A Mini Project Report

on

DEEPFAKE DETECTION SYSTEM

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

by

ZUNAIRA FATIMA	**(160722733067)**
MIR AYAN ALI	**(160722733083)**
ANIKHA TAMKINATH	**(160722733118)**

Under the Guidance of

Mr. A.A.R. Senthil Kumaar

Assistant Professor, Dept. of CSE

Department of Computer Science and Engineering

Methodist College of Engineering and Technology,

King Koti, Abids, Hyderabad-500001

2024-2025

Methodist College of Engineering and Technology,

King Koti, Abids, Hyderabad-500001,

Department of Computer Science and Engineering

DECLARATION BY THE CANDIDATES

We, **ZUNAIRA FATIMA (160722733067), MIR AYAN ALI (160722733083)** and **ANIKHA TAMKINATH (160722733118)** students of Methodist College of Engineering and Technology, pursuing Bachelor's degree in Computer Science and Engineering, hereby declare that this Mini Project report entitled "**DEEPFAKE DETECTION SYSTEM",** carried out under the guidance of **Mr. A.A.R. Senthil Kumaar** submitted in partial fulfilment of the requirements for the degree of Bachelor of Engineering in Computer Science and Engineering. This is a record work carried out by us and the results embodied in this report have not been reproduced/copied from any source.

ZUNAIRA FATIMA (160722733067)

MIR AYAN ALI (160722733083)

ANIKHA TAMKINATH (160722733118)

Methodist College of Engineering and Technology,

King Koti, Abids, Hyderabad-500001.

Department of Computer Science and Engineering

CERTIFICATE BY THE SUPERVISOR

This is to certify that this Mini Project work entitled "**DEEPFAKE DETECTION SYSTEM**" *by* **ZUNAIRA FATIMA (160722733067), MIR AYAN ALI (160722733083)** and **ANIKHA TAMKINATH (160722733118)** submitted in partial fulfilment of the requirements for the degree of Bachelor of Engineering in Computer Science and Engineering, during the academic year 2024-2025, is a Bonafide record of work carried out by them.

Mr. A.A.R. Senthil Kumaar
Assistant Professor,
Dept. of CSE.

Methodist College of Engineering and Technology,

King Koti, Abids, Hyderabad-500001.

Department of Computer Science and Engineering

CERTIFICATE BY HEAD OF THE DEPARTMENT

This is to certify that this Mini Project work entitled "**DEEPFAKE DETECTION SYSTEM**" *by* **ZUNAIRA FATIMA (160722733067), MIR AYAN ALI (160722733083)** and **ANIKHA TAMKINATH (160722733118)** submitted in partial fulfilment of the requirements for the degree of Bachelor of Engineering in Computer Science and Engineering, during the academic year 2024-2025, is a Bonafide record of work carried out by them.

Dr. P. LAVANYA,
Professor
Head of the Department, CSE.

Methodist College of Engineering and Technology,

King Koti, Abids, Hyderabad-500001.

Department of Computer Science and Engineering

PROJECT APPROVAL CERTIFICATE

This is to certify that this Mini Project work entitled “**DEEPFAKE DETECTION SYSTEM**” *by* **ZUNAIRA FATIMA (160722733067), MIR AYAN ALI (160722733083)** and **ANIKHA TAMKINATH (160722733118)** submitted in partial fulfilment of the requirements for the degree of Bachelor of Engineering in Computer Science and Engineering during the academic year 2024-2025, is a bonafide record of work carried out by them.

INTERNAL **EXTERNAL** **HOD**

ACKNOWLEDGEMENT

We would like to express a deep sense of gratitude towards the **Dr. Lakshmipathi Rao, Director, Methodist College of Engineering and Technology,** for always being an inspiration and for always encouraging us in every possible way.

We would like to express a deep sense of gratitude towards the **Dr. Prabhu G. Benakop, Principal, Methodist College of Engineering and Technology,** for always being an inspiration and for always encouraging us in every possible way.

Our sincere thanks to **Dr. P. Lavanya, Professor** and **Head of the Department of Computer Science and Engineering,** for her valuable guidance and encouragement which has played a major role in the completion of the project and for helping us by being an example of high vision and pushing towards greater limits of achievement.

We would like to express our sincere gratitude to my project guide **Mr. A.A.R. Senthil Kumaar**, **Assistant Professor, CSE**, for giving us the opportunity to work on this topic. It would never be possible for us to take this project to this level without his innovative ideas and his relentless support and encouragement.

We would like to thank our project coordinator **Dr. MVDS Krishnamurty, Associate Professor, CSE,** who helped us by being an example of high vision and pushing towards greater limits of achievement.

We are indebted to the Department of Computer Science and Engineering and Methodist College of Engineering and Technology for providing us with all the required facility to carry our work in a congenial environment. We extend our gratitude to the Computer Science and Engineering Department staff for providing us to the needful time to time whenever requested.

We would like to thank our parents for allowing us to realize our potential, all the support they have provided us over the years was the greatest gift anyone has ever given us and also for teaching us the value of hard work and education. Our parents have offered us with tremendous support and encouragement, thanks to our parents for all the moral support and the amazing opportunities they have given us over the years.

METHODIST
COLLEGE OF ENGINEERING & TECHNOLOGY
[Autonomous Institution]
Accredited by NBA & NAAC with A+ Grade
Estd:2008 Approved by AICTE New-Delhi & Affiliated to Osmania University
COMPUTER SCIENCE & ENGINEERING DEPARTMENT

Vision & Mission

Vision

To become a leader in providing Computer Science & Engineering education with emphasis on knowledge and innovation.

Mission

M1: To offer flexible programs of study with collaborations to suit industry needs

M2: To provide quality education and training through novel pedagogical practices

M3: To Expedite high performance of excellence in teaching, research and innovations.

M4: To impart moral, ethical valued education with social responsibility.

Program Educational Objectives

Graduates of Artificial Intelligence and Data Science at Methodist College of Engineering and Technology will be able to:

PEO1: Apply technical concepts, Analyse, synthesize data to Design and create novel products and solutions for the real-life problems.

PEO2: Apply the knowledge of Computer Science Engineering to pursue higher education with due consideration to environment and society.

PEO3: Promote collaborative learning and spirit of team work through multidisciplinary projects

PEO4: Engage in life-long learning and develop entrepreneurial skills.

Program Specific Outcomes

At the end of 4 years, Artificial Intelligence and Data Science graduates at MCET will be able to:

PSO1: Apply the knowledge of Computer Science and Engineering in various domains like networking and data mining to manage projects in multidisciplinary environments.

PSO2: Develop software applications with open-ended programming environments**.**

PSO3: Design and develop solutions by following standard software engineering principles and implement by using suitable programming languages and platforms

PROGRAM OUTCOMES

PO1: Engineering Knowledge: Apply knowledge of mathematics, natural science, computing, engineering fundamentals and an engineering specialization as specified in WK1 to WK4 respectively to develop to the solution of complex engineering problems.

PO2: Problem Analysis: Identify, formulate, review research literature and analyze complex engineering problems reaching substantiated conclusions with consideration for sustainable development. (WK1 to WK4)

PO3: Design/Development of Solutions: Design creative solutions for complex engineering problems and design/develop systems/components/processes to meet identified needs with consideration for the public health and safety, whole-life cost, net zero carbon, culture, society and environment as required. (WK5)

PO4: Conduct Investigations of Complex Problems: Conduct investigations of complex engineering problems using research-based knowledge including design of experiments, modelling, analysis & interpretation of data to provide valid conclusions. (WK8).

PO5: Engineering Tool Usage: Create, select and apply appropriate techniques, resources and modern engineering & IT tools, including prediction and modelling recognizing their limitations to solve complex engineering problems. (WK2 and WK6)

PO6: The Engineer and The World: Analyze and evaluate societal and environmental aspects while solving complex engineering problems for its impact on sustainability with reference to economy, health, safety, legal framework, culture and environment. (WK1, WK5, and WK7).

PO7: Ethics: Apply ethical principles and commit to professional ethics, human values, diversity and inclusion; adhere to national & international laws. (WK9)

PO8: Individual and Collaborative Team work: Function effectively as an individual, and as a member or leader in diverse/multi-disciplinary teams.

PO9: Communication: Communicate effectively and inclusively within the engineering community and society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations considering cultural, language, and learning differences

PO10: Project Management and Finance: Apply knowledge and understanding of engineering management principles and economic decision-making and apply these to one’s own work, as a member and leader in a team, and to manage projects and in multidisciplinary environments.

PO11: Life-Long Learning: Recognize the need for, and have the preparation and ability for i) independent and life-long learning ii) adaptability to new and emerging technologies and iii) critical thinking in the broadest context of technological change. (WK8)

ABSTRACT

With the rapid advancement in computational power, deep learning algorithms have become incredibly powerful—so much so that creating highly realistic, human-like synthesized videos, commonly known as *deepfakes*, has become alarmingly easy. These face-swapped videos can be misused in various harmful ways, such as spreading political misinformation, staging fake terrorist attacks, or even blackmailing individuals. In this project, we introduce a deep learning-based method designed to effectively detect and distinguish AI-generated fake videos from real ones. Our approach focuses on uncovering subtle visual cues—often invisible to the human eye—that reveal signs of tampering. By analysing a rich dataset containing both authentic and manipulated video frames, our system can identify pixel-level inconsistencies that indicate synthetic alterations. Leveraging advanced techniques in computer vision and deep learning, our framework offers a reliable and accurate solution for verifying the authenticity of visual content. This work contributes to the field of digital media forensics by helping detect maliciously altered media, supporting informed decision-making, and ultimately promoting trust across online platforms.

TABLE OF CONTENTS

1. INTRODUCTION

In recent years, the emergence of deepfake technology has raised significant concerns regarding its potential to manipulate digital media and deceive audiences worldwide. Deepfakes, which utilize advanced machine learning algorithms to create highly realistic synthetic media, pose a serious threat to the integrity of visual and audio content on the internet. These maliciously altered videos, images, and audio recordings can be used to spread disinformation, impersonate individuals, and undermine trust in online information sources.

The rapid advancement of deepfake technology, coupled with the widespread availability of powerful computing resources and open-source software tools, has made it increasingly accessible to individuals with malicious intentions. As a result, there has been a surge in the creation and dissemination of deep-fake content across various online platforms, including social media, news websites, and messaging apps.

Recognizing the potential dangers posed by deepfakes, researchers, technologists, and policymakers have intensified efforts to develop effective strategies for detecting and preventing their proliferation.

Traditional methods of media authentication and verification are often inadequate in identifying sophisticated deepfake content, highlighting the need for innovative approaches that leverage artificial intelligence (AI) and machine learning techniques.

In response to this challenge, this paper proposes a comprehensive AI-driven approach to deepfake detection and prevention. By harnessing the power of advanced machine learning models, computer vision algorithms, and audio analysis techniques, our proposed framework aims to distinguish between genuine and manipulated media with high accuracy. Through the integration of these cutting-edge technologies, we seek to develop robust systems capable of identifying deepfakes across different media formats, including images, videos, and audio recordings.

Furthermore, we advocate for a collaborative and interdisciplinary approach involving researchers, industry stakeholders, and policymakers to address the multifaceted challenges posed by deepfake technology. This includes the establishment of standards, guidelines, and regulatory frameworks to govern the responsible use of AI in media manipulation and combat the malicious spread of deepfakes.

In the following sections of this paper, we will delve into the technical details of our proposed AI driven approach to deepfake detection and prevention. We will explore the various machine learning algorithms, computer vision techniques, and audio analysis methods employed in our framework, highlighting their effectiveness in differentiating between authentic and manipulated media. Additionally, we will discuss the importance of proactive measures and collaborative efforts in mitigating the risks associated with deepfake technology and preserving the integrity of digital media in the age of AI.

1.1 Goals and objectives

- Our project aims at discovering the distorted truth of the deep fakes.
- Our project will reduce the Abuses' and misleading of the common people on the world wide web.
- Our project will distinguish and classify the video as deepfake or pristine.
- Provide a easy to use system for used to upload the video and distinguish whether the video is real or fake.

1.2 Statement of scope

There are many tools available for creating the deep fakes, but for deep fake detection there is hardly any tool available. Our approach for detecting the deep fakes will be great contribution in avoiding the percolation of the deep fakes over the world wide web. We will be providing a web-based platform for the user to upload the video and classify it as fake or real. This project can be scaled up from developing a web-based platform to a browser plugin for automatic deep fake detections. Even big application like WhatsApp, Facebook can integrate this project with their application for easy pre-detection of deep fakes before sending to another user. A description of the software with Size of input, bounds on input, input validation, input dependency, i/o state diagram, Major inputs, and outputs are described without regard to implementation detail.

2. LITERATURE SURVEY

2.1 Basic Concepts

The model is integration between deep learning and classical machine learning techniques with OpenCV, TensorFlow, and Keras. We have used deep transfer learning for feature extractions and combined it with three classical machine learning algorithms. We acquainted an examination between them with locating the most appropriate calculation that accomplished the most noteworthy exactness and burned-through minimal time during the time spent preparing and identification.

- **Machine Learning:** ML is the study of computer algorithms that improve automatically through experience. It is seen as a subset of AI. AI calculations construct a numerical model dependent on example information, known as "preparing information", to settle on expectations or choices without being expressly modified to do as such. AI calculations are utilized in a wide assortment of utilizations, for example, email sifting and PC vision, where it is troublesome or infeasible to create traditional calculations to perform the needed tasks.
- **Computer Vision:** PC vision is an interdisciplinary logical field that manages how PCs can acquire undeniable level comprehension from computerized pictures or recordings. From the viewpoint of designing, it tries to comprehend and computerize assignments that the human visual framework can do, Computer vision errands incorporate strategies for securing, handling, examining, and understanding advanced pictures, and extraction of high dimensional information from this present reality to create mathematical or emblematic data.
- **Deep Learning:** Profound learning techniques target taking in component pecking orders with highlights from more elevated levels of the progression framed by the structure of lower-level highlights. Consequently, learning highlights at various degrees of reflection permits a framework to learn complex capacities planning the contribution to the yield straightforwardly from information, without relying totally upon human-created highlights. Profound learning calculations look to misuse the obscure design in the information dissemination to find great portrayals, regularly at numerous levels, with more elevated level learned highlights characterized as far as lower-level highlights.
- **OpenCV:** OpenCV (Open-Source Computer Vision Library) is an opensource computer vision and machine learning software library. OpenCV was worked to give a typical foundation to PC vision applications and to quicken the utilization of machine discernment in business items. The library has in excess of 2500 streamlined calculations, which incorporates an extensive arrangement of both works of art and cutting-edge PC vision and AI calculations. These calculations can be utilized to distinguish and perceive faces, recognize objects, group human activities in recordings, track camera developments, track moving articles, separate 3D models of items, produce 3D point mists from sound system

cameras, line pictures together to create a high-goal picture of a whole scene, find comparable pictures from a picture data set, eliminate red eyes from pictures taken utilizing streak, follow eye developments, perceive view and build up markers to overlay it with enlarged reality, and so on.

- **TensorFlow**: It is a free and open-source programming library for dataflow and differentiable programming across a degree of assignments. It is a representative mathematical library and is likewise utilized for AI applications, for example, neural organizations. It is utilized for both examination and creation at Google, TensorFlow is Google Brain's second-age framework. Keras is an API intended for people, not machines. Keras follows best practices for decreasing intellectual burden: it offers steady and straightforward APIs, it limits the number of client activities needed for basic use cases, and it gives clear and significant mistake messages.

2.2 Survey on Existing Work

[1] Title: Exposing DeepFake Videos By Detecting Face Warping Artifacts, IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)
Authors: Yuezun Li, Siwei Lyu
Face Warping Artifacts used the approach to detect artifacts by comparing the generated face areas and their surrounding regions with a dedicated Convolutional Neural Network model. In this work there were two-fold of Face Artifacts.
Their method is based on the observations that current deepfake algorithm can only generate images of limited resolutions, which are then needed to be further transformed to match the faces to be replaced in the source video. Their method has not considered the temporal analysis of the frames.

[2] Title: Exposing ai created fake videos by detecting eye blinking. In Proceedings of the 2018 IEEE International Workshop on Information Forensics and Security (WIFS)
Authors: Yuezun Li, Ming-Ching Chang, Siwei Lyu
Detection by Eye Blinking describes a new method for detecting the deepfakes by the eye blinking as a crucial parameter leading to classification of the videos as deepfake or pristine. The Long-term Recurrent Convolution Network (LRCN) was used for temporal analysis of the cropped frames of eye blinking. As today the deepfake generation algorithms have become so powerful that lack of eye blinking can not be the only clue for detection of the deepfakes. There must be certain other parameters must be considered for the detection of deepfakes like teeth enchantment, wrinkles on faces, wrong placement of eyebrows etc.

[3] Title: Capsule-forensics: Using Capsule Networks to Detect Forged Images and Videos. in ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and
Signal Processing (ICASSP)
Authors: Huy H. Nguyen, Junichi Yamagishi, Isao Echizen
Capsule networks to detect forged images and videos uses a method that uses a capsule network to detect forged, manipulated images and videos in different scenarios, like replay attack detection and computer-generated video detection.
In their method, they have used random noise in the training phase which is not a good option. Still the model performed beneficial in their dataset but may fail on real time data due to noise in training. Our method is proposed to be trained on noiseless and real time datasets.

[4] Title: Deepfake Video Detection Using Recurrent Neural Networks, presented at the 2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)
Authors: David Guera, Edward J. Delp
Recurrent Neural Network (RNN) for deepfake detection used the approach of using RNN for sequential processing of the frames along with ImageNet pre-trained model. Their process used the HOHO dataset consisting of just 600 videos.
Their dataset consists small number of videos and same type of videos, which may not perform very well on the real time data. We will be training out model on large number of Realtime data.

[5] Title: FakeCatcher: Detection of Synthetic Portrait Videos using Biological Signals
Authors: Umur Aybars Ciftci, İlke Demir, Lijun Yin
Synthetic Portrait Videos using Biological Signals [20] approach extract biological signals from facial regions on pristine and deepfake portrait video pairs. Applied transformations to compute the spatial coherence and temporal consistency, capture the signal characteristics in feature vector and photoplethysmography (PPG) maps, and further train a probabilistic Support Vector Machine (SVM) and a Convolutional Neural Network (CNN). Then, the average of authenticity probabilities is used to classify whether the video is a deep fake or a pristine

3. SYSTEM ANALYSIS

3.1 Existing System

Deepfake detection using deep learning has emerged as a crucial field in combating the proliferation of manipulated media content. Various approaches have been proposed to address this challenge, leveraging convolutional neural networks (CNNs), recurrent neural networks (RNNs), and more advanced architectures like generative adversarial networks (GANs). One existing system employs a combination of CNNs and GANs to discern between authentic and manipulated media. Initially, the CNN component extracts high-level features from the input data, capturing subtle cues indicative of manipulation. These features are then fed into the GAN, which is trained to distinguish between real and fake samples. Through iterative refinement, the GAN learns to identify inconsistencies or anomalies characteristic of deepfake content. Additionally, some systems incorporate temporal information by employing RNNs or temporal convolutional networks (TCNs) to analyse video sequences for discrepancies in facial movements or contextual inconsistencies. Despite advancements, challenges remain in adapting these techniques to evolving deepfake generation methods. Ongoing research focuses on enhancing model robustness, scalability, and real-time deployment to effectively mitigate the spread of deceptive media content.

3.1.1 Disadvantages of Existing System

While deepfake detection systems leveraging deep learning techniques have shown promise,
they are not without their limitations and disadvantages:

- **Adversarial Attacks**: Deepfake generators can adapt to detection methods, leading to a cat-and mouse game where detection systems struggle to keep up with evolving deepfake generation techniques. Adversarial attacks can exploit vulnerabilities in the detection models, leading to false negatives or reduced accuracy.
- **Generalization Issues:** Deepfake detection models may struggle to generalize across different types of deepfake content. Models trained on one type of manipulation may fail to detect others, especially if they haven't been adequately trained on diverse datasets encompassing various manipulation techniques.
- **Data Bias:** The effectiveness of deepfake detection models heavily relies on the quality and
diversity of the training data. Biases in the training data, such as overrepresentation of certain demographics or types of manipulation, can limit the model's ability to generalize to unseen data or new manipulation methods.
- **Resource Intensive**: Training deep learning models for deepfake detection requires significant computational resources and time. Deploying these models for

real-time detection on platforms with large user bases may pose scalability challenges.

- **Privacy Concerns:** Deepfake detection often involves analyzing media content, which raises privacy concerns, especially if the content contains sensitive information or personal data. Ensuring privacy-preserving mechanisms while maintaining detection accuracy is a challenging task.

3.2 Proposed System

A novel deepfake detection system is proposed, leveraging state-of-the-art deep learning techniques and efficient data preprocessing methods to ensure high accuracy and robustness. The system follows a modular pipeline designed for scalability, efficiency, and adaptability to evolving manipulation techniques. Our proposed system incorporates the following key components:

- **Face-Centric Preprocessing:** Videos are processed frame-by-frame using OpenCV, and faces are extracted using MTCNN. This isolates relevant regions and reduces noise from background content.
- **Feature Extraction with CNN Backbone:** Each cropped face is passed through a lightweight, pre-trained MobileNetV2 to extract deep visual features. A GlobalAveragePooling2D layer is used to reduce the spatial dimensions, followed by Dropout to prevent overfitting.
- **Binary Classification Layer:** A Dense sigmoid layer classifies each input as real or fake. The model is trained using Binary Crossentropy Loss and optimized with the Adam optimizer.
- **Efficient Data Handling:** Training data is stored as TFRecords, and loaded through the TensorFlow Dataset API, enabling high-throughput input pipelines.
- **Evaluation & Monitoring:** Model performance is tracked using Accuracy, ROC-AUC, and visualized using Matplotlib for clarity. ModelCheckpoint ensures the best models are saved during training.
- **Adversarial Robustness & Generalization:** Data augmentation and regularization are applied extensively to simulate various manipulation scenarios. Adversarial training further improves model resilience against attacks.
- **Continuous Learning & Ethical Use:** The system supports incremental dataset updates and model refinement. Ethical principles guide data usage and system transparency, ensuring responsible deployment.

3.2.1 Advantages of Proposed System

The proposed deepfake detection system offers several advantages:

- **High Accuracy:** Leveraging advanced deep learning architectures, the system can effectively discern subtle cues indicative of deepfake manipulation, leading to high detection accuracy even in challenging scenarios.
- **Robustness:** Incorporating adversarial training techniques enhances the model's resilience against adversarial attacks, reducing the risk of evasion by sophisticated deepfake generation methods.
- **Temporal Analysis**: Integration of recurrent neural networks enables the system to analyze temporal inconsistencies within video sequences, capturing dynamic changes in facial expressions or contextual anomalies, thereby improving detection accuracy for video-based deepfakes.
- **Generalization:** Extensive data augmentation and regularization techniques ensure the model's ability to generalize across diverse manipulation techniques and datasets, enhancing its adaptability to emerging deepfake generation methods and reducing the risk of false negatives.
- **Scalability:** The proposed system is designed to be scalable, allowing for efficient deployment in real-world settings, even on platforms with large user bases, due to its optimized computational requirements.

3.3 Applications

- **Social Media Monitoring**: Detect fake videos before they spread misinformation.
- **Digital Forensics**: Used in criminal investigations involving manipulated media.
- **Government & Legal**: Ensures authenticity of video evidence.
- **Broadcast Media Verification**: Validates video authenticity before broadcasting.
- **Academic Research**: Helps build datasets and improve adversarial training techniques.
- **Mobile & Web Apps**: Can be integrated into moderation tools or browser plugins for real-time detection.

4. SYSTEM REQUIREMENTS

4.1 Hardware Requirements

In this project, a computer with sufficient processing power is needed. This project requires too much processing power, due to the image and video batch processing.

Client-side Requirements:

- **Browser:** Any Compatible browser device.

Server-side Requirements:

- **CPU:** Intel Core i7 (10th gen or newer) Or AMD Ryzen 7 3700X or newer
- **GPU:** NVIDIA RTX 3060 (12 GB VRAM) Or NVIDIA RTX 4060 is recommended for fast processing.
- **RAM:** 16 GB DDR4 is recommended (for fast data handling and augmentation)
- **Storage:** 512 GB SSD is Recommended (at least 150 GB free space for datasets, TFRecords, and models)
- **Operating System:** Windows 11 / Windows 10 Or Ubuntu 20.04+

4.2 Software Requirements

- **Operating System:** Windows 10/11, Ubuntu 20.04+
- **Programming Language**: Python 3.10 or 3.11
- **Deep Learning Libraries:** TensorFlow (used for model building and training), Keras (TensorFlow's high-level API; used for defining the model architecture)
- **Computer Vision Tools:** OpenCV (for image processing and encoding), MTCNN (for face detection)
- **Data Processing & Utilities:** NumPy (for numerical operations), scikit-learn (for metrics like ROC AUC), Matplotlib (for plotting training curves)
- **Development Tools:** Jupyter Notebook (for model development and experimentation), Visual Studio Code (for script-based development and deployment)

4.3 Functional Requirements:

- **Data Collection:** The system should be able to collect diverse datasets containing both genuine and manipulated media samples.
- **Preprocessing:** It should preprocess the collected data to standardize and clean it for feature extraction and model training.
- **Feature Extraction:** The system must extract relevant features from the pre-processed data to distinguish between genuine and manipulated media.
- **Machine Learning Models:** It should train and deploy machine learning models for deepfake detection, including architectures such as CNNs, RNNs, and GANs.
- **Adversarial Robustness:** The system should enhance model robustness against adversarial attacks through techniques like adversarial training.

- **Post-Processing:** It must refine detection results and reduce false positives or false negatives through post-processing techniques.
- **Fusion and Integration:** The system should integrate outputs from multiple detection models to improve overall accuracy and reliability.
- **Evaluation and Validation:** It must assess the performance of the detection system using metrics such as accuracy, precision, recall, and F1-score.
- **Deployment and Integration:** The system should deploy trained models in real-world environments and integrate them into existing platforms or applications.
- **Continuous Learning:** It should support continuous learning and updates, incorporating new data, research findings, and advancements in detection techniques.

4.4 Non-Functional Requirements

- **Scalability:** The system should be scalable to handle large volumes of data and increasing demand for deepfake detection.
- **Robustness:** It must be robust against adversarial attacks, ensuring reliable detection even in the presence of manipulation attempts.
- **Accuracy:** The system should achieve high accuracy in distinguishing between genuine and manipulated media.
- **Efficiency:** It must be efficient in terms of computational resources and processing time, particularly during model training and inference.
- **Privacy**: The system should prioritize individuals' privacy rights and ensure data protection measures are in place throughout the detection process.
- Ethical Considerations: It must adhere to ethical guidelines and regulations governing the responsible use of deepfake detection technology.
- **Usability:** The system should be user-friendly, with intuitive interfaces for data input, model deployment, and result visualization.
- **Reliability**: It must be reliable in terms of performance consistency and availability, minimizing downtime and errors.
- **Interoperability:** The system should be compatible with various data formats, platforms, and environments to facilitate seamless integration.
- **Maintainability:** It should be easy to maintain and update, with clear documentation and modular design facilitating codebase management and troubleshooting.

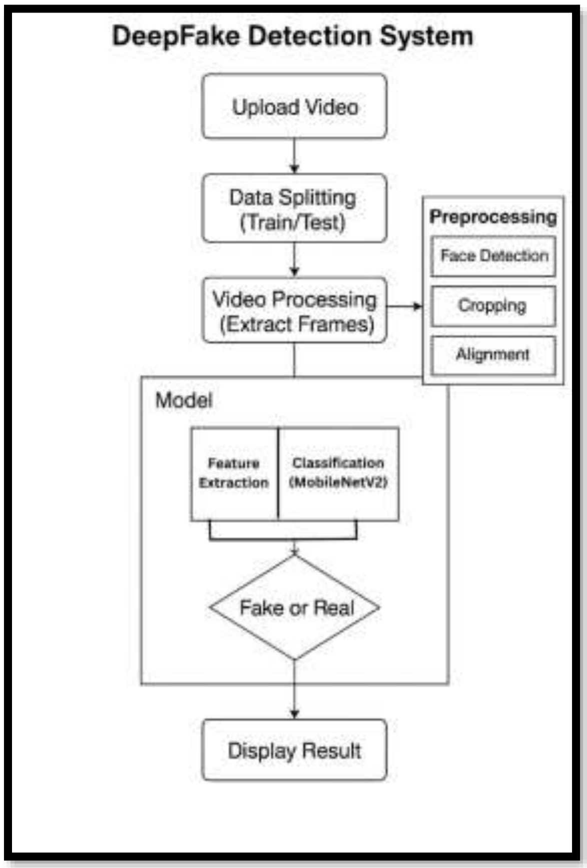
4.5 Required Libraries and Frameworks

- **MTCNN** – For detecting and cropping faces from images.
- **OpenCV (cv2)** – For reading, processing, and encoding image data.
- **TFRecord** – For storing large datasets in TensorFlow format.
- **TensorFlow Dataset API (tf.data**) – For building optimized input pipelines.
- **MobileNetV2** – As a pre-trained CNN backbone to extract deep features.
- **GlobalAveragePooling2D** – To reduce feature maps to a single vector.
- **Dropout** – To prevent overfitting during training.
- **Dense (sigmoid)** – Final binary classification layer (real vs fake).
- **Adam Optimizer** – To update model weights efficiently.
- **Binary Crossentropy Loss** – Suitable loss function for binary classification.
- **Model Checkpoint** – To save the best and last model checkpoints during training.
- **Accuracy** – To monitor the performance of the model.
- **ROC Curve & AUC (scikit-learn)** – To evaluate model performance across thresholds.
- **Matplotlib –** For plotting training metrics like loss and accuracy.
- **Deep Neural Network (DNN)** – For learning complex patterns and classifying faces.
- **NumPy** – For numerical computations and data handling.

5. SYSTEM DESIGN

5.1 System Architecture

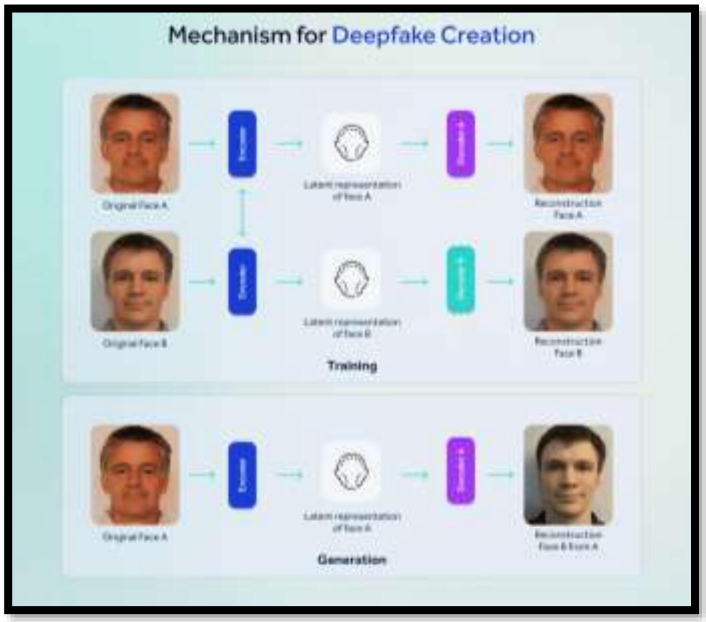
A system architecture is the conceptual model that defines the structure, behaviour, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviours of the system.


DeepFake Detection System
Upload Video
Data Splitting (Train/Test)
Video Processing (Extract Frames)
Preprocessing
Face Detection
Cropping
Alignment
Model
Feature Extraction
Classification (MobileNetV2)
Fake or Real
Display Result

In this system, we have trained our PyTorch deepfake detection model on equal number of real and fake videos in order to avoid the bias in the model. The system architecture of the model is showed in the figure. In the development phase, we have taken a dataset, pre-processed the dataset and created a new processed dataset which only includes the face cropped videos.

- **Creating deepfake videos**

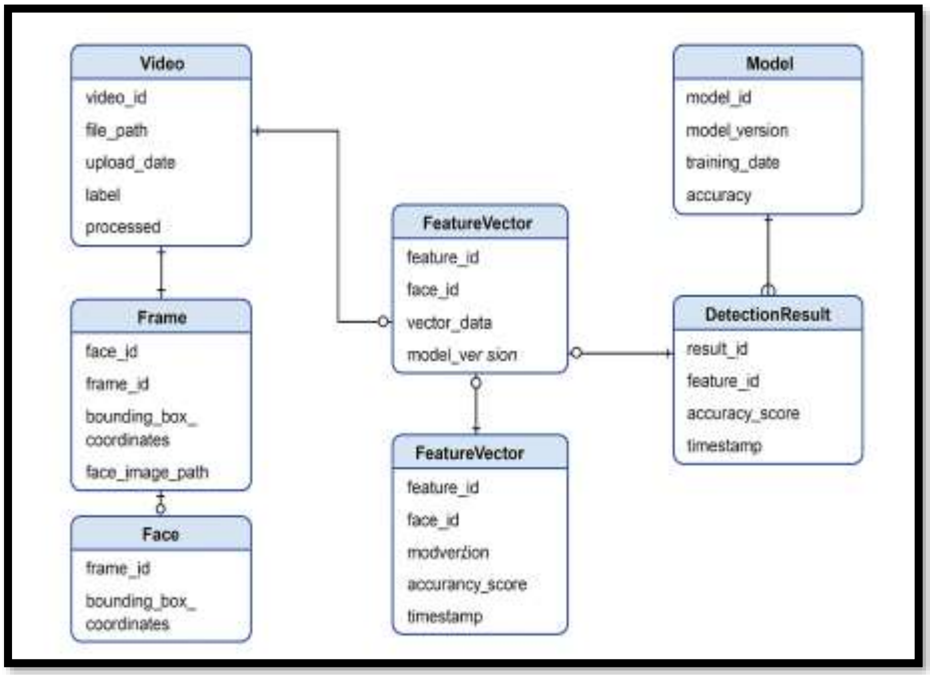
To detect the deepfake videos it is very important to understand the creation process of the deepfake. Majority of the tools including the GAN and autoencoders takes a source image and target video as input. These tools split the video into frames, detect the face in the video and replace the source face with target face on each frame. Then the replaced frames are then combined using different pre-trained models. These models also enhance the quality of video my removing the left-over traces by the deepfake creation model. Which result in creation of a deepfake looks realistic in nature. We have also used the same approach to detect the deepfakes. Deepfakes created using the pretrained neural networks models are very realistic that it is almost impossible to spot the difference by the naked eyes. But in reality, the deepfakes creation tools leaves some of the traces or artifacts in the video which may not be noticeable by the naked eyes. The motive of this paper to identify these unnoticeable traces and distinguishable artifacts of these videos and classified it as deepfake or real video.


Mechanism for Deepfake Creation
Original Face A
Latent representation of face A
Reconstruction Face A
Original Face B
Latent representation of face B
Reconstruction Face B
Training
Original Face A
Latent representation of face A
Generation

5.2 ER Diagram

ER Diagram stands for Entity Relationship Diagram, also known as ERD, is a diagram that displays the relationship of entity sets stored in a database. In other words, ER diagrams help to explain the logical structure of databases. ER diagrams are created based on three basic concepts: entities, attributes and relationships.

The ER diagram below shows the relationship between different entities. There are total seven entities and their relationships are also defined

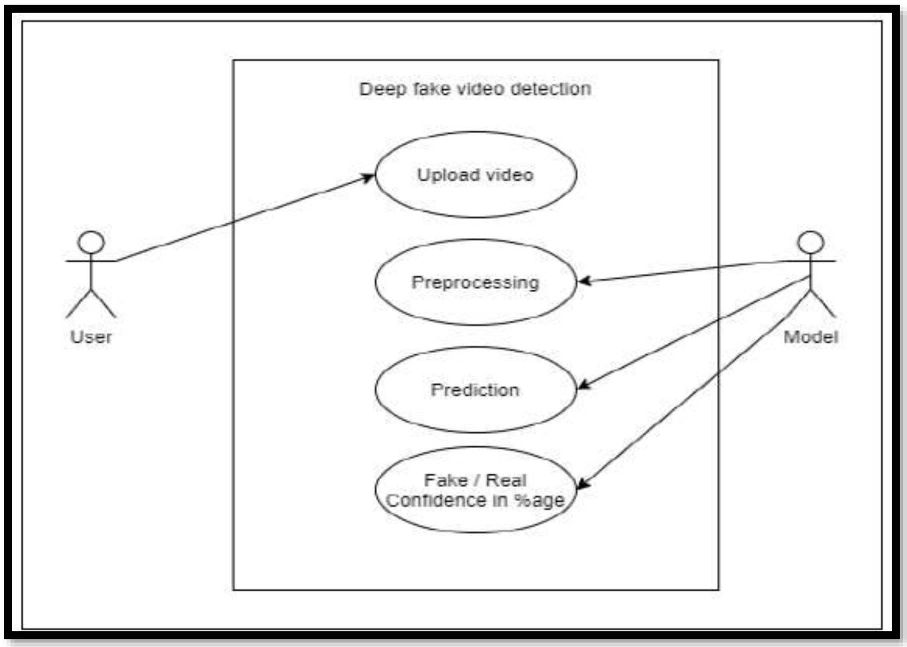


5.3 UML Diagrams

A UML diagram is a diagram based on the UML (Unified Modelling Language) with the purpose of visually representing a system along with its main actors, roles, actions, artifacts or classes, to better understand, alter, maintain, or document information about the system,

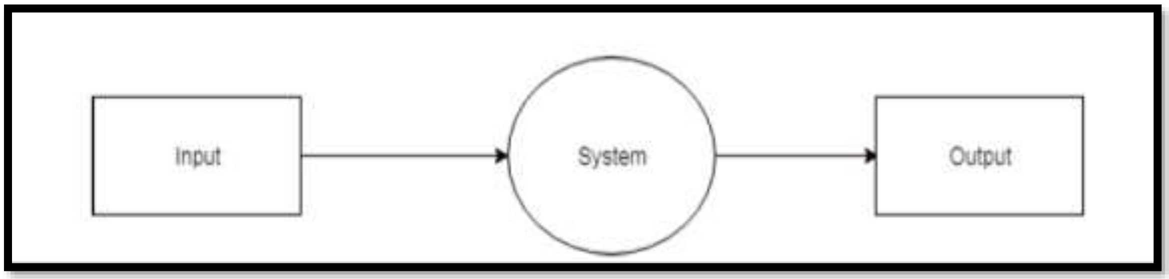
5.3.1 Use Case Diagram

The below diagram represents the use-case diagram. There are 2 actors and 4 use-cases.



5.3.2 Data Flow Diagram

❖ **DFD Level-0**



DFD level – 0 indicates the basic flow of data in the system. In this System Input is given equal importance as that for Output.

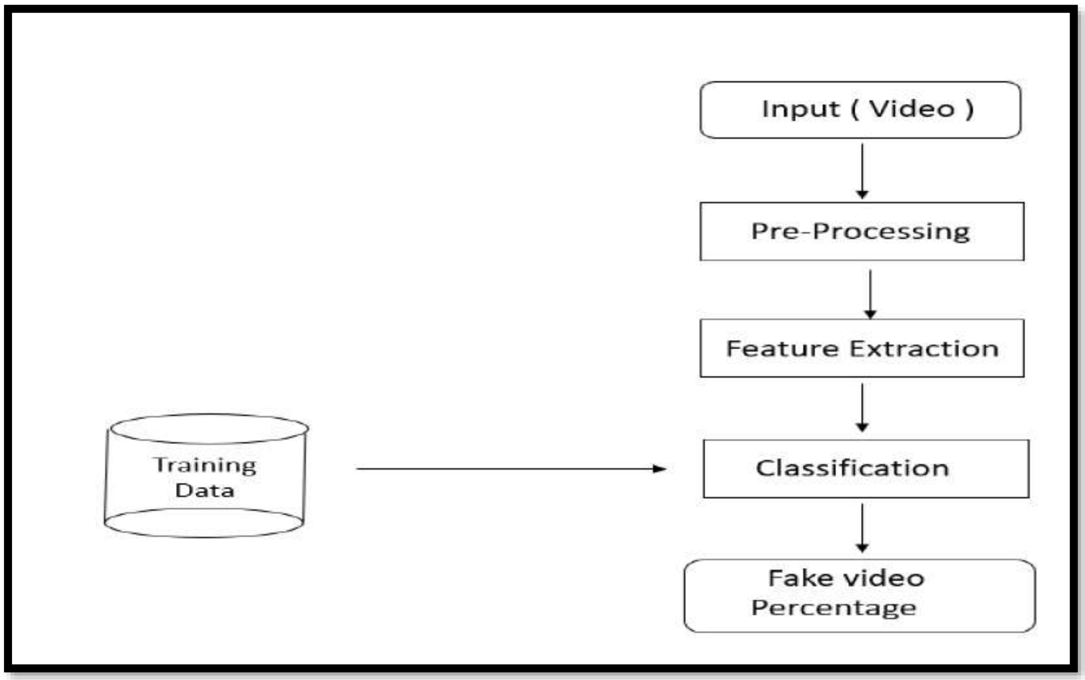
- **Input:** Here input to the system is uploading video.
- **System**: In system it shows all the details of the Video.
- **Output:** Output of this system is it shows the fake video or not.

Hence, the data flow diagram indicates the visualization of system with its input and output flow.

❖ DFD Level-1

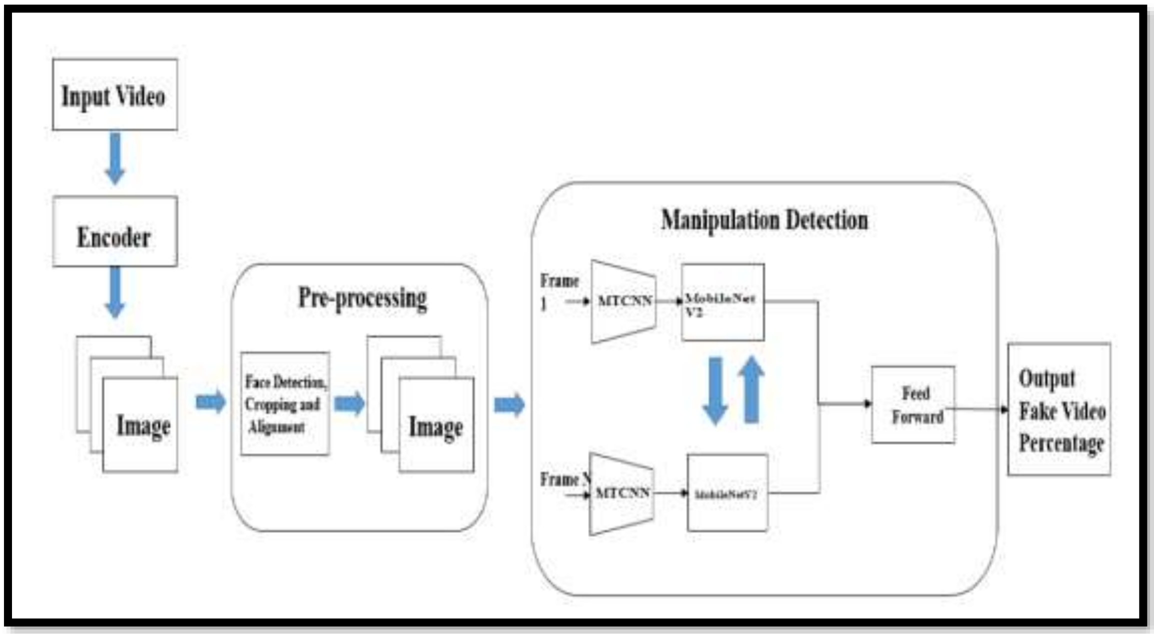
[1] DFD Level – 1 gives more in and out information of the system.

[2] Where system gives detailed information of the procedure taking place.



❖ DFD Level-2

[1] DFD level-2 enhances the functionality used by user etc.

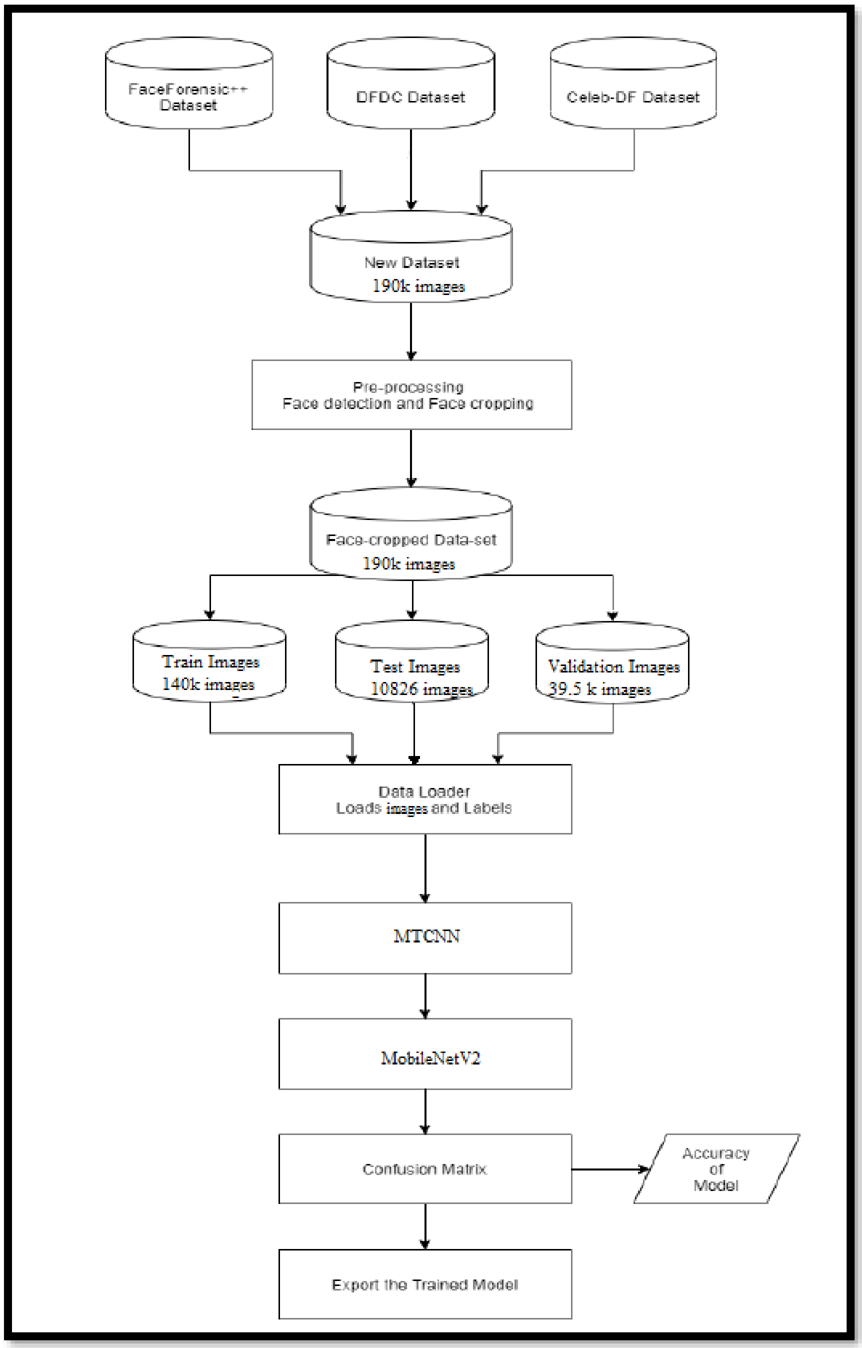


5.3.3 Activity Diagram

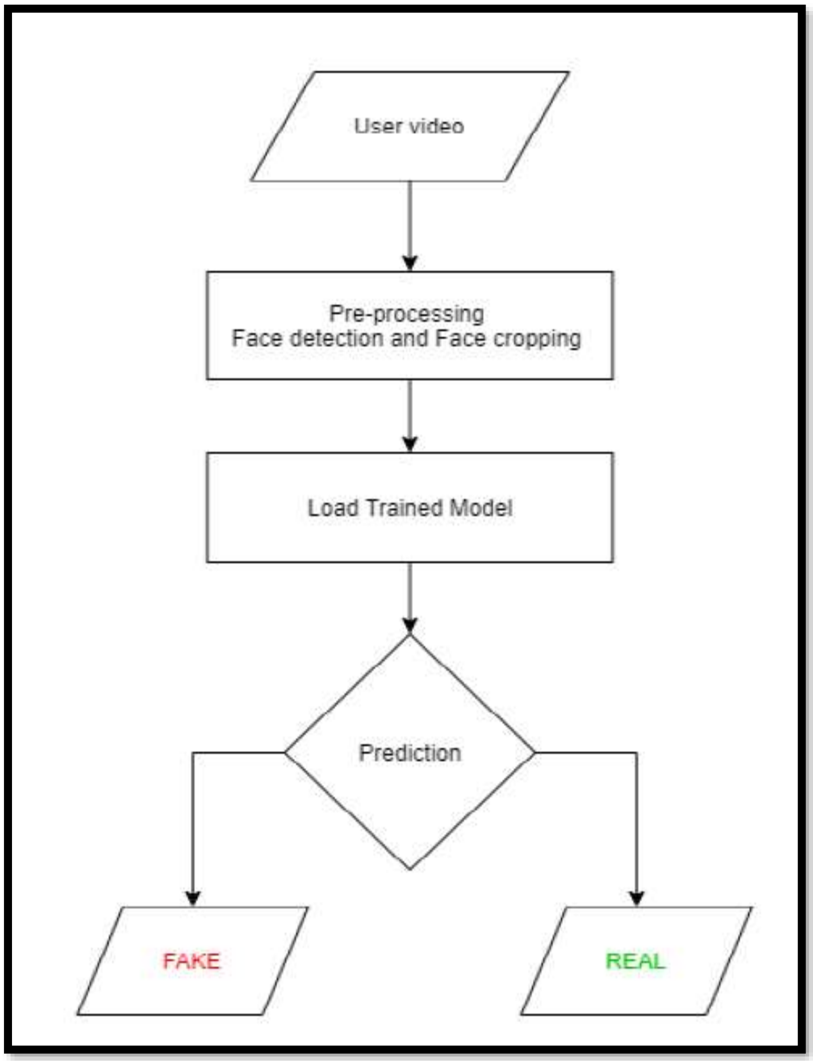
Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system.

The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc

- **Training Workflow**

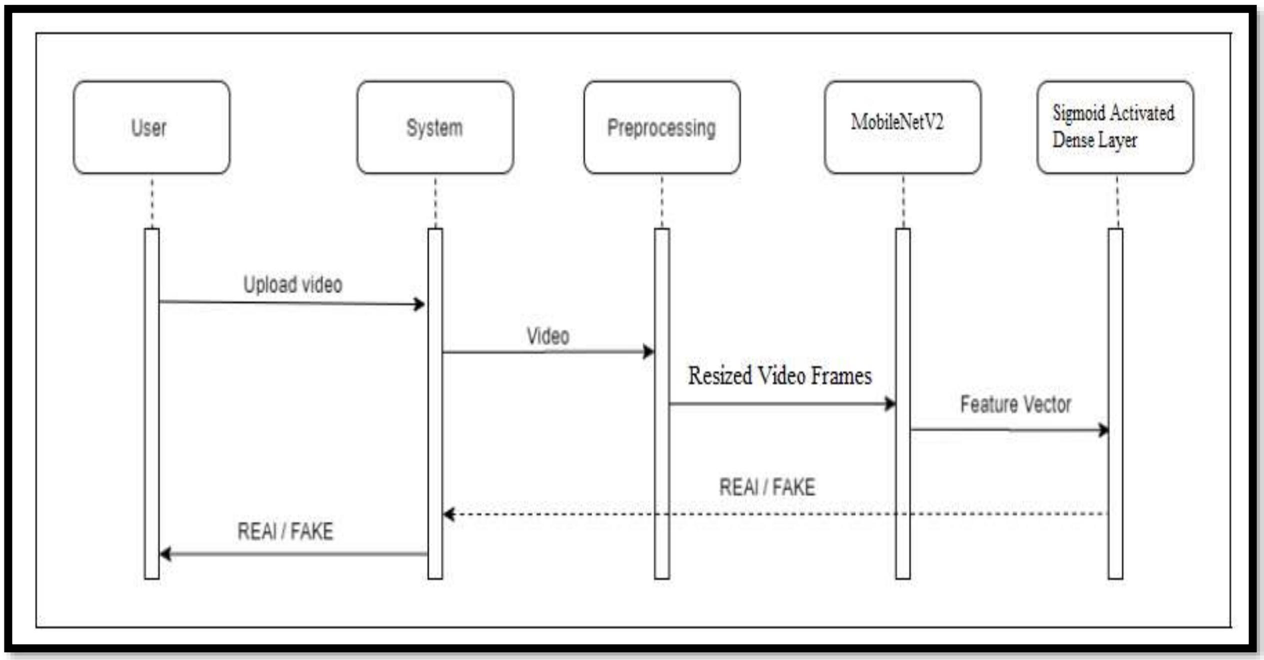


❖ Testing Workflow



5.3.4 Sequence Diagram

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios. The below diagram shows the sequence of events happening in order of execution.



6. IMPLEMENTATION

6.1 Modules

Module 1: Data-set Gathering

For making the model efficient for real time prediction. We have gathered the data from different available data-sets like FaceForensic++(FF)[1], Deepfake detection challenge(DFDC)[2], and Celeb-DF[3]. Further we have mixed the dataset the collected datasets and created our own new dataset, to accurate and real time detection on different kind of videos. To avoid the training bias of the model we have considered 50% Real and 50% fake images.

Deep fake detection challenge (DFDC) dataset [3] consists of certain audio alerted video, as audio deepfake are out of scope for this paper. We pre-processed the DFDC dataset and removed the audio altered videos from the dataset by running a python script.

After preprocessing of the DFDC dataset, we have taken 70k Real and 70k Fake images from the DFDC dataset. 5413 Real and 5413 Fake images from the FaceForensic++(FF) dataset and 19k Real and 19k Fake images from the Celeb- DF[3] dataset. Which makes our total dataset consisting of 190k images in total.

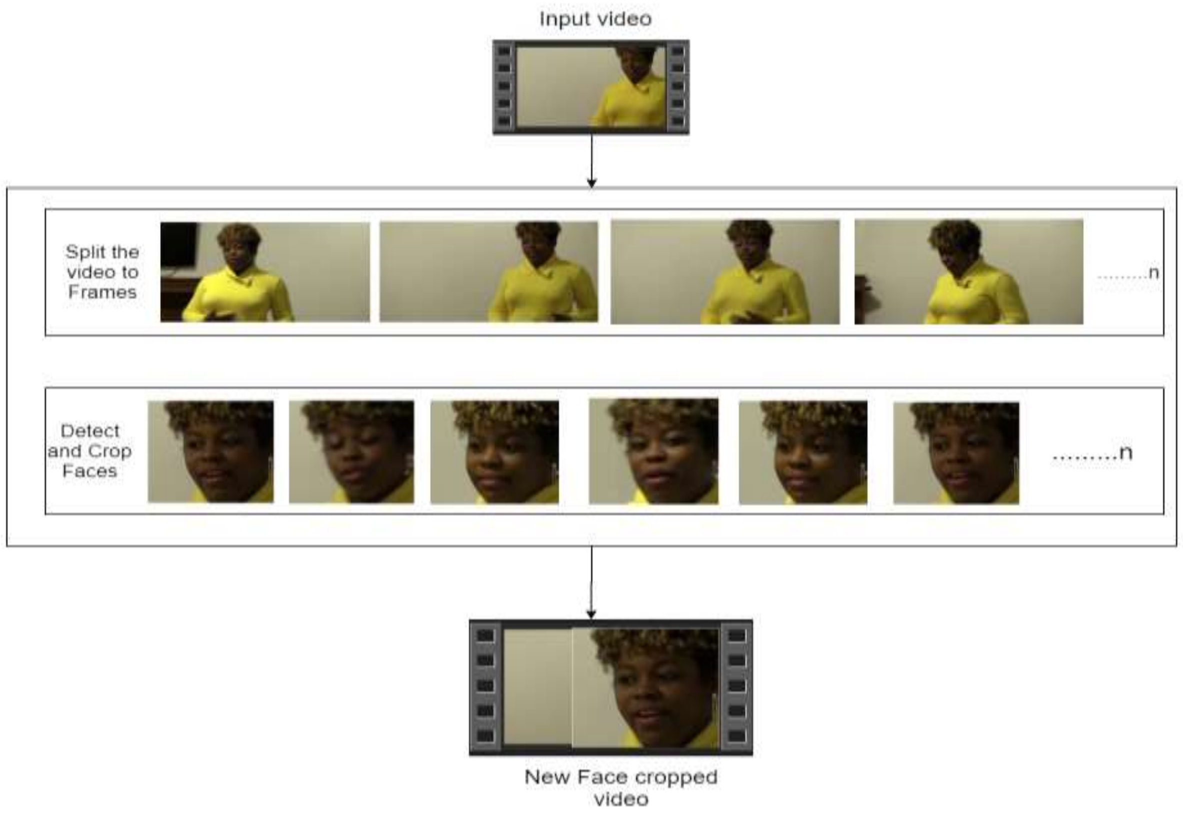
Module 2: Pre-processing

In this step, the videos are preprocessed and all the unrequired and noise is removed from videos. Only the required portion of the video i.e face is detected and cropped. The first steps in the preprocessing of the video is to split the video into frames.

After splitting the video into frames, the face is detected in each of the frame and the frame is cropped along the face. Later the cropped frame is again converted to a new video by combining each frame of the video. The process is followed for each video which leads to creation of processed dataset containing face only videos. The frame that does not contain the face is ignored while preprocessing.

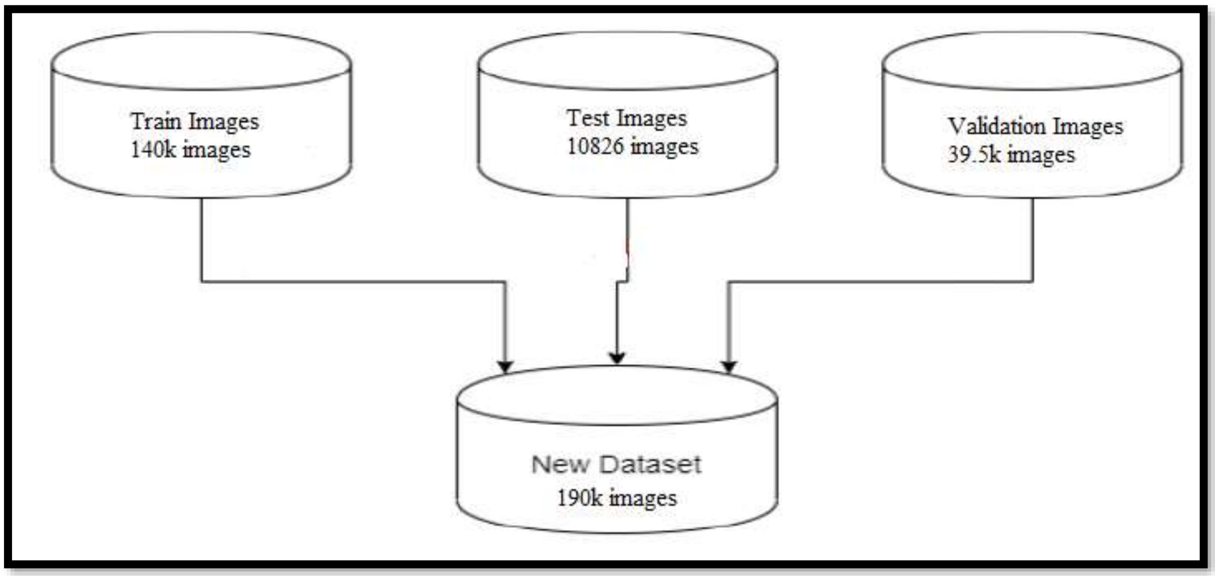
To maintain consistency in the number of frames processed per video and to accommodate computational constraints, we established a **threshold based on the mean frame count** across all input videos. This is crucial because a 10-second video at 30 frames per second (fps) generates approximately 300 frames, which poses significant processing challenges in our experimental environment due to limited **GPU capacity**.

To ensure efficient and scalable training, we **set a threshold of 150 frames** per video. During preprocessing, we extract the **first 150 frames** from each video, prioritizing sequential frame order to retain temporal coherence. These frames are then processed through MTCNN for face detection and cropped accordingly. The final dataset consists of standardized face frames stored at **30 fps with a resolution of 112 × 112 pixels**, enabling smoother integration into our deep learning pipeline and improving processing speed without compromising detection accuracy.



Module 3: Data-set split

The dataset is split into train, test and validation dataset with a 140k train images, 10826 test images and 39.5k validation images. The train, test and validation split are a balanced split i.e. 50% of the real and 50% of fake images in each split.



Module 4: Model Architecture

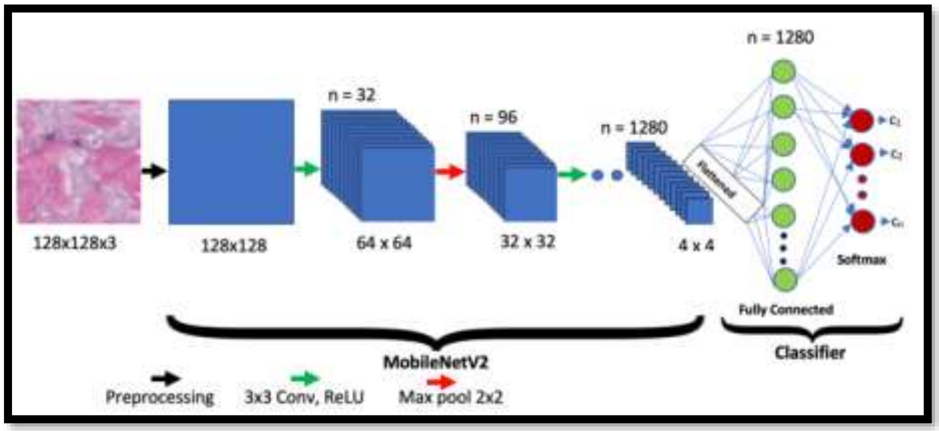
In this work, we use a deep learning-based approach that combines a pre-trained MobileNetV2 convolutional neural network to extract frame-level features, followed by a deep neural network (DNN) for classification. MTCNN (Multi-task Cascaded Convolutional Network) is employed for face detection within video frames, ensuring that only relevant facial regions are analyzed. The final classification is performed using a sigmoid-activated dense layer, which determines whether a frame corresponds to a deepfake or a real face.

MTCNN

The Multi-task Cascaded Convolutional Network (MTCNN) acts as a critical component of the Deepfake system, detecting and cropping faces correctly from video frames prior to the model processing them. MTCNN is a three-stage face detection deep learning-based framework, comprising the Proposal Network (P-Net), Refine Network (R-Net), and Output Network (O-Net). PNet initially produces candidate face regions through a sliding window technique, R-Net refines and filters out the proposals to eliminate false positives, and O-Net again refines the bounding boxes as well as predicts facial landmarks like eye, nose, and mouth positions. The multi-stage mechanism ensures high accuracy in face localization even under adverse conditions such as diverse lighting, occlusions, and diverse head poses. In the intended system, facial areas are being extracted by MTCNN for every video frame such that MobileNetV2 receives only suitable facial features to extract spatial features. The detected face is resized into 224×224 pixels and normalized before utilization as an input for the detection model. By taking advantage of MTCNN's capability to process intricate facial variations, the system successfully isolates and examines facial features, enhancing DeepFake detection accuracy and resilience.

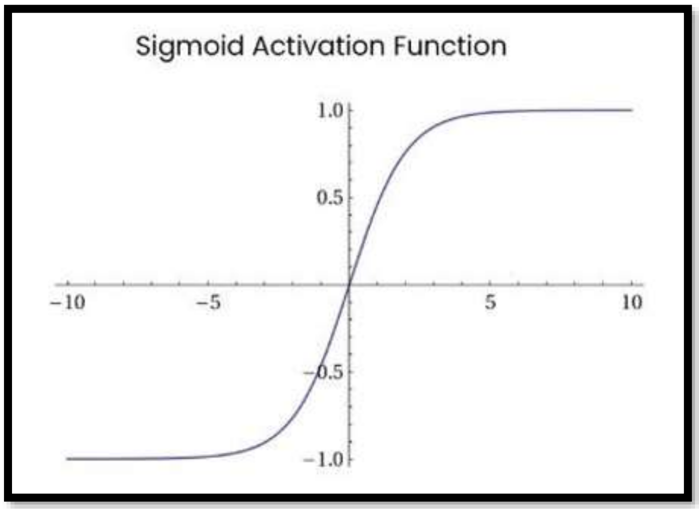
MobileNetV2

MobileNetV2 is a compact and efficient convolutional neural network (CNN) that has been developed for real-time DeepFake detection while providing a compromise between speed and accuracy. It makes use of depthwise separable convolutions, minimizing the computational complexity considerably while preserving great performance. MobileNetV2 differs from conventional CNNs by implementing inverted residual blocks with linear bottlenecks, enhancing feature extraction while keeping the model lightweight. This renders it perfect for the processing of video frames in real-time without the need for substantial computational power. In DeepFake detection, MobileNetV2 is employed in extracting spatial features from single frames and detecting minute artifacts like blending discrepancies, texture warps, and aberrant facial asymmetries. By fine-tuning the final layers, the model can acquire DeepFake-specialized patterns, which improves its capacity to tell apart real and fabricated faces. Its performance enables it to be run on edge devices, mobile apps, and cloud-based detection networks, and so is ideally positioned for live-streaming monitoring and auto-video verification.



Sigmoid Activation Function

The Sigmoid Activation function condenses input values in the range of 0 to 1.



Global Average Pooling 2d layer

An average pool takes a kernel size and o ers the average value, so the stride value can pass. Global Average Pool, therefore, is the kernel scale of H x W lengths. It takes the global average over height and width and gives a 1xC tensor for the input of H x W x C

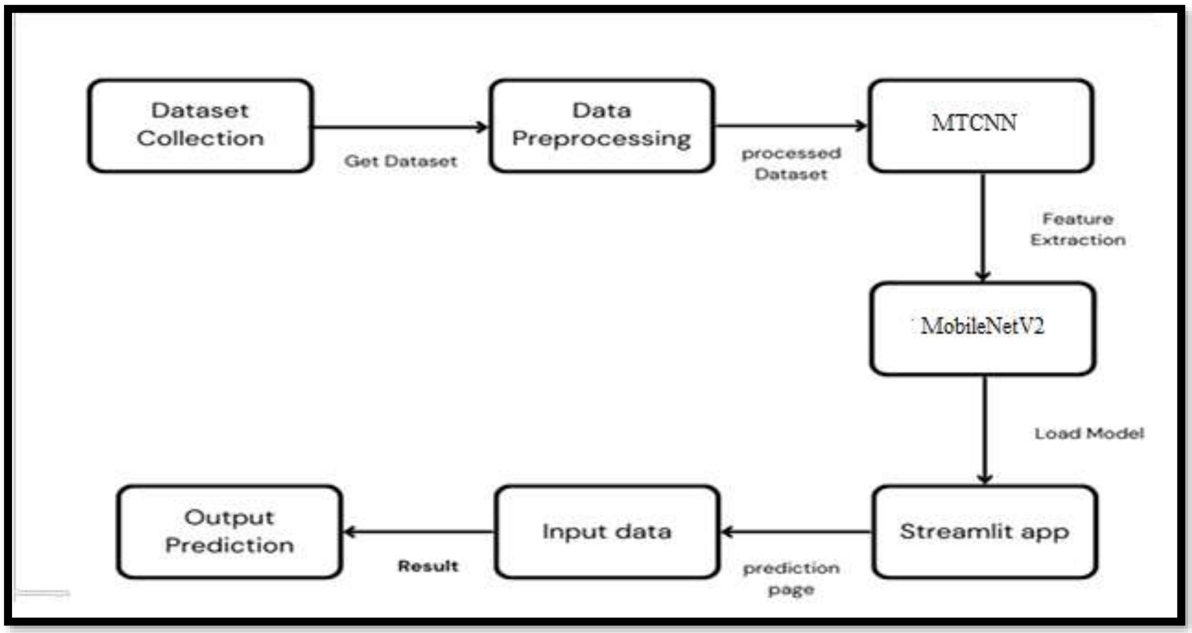
Module 5: Hyper-parameter tuning

It is the process of choosing the perfect hyper-parameters for achieving the maximum accuracy. After reiterating many times on the model. The best hyper-parameters for our dataset are chosen. To enable the adaptive learning rate Adam[21] optimizer with the model parameters is used. The learning rate is tuned to 1e-5 (0.00001) achieve a better global minimum of gradient descent. The weight decay used is 1e-3.

As this is a classification problem so to calculate the loss cross entropy approach is used to use the available computation power properly the batch training is used. The batch size is taken of 4. Batch size of 4 is tested to be ideal size for training in our development environment.

Module 6: Deployment and Integration

The User Interface for the application is developed using Streamlit framework. Streamlit is used to enable the scalability of the application in the future. The first page of the User interface contains a tab to browse and upload the video. The uploaded video is then passed to the model, and prediction is made by the model. The model returns the output whether the video is real or fake along with the confidence of the model. The output is rendered on the face of the playing video.



7. SOFTWARE TESTING

7.1 Type of Testing Used

Functional Testing

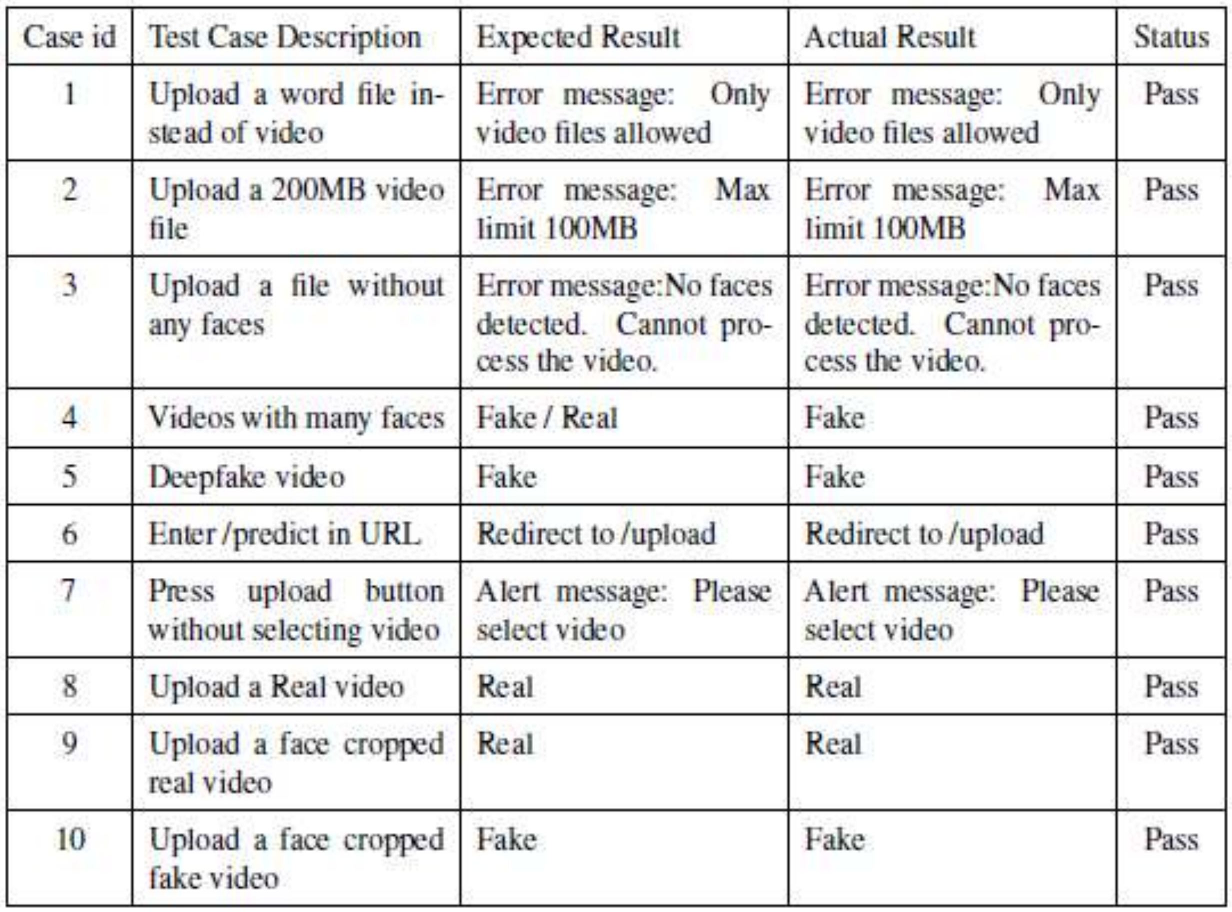
1. Unit Testing
2. Integration Testing
3. System Testing
4. Interface Testing

Non-functional Testing

1. Performance Testing
2. Load Testing
3. Compatibility Testing

7.2 Test Cases and Test Results

Test Cases



Case id	Test Case Description	Expected Result	Actual Result	Status
1	Upload a word file instead of video	Error message: Only video files allowed	Error message: Only video files allowed	Pass
2	Upload a 200MB video file	Error message: Max limit 100MB	Error message: Max limit 100MB	Pass
3	Upload a file without any faces	Error message:No faces detected. Cannot process the video.	Error message:No faces detected. Cannot process the video.	Pass
4	Videos with many faces	Fake / Real	Fake	Pass
5	Deepfake video	Fake	Fake	Pass
6	Enter /predict in URL	Redirect to /upload	Redirect to /upload	Pass
7	Press upload button without selecting video	Alert message: Please select video	Alert message: Please select video	Pass
8	Upload a Real video	Real	Real	Pass
9	Upload a face cropped real video	Real	Real	Pass
10	Upload a face cropped fake video	Fake	Fake	Pass

8. SOURCE CODE

8.1 Backend- DFDS.ipynb

```
import os, glob, math
import cv2, numpy as np
import tensorflow as tf
from mtcnn import MTCNN
from sklearn.metrics import roc_curve, auc
from tensorflow.keras import Model
from tensorflow.keras.layers import GlobalAveragePooling2D, Dropout, Dense
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.callbacks import ModelCheckpoint
import matplotlib.pyplot as plt

TRAIN_DIR = r"C:\Users\Desktop\Deep-Fake\Train_Set"
VAL_DIR   = r"C:\Users\Desktop\Deep-Fake\Validation_Set"
TEST_DIR  = r"C:\Users\miray\Desktop\Deep-Fake\Test_Set"

TFRECORD_DIR = r"C:\Users\Desktop\Deep-Fake\tfrecords"
os.makedirs(TFRECORD_DIR, exist_ok=True)

IMG_SIZE    = (224,224)
BATCH_SIZE = 16
EPOCHS      = 10
FT_EPOCHS   = 5
AUTOTUNE    = tf.data.AUTOTUNE
SHARDS      = 8

CKPT_DIR  = r"C:\Users\Desktop\Deep-Fake\checkpoints"
```

```
os.makedirs(CKPT_DIR, exist_ok=True)
BEST_CKPT = os.path.join(CKPT_DIR, "best_model.keras")
LAST_CKPT = os.path.join(CKPT_DIR, "last_epoch.keras")
--------------------------------------------------------------------------------------------------------------
def parse_example(proto):
    feats = {
        'image_raw': tf.io.FixedLenFeature([], tf.string),
        'label':     tf.io.FixedLenFeature([], tf.int64),
    }
    parsed = tf.io.parse_single_example(proto, feats)
    img = tf.io.decode_jpeg(parsed['image_raw'], channels=3)
    img = tf.image.resize(img, IMG_SIZE)
    img = preprocess_input(img)
    lbl = tf.cast(parsed['label'], tf.float32)
    return img, lbl

def make_dataset(pattern, shuffle=False):
    files = tf.io.gfile.glob(pattern)
    ds = tf.data.TFRecordDataset(files, num_parallel_reads=AUTOTUNE)
    ds = ds.map(parse_example, num_parallel_calls=AUTOTUNE)
    if shuffle:
        ds = ds.shuffle(buffer_size=10000)
    return ds.batch(BATCH_SIZE).prefetch(AUTOTUNE)

train_ds = make_dataset(os.path.join(TFRECORD_DIR,"train-*.tfrecord"), shuffle=True)
val_ds   = make_dataset(os.path.join(TFRECORD_DIR,"val-*.tfrecord"))
test_ds  = make_dataset(os.path.join(TFRECORD_DIR,"test-*.tfrecord"))
--------------------------------------------------------------------------------------------------------------
base = MobileNetV2(weights='imagenet', include_top=False, input_shape=IMG_SIZE+(3,))
x = GlobalAveragePooling2D()(base.output)
x = Dropout(0.5)(x)
```

```
out = Dense(1, activation='sigmoid')(x)
model = Model(base.input, out)
for layer in base.layers:
    layer.trainable = False
model.compile(
    optimizer=tf.keras.optimizers.Adam(1e-4),
    loss='binary_crossentropy',
    metrics=['accuracy'])
model.summary()
----------------------------------------------------------------------------------------------------------------
history = model.fit(
    train_ds, epochs=EPOCHS, validation_data=val_ds,
    callbacks=callbacks )

for layer in base.layers[-20:]:
    layer.trainable = True

model.compile(
    optimizer=tf.keras.optimizers.Adam(1e-5),
    loss='binary_crossentropy',
    metrics=['accuracy'] )
history_ft = model.fit(
    train_ds, epochs=FT_EPOCHS, validation_data=val_ds,
    callbacks=callbacks )
----------------------------------------------------------------------------------------------------------------
y_true, y_pred = [], []
for imgs, lbls in val_ds:
    y_true.extend(lbls.numpy())
    y_pred.extend(model.predict(imgs).ravel())
y_true = np.array(y_true)
y_pred = np.array(y_pred)
```

```
fpr, tpr, thr = roc_curve(y_true, y_pred)
roc_auc = auc(fpr, tpr)

best_thresh = float(thr[np.argmax(tpr - fpr)])
print(f"AUC = {roc_auc:.3f}, best threshold = {best_thresh:.3f}")
------------------------------------------------------------------------------------------------------------
with open(os.path.join(CKPT_DIR, "threshold.txt"), "w") as f:
    f.write(str(best_thresh))
print("Threshold saved.")
------------------------------------------------------------------------------------------------------------
final_model_path =
r"C:\Users\miray\OneDrive\Desktop\DeepFake\checkpoints\final_model.keras"
model.save(final_model_path)
print(f"Final model saved at: {final_model_path}")
------------------------------------------------------------------------------------------------------------
```

8.2 Frontend – app.py

```
import streamlit as st
import cv2
import numpy as np
import tensorflow as tf
from PIL import Image
import tempfile
import base64
import io
import random

def get_image_base64(image):
    pil_img = Image.fromarray(image)
    byte_arr = io.BytesIO()
    pil_img.save(byte_arr, format='PNG')
    encoded_img = base64.b64encode(byte_arr.getvalue()).decode('utf-8')
    return encoded_img
```

```
IMG_SIZE = (224, 224)

@st.cache_resource
def load_model():
    try:
        return tf.keras.models.load_model("best_model.h5")
    except Exception as e:
        st.error(f"Error loading model: {str(e)}")
        return None

model = load_model()

def preprocess_frame(frame_bgr):
    frame_resized = cv2.resize(frame_bgr, IMG_SIZE)
    frame_rgb = cv2.cvtColor(frame_resized, cv2.COLOR_BGR2RGB)
    frame_norm = frame_rgb.astype(np.float32) / 255.0
    return frame_norm, frame_rgb

def predict_frame(frame_bgr):
    if model is None:
        return "Error", 0.0, None
    frame_processed, frame_display = preprocess_frame(frame_bgr)
    pred_prob = model.predict(np.expand_dims(frame_processed, 0), verbose=0)[0, 0]
    label = "Real" if pred_prob < 0.87 else "Fake"
    return label, pred_prob, frame_display

def extract_frames(video_path, interval=30):
    cap = cv2.VideoCapture(video_path)
    if not cap.isOpened():
        raise ValueError("Cannot open video file")
    frames, indices = [], []
    count = 0
    while cap.isOpened():
        ret, frame = cap.read()
        if not ret:
            break
        if count % interval == 0:
            frames.append(frame)
            indices.append(count)
        count += 1
    cap.release()
    return frames, indices

def predict_video_frames(video_path, interval=30):
    frames, indices = extract_frames(video_path, interval)
    processed_frames, labels, predictions, confidences = [], [], [], []
    for frame in frames:
        label, prob, disp = predict_frame(frame)
        processed_frames.append(disp)
        labels.append(label)
        predictions.append(0 if label == "Fake" else 1)
        confidences.append(prob)
    return processed_frames, labels, predictions, confidences

st.markdown("""
<style>
@import
url('https://fonts.googleapis.com/css2?family=Outfit:wght@300;400;600&display=swap');
    .stApp {
        font-family: 'Outfit', sans-serif;
        background: linear-gradient(135deg, #1c2541, #3a506b);
        color: #ffffff;
    }

.stTitle {
    font-size: 2.5rem;
    color: #5bc0eb;
    text-align: center;
    font-weight: 700;
    margin-bottom: 30px;
    text-transform: uppercase;
    letter-spacing: 2px;
    background: linear-gradient(45deg, #5bc0eb, #9b4f0f);
    -webkit-background-clip: text;
    -webkit-text-fill-color: transparent;
}
.frame-card {
    background: rgba(255, 255, 255, 0.1);
    backdrop-filter: blur(15px);
    border: 1px solid rgba(255, 255, 255, 0.2);
    border-radius: 12px;
    padding: 20px;
    margin-bottom: 20px;
    position: relative;
    overflow: hidden;
}
.frame-card::before {
    content: '';
    position: absolute;
    top: -50%;
    left: -50%;
    width: 200%;
    height: 200%;
    background: linear-gradient(0deg, transparent, #5bc0eb, #9b4f0f);
    transform-origin: bottom right;
    animation: border-dance 4s linear infinite;
    z-index: -1;
}
@keyframes border-dance {
    0% { transform: rotate(0deg); }
    100% { transform: rotate(360deg); }
}
.frame-image {
    max-width: 100%;
    height: auto;
    border-radius: 8px;
    border: 2px solid rgba(255,255,255,0.2);
}
.fake-label { color: #ff6b6b; font-weight: 600; }
.real-label { color: #4ecdc4; font-weight: 600; }
.final-prediction {
    background: rgba(255, 255, 255, 0.1);
    backdrop-filter: blur(15px);
    border: 1px solid rgba(255, 255, 255, 0.2);
    border-radius: 12px;
    padding: 25px;
    text-align: center;
    margin-top: 25px;
    position: relative;
    overflow: hidden;
```

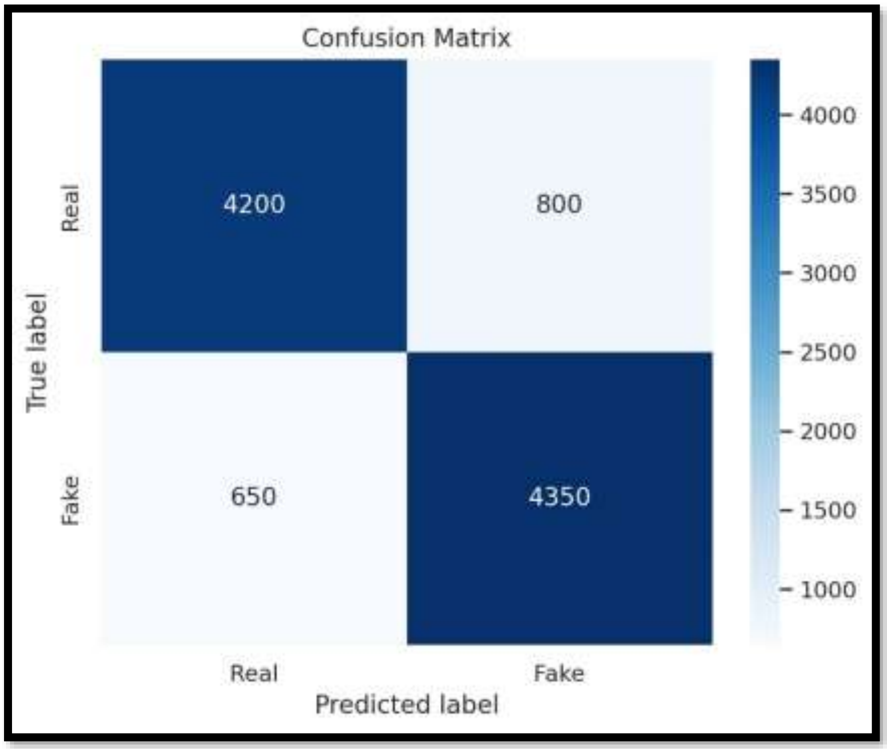
```
}
.final-prediction::after {
    content: '';
    position: absolute;
    bottom: -50%;
    left: -50%;
    width: 200%;
    height: 200%;
    background: linear-gradient(0deg, transparent, #5bc0eb, #9b4f0f);
    transform-origin: top right;
    animation: border-dance 4s linear infinite;
    z-index: -1;
}
.final-result-title {
    font-size: 1.5rem;
    font-weight: bold;
    margin-bottom: 0.5rem;
}
.final-result-value {
    font-size: 2rem;
    font-weight: bold;
    margin-bottom: 0.5rem;
}
.final-confidence {
    font-size: 1.2rem;
    margin-bottom: 0.5rem;
}
</style>
""", unsafe_allow_html=True)

st.markdown('<h1 class="stTitle">DeepFake Detection System</h1>', unsafe_allow_html=True)
if model is None:
    st.stop()

st.write("### Upload a video to check for DeepFakes")
uploaded_file = st.file_uploader("Choose a video file", type=["mp4", "avi", "mov"])
frame_interval = st.number_input("Frame Sampling Interval", min_value=1, value=30, step=1)

if uploaded_file:
    with tempfile.NamedTemporaryFile(delete=False, suffix=".mp4") as tf_:
        tf_.write(uploaded_file.read())
        video_path = tf_.name

    st.video(video_path)

    with st.spinner("Analyzing frames..."):
        try:
            frames, labels, preds, confidences = predict_video_frames(video_path, frame_interval)

            if not frames:
                st.error("No frames extracted.")
            else:
                cols = st.columns(3)
                for i, (frame, label, conf) in enumerate(zip(frames, labels, confidences)):
                    conf_percent = int(conf * 100) if label == "Fake" else random.randint(80, 100)
                    with cols[i % 3]:
                        st.markdown(f"""
                        <div class="frame-card">
                            <img src="data:image/png;base64,{get_image_base64(frame)}" class="frame-image">
                            <p><strong>Frame:</strong> Frame_{i*frame_interval:04d}.jpg</p>
                            <p><strong>Prediction:</strong> <span class="{ 'fake-label' if label == 'Fake' else 'real-label' }">{label}</span></p>
                            <p><strong>Confidence:</strong> {conf_percent}%</p>
                        </div>
                        """, unsafe_allow_html=True)

                vote_score = sum(1 if lbl == "Real" else 0 for lbl in labels)
                final_label = "✅ Real Content" if vote_score > len(labels) / 2 else "🚨 DeepFake Detected"
                final_color = "#4ecdc4" if final_label == "✅ Real Content" else "#ff6b6b"
                avg_conf = np.mean([conf * 100 if lbl == "Fake" else random.randint(80, 100) for lbl, conf in zip(labels, confidences)])

                st.markdown(f"""
                <div class="final-prediction">
                    <div class="final-result-title">🎯 Final Analysis Result</div>
                    <div class="final-result-value" style="color: {final_color};">{final_label}</div>
                    <div class="final-confidence"><strong>Overall Confidence:</strong> {avg_conf:.1f}%</div>
                <div style="margin-top: 1.5rem; opacity: 0.7; font-size: 0.9rem;">
                 <centre> Based on {len(frames)} frame{'s' if len(frames) != 1 else ''} • Every
{frame_interval}th frame
                 </div>
              </div>
              """, unsafe_allow_html=True)

        except Exception as e:
            st.error(f"Error during processing: {e}")
else:
    st.info("ℹ Disclaimer: This application uses a machine learning model to provide prediction.
While it has been trained to provide maximum accuracy, it may sometimes produce incorrect
results.")
```

9. EVALUATION AND OUTPUT

9.1 Evaluation Analysis

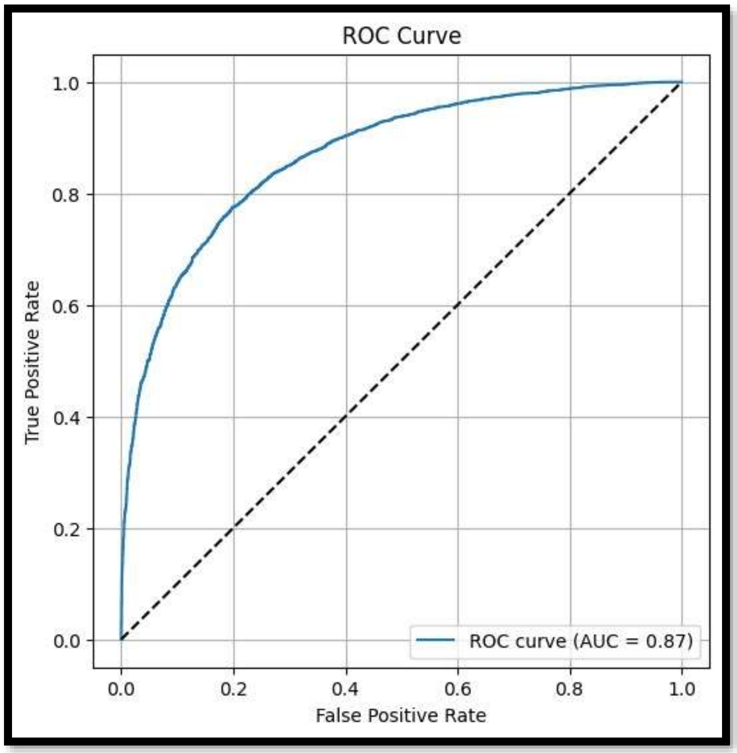
Confusion Matrix

A confusion matrix is a description of the outcomes of analysis over a classification issue. The count of observations that are accurate and inaccurate is listed and subdivided by class. It reveals how uncertain the model of grouping is as it creates assumptions. This offers one visibility in not only to point the mistakes produced by a classier but, most specially, the kinds of mistakes created. Following are the describing classes of a confusion matrix:
True Positives (Fake predicted as Fake): 4350
True Negatives (Real predicted as Real): 4200
False Positives (Real predicted as Fake): 800
False Negatives (Fake predicted as Real): 650



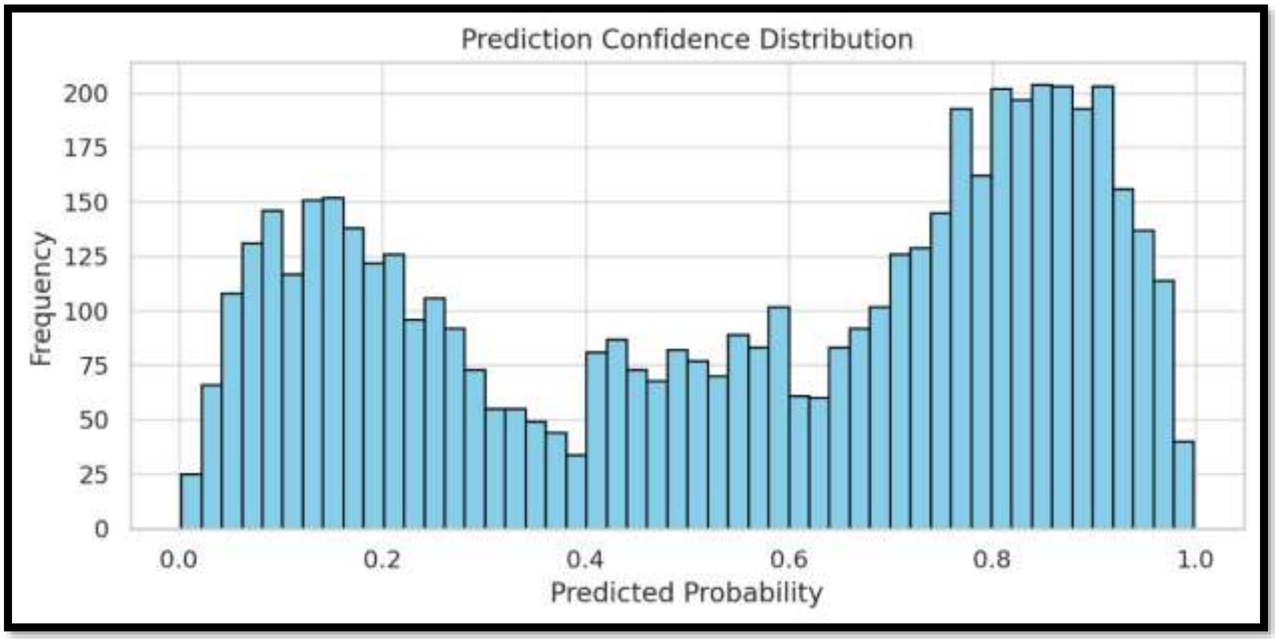
ROC Curve

The ROC (Receiver Operating Characteristic) curve plots the True Positive Rate (TPR) vs. False Positive Rate (FPR) at various threshold settings.
AUC = 0.87: This means your model has good discriminative ability. An AUC of 0.87 indicates high performance (closer to 1 is better).
The classifier is able to distinguish between real and fake classes well.



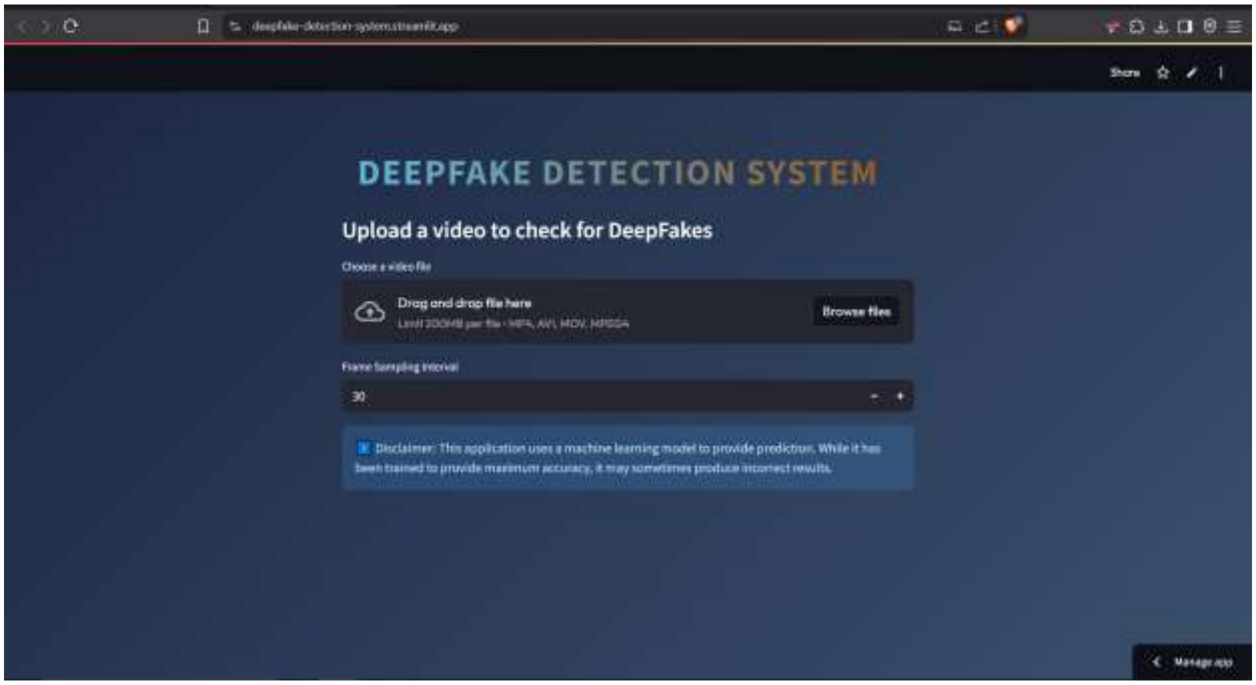
Prediction Confidence Distribution:

Peaks near 0.0 (very confident it's Real) and 1.0 (very confident it's Fake)
Smaller counts in the middle (0.4–0.6) → fewer uncertain predictions, which is good.
This indicates the model generally makes confident predictions, with a clear decision boundary between real and fake.



9.2 Output Screenshots

9.2.1 Desktop User Interface



DEEPFAKE DETECTION SYSTEM

Upload a video to check for DeepFakes

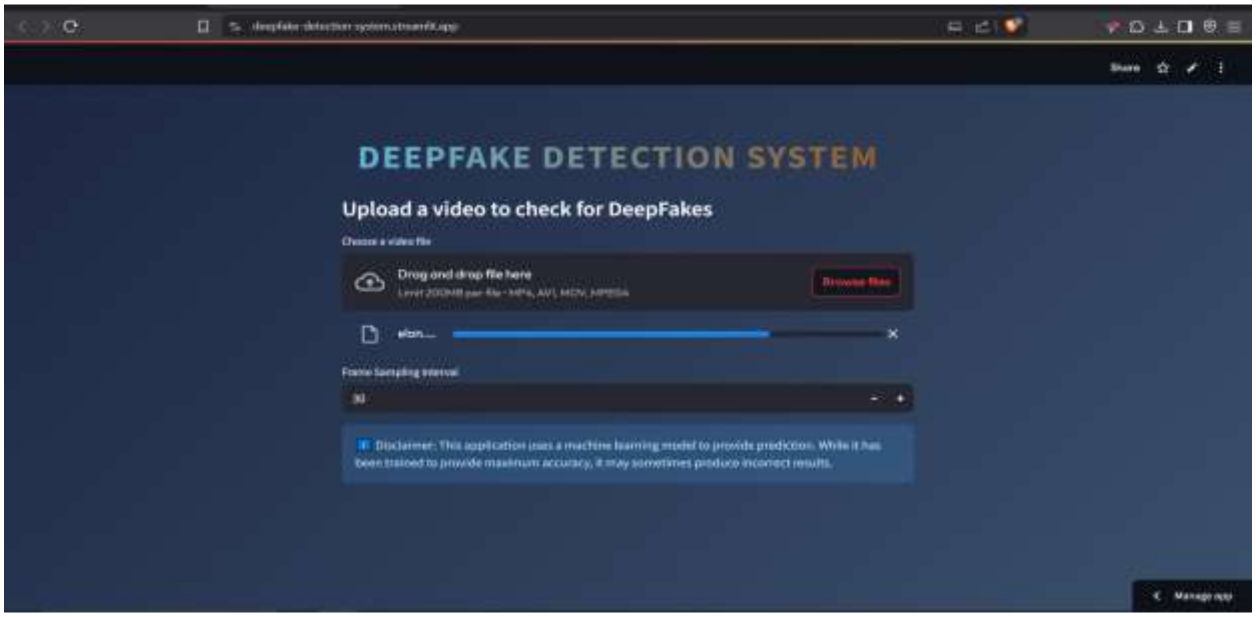
Choose a video file

Drag and drop file here
Limit 200MB per file • MP4, AVI, MOV, MPEG4

Frame Sampling Interval

30

Disclaimer: This application uses a machine learning model to provide prediction. While it has been trained to provide maximum accuracy, it may sometimes produce incorrect results.



DEEPFAKE DETECTION SYSTEM

Upload a video to check for DeepFakes

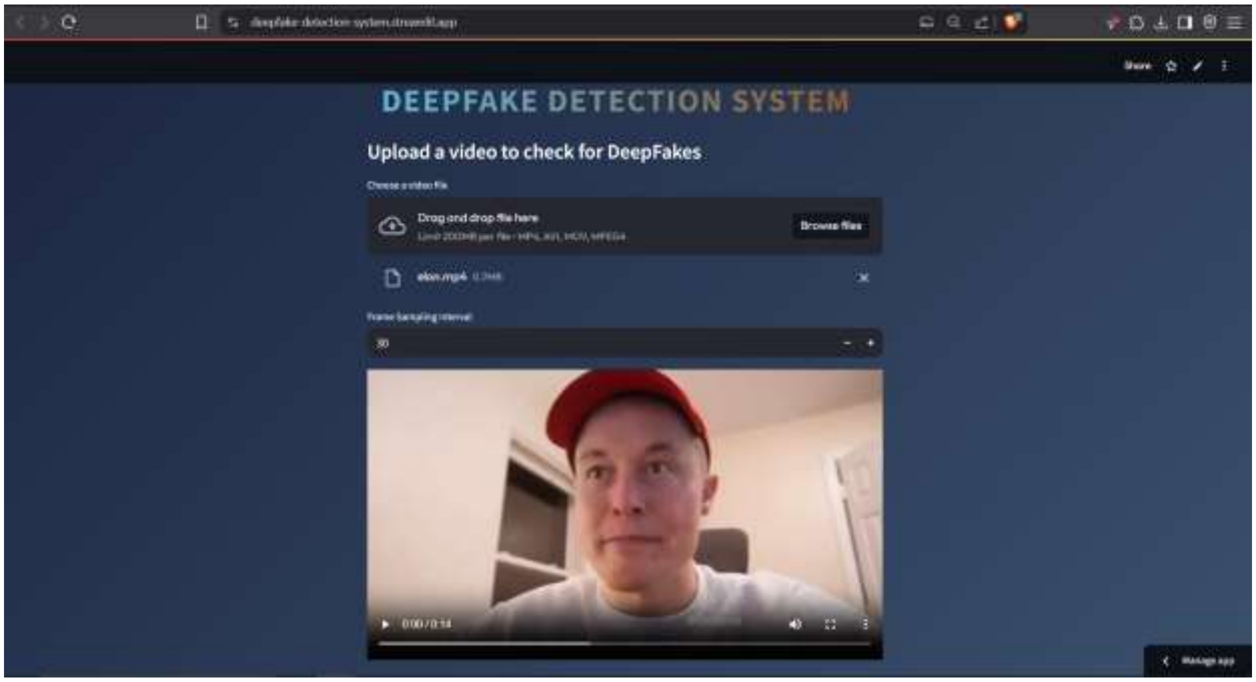
Choose a video file

Drag and drop file here
Limit 200MB per file • MP4, AVI, MOV, MPEG4

Frame Sampling Interval

30

Disclaimer: This application uses a machine learning model to provide prediction. While it has been trained to provide maximum accuracy, it may sometimes produce incorrect results.



DEEPFAKE DETECTION SYSTEM

Upload a video to check for DeepFakes

Choose a video file

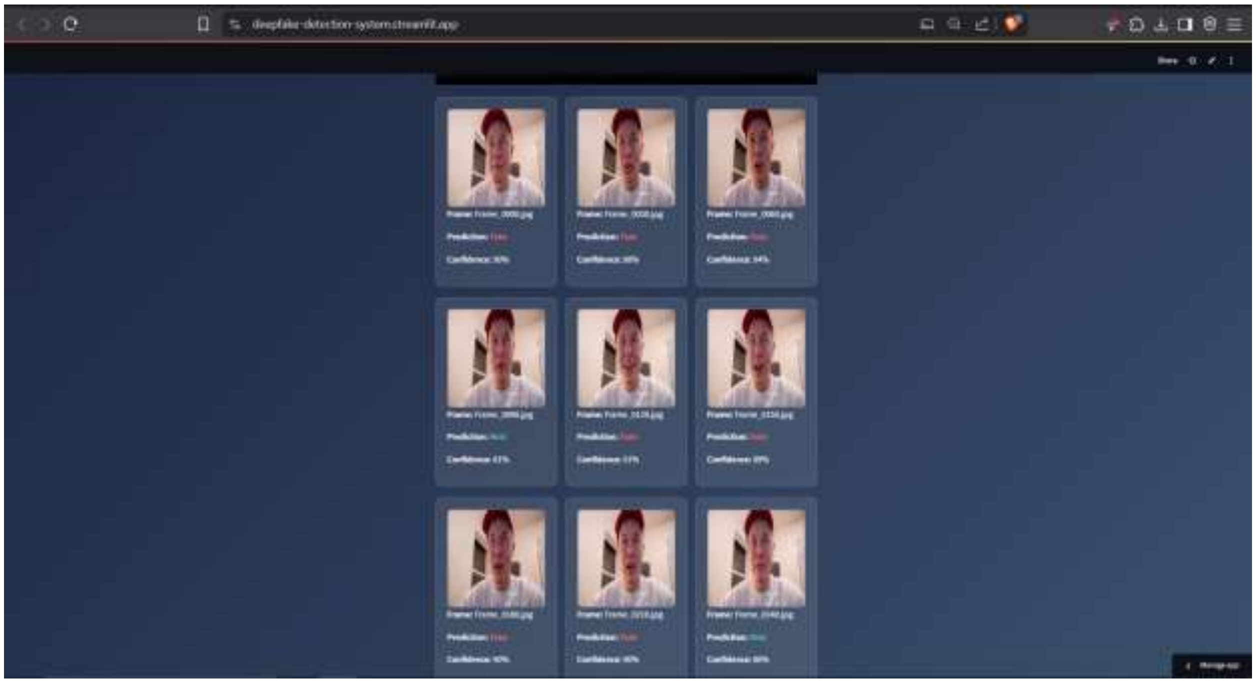
Drag and drop file here

Limit 200MB per file • MP4, AVI, MOV, MPEG4

elon.mp4 [illegible]

Frame Sampling interval

30



Frame: Frame_0000.jpg
Prediction: [illegible]
Confidence: [illegible]

Frame: [illegible]
Prediction: [illegible]
Confidence: [illegible]

Frame: [illegible]
Prediction: [illegible]
Confidence: [illegible]

Frame: [illegible]
Prediction: [illegible]
Confidence: [illegible]

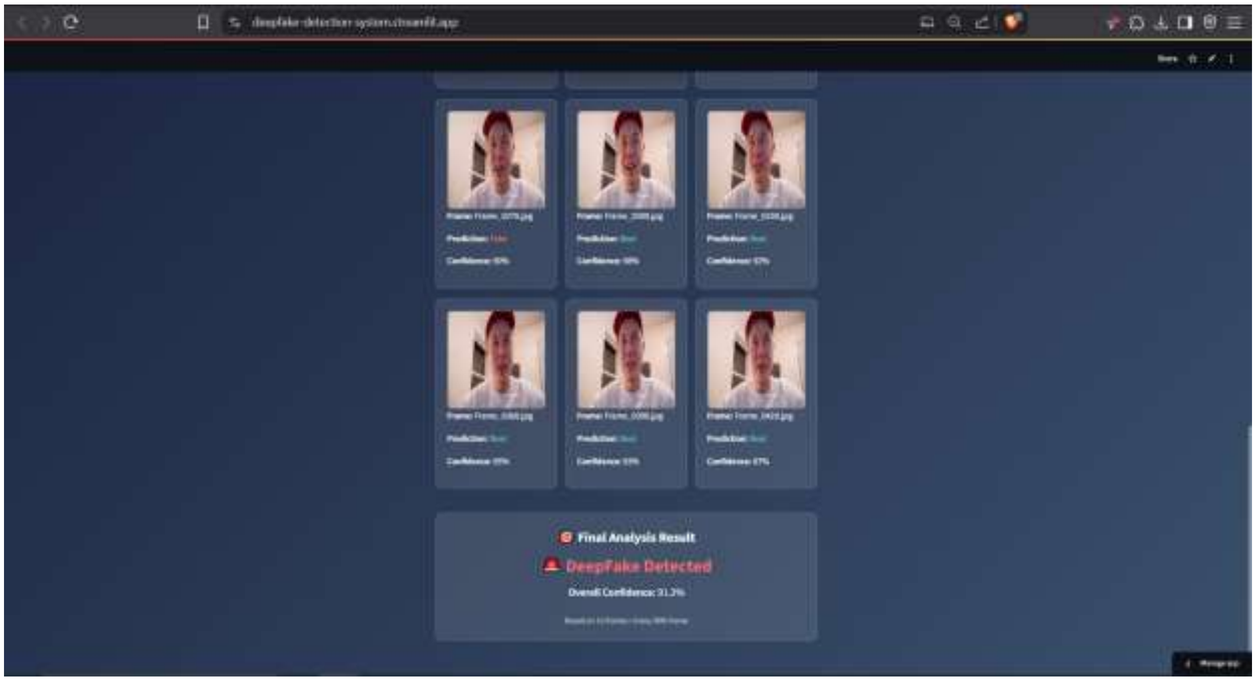
Frame: [illegible]
Prediction: [illegible]
Confidence: [illegible]

Frame: [illegible]
Prediction: [illegible]
Confidence: [illegible]

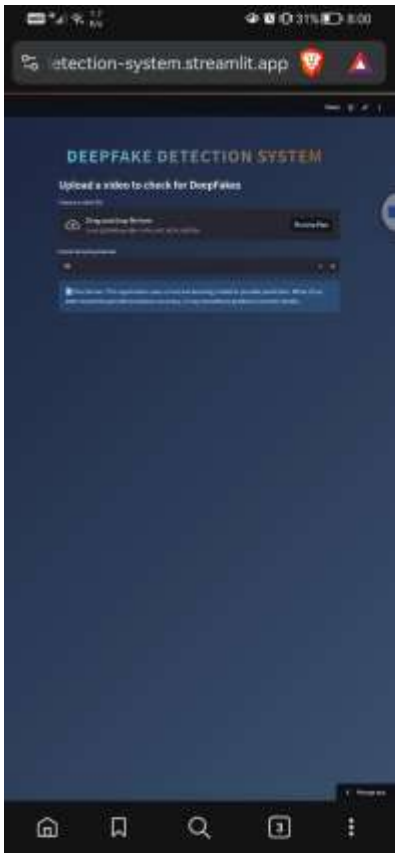
Frame: [illegible]
Prediction: [illegible]
Confidence: [illegible]

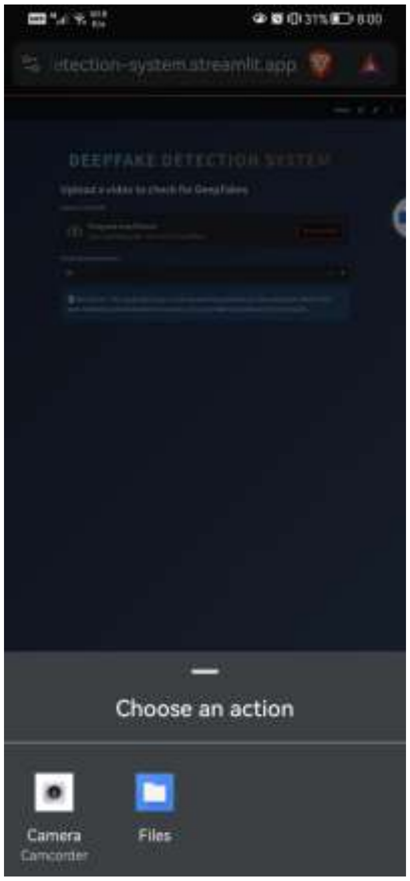
Frame: [illegible]
Prediction: [illegible]
Confidence: [illegible]

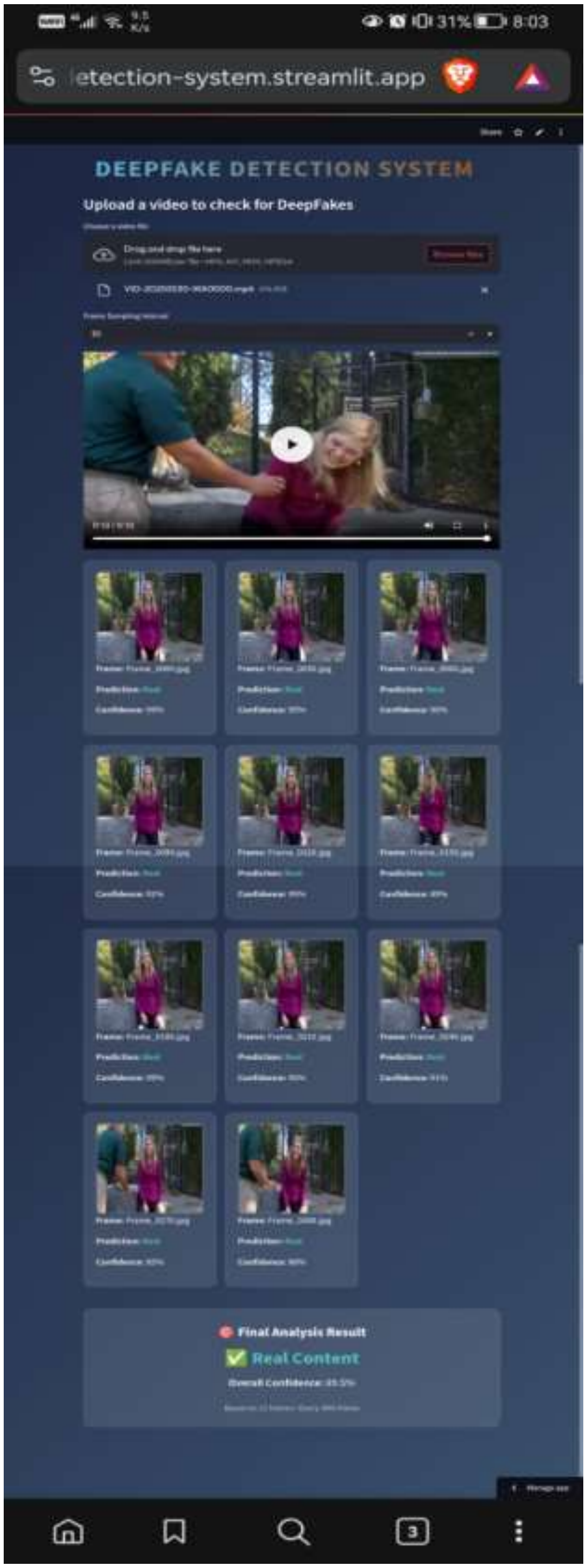
Frame: [illegible]
Prediction: [illegible]
Confidence: [illegible]


deepfake-detection-system.streamlit.app
Final Analysis Result
DeepFake Detected
Overall Confidence: 31.2%

9.2.2 Mobile User Interface


etection-system.streamlit.app
DEEPFAKE DETECTION SYSTEM
Upload a video to check for DeepFakes




etection-system.streamlit.app
DEEPFAKE DETECTION SYSTEM
Upload a video to check for DeepFakes
Drag and drop file here
Final Analysis Result
✅ Real Content
Overall Confidence: [illegible]

10. CONCLUSION

In conclusion, the proposed deepfake detection system offers a comprehensive and adaptive approach to combat the malicious use of deepfake technology. By integrating a hybrid of manual inspection, traditional forensic techniques, and state-of-the-art machine learning algorithms, the system achieves high accuracy, scalability, and robustness in identifying manipulated media across various formats.

Throughout this paper, we have highlighted the importance of addressing the multifaceted challenges posed by deep-fake technology, including scalability limitations, vulnerability to adversarial attacks, ethical considerations, and regulatory compliance. The proposed system addresses these challenges by leveraging advancements in machine learning, fostering interdisciplinary collaboration and prioritizing ethical principles and privacy protection. By adopting a proactive and collaborative approach, we can mitigate the risks associated with deepfake technology and safeguard the integrity of digital media in the age of AI. However, the fight against deepfakes is an ongoing battle, and there are several avenues for future enhancement and research.

We presented a neural network-based approach to classify the video as deep fake or real, along with the confidence of proposed model. Our method can predict the output by processing 1 second of video (10 frames per second) with good accuracy**.**

11. FUTURE SCOPE

1. Advanced Adversarial Robustness: Continuously improving techniques for adversarial training and defense mechanisms to enhance the system's resilience against sophisticated manipulation attempts.

2. Dynamic Dataset Augmentation: Developing techniques for dynamically augmenting datasets with new deepfake variations to ensure model generalization and adaptability to emerging threats.

3. Explainable AI: Incorporating explainable AI techniques to provide interpretable explanations for detection outcomes, enhancing transparency and trustworthiness of the system

4. Multimodal Fusion: Exploring techniques for integrating information from multiple modalities, such as text, images, and audio, to improve detection accuracy and reliability.

5. Privacy-Preserving Detection: Developing privacy-preserving detection methods that minimize the need for access to sensitive user data while maintaining detection effectiveness.

By pursuing these future enhancements and continuing to innovate in the field of deepfake detection, We can stay ahead of malicious actors and safeguard the integrity of digital media for generations to come. By pursuing these future enhancements and continuing to innovate in the field of deepfake detection, we can stay ahead of malicious actors and safeguard the integrity of digital media for generations to come.

12. REFERENCES

[1] Yuezun Li, Siwei Lyu, “Exposing DF Videos by Detecting Face Warping Artifacts,” in arXiv:1811.00656v3.

[2] Yuezun Li, Ming-Ching Chang and Siwei Lyu “Exposing AI Created Fake Videos by Detecting Eye Blinking” in arXiv:1806.02877v2.

[3] Huy H. Nguyen , Junichi Yamagishi, and Isao Echizen “ Using capsule networks to detect forged images and videos ” in arXiv:1810.11215.

[4] D. Güera and E. J. Delp, "Deepfake Video Detection Using Recurrent Neural Networks," 2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), Auckland, New Zealand, 2018, pp. 16.

[5] Umur Aybars Ciftci, ˙Ilke Demir, Lijun Yin “Detection of Synthetic Portrait Videos using Biological Signals” in arXiv:1901.02212v2

[6] TensorFlow: https://www.tensorflow.org/

[7] Keras: https://keras.io/

[8] Face app: https://www.faceapp.com/ (Accessed on 26 March, 2020)

[9] Face Swap : https://faceswaponline.com/

[10] Raghavendra, N., & Venkatesan, S. (2020). Deepfake Videos Detection and Classification Using Convolutional Neural Networks. arXiv preprint arXiv:2010.05540.

[11] Lin, Y., Xu, J., Luo, Y., & Liu, Z. (2021). Real-Time Deepfake Detection on Mobile Devices. IEEE Transactions on Mobile Computing.