A

Major Project Report

on

DEVELOPMENT OF AI/ML BASED SOLUTION FOR DETECTION OF FACE-SWAP BASED DEEP FAKE VIDEOS

Submitted in Partial Fulfillment of
the Requirements for the Fourth Year
of

Bachelor of Engineering

in

Computer Engineering

to

**Kavayitri Bahinabai Chaudhari
North Maharashtra University, Jalgaon**

Submitted by

Prajwal Chaudhari
Vaishnavi Patil
Akansha Patil
Sakshi Marathe

Under the Guidance of

Dr.A.D.Waghmare

DEPARTMENT OF COMPUTER ENGINEERING
SSBT's COLLEGE OF ENGINEERING AND TECHNOLOGY,
BAMBHORI, JALGAON - 425 001 (MS)
2024 - 2025

SSBT's COLLEGE OF ENGINEERING AND TECHNOLOGY,
BAMBHORI, JALGAON - 425 001 (MS)
DEPARTMENT OF COMPUTER ENGINEERING

CERTIFICATE

This is to certify that the major project entitled *Development of AI/ML based solution for detection of face-swap based deep fake videos*, submitted by

Prajwal Chaudhari
Vaishnavi Patil
Akansha Patil
Sakshi Marathe

in partial fulfillment of the third year of *Bachelor of Engineering* in *Computer Engineering* has been satisfactorily carried out under my guidance as per the requirement of Kavayitri Bahinabai Chaudhari North Maharashtra University, Jalgaon.

Date: 28 April 2025
Place: Jalgaon

Dr.A.D.Waghmare
Guide

Dr. Manoj E. Patil
Head

Prof. Dr. G.K.Patnaik
Principal

Acknowledgement

We would like to express our deep gratitude and sincere thanks to all who helped us in completing Alumini Association Platform project report successfully. Many thanks to almighty God who gave us the strength to do Project. Our sincere thanks to Principal Prof. Dr. Girish K. Patnaik for providing the facilities to complete this Project report. We would like to express our gratitude and appreciation to all those who gave us the possibility to complete this report. A special thanks to Prof. Dr. Manoj E. Patil, Head of the Department, whose help, stimulating suggestions and encouragement, helped us in writing the report. We would also like to thank to Dr.A.D.Waghmare Project Guide, who has given his full effort in guiding us and achieving the goal as well as his encouragement to maintain the progress in track. I am also sincerely thankful to Mrs. Shital Patil, Incharge of Project, for her valuable suggestions and guidance. We would also like to appreciate the guidance given by other supervisor that has improved our presentation skills by their comments and tips. Last but not the least, we are extremely thankful to our parents and friends without whom it could not reach its successful completion.

Vaishnavi Deepak patil
Akansha Yashwant Patil
Prajwal Ganesh Chaudhari
Sakshi Nitin Marathe

Contents

List of Figures

Abstract

The growing computation power has made the deep learning algorithms so powerful that creating a indistinguishable human synthesized video popularly called as deep fakes have became very simple. Scenarios where these realistic face swapped deep fakes are used to create political distress, fake terrorism events, revenge porn, blackmail peoples are easily envisioned. New deep learning-based method that can effectively distinguish AI-generated fake videos from real videos.Method is capable of automatically detecting the replacement and reenactment deep fakes.Trying to use Artificial Intelligence to fight Artificial Intelligence. System uses a Res-Next Convolution neural network to extract the frame-level features and these features and further used to train the Long Short Term Memory based Recurrent Neural Network to classify whether the video is subject to any kind of manipulation or not, i.e whether the video is deep fake or real video. To emulate the real time scenarios and make the model perform better on real time data, evaluate method on large amount of balanced and mixed data-set prepared by mixing the various available data-set like Face-Forensic++, Deepfake detection challenge, and Celeb-DF. Also show how system can achieve competitive result using very simple and robust approach.

Keywords:
Res-Next Convolution neural network, Recurrent Neural Network , Long Short Term Memory, Computer vision

Chapter 1

Introduction

In the world of ever growing Social media platforms, Deepfakes are considered as the major threat of the AI. There are many Scenarios where these realistic face swapped deepfakes are used to create political distress, fake terrorism events, revenge porn, blackmail peoples are easily envisioned.Some of the examples are Brad Pitt, Angelina Jolie nude videos.In project using AI to fight AI.Deepfakes are created using tools like FaceApp and Face Swap , which using pre-trained neural networks like GAN or Auto encoders for these deepfakes creation. Method uses a LSTM based artificial neural network to process the sequential temporal analysis of the video frames and pre-trained Res-Next CNN to extract the frame level features. ResNext Convolution neural network extracts the frame-level features and these features are further used to train the Long Short Term Memory based artificial Recurrent Neural Network to classify the video as Deepfake or real. To emulate the real time scenarios and make the model perform better on real time data, trained method with large amount of balanced and combination of various available dataset like FaceForensic, Deepfake detection challenge, and Celeb-DF.

Section 1.1 introduces the Background of the topic. The Motivation behind the project is in Section 1.2. Section 1.3 defines the Problem Statement of the system. The Scope of the project is detailed in Section 1.4, followed by the Objectives in Section 1.5. A summary of the Organization of the Report is presented in Section 1.6.

1.1 Background

The project Deepfake videos are synthetic media created using artificial intelligence , where a person's face or voice is manipulated to resemble someone else. These videos are generated using techniques like Generative Adversarial Networks or deep learning algorithms, which can mimic realistic facial expressions, movements, and even speech patterns. While deepfake technology has applications in entertainment and education, it poses significant threats when misused for misinformation, fraud, and privacy invasion.

1.2 Motivation

The increasing sophistication of mobile camera technology and the ever growing reach of social media and media sharing portals have made the creation and propagation of digital videos more convenient than ever before. Deep learning has given rise to technologies that would have been thought impossible only a handful of years ago. Modern generative models are one example of these, capable of synthesizing hyper realistic images, speech, music, and even video. These models have found use in a wide variety of applications, including making the world more accessible through text-to-speech, and helping generate training data for medical imaging.

Like any trans-formative technology, this has created new challenges. So-called "deep fakes" produced by deep generative models that can manipulate video and audio clips. Since their first appearance in late 2017, many open-source deep fake generation methods and tools have emerged now, leading to a growing number of synthesized media clips. While many are likely intended to be humorous, others could be harmful to individuals and society. Until recently, the number of fake videos and their degrees of realism has been increasing due to availability of the editing tools, the high demand on domain expertise.

Deep fake detection is very important. Describe a new deep learning-based method that can effectively distinguish AI- generated fake videos (Deep Fake Videos) from real videos. It's incredibly important to develop technology that can spot fakes, so that the deep fakes can be identified and prevented from spreading over the internet.

1.3 Problem Definition

Convincing manipulations of digital images and videos have been demonstrated for several decades through the use of visual effects, recent advances in deep learning have led to a dramatic increase in the realism of fake content and the accessibility in which it can be created. These so-called AI-synthesized media (popularly referred to as deep fakes).Creating the Deep Fakes using the Artificially intelligent tools are simple task. But, when it comes to detection of these Deep Fakes, it is major challenge. Already in the history there are many examples where the deepfakes are used as powerful way to create political tension, fake terrorism events, revenge porn, blackmail peoples etc. It becomes very important to detect these deepfake and avoid the percolation of deepfake through social media platforms. Detecting the deep fakes using LSTM based artificial Neural network.

1.4 Scope

There are many tools available for creating the deep fakes, but for deep fake detection there is hardly any tool available. Approach for detecting the deep fakes will be great contribution in avoiding the percolation of the deep fakes over the world wide web. 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 detection's. Even big application like Whats App, Facebook can integrate the 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, input state diagram, Major inputs, and outputs are described without regard to implementation detail.

1.5 Objectives

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

1.6 Selection of Life cycle model for development

The software development life cycle model selected for this project is the Waterfall Model. Here is a diagram of the Waterfall Model for deepfake video detection using AI/ML. Each stage is clearly represented, highlighting the systematic approach to building and deploying the detection system. Figure 1.1 shows waterfall model

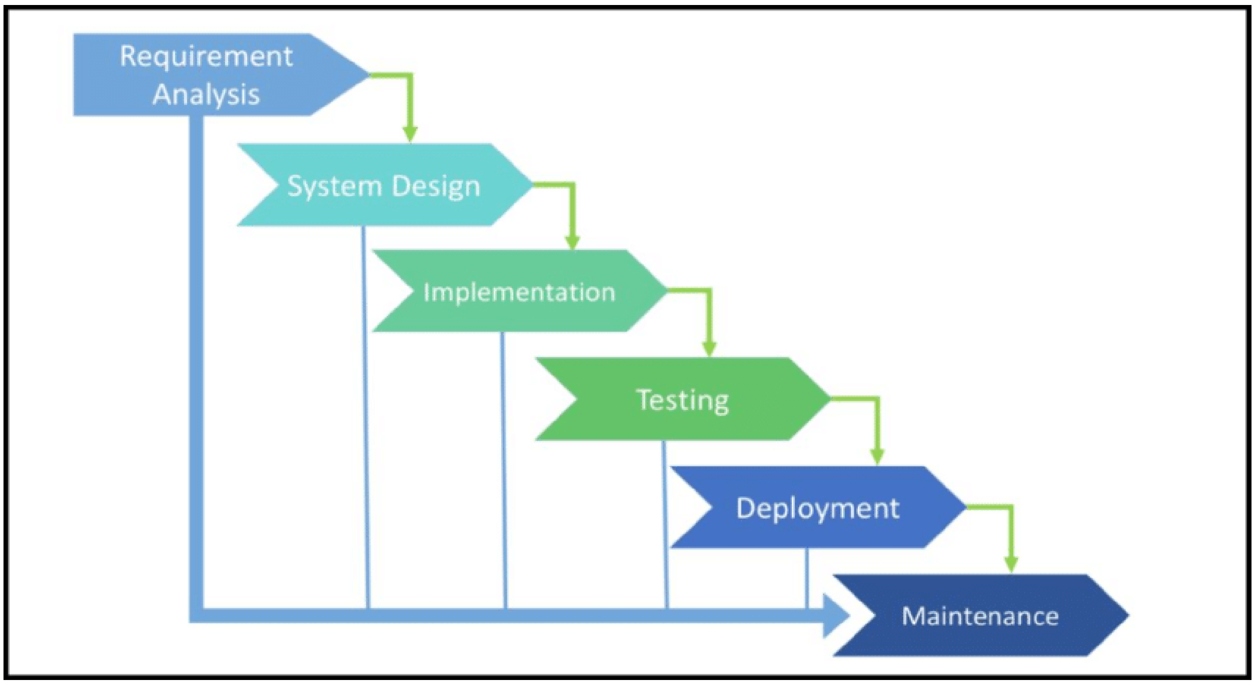


Figure 1.1: Waterfall Model

Waterfall Model is best suited model for project.

- Because requirements are easily understandable and defined
- We can define requirements in early stage of development
- User involvement in all phases is not necessary
- Limited user's participation

1.7 Organization of Report

Chapter 1: Entitled as Introduction describes the details about Background, Problem Definition, Scope and Objective of the project, Identification of Software Development Process Model and Organization of report.

Chapter 2: Entitled as Project Planning and Management consists of details about the Feasibility Study, Risk Analysis, Project Scheduling, Effort Allocation and Cost Estimation of the project.

Chapter 3: Entitled as Analysis describes in detail, the Requirement Collection and Identification, H/w and S/w Requirements, Functional and Non-Functional Requirements and a Software Requirements Specification(SRS).

Chapter 4:Includes design about System Architecture, Data Flow Diagram and various UML Diagrams.

Chapter 5:Conclusion.

Chapter 6:Bibliography.

1.8 Summary

The AI/ML-based Deepfake Video Detection Software provides an automated solution to identify manipulated or synthetic media using advanced machine learning techniques. Addressing growing concerns around misinformation, identity fraud, and media authenticity, the system offers a robust and efficient method for detecting deepfakes in video content. The software analyzes facial expressions, visual inconsistencies, and audio-visual mismatches to accurately classify videos as real or fake. Cost-effective implementation is achieved through the use of open-source frameworks and pre-trained models, with a focus on user-friendly interfaces and compatibility with digital forensics or content moderation platforms. Developed following the structured Waterfall Model, the project defines requirements early, minimizing user intervention during development. Key phases include requirement gathering, model selection and training, system design, and risk mitigation. Provisions for future enhancements ensure the system can adapt to evolving deepfake techniques and broader real-world use cases.In net chapter,project planning and management is described.

Chapter 2

Project Planning And Management

Feasibility study confirms that the AI/ML-based Deepfake Video Detection Software is viable economically, operationally, and technically by leveraging open-source tools such as Python, TensorFlow, and OpenCV. The system offers ease of use, scalability, and requires only moderate hardware resources, making it accessible for a wide range of users. Risk analysis identifies key technical and ethical risks, including false positives, adversarial attacks, and misuse of detection results, with mitigation strategies such as model tuning, continuous updates, and clear usage policies. Project scheduling is guided by well-defined phases, timelines, and milestones to ensure on-time delivery. Effort allocation is optimized across tasks like dataset preparation, model training, system integration, and testing. Cost estimation considers factors such as model complexity, resource requirements, development duration, and operational expenses, ensuring efficient budgeting and effective project execution.

Section 2.1 presents the Feasibility Study of the system. The Risk Analysis is discussed in Section 2.2. Section 2.3 outlines the Project Scheduling. The Effort Allocation is explained in Section 2.4. A detailed Cost Estimation is provided in Section 2.5.

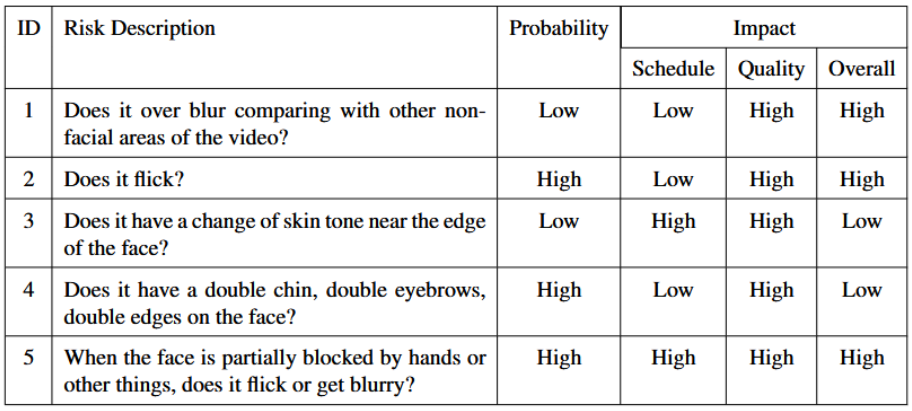
2.1 Risk Analysis

In Deepfakes, it creates a mask on the created face so it can blend in with the target video. To further eliminate the artifacts. Table 2.1 shows risk description and table 2.2 shows risk probability definitions.

2.2 Project Scheduling

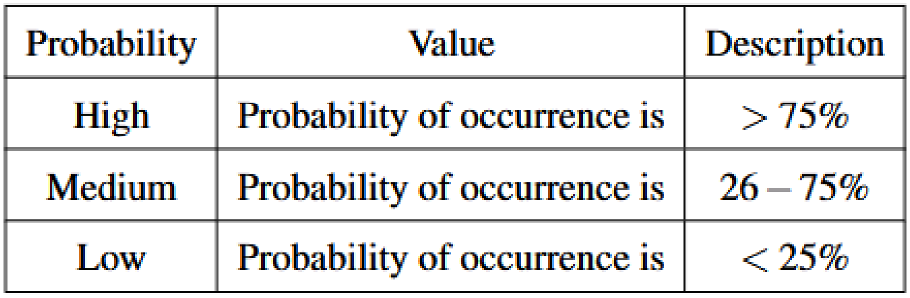
Generally, project scheduling can be stated as the estimated time required for any project from its time of beginning to the end of the project. In detail, for every task, there is a deadline because all the tasks for the completion of project are planned earlier. So that, each task

Table 2.1: Risk Description



ID	Risk Description	Probability	Impact		
			Schedule	Quality	Overall
1	Does it over blur comparing with other non-facial areas of the video?	Low	Low	High	High
2	Does it flick?	High	Low	High	High
3	Does it have a change of skin tone near the edge of the face?	Low	High	High	Low
4	Does it have a double chin, double eyebrows, double edges on the face?	High	Low	High	Low
5	When the face is partially blocked by hands or other things, does it flick or get blurry?	High	High	High	High

Table 2.2: Risk Probability Definitions



Probability	Value	Description
High	Probability of occurrence is	$> 75\%$
Medium	Probability of occurrence is	$26 - 75\%$
Low	Probability of occurrence is	$< 25\%$

is scheduled to certain time limit. In short, in project management, listing of projects milestones, activities and all from starting to end date, are considered in the project scheduling. A schedule is generally used in the project planning and management of the project with some kind of attributes as budget, task allocation and duration, resource allocation and all. Table 2.3 shows task scheduling for the project

Table 2.3: Task Scheduling for the project

Task	Start Date	End Date
Selection of title	20 July 2024	09 Aug 2024
Gathering information	09 Aug 2024	30 Aug 2024
Project discussion	1 Sep 2024	15 Sep 2024
Planning/ requirement gathering and analysis	15 Sep 2024	07 Oct 2024
Implementation	15 Jan 2025	30 Jan 2025
Documentation	Feb 2025	3 Mar 2025
Result	18 Mar 2025	20 April 2025

2.3 Effort Allocation

Effort Allocation is necessary so every team member can give its best to the project. Project was divided into smaller module and task form, for simplification and easy understanding of project overall. Some modules include every team associate's presence to take advantage of team decision taking skills, and some task include some individual member to work on it with precision. Divided the project into 6 modules. Table 2.4 shows effort allocation.

- 1. Gathering of Information
- 2. Planning/Requirement Analysis
- 3. Selection of Life Cycle
- 4. Planning and Management
- 5. Analysis Design UML

Table 2.4: Effort Allocation

	Vaishnavi	Sakshi	Akansha	Prajwal
Gathering of information	✓	✓	✓	✓
planning / requirement analysis		✓	✓	✓
selection of life cycle	✓			
planning and management	✓	✓		✓
analysis and design			✓	✓

2.4 Cost Estimation

The cost estimation for the project is performed using a simplified approach similar to the basic COCOMO model, which helps in estimating the effort, time, and cost required for software development. Table 2.5 shows the total number of persons working on the project is 4, with each person contributing approximately 3 hours per day over a period of 4 months. The actual working hours are calculated as 430 hours in total. The cost per hour is assumed to be Rs 40, resulting in an individual project cost of Rs 8000 (i.e., 430 hrs $\times$ Rs 50). For 4 team members, the total estimated manpower cost becomes Rs 21500. Additional costs such as tools, licenses, and hardware amount to Rs 3200. Hence, the Total Estimated Project Cost sums up to Rs 24700. This method provides a basic estimation and is comparable to the COCOMO Organic mode, which is best suited for small teams working on relatively simple software projects with well-understood requirements.

Table 2.5: Cost Estimation

1	Total number of persons working on the project	4 Person
2	Time Taken (in months)	4 Month
3	Total Time Allocated per Day (in terms of hrs)	3 hrs
4	Actual Working Hours	430 hrs
5	Cost per hour	Rs 40
6	Total Estimated Project Cost for a Person	Rs 5,375
7	Total Estimated Project Cost For 4 Person	Rs 21,500
8	Total Estimated Project Cost For Total Project	Rs 24,700

2.5 Summary

In Project Planning and Management chapter, the project Planning and Management of the project is described. In next chapter, the project analysis is described.

Chapter 3

Analysis

As a result of deepfakes, you can open up additional opportunities in computerized media, virtual reality, mechanics, education, and numerous other fields. In another context, they represent innovations that can ruin and undermine the whole society. Developed a model that combines CNNs and LSTMs for the deepfake video identification task. LSTMs can deal with sequences of consecutive frames, whereas CNNs are good at learning local highlights. In model leverages this combined limit to associate each pixel in a picture and comprehend nonlocal highlights. When preparing and grouping, gave equal emphasis to the preprocessing of the information.

Section 3.1 focuses on Requirement Collection and Identification. The Software Requirement Specification (SRS) is detailed in Section 3.2.

3.1 Requirement Collection and Identification

Requirement collection is the process which is used to gather, analyze, and documentation and reviews the requirements. Requirements describe what the system will do in place of how. Practical application, most projects will involve some combination of these various methods in order to collect a full set of useful requirements. Requirements collection is initiated when the project need is first identified and the project “solution” is to be proposed. Requirements refinement continues after the project is “selected” and as the scope is defined, aligned and the scope is approved.The system will require only images of the road to be analysed which will be uploaded by user either through the web-application or CLI.

3.2 Software Requirements Specifications (SRS)

Software Specification will provide a broad understanding of the requirement specification of this system. Also, understand features of system along with the requirements. Software requirement specification documents guide the developers the development process and it will help to reduce the ambiguity of the requirements provided by the end-user.

3.2.1 Product Feature

The product features are high level attributes of a software or product such as software performance, user-friendly interface, security portability, etc. These attributes are defined according to the product.
They are as follows:

- Detects deepfake patterns, including facial inconsistencies, unnatural movements, and blending artifacts.
- Analyzes video frames and metadata for manipulation signs.
- It aims to ensure authenticity, combat misinformation, and provide reliable tools for video verification.

3.2.2 Operating Environment

1. Hardware Requirement :
    - CPU: Multi-core processor (Intel i5/AMD Ryzen or better).
    - RAM: Minimum 8 GB or higher.
    - Storage: SSD with 50 GB free space.
2. Software Requirement :
    - Programming Languages: Python3 , JavaScript
    - Programming Frameworks: PyTorch , Django
    - IDE: Google Colab, Jupyter Notebook, Visual Studio Code
    - Libraries: torch, torchvision, os, numpy, cv2, matplotlib, face recognition, json, pandas, copy, glob, random, sklearn

3.2.3 Functional Requirements

Functional requirements are the functions which are expected from the software or platform. Functional requirements along with requirement analysis help identify missing requirements. They help clearly define the expected system service and behavior.
Functional requirements are as follows:

- Allows video uploads via drag-and-drop or file selection.
- Detects blending artifacts and unnatural transitions between frames.
- Displays a confidence score indicating the likelihood of the video being fake.
- Intuitive and user-friendly interface for uploading videos, viewing results, and downloading reports.

3.2.4 Non-Functional Requirements

Non-functional Requirement is mostly quality requirement.Non-functional requirements (NFRs) define the quality attributes and operational standards of a software system. For an AI/ML-based fake video detection software
Non-functional requirements are as follows:

- The software should detect fake videos within a specified time frame
- Ensure user-uploaded videos are encrypted during transit and at rest
- Use secure user authentication mechanisms (e.g., two-factor authentication) for access.

3.2.5 External Interfaces

- ■ User interface
  The proposed system has several options for users to interact with. Following are the user interfaces available:

- Web-application (GUI)
- Application Programming Interface(API)
  The application programming interface will be available so that the users will be able to upload video.

- ■ Software interface
  Designing a software interface for an AI/ML-based fake video detection system involves several considerations, including user-friendliness, clarity of results, and providing actionable insights.

3.3 Summary

In Analysis chapter, Requirement collection phase and SRS provide a clear and structured framework for the development of the AI/ML-based deepfake video detection software. This comprehensive plan defines the system's features, technical requirements, and external interfaces, ensuring a smooth development process and successful project execution. It outlines the core functionalities such as video ingestion, deepfake analysis using machine learning models, result visualization, and user feedback mechanisms. By establishing performance benchmarks, input/output specifications, and integration points, the requirement analysis ensures the software can accurately detect forged content and operate effectively across diverse use cases and media formats.In next chapter, Designing for project is described.

Chapter 4

Design

Design is the activity to design and model the various component of software system. The system design provides the understanding and procedural details necessary for implementing the system. Design is helpful for a better understanding of the project. Design contains the UML diagrams, data flow diagrams. UML is a modeling language which is used to document the object-oriented analysis and design.

Section 4.1 explains the System Architecture of the group face recognition system. The Data Flow Diagram is presented in Section 4.2, illustrating the flow of data within the system. Section 4.3 provides the UML Diagrams, which include visual representations like use case, sequence, collaboration, class, state chart, component, and deployment diagrams.

4.1 System Architecture

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 4.1. In the development phase, taken a dataset, preprocessed the dataset and created a new processed dataset which only includes the face cropped videos.

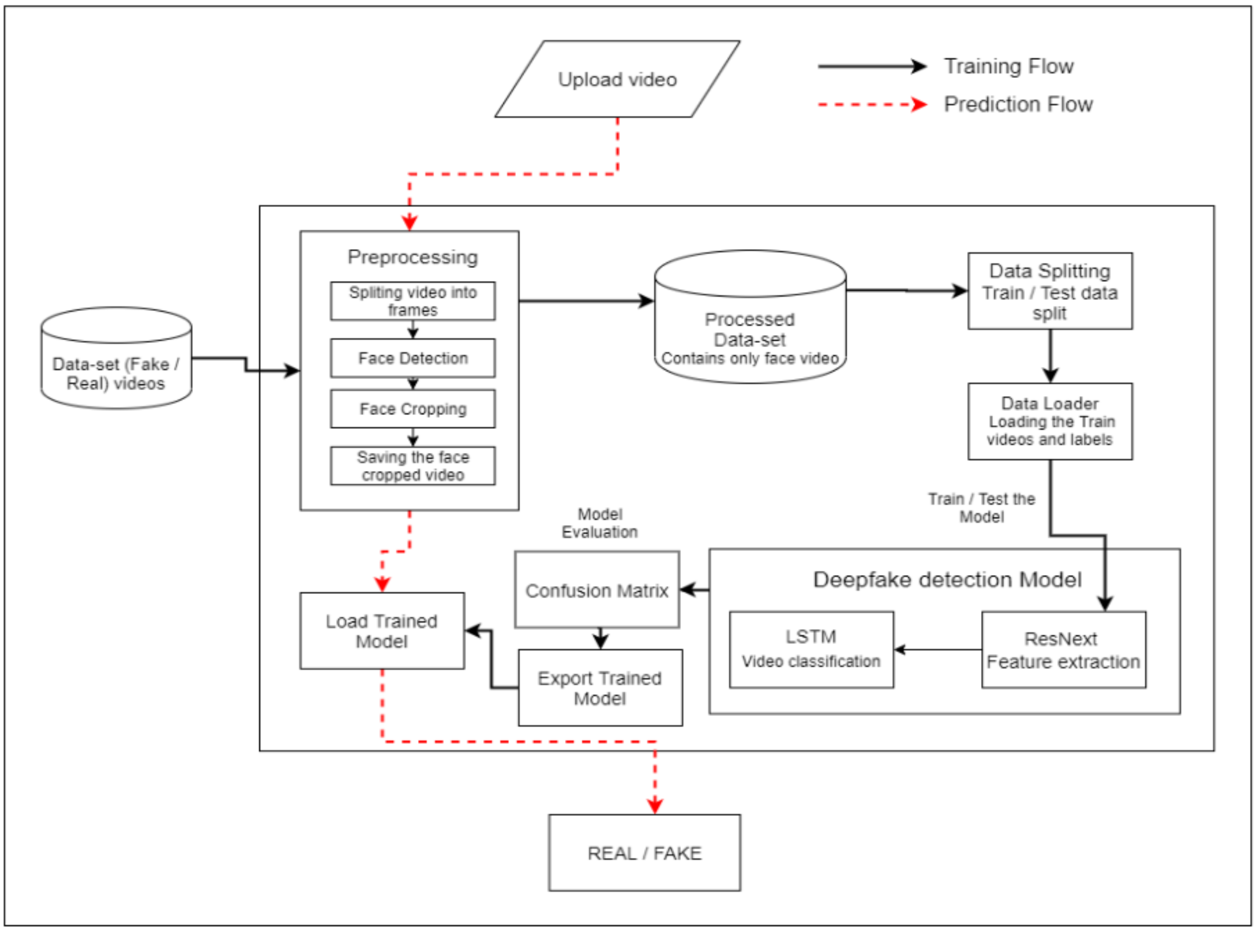

Upload video
Training Flow
Prediction Flow
Preprocessing
Spliting video into frames
Face Detection
Face Cropping
Saving the face cropped video
Processed Data-set Contains only face video
Data Splitting Train / Test data split
Data Loader Loading the Train videos and labels
Data-set (Fake / Real) videos
Train / Test the Model
Model Evaluation
Confusion Matrix
Deepfake detection Model
LSTM Video classification
ResNext Feature extraction
Load Trained Model
Export Trained Model
REAL / FAKE

Figure 4.1: System Architecture

4.2 Data Flow Diagram

A data flow diagram (DFD) is a graphical representation of the 'flow' of data through an information system, modelling its process aspects. A DFD is often used as a preliminary step to create an overview of the system, which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design). A DFD shows what kind of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored.

4.2.1 Level 0 DFD

Level 0 contains one input and one output. The system provides information to the user means system is input and the user is output. Figure 4.2 shows Level 0 DFD of project.

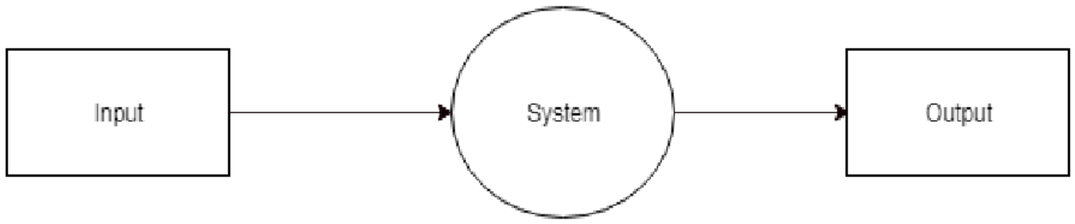


Figure 4.2: 0 Level Data Flow Diagram

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

4.2.2 Level 1 DFD

A level 1 DFD notates each of the main sub-processes together form the complete system. A level 1 DFD as an "exploded view" of the context diagram. Figure 4.3 shows Level 1 DFD of project.

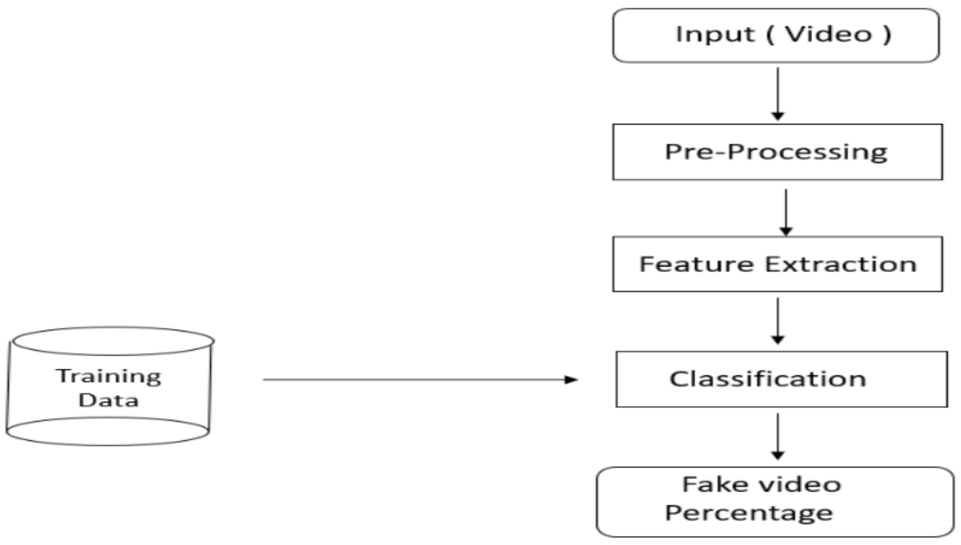


Figure 4.3: 1 Level Data Flow Diagram

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

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

4.2.3 Level 2 DFD

A level 2 data flow diagram offers a more detailed look at the processes that make up an information system a level 1 DFD does. Level 2 DFD can be used to plan or record the specific makeup of a system. Figure 4.4 shows Level 2 DFD of project.

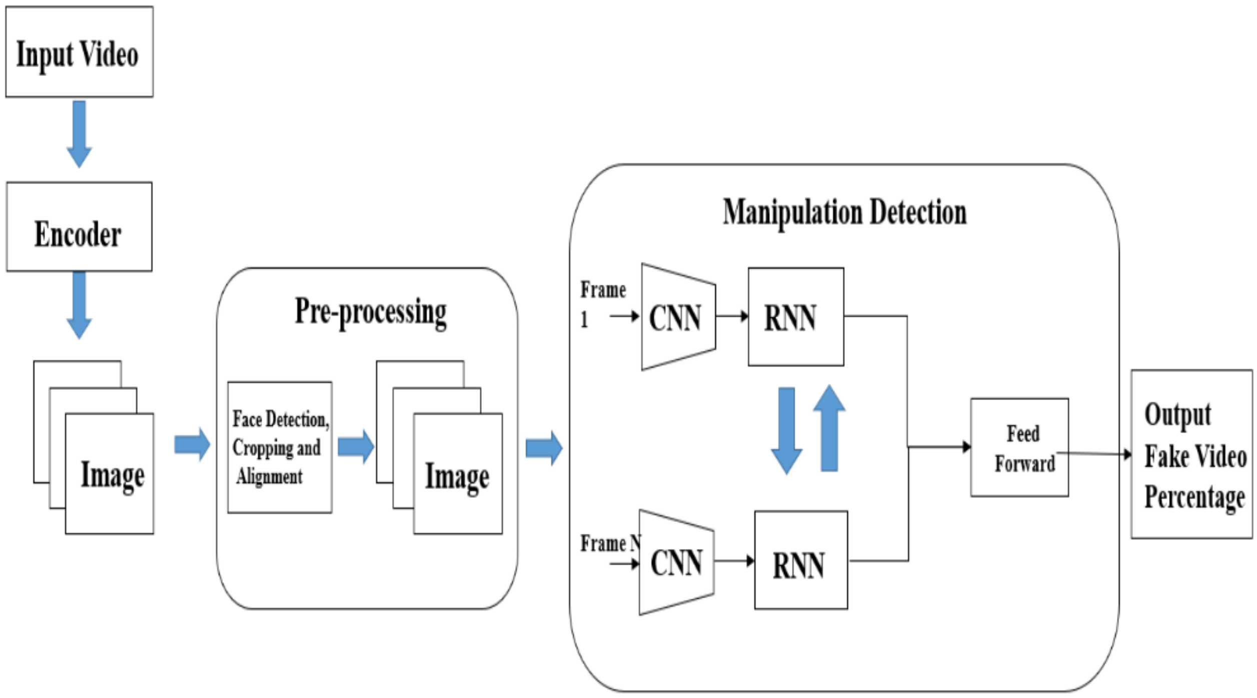


Figure 4.4: 2 Level Data Flow Diagram

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

4.3 UML Diagrams

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

4.3.1 Use Case Diagram

Use case diagram shows the interaction between Use case which represents system functionality and actor which represent the people or system. Figure 4.5 shows use case diagram.

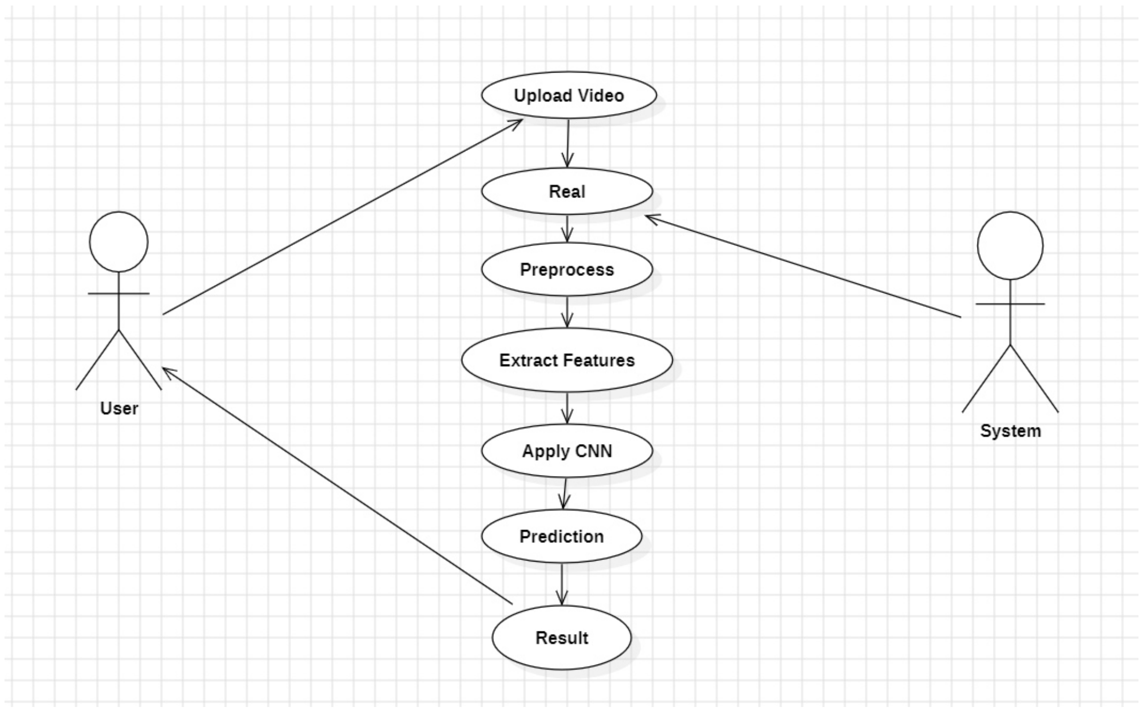


Figure 4.5: Use Case Diagram deepfake videos detection

4.3.2 Sequence Diagram

The sequence diagram shows the flow of functionality through Use case. A sequence diagram is a type of interaction diagram because it describes how and in what order a group of objects works together. These diagrams are used by software developers and business professionals to understand requirements for a new system or to document an existing process.
Figure 4.6 shows sequence diagram.

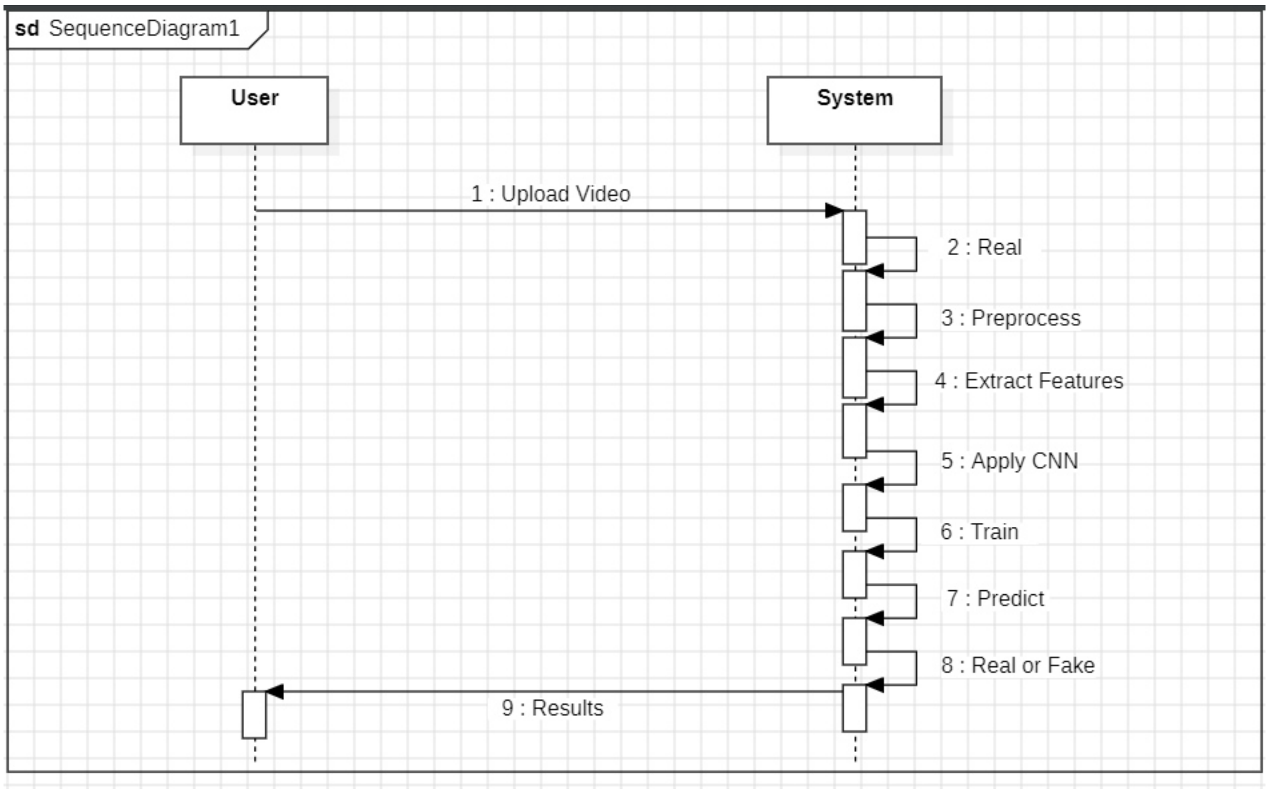


Figure 4.6: Sequence Diagram For deepfake video detection

4.3.3 Activity Diagram

The activity diagram for detecting deepfake videos using AI/ML techniques. It illustrates the workflow from input video to final detection results with detailed steps and decision points. Figure 4.7 shows activity diagram

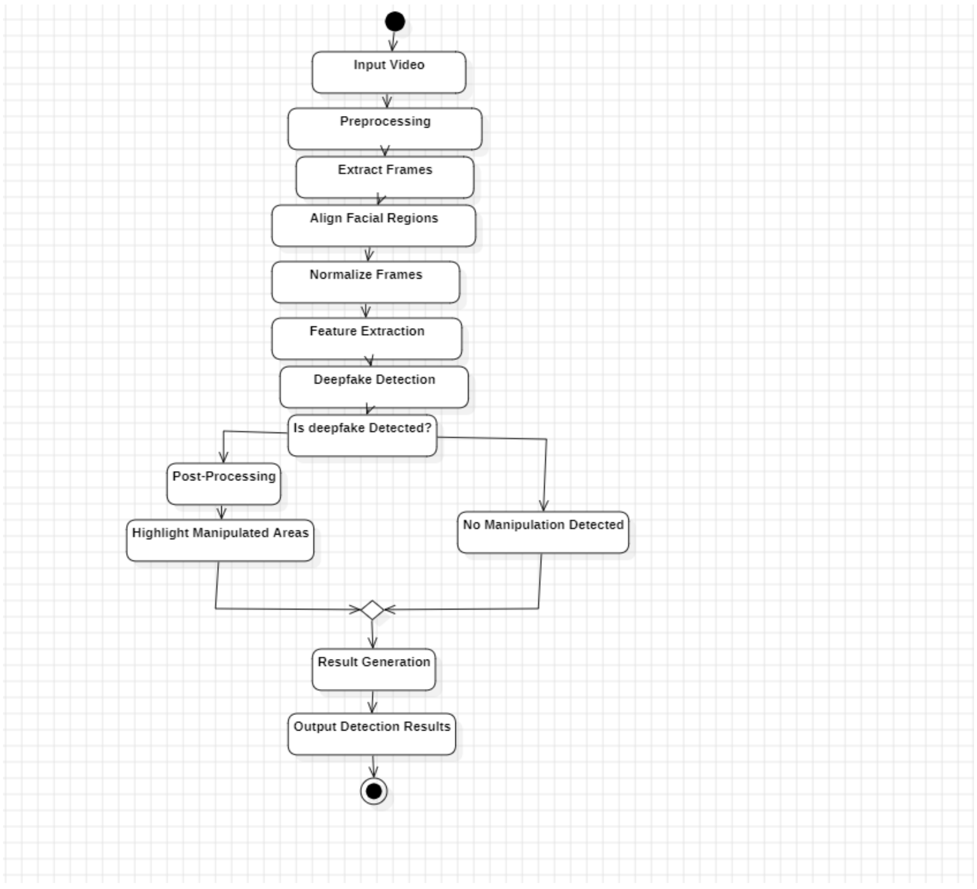


Figure 4.7: Activity Diagram

4.4 Summary

This chapter contains important aspects of designing the model, such as the System Architecture, Data Flow Diagrams (DFDs), and UML diagrams, which serve as the blueprints of the system. These diagrams highlight the architecture, flow, and sequential execution of processes involved in deepfake detection. Designing is one of the most crucial processes for building a project and also plays a significant role in the coding and implementation phase. The architecture illustrates components such as video input modules, preprocessing layers, deep learning inference engines, and result interpretation units. UML diagrams are useful for understanding the system's components, interactions between modules, and overall behavior of the AI pipeline. In next chapter,Coding and Implementation are described.

Chapter 5

Coding/ Implementation

To detect deepfake videos, AI models first extract frames from videos and detect faces. These faces are processed using convolutional neural networks like Xception or ResNet to learn subtle differences between real and fake videos. Sometimes, sequences of frames are fed into CNN+LSTM or transformer models to capture temporal inconsistencies. After training on labeled real and fake datasets, the model predicts whether a video is genuine or manipulated based on aggregated frame-level results.

Section 5.1 outlines the Implementation phase of the system. The algorithms used for deep fake detection in Section 5.1 Section 5.2 explains the flow of system development, detailing processes like user registration, dataset preparation, and deep fake video analysis.Section 5.3 presents the software and hardware components required for development. Section 5.4 Finally, provides a summary of the overall implementation process.

5.1 Algorithm

Algorithm for AI/ML based deep-fake video detection is as follows:

Step 1: User/Video Submission

```
video_zip ← receive input from user
extract video_files from video_zip
for each video in video_files:
save video to Uploaded_Videos/
store_video_metadata(video_id, uploader_name, timestamp, etc.)
```

Purpose: Accept videos from users and store them for analysis.

Step 2: Frame Extraction and Preprocessing

```
frames_list ← []
for each video in Uploaded_Videos/:
extract frames at intervals (e.g., 1 frame per second)
for each frame:
resize and normalize
append to frames_list
```

Purpose: Prepare video data for model input

Step 3: Load Pre-trained Deepfake Detection Model

```
load deepfake_detection_mode(e.g., EfficientNet, XceptionNet, CNN-LSTM)
```

Purpose: Use a deep learning model trained on datasets like FaceForensics++, DFDC, or Celeb-DF

Step 4: Frame Analysis for Deepfake Detectiong

```
results ← []
for each frame in frames_list:
prediction = deepfake_detection_model.predict(frame)
results.append(prediction)  Prediction: ”Real” or ”Fake” with confidence score
```

Purpose: Run inference to detect deepfakes frame-by-frame.

Step 5: Aggregate Frame-level Results

```
fake_count = count of predictions labeled ”Fake”
real_count = count of predictions labeled ”Real”
deepfake_confidence = fake_count / (fake_count + real_count)
if deepfake_confidence ¿ THRESHOLD:
label video as ”Deepfake”
else:
label video as ”Authentic”
```

Purpose: Combine frame-level predictions to give a video-level decision.

Step 6: Update Detection Results in Database

store_detection_result(video_id, deepfake_confidence, final_label)
Purpose: Save detection outcomes for each video.

Step 7: Generate Report or Alert

report = generate_detection_report(video_id)
display(report)
if final_label == "Deepfake":
alert user/admin with reason and confidence level Purpose: Provide user feedback and take necessary action if a deepfake is detected.

5.2 Flow of System Development

The system follows a structured workflow to ensure seamless interaction between the database, user interface, and the deepfake detection engine. Below is the step-by-step process:

- User Registration

  Researchers and analysts must first register on the web portal. Users upload a zip file containing original video samples (non-deepfake) for dataset preparation.

- Dataset Preparation

  The system processes the uploaded videos using the dataset_prepare.py script. This script extracts key frames and facial features, organizing them into a dataset for model training.

- Model Training

  Once sufficient data is collected, the admin triggers the training process using trainModel.py. The model is trained to distinguish between real and deepfake videos by learning from labeled datasets.

- Video Analysis

  Users upload MP4 videos suspected to be deepfakes. The detectFake.py module analyzes the videos frame-by-frame, detects manipulations, and outputs a confidence score indicating the likelihood of a deepfake.

- Result Reporting and Management

  Users can view detailed analysis results in the “Detection Reports” section. The system stores results, timestamps, and detection metrics for each video, allowing users to track, compare, and manage reports from their dashboards.

5.3 Software and Hardware for Development

- Software Requirements

  The proposed system is developed using the following software: Programming Language: Python 3, Javascript - Framework: PyTorch, Django, -IDE: Google Colab, Jupyter Notebook, Visual Studio Code -Frontend Technologies: HTML, CSS, JavaScript (for web portal) - Libraries and Dependencies: OpenCV, NumPy, Pandas, TensorFlow (for face recognition and data processing)

- Hardware Requirements

  - Intel Core i3 or higher processor - Minimum 4GB RAM - At least 50GB of available storage space - Camera (720p or above)

5.4 Summary

In this chapter, we discussed the algorithms, system workflow, and development environment of the Group Face Recognition-based Attendance System. The next chapter will focus on system testing and performance evaluation.

Chapter 6

Testing

Testing is a critical phase in the development of the AI/ML-based deepfake detection software. It ensures that the model accurately identifies manipulated videos, adheres to the defined requirements, and maintains performance across diverse and challenging scenarios. Various test cases are designed and executed to uncover potential errors, assess detection reliability, and enhance model robustness. Testing involves the evaluation of both individual video frames and complete video sequences, ensuring that the system can generalize well to different types of deepfakes and sources. Comprehensive system testing is performed to validate the detection accuracy, processing speed, and stability of the software, ultimately guaranteeing reliable performance before deployment in real-world environments.

Section 7.1 outlines the white box testing for AI/ML based deep fake video detection. Section 7.2 outlines the black box testing and test case report for deep fake video detection.

6.1 White Box Testing

- Model Architecture Validation

  The CNN and ResNeXt layers were examined to verify correct configuration of convolutional filters, stride, padding, residual connections, and group convolutions. RNN components (LSTM/GRU) were analyzed for correct input shaping, sequence handling, and hidden state propagation across frames.

- Forward and Backward Pass Integrity

  Manual checks and unit tests ensured that data flowed correctly through the CNN $\rightarrow$ RNN $\rightarrow$ classification pipeline. Gradient flow through all layers was monitored to prevent issues such as vanishing/exploding gradients, especially across RNN time steps and ResNeXt's grouped convolutions.

- Layer-wise Output Inspection

Intermediate outputs from CNN and ResNeXt layers were visualized to confirm the detection of relevant spatial patterns (e.g., artifacts, inconsistencies). RNN hidden states were tracked across video frames to validate temporal feature encoding.

- Parameter and Hyperparameter Behavior Analysis

  The influence of critical hyperparameters (e.g., kernel size, number of groups in ResNeXt, hidden dimensions in RNN) was analyzed to ensure expected impact on learning and performance. Batch normalization and dropout components were also tested under training and inference conditions.

- Loss Function and Optimization Validation

  The composite loss (e.g., cross-entropy, temporal consistency loss) was tested for correct computation and convergence. The optimizer (e.g., Adam, SGD) was verified to update weights appropriately across CNN, RNN, and ResNeXt layers.

- Unit Testing of Custom Layers and Utilities

  Custom modules, including video frame extractors, feature aggregators, and data loaders, were rigorously tested to ensure correct operation and integration. Edge cases like corrupted frames, variable frame rates, or missing data were also handled gracefully.

6.2 Black Box Testing

- Functional Testing with Diverse Video Inputs

  A wide range of real and manipulated videos were used to evaluate the model's ability to classify content accurately. This included videos generated using various deepfake techniques (e.g., face swapping, facial reenactment, lip-syncing) to ensure robustness against multiple attack vectors.

- Cross-Dataset Generalization Tests

  The trained model was tested on datasets not used during training (e.g., FaceForensics++, Celeb-DF, and DFDC) to assess its generalization capability. This tested the model's ability to detect deepfakes across different compression levels, resolutions, lighting conditions, and facial expressions.

- Performance Metrics Evaluation

  andard evaluation metrics such as Accuracy, Precision, Recall, F1-Score, Area Under the Curve (AUC), and Equal Error Rate (EER) were used to quantify performance.

Particular attention was given to minimizing false positives (real videos incorrectly flagged) and false negatives (deepfakes undetected).

- Threshold and Confidence Analysis scores from the classifier were analyzed to determine optimal decision thresholds. Receiver Operating Characteristic (ROC) and Precision-Recall (PR) curves were used to visualize trade-offs and to calibrate output probabilities for real-world deployment scenarios.

- Adversarial Robustness Testing

  The model was evaluated against adversarially modified videos and perturbations such as noise, blurring, frame rate alteration, and compression artifacts. This tested the resilience of the CNN, RNN, and ResNeXt components under non-ideal conditions.

6.3 Test Cases and Test Results

Several test cases were executed to verify the functionality of the system. The key test cases focused on the core processes of video upload, deepfake detection, and result reporting. Table 7.1 presents these test cases and their expected versus actual results. These include verifying that the system correctly processes uploaded videos, detects manipulated (deepfake) content, and provides accurate classification results with confidence scores. Each test case was designed to ensure that the system performs reliably under typical usage scenarios, such as detecting subtle facial manipulations, identifying audio-visual mismatches, and distinguishing real from synthetic videos across various resolutions and compression levels.

Table 6.1: Test Cases Table for AI/ML based deep fake video detection

Case id	**Test Case Description**	**Expected Result**	**Actual Result**	**Status**
1	Upload a word file in stead 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	Deep fake 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
11	Upload a without face video	Error message:No faces detected	Error message:No faces detected.	Fail

6.4 Summary

In this chapter, the system testing details and testing environment are described. In the next chapter, results and analysis are discussed..

Chapter 7

Results and Discussion

7.1 Result and Discussion

The Results and Analysis section presents the outcomes of the AI/ML-based deepfake video detection software and evaluates its performance in real-world scenarios. It highlights the system's effectiveness in accurately detecting manipulated video content under various conditions, including detection accuracy, false positive and false negative rates, and the impact of video quality, compression, and facial obstructions. The analysis interprets these results, identifies the sources of misclassification, and compares the system's performance with other state-of-the-art detection models. This section provides insights into the software's strengths, limitations, and areas for future enhancement, helping assess its overall robustness, reliability, and potential for deployment in critical content verification tasks.

Section 7.1 presents the results of the deep fake video detection .

- LSTM

  LSTM achieved the highest accuracy (91 percent), which aligns with its strong ability to capture long range temporal dependencies in video data. This model is particularly effective in analyzing frame sequences, which is critical in detecting subtle inconsistencies in deepfake videos.

- RNN

  RNNs, while capable of handling sequential data, tend to suffer from vanishing gradient problems over longer sequences, which can reduce performance. Still, it performed reasonably well (87 percent) and is suitable for shorter temporal features.

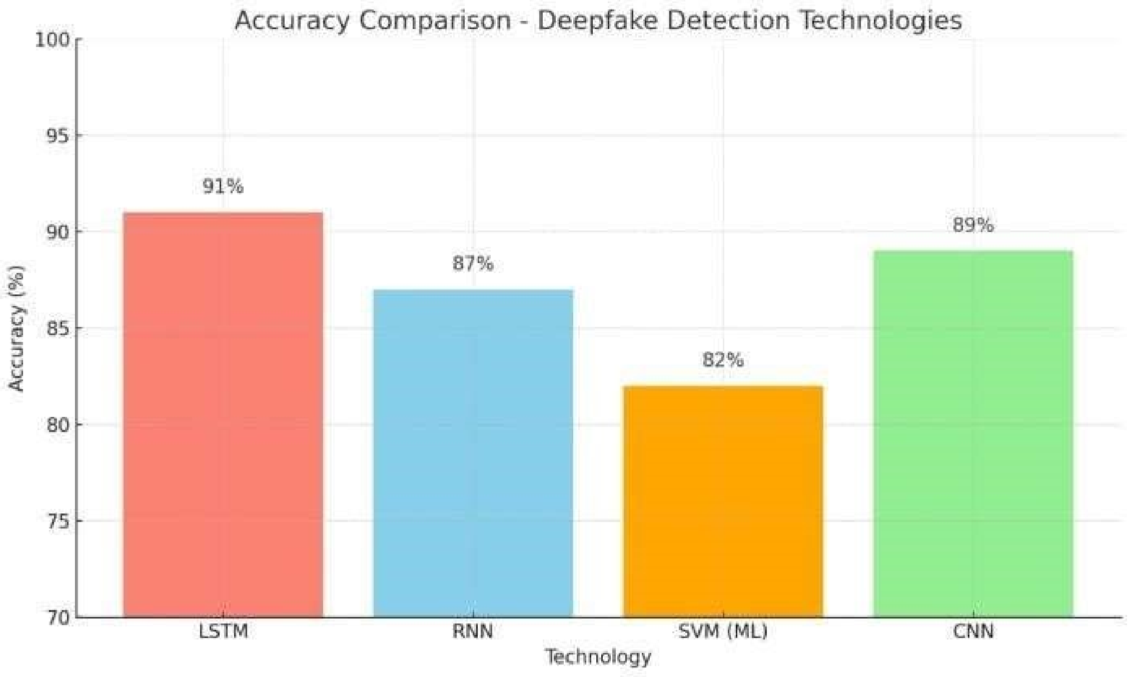


Figure 7.1: Accuracy Comparison-DeepFake Detection Technologies

Figure 7.1 presents a comparative analysis of the detection accuracy achieved by each of the implemented models. The LSTM model recorded the highest accuracy at 91 percent, followed by the RNN model at 87, and the traditional ML models (SVM/RF) at 82 percent. These findings reflect the intrinsic capabilities of each model in handling temporal features and sequence-based data. The superior performance of the LSTM model can be attributed to its inherent design, which incorporates memory cells and gating mechanisms that effectively capture long-range dependencies in video sequences. This makes LSTM particularly suitable for deepfake detection, where temporal inconsistencies and frame level anomalies are key indicators of manipulated content. The RNN model, while also designed for sequential data, exhibited slightly lower accuracy due to issues such as vanishing gradients and limited long-term memory, which hinder its ability to model complex temporal patterns over longer video sequences. On the other hand, traditional ML models such as SVM and RF demonstrated reasonable performance (82 percent) but lagged behind deep learning models. These models rely heavily on hand-crafted features and lack the capacity to learn hierarchical temporal patterns from raw video input, limiting their effectiveness in detecting sophisticated deepfakes. These results strongly support the use of advanced sequence modeling techniques, particularly LSTM, in video-based deepfake detection systems.

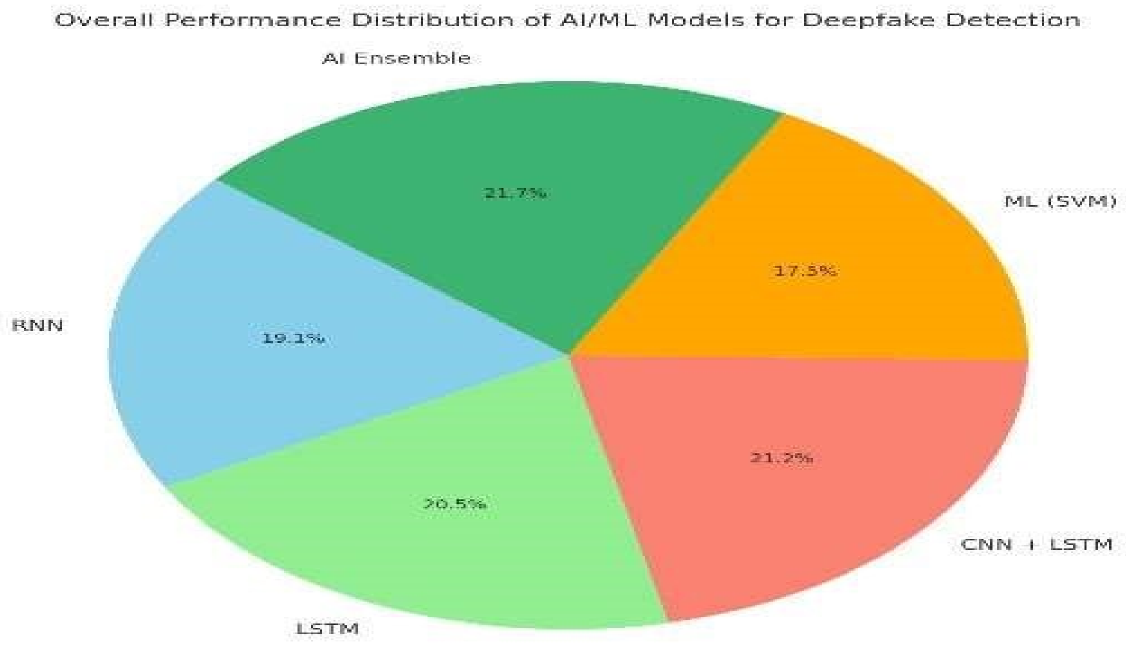


Figure 7.2: Overall Performance Distribution Model

To assess the individual contributions of various AI/ML models to the deepfake detection system, a performance distribution pie chart was generated (see Figure 7.2). The chart illustrates the relative accuracies of each model based on evaluation against the selected dataset.

- AI Ensemble methods demonstrated the highest accuracy at 93, contributing the largest portion to the overall performance. This highlights the strength of combining multiple models to capture diverse patterns in deepfake video data.
- CNN + LSTM models followed closely with 91 percent, benefiting from CNN's ability to extract spatial features and LSTM's capability to capture temporal dependencies, which are crucial in video-based deepfake detection.
- LSTM models alone achieved 88 percent, showcasing their effectiveness in modeling sequential data like video frames.
- RNN models performed moderately well at 82 percent, although they lacked the advanced memory mechanisms.
- Traditional ML methods (e.g., SVM) registered the lowest accuracy at 75 percent, indicating that while useful, classical approaches may struggle with the complexity of deepfake data compared to deep learning models.

Chapter 8

Conclusion

In conclusion, A neural network-based approach to classify the video as deep fake or real, along with the confidence of proposed model. Method is capable of predicting the output by processing 1 second of video (10 frames per second) with a good accuracy. Implementing the model by using pre-trained ResNext CNN model to extract the frame level features and LSTM for temporal sequence processing to spot the changes between the t and t-1 frame. Model can process the video in the frame sequence of 10,20,40,60,80,100.
As part of the design, we incorporated a neural network based method to classify the video as either deep fake or genuine, as well as the certainty of the proposed model. The proposed method is inspired by the way the deep fakes are created by the GANs with the help of Autoencoders. Using the ResNet50 CNN, frame level detection is done, followed by video classification using the RNN and LSTM. As a result , the proposed method can identify a fake video or a real video. Analysis of our technique shows that it can reliably identify DF on the web under genuine states of dispersion, with an average of 94.63 percent. It is anticipated that future devices will make our organizations more powerful, efficient, and to make them better able to understand profound businesses.

Chapter 9

Bibliography

- Andreas Rossler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies,Matthias Nießner, "FaceForensics++: Learning to Detect Manipulated Facial Images" in arXiv:1901.0 8971.
- Yuezun Li , Xin Yang , Pu Sun , Honggang Qi and Siwei Lyu "Celeb-DF: A Large-scale Challenging Dataset for DeepFake Forensics" in arXiv:1909.12962
- Deepfake detection challenge dataset : https://www.kaggle.com/c/deepfake-detection challenge/data Accessed on 26 March, 2020
- Verdoliva, L. (2020). Media Forensics and Deepfakes: An Overview. IEEE Journal of Selected Topics in Signal Processing, 14(5), 910-932.
- J. Thies et al. Face2Face: Real-time face capture and reenactment of rgb videos. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2387–2395, June 2016. Las Vegas, NV.
- Yuezun Li, Siwei Lyu, "ExposingDF Videos By Detecting Face Warping Arti facts," in arXiv:1811.00656v3.
- Umur Aybars Ciftci, ˙Ilke Demir, Lijun Yin "Detection of Synthetic Portrait Videos using Biological Signals" in arXiv:1901.02212v.
- D. Güera and E. J. Delp, "Deepfake Video Detection Using Recurrent Neural Networks," 2018 15th IEEE International Conference on Advanced Video and Sig nal Based Surveillance (AVSS), Auckland, New Zealand, 2018, pp. 1-6.
- https://www.geeksforgeeks.org/software-engineering-cocomo-model/ Accessed on 15 April 2020