DETECTING THE UNDETECTABLE : ADVANCES IN DEEPFAKE IDENTIFICATION

A PROJECT REPORT

Submitted by

DHINESH KUMAR.R	**(510821104004)**
DINESH KUMAR.M	**(510821104005)**
VIVEK.M	**(510821104028)**

In partial fulfilment for the award of the degree

Of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

GANADIPATHYTULSI'S JAIN ENGINEERING COLLEGE, VELLORE

ANNA UNIVERSITY: CHENNAI 600 025

MAY 2025

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report " **DETECTING THE UNDETECTABLE : ADVANCES IN DEEPFAKE IDENTIFICATION"** the bonafide work of

DHINESH KUMAR.R	**(510821104004)**
DINESH KUMAR.M	**(510821104005)**
VIVEK.M	**(510821104028)**

who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Mrs.S.I.SanthanaLakshmi, M.E.,	Mrs.Y.Mohanapriya, M.E.,
HEAD OF THE DEPARTMENT,	SUPERVISOR
Department Of Computer	Department Of Computer
Science And Engineering	Science And Engineering
GTEC	GTEC
Kanniyambadi	Kanniyambadi

Submitted for the university examination held on ___________________

Internal Examiner **External Examiner**

ACKNOWLEDGEMENT

We first offer our deepest gratitude to our **GOD** the almighty who has given us strength and good health during the course of the project.

We are conscious about our indebtedness to our **Managing Trustee Shri.N.Sugalchand Jain**, our Chairman **Shri.P.Pyarelal Jain**, our **Secretary Shri.T.Amarchand Jain**, our **PrincipalDr.M.BARATHI,** who inspired us to reach greater heights in the pursuits of knowledge.

We articulate our sincere gratitude to our **Head of the Department Mrs.S.I.Santhana Lakshmi, M.E.,** who had taught us the way to do a successful project and without whose constant encouragement and whole idea, the project would not have been possible.

We wish to express our sincere thanks to our **Internal Guide Mrs.Y.Mohanapriya, M.E.,** who had helped us a lot and had given valuable suggestion, which made us to finish this project in very successful and neat way.

We wish to express our sincere thanks and honour to our **Project Coordinator Mrs.S.I.Santhana Lakshmi, M.E.,**

We extend our thanks to forto **Mrs.S.Vennila, M.E., Mrs.G.V.Varshini, M.E, Ms.S.Jothika, M.E.,** and **Mrs.G.Pushpa Antanet Sheeba, M.E.,** their motivation**.** We also express our sincere thanks to the entire team of teaching and non-teaching staffs of CSEDepartment.

We take this opportunity to thank our parents and friends who have been the source of inspiration and an encouraging factor throughout the project period.

ABSTRACT

Advancements in computational power have significantly strengthened deep learning algorithms, making it easier to generate highly realistic, human-like synthetic videos—commonly known as *deepfakes*. These face-swapped videos can be misused to incite political unrest, fabricate terrorist events, create revenge porn, or blackmail individuals, raising serious ethical and security concerns. In this work, we present a deep learning-based approach capable of automatically distinguishing AI-generated fake videos from real ones. Our method is designed to detect both *replacement* and *reenactment* types of deepfakes.The proposed system uses a ResNeXt Convolutional Neural Network to extract frame-level features from video input. These features are then processed by a Long Short-Term Memory (LSTM) based Recurrent Neural Network (RNN) to classify whether a video has been manipulated. To better simulate real-world conditions, we evaluate our model on a large, balanced dataset created by combining several benchmark datasets: FaceForensics++ [1], the Deepfake Detection Challenge [2], and Celeb-DF [3]. Our results show that the system can achieve competitive performance using a simple yet effective and robust architecture.

Keywords: ResNeXt Convolutional Neural Network, Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), Deepfake Detection, Computer Vision.

TABLE OF CONTENTS

LIST OF TABLES

Chapter 1

Synopsis

Deepfake is a technique for human image synthesis based on neural network tools like GAN(Generative Adversarial Network) or Auto Encoders etc. These tools su per impose target images onto source videos using a deep learning techniques and create a realistic looking deep fake video. These deep-fake video are so real that it becomes impossible to spot difference by the naked eyes. In this work, we describe a newdeeplearning-based method that can effectively distinguish AI-generated fake videos from real videos. We are using the limitation of the deep fake creation tools as a powerful way to distinguish between the pristine and deep fake videos. Dur ing the creation of the deep fake the current deep fake creation tools leaves some distinguishable artifacts in the frames which may not be visible to the human being but the trained neural networks can spot the changes. Deepfake creation tools leave distinctive artefacts in the resulting Deep Fake videos, and we show that they can be effectively captured by Res-Next Convolution Neural Networks. Our system uses a Res-Next Convolution Neural Networks to extract frame-level features. These features are then used to train a Long Short Term Memory(LSTM) based Recurrent Neural Network(RNN) 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. We proposed to evaluate our method against a large set of deep fake videos collected from multiple video websites. We are tried to make the deep fake detection model perform better on real time data. To achieve this we trained our model on combi nation of available data-sets.So that our model can learn the features from different kind of images. Weextracteda adequate amountofvideosfromFace-Forensic++[1], Deepfake detection challenge[2], a

nd Celeb-DF[3] data-sets.

Chapter 2

Technical Keywords

2.1 Area of Project

Our project is a Deep learning project which is a sub branch of Artificial Intelligence and deals with the human brain inspired neural network technology. Computer vi sion plays an important role in our project. It helps in processing the video and frames with the help of Open-CV. A PyTorch trained model is a classifier to classify the source video as deepfake or pristine.

2.2 Technical Keywords

- Deep learning
- Computer vision
- Res-Next
- Convolution Neural Network
- Long short-term memory (LSTM)
- OpenCV
- Face Recognition
- GAN(Generative Adversarial Network)
- PyTorch.

Chapter 3

Introduction

3.1 Project Idea

In the world of ever growing Social media platforms, Deepfakes are con sidered 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. It becomes very important to spot the difference between the deepfake and pris tine video.

We are using AI to fight AI.Deepfakes are created using tools like FaceApp[11] and Face Swap[12], which using pre-trained neural networks like GANorAutoencoders for these deepfakes creation. Our 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, we trained our method with large amount of balanced and combi nation of various available dataset like FaceForensic++[1], Deepfake detection challenge[2], and Celeb-DF[3].

Further to make the ready to use for the customers, we have developed a front end application where the user the user will upload the video. The video will be processed by the model and the output will be rendered back to the user with the classification of the video as deepfake or real and confidence of the model.

3.2 Motivation of the Project

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 anytrans-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 grow ing 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.

Spreading of the Deep fakes over the social media platforms have become very common leading to spamming and peculating wrong information over the plat form. Just imagine a deep fake of our prime minister declaring war against neighboring countries, or a Deep fake of reputed celebrity abusing the fans. These types of the deep fakes will be terrible, and lead to threatening, mislead ing of common people. To overcome such a situation, Deep fake detection is very important. So, we 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.

3.3 Literature Survey

Face Warping Artifacts [15] used the approach to detect artifacts by com paring the generated face areas and their surrounding regions with a dedicated Convolutional Neural Network model. In this work there were two-fold of Face Artifacts. Their method is based on the observations that current deepfake algorithm can only generate images of limited resolutions, which are then needed to be fur ther transformed to match the faces to be replaced in the source video. Their method has not considered the temporal analysis of the frames. Detection by Eye Blinking [16] describes a new method for detecting the deep fakes by the eye blinking as a crucial parameter leading to classification of the videos as deepfake or pristine. The Long-term Recurrent Convolution Network (LRCN) was used for temporal analysis of the cropped frames of eye blinking. Astodaythedeepfake generation algorithms have become so powerful that lack of eye blinking can not be the only clue for detection of the deepfakes. There must be certain other parameters must be considered for the detection of deep fakes like teeth enchantment, wrinkles on faces, wrong placement of eyebrows etc. Capsule networks to detect forged images and videos [17] uses a method that uses a capsule network to detect forged, manipulated images and videos in different scenarios, like replay attack detection and computer-generated video detection. In their method, they have used random noise in the training phase which is not a good option. Still the model performed beneficial in their dataset but may fail on real time data due to noise in training. Our method is proposed to be trained on noiseless and real time datasets. Recurrent Neural Network [18] (RNN) for deepfake detection used the ap proach of using RNN for sequential processing of the frames along with Ima geNet pre-trained model. Their process used the HOHO [19] dataset consisting of just 600 videos. Their dataset consists small number of videos and same type of videos, which may not perform very well on the real time data. We will be training out model on large number of Realtime data.

Synthetic Portrait Videos using Biological Signals [20] approach extract bio logical signals from facial regions on pristine and deepfake portrait video pairs. Applied transformations to compute the spatial coherence and temporal consistency, capture the signal characteristics in feature vector and photoplethysmography (PPG) maps, and further train a probabilistic Support Vector Machine (SVM) and a Convolutional Neural Network (CNN). Then, the average of au thenticity probabilities is used to classify whether the video is a deepfake or a pristine. Fake Catcher detects fake content with high accuracy, independent of the generator, content, resolution, and quality of the video. Due to lack of discriminator leading to the loss in their findings to preserve biological signals, formulating a differentiable loss function that follows the proposed signal processing steps is not straight forward process.

Chapter 4

Problem Definition and scope

4.1 Problem Statement

Convincing manipulations of digital images and videos have been demonstrated for several decades through the use of visual effects, recent advances in deep learn ing 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[14], fake terrorism events, revenge porn, blackmail peoples etc.So it becomes very important to detect these deepfake and avoid the percolation of deepfake through social media platforms. We have taken a step forward in detecting the deep fakes using LSTM based artificial Neural network.

4.1.1 Goals and objectives

Goal and Objectives:

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

4.1.2 Statement of scope

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

4.2 Major Constraints

- User: User of the application will be able detect the whether the uploaded video is fake or real, Along with the model confidence of the prediction.
- Prediction: The User will be able to see the playing video with the output on the face along with the confidence of the model.
- Easy and User-friendly User-Interface: Users seem to prefer a more sim plified process of Deep Fake video detection. Hence, a straight forward and user-friendly interface is implemented.The UI contains a browse tab to select the video for processing. It reduces the complications and at the same time enrich the user experience.
- Cross-platform compatibility: with an ever-increasing target market, accessibility should be your main priority. By enabling a cross-platform compatibility feature, you can increase your reach to across different platforms. Being a server side application it will run on any device that has a web browser installed in it.

4.3 Methodologies of Problem solving

4.3.1 Analysis

- **Solution Requirement**

We analysed the problem statement and found the feasibility of the solution of the problem. We read different research paper as mentioned in 3.3. After checking the feasibility of the problem statement. The next step is the data set gathering and analysis. We analysed the data set in different approach of training like negatively or positively trained i.e training the model with only fake or real video’s but found that it may lead to addition of extra bias in the model leading to inaccurate predictions. So after doing lot of research we found that the balanced training of the algorithm is the best way to avoid the bias and variance in the algorithm and get a good accuracy.

- **Solution Constraints**
  We analysed the solution in terms of cost,speed of processing,requirements,level of expertise, availability of equipment’s.
- **Parameter Identified**

1. Blinking of eyes

2. Teeth enchantment

3. Bigger distance for eyes

4. Moustaches

5. Double edges, eyes, ears, nose

6. Iris segmentation

7. Wrinkles on face

8. Inconsistent head pose

9. Face angle

10. Skin tone

11. Facial Expressions

12. Lighting

13. Different Pose

14. Double chins

15. Hairstyle

16. Higher cheek bones

4.3.2 Design

After research and analysis we developed the system architecture of the solution as mentioned in the Chapter 6. We decided the baseline architecture of the Model which includes the different layers and their numbers.

4.3.3 Development

After analysis we decided to use the PyTorch framework along with python3 language for programming. PyTorch is chosen as it has good support to CUDA i.e Graphic Processing Unit (GPU) and it is customize-able. Google Cloud Platform for training the final model on large number of data-set.

4.3.4 Evaluation

We evaluated our model with a large number of real time dataset which include YouTubevideos dataset. Confusion Matrix approach is used to evaluate the accuracy of the trained model.

4.4 Outcome

The outcome of the solution is trained deepfake detection models that will help the users to check if the new video is deepfake or real.

4.5 Applications

Web based application will be used by the user to upload the video and submit the video for processing. The model will pre-process the video and predict whether the uploaded video is a deepfake or real video.

4.6 Hardware Resources Required

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

Client-side Requirements: Browser: Any Compatible browser device

Sr. No.	Parameter	Minimum Requirement
1	SPEED	3.5 GHz
2	RAM	16GB
3	Hard Disk	100GB
4	PROCESSOR	I5

4.7 Software Resources Required

Platform :

1. Operating System: Windows11
2. Programming Language : Python 3.0
3. Framework: PyTorch 1.4 , Django 3.0
4. Libraries : OpenCV, Face-recognition

Chapter 5

Project Plan

5.1 Project Model Analysis

WeUseSpiral model As the Software development model focuses on the people do ing the work,how they work together and risk handling. We are using Spiral because, It ensures changes can be made quicker and throughout the development process by having consistent evaluations to assess the product with the expected outcomes re quested. As we developed the application in the various modules, spiral model is best suited for this type of application. An Spiral approach provides a unique op portunity for clients to be involved throughout the project, from prioritizing features to iteration planning and review sessions to frequent algorithms containing new features. However, this also requires clients to understand that they are seeing a work in progress in exchange for this added benefit of transparency. As our model consists of lot of risk and spiral model is capable of handling the risks that’s the reason we are using spiral model for product development.

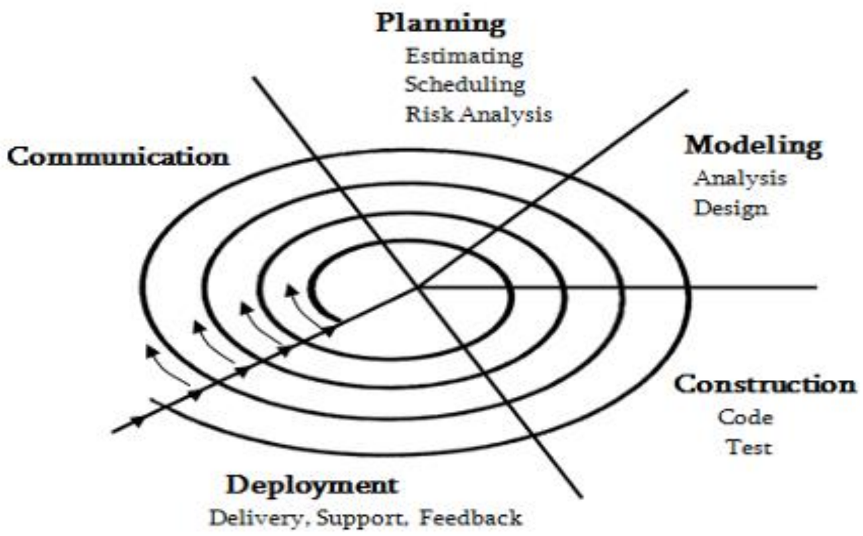


Figure 5.1: Spiral Methodology SDLC

5.1.1 Reconciled Estimates

1. Cost Estimate : Rs 11,600

Table 5.1: Cost Estimation

Cost(in Rs)	Description
5260	Pre-processing the dataset on GCP
2578	Training models on on GCP
761	Google Colab Pro subscription
3000	Deploying project to GCP using Cloud engine

2. Time Estimates :4 months

5.1.2 Cost Estimation using COCOMO(Constructive Cost) Model

Since we have small team , less-rigid requirements, long deadline we are using the organic COCOMO[23] model.

1. Efforts Applied: It defines the Amount of labor that will be required to com plete a task. It is measured in person-months units.

$$EffortApplied(E) = a_b(KLOC)^{b_b}$$

$$E = 2.4(20.5)^{1.05}$$

$$E = 57.2206PM$$

2. Development Time: Simply means the amount of time required for the com pletion of the job, which is, of course, proportional to the effort put. It is measured in the units of time such as weeks, months.

$$DevelopmentTime(D) = c_b(E)^{d_b}$$

$$D = 2.5(57.2206)^{0.38}$$

D=116M

3. People Required: The number of developed needed to complete the project.

People Required(P) = E /D

P=57.2206 /11/6

P=4.93

5.2 Risk Management w.r.tNP Hard analysis

5.2.1 Risk Identification

Before the training, we need to prepare thousands of images for both persons. We can take a shortcut and use a face detection library to scrape facial pictures from their videos. Spend significant time to improve the quality of your facial pictures. It impacts your final result significantly.

1. Remove any picture frames that contain more than one person.

2. Make sure you have an abundance of video footage. Extract facial pictures contain different pose, face angle, and facial expressions.

3. Some resembling of both persons may help, like similar face shape.

5.2.2 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

1. Apply a Gaussian filter to further diffuse the mask boundary area.

2. Configure the application to expand or contract the mask further.

3. Control the shape of the mask.

Table 5.2: 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 5.3: Risk Probability definitions

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

5.3 Project Schedule

5.3.1 Project task set

Major Tasks in the Project stages are

- Task 1: Data-set gathering and analysis This task consists of downloading the dataset. Analysing the dataset and making the dataset ready for the preprocessing.
- Task 2 : Module 1 implementation Module 1implementation consists of splitting the video to frames and cropping each frame consisting of face.
- Task 3: Pre-processing Pre-processing includes the creation of the new dataset which includes only face cropped videos.
- Task 4: Module 2 implementation Module 2 implementation consists of implementation of DataLoader for load ing the video and labels. Training a base line model on small amount of data.
-

- Task 5 : Hyper parameter tuning This task includes the changing of the Learning rate, batch size, weight decay and model architecture until the maximum accuracy is achieved.
- Task 6 : Training the final model The final model on large dataset is trained based on the best hyper parameter identified in the Task 5.
- Task 7 : Front end Development This task includes the front end development and integration of the back-end and front-end.
- Task 8 : Testing The complete application is tested using unit testing,.

Chapter 6

Software requirement specification

6.1 Introduction

6.1.1 Purpose and Scope of Document

This document lays out a project plan for the development of Deepfake video de tection using neural network.The intended readers of this document are current and future developers working on Deepfake video detection using neural network and the sponsors of the project. The plan will include, but is not restricted to, a sum mary of the system functionality, the scope of the project from the perspective of the “Deepfake video detection” team (me and my mentors), use case diagram, Data f low diagram,activity diagram, functional and non- functional requirements, project risks and how those risks will be mitigated, the process by which we will develop the project, and metrics and measurements that will be recorded throughout the project.

6.1.2 Use Case View

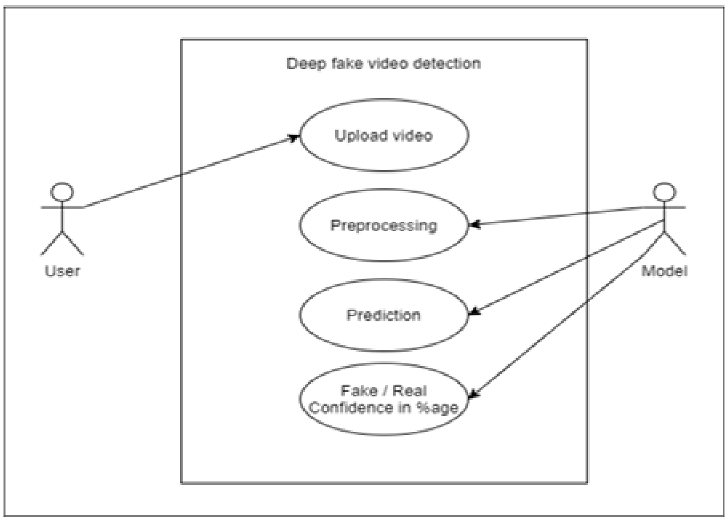


Figure 6.1: Use case diagram

6.2 Functional Model and Description

Adescription of each major software function, along with data flow (structured anal ysis) or class hierarchy (Analysis Class diagram with class description for object oriented system) is presented.

6.2.1 Data Flow Diagram

DFDLevel-0

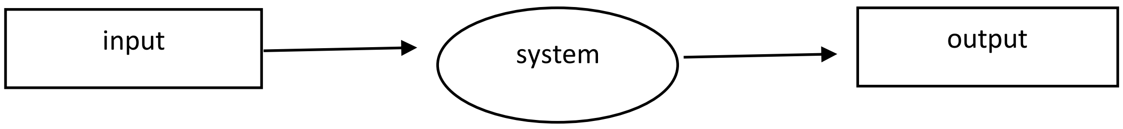


Figure 6.2: DFD Level 0

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

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

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

DFDLevel-1

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

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

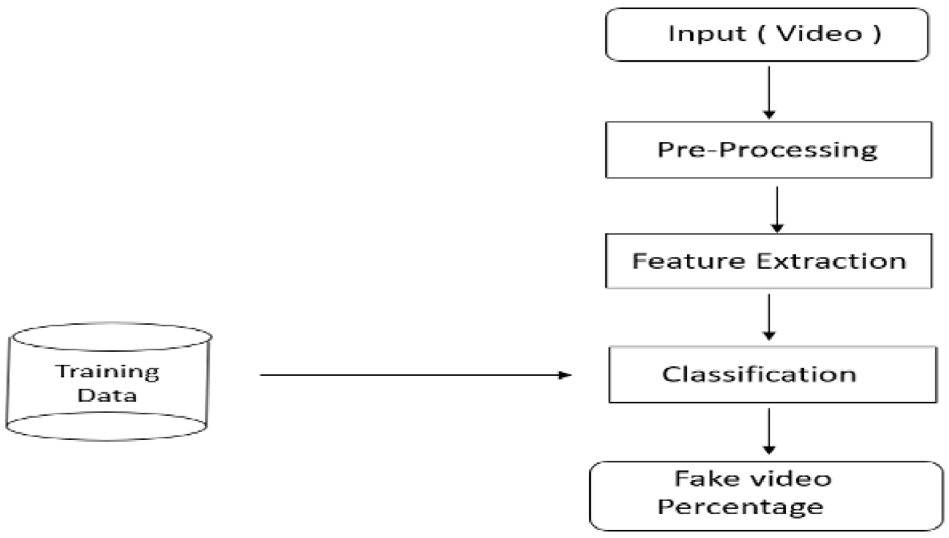


Figure 6.3: DFD Level 1

DFDLevel-2

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

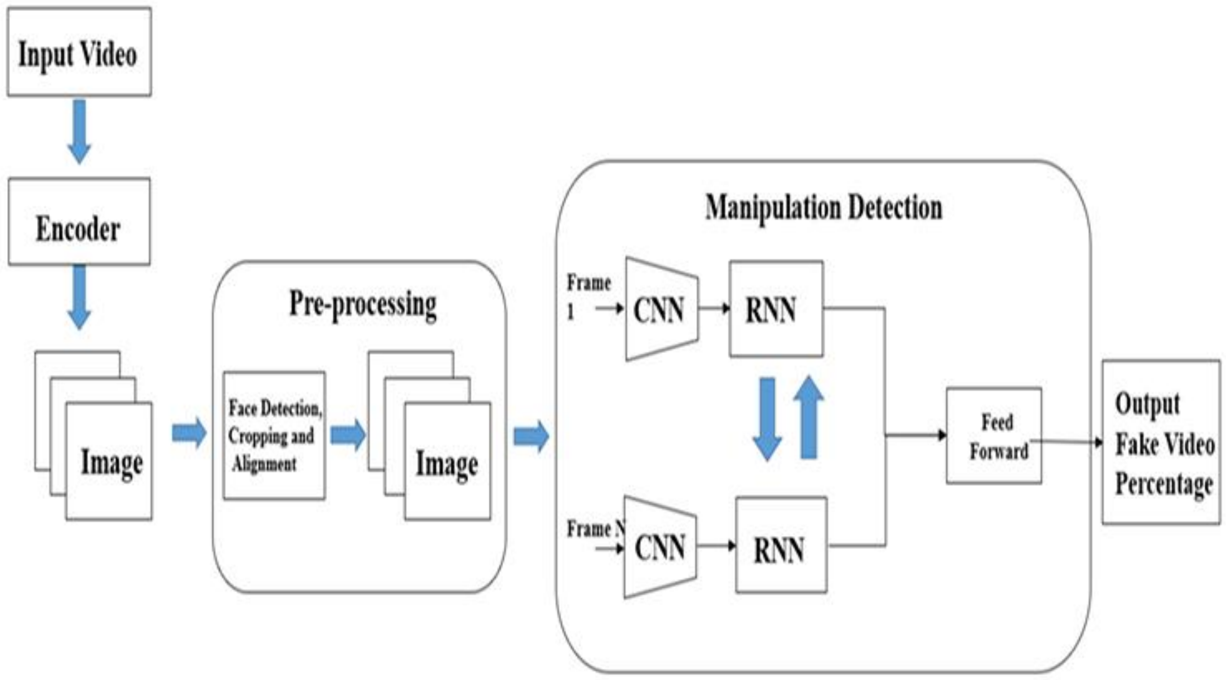


Figure 6.4: DFD Level 2

6.2.2 Activity Diagram:

Training Workflow:

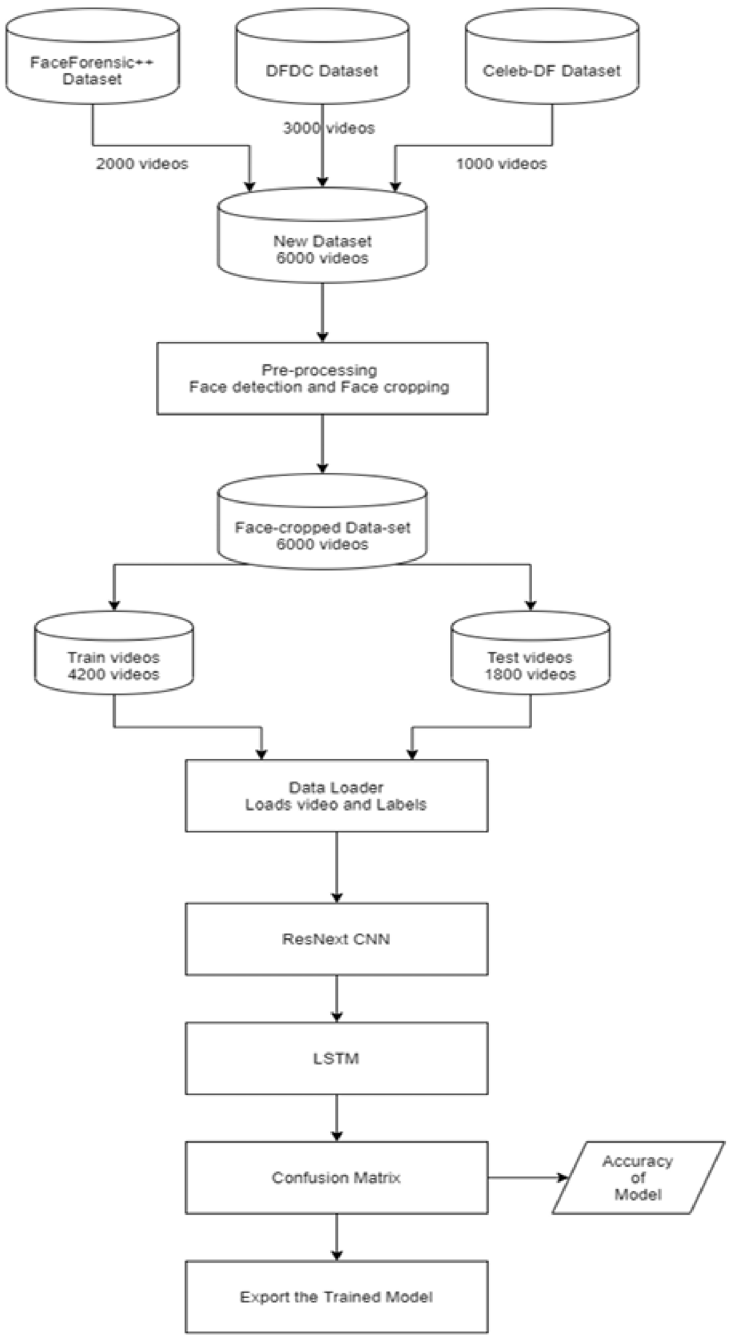


Figure 6.5: Training Workflow

Testing Workflow:

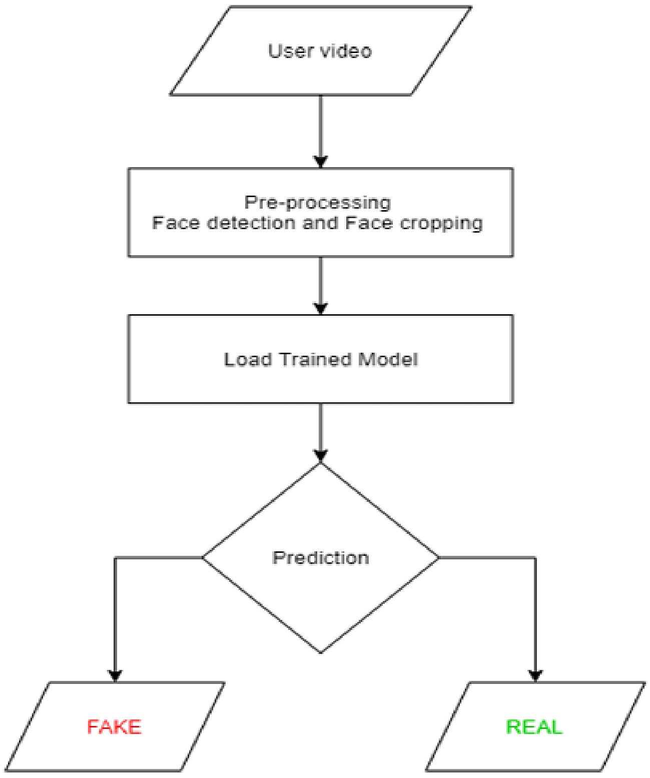


Figure 6.6: Testing Workflow

6.2.3 Non Functional Requirements:

Performance Requirements:

- The software should be efficiently designed so as to give reliable recognition of fake videos and so that it can be used for more pragmatic purpose
- The design is versatile and user friendly
- The application is fast, reliable and time saving
- The system have universal adaptations
-
- The system is compatible with future upgradation and easy integration.

Safety Requirement

- The Data integrity is preserved. Once the video is uploaded to the system. It is only processed by the algorithm. The videos are kept secured from the human interventions, as the uploaded video is not are not able for human manipulation.
- To extent the safety of the videos uploaded by the user will be deleted after 30 min from the server.

Security Requirement

- While uploading the video, the video will be encrypted using a certain symmetric encryption algorithm. On server also the video is in encrypted format only. The video is only decrypted from preprocessing till we get the output. After getting the output the video is again encrypted
- This cryptography will help in maintain the security and integrity of the video.
- SSL certification is made mandatory for Data security.

6.2.4 Sequence Diagram.

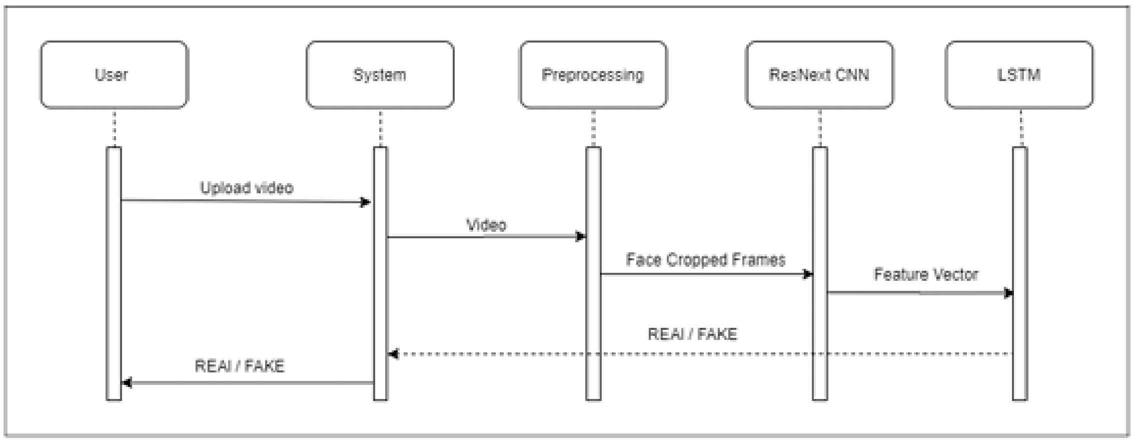


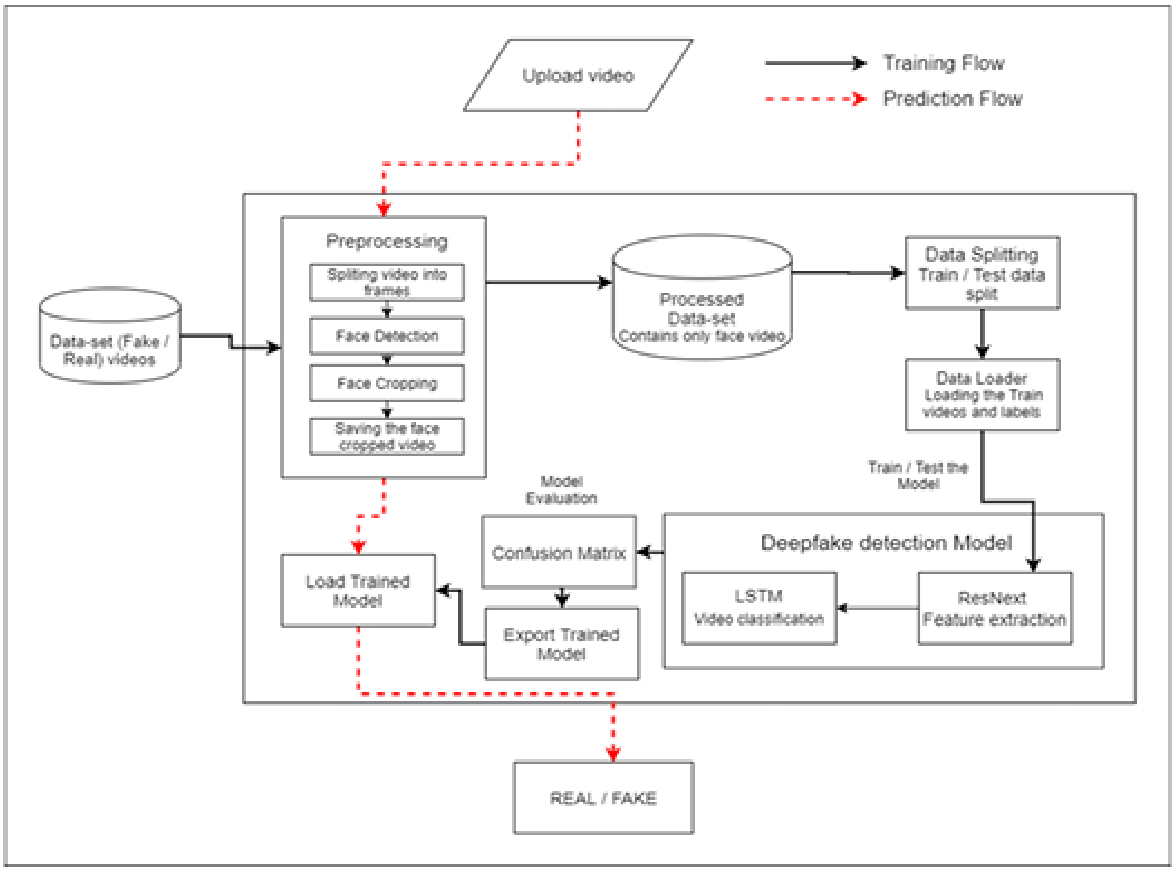
Figure 6.7: Sequence Diagram

Chapter 7

Detailed Design Document

7.1 Introduction

7.1.1 System Architecture



Upload video
Training Flow
Prediction Flow
Preprocessing
Spliting video into frames
Face Detection
Face Cropping
Saving the face cropped video
Data-set (Fake / Real) videos
Processed Data-set Contains only face video
Data Splitting Train / Test data split
Data Loader Loading the Train videos and labels
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 7.1: System Architecture

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

- **Creating deepfake videos**

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

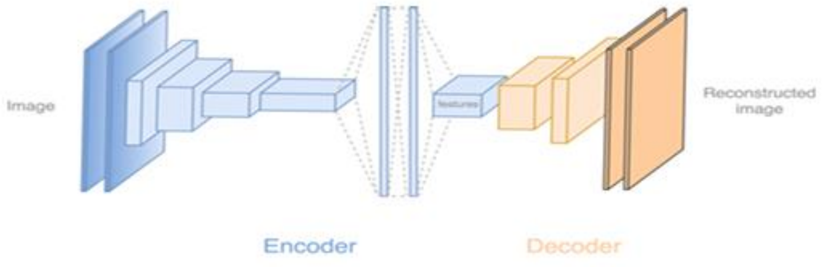


Figure 7.2: Deepfake generation

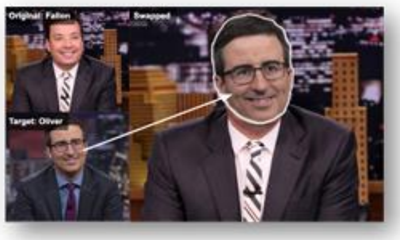


Figure 7.3: Face Swapped deepfake generation

Tools for deep fake creation.

1. Faceswap

2. Faceit

3. Deep Face Lab

4. Deepfake Capsule GAN

5. Large resolution face masked

7.2 Architectural Design

7.2.1 Module 1 : Data-set

For making the model efficient for real time prediction. We have gathered the data from different available data-sets like FaceForensic++(FF)[1], Deepfake detection challenge(DFDC)[2], and Celeb-DF[3]. Futher we have mixed the dataset the col lected datasets and created our own new dataset, to accurate and real time detection on different kind of videos.

To avoid the training bias of the model we have considered 50% Real and 50% fake videos. Deep fake detection challenge (DFDC) dataset [3] consist of certain audio alerted video, as audio deepfake are out of scope for this paper. We preprocessed the DFDCdataset and removed the audio altered videos from the dataset by running a python script. After preprocessing of the DFDC dataset, we have taken 1500 Real and 1500 Fake videos from the DFDC dataset. 1000 Real and 1000 Fake videos from the FaceForensic++(FF)[1] dataset and 500 Real and 500 Fake videos from the Celeb DF[3]dataset. Which makesourtotal dataset consisting 3000 Real, 3000 fake videos and 6000 videos in total. Figure 2 depicts the distribution of the data-sets.

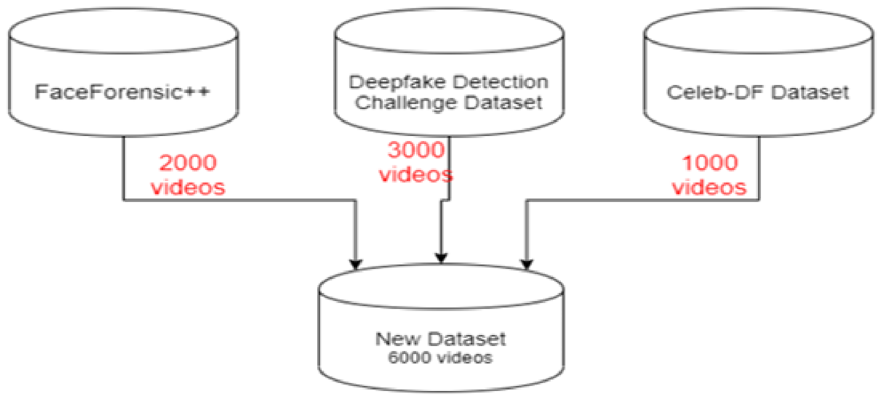


Figure 7.4: Dataset

7.2.2 Module 2 : Pre-processing

In this step, the videos are preprocessed and all the unrequired and noise is removed from videos. Only the required portion of the video i.e face is detected and cropped. The first steps in the preprocessing of the video is to split the video into frames. After splitting the video into frames the face is detected in each of the frame and the frame is cropped along the face. Later the cropped frame is again converted to a new video by combining each frame of the video. The process is followed for each video which leads to creation of processed dataset containing face only videos. The frame that does not contain the face is ignored while preprocessing. To maintain the uniformity of number of frames, we have selected a threshold value based on the mean of total frames count of each video.

Another reason for selecting a threshold value is limited computation power. As a video of 10 second at 30 frames per second(fps) will have total 300 frames and it is computationally very difficult to process the 300 frames at a single time in the experimental environment. So, based on our Graphic Processing Unit (GPU) computational power in experimental environment we have selected 150 frames as the threshold value. While saving the frames to the new dataset we have only saved the first 150 frames of the video to the new video.

To demonstrate the proper use of Long Short-Term Memory (LSTM) we have considered the frames in the sequential manner i.e. first 150 frames and not randomly. The newly created video is saved at frame rate of 30 fps and resolution of 112 x 112.

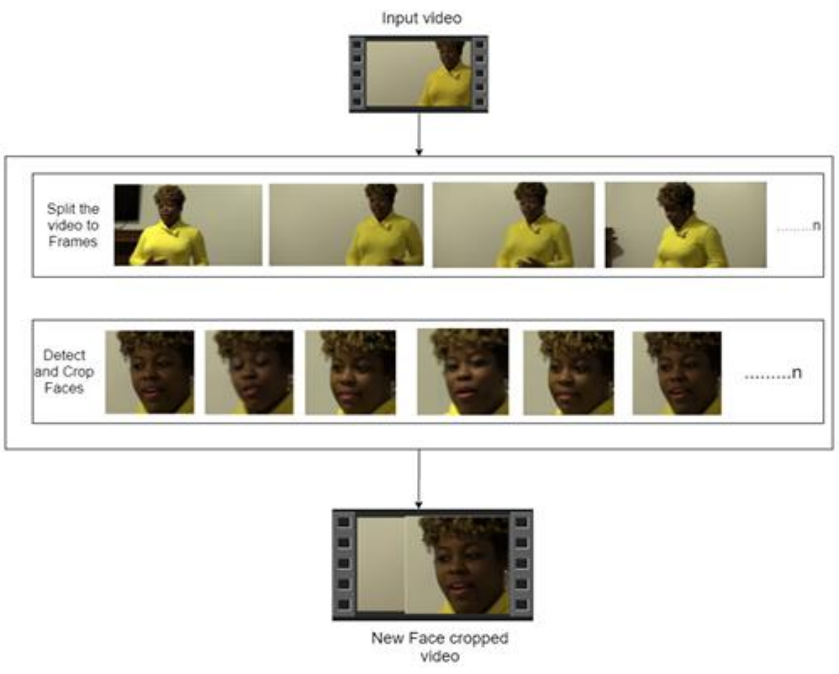


Figure 7.5: Pre-processing of video

7.2.3 Module 3: Data-set split

The dataset is split into train and test dataset with a ratio of 70% train videos (4,200) and 30% (1,800) test videos. The train and test split is a balanced split i.e 50% of the real and 50% of fake videos in each split.

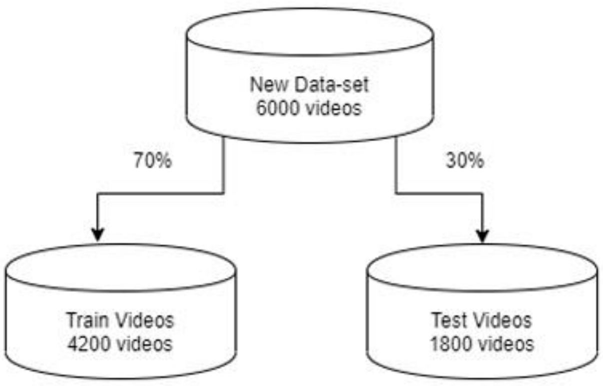


Figure 7.6: Train test split

7.2.4 Module 4: Model Architecture

Our model is a combination of CNN and RNN. We have used the Pre-trained ResNextCNNmodeltoextract the features at frame level and based on the extracted features a LSTM network is trained to classify the video as deepfake or pristine. Us ing the Data Loader on training split of videos the labels of the videos are loaded and fitted into the model for training.

ResNext :

Instead of writing the code from scratch, we used the pre-trained model of ResNext for feature extraction. ResNext is Residual CNN network optimized for high per formance on deeper neural networks. For the experimental purpose we have used resnext50_32x4d model. We have used a ResNext of 50 layers and 32 x 4 dimensions. Following, we will be fine-tuning the network by adding extra required layers and selecting a proper learning rate to properly converge the gradient descent of the model. The 2048-dimensional feature vectors after the last pooling layers of ResNext is used as the sequential LSTM input.

LSTM for Sequence Processing:

2048-dimensional feature vectors is fitted as the input to the LSTM. We are using 1 LSTM layer with 2048 latent dimensions and 2048 hidden layers along with 0.4 chance of dropout, which is capable to do achieve our objective. LSTM is used toprocess the frames in a sequential manner so that the temporal analysis of the video can be made, by comparing the frame at ‘t’ second with the frame of ‘t-n’ seconds. Where n can be any number of frames before t. The model also consists of Leaky Relu activation function.

A linear layer of 2048 input features and 2 output features are used to make the model capable of learning the average rate of correlation between eh input and output. An adaptive average polling layer with the output parameter 1 is used in the model. Which gives the the target output size of the image of the form H x W. For sequential processing of the frames a Sequential Layer is used. The batch size of 4 is used to perform the batch training. A SoftMax layer is used to get the confidence of the model during predication.

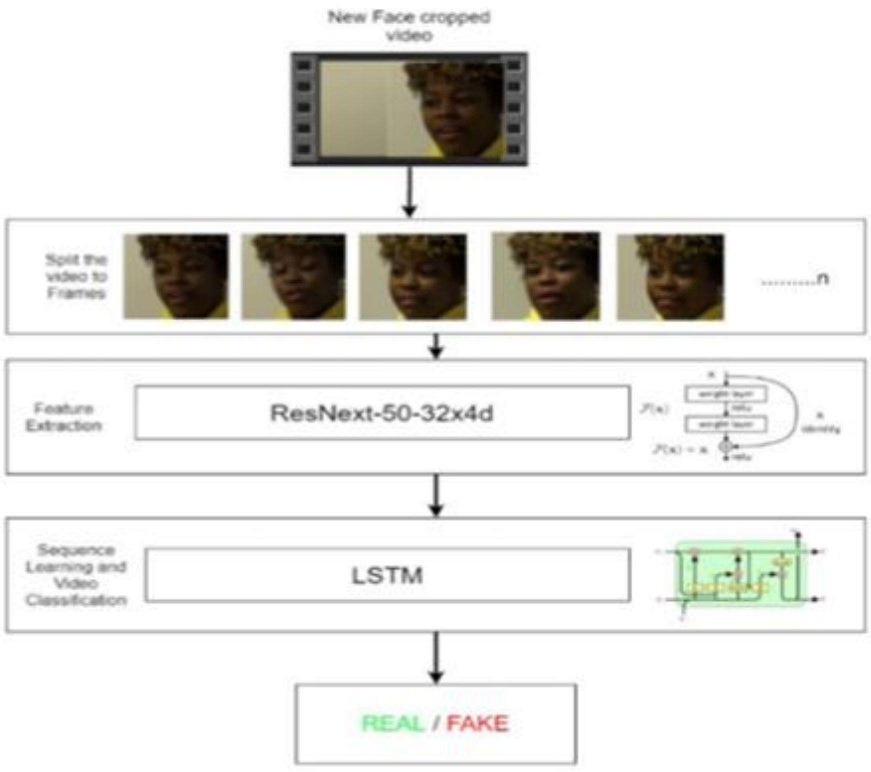


Figure 7.7: Overview of our model

7.2.5 Module 5: Hyper-parameter tuning

It is the process of choosing the perfect hyper-parameters for achieving the maxi mumaccuracy. Afterreiterating many times onthemodel. Thebesthyper-parameters for our dataset are chosen. To enable the adaptive learning rate Adam[21] optimizer with the model parameters is used. The learning rate is tuned to 1e-5 (0.00001) toachieve a better global minimum of gradient descent. The weight decay used is 1e-3. Asthis is a classification problem so to calculate the loss cross entropy approach is used.To use the available computation power properly the batch training is used.

The batch size is taken of 4. Batch size of 4 is tested to be ideal size for training in our development environment. The User Interface for the application is developed using Django framework. Django is used to enable the scalability of the application in the future. The first page of the User interface i.e index.html contains a tab to browse and upload the video. The uploaded video is then passed to the model and prediction is made by the model. The model returns the output whether the video is real or fake along with the confidence of the model. The output is rendered in the predict.html on the face of the playing video.

Chapter 8

Project Implementation

8.1 Introduction

There are many examples where deepfake creation technology is used to mis lead the people on social media platform by sharing the false deepfake videos of the famous personalities like Mark Zuckerberg Eve of House A.I. Hearing, Don ald Trump's Breaking Bad series where he was introduces as James McGill, Barack Obama'spublic service announcement and many more [5]. These types of deepfakes creates a huge panic among the normal people, which arises the need to spot these deepfakes accurately so that they can be distinguished from the real videos.

Latest advances in the technology have changed the field of video manipulation. The advances in the modern open source deep learning frameworks like TensorFlow, Keras, PyTorch along with cheap access to the high computation power has driven the paradigm shift. The Conventional autoencoders[10] and Generative Adversarial Network (GAN) pretrained models have made the tampering of the realistic videos and images very easy. Moreover, access to these pretrained models through the smartphones and desktop applications like FaceApp and Face Swap has made the deepfake creation a childish thing. These applications generate a highly realistic synthesized transformation of faces in real videos. These apps also provide the user with more functionalities like changing the face hair style, gender, age and other attributes. These apps also allow the user to create a very high quality and indistinguishable deepfakes. Although some malignant deepfake videos exist, but till now they remain a minority. So far, the released tools [11,12] that generate deepfake videos are being extensively used to create fake celebrity pornographic videos or revenge porn [13]. Some of the examples are Brad Pitt, Angelina Jolie nude videos. The real looking nature of the deepfake videos makes the celebraties and other fa mouspersonalities the target of pornographic material, fake surveillance videos, fakenews and malicious hoaxes.

The Deepfakes are very much popular in creating the political tension [14]. Due to which it becomes very important to detect the deepfake videos and avoid the percolation of the deepfakes on the social media platforms.

8.2 Tools and Technologies Used

8.2.1 Planning

1. OpenProject

8.2.2 UMLTools

1. draw.io

8.2.3 Programming Languages

1. Python3
2. JavaScript

8.2.4 Programming Frameworks

1. PyTorch
2. Django

8.2.5 IDE

1. Google Colab
2. Jupyter Notebook
3. Visual Studio Code

8.2.6 Versioning Control

1. Git

8.2.7 Cloud Services

1. Google Cloud Platform

8.2.8 Application and web servers:

1. Google Cloud Engine

8.2.9 Libraries

1. torch
2. torchvision
3. os
4. numpy
5. cv2

6. matplotlib

7. face_recognition

8. json

9. pandas

10. copy

11. glob

12. random

13. sklearn

8.3 Algorithm Details

8.3.1 Dataset Details

Refer 7.2.1

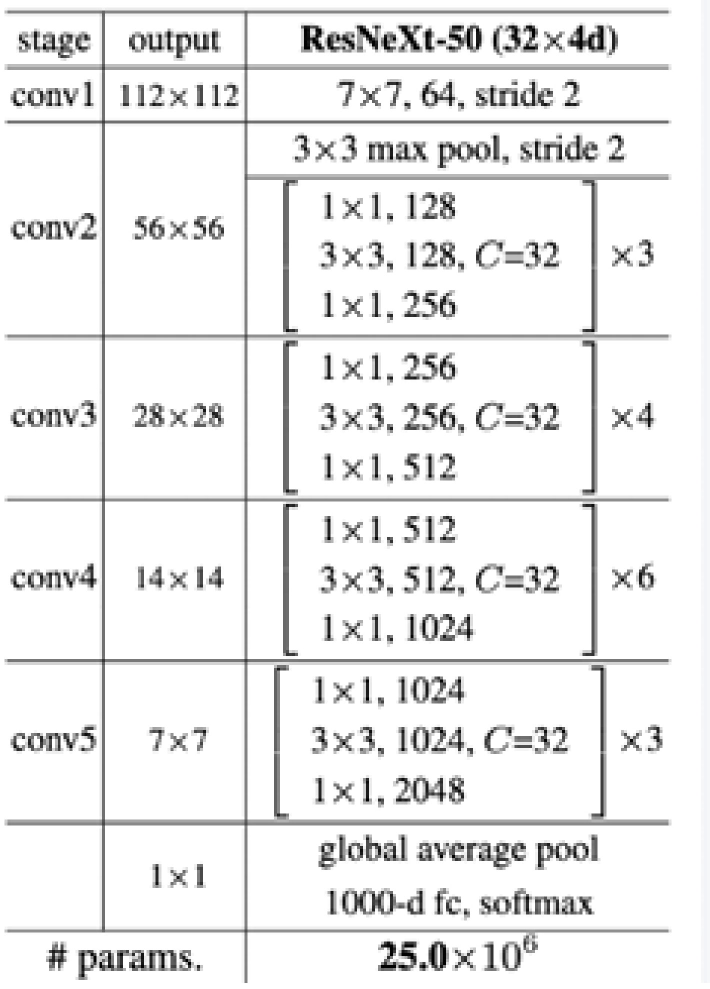
8.3.2 Preprocessing Details

- Using glob we imported all the videos in the directory in a python list. cv2.
- VideoCapture is used to read the videos and get the mean number of frames in each video.
- To maintain uniformity, based on mean a value 150 is selected as idea value for creating the new dataset.
- The video is split into frames and the frames are cropped on face location. The face cropped frames are again written to new video using VideoWriter.
- The newvideo is written at 30 frames per second and with the resolution of 112 x 112 pixels in the mp4 format.
- Instead of selecting the random videos, to make the proper use of LSTM for temporal sequence analysis the first 150 frames are written to the new video.

8.3.3 Model Details

The model consists of following layers:

- **ResNext CNN** : The pre-trained model of Residual Convolution Neural Net work is used. The model name is resnext50_32x4d()[22]. This model consists of 50 layers and 32 x 4 dimensions. Figure shows the detailed implementation of model.



stage	output	ResNeXt-50 (32×4d)	
conv1	112×112	7×7, 64, stride 2	
conv2	56×56	3×3 max pool, stride 2	
		1×1, 128 3×3, 128, C=32 1×1, 256	×3
conv3	28×28	1×1, 256 3×3, 256, C=32 1×1, 512	×4
conv4	14×14	1×1, 512 3×3, 512, C=32 1×1, 1024	×6
conv5	7×7	1×1, 1024 3×3, 1024, C=32 1×1, 2048	×3
	1×1	global average pool 1000-d fc, softmax	
# params.		**25.0**×10^6	

Figure 8.1: ResNext Architecture

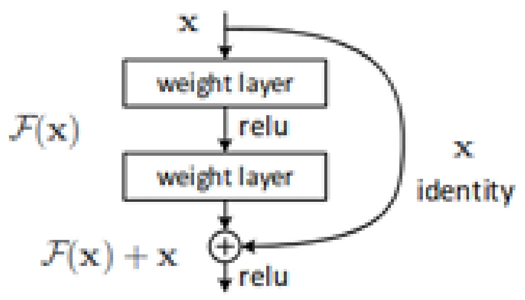


Figure 8.2: ResNext Working

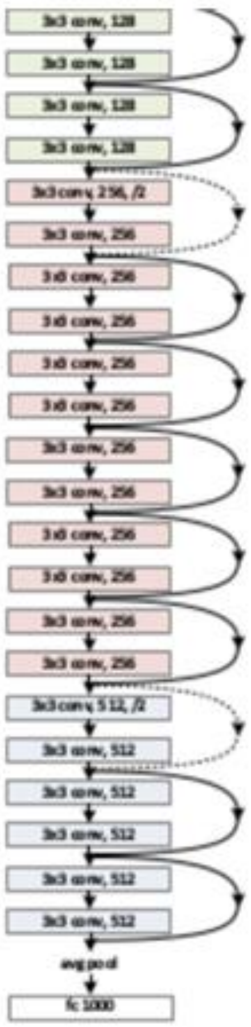


Figure 8.3: Overview of ResNext Architecture

- **Sequential Layer** : Sequential is a container of Modules that can be stacked together and run at the same time. Sequential layer is used to store feature vector returned by the ResNext model in a ordered way. So that it can be passed to the LSTM sequentially.
- **LSTM Layer :** LSTM is used for sequence processing and spot the temporal change between the frames.2048-dimensional feature vectors is fitted as the input to the LSTM. We are using 1 LSTM layer with 2048 latent dimensions and 2048 hidden layers along with 0.4 chance of dropout, which is capable to do achieve our objective. LSTM is used to process the frames in a sequential manner so that the temporal analysis of the video can be made, by comparing the frame at ‘t’ second with the frame of ‘t-n’ seconds. Where n can be any number of frames before t.

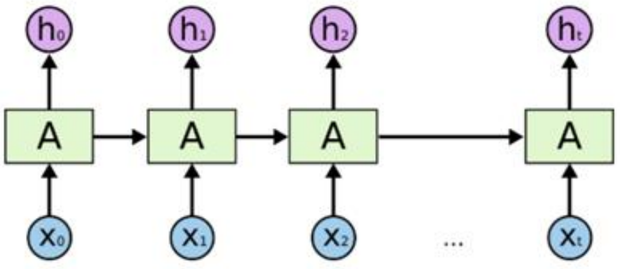


Figure 8.4: Overview of LSTM Architecture

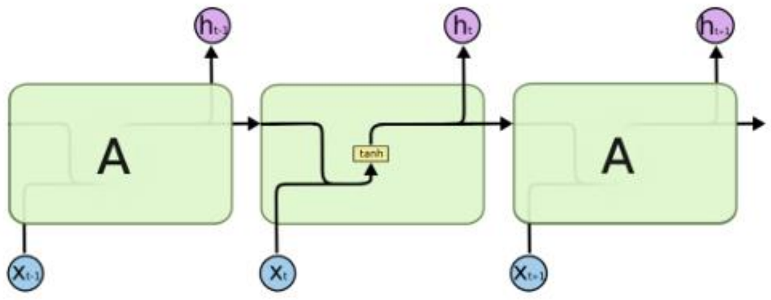


Figure 8.5: Internal LSTM Architecture

- **ReLU**:A Rectified Linear Unit is activation function that has output 0 if the input is less than 0, and raw output otherwise. That is, if the input is greater than 0, the output is equal to the input. The operation of ReLU is closer to the way our biological neurons work. ReLU is non-linear and has the advantage of not having any backpropagation errors unlike the sigmoid function, also for larger Neural Networks, the speed of building models based off on ReLU is very fast.

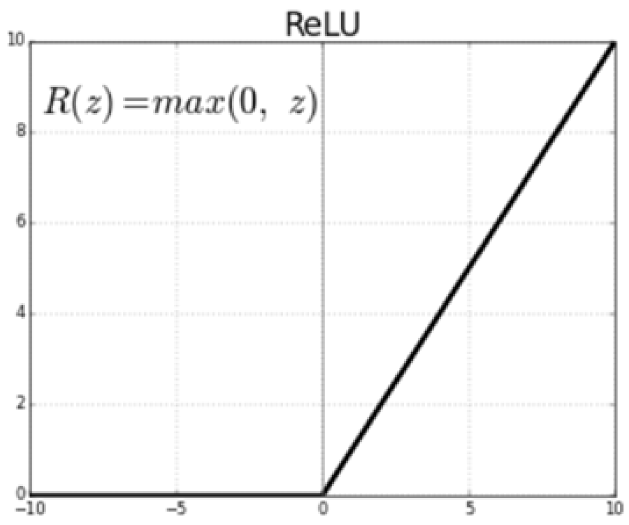


Figure 8.6: Relu Activation function

- **Dropout Layer** :Dropout layer with the value of 0.4 is used to avoid over fitting in the model and it can help a model generalize by randomly setting the output for a given neuron to 0. In setting the output to 0, the cost function becomes more sensitive to neighbouring neurons changing the way the weights will be updated during the process of backpropagation.

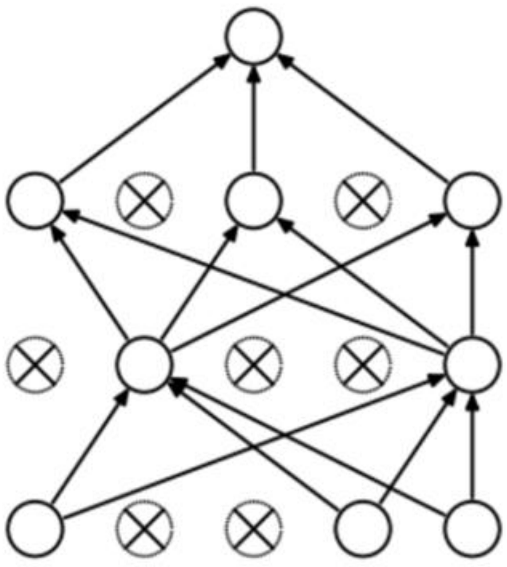


Figure 8.7: Dropout layer overview

- **Adaptive Average Pooling Layer :** It is used To reduce variance, reduce com putation complexity and extract low level features from neighbourhood.2 di mensional Adaptive Average Pooling Layer is used in the model.
- **8.3.4 Model Training Details**
- **Train Test Split:**The dataset is split into train and test dataset with a ratio of 70%train videos (4,200) and 30% (1,800) test videos. The train and test split is a balanced split i.e 50% of the real and 50% of fake videos in each split. Refer f igure 7.6
- **Data Loader**: It is used to load the videos and their labels with a batch size of 4.
- **Training:** The training is done for 20 epochs with a learning rate of 1e-5 (0.00001),weight decay of 1e-3 (0.001) using the Adam optimizer.
- **Adam optimizer[21]:** To enable the adaptive learning rate Adam optimizer with the model parameters is used.

- **Cross Entropy:** To calculate the loss function Cross Entropy approach is used because we are training a classification problem.
- **Softmax Layer:** A Softmax function is a type of squashing function. Squash ing functions limit the output of the function into the range 0 to 1. This allows the output to be interpreted directly as a probability. Similarly, softmaxfunctions are multi-class sigmoids, meaning they are used in determining probability of multiple classes at once. Since the outputs of a softmax function can be interpreted as a probability (i.e.they must sum to 1), a softmax layer is typically the final layer used in neural network functions. It is important to note that a softmax layer must have the same number of nodes as the output later. In our case softmax layer has two output nodes i.e REAL or FAKE, also Soft max layer provide us the confidence(probability) of prediction.

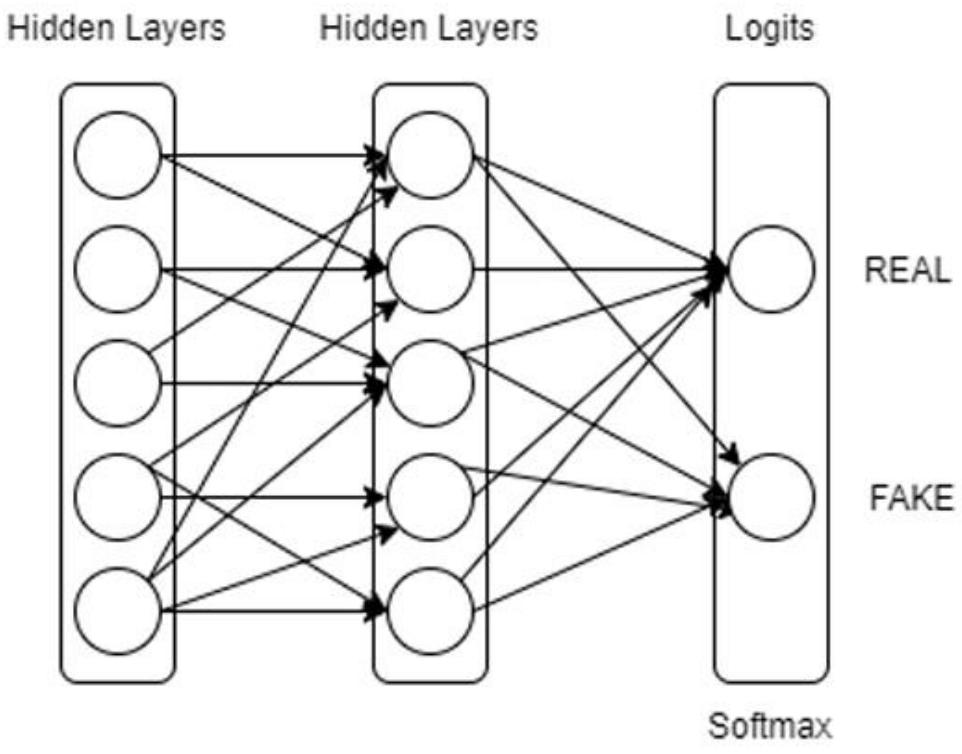


Figure 8.8: Softmax Layer

- **Confusion Matrix:** A confusion matrix is a summary of prediction results on a classification problem. The number of correct and incorrect predictions are summarized with count values and broken down by each class. This is the key to the confusion matrix. The confusion matrix shows the ways in which yourclassification model is confused when it makes predictions. It gives us insight not only into the errors being made by a classifier but more importantly the types of errors that are being made. Confusion matrix is used to evaluate our model and calculate the accuracy.

- **Export Model:** After the model is trained, we have exported the model. So that it can be used for prediction on real time data.

8.3.5 Model Prediction Details

- The model is loaded in the application

- The new video for prediction is preprocessed(refer 8.3.2, 7.2.2) and passed to the loaded model for prediction

- The trained model performs the prediction and return if the video is a real or fake along with the confidence of the prediction.

Chapter 9

Software Testing

9.1 Type of Testing Used

Functional Testing

1. Unit Testing

2. Integration Testing

3. System Testing

4. Interface Testing

Non-functional Testing

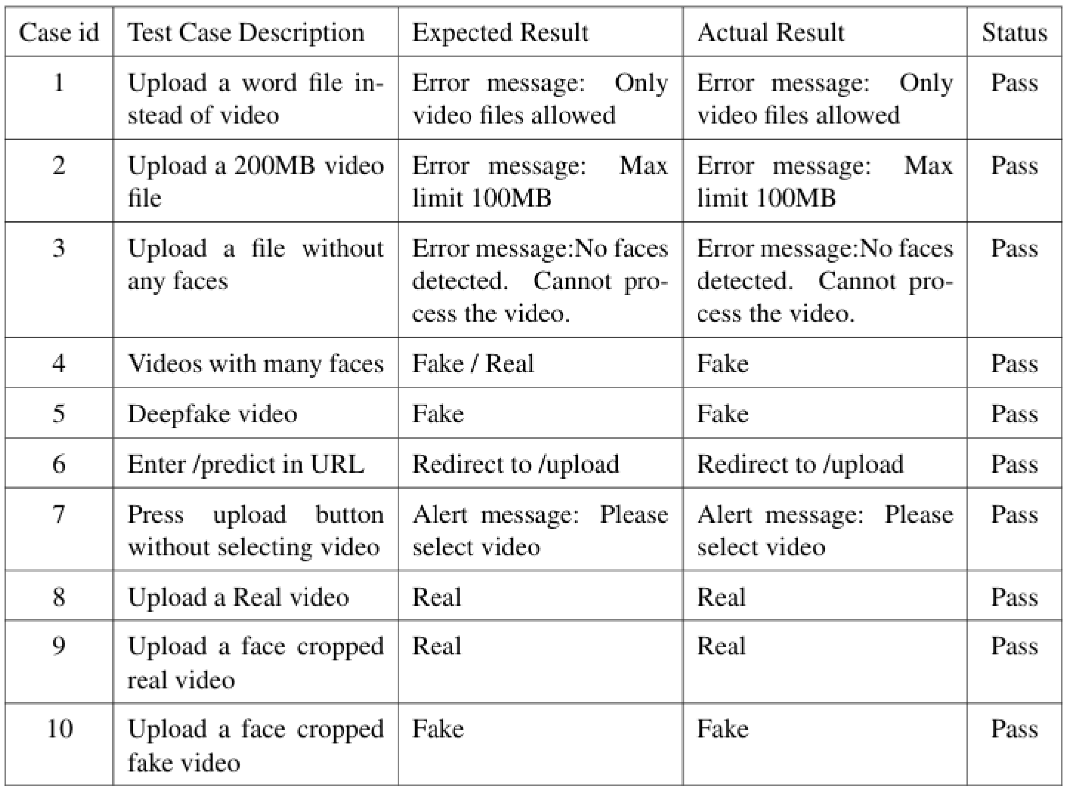
1. Performance Testing

2. Load Testing

3. Compatibility Testing.

9.2 TestCasesandTestResults

TestCases Table Table9.1:TestCaseReport



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

Chapter 10

Deployment and Maintenance

11.1 Deployment

Following are the steps to be followed for the deployment of the application.

1.clone the repository using the below command. git clone https://github.com/abhijitjadhav1998/Deefake_detection_Django_app.git Note: As its a private repository only authorized users will be able see the code and do the process of deployment

2.pipinstall torch===1.4.0 torchvision===0.5.0-f https://download.pytorch.org/whl/torch_stable.html

3. pip install-r requirement.txt

4. python manage.py migrate

5. Copy all the trained models into models folder.

6. python manage.py runserver 0.0.0.0:8000

11.2 Maintenance

Following are the steps to be followed for the updating the code to the latest version of the application.

1. Stop the production server using Ctrl + C.

2. git pull

Note: As its a private repository only authorized users will be able see the code and do the process of deployment 3. pip install-r requirement.txt

4. python manage.py migrate

5. Copy all the trained models into models folder(optional: Do this if any new models are added).

6. python manage.py runserver 0.0.0.0:8000

Chapter 11

Conclusion and Future Scope

12.1 Conclusion

We presented a neural network-based approach to classify the video as deep fake or real, along with the confidence of proposed model. Our method is capable of predicting the output by processing 1 second of video (10 frames per second) with a good accuracy. We implemented the model by using pre-trained ResNext CNN model to extract the frame level features and LSTM for temporal sequence process ing to spot the changes between the t and t-1 frame. Our model can process the video in the frame sequence of 10,20,40,60,80,100.

12.2 Future Scope

There is always a scope for enhancements in any developed system, especially when the project build using latest trending technology and has a good scope in future.

- Web based platform can be upscaled to a browser plugin for ease of access to the user.
- Currently only Face Deep Fakes are being detected by the algorithm, but the algorithm can be enhanced in detecting full body deep fakes.

APPENDIX 1

```
from django.shortcuts import render, redirect
import torch
import torchvision
from torchvision import transforms, models
from torch.utils.data import DataLoader
from torch.utils.data.dataset import Dataset
import os
import numpy as np
import cv2
import matplotlib.pyplot as plt
import face_recognition
from torch.autograd import Variable
import time
import sys
from torch import nn
import json
import glob
import copy
from torchvision import models
import shutil
from PIL import Image as pImage
import time
from django.conf import settings
from .forms import VideoUploadForm
```

```
index_template_name = 'index.html'
predict_template_name = 'predict.html'
about_template_name = "about.html"

im_size = 112
mean=[0.485, 0.456, 0.406]
std=[0.229, 0.224, 0.225]
sm = nn.Softmax()
inv_normalize = transforms.Normalize(mean=-1*np.divide(mean,std),std=np.divide([1,1,1],std))
if torch.cuda.is_available():
    device = 'gpu'
else:
    device = 'cpu'

train_transforms = transforms.Compose([
transforms.ToPILImage(),
transforms.Resize((im_size,im_size)),
transforms.ToTensor(),
transforms.Normalize(mean,std)])

class Model(nn.Module):

    def __init__(self, num_classes,latent_dim= 2048, lstm_layers=1 , hidden_dim = 2048, bidirectional = False):
        super(Model, self).__init__()
        model = models.resnext50_32x4d(pretrained = True)
```

```
self.model = nn.Sequential(*list(model.children())[:-2])
self.lstm = nn.LSTM(latent_dim,hidden_dim, lstm_layers,  bidirectional)
self.relu = nn.LeakyReLU()
self.dp = nn.Dropout(0.4)
        self.linear1 = nn.Linear(2048,num_classes)
self.avgpool = nn.AdaptiveAvgPool2d(1)

    def forward(self, x):
batch_size,seq_length, c, h, w = x.shape
        x = x.view(batch_size * seq_length, c, h, w)
fmap = self.model(x)
        x = self.avgpool(fmap)
        x = x.view(batch_size,seq_length,2048)
x_lstm,_ = self.lstm(x,None)
        return fmap,self.dp(self.linear1(x_lstm[:,-1,:]))



class validation_dataset(Dataset):
    def __init__(self,video_names,sequence_length=60,transform = None):
self.video_names = video_names
self.transform = transform
self.count = sequence_length

    def __len__(self):
        return len(self.video_names)

    def __getitem__(self,idx):
```

```
video_path = self.video_names[idx]
        frames = []
        a = int(100/self.count)
first_frame = np.random.randint(0,a)
        for i,frame in enumerate(self.frame_extract(video_path)):
            #if(i % a == first_frame):
            faces = face_recognition.face_locations(frame)
            try:
top,right,bottom,left = faces[0]
              frame = frame[top:bottom,left:right,:]
            except:
              pass
frames.append(self.transform(frame))
            if(len(frames) == self.count):
                break
        """
        for i,frame in enumerate(self.frame_extract(video_path)):
            if(i % a == first_frame):
frames.append(self.transform(frame))
        """
        # if(len(frames)<self.count):
        #   for i in range(self.count-len(frames)):
        #         frames.append(self.transform(frame))
        #print("no of frames", self.count)
        frames = torch.stack(frames)
        frames = frames[:self.count]
        return frames.unsqueeze(0)
```

```
    def frame_extract(self,path):
vidObj = cv2.VideoCapture(path)
      success = 1
      while success:
          success, image = vidObj.read()
          if success:
              yield image

def im_convert(tensor, video_file_name):
    """ Display a tensor as an image. """
    image = tensor.to("cpu").clone().detach()
    image = image.squeeze()
    image = inv_normalize(image)
    image = image.numpy()
    image = image.transpose(1,2,0)
    image = image.clip(0, 1)
    # This image is not used
    #      cv2.imwrite(os.path.join(settings.PROJECT_DIR,      'uploaded_images',
video_file_name+'_convert_2.png'),image*255)
    return image

def im_plot(tensor):
    image = tensor.cpu().numpy().transpose(1,2,0)
b,g,r = cv2.split(image)
    image = cv2.merge((r,g,b))
    image   =   image*[0.22803,   0.22145,   0.216989]   +    [0.43216,   0.394666,
0.37645]
```

```
    image = image*255.0
plt.imshow(image.astype('uint8'))
plt.show()



def predict(model,img,path = './', video_file_name=""):
fmap,logits = model(img.to(device))
img = im_convert(img[:,-1,:,:,:], video_file_name)
  params = list(model.parameters())
weight_softmax = model.linear1.weight.detach().cpu().numpy()
  logits = sm(logits)
  _,prediction = torch.max(logits,1)
  confidence = logits[:,int(prediction.item())].item()*100
  print('confidence of prediction:',logits[:,int(prediction.item())].item()*100)
  return [int(prediction.item()),confidence]


def plot_heat_map(i, model, img, path = './', video_file_name=''):
fmap,logits = model(img.to(device))
  params = list(model.parameters())
weight_softmax = model.linear1.weight.detach().cpu().numpy()
  logits = sm(logits)
  _,prediction = torch.max(logits,1)
idx = np.argmax(logits.detach().cpu().numpy())
bz, nc, h, w = fmap.shape
  #out          =          np.dot(fmap[-1].detach().cpu().numpy().reshape((nc,
h*w)).T,weight_softmax[idx,:].T)
  out          =          np.dot(fmap[i].detach().cpu().numpy().reshape((nc,
h*w)).T,weight_softmax[idx,:].T)
```

```
  predict = out.reshape(h,w)
  predict = predict - np.min(predict)
predict_img = predict / np.max(predict)
predict_img = np.uint8(255*predict_img)
  out = cv2.resize(predict_img, (im_size,im_size))
  heatmap = cv2.applyColorMap(out, cv2.COLORMAP_JET)
img = im_convert(img[:,-1,:,:,:], video_file_name)
  result = heatmap * 0.5 + img*0.8*255
  # Saving heatmap - Start
heatmap_name = video_file_name+"_heatmap_"+str(i)+".png"
image_name = os.path.join(settings.PROJECT_DIR, 'uploaded_images', heatmap_name)
  cv2.imwrite(image_name,result)
  # Saving heatmap - End
  result1 = heatmap * 0.5/255 + img*0.8
r,g,b = cv2.split(result1)
  result1 = cv2.merge((r,g,b))
  return image_name


# Model Selection
def get_accurate_model(sequence_length):
model_name = []
sequence_model = []
final_model = ""
list_models = glob.glob(os.path.join(settings.PROJECT_DIR, "models", "*.pt"))


    for model_path in list_models:
```

```
model_name.append(os.path.basename(model_path))

    for model_filename in model_name:
        try:
seq = model_filename.split("_")[3]
            if int(seq) == sequence_length:
sequence_model.append(model_filename)
        except IndexError:
            pass  # Handle cases where the filename format doesn't match expected

    if len(sequence_model) > 1:
        accuracy = []
        for filename in sequence_model:
acc = filename.split("_")[1]
accuracy.append(acc)  # Convert accuracy to float for proper comparison
max_index = accuracy.index(max(accuracy))
final_model = os.path.join(settings.PROJECT_DIR, "models", sequence_model[max_index])
eliflen(sequence_model) == 1:
final_model = os.path.join(settings.PROJECT_DIR, "models", sequence_model[0])
    else:
        print("No model found for the specified sequence length.")  # Handle no models found case

    return final_model

ALLOWED_VIDEO_EXTENSIONS = set(['mp4','gif','webm','avi','3gp','wmv','flv','mkv'])
```

```
def allowed_video_file(filename):
   #print("filename" ,filename.rsplit('.',1)[1].lower())
   if (filename.rsplit('.',1)[1].lower() in ALLOWED_VIDEO_EXTENSIONS):
      return True
   else:
      return False
def index(request):
   if request.method == 'GET':
video_upload_form = VideoUploadForm()
      if 'file_name' in request.session:
         del request.session['file_name']
      if 'preprocessed_images' in request.session:
         del request.session['preprocessed_images']
      if 'faces_cropped_images' in request.session:
         del request.session['faces_cropped_images']
      return render(request, index_template_name, {"form": 
video_upload_form})
   else:
video_upload_form = VideoUploadForm(request.POST, request.FILES)
      if video_upload_form.is_valid():
video_file = video_upload_form.cleaned_data['upload_video_file']
video_file_ext = video_file.name.split('.')[-1]
sequence_length = video_upload_form.cleaned_data['sequence_length']
video_content_type = video_file.content_type.split('/')[0]
         if video_content_type in settings.CONTENT_TYPES:
            if video_file.size> int(settings.MAX_UPLOAD_SIZE):
```

```
video_upload_form.add_error("upload_video_file", "Maximum file size 100
MB")
            return render(request, index_template_name, {"form":
video_upload_form})


        if sequence_length<= 0:
video_upload_form.add_error("sequence_length", "Sequence Length must be
greater than 0")
          return render(request, index_template_name, {"form":
video_upload_form})


        if allowed_video_file(video_file.name) == False:
video_upload_form.add_error("upload_video_file","Only video files are
allowed ")
          return render(request, index_template_name, {"form":
video_upload_form})


saved_video_file = 'uploaded_file_'+str(int(time.time()))+"."+video_file_ext
        if settings.DEBUG:
          with open(os.path.join(settings.PROJECT_DIR, 'uploaded_videos',
saved_video_file), 'wb') as vFile:
shutil.copyfileobj(video_file, vFile)
request.session['file_name'] = os.path.join(settings.PROJECT_DIR,
'uploaded_videos', saved_video_file)
        else:
          with open(os.path.join(settings.PROJECT_DIR,
'uploaded_videos','app','uploaded_videos', saved_video_file), 'wb') as vFile:
shutil.copyfileobj(video_file, vFile)
request.session['file_name'] = os.path.join(settings.PROJECT_DIR,
'uploaded_videos','app','uploaded_videos', saved_video_file)
request.session['sequence_length'] = sequence_length
```

```
            return redirect('ml_app:predict')
        else:
            return render(request, index_template_name, {"form": video_upload_form})

def predict_page(request):
    if request.method == "GET":
        # Redirect to 'home' if 'file_name' is not in session
        if 'file_name' not in request.session:
            return redirect("ml_app:home")
        if 'file_name' in request.session:
video_file = request.session['file_name']
        if 'sequence_length' in request.session:
sequence_length = request.session['sequence_length']
path_to_videos = [video_file]
video_file_name = os.path.basename(video_file)
video_file_name_only = os.path.splitext(video_file_name)[0]
        # Production environment adjustments
        if not settings.DEBUG:
production_video_name = os.path.join('/home/app/staticfiles/', video_file_name.split('/')[3])
            print("Production file name", production_video_name)
        else:
production_video_name = video_file_name

        # Load validation dataset
video_dataset = validation_dataset(path_to_videos, sequence_length=sequence_length, transform=train_transforms)
```

```
        # Load model
        if(device == "gpu"):
            model = Model(2).cuda()  # Adjust the model instantiation according to
your model structure
        else:
            model = Model(2).cpu()  # Adjust the model instantiation according to
your model structure
model_name = os.path.join(settings.PROJECT_DIR, 'models',
get_accurate_model(sequence_length))
path_to_model = os.path.join(settings.PROJECT_DIR, model_name)
model.load_state_dict(torch.load(path_to_model,
map_location=torch.device('cpu')))
model.eval()
start_time = time.time()
        # Display preprocessing images
        print("<=== | Started Videos Splitting | ===>")
preprocessed_images = []
faces_cropped_images = []
        cap = cv2.VideoCapture(video_file)
        frames = []
        while cap.isOpened():
            ret, frame = cap.read()
            if ret:
frames.append(frame)
            else:
                break
cap.release()
```

```
        print(f"Number of frames: {len(frames)}")
        # Process each frame for preprocessing and face cropping
        padding = 40
faces_found = 0
        for i in range(sequence_length):
            if i>= len(frames):
                break
            frame = frames[i]

            # Convert BGR to RGB
rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

            # Save preprocessed image
image_name = f"{video_file_name_only}_preprocessed_{i+1}.png"
image_path = os.path.join(settings.PROJECT_DIR, 'uploaded_images', image_name)
img_rgb = pImage.fromarray(rgb_frame, 'RGB')
img_rgb.save(image_path)
preprocessed_images.append(image_name)

            # Face detection and cropping
face_locations = face_recognition.face_locations(rgb_frame)
            if len(face_locations) == 0:
                continue

            top, right, bottom, left = face_locations[0]
frame_face = frame[top - padding:bottom + padding, left - padding:right + padding]
```

```
            # Convert cropped face image to RGB and save
rgb_face = cv2.cvtColor(frame_face, cv2.COLOR_BGR2RGB)
img_face_rgb = pImage.fromarray(rgb_face, 'RGB')
image_name = f"{video_file_name_only}_cropped_faces_{i+1}.png"
image_path = os.path.join(settings.PROJECT_DIR, 'uploaded_images',
image_name)
img_face_rgb.save(image_path)
faces_found += 1
faces_cropped_images.append(image_name)

        print("<=== | Videos Splitting and Face Cropping Done | ===>")
        print("--- %s seconds ---" % (time.time() - start_time))

        # No face detected
        if faces_found == 0:
            return render(request, 'predict_template_name.html', {"no_faces":
True})

        # Perform prediction
        try:
heatmap_images = []
            output = ""
            confidence = 0.0

            for i in range(len(path_to_videos)):
                print("<=== | Started Prediction | ===>")
                prediction = predict(model, video_dataset[i], './',
video_file_name_only)
                confidence = round(prediction[1], 1)
                output = "REAL" if prediction[0] == 1 else "FAKE"
                print("Prediction:", prediction[0], "==", output, "Confidence:",
confidence)
                print("<=== | Prediction Done | ===>")
                print("--- %s seconds ---" % (time.time() - start_time))

                # Uncomment if you want to create heat map images
                # for j in range(sequence_length):
                #     heatmap_images.append(plot_heat_map(j, model,
video_dataset[i], './', video_file_name_only))

            # Render results
            context = {
                'preprocessed_images': preprocessed_images,
                'faces_cropped_images': faces_cropped_images,
                'heatmap_images': heatmap_images,
                'original_video': production_video_name,
                'models_location': os.path.join(settings.PROJECT_DIR, 'models'),
                'output': output,
                'confidence': confidence
            }

            if settings.DEBUG:
                return render(request, predict_template_name, context)
            else:
            return render(request, predict_template_name, context)


        except Exception as e:
            print(f"Exception occurred during prediction: {e}")
            return render(request, 'cuda_full.html')
def about(request):
    return render(request, about_template_name)


def handler404(request,exception):
    return render(request, '404.html', status=404)
def cuda_full(request):
    return render(request, 'cuda_full.html')
```

APPENDIX 2

Results and Discussion

10.1 Screen shot

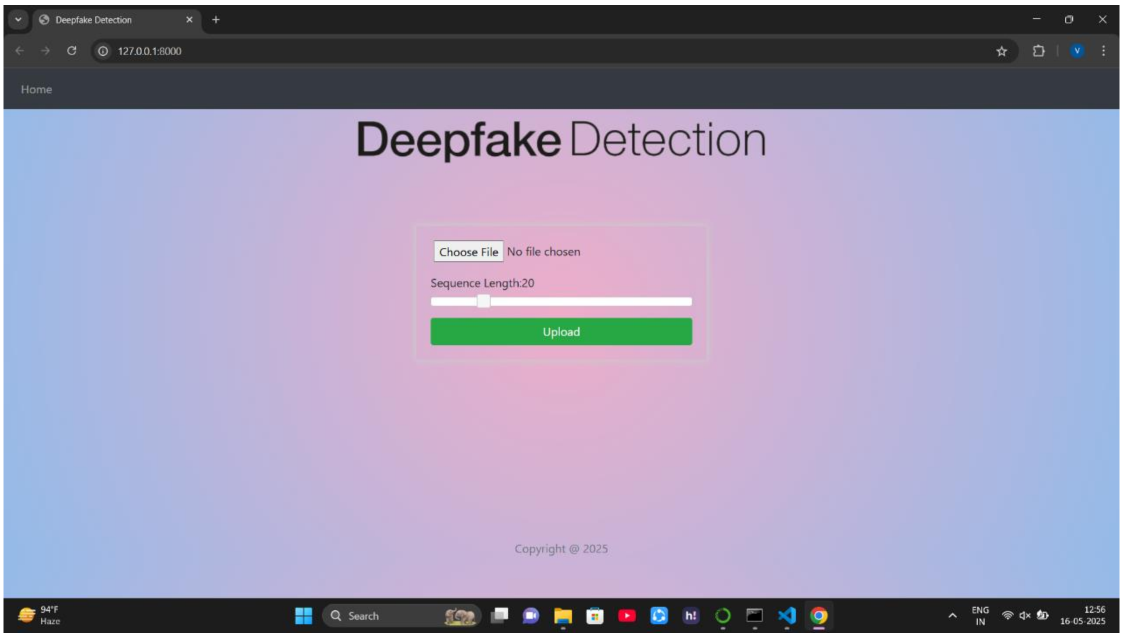


Figure 10.1: Home Page

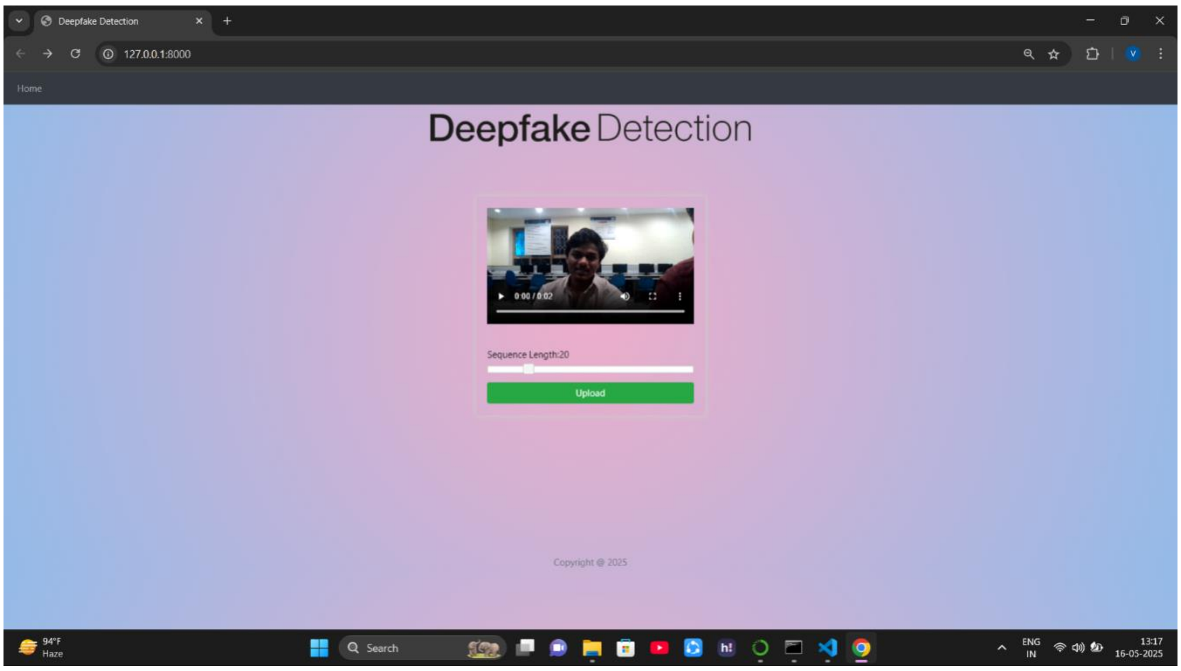


Figure 10.2: Uploading Real Video

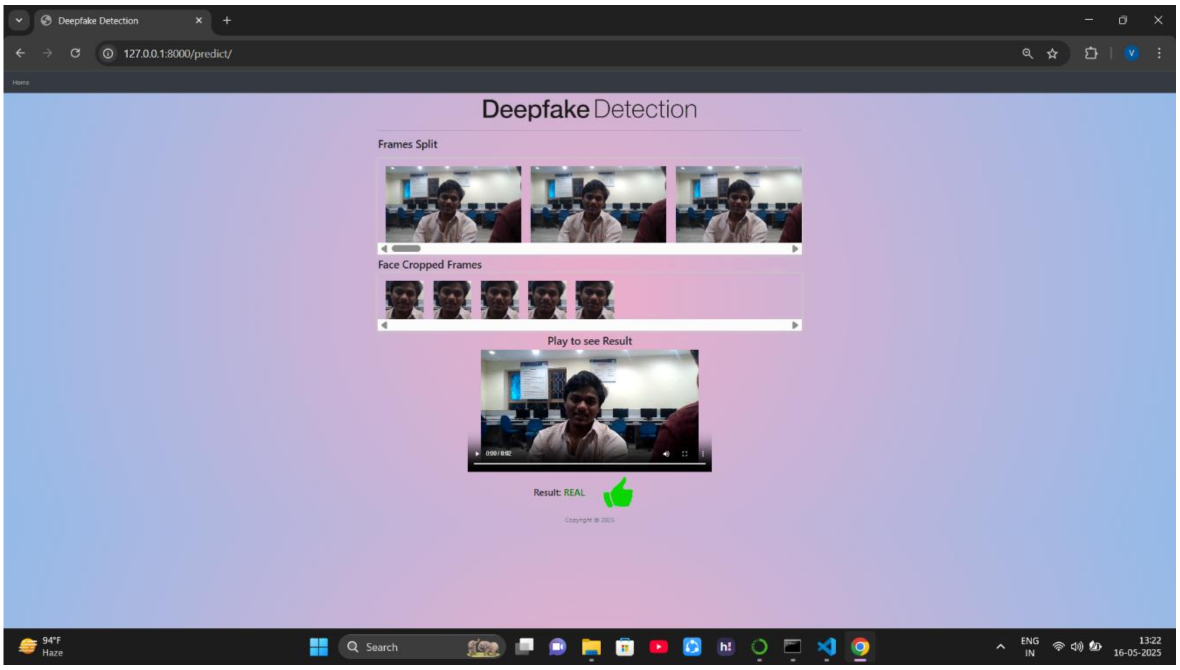


Figure 10.3: Real Video Output

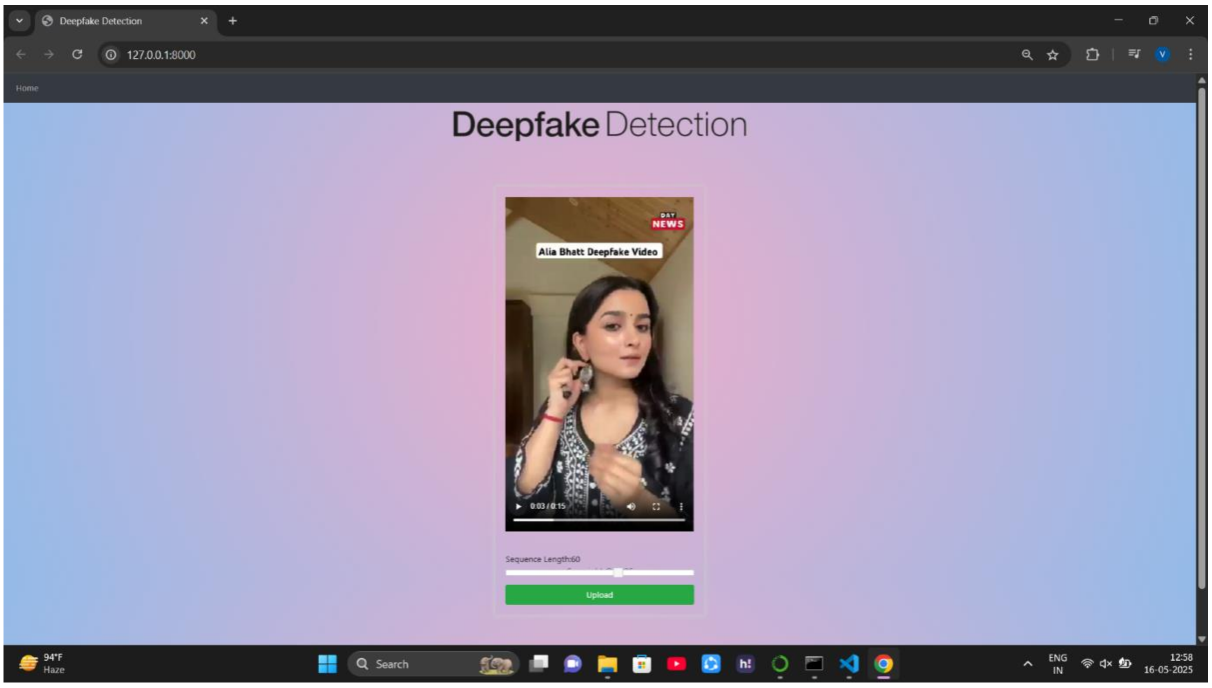


Figure 10.4: Uploading Fake Video

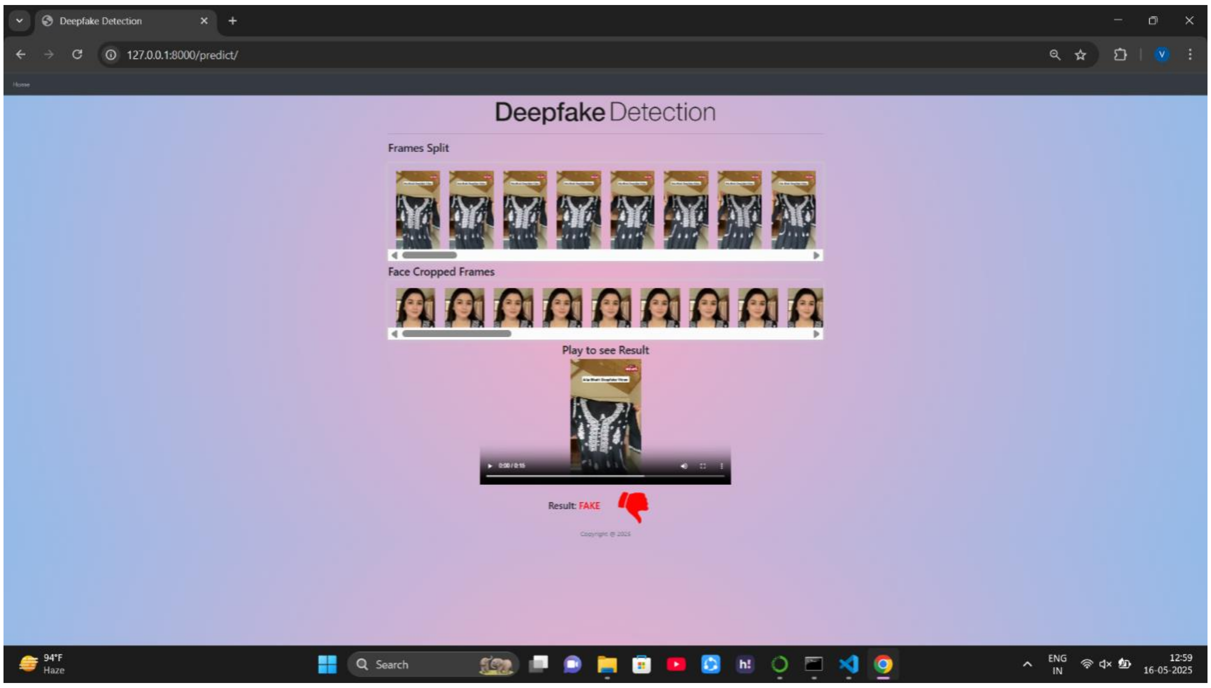
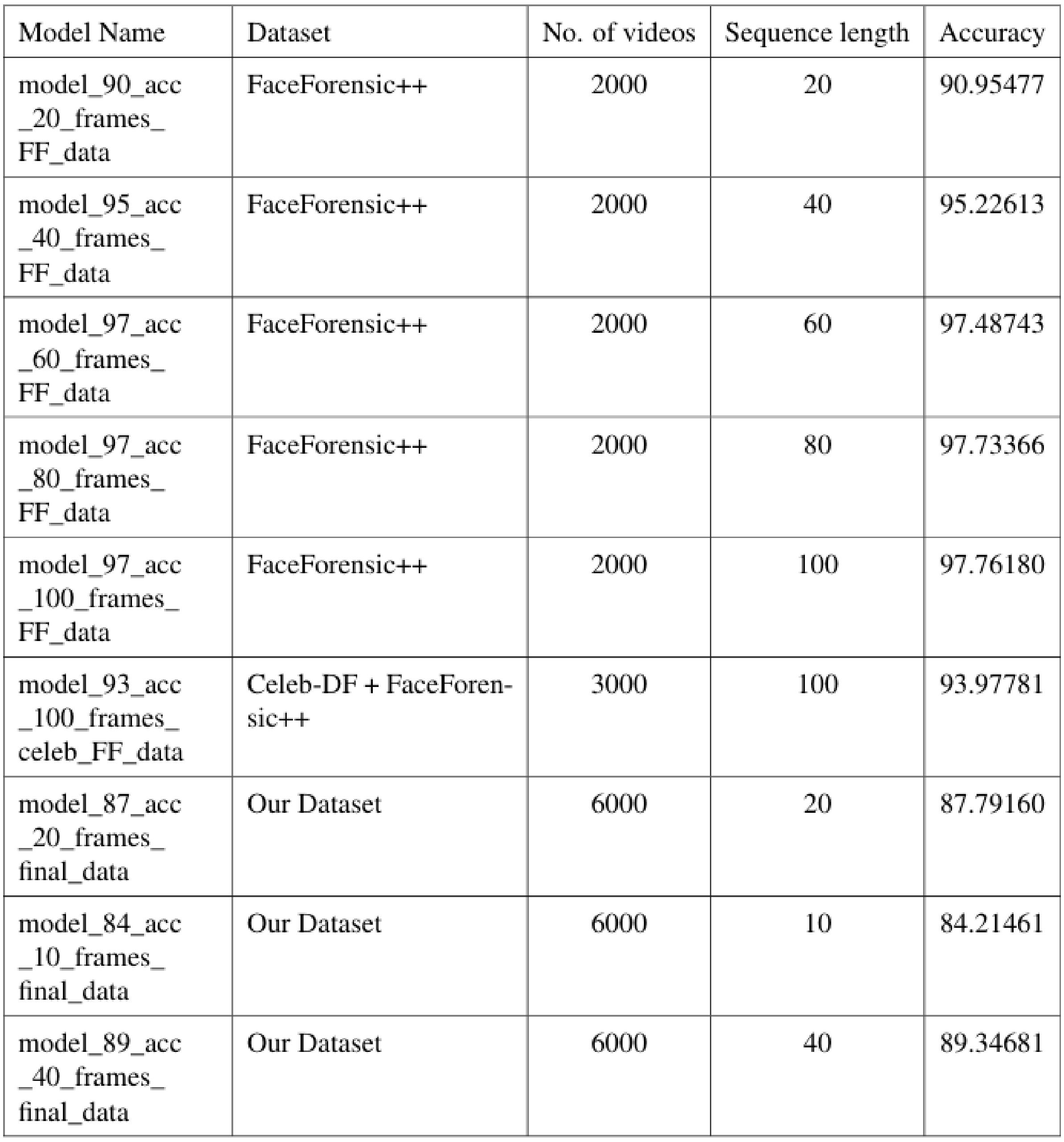


Figure 10.5: Fake video Output

Model results

Table 10.1: Trained Model Results



Model Name	Dataset	No. of videos	Sequence length	Accuracy
model_90_acc _20_frames_ FF_data	FaceForensic++	2000	20	90.95477
model_95_acc _40_frames_ FF_data	FaceForensic++	2000	40	95.22613
model_97_acc _60_frames_ FF_data	FaceForensic++	2000	60	97.48743
model_97_acc _80_frames_ FF_data	FaceForensic++	2000	80	97.73366
model_97_acc _100_frames_ FF_data	FaceForensic++	2000	100	97.76180
model_93_acc _100_frames_ celeb_FF_data	Celeb-DF + FaceForen-sic++	3000	100	93.97781
model_87_acc _20_frames_ final_data	Our Dataset	6000	20	87.79160
model_84_acc _10_frames_ final_data	Our Dataset	6000	10	84.21461
model_89_acc _40_frames_ final_data	Our Dataset	6000	40	89.34681

References

01. Deng, Liwei, Hongfei Suo, and Dongjie Li. "Deepfake Video EfficientNet-V2 Detection Based on Network." Computational Intelligence and Neuroscience 2022.1 (2022): 3441549.

02. Khan, Sohail Ahmed, and Duc-Tien Dang Nguyen. "Hybrid transformer network for deepfake detection." Proceedings of the 19th international conference on content-based multimedia indexing. 2022.

03. Coccomini, Davide Alessandro, et al. "Cross-forgery analysis of vision transformers and cnns for deepfake image detection." Proceedings of the 1st International Workshop on Multimedia Disinformation. 2022. AI against

04. Coccomini, Davide Alessandro, et al. "Combining efficientnet and vision transformers for video deepfake detection." International conference on image analysis and processing. Cham: Springer International Publishing, 2022.

05. Mahmud, Faysal, et al. "Unmasking deepfake faces from videos using an explainable cost-sensitive deep learning approach." 2023 26th International Conference on Computer and Information Technology (ICCIT). IEEE, 2023.

06. Yasser, Basma, et al. "Deepfake Detection Using EfficientNet and XceptionNet." 2023 Eleventh International Conference on Intelligent Computing and Information Systems (ICICIS). IEEE, 2023

07. Verdoliva, Luisa. "Media forensics and deepfakes: an overview." IEEE journal of selected topics in signal processing 14.5 (2020): 910-932.

08Matern, Falko, Christian Riess, and Marc Stamminger. "Exploiting visual artifacts to expose deepfakes and face manipulations." 2019 IEEE Winter Applications of Computer Vision Workshops (WACVW). IEEE, 2019

09. Li, Lingzhi, et al. "Face x-ray for more general face forgery detection." Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020.

10. Guarnera, Luca, Oliver Giudice, and Sebastiano Battiato. "Deepfake detection by analyzing convolutional traces." Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops. 2020.