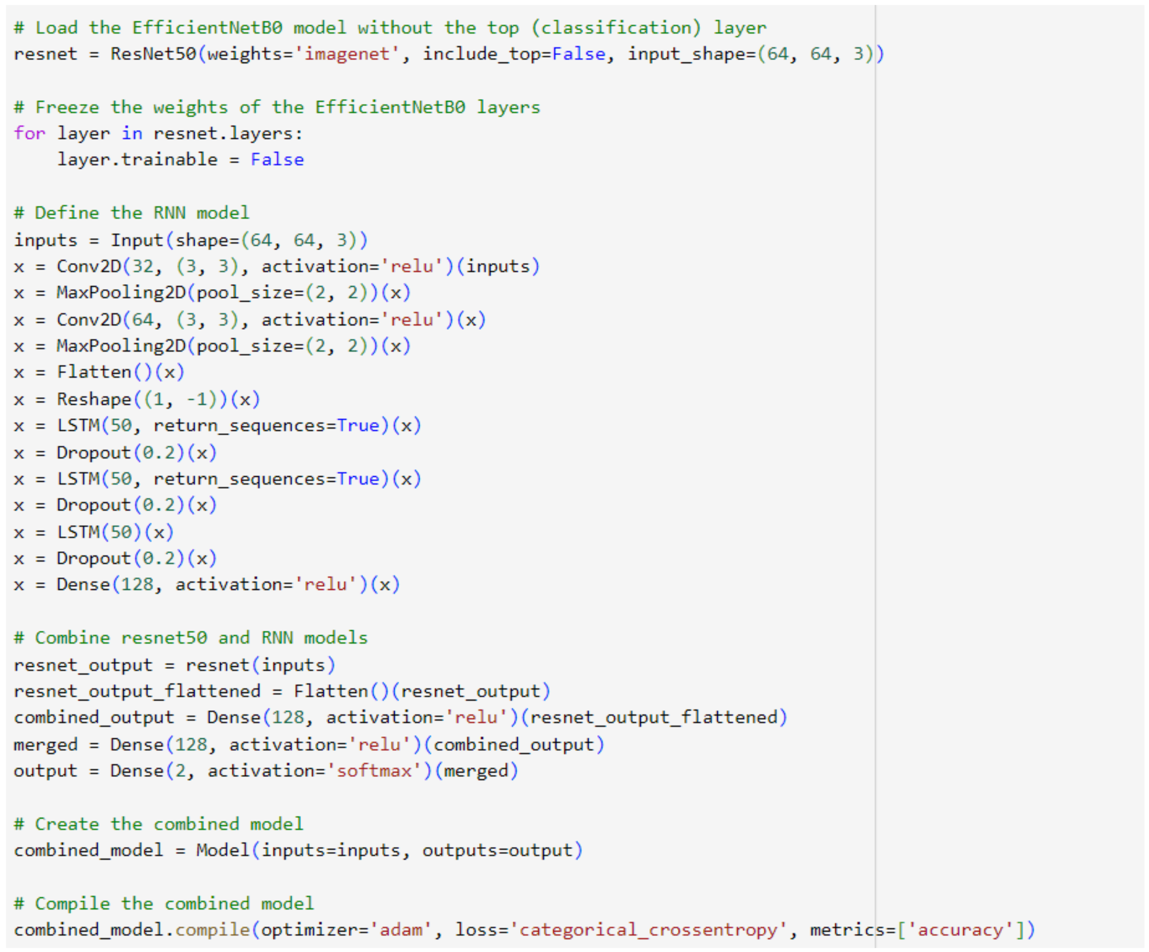
Fig 4: Activity Diagram

- **Registered User**: The process starts with a registered user.
- **User Enters Credentials**: The user inputs their username and password.
    - If login is **unsuccessful**, they are prompted to re-enter credentials.
    - If login is **successful**, the user can proceed.
- **Upload Video**: The user uploads a video for analysis.
- **Processing Video**: The system processes the uploaded video.
- **Apply Deepfake Detection Model**: The processed video is analyzed using the deepfake detection model.
- **Decision – Is Fake**: The system checks if the video is a deepfake.
    - If **yes** (fake), the system displays the result and generates a report.

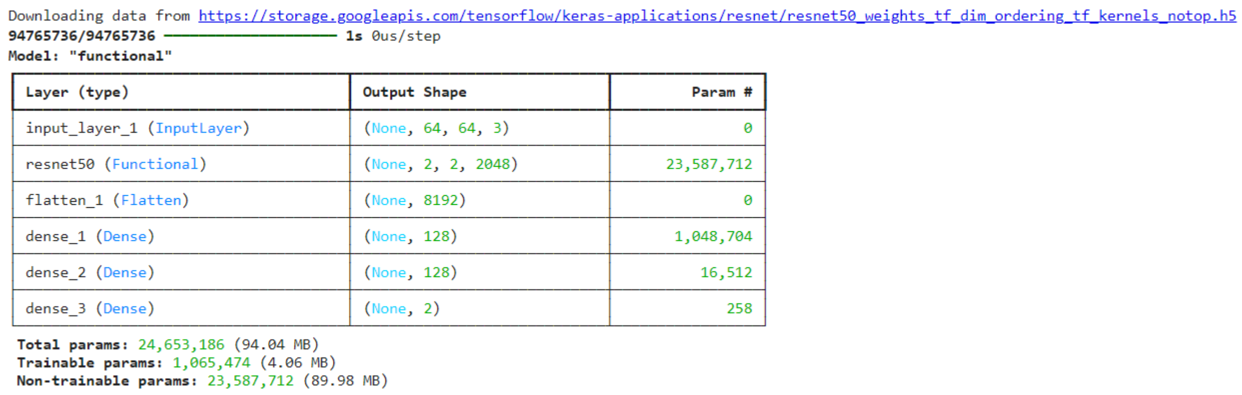
    ○ If **no** (authentic), the system simply displays the result.
- **End**: The process concludes after displaying the results.

d. Screenshots of Implementation (at least four)

ResNet50 model



```
# Load the EfficientNetB0 model without the top (classification) layer
resnet = ResNet50(weights='imagenet', include_top=False, input_shape=(64, 64, 3))

# Freeze the weights of the EfficientNetB0 layers
for layer in resnet.layers:
    layer.trainable = False

# Define the RNN model
inputs = Input(shape=(64, 64, 3))
x = Conv2D(32, (3, 3), activation='relu')(inputs)
x = MaxPooling2D(pool_size=(2, 2))(x)
x = Conv2D(64, (3, 3), activation='relu')(x)
x = MaxPooling2D(pool_size=(2, 2))(x)
x = Flatten()(x)
x = Reshape((1, -1))(x)
x = LSTM(50, return_sequences=True)(x)
x = Dropout(0.2)(x)
x = LSTM(50, return_sequences=True)(x)
x = Dropout(0.2)(x)
x = LSTM(50)(x)
x = Dropout(0.2)(x)
x = Dense(128, activation='relu')(x)

# Combine resnet50 and RNN models
resnet_output = resnet(inputs)
resnet_output_flattened = Flatten()(resnet_output)
combined_output = Dense(128, activation='relu')(resnet_output_flattened)
merged = Dense(128, activation='relu')(combined_output)
output = Dense(2, activation='softmax')(merged)

# Create the combined model
combined_model = Model(inputs=inputs, outputs=output)

# Compile the combined model
combined_model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

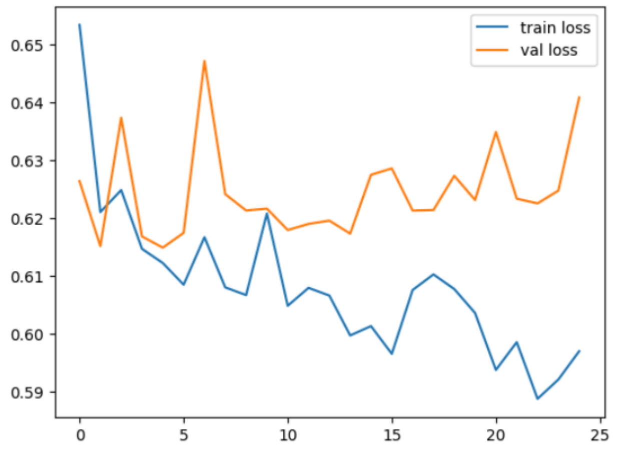


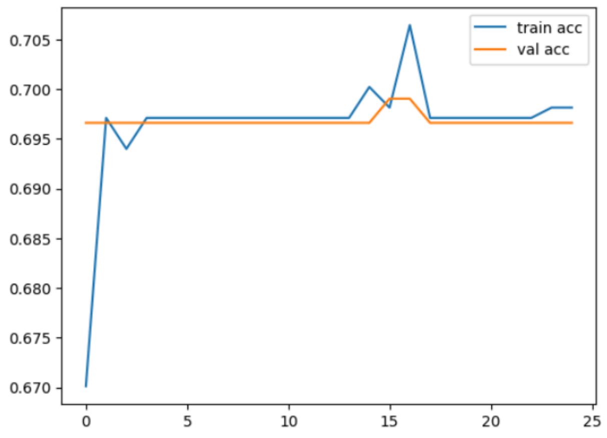
```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5
94765736/94765736 ━━━━━━━━━━━━━━━━━━━━ 1s 0us/step
Model: "functional"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━┓
┃ Layer (type)                  ┃ Output Shape          ┃      Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━┩
│ input_layer_1 (InputLayer)    │ (None, 64, 64, 3)     │            0 │
├───────────────────────────────┼───────────────────────┼──────────────┤
│ resnet50 (Functional)         │ (None, 2, 2, 2048)    │   23,587,712 │
├───────────────────────────────┼───────────────────────┼──────────────┤
│ flatten_1 (Flatten)           │ (None, 8192)          │            0 │
├───────────────────────────────┼───────────────────────┼──────────────┤
│ dense_1 (Dense)               │ (None, 128)           │    1,048,704 │
├───────────────────────────────┼───────────────────────┼──────────────┤
│ dense_2 (Dense)               │ (None, 128)           │       16,512 │
├───────────────────────────────┼───────────────────────┼──────────────┤
│ dense_3 (Dense)               │ (None, 2)             │          258 │
└───────────────────────────────┴───────────────────────┴──────────────┘
 Total params: 24,653,186 (94.04 MB)
 Trainable params: 1,065,474 (4.06 MB)
 Non-trainable params: 23,587,712 (89.98 MB)
```

```
import numpy as np
from glob import glob
import matplotlib.pyplot as plt

# Assuming model.fit() is called like this
history = combined_model.fit(x=training_set, validation_data=test_set, epochs=25)
```

```
Epoch 1/25
/usr/local/lib/python3.10/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWarning: Your `PyDat
  self._warn_if_super_not_called()
31/31 ━━━━━━━━━━━━━━━━━━━━ 144s 4s/step - accuracy: 0.6101 - loss: 0.7028 - val_accuracy: 0.6966 - val_loss: 0.6264
Epoch 2/25
31/31 ━━━━━━━━━━━━━━━━━━━━ 40s 1s/step - accuracy: 0.7034 - loss: 0.6172 - val_accuracy: 0.6966 - val_loss: 0.6152
Epoch 3/25
31/31 ━━━━━━━━━━━━━━━━━━━━ 46s 1s/step - accuracy: 0.6865 - loss: 0.6286 - val_accuracy: 0.6966 - val_loss: 0.6373
Epoch 4/25
31/31 ━━━━━━━━━━━━━━━━━━━━ 72s 1s/step - accuracy: 0.6839 - loss: 0.6255 - val_accuracy: 0.6966 - val_loss: 0.6168
Epoch 5/25
31/31 ━━━━━━━━━━━━━━━━━━━━ 41s 1s/step - accuracy: 0.6891 - loss: 0.6214 - val_accuracy: 0.6966 - val_loss: 0.6149
Epoch 6/25
31/31 ━━━━━━━━━━━━━━━━━━━━ 43s 1s/step - accuracy: 0.7031 - loss: 0.6056 - val_accuracy: 0.6966 - val_loss: 0.6174
Epoch 7/25
31/31 ━━━━━━━━━━━━━━━━━━━━ 38s 1s/step - accuracy: 0.7030 - loss: 0.6102 - val_accuracy: 0.6966 - val_loss: 0.6471
Epoch 8/25
31/31 ━━━━━━━━━━━━━━━━━━━━ 35s 1s/step - accuracy: 0.7012 - loss: 0.6093 - val_accuracy: 0.6966 - val_loss: 0.6241
Epoch 9/25
31/31 ━━━━━━━━━━━━━━━━━━━━ 36s 1s/step - accuracy: 0.6910 - loss: 0.6060 - val_accuracy: 0.6966 - val_loss: 0.6213
Epoch 10/25
31/31 ━━━━━━━━━━━━━━━━━━━━ 39s 1s/step - accuracy: 0.7075 - loss: 0.6032 - val_accuracy: 0.6966 - val_loss: 0.6216
Epoch 11/25
31/31 ━━━━━━━━━━━━━━━━━━━━ 37s 1s/step - accuracy: 0.6949 - loss: 0.6052 - val_accuracy: 0.6966 - val_loss: 0.6179
Epoch 12/25
31/31 ━━━━━━━━━━━━━━━━━━━━ 35s 1s/step - accuracy: 0.6789 - loss: 0.6237 - val_accuracy: 0.6966 - val_loss: 0.6190
```

```
# Plot the loss
plt.plot(history.history['loss'], label='train loss')
plt.plot(history.history['val_loss'], label='val loss')
plt.legend()
plt.show()
plt.savefig('LossVal_loss')

# Plot the accuracy
plt.plot(history.history['accuracy'], label='train acc')
plt.plot(history.history['val_accuracy'], label='val acc')
plt.legend()
plt.show()
plt.savefig('AccVal_acc')
```





```
<Figure size 640x480 with 0 Axes>
```

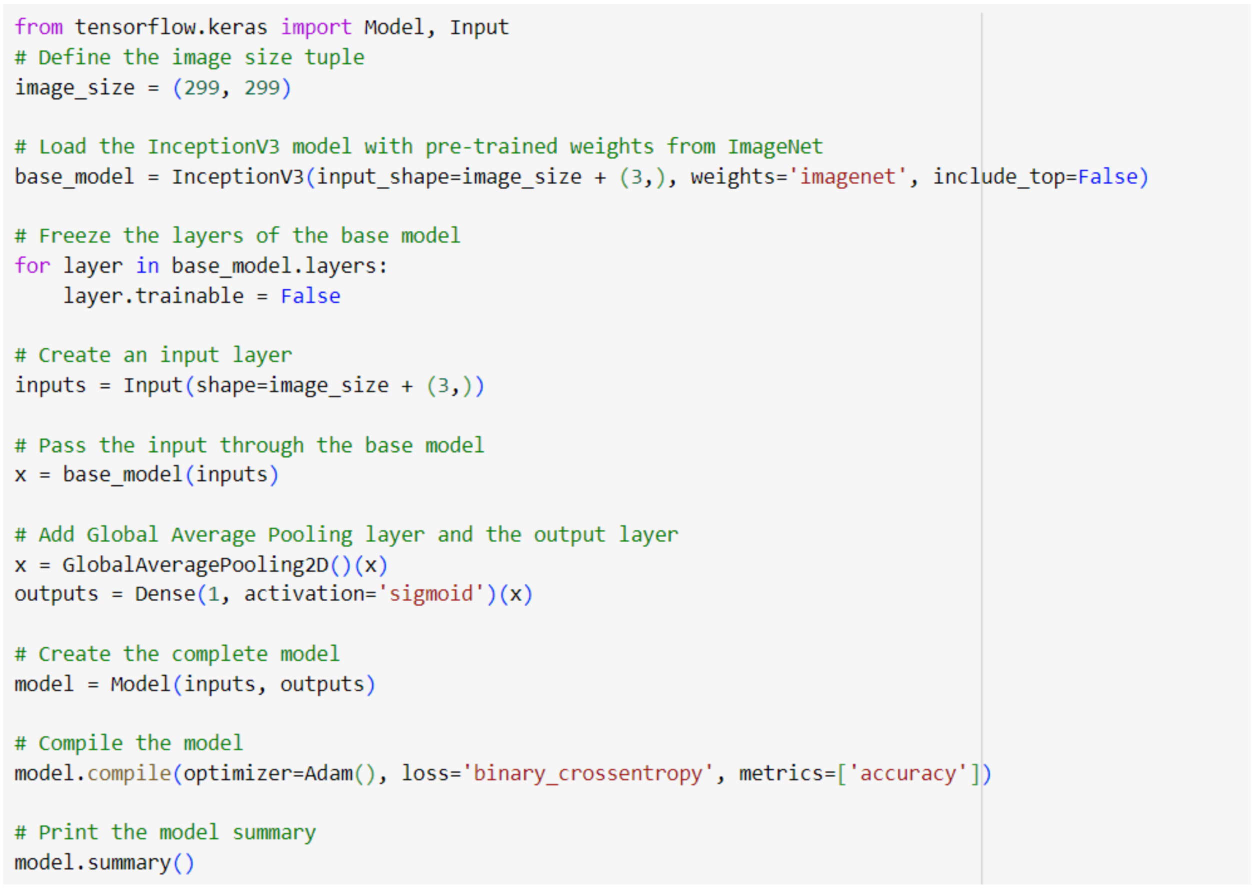
```
# Print the final training and validation accuracy
final_train_accuracy = history.history['accuracy'][-1]
final_val_accuracy = history.history['val_accuracy'][-1]

print("Final Training Accuracy:", final_train_accuracy)
print("Final Validation Accuracy:", final_val_accuracy)
```

```
Final Training Accuracy: 0.6981327533721924
Final Validation Accuracy: 0.696601927280426
```

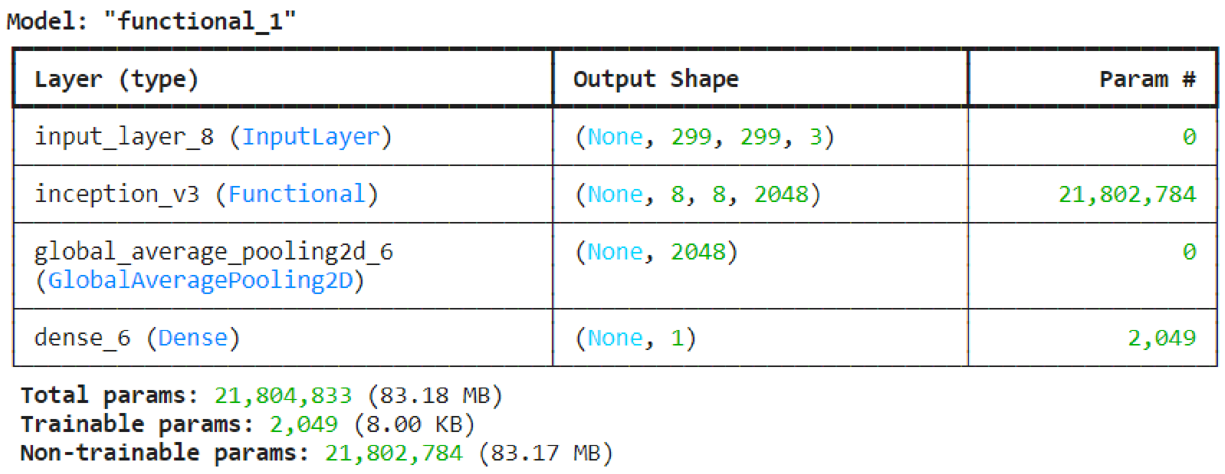
```
# Evaluate the model on the test set
test_results = combined_model.evaluate(test_set, steps=len(test_set))

# Print the evaluation results
print("Test Loss:", test_results[0])
print("Test Accuracy:", test_results[1])
```

```
13/13 ━━━━━━━━━━━━━━━━━━━━ 9s 627ms/step - accuracy: 0.6950 - loss: 0.6482
Test Loss: 0.6408353447914124
Test Accuracy: 0.696601927280426
```

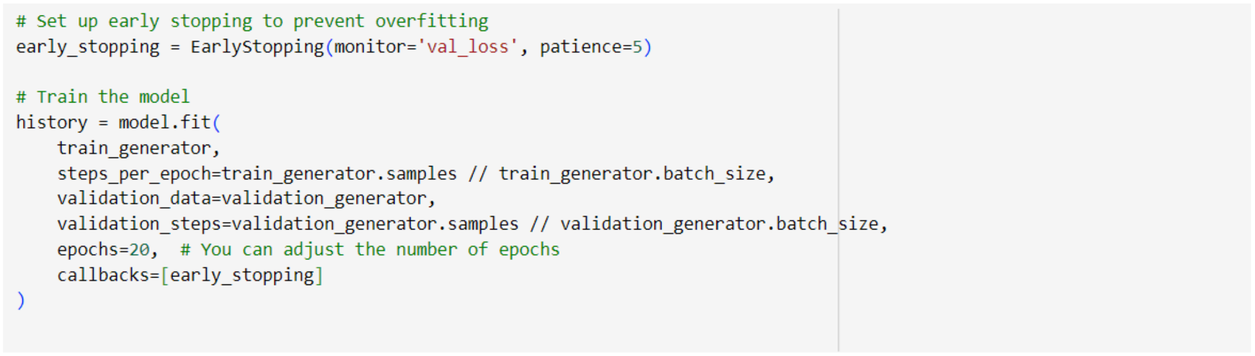
InceptionV3 model



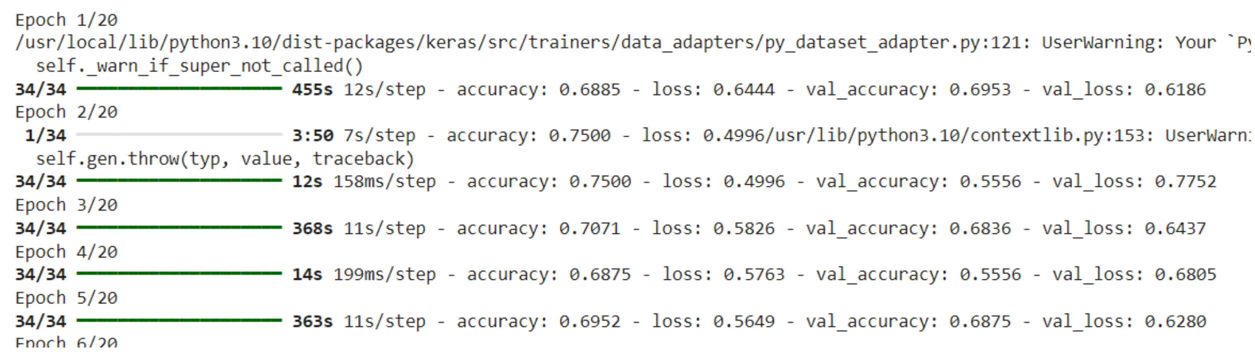
```python
from tensorflow.keras import Model, Input
# Define the image size tuple
image_size = (299, 299)

# Load the InceptionV3 model with pre-trained weights from ImageNet
base_model = InceptionV3(input_shape=image_size + (3,), weights='imagenet', include_top=False)

# Freeze the layers of the base model
for layer in base_model.layers:
    layer.trainable = False

# Create an input layer
inputs = Input(shape=image_size + (3,))

# Pass the input through the base model
x = base_model(inputs)

# Add Global Average Pooling layer and the output layer
x = GlobalAveragePooling2D()(x)
outputs = Dense(1, activation='sigmoid')(x)

# Create the complete model
model = Model(inputs, outputs)

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

# Print the model summary
model.summary()
```



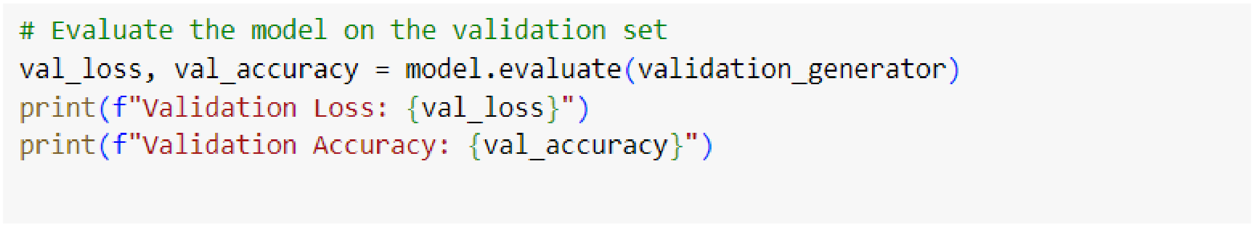
```
Model: "functional_1"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                         ┃ Output Shape                ┃         Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━┩
│ input_layer_8 (InputLayer)           │ (None, 299, 299, 3)         │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ inception_v3 (Functional)            │ (None, 8, 8, 2048)          │      21,802,784 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ global_average_pooling2d_6           │ (None, 2048)                │               0 │
│ (GlobalAveragePooling2D)             │                             │                 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dense_6 (Dense)                      │ (None, 1)                   │           2,049 │
└──────────────────────────────────────┴─────────────────────────────┴─────────────────┘
 Total params: 21,804,833 (83.18 MB)
 Trainable params: 2,049 (8.00 KB)
 Non-trainable params: 21,802,784 (83.17 MB)
```



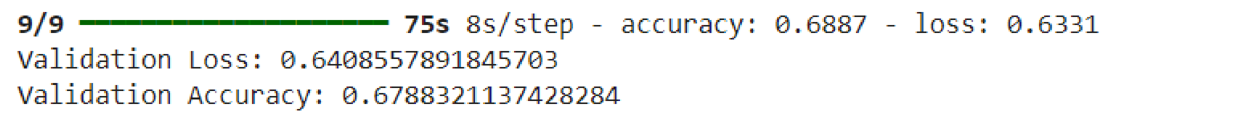
```python
# Set up early stopping to prevent overfitting
early_stopping = EarlyStopping(monitor='val_loss', patience=5)

# Train the model
history = model.fit(
    train_generator,
    steps_per_epoch=train_generator.samples // train_generator.batch_size,
    validation_data=validation_generator,
    validation_steps=validation_generator.samples // validation_generator.batch_size,
    epochs=20,  # You can adjust the number of epochs
    callbacks=[early_stopping]
)
```



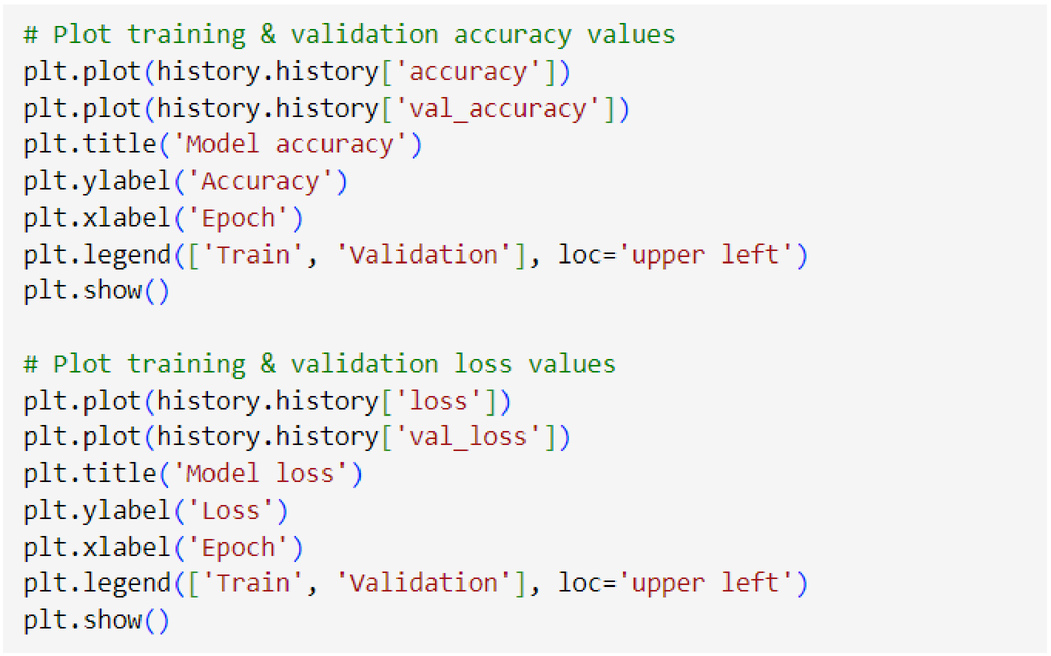
```
Epoch 1/20
/usr/local/lib/python3.10/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWarning: Your `P
  self._warn_if_super_not_called()
34/34 ━━━━━━━━━━━━━━━━━━━━ 455s 12s/step - accuracy: 0.6885 - loss: 0.6444 - val_accuracy: 0.6953 - val_loss: 0.6186
Epoch 2/20
 1/34 ━━━━━━━━━━━━━━━━━━━━ 3:50 7s/step - accuracy: 0.7500 - loss: 0.4996/usr/lib/python3.10/contextlib.py:153: UserWarn
  self.gen.throw(typ, value, traceback)
34/34 ━━━━━━━━━━━━━━━━━━━━ 12s 158ms/step - accuracy: 0.7500 - loss: 0.4996 - val_accuracy: 0.5556 - val_loss: 0.7752
Epoch 3/20
34/34 ━━━━━━━━━━━━━━━━━━━━ 368s 11s/step - accuracy: 0.7071 - loss: 0.5826 - val_accuracy: 0.6836 - val_loss: 0.6437
Epoch 4/20
34/34 ━━━━━━━━━━━━━━━━━━━━ 14s 199ms/step - accuracy: 0.6875 - loss: 0.5763 - val_accuracy: 0.5556 - val_loss: 0.6805
Epoch 5/20
34/34 ━━━━━━━━━━━━━━━━━━━━ 363s 11s/step - accuracy: 0.6952 - loss: 0.5649 - val_accuracy: 0.6875 - val_loss: 0.6280
Epoch 6/20
```



```python
# Evaluate the model on the validation set
val_loss, val_accuracy = model.evaluate(validation_generator)
print(f"Validation Loss: {val_loss}")
print(f"Validation Accuracy: {val_accuracy}")
```



```
9/9 ━━━━━━━━━━━━━━━━━━━━ 75s 8s/step - accuracy: 0.6887 - loss: 0.6331
Validation Loss: 0.6408557891845703
Validation Accuracy: 0.6788321137428284
```



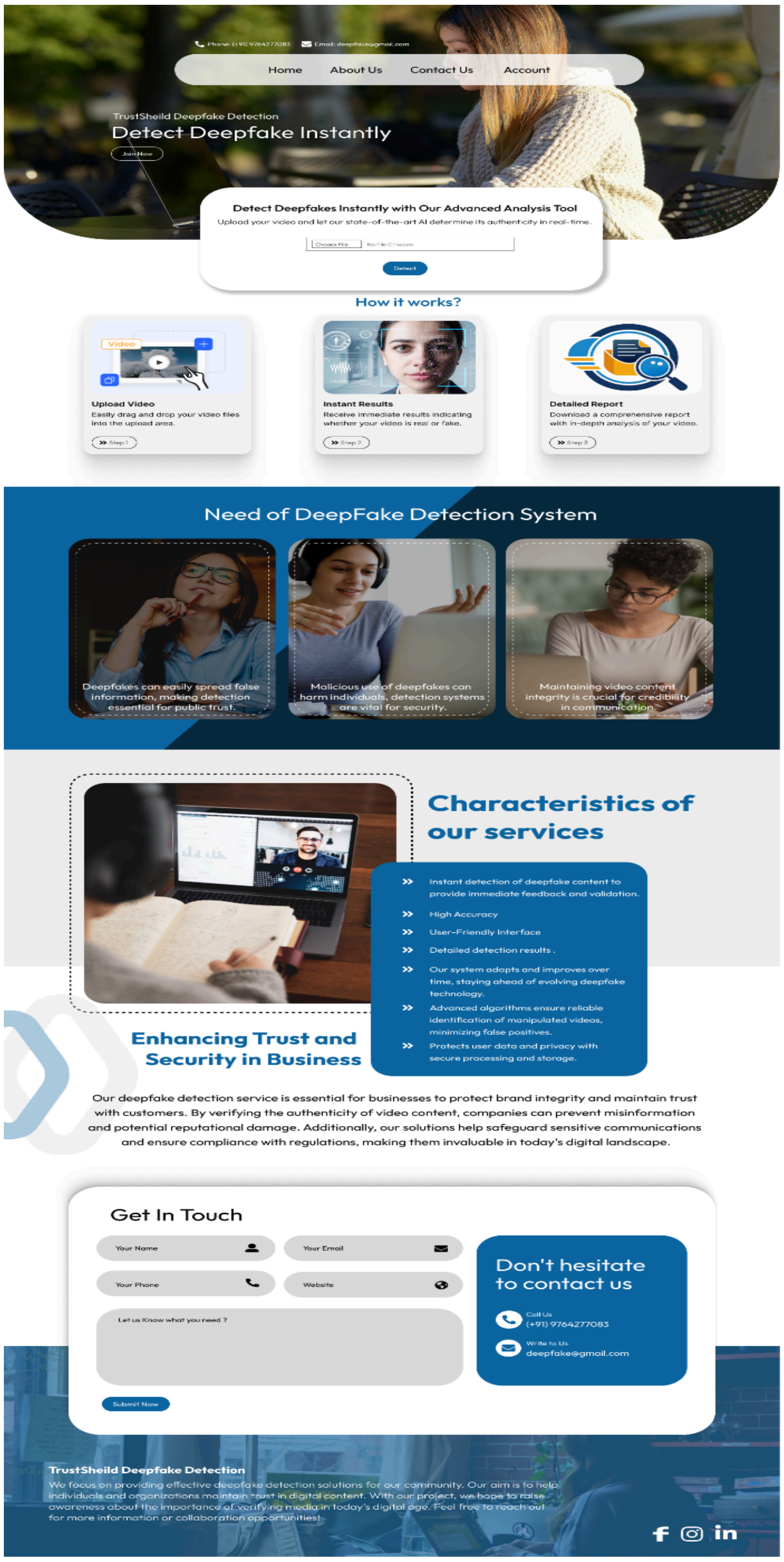
```python
# Plot training & validation accuracy values
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()

# Plot training & validation loss values
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()
```

Result Comparision :

Feature	**ResNet50**	**InceptionNetV3**
Training Accuracy	0.6950	0.6887
Training Loss	0.6482	0.6331
Test/Validation Loss	0.640835	0.640855
Test/Validation Accuracy	0.696601	0.678832
Training Time per Step	~627 ms	~8 s
Strengths	Faster training time, slightly better accuracy	accuracy Lower loss, powerful at capturing multi-scale features
Weaknesses	Slightly higher loss, potential overfitting in fewer epochs	Slower training, slightly lower accuracy

Table No. 1: Result Comparison

GUI :



TrustSheild Deepfake Detection

Detect Deepfake Instantly

Detect Deepfakes Instantly with Our Advanced Analysis Tool

Upload your video and let our state-of-the-art AI determine its authenticity in real-time.

How it works?

Upload Video
Easily drag and drop your video files into the upload area.

Instant Results
Receive immediate results indicating whether your video is real or fake.

Detailed Report
Download a comprehensive report with in-depth analysis of your video.

Need of DeepFake Detection System

Deepfakes can easily spread false information, making detection essential for public trust.

Malicious use of deepfakes can harm individuals, detection systems are vital for security.

Maintaining video content integrity is crucial for credibility in communication.

Characteristics of our services

- Instant detection of deepfake content to provide immediate feedback and validation.
- High Accuracy
- User-Friendly Interface
- Detailed detection results .
- Our system adapts and improves over time, staying ahead of evolving deepfake technology.
- Advanced algorithms ensure reliable identification of manipulated videos, minimizing false positives.
- Protects user data and privacy with secure processing and storage.

Enhancing Trust and Security in Business

Our deepfake detection service is essential for businesses to protect brand integrity and maintain trust with customers. By verifying the authenticity of video content, companies can prevent misinformation and potential reputational damage. Additionally, our solutions help safeguard sensitive communications and ensure compliance with regulations, making them invaluable in today's digital landscape.

Get In Touch

Don't hesitate to contact us

Call Us
(+91) 9764277083

Write to Us
deepfake@gmail.com

TrustSheild Deepfake Detection

We focus on providing effective deepfake detection solutions for our community. Our aim is to help individuals and organizations maintain trust in digital content. With our project, we hope to raise awareness about the importance of verifying media in today's digital age. Feel free to reach out for more information or collaboration opportunities!

4.4 Project Scheduling & Tracking using Timeline / Gantt Chart

A Gantt chart is used to track the progress of each development phase:

Task	Start Date	End Date	Duration
Project Initiation	Aug 9	Aug 13	4 days
Literature Review	Aug 13	Aug 20	3 days
Dataset Collection and Preparation	Aug 20	Sep 11	1 week
Data Preprocessing	Sep 11	Sep 20	2 weeks
Feature Selection and Extraction	Sep 20	Sep 27	2 weeks
Model Development and Training	Sep 27	October 4	4 weeks
Model Evaluation and Refinement	Oct 4	Oct 11	2 weeks
Documentation and Report Preparation	Oct 11	Oct 18	1 week

Table No. 2: Project Scheduling

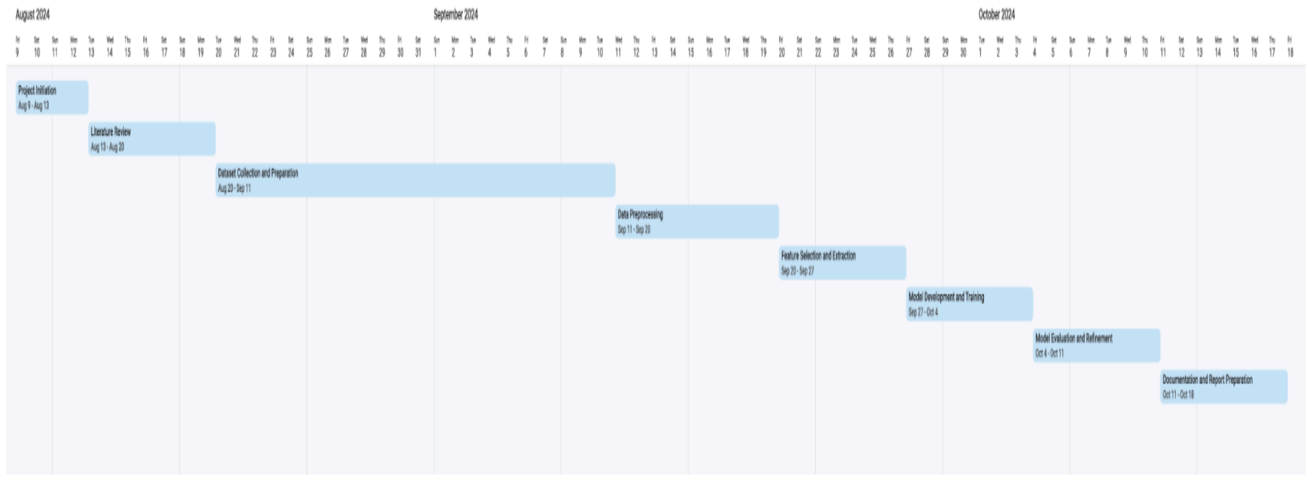
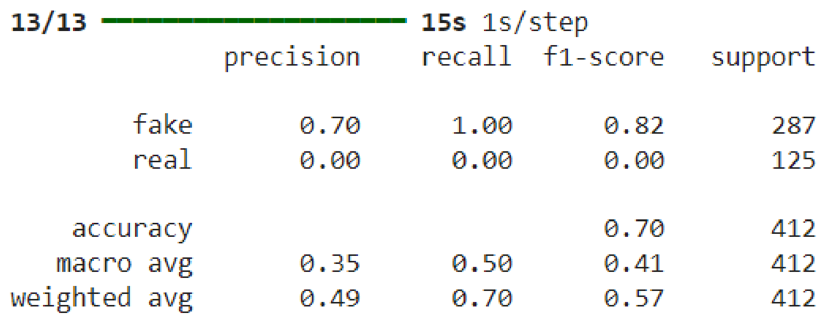


Fig 5: Gantt Chart

Chapter 5: Proposed Results and Discussions

5.1 Determination of Efficiency



```
13/13 ━━━━━━━━━━━━━━━━━━━━ 15s 1s/step
              precision    recall  f1-score   support

        fake       0.70      1.00      0.82       287
        real       0.00      0.00      0.00       125

    accuracy                           0.70       412
   macro avg       0.35      0.50      0.41       412
weighted avg       0.49      0.70      0.57       412
```

The classification report presents the performance metrics of our deep fake detection model. Below is a detailed analysis of these metrics:

1. Precision

- **Fake Class**: The precision for detecting fake images is **0.70**. This means that when the model predicts an image as fake, it is correct 70% of the time.
- **Real Class**: The precision for detecting real images is **0.00**. This indicates that the model fails to correctly identify any real images, resulting in no true positive predictions for this class.

2. Recall

- **Fake Class**: The recall for fake images is **1.00**. This signifies that the model successfully identifies all actual fake images, with no false negatives.
- **Real Class**: The recall for real images is **0.00**. The model does not identify any real images, leading to a situation where all real images are classified as fake (false positives).

3. F1-Score

- **Fake Class**: The F1-score for fake images is **0.82**, reflecting a good balance between precision and recall for this class.
- **Real Class**: The F1-score for real images is **0.00**, indicating a complete lack of effective detection.

4. Support

- The support values indicate the number of true instances for each class in the dataset:
  - **Fake**: 287 instances
  - **Real**: 125 instances

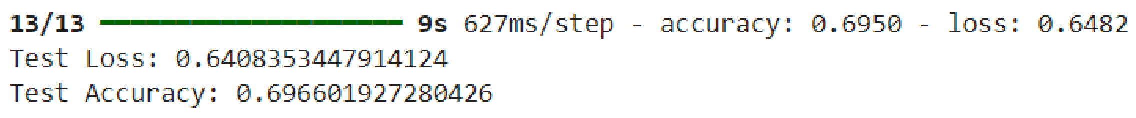
5. Overall Accuracy

- The overall accuracy of the model is **70%**, which suggests that the model is able to classify 70% of the total instances correctly. However, this metric can be misleading, especially in cases of class imbalance.

6. Macro and Weighted Averages

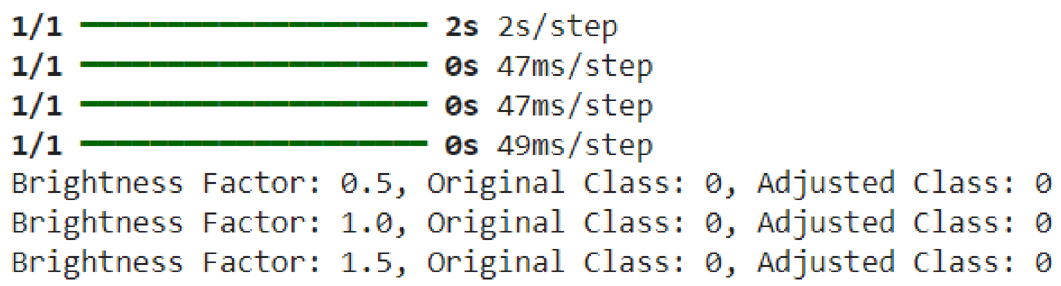
- **Macro Average**: The macro averages for precision (0.35), recall (0.50), and F1-score (0.41) indicate that, on average, the model performs poorly across both classes, particularly due to the lack of successful detections for real images.
- **Weighted Average**: The weighted averages (0.49 precision, 0.70 recall, and 0.57 F1-score) take class support into account, showing that the performance on the fake class heavily influences these results.

5.2 Determination of Accuracy



```
13/13 ━━━━━━━━━━━━━━━━━━━━ 9s 627ms/step - accuracy: 0.6950 - loss: 0.6482
Test Loss: 0.6408353447914124
Test Accuracy: 0.696601927280426
```

5.3 Reports on Sensitivity Analysis



```
1/1 ━━━━━━━━━━━━━━━━━━━━ 2s 2s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 49ms/step
Brightness Factor: 0.5, Original Class: 0, Adjusted Class: 0
Brightness Factor: 1.0, Original Class: 0, Adjusted Class: 0
Brightness Factor: 1.5, Original Class: 0, Adjusted Class: 0
```

The goal of this sensitivity analysis was to assess the robustness of the trained model's predictions concerning variations in image brightness. By modifying the brightness of a test image and observing the model's classification outcomes, we aimed to understand how sensitive the model is to such changes.

Methodology:

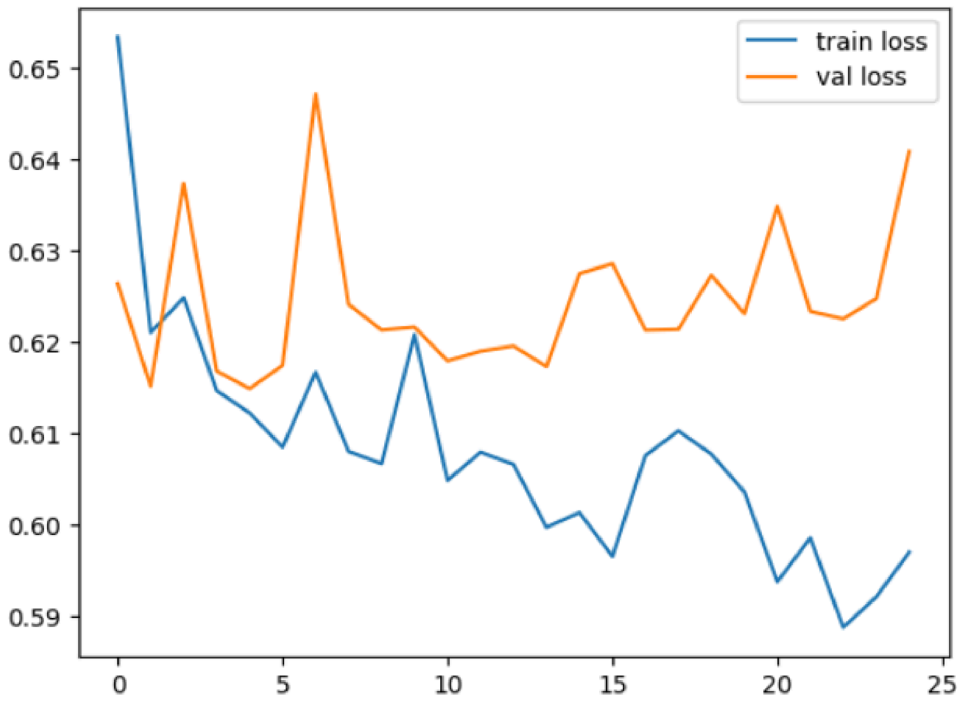
1. **Image Selection**: A sample image was selected from the dataset, specifically real_00017.jpg, which is classified as belonging to the "real" category (class 0).
2. **Preprocessing**: The image was preprocessed by resizing it to the required input dimensions of (64, 64) pixels, converting it to a NumPy array, and normalizing pixel values by dividing by 255.0.
3. **Brightness Adjustment**: The brightness of the image was systematically altered using three factors:
   - **0.5**: Represents a decrease in brightness.
   - **1.0**: Represents normal brightness (original).
   - **1.5**: Represents an increase in brightness.
4. **Prediction**: For each brightness-adjusted image, predictions were made using the trained model. The predicted class was determined by identifying the index of the maximum value in the model's output probabilities.

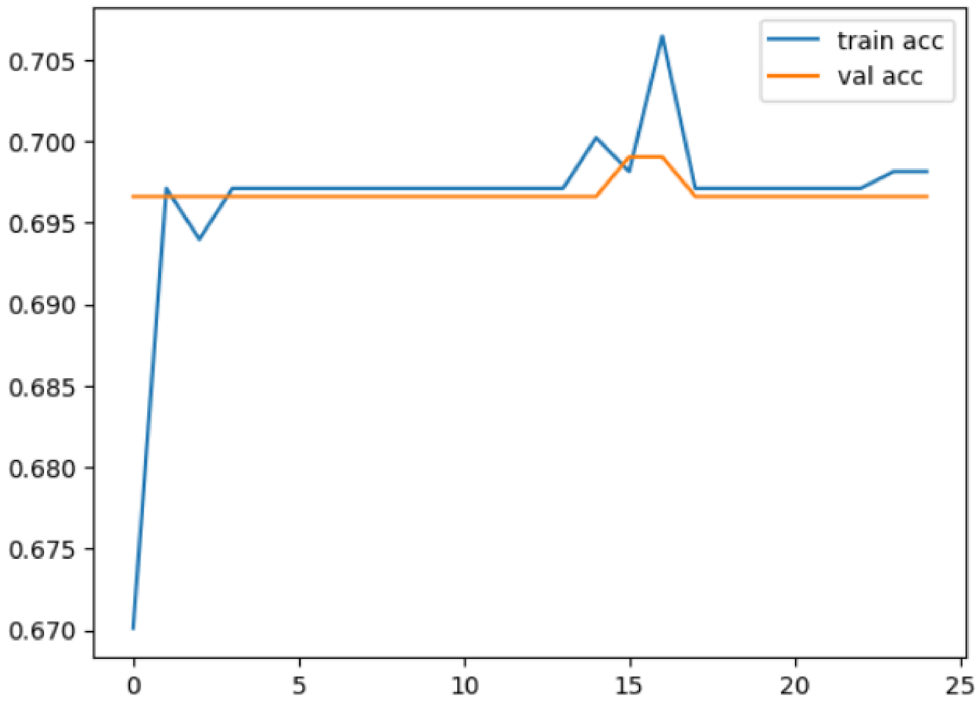
Interpretation of Results:

- **Brightness Factor 0.5**: The model maintained its classification of the image as "real" (class 0) even when the brightness was reduced by half.

- **Brightness Factor 1.0**: The original image, with normal brightness, was also classified as "real" (class 0).
- **Brightness Factor 1.5**: The model continued to classify the image as "real" (class 0) when the brightness was increased by 50%.

5.4 Graphs of Accuracy vs. Time





Chapter 6: Plan Of Action For the Next Semester

6.1 Work Done Till Date

In the current semester, significant progress was made on the Deepfake Detection Project:

1. **Literature Review**: Conducted an extensive review of deepfake detection methodologies.
2. **Data Collection**: Collected a dataset comprising genuine and deepfake videos.
3. **Data Preprocessing**: Performed preprocessing, including video frame extraction and data augmentation using Keras' ImageDataGenerator.
4. **Model Development**: Developed a hybrid model combining CNNs and LSTMs, utilizing ResNet50 for feature extraction.
5. **Training Pipeline**: Established a training pipeline, enabling efficient model training and validation.
6. **Initial Training Results**: Conducted preliminary training runs with promising accuracy outcomes.

6.2 Plan of Action for Project II

For the next semester, the focus will be on model refinement and project expansion:

1. **Model Optimization**: Conduct hyperparameter tuning and implement regularization techniques to enhance model performance.
2. **Advanced Architectures**: Explore alternative architectures, including transformers and GANs, and consider model ensembling.
3. **Comprehensive Evaluation**: Develop a framework for assessing model performance using metrics like precision, recall, and F1-score.
4. **Deployment Strategy**: Investigate deployment options, including user interfaces or APIs for real-time detection.
5. **Documentation**: Continue thorough documentation of methodologies and findings, culminating in a final report.

Chapter 7: Conclusion

The proposed AI and machine learning-based system for detecting face-swap deepfake videos represents a significant step toward addressing the growing concerns around digital media manipulation. By leveraging advanced neural network architectures, including Convolutional Neural Networks (CNNs) for spatial analysis and Recurrent Neural Networks (RNNs) for temporal dynamics, the system demonstrates a comprehensive approach to identifying subtle artifacts and anomalies in face-swapped content. The combination of spatial and temporal feature analysis allows for more accurate detection of both image and video deepfakes, ensuring a robust defense against sophisticated manipulations.

This project emphasizes the importance of real-time processing, enabling dynamic and rapid classification of media content. The performance of the system will be evaluated across multiple metrics, including accuracy, precision, recall, and F1 score, to ensure its efficacy in practical applications. Additionally, its robustness against diverse face-swap techniques and evolving deepfake technologies will be assessed through adversarial testing. These evaluations will ensure that the system not only meets current detection needs but also remains resilient against future advancements in deepfake generation.

In conclusion, the development of such an advanced detection system is crucial for safeguarding the integrity of digital media. As deepfake technologies continue to evolve, the proposed solution provides a foundation for future improvements and adaptations in detecting and mitigating the impact of manipulated content, contributing significantly to maintaining trust in visual media.

Chapter 8: References

[1] M. S. Rana, M. N. Nobi, B. Murali, and A. H. Sung, “Deepfake detection: A systematic literature review,” *IEEE Journals & Magazine*, 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9721302.

[2] 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. [Online]. Available: https://link.springer.com/article/10.1007/s11042-020-09548-2.

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

[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 *2022 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, Ankara, Turkey, 2022, pp. 1-7. doi: 10.1109/HORA55278.2022.9799858.

[5] 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.

[6] 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: https://doi.org/10.1007/s11042-020-10420-8.

[7] 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: https://doi.org/10.1007/s10462-024-10810-6.

[8] S. Waseem, S. A. R. S. Abu Bakar, B. A. Ahmed, Z. Omar, T. A. E. Eisa, and M. E. E. Dalam, "DeepFake on Face and Expression Swap: A Review," *IEEE Access*, vol. 11, pp. 117865-117906, 2023. doi: 10.1109/ACCESS.2023.3324403.

Chapter 9: Appendix

9.1 List of Figures

9.2 List of Tables