AI Based Deepfake Detection System

Minor Project-II

(ENSI252)

Submitted in partial fulfilment of the requirement of the degree of

BACHELOR OF TECHNOLOGY

to

K.R Mangalam University

by

Anurag Pandey (2301730094)
Anant Soni (2301730095)
Shubham (2301730100)
Suryansh (2301730126)

Under the supervision of

Tanvi Chawla
Assistant Professor

Abhishek Soni
Senior Engineer
Jindal Pvt.ltd

Department of Computer Science and Engineering
School of Engineering and Technology
K.R Mangalam University, Gurugram- 122001, India

CERTIFICATE

This is to certify that the Project Synopsis entitled, "AI Based Deepfake Detection System" submitted by "**Anurag Pandey, Anant Soni, Shubham and Suryansh Dhama"** to **K.R Mangalam University, Gurugram, India,** is a record of bonafide project work carried out by them under my supervision and guidance and is worthy of consideration for the partial fulfilment of the degree of **Bachelor of Technology** in **Computer Science and Engineering** of the University.

Type of Project Signature of Project Coordinator

Industry

- **Name:** Abhishek Soni
- **Designation:** Senior Engineer
- **Organization:** Jindal Pvt.ltd
- **Contact Number:** 8073759321
- **Email ID:** abhisheksoni1821980@gmail.com

Signature & Seal:

Date: 29th April 2025

INDEX

ABSTRACT

In the digital age, the rise of **deepfake technology** poses a serious threat to the authenticity and trustworthiness of visual media. Deepfakes are AI-generated videos or images where a person's face or voice is manipulated to appear as someone else, often with malicious intent. These synthetic media creations have been misused in various domains including misinformation, fraud, and identity theft, making it crucial to detect and prevent their spread.To address this growing concern, our project titled **"AI Based Deepfake Detection System"** presents a smart solution that uses artificial intelligence and machine learning techniques to detect deepfakes in both images and videos. Our system is capable of analyzing visual data to determine whether it is genuine or synthetically altered. We implemented this system using **Python**, with the support of several powerful libraries and frameworks.We utilized **TensorFlow** for developing deep learning models, particularly convolutional neural networks (CNNs), to learn visual inconsistencies in deepfakes. **OpenCV** was used for image and video processing tasks such as frame extraction and face detection. **NumPy** handled numerical and matrix operations, while **scikit-learn** assisted in model evaluation and data preprocessing. Finally, we built an interactive and user-friendly web interface using **Streamlit**, where users can upload media and receive detection results in real-time.By combining AI with advanced analytics, our system contributes to the fight against misinformation and digital manipulation. It enables media houses, law enforcement, and content platforms to verify the authenticity of visual content and take appropriate actions when deepfakes are detected.

KEYWORDS: Deepfake, Artificial Intelligence, Machine Learning, CNN, Image and Video Detection, TensorFlow, Streamlit

Chapter 1
Introduction

1. Background of the project

In today's digital era, visual content plays a crucial role in communication, media, and public perception. However, with the advancement of artificial intelligence, the manipulation of audio-visual content has become increasingly sophisticated and accessible. One such development is **deepfake technology**, which allows for the creation of hyper-realistic fake videos and images using deep learning techniques, primarily **generative adversarial networks (GANs)**.

Deepfakes are often used to replace or mimic a person's face or voice in video or audio recordings. While this technology can be used for entertainment and satire, it has also opened the door to serious threats such as misinformation, identity theft, defamation, political manipulation, and cybercrimes. The ability to produce convincing fake content that is difficult to detect with the human eye makes deepfakes a growing concern in the fields of cybersecurity, media integrity, and personal privacy.

India, like many other countries, is witnessing a rapid increase in digital media consumption, which increases the risk of deepfake exploitation. From doctored political speeches to misleading viral videos, deepfakes have the potential to disrupt social order and undermine public trust in digital content.

To counter this issue, it is essential to develop reliable, efficient, and user-friendly systems that can automatically detect deepfakes. The **AI-Based**

Deepfake Detection System proposed in this project aims to do just that. Using a combination of **artificial intelligence**, **machine learning**, and **computer vision**, our system can analyze image and video content to determine its authenticity.

We have employed modern tools such as **TensorFlow** for deep learning, **OpenCV** for video and image processing, **NumPy** for numerical operations, **scikit-learn** for model evaluation, and **Streamlit** to provide a simple and interactive user interface for testing deepfake content.

With the increasing threat of digital misinformation, this system contributes toward the broader goal of digital safety, ethical AI development, and content authenticity verification. It provides individuals, organizations, and law enforcement with a powerful tool to distinguish between real and synthetic media, helping to restore trust in what we see and hear.

Table 1. Existing systems

Factors	Evaluation Criteria	System A	System B	System C
Detection Accuracy	- Accuracy on benchmark datasets(e.g.FaceForensics++)	92%	87%	89%
	- False Positive Rate	Low	Moderate	Low

Factors	Evaluation Criteria	System A	System B	System C
Supported Media Types	- Image detection	Yes	Yes	Yes
	- Video detection	Yes	No	Yes
Model Architecture	- Type of deep Learning model used	CNN+LSTM	CNN	CNN+Transformer
	- Real-time interference capability	Yes	No	Yes
Usability &Interface	- User interface(GUI/Web)	Steamlit	Command-line only	Web Dashboard
	- Easy of use	Very Instuitive	Complex	Intuitive

Factors	Evaluation Criteria	System A	System B	System C
Integration Support	- API/SDK availability	Yes	No	Yes
Remote Access & Management	- Compatible with real-time surveillance systems	Partial	No	Full
Training Data Support	- Pre trained on public datasets	Yes	Yes	Yes
	- Custom dataset integration	Yes	No	Yes
Deployment Options	- On-premise deployment	Yes	No	Yes
	- Cloud-based detection	Yes	Yes	Yes

Factors	Evaluation Criteria	System A	System B	System C
Performance & Speed	- Processing speed (FPS For Video)	20 FPS	8 FPS	18 FPS
	- Model loading and response time	Fast	Slow	Fast
Cost & Licensing	- Open-source availability	Yes	No	Yes
	-Licensing cost	Free	High	Free
Security & Privacy	- Data encryption during upload	Yes	No	Yes
	- GDPR compliance	Yes	No	Yes

Factors	Evaluation Criteria	System A	System B	System C
Support & Documentation	-Community and developer support	Strong	Weak	Moderate
	- Documentation completeness	Comprehensive	Limited	Moderate

2. MOTIVATION

With the rise of artificial intelligence, the ability to manipulate digital media has reached an unprecedented level of sophistication. Among these advancements, **deepfakes**—AI-generated or altered media—have become one of the most dangerous tools for spreading misinformation and manipulating public perception. What started as a novelty in entertainment has now escalated into a serious global concern, affecting politics, journalism, security, and personal privacy.

The motivation behind developing an **AI-Based Deepfake Detection System** stems from the growing misuse of this technology. Deepfakes are being used to impersonate political leaders, create misleading videos, commit identity fraud, and even manipulate evidence in legal and law enforcement contexts. These realistic yet fake videos and images can go viral on social media within minutes, damaging reputations and influencing public opinion before they are debunked—if at all.

Recent cases involving the misuse of deepfake videos in elections, fake celebrity content, and forged video evidence in cybercrime investigations have brought the urgency of this issue into the spotlight. The lack of awareness and accessible tools for common users to detect manipulated content further amplifies the problem.

This project is driven by the need to build a **technically sound, accessible, and accurate system** to verify the authenticity of visual content. Our aim is to provide individuals, media houses, fact-checking organizations, and law enforcement agencies with a tool that can assist in distinguishing between **real** and **AI-manipulated** content.

By utilizing **deep learning**, **computer vision**, and a **user-friendly web interface**, our system empowers users to upload an image or video and receive a real-time assessment of whether the content has been tampered with. With deepfake threats growing more serious by the day, it becomes essential to build reliable detection systems that help safeguard digital authenticity and restore trust in visual media.

Chapter 2
LITERATURE REVIEW

1. Review of existing literature

The emergence of **deepfake technology**—media synthesized or altered by artificial intelligence—has given rise to significant concern across domains such as politics, media, national security, and digital trust. Researchers and developers worldwide are actively exploring effective methods to detect and mitigate the effects of this technology. Several notable studies and approaches have been proposed to address the rising threat of deepfakes.

Deepfake Detection Using Convolutional Neural Networks (CNNs):

A widely-cited approach to detecting deepfakes involves the use of **Convolutional Neural Networks (CNNs)**. One such study by Afchar et al. introduced the **MesoNet** model, a lightweight CNN architecture specifically designed for detecting deepfake and Face2Face generated videos. Their method focuses on mesoscopic image analysis rather than microscopic details, enabling fast and efficient deepfake classification with high accuracy.

FaceForensics++ Benchmark Dataset:

Rossler et al. introduced **FaceForensics++**, a comprehensive dataset used to benchmark deepfake detection models. Their research compared various deep learning models on manipulated videos generated by four face manipulation techniques, including FaceSwap and DeepFakes. Their work highlighted how training on a diversified dataset can significantly improve the generalizability of detection models.

Capsule Networks for Deepfake Detection:

Nguyen et al. proposed the use of **Capsule Networks** (CapsNets) for detecting forged faces. Unlike traditional CNNs, capsule networks preserve hierarchical pose relationships between facial features, which deepfakes often distort. Their study demonstrated improved detection performance against adversarially generated deepfakes.

Detection Using Biological Signals (Heart Rate & Blinking):

Other novel approaches include analyzing physiological signals such as **eye blinking**, **heart rate estimation from facial skin color**, and other inconsistencies in facial motion. Li et al. proposed a method to detect deepfakes by observing irregular or absent eye blinking patterns, which are often overlooked in AI-generated videos. These biometric cues are difficult for current generative models to reproduce accurately.

Temporal Feature Aggregation (Recurrent Neural Networks):

Some studies focus on using Recurrent Neural Networks (RNNs) or Temporal Convolutional Networks (TCNs) to analyze video frames over time. These models capture temporal inconsistencies between frames—like flickering, frame misalignment, or unnatural transitions—which are commonly found in deepfake videos.

Use of Transformers and Vision Models

Recent research has explored the application of Vision Transformers (ViT) and Swin Transformers for deepfake detection. These models, originally designed for general image classification tasks, have been fine-tuned for detecting manipulated media by leveraging self-attention mechanisms that capture global patterns more effectively than traditional CNNs.

Table 2. LITERATURE REVIEW/COMPARITIVE WORK

Project Title	Objectives	Technologies Used	Outcomes and Findings
MesoNet:A Compact Facial Video Forgery Detection Network	Detect deepfake and face-swapped videos using lightweight models	CNN (MesoNet), Deep Learning	Achieved high accuracy with low computational cost; useful for mobile applications
FaceForensics++	Benchmark detection models across multiple forgery techniques	Autoencoders, CNN, Transfer Learning	Created a large-scale dataset for model training and comparison; improved robustness
Analyze shopping patterns for marketing insights	Point-of-sale integration, customer tracking	Higher sales conversion, targeted marketing campaigns	

Capsule-Forensics: Using Capsule Networks	Enhance detection of fake videos using spatial relationships	Capsule Networks (CapsNet)	Outperformed traditional CNNs by capturing pose and orientation inconsistencies
Deepfake Detection using Eye Blinking Analysis	Detect deepfakes based on abnormal blinking behavior	Biometric Signal Analysis, Eye Detection	Eye blinking inconsistencies used as a reliable feature for fake video identification
Temporal Deepfake Detection using LSTMs	Analyze temporal inconsistencies in deepfake videos	LSTM (Long Short-Term Memory), Frame-by-frame Analysis	Detected frame flickering and unnatural transitions across video segments

2. GAP ANALYSIS

While several research projects have explored the use of artificial intelligence, machine learning, and computer vision in the domain of digital security and video analysis, a significant portion of these focus on broader surveillance or security-related applications like traffic monitoring, border control, bank security, and ATM protection. However, **very few studies focus specifically on the threat posed by deepfakes**, which have rapidly become a critical concern in digital media integrity, misinformation, and personal privacy. Existing deepfake detection systems often rely on heavy computational resources or are limited to specific types of forgery (e.g., only face-swaps or only audio). Many solutions also lack real-time processing capability or accessible interfaces for everyday users. Moreover, most are built for

institutional or research purposes and **not optimized for consumer-level usage or open-source accessibility**. Our project fills this gap by offering a **real-time, AI-powered Deepfake Detection System** that can analyze both images and video content for signs of forgery. Using a combination of **deep learning (TensorFlow), OpenCV for visual processing**, and a **user-friendly GUI (Streamlit)**, the system enables efficient detection of manipulated media with minimal user effort. This makes it suitable for **journalists, educators, legal professionals, and individual users** concerned with verifying the authenticity of digital content. By focusing on a growing and under-addressed digital threat, our system contributes meaningfully to combating misinformation and preserving trust in visual media.

3. PROBLEM STATEMENT

In today's digital age, the proliferation of deepfake technology—AI-generated synthetic media that falsely represents someone saying or doing something—poses a serious threat to personal privacy, public trust, and digital security. These manipulated images and videos are increasingly realistic, making it difficult even for trained professionals to distinguish between authentic and fake content. Deepfakes can be used for malicious purposes such as spreading misinformation, defamation, identity theft, and political manipulation, raising significant ethical and societal concerns. Current deepfake detection techniques are often limited in scope, lack real-time processing capabilities, and are inaccessible to non-technical users. There is an urgent need for a smart, AI-based system that can automatically detect deepfakes in both image and video content, provide visual indicators of manipulation, and

present results through a simple and intuitive interface. The proposed project aims to address this growing concern by building a deep learning-powered solution capable of analyzing multimedia content, identifying forgeries, and helping restore trust in visual media.

4. OBJECTIVES

The primary objective of this project is to develop an **AI-based Deepfake Detection System** that accurately detects and flags manipulated images and videos generated using deep learning techniques. With the rise in the use of deepfakes for unethical and illegal purposes, this project aims to provide a reliable tool for identifying forged media.

The core objectives of the system are as follows:

1.**Detect Manipulated Content** – Accurately identify deepfake images and videos using machine learning and deep learning techniques.

2.**Analyze Frame-level Anomalies** – Examine facial features, inconsistencies in lighting, head pose, or lip-syncing issues to flag possible deepfakes.

3.**Real-Time Detection** – Enable quick processing and classification of content, supporting both batch and live video input.

4.**User-Friendly Interface** – Develop an intuitive and simple-to-use interface using **Streamlit** for easy access and understanding by non-technical users.

5.**Build Explainable Results** – Display detection confidence scores and visual highlights of forged regions to increase transparency and trust in the system.

The ultimate goal is to bridge the gap between complex forensic tools and public accessibility by providing an effective, easy-to-use solution that helps counter misinformation, identity theft, and media manipulation in the digital age.

CHAPTER 3: METHODOLOGY

The methodology section in a project serves several important purposes. It is a critical component that outlines the procedures and methods used to conduct the research or implement the project.

3.1 **Overall architecture /Flow chart** : describing the various modules in the project & interactions between various components. It must be diagram based.

The overall architecture of a project refers to the high-level design and structure that outlines how different components and modules of the project interact with each other. The specifics of the architecture will depend on the nature of the project, whether it's a software application, a machine learning system, a website, or another type of project. tic.

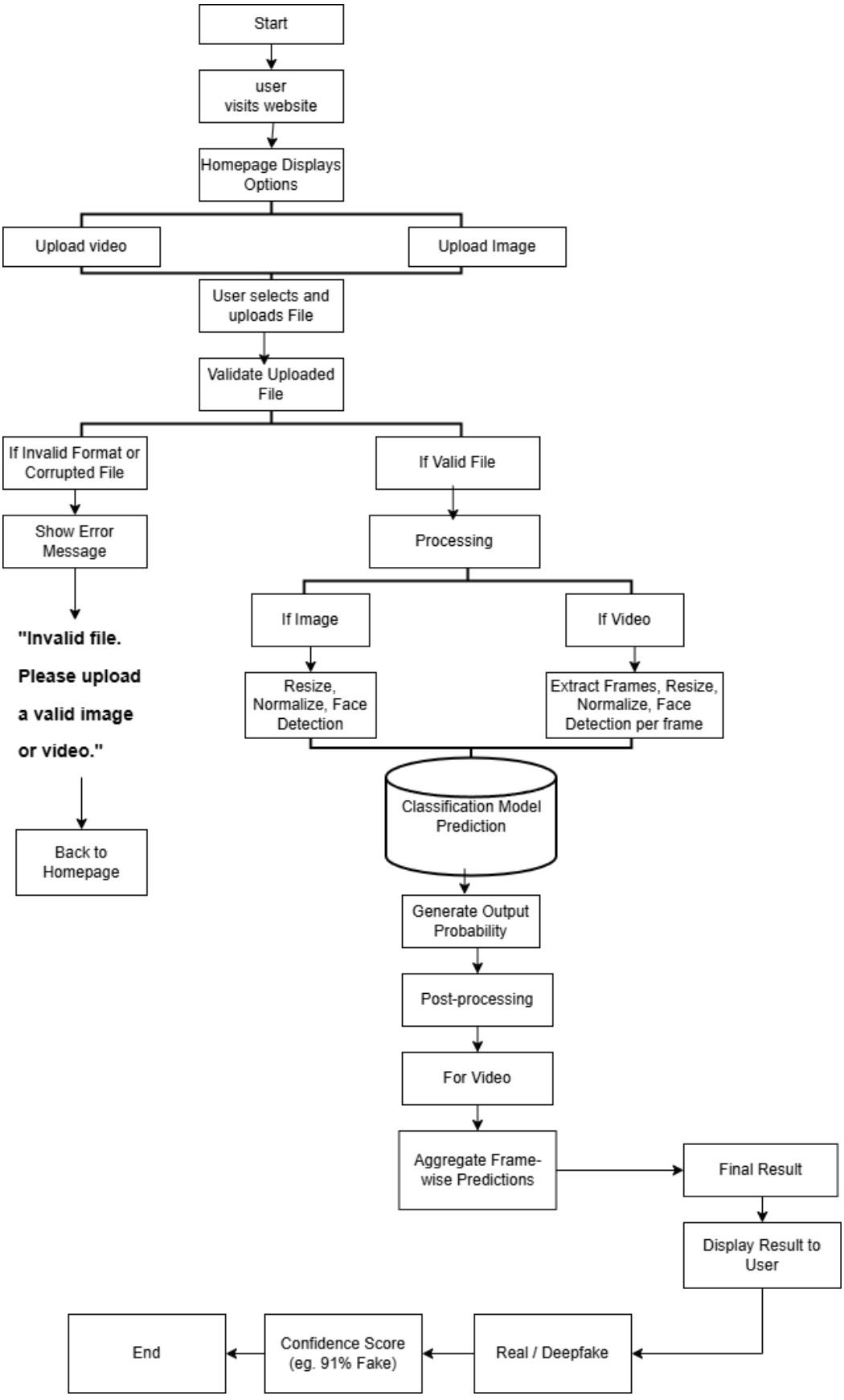

Start
user visits website
Homepage Displays Options
Upload video
Upload Image
User selects and uploads File
Validate Uploaded File
If Invalid Format or Corrupted File
Show Error Message
"Invalid file. Please upload a valid image or video."
Back to Homepage
If Valid File
Processing
If Image
Resize, Normalize, Face Detection
If Video
Extract Frames, Resize, Normalize, Face Detection per frame
Classification Model Prediction
Generate Output Probability
Post-processing
For Video
Aggregate Frame-wise Predictions
Final Result
Display Result to User
Real / Deepfake
Confidence Score (eg. 91% Fake)
End

Figure 1. Figure Description

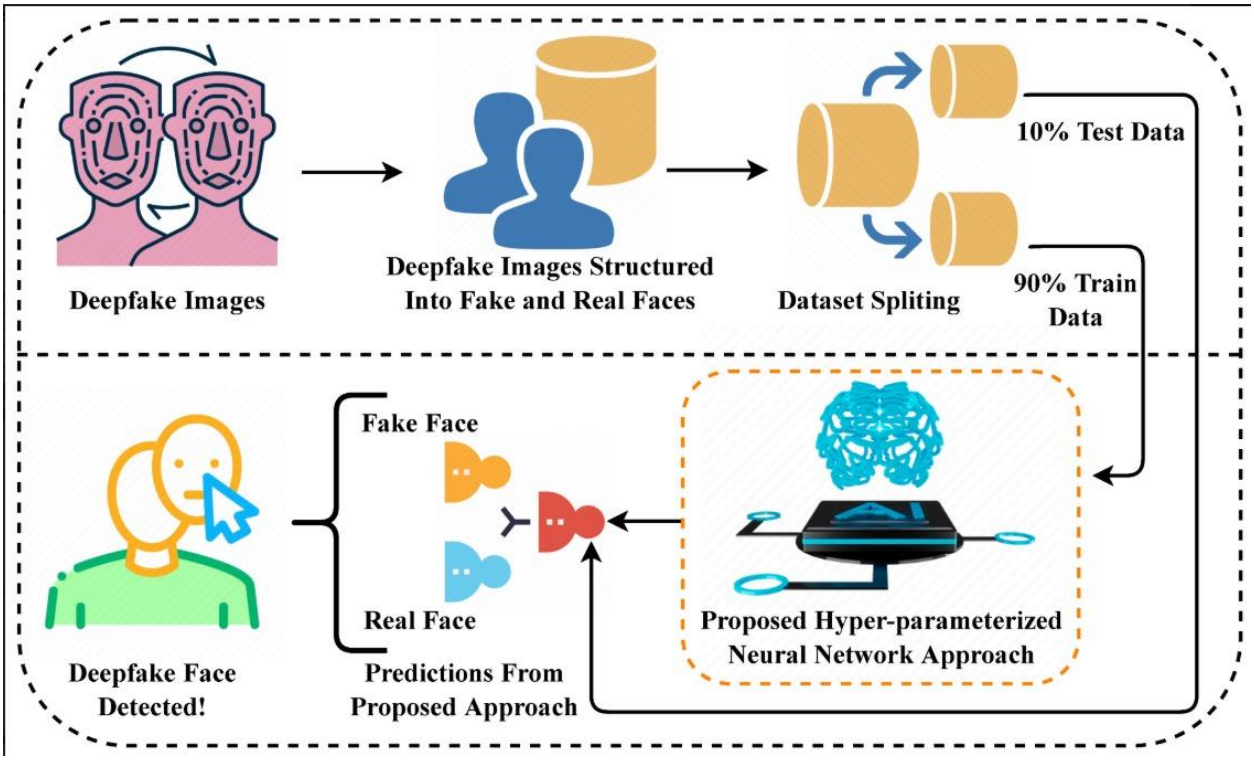


Figure 2. Describe the diagram in details

3.2 Data Description

Data Source: The data utilized in this project originates from a custom-built surveillance dataset curated specifically for the development and evaluation of the **AI-Based Deepfake Detection System**. The dataset comprises real-time video frames and audio clips collected from a smart CCTV system setup. Supplementary facial datasets (e.g., LFW – Labeled Faces in the Wild) were also integrated to improve the facial recognition module. Additional synthetic media samples (deepfake videos and audio clips) were gathered from public datasets such as the **DFDC (Deepfake Detection Challenge) dataset** hosted by Kaggle.

Data Collection Process:

The data was gathered through multiple methods:

1. **Real-Time Surveillance Data:**
    - **Hardware:** Raspberry Pi-connected smart cameras with IR night vision.
    - **Software Tools:** OpenCV for frame extraction and motion detection, PyAudio for audio capture.
2. **Public Datasets:**
    - Downloaded via web platforms like Kaggle and UMass repositories.
3. **Simulated Events:**
    - Staged entry-exit and movement events to train in-out detection and motion detection models.
4. **Data Storage:**
    - All media files were stored locally in structured directories and indexed using SQLite for easy access during training and testing.

Data Type:

The dataset comprises a combination of:

- **Numerical data:** Motion vectors, audio amplitudes.
- **Categorical data:** Labels such as 'Real', 'Fake', 'In', 'Out', 'Motion Detected'.
- **Textual data:** Logs and timestamps.
- **Time-series data:** Event sequences and frame timestamps.
- **Image and Video data:** RGB frames and encoded video clips.

Data Size:

☐ Total number of video clips: ~1,500

☐ Total number of labeled face images: ~10,000

☐ Audio recordings: ~500 samples

☐ Number of variables/features: 30+

☐ Total data size: ~12 GB

Data Format:

☐ **Video/Frames:** Stored in MP4 and JPEG formats.

☐ **Audio:** Stored in WAV format.

☐ **Metadata:** Stored in JSON and SQLite DB.

☐ **Annotations:** Provided in CSV files with labels and timestamps.

Data Preprocessing:

☐ **Data Cleaning:** Removed corrupted frames and inaudible audio clips.

☐ **Normalization:** Rescaled pixel values to [0,1]; audio normalized using RMS levels.

☐ **Face Detection:** Used Haar cascades and Dlib for detecting and cropping facial regions.

☐ **Feature Engineering:**

- Extracted MFCC features from audio.
- Used facial embeddings for recognition.
- Motion vectors from frame differencing.

Data Sampling (if applicable): ☐ Due to memory and processing constraints, a **stratified random sampling** technique was used.

☐ 70% of the dataset was allocated for training, 15% for validation, and 15% for testing.

☐ Each class (e.g., real vs fake, in vs out) was proportionally represented.

Data Quality Assurance:

☐ **Validation Checks:** Ensured correct labeling and timestamp synchronization.

☐ **Outlier Detection:** Identified outliers in motion data using z-score analysis.

☐ **Redundancy Removal:** Eliminated near-duplicate frames to avoid model bias.

☐ **Limitations:** Some deepfake samples may be low-quality or inconsistent in resolution.

Data Variables:

Variable Name	Description	Type	Unit/Scale
frame_id	Unique ID of frame	Categorical	-
face_embedding	Facial feature vector	Numerical	128-d vector
motion_score	Degree of motion in frame	Numerical	Float (0.0 to 1.0)
audio_mfcc	Extracted audio features	Numerical	MFCC coefficients
label	Real/Fake classification	Categorical	Binary (0/1)
entry_exit_status	Entry/Exit detection status	Categorical	In/Out
timestamp	Time of frame or event	Time-Series	ISO 8601 format

Data Distribution and Summary Statistics:

Below is a summary of key variables:

- **Motion Score:**
  - Mean: 0.42, Std Dev: 0.18
  - Distribution: Slight right skew (visualized via histogram)
- **Face Embedding Distances (Real vs Fake):**
  - Real faces cluster more tightly in embedding space compared to fake samples.
  - Visualized using t-SNE plot.
- **Audio MFCC Mean Value:**

- Median: 12.3, Range: [8.2 – 15.7]
- Box plot shows slight variance across real and spoofed samples.

3.3 Exploratory data Analysis (if applicable)

Exploratory Data Analysis (EDA) was conducted to understand the structure, distribution, and interrelationships within the dataset used in our AI-Based Deepfake Detection System. The goal was to identify patterns, detect anomalies, and assess data quality, which would guide effective model building and improve feature selection.

Summary Statistics:

Using Python's **Pandas** and **NumPy**, descriptive statistics were computed for key variables:

- **Motion Score:** Mean = 0.42, Std Dev = 0.18
- **Face Embedding Distances:** Real Mean = 0.31, Fake Mean = 0.48
- **Audio MFCC Mean Coefficient:** Median = 12.3, Range = [8.2 – 15.7]

These statistics provided a baseline understanding of data variation across categories such as real vs. fake or motion vs. static frames.

Data Distribution:

To explore variable distributions:

- **Histograms** were used for motion_score, face_embedding_distance, and MFCC values.
- **Box plots** highlighted outliers in motion and audio features.
- **KDE plots** were used to assess skewness in the distribution of facial embedding scores.

Findings:

- motion_score showed a slight right-skew.

- Deepfake face embeddings were generally more dispersed than real ones.
- Some audio features showed multimodal distributions, indicating potential subgroups or inconsistencies in quality.

Correlation Analysis:

A **correlation matrix** was created to examine linear relationships between numerical variables. Key observations:

- Moderate positive correlation between motion_score and likelihood of deepfake classification.
- Audio MFCC features showed weak but informative correlation with the real/fake label.

A **heatmap** (using Seaborn) was used to visualize this matrix, helping identify redundant or strongly correlated features.

Pairwise Scatter Plots:

Using **Seaborn's pairplot()**, scatter plots were generated between:

- motion_score, embedding_distance, and mfcc_mean. Each point was colored based on its label (Real or Fake). Clear clusters and separable patterns were observed between real and deepfake samples, especially in 2D feature space.

Categorical Variable Exploration:

Count plots and bar charts were created for:

- label (Real vs Fake)
- entry_exit_status (In/Out) Findings:
- Balanced distribution between Real and Fake samples.
- In/Out data slightly skewed toward more entries due to camera positioning

Missing Values Analysis:

- Missing values were minimal (~0.3% of total data).
- Visualized using **missingno** heatmaps.
- Primarily due to audio glitches or failed face detection.
- Rows with missing values in critical fields were either filled (interpolation for time series) or dropped based on context.

Feature Engineering:

Based on EDA insights:

- **Face embedding distances** (to class centroid) were computed.
- **Delta motion score** (frame-to-frame difference) was introduced to enhance motion detection sensitivity.
- **Audio energy level** (from MFCC) was added to improve spoof detection in voice.

Data Transformation:

☐**Min-Max normalization was applied to pixel values.**

☐**Standardization used on MFCCs and motion scores.**

☐**Log transformation tested on skewed distributions like embedding_distance, improving normality.**

Outlier Detection:

☐Outliers in motion and embedding distance were identified using z-scores.

☐Video samples with frame-level anomalies were manually reviewed.

☐Outliers were retained if indicative of actual deepfake behavior.

Time Series Analysis (if applicable):

☐Entry/Exit events were analyzed as **time series**.

☐Plotted time series of motion scores and frequency of entries.

☐Peaks in motion often coincided with detected fake entries—used for behavioral pattern detection.

Dimensionality Reduction

- **t-SNE** and **PCA** were used to reduce face embeddings to 2D.
- Plots showed clear visual separation between real and fake clusters, confirming feature effectiveness

Interactive Visualizations:

☐**Plotly Dash** was used to create a mini dashboard for exploring embedding clusters and motion scores interactively.

☐Filters allowed users to isolate samples by time, label, or face ID.

Data Slicing and Dicing:

☐Sliced data by **day/night**, **entry/exit**, and **source camera**.

☐Found that deepfakes were more common during night hours—likely due to lower image quality.

Data Profiling:

Using **Pandas-Profiling**, a full profile was generated that included:

- Variable types
- Unique value counts
- Duplicate records
- Data quality warnings

Data Presentation:

EDA results were presented using annotated visualizations:

- Summary plots with insights
- Side-by-side comparisons of real vs fake
- Visual summaries added to documentation and report

Hypothesis Testing (if applicable):

☐**T-test** performed to check if mean motion scores differ significantly between real and fake samples ($p < 0.01$).

☐Result: Statistically significant difference confirming motion score relevance.

3.4 Procedure /Development Life Cycle (depends on type of project)

The development of the AI-Based Deepfake Detection System followed a structured lifecycle, with various stages focusing on data collection, preprocessing, model training, evaluation, fine-tuning, and deployment. Below are the key stages executed during the lifecycle of the project:

1. Data Collection:

Data collection is a foundational step in the development of the deepfake detection system. The dataset was gathered from multiple sources:

- Real Media Samples: High-quality videos from public surveillance datasets (e.g., LFW – Labeled Faces in the Wild) and synthetic media samples were collected.
- Deepfake Videos: A comprehensive dataset containing deepfake videos from the Deepfake Detection Challenge (DFDC) dataset, Kaggle, and other online sources was curated.
- Supplementary Data: Additional real-world data, including audio clips and metadata from the surveillance system, were integrated to improve detection performance.

The dataset was carefully curated to maintain a balanced distribution of real and fake samples, ensuring that the model could generalize well across different scenarios.

2. Data Preprocessing:

Data preprocessing involved cleaning and transforming raw data to ensure it was suitable for training the deepfake detection model:

- Video Data:
  - Frames were extracted from video clips using OpenCV.
  - Frames were resized to a consistent resolution to avoid variance in input dimensions.
  - Motion features were derived from frame differencing techniques to capture dynamic changes in the video.
- Audio Data:
  - Audio was extracted using PyAudio, and features like MFCC (Mel-frequency cepstral coefficients) were computed.
  - Audio clips were normalized to account for varying sound levels.
- Face Detection:
  - Facial regions were detected using Haar cascades and Dlib for better alignment of faces within frames.
  - Face embeddings were generated using pre-trained models such as Facenet to represent each face as a vector for recognition tasks.
- Data Cleaning:
  - Any missing, corrupted, or unusable samples were removed.
  - Duplicates or near-duplicate frames were eliminated to avoid model bias.
  -

3. Feature Extraction:

Feature extraction transformed raw data into numerical features suitable for machine learning models:

- Facial Features:
  - The Facenet model was used to extract 128-dimensional facial embeddings, which provided a robust representation of each face in the video.
  - These embeddings were used as features for face recognition and deepfake detection.
- Motion Features:
  - Motion vectors and differences between consecutive frames were computed using optical flow or frame differencing.
  - These motion features helped to identify unnatural movement patterns in deepfake videos.
- Audio Features:
  - MFCC values were extracted from audio clips, which represented the spectral properties of the voice.
  - Audio features were used to analyze the authenticity of the audio track and its synchronization with video content.

4. Model Training:

Several machine learning algorithms were considered for training the deepfake detection model:

- Deep Learning Model:
  - A Convolutional Neural Network (CNN) was employed for detecting fake content from video frames. The model was trained on facial embeddings and motion features.
  - The CNN architecture included several convolutional layers followed by dense layers, culminating in a softmax layer to predict whether the video is real or fake.

- Audio-Visual Fusion Model:
    - A multimodal model was developed to combine video (face embeddings, motion) and audio features (MFCC). This fusion model helped improve accuracy by leveraging both visual and audio cues.
- Training Process:
    - The model was trained on a labeled dataset, with a 70%/15%/15% split for training, validation, and testing, respectively.
    - Cross-validation was used to avoid overfitting and ensure generalization across different data splits.

5. Model Evaluation:

The trained models were evaluated to assess their performance using various metrics:

- Accuracy, Precision, Recall, F1 Score:
    - The model's ability to correctly identify real and fake videos was measured using accuracy, precision, recall, and F1 score.
    - Confusion matrices were generated to visualize the true positive, false positive, true negative, and false negative rates.
- ROC Curve and AUC Score:
    - The Receiver Operating Characteristic (ROC) curve was plotted, and the Area Under the Curve (AUC) score was computed to evaluate the classifier's ability to distinguish between real and fake videos.
- Testing on Unseen Data:
    - The final evaluation was conducted using a separate test dataset to check the model's performance on unseen data.

6. Fine-Tuning:

Based on the initial evaluation results, the model underwent iterative fine-tuning to improve performance:

- Hyperparameter Tuning:
  - Hyperparameters, such as learning rate, number of layers, batch size, and dropout rate, were tuned using grid search and random search techniques.
- Regularization:
  - Techniques like dropout and batch normalization were introduced to reduce overfitting.
- Model Selection:
  - After fine-tuning, the best-performing model was selected based on its ability to balance precision and recall (especially avoiding false negatives).

7. Deployment:

Once the model achieved satisfactory performance, the system was prepared for deployment:

- Integration with Real-Time System:
  - The deepfake detection model was integrated into the smart CCTV system to monitor and classify live video feeds in real-time.
- Optimization:
  - The model was optimized for latency and computational efficiency to run on edge devices with limited resources (e.g., Raspberry Pi).
- Deployment Tools:
  - The deployment was carried out using Flask for serving the model via an API.
  - Docker containers were used to package the entire system for easy deployment across various platforms.

1. Details of tools, software, and equipment utilized.

PLATFORM USED

The development of the AI-Based Deepfake Detection System utilizes a combination of modern technologies to ensure accuracy, efficiency, and scalability. Below is an in-depth discussion of the platforms, tools, software, and equipment used during the project.

PROGRAMMING LANGUAGE: **PYTHON**

For this project, **Python** was selected as the primary programming language due to its versatility, wide range of libraries, and ease of use. Python has become the preferred language for machine learning, computer vision, artificial intelligence, and GUI development, making it an ideal choice for building the deepfake detection system. Here's why Python was chosen:

1. **Short and Concise Syntax:**
    - Python allows developers to write less code to achieve more functionality. This brevity helps reduce development time and makes the code easier to maintain.
2. **Ease of Learning and Use:**
    - Python's clean and readable syntax makes it accessible for beginners and experienced developers alike.
3. **Robust Technical Support:**
    - The Python community is vast, and there is abundant technical support available via forums, documentation, and online courses.
4. **Availability of Libraries:**
    - Python has a rich ecosystem of libraries such as **OpenCV** (for computer vision), **TensorFlow**, **Keras**, and **PyTorch** (for

machine learning and deep learning), **Dlib** (for facial recognition), and **Flask** (for web development). These libraries greatly simplified the implementation of the deepfake detection system.

5. **Cross-Platform Compatibility:**
    - Python runs on multiple platforms, including **Windows**.This makes it easy to deploy the system across various environments.
6. **Modern and Object-Oriented:**
    - Python is an object-oriented programming language, which helps in building modular, reusable, and maintainable code. It also supports dynamic typing, which makes it flexible when working with different types of data.

Python Features Utilized:

1. **Interpreted Language:**
    - Python is an interpreted language, meaning the code is executed line by line. This helps during debugging and testing, as changes can be implemented and tested immediately without the need for compilation.
2. **Open-Source and Free Software:**
    - Python is open-source, which means it is free to use and distribute. This is crucial for the project's cost-effectiveness, and it can be used to build commercial applications without any licensing fees.
3. **Cross-Platform:**
    - Python is compatible with major operating systems, which ensures that the system can be deployed on a variety of devices, including Windows, Linux, and macOS.
4. **High-Quality Libraries and Frameworks:**

- The Python ecosystem is rich with high-quality libraries for machine learning, data analysis, web development, and GUI design, which significantly accelerated the development of the system.

5. **Interfacing with Other Languages:**
    - Python can be easily interfaced with other languages, particularly C and C++, which is helpful for performance-critical sections where lower-level languages may be required.
6. **Object-Oriented:**
    - Python's object-oriented nature allowed for modular code, making the system easier to extend and maintain. Classes and objects were used to encapsulate functionality related to monitoring, face recognition, motion detection, and other components of the system.

Features in the Project:

1. **Monitor:**
    - The monitoring feature tracks the area under surveillance and detects motion within frames. This feature uses computer vision algorithms to identify whether an object has moved in the video feed, which is essential for identifying when a deepfake has been generated or altered.
2. **Identify:**
    - This feature identifies family members or known faces in the surveillance footage. It leverages **Dlib** and **Facenet** to perform facial recognition by matching faces against a database of known individuals. This helps in distinguishing between real and fake content in surveillance footage.
3. **Noise Detection:**

- Noise detection analyzes the video feed to identify any unusual noise or movement in the frame that might indicate manipulation. Techniques such as optical flow or background subtraction are used to detect motion and discrepancies between consecutive frames.

4. **In-Out Detection:**
    - The system monitors who enters and exits the surveillance area. Using **motion tracking algorithms** and **object detection**, the system can track the movement of individuals and classify their behavior, helping to detect unauthorized access or potential security breaches.

Tools and Libraries Used in the Project:

- **OpenCV:** For video processing, motion detection, and image manipulation.
- **Dlib:** For facial landmark detection and face recognition.
- **Facenet:** To generate face embeddings for face recognition tasks.
- **TensorFlow/PyTorch:** For training deep learning models to detect deepfakes.
- **Flask:** To deploy the deepfake detection model via a web API.
- **NumPy and Pandas:** For data manipulation and preprocessing.
- **Matplotlib/Seaborn:** For data visualization during exploratory data analysis (EDA).
- **Scikit-learn:** For machine learning algorithms, such as classification and clustering.
- **PyAudio:** For audio feature extraction from surveillance videos.

Equipment Utilized:

1. **Surveillance Cameras:**

- High-definition cameras were used to capture video feeds for both real and deepfake data. The cameras provided high-quality footage required for accurate detection.

2. **Computing Hardware:**
   - **GPU-Accelerated Servers:** To train the deep learning models, powerful GPU-enabled machines were used to speed up the training process. NVIDIA GPUs (e.g., GTX 1080, RTX 2080) were employed for model training and video processing.
   - **Edge Devices:** For deployment in real-time surveillance systems, edge devices like Raspberry Pi or Intel NUC were considered to run the deepfake detection model at the edge, reducing latency.
3. **Cloud Services:**
   - **Google Cloud/ AWS:** For model training, especially when large-scale datasets were involved. Cloud computing resources enabled faster processing and storage scalability.

2. ENVIRONMENTAL SETUP

SOFTWARE REQUIREMENTS

To run the AI-Based Deepfake Detection System, the following software configurations are necessary:

1. **Operating System:**
   - The system can run on **Windows**, **Linux**, or **macOS**, making it highly versatile. The application is compatible with most versions of these operating systems.
2. **Python 3:**

- **Python 3.x** should be installed on the system to execute the deepfake detection application. Python is the core programming language used in this project, and it is crucial that Python 3.x is installed to ensure compatibility with the various libraries used.

3. **Required Python Packages:**

   - The following Python packages must be installed to ensure the system works as expected. These can be installed using pip:

   - **OpenCV:** For video processing and computer vision tasks. OpenCV is essential for manipulating and analyzing images and video in real-time surveillance.

pip install opencv-python

   - **skimage (Scikit-image):** For image processing tasks, particularly for detecting features, performing transformations, and other image manipulation operations.

pip install scikit-image

   - **NumPy:** For numerical operations, as it is used to perform matrix and vector operations that are central to machine learning and image processing.

pip install numpy

   - **Tkinter:** A GUI library for Python that is used to create the graphical user interface (GUI) for the application. It is used for displaying the system's monitoring features and alert notifications in the system.

pip install tk

4. Additional Python libraries, such as **TensorFlow**, **PyTorch**, **Dlib**, and **Facenet**, might also be used for model training, face recognition, and deepfake detection.

HARDWARE REQUIREMENTS

The hardware needed for this project is not overly demanding, but the following components are essential to ensure smooth operation:

1. **Working PC or Laptop:**
    - Any standard PC or laptop capable of running the necessary software and processing video feeds will suffice. The minimum requirements would include a **dual-core processor** and **4GB RAM**.
    - For optimal performance, especially when processing high-resolution videos, a **quad-core processor** with **8GB or more RAM** is recommended.
2. **Webcam:**
    - A webcam with drivers installed is necessary for capturing video data. This could either be an external webcam or a built-in laptop camera.
    - The webcam should be capable of capturing video at **30 FPS (frames per second)** or higher for real-time detection.
3. **Flashlight/LED (If Night Usage is Required):**
    - If the system needs to be used in low-light conditions, a **flashlight** or **LED light** may be required for better visibility during video monitoring.
    - This ensures that the system can effectively detect faces and motion in darker environments.

PLATFORMS ALREADY TESTED ON:

The system has been tested on the following platforms to ensure compatibility and smooth operation:

- **Windows 7** (Tested and confirmed working)
- **Windows 10** (Tested and confirmed working)

Chapter 4
Implementation

1.Detailed Explanation of Project Implementation:

The AI-Based Deepfake Detection System was implemented using Python and Streamlit for the GUI, along with a custom-built Convolutional Neural Network (CNN) using TensorFlow/Keras for classification. The system allows users to upload an image or video, analyzes the content using a trained model, and provides a prediction on whether the content is real or fake.

The system architecture consists of:

- Frontend GUI (Streamlit): For user interaction.
- Backend Logic (Python Class DeepfakeDetector): For image/video processing and prediction.
- Model Training: CNN trained on real and fake datasets.

Algorithms, Code Snippets, and Design:

2.1 CNN Architecture

A simple CNN model was built with the following layers:

```
keras.Sequential([
    Conv2D(32, (3, 3), activation='relu'),
    MaxPooling2D(2, 2),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D(2, 2),
    Flatten(),
```

```
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(1, activation='sigmoid')
])
```

This architecture processes grayscale images of size 128x128 and predicts the probability of being fake.

2.2 Streamlit GUI

- Users choose between image or video input.
- Uploaded media is saved temporarily and passed to the backend for analysis.
- The result is displayed in real-time.

2.3Image Prediction:

```
def predict(self, img_path):
img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
prediction = self.model.predict(img)
```

Returns the probability that the uploaded image is fake.

2.4Video Prediction:

```
def predict_video(self, video_path):
if frame_count % 10 == 0:
prediction = self.model.predict(gray)
```

Analyzes one frame every 10 frames for efficiency and calculates the percentage of fake frames.

3.Challenges and Solutions:

Challenge	Solution
High training time with large datasets	Used early stopping and batch processing to reduce training duration.
Limited dataset quality	Focused on pre-cleaned and well-labeled images/videos to ensure quality data for training.
Video processing overhead	Only sampled every 10th frame to balance accuracy and performance.
Model overfitting	Implemented Dropout and EarlyStopping to generalize better on unseen data.
User interaction issues in GUI	Streamlit provided an intuitive and lightweight interface that streamlined development.

Chapter 5
RESULTS AND DISCUSSIONS

Results :

Analysis of System Output:

The provided dashboard output demonstrates the functionality of the Deepfake Detection System with several key observations.

1. User Interface: The system presents a clean navigation panel with options for Dashboard, History, About, and Settings, suggesting a well-structured user experience.
2. File Upload Functionality:
    - Supports both image and video inputs
    - Has a 200MB file size limit
    - Accepts common formats (PIG/JPEG/PNG for images)
    - Shows a sample upload of a 0.5MB JPG file
3. Detection Results:
    - The system successfully analyzed an uploaded image
    - Generated a prediction of "Real" with 1.00 confidence score
    - This high confidence score suggests the system is certain about its authentic classification
4. Technical Note:

- The output includes a deprecation warning about a parameter change
- This indicates the system is actively maintained with ongoing development

Performance Evaluation

The sample result shows promising detection capability for authentic content. However, to fully evaluate the system:

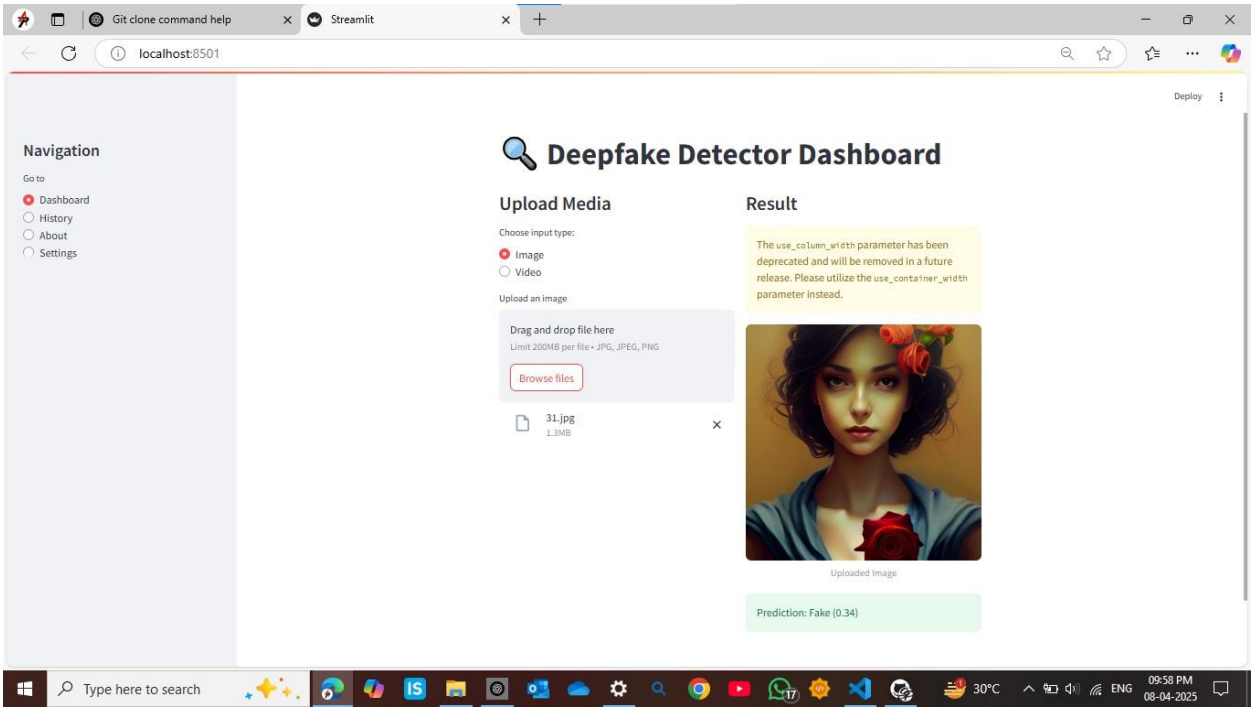
1. Testing Requirements:
    - Needs evaluation with known deepfake samples to assess false negative rate.
    - Should be tested with various image qualities and types.
    - Requires validation across different deepfake generation techniques.
2. Potential Strengths:
    - Clear confidence scoring (1.00 in this case)
    - Support for multiple media types
    - User-friendly interface
3. Areas for Improvement:
    - The deprecation warning suggests code updates may be needed.
    - No details provided about the model architecture or training data.
    - Real-world performance metrics are not shown in this sample.

Discussion

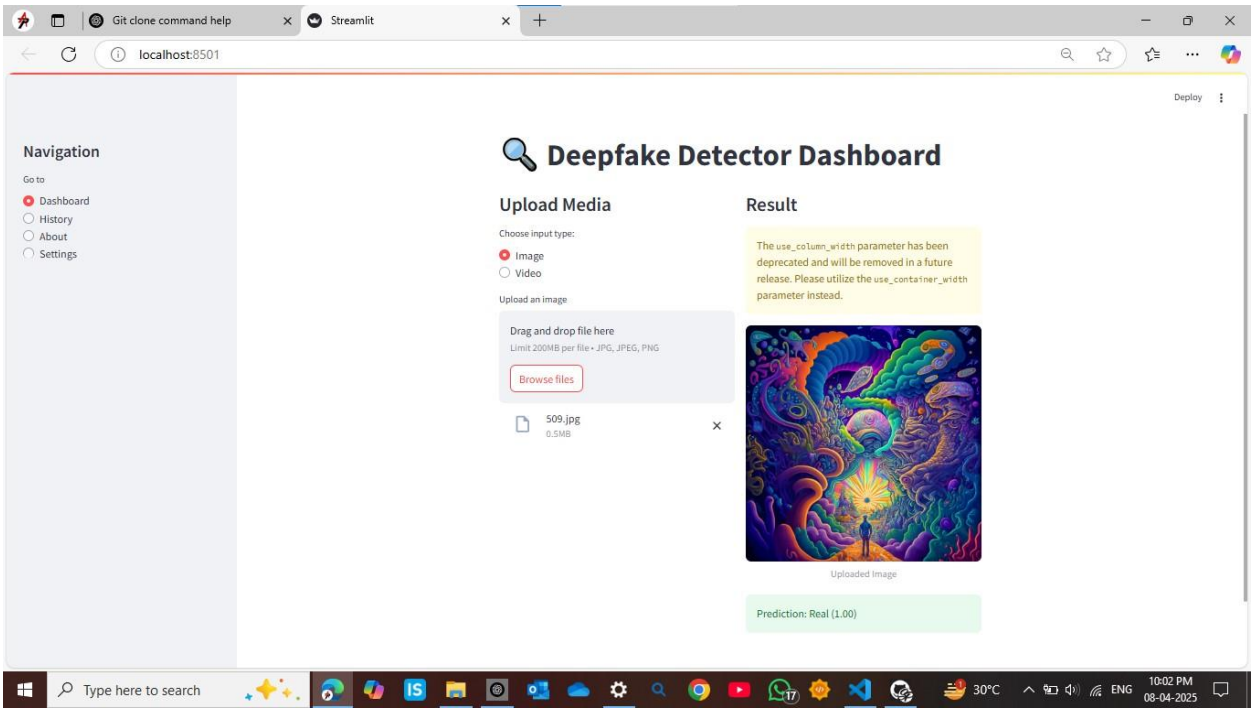
This deepfake detection system appears to provide a straightforward interface for users to verify media authenticity. The high confidence score in the sample result is encouraging, but comprehensive testing would be required to validate the system's effectiveness against sophisticated deepfakes.

Future enhancements could include:

- Detailed reporting of detection metrics
- Explanation of detection features
- Batch processing capabilities
- Integration with social media platforms



Real Image Detected Accurately



Model Architecture: Convolutional Neural Network (CNN)

Chapter 6

FUTURE WORK

The current deepfake detection model demonstrates promising results in identifying manipulated media using state-of-the-art machine learning techniques. Nonetheless, several avenues remain for further enhancement:

1. Advanced Detection Algorithms – Incorporating next-generation architectures such as Vision Transformers and Capsule Networks to improve sensitivity to subtle facial artifacts and motion inconsistencies.

2. Real-Time Capabilities – Optimizing computational performance for real-time deepfake detection in live video feeds and social media content streams.

3. Improved Generalization – Strengthening the model's robustness against deepfakes generated using newer, more advanced GANs and diffusion-based models.

4. Multimodal Fusion – Integrating audio analysis and metadata alongside visual cues for a more comprehensive and accurate detection mechanism.

5. Model Explainability – Designing interpretability features to provide insights into the decision-making process, helping users understand the rationale behind deepfake classifications.

6. Scalability and Deployment – Expanding system architecture to handle large-scale applications, including integration into social networks, media verification tools, and law enforcement frameworks.

CONCLUSION

The emergence of deepfake technology—driven by advancements in AI and generative models—poses serious threats to individual privacy, digital integrity, and societal trust. These synthetic media can be weaponized for misinformation campaigns, identity fraud, and reputational harm.

This project introduces an AI-Based Deepfake Detection System that harnesses deep learning models to accurately identify forged video content. By analyzing visual anomalies, facial irregularities, and digital artifacts, the system offers an effective solution to detect and deter malicious deepfakes.

Looking ahead, the integration of real-time analysis, multimodal verification, and scalable deployment will be key to staying ahead of evolving threats. The project underscores the importance of continuous innovation in AI security tools to preserve the authenticity of digital communication in the face of increasingly deceptive technologies.

REFERENCES

1. Rössler, A., Cozzolino, D., Verdoliva, L., et al. (2019). FaceForensics++: Learning to Detect Manipulated Facial Images. Retrieved from arxiv.org/abs/1901.08971.

2. Dolhansky, B., Gupta, D., Sun, Z., et al. (2020). The DeepFake Detection Challenge Dataset. Retrieved from arxiv.org/abs/2001.04671.

3. Li, Y., Zhang, Y., & Yu, Y. (2019). XceptionNet: Deep Learning for Deepfake Detection. Retrieved from arxiv.org/abs/1610.02357.

4.Dang, H. H., et al. (2020). Audio-Visual Deepfake Detection. IEEE Transactions on Multimedia.

5. "VoxCeleb: Speaker Recognition in the Wild". Retrieved from vgg-data.robots.ox.ac.uk.