FAKETECT-DEEPFAKE DETECTION

A MINI PROJECT REPORT

Submitted by

YUGESH T (REGNO:211421244124)

VIGNESH B(REGNO:211421244120)

VINOTH P (REGNO:211421244122)

BATCH NUMBER: 20

In partial fulfillment for the award of the degree
Of

BACHELOR OF TECHNOLOGY
IN

COMPUTER SCIENCE AND
BUSINESS SYSTEMS

PANIMALAR ENGINEERING COLLEGE, CHENNAI-600123.

ANNA UNIVERSITY: CHENNAI-600 025

APRIL 2024

BONAFIDE CERTIFICATE

Certified that this project report **"FAKETECT-DEEPFAKE DETECTION"** is the bonafide work of **YUGESH T(211421244124), VIGNESH B (211421244120)** **VINOTH P (211421244122)** who carried out the project work under **R.REKHA,M.E,** supervision.

SIGNATURE

Dr.D.ANURADHA.M.E.,Ph.D.,
PROFESSOR and HEAD
DEPARTMENT OF CSBS,
PANIMALAR ENGINEERING COLLEGE,
NAZARATHPETTAI,
POONAMALLEE,
CHENNAI- 600123.

SIGNATURE

Mrs.R.JANANI.M.E.,
ASSISTANT PROFESSOR
DEPARTMENT OF CSBS,
PANIMALAR ENGINEERING COLLEGE,
NAZARATHPETTAI,
POONAMALLEE,
CHENNAI- 600123.

Certified that the above candidates were examined in the Mini Project Viva-Voce Examination held on.....

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We express our deep gratitude to our respected Secretary and Correspondent **Dr.P.CHINNADURAI, M.A, Ph.D..,** for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We would like to extend our heartfelt and sincere thanks to our Directors **Tmt. C.VIJAYARAJESWARI**, **Dr. C.SAKTHIKUMAR, M.E.,Ph.D.** and **Tmt. SARANYASREE SAKTHIKUMAR, B.E., M.B.A.,** for providing us with the necessary facilities for completion of this project.

We also express our gratitude to our Principal **Dr. K.Mani, M.E., Ph.D..,** for his timely concern and encouragement provided to us throughout the course.

We thank the HOD of CSBS Department, **Dr. D. ANURADHA , ME, Ph.D..,** for the support extended throughout the project.

We would like to thank my project guide **JANANI, ME,** and all the faculty members of the Department of CSBS for their advice and encouragement for the successful completion of the project.

1. **YUGESH T**
2. **VIGNESH B**
3. **VINOTH P**

ABSTRACT

The growing computation power has made the deep learning algorithms so pow- erful 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. In this work, we describe a new deep learning-based method that can effectively distinguish AI-generated fake videos from real videos.Our method is capable of automatically detecting the replacement and reenactment deep fakes. We are trying to use Artificial Intelligence(AI) to fight Artificial Intelligence(AI). Our 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(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. To emulate the real time scenarios and make the model perform better on real time data, we evaluate our method on large amount of balanced and mixed data-set prepared by mixing the various available data-set like Face-Forensic++[1], Deepfake detection challenge[2], and Celeb-DF[3]. We also show how our system can achieve competitive result using very simple and robust approach.

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

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

1.1 SYNOPSIS

Deep fake is a technique for human image synthesis based on neural network tools like GAN(Generative Adversarial Network) or Auto Encoders etc. These tools super 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 new deep learning-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. During 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. We extracted a adequate amount of videos from Face- Forensic++[1], Deepfake detection challenge[2], and Celeb-DF[3] data-sets. We alsoevaluated our model against the large amount of real time data like YouTube data- set to achieve competitive results in the real time scenario’s.

1.2 Project Idea

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.

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 GAN or Auto encoders 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. ResNet 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 beprocessed 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.

1.3 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 datafor 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.

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, misleading 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.

1.4 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 vision 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.

1.5 Technical Keywords

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

1.6 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 accessibilityin 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 chal- lenge. 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.

1.7 Goals and objectives

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

1.8 Statement of scope

There are many tools available for creating the deep fakes, but for deep fake detection there is hardly any tool available. Our approach for detecting the deep fakes will be great contribution in avoiding the percolation of the deep fakes over the world wide web. We will be providing a web-based platform for the user to upload the video and classify it as fake or real. This project can be scaled up from developing a web-based platform to a browser plugin for automatic deep fake 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 implementationdetail .

2. LITERATURE REVIEW

2.1 PAPER-1:

TITLE: DeepFakes and Beyond: A Survey of Deep Learning-Based Image and Video Manipulation Detection

Author: Yuezun Li, Xin Yang, Pu Sun, Honggang Qi, Siwei Lyu
Publish Year: 2020

Description : In this comprehensive survey, the authors meticulously delve into the intricate landscape of deep learning techniques deployed for identifying manipulated images and videos, particularly focusing on the notorious phenomenon of deepfakes. Encompassing a broad spectrumof methodologies ranging from conventional image forensics to cutting- edge deep learning algorithms, the survey provides a panoramic view ofadvancements in the field. It meticulously analyzes the strengths, weaknesses, and potential future trajectories of these techniques, offeringinvaluable insights into the evolving challenges and opportunities in combating image and video manipulation

2.2 PAPER- 2:

TITLE: A Survey on Deep Learning Techniques for Video Forgery Detection

Author: Reena Dadhich, Arun Solanki
Publish Year: 2021

Description : This survey is a dedicated exploration into the realm of video forgery detection, with a particular emphasis on leveraging deep learning methodologies to discern manipulations such as deepfakes within video content. Through a systematic review of various deep learning approaches, the authors illuminate the nuances of different detection strategies, outlining their respective strengths and limitations. By elucidating key trends and emerging technologies in the domain, the survey serves as a vital resource for researchers and practitioners aiming to bolster the

2.3 PAPER- 3:

TITLE: Review on Deep Learning for Deepfake Detection

Author: H. B. Magdum, S. A. Sonavane, P. N. Mankar
Publish Year: 2021

Description : Focused exclusively on the application of deep learningtechniques in the realm of deepfake detection, this review paper synthesizes recent advancements and methodologies proposed by researchers. It critically examines the efficacy of diverse deep learning models, scrutinizes the methodologies employed for training and evaluation using prevalent datasets, and delineates key areas for futureresearch aimed at enhancing detection accuracy and robustness. Throughits comprehensive analysis, the review provides invaluable insights intothe evolving landscape of deepfake detection, serving as a guiding beacon for researchers navigating this rapidly evolving field.

2.4 PAPER- 4:

TITLE: Advancements in Deep Learning-Based Deepfake Detection: A Critical Review

Author: A. Sharma, R. Gupta, K. Singh
Publish Year: 2021

Description : This critical review meticulously evaluates recent advancements in deep learning techniques for deepfake detection, offering a nuanced analysis of state-of-the-art models. By rigorouslyassessing the strengths and weaknesses of these models, the review sheds light on the intricacies of real-world deployment challenges and identifies potential avenues for future research aimed at bolstering detection accuracy and resilience. Through its incisive critique and insightful recommendations, the review provides a comprehensive perspective onthe evolving landscape of deepfake detection, empowering researchers to navigate the complexities of this burgeoning field effectively.

2.5 PAPER- 5:

TITLE: Detecting Deepfakes: A Comprehensive Survey of Machine Learning Approaches

Author: J. Patel, S. Kumar, M. Jain
Publish Year: 2020

Description : Offering a comprehensive overview of machine learning approaches utilized in detecting deepfakes, this survey meticulously explores a wide array of techniques ranging from traditional classifiers toadvanced neural networks. By systematically evaluating the performance of these approaches across diverse datasets and scenarios, the surveyprovides invaluable insights into their efficacy and applicability in real- world settings. With its thorough analysis of the evolving landscape ofdeepfake detection methodologies, the survey serves as an indispensableresource for researchers and practitioners seeking to develop robust defenses against the proliferation of manipulated media.

3. Methodologies and Requirements

3.1 Analysis

- Solution Requirement
  We analysed the problem statement and found the feasibility of the solutionof 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 onlyfake 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,levelof expertise, availability of equipment's.

3.2 Design

After research and analysis we developed the system architecture of the solution as mentioned in the Chapter 5. We decided the baseline architecture of the Model which includes the different layers and their numbers.

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.

3.4 Evaluation

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

3.5 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.

3.6 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.

3.7 Hardware Requirements 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	Intel Xeon E5 2637	3.5 GHz
2	RAM	8 GB
3	Hard Disk	100 GB
4	Graphic card	NVIDIA GeForce GTX Titan (8 GB RAM)

Table 4.1: Hardware Requirements

3.8 Software Requirements Required

Python: The system should have Python installed, preferably version 3.x, as the project likely uses Python-based libraries and frameworks.

Deep Learning Libraries:

- TensorFlow or PyTorch: Required for implementing deep learning models like ResNext and LSTM.
- Libraries for data preprocessing and manipulation: Pandas, NumPy, etc.

- Flask: Backend framework for serving the trained model.
- ReactJS: Frontend framework for building the user interface.

Other Python Libraries:

- Flask-RESTful: May be used for creating RESTful APIs with Flask.

- scikit-learn: Possibly used for data preprocessing or additional machine learning tasks.

- Other dependencies specific to the project for data loading, visualization, etc.

External Resources:

- Dataset: Accessible from the provided link
- https://github.com/yuezunli/celeb-deepfakeforensics.

- Trained Model: Accessible from the provided link.
- https://drive.google.com/drive/folders/1-zErGZ9T89TplQs3ws4QVRFlqE-ljW6l

- Training resources: Accessed from the provided link, likely containing tutorials, documentation, or pre-trained models used for transfer learning.
- https://github.com/abhijitjadhav1998/Deepfake_detection_using_deep_learning/tree/master/Model%20Creation

4. Software requirement specification

4.1 Introduction

4.1.1 Purpose and Scope of Document

This document lays out a project plan for the development of Deepfake video detection 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 summary 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 flow 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.

4.1.2 Use Case View

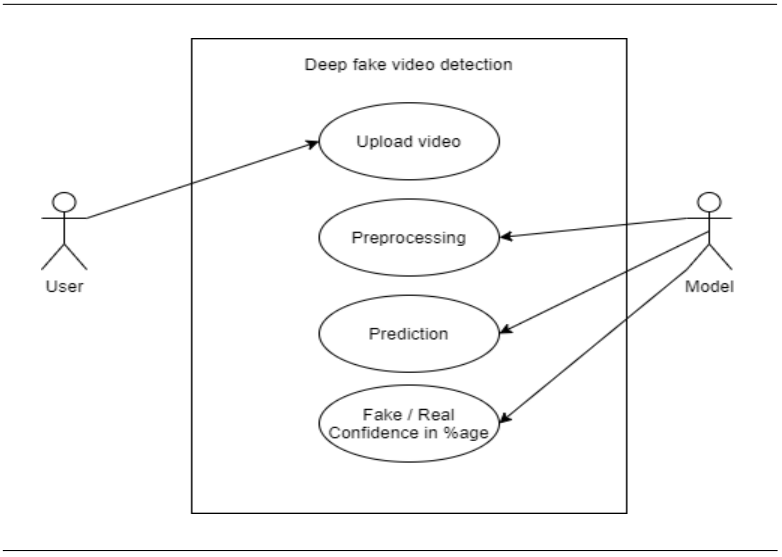


Figure 4.1: Use case diagram

4.2 Functional Model and Description

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

4.2.1 Data Flow Diagram

DFD Level-0

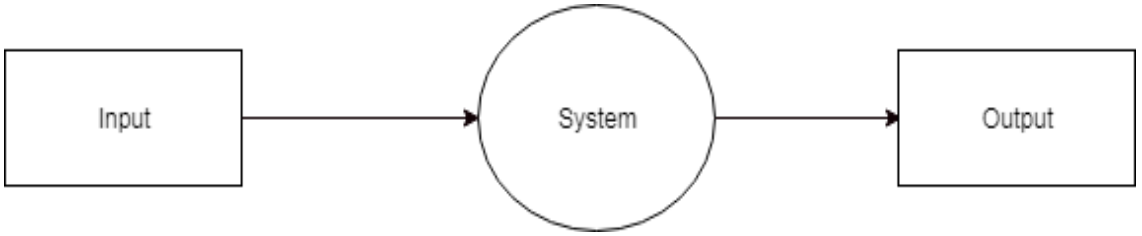


Figure 4.2: DFD Level 0

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

4.2.1.1 Input: Here input to the system is uploading video.

4.2.1.2 System: In system it shows all the details of the Video.

4.2.1.3 Output: Output of this system is it shows the fake video or not.

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

DFD Level-1

[1] DFD Level – 1 gives more in and out information of the system.
[2] Where system gives detailed information of the procedure taking place.

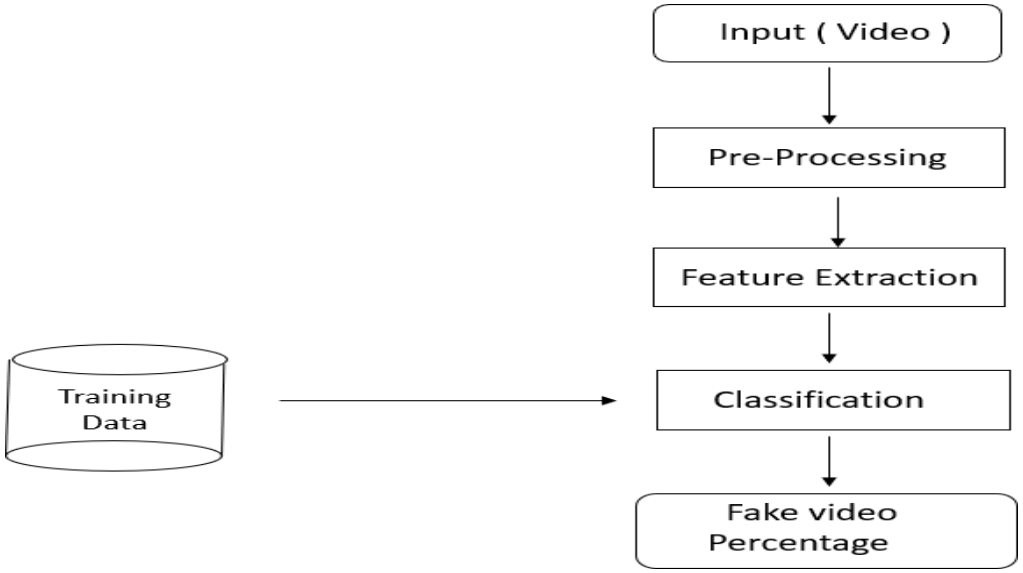


Figure 4.3: DFD Level 1

DFD Level-2

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

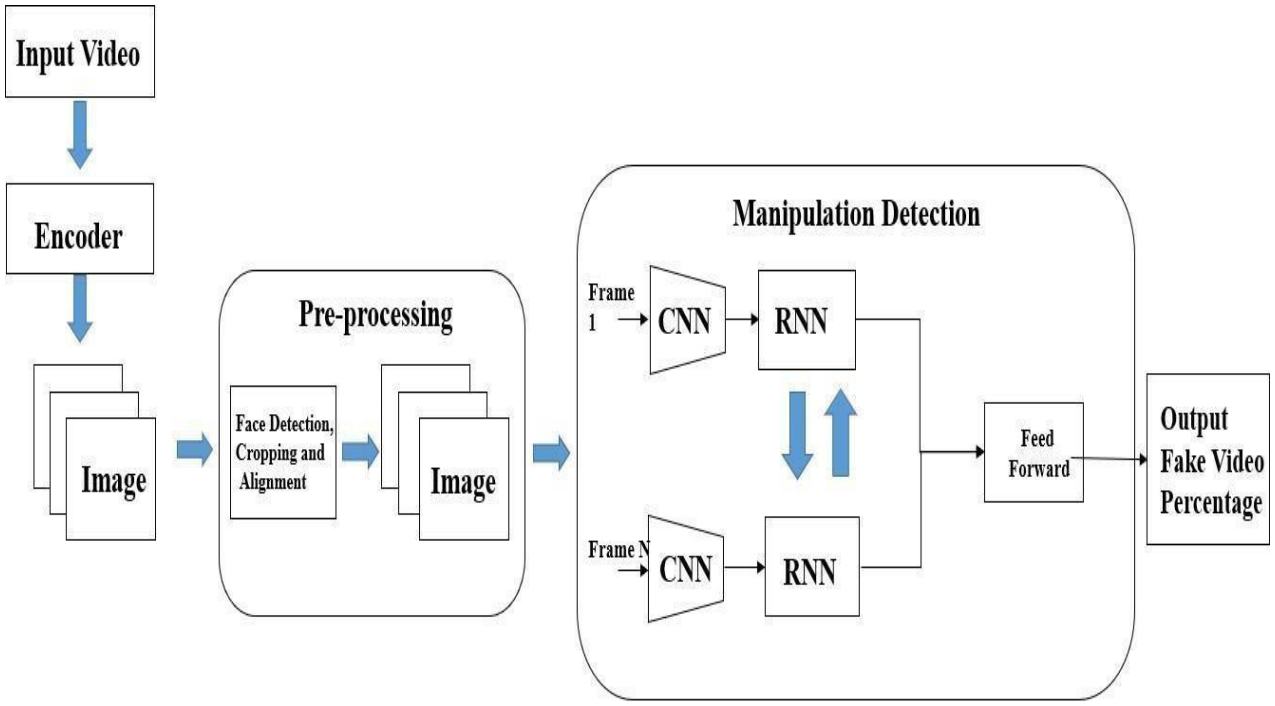


Figure 4.4: DFD Level 2

4.2.2 Activity Diagram:

Training Workflow:

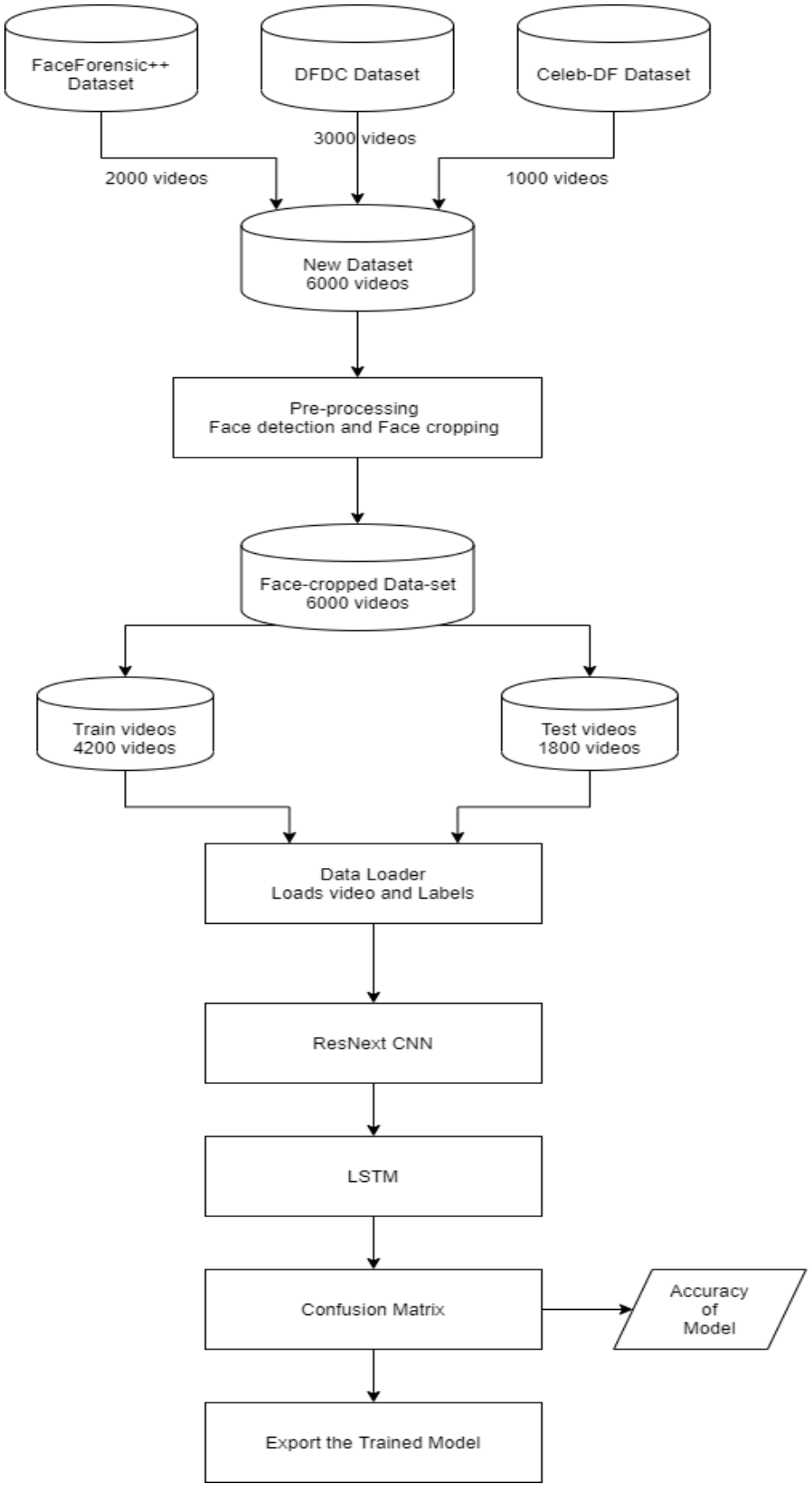


Figure 4.5: Training Workflow

Testing Workflow:

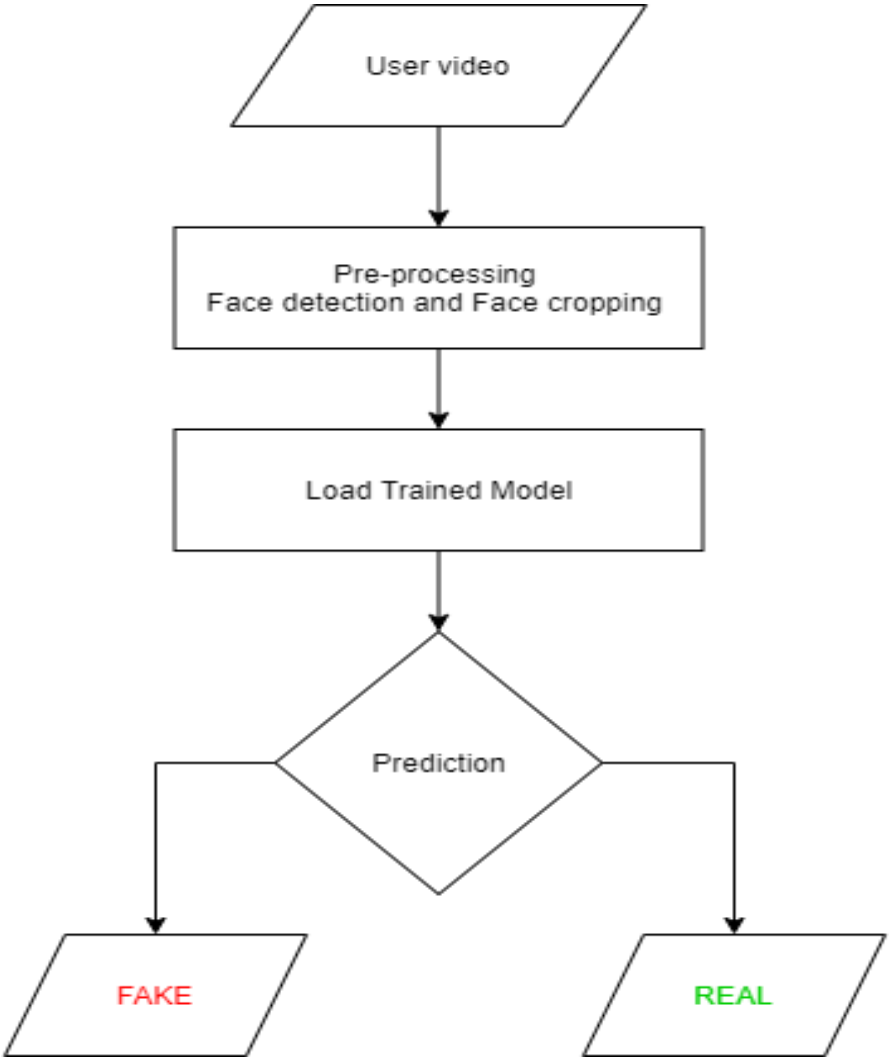


Figure 4.6: Testing Workflow

4.2.3 Non Functional Requirements:

Performance Requirement

- 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.

4.2.4 Sequence Diagram

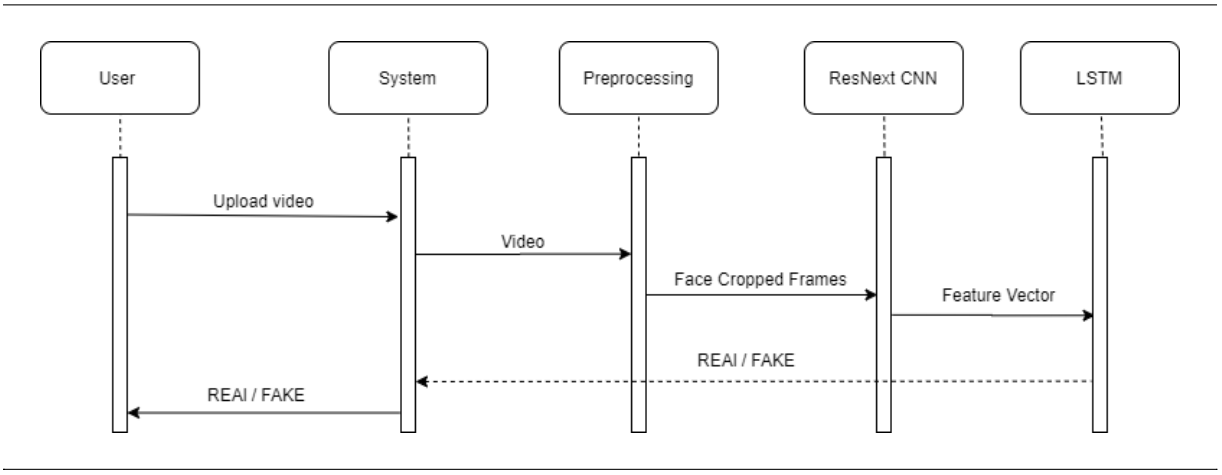


Figure 4 .7: Sequence Diagram

5. Detailed Design Document

5.1 Introduction

5.1.1 System Architecture

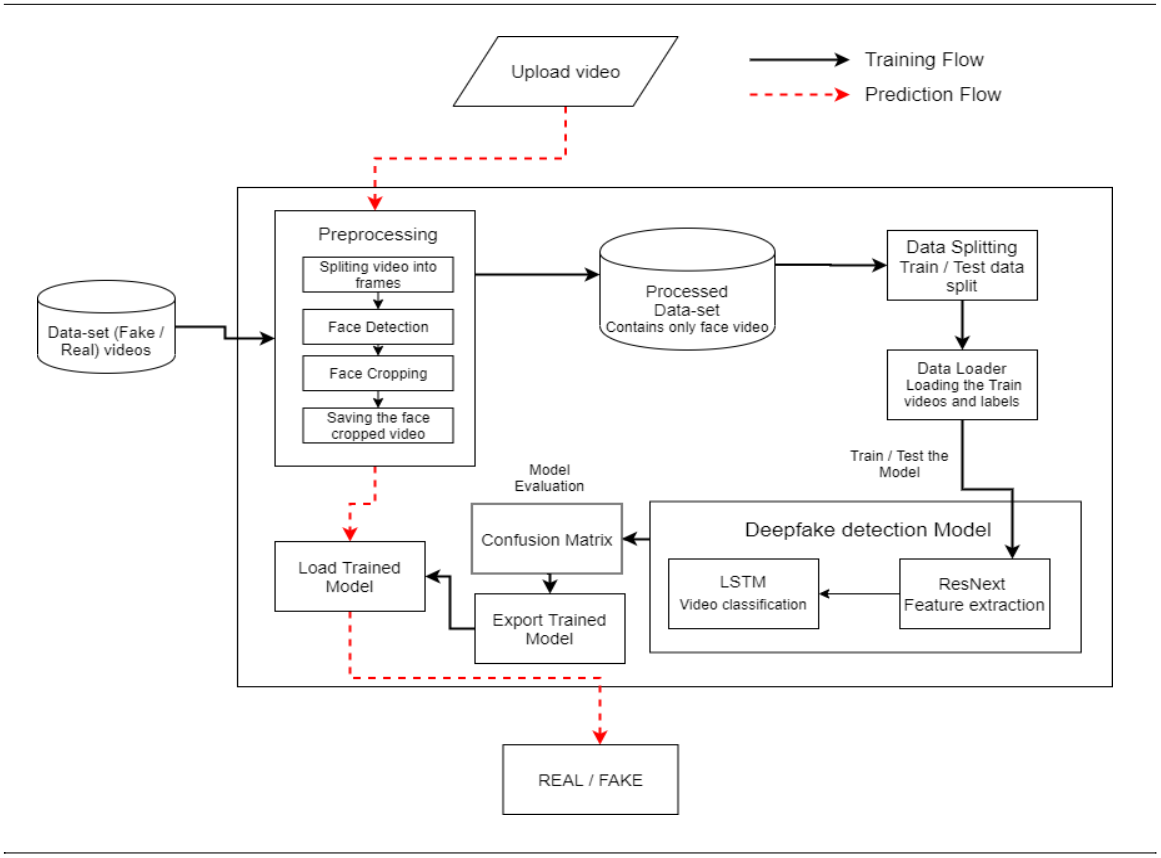


Figure 5 .1: System Architecture

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 have taken a dataset, preprocessed the dataset and created a new processed dataset which only includes the face cropped videos

.

5.1.1.1 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.

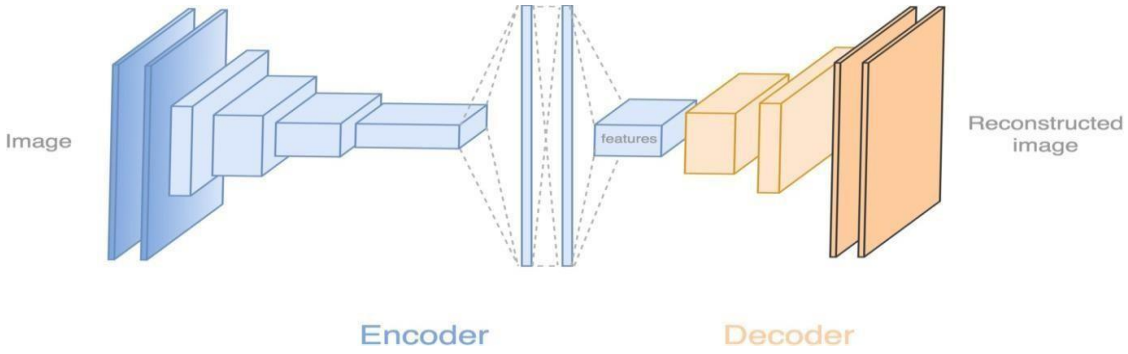


Figure 5 .2: Deepfake generation

Figure 5 .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

5.2 Architectural Design

5.2.1 Module 1 : Data-set Gathering

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

Deep fake detection challenge (DFDC) dataset [3] consist of certain audioalerted video, as audio deepfake are out of scope for this paper. We preprocessedthe DFDC dataset and removed the audio altered videos from the dataset by runninga 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 makes our total dataset consisting 3000 Real, 3000 fake videos and 6000 videos in total. Figure 2 depicts the distribution of the data-sets.

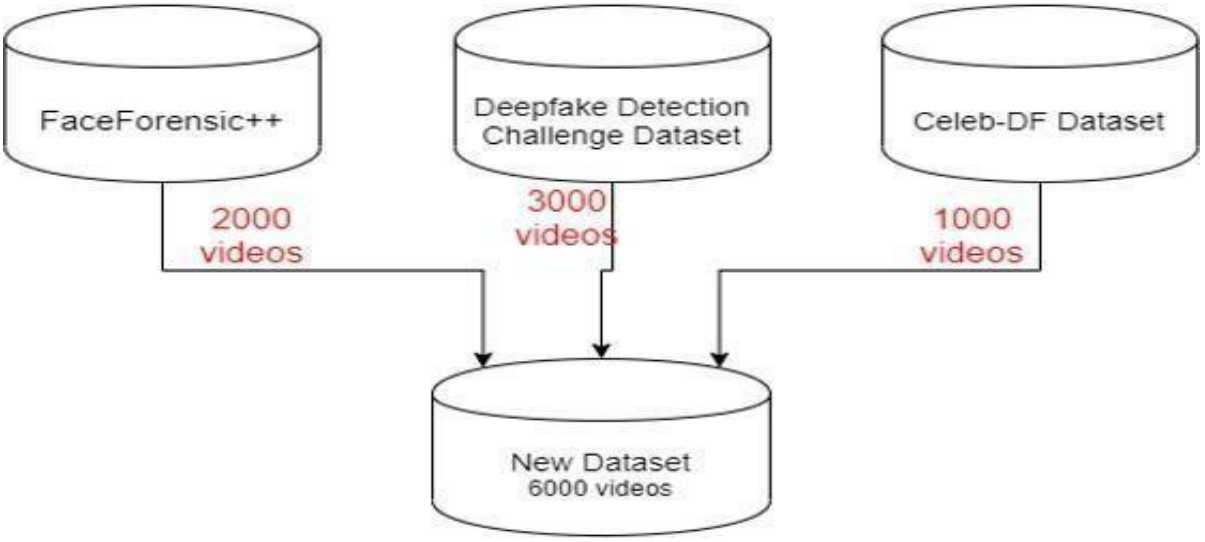


Figure 5 .4: Dataset

5.2.2 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 powerin 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 framesof 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.

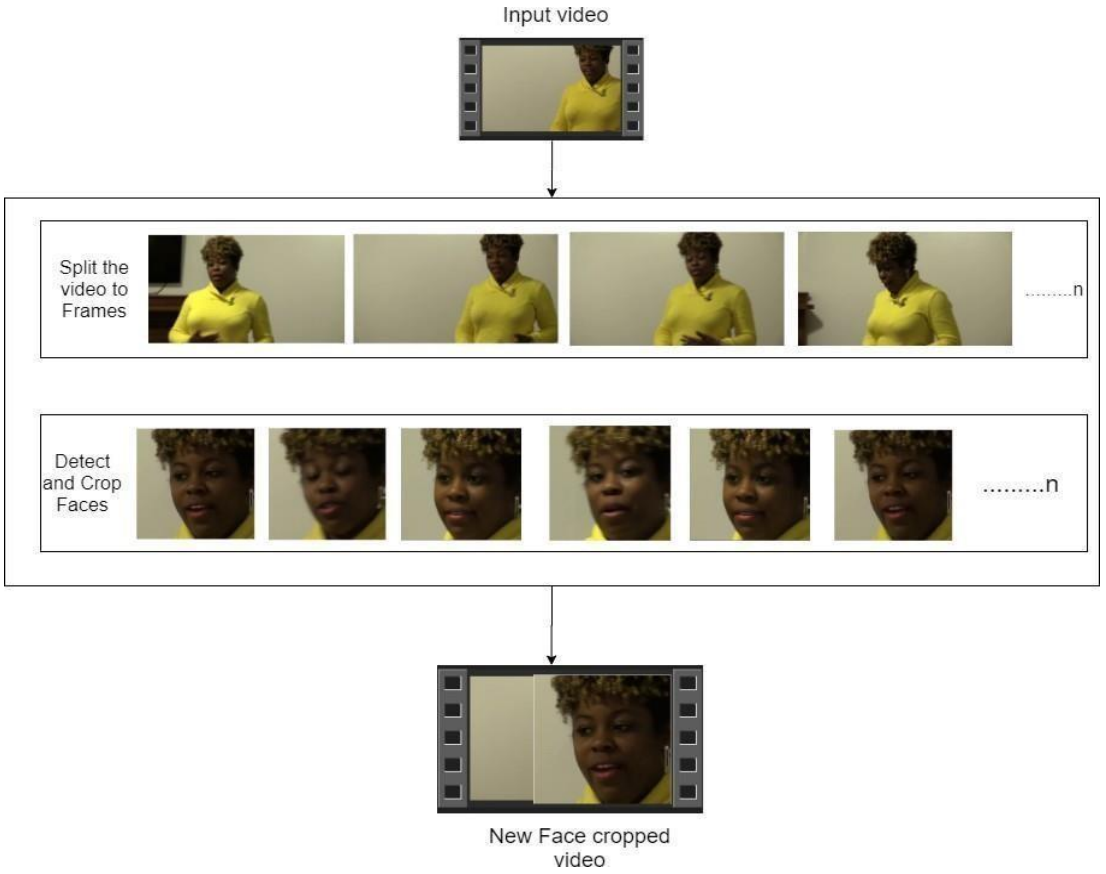


Figure 5 .5: Pre-processing of video

5.2.3 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.

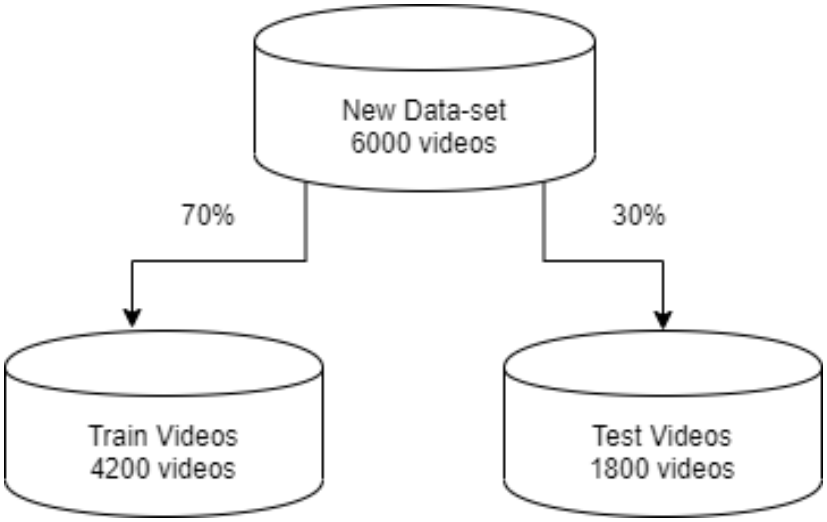


Figure 5 .6: Train test split

5.2.4 Module 4: Model Architecture

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

ResNext :

Instead of writing the code from scratch, we used the pre-trained model of ResNet for feature extraction. ResNet is Residual CNN network optimized for high performance on deeper neural networks. For the experimental purpose we have used resnext50_32x4d model. We have used a ResNet 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 ofthe model. The 2048-dimensional feature vectors after the last pooling layers of ResNet 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 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.

The model also consists of Leaky Reactivation 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 ofthe 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.

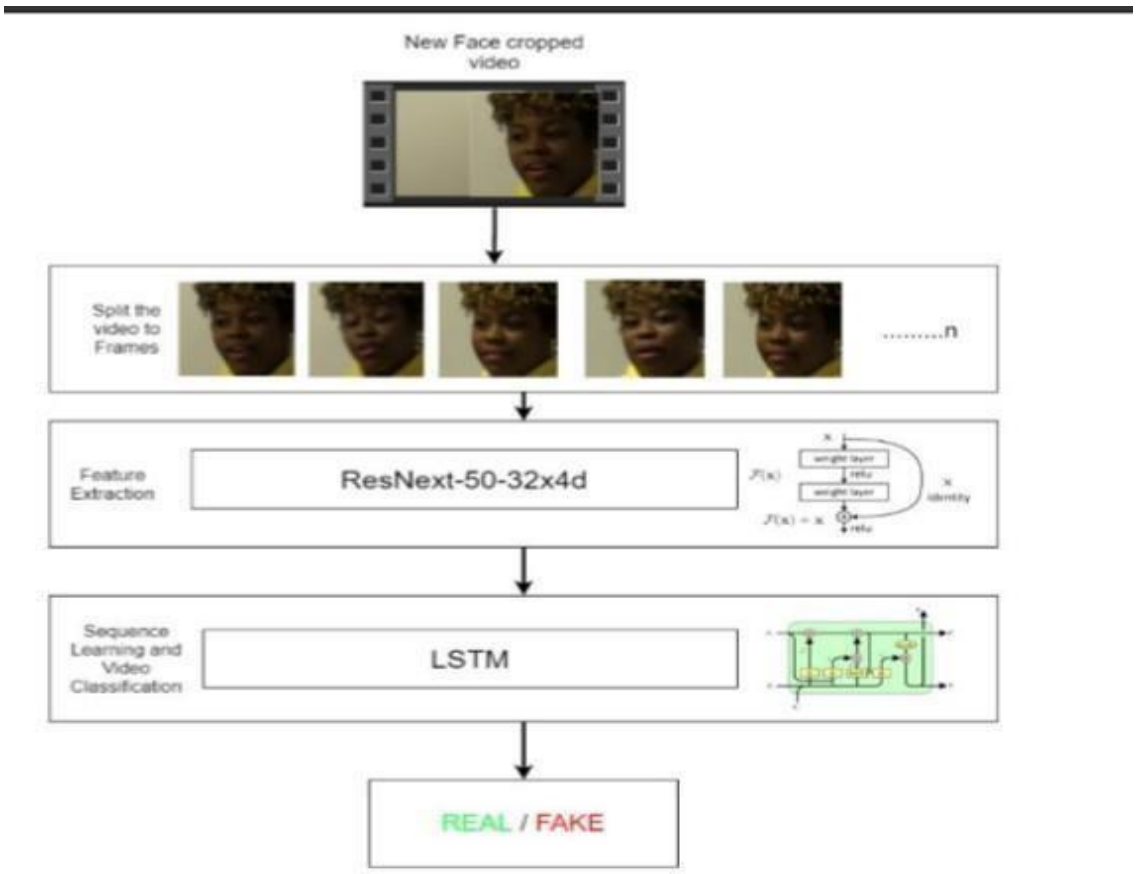


Figure 5 .7: Overview of our model

5.2.5 Module 5: Hyper-parameter tuning

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

achieve a better global minimum of gradient descent. The weight decay used is 1e-3.

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

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.

6. Project Implementation

6.1 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, Donald Trump's Breaking Bad series where he was introduces as James McGill, Barack Obama's public 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 theparadigm 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 celebrities and other famous personalities the target of pornographic material, fake surveillance videos, fake news 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.

6.2 Tools and Technologies Used

6.2.1 Planning

1. OpenProject

6.2.2 UML Tools

2. draw.io

6.2.3 Programming Languages

3. Python3
4. JavaScript

6.2.4 Programming Frameworks

5. PyTorch
6. Django

6.2.5 IDE

7. Google Colab
8. Jupyter Notebook
9. Visual Studio Code

6.2.6 Versioning Control

10. Git

6.2.7 Libraries

- backports.entry-points-selectable==1.1.0
- click==8.0.1
- cycler==0.10.0
- distlib==0.3.2
- dlib==19.22.1
- et-xmlfile==1.1.0
- face-recognition==1.3.0
- face-recognition-models==0.3.0
- filelock==3.0.12
- imageio==2.9.0
- itsdangerous==2.0.1
- joblib==1.0.1
- kiwisolver==1.3.2
- MarkupSafe==2.0.1
- matplotlib==3.4.3
- networkx==2.6.3
- nltk==3.6.6
- numpy==1.21.0
- opencv-python==4.5.3.56
- openpyxl==3.0.7
- pandas==1.3.0
- Pillow==9.0.0
- platformdirs==2.2.0
- pyparsing==2.4.7

- python-dateutil==2.8.2
- pytz==2021.1
- PyWavelets==1.1.1
- regex==2021.7.6
- scikit-image==0.18.3
- scikit-learn==0.24.2
- scipy==1.7.0
- six==1.16.0
- threadpoolctl==2.2.0
- tifffile==2021.8.30
- torch==1.9.0
- torchvision==0.10.0
- tqdm==4.61.2
- typing-extensions==3.10.0.2

6.3 Algorithm Details

6.3.1 Dataset Details

Refer 3.8

6.3.2 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 new video 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.

6.3.3 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 6 .1: ResNext Architecture

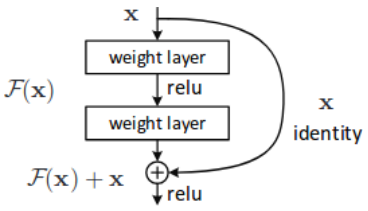


Figure 6 .2: ResNext Working

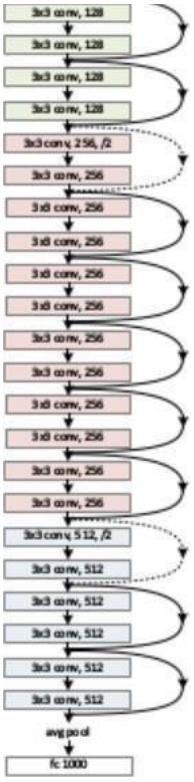


Figure 6 .3: Overview of ResNext Architecture

- 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 todo 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.

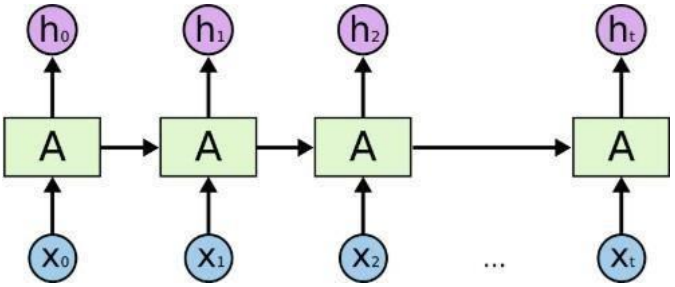


Figure 6 .4: Overview of LSTM Architecture

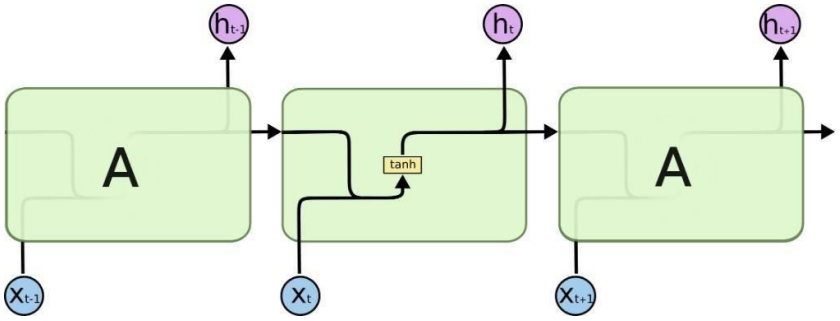


Figure 6 .5: Internal LSTM Architecture

- 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 isvery fast.

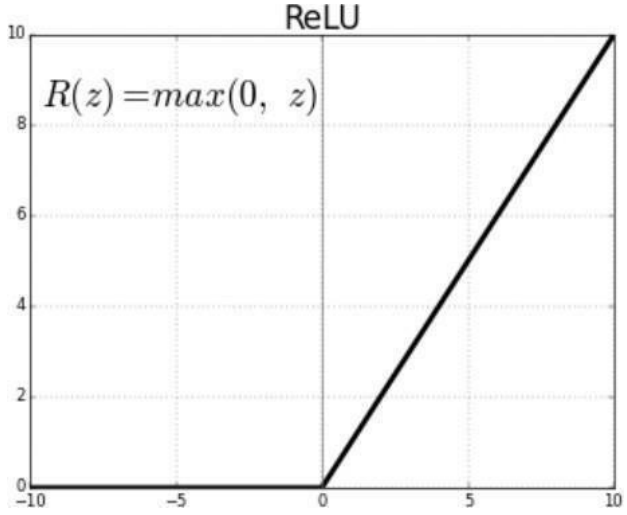


Figure 6 .6: RElU Activation function

- 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 neighboring neurons changing the way the weights will be updated during the process of backpropagation.

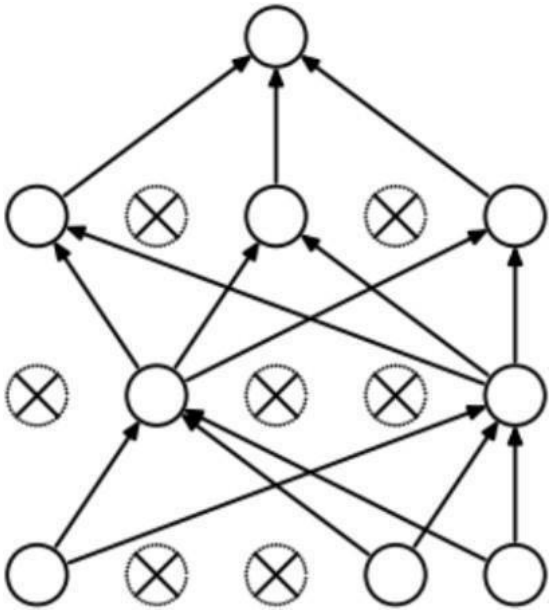


Figure 6 .7: Dropout layer overview

- Adaptive Average Pooling Layer : It is used To reduce variance, reduce computation complexity and extract low level features from neighbourhood.2 dimensional Adaptive Average Pooling Layer is used in the model.

6.3.4 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 figure 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. Squashing 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, softmax functions 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 Softmax layer provide us the confidence(probability) of prediction.

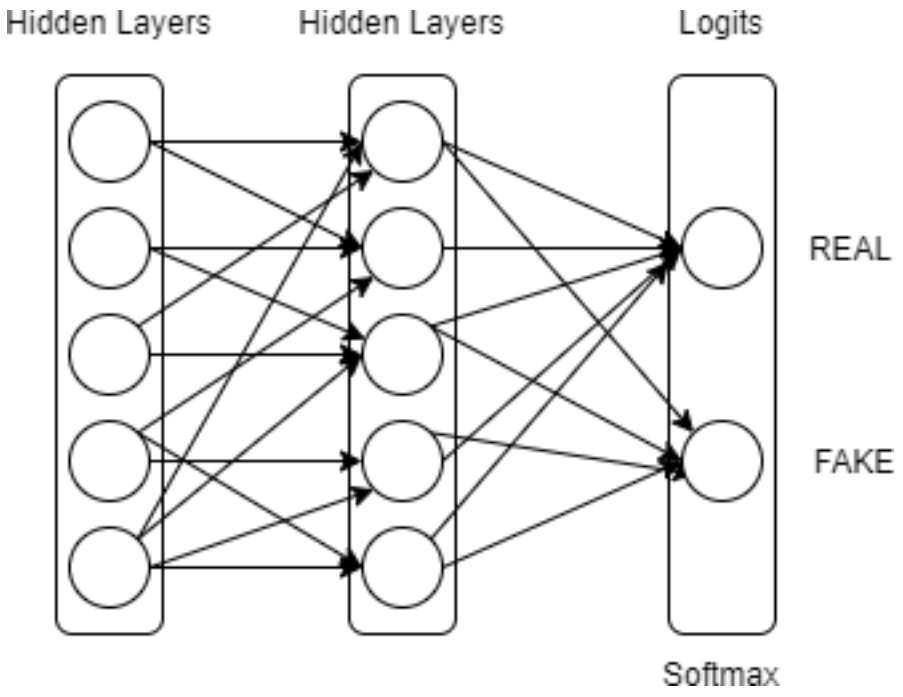


Figure 6 .8: Softmax Layer

- 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 your

classification 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.

6.3.5 Model Prediction Details

- The model is loaded in the application
- The new video for prediction is preprocessedand 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.

7. CODE IMPLEMENTATION

7.1 PYTHON CODE

```
from flask import Flask, render_template, redirect, request, url_for,
send_file
from flask import jsonify, json
from werkzeug.utils import secure_filename

# Interaction with the OS
import os
os.environ['KMP_DUPLICATE_LIB_OK']='True'

# Used for DL applications, computer vision related processes
import torch
import torchvision

# For image preprocessing
from torchvision import transforms

# Combines dataset & sampler to provide iterable over the dataset
from torch.utils.data import DataLoader
from torch.utils.data.dataset import Dataset

import numpy as np
import cv2

# To recognise face from extracted frames
import face_recognition

# Autograd: PyTorch package for differentiation of all operations on
Tensors
# Variable are wrappers around Tensors that allow easy automatic
differentiation
from torch.autograd import Variable

import time

import sys

# 'nn' Help us in creating & training of neural network
```

```
from torch import nn

# Contains definition for models for addressing different tasks i.e. image
classification, object detection e.t.c.
from torchvision import models

from skimage import img_as_ubyte
import warnings
warnings.filterwarnings("ignore")

UPLOAD_FOLDER = 'Uploaded_Files'
video_path = ""

detectOutput = []

app = Flask("__main__", template_folder="templates")
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

# Creating Model Architecture

class Model(nn.Module):
  def __init__(self, num_classes, latent_dim= 2048, lstm_layers=1,
hidden_dim=2048, bidirectional=False):
    super(Model, self).__init__()

    # returns a model pretrained on ImageNet dataset
    model = models.resnext50_32x4d(pretrained= True)

    # Sequential allows us to compose modules nn together
    self.model = nn.Sequential(*list(model.children())[:-2])

    # RNN to an input sequence
    self.lstm = nn.LSTM(latent_dim, hidden_dim, lstm_layers,
bidirectional)

    # Activation function
    self.relu = nn.LeakyReLU()

    # Dropping out units (hidden & visible) from NN, to avoid overfitting
    self.dp = nn.Dropout(0.4)

    # A module that creates single layer feed forward network with n
inputs and m outputs
```

```
    self.linear1 = nn.Linear(2048, num_classes)

    # Applies 2D average adaptive pooling over an input signal composed
of several input planes
    self.avgpool = nn.AdaptiveAvgPool2d(1)



  def forward(self, x):
    batch_size, seq_length, c, h, w = x.shape

    # new view of array with same data
    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,:]))




im_size = 112

# std is used in conjunction with mean to summarize continuous data
mean = [0.485, 0.456, 0.406]

# provides the measure of dispersion of image grey level intensities
std = [0.229, 0.224, 0.225]

# Often used as the last layer of a nn to produce the final output
sm = nn.Softmax()

# Normalising our dataset using mean and std
inv_normalize = transforms.Normalize(mean=-1*np.divide(mean, std),
std=np.divide([1,1,1], std))

# For image manipulation
def im_convert(tensor):
  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)
  cv2.imwrite('./2.png', image*255)
  return image

# For prediction of output
def predict(model, img, path='./'):
  # use this command for gpu
  # fmap, logits = model(img.to('cuda'))
  fmap, logits = model(img.to())
  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]


# To validate the dataset
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

  # To get number of videos
  def__len__(self):
    return len(self.video_names)

  # To get number of frames
  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)):
      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
        frames = torch.stack(frames)
        frames = frames[:self.count]
        return frames.unsqueeze(0)

      # To extract number of frames
      def frame_extract(self, path):
        vidObj = cv2.VideoCapture(path)
        success = 1
        while success:
          success, image = vidObj.read()
          if success:
            yield image


def detectFakeVideo(videoPath):
    im_size = 112
    mean = [0.485, 0.456, 0.406]
    std = [0.229, 0.224, 0.225]

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

    video_dataset = validation_dataset(path_to_videos,sequence_length =
20,transform = train_transforms)
    # use this command for gpu
    # model = Model(2).cuda()
    model = Model(2)
    path_to_model = 'model/df_model.pt'
    model.load_state_dict(torch.load(path_to_model,
map_location=torch.device('cpu')))
    model.eval()
    for i in range(0,len(path_to_videos)):
        print(path_to_videos[i])
```

```
        prediction = predict(model,video_dataset[i],'./')
        if prediction[0] == 1:
            print("REAL")
        else:
            print("FAKE")
    return prediction

@app.route('/', methods=['POST', 'GET'])
def homepage():
  if request.method == 'GET':
            return render_template('index.html')
  return render_template('index.html')

@app.route('/Detect', methods=['POST', 'GET'])
def DetectPage():
    if request.method == 'GET':
        return render_template('index.html')
    if request.method == 'POST':
        video = request.files['video']
        print(video.filename)
        video_filename = secure_filename(video.filename)
        video.save(os.path.join(app.config['UPLOAD_FOLDER'],
video_filename))
        video_path = "Uploaded_Files/" + video_filename
        prediction = detectFakeVideo(video_path)
        print(prediction)
        if prediction[0] == 0:
            output = "FAKE"
        else:
            output = "REAL"
        confidence = prediction[1]
        data = {'output': output, 'confidence': confidence}
        data = json.dumps(data)
        os.remove(video_path);
        return render_template('index.html', data=data)
        app.run(port=3000);
```

STATIC/REACT –CSS AND JS :

Entire code is given in below link
https://drive.google.com/drive/folders/1gW2704rXHfHO4iw4fBLOufm-8nWLYAHV?usp=sharing

8. Software Testing

8.1 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
4. Compliance Testing

8.2 Test Cases and Test Results

Test Cases

Table 9.1: Test Case Report

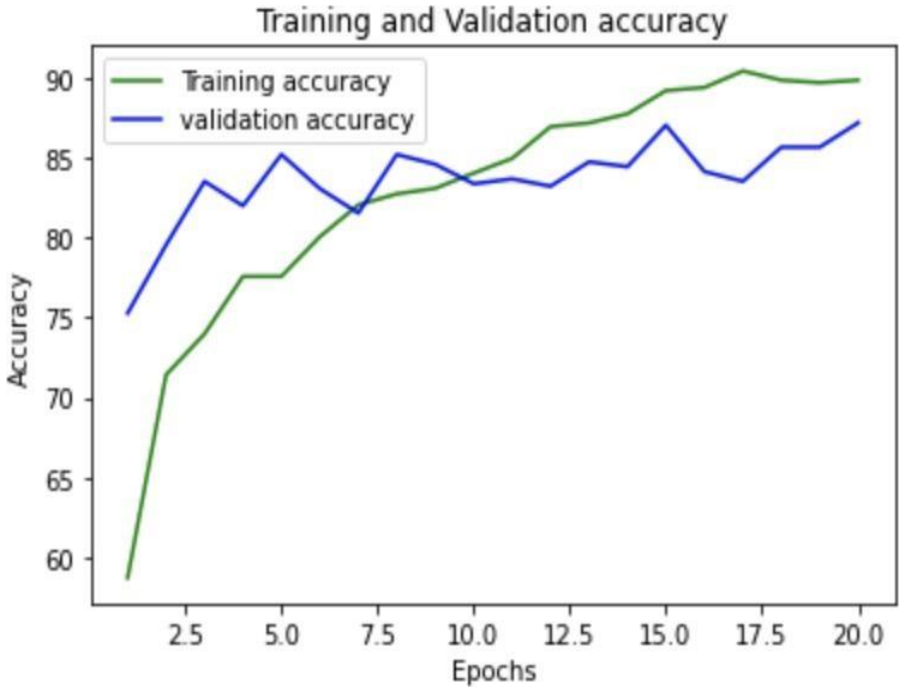
Case id	Test Case Description	Expected Result	Actual Result	Status
1	Upload a word file instead of video	Error message: Only video files allowed	Error message: Only video files allowed	Pass
2	Upload a 200MB video file	Error message: Max limit 100MB	Error message: Max limit 100MB	Pass
3	Upload a file 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

9. Results and Discussion

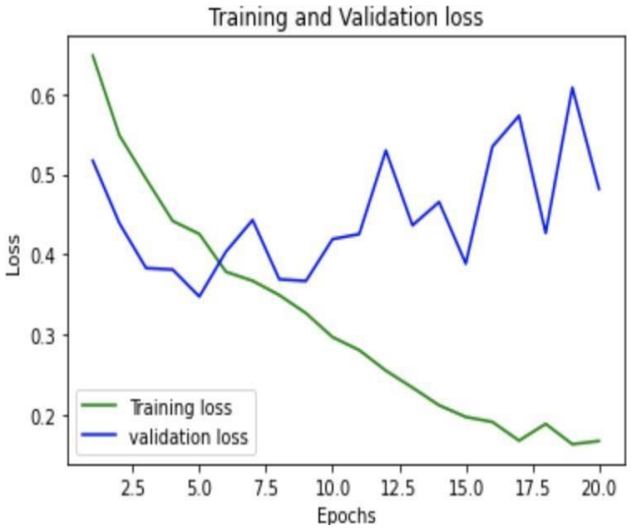
9.1 Our Results

1. Accuracy of the Model:

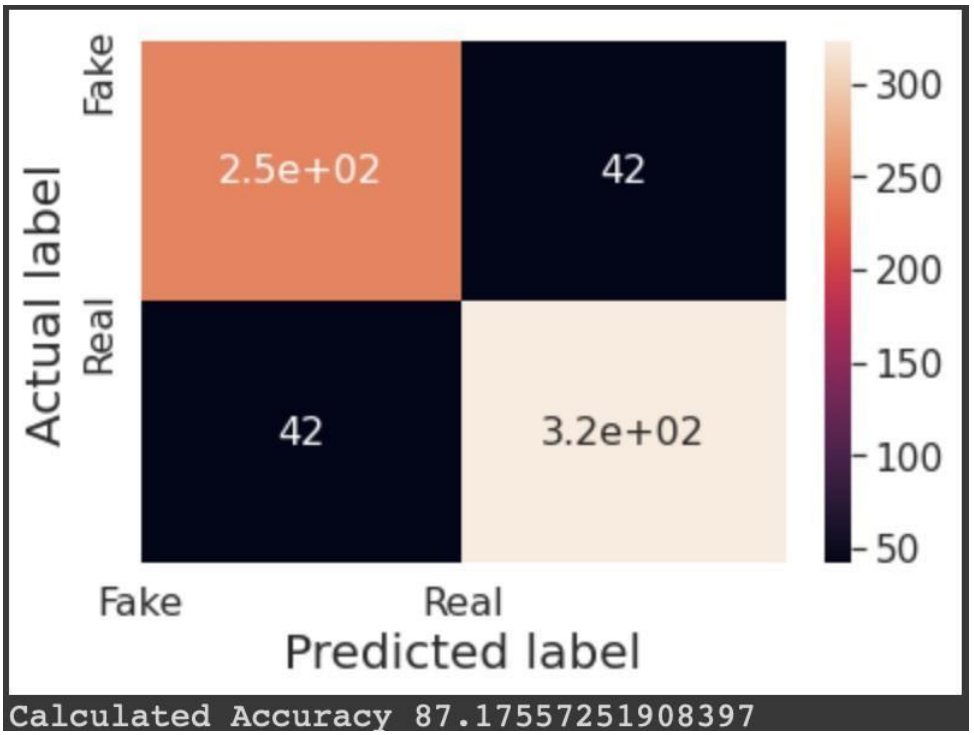
2. Training and Validation accuracy



3. Training and Validation Loss Graph:



4. Confusion Matrix:



9.2 Screenshots

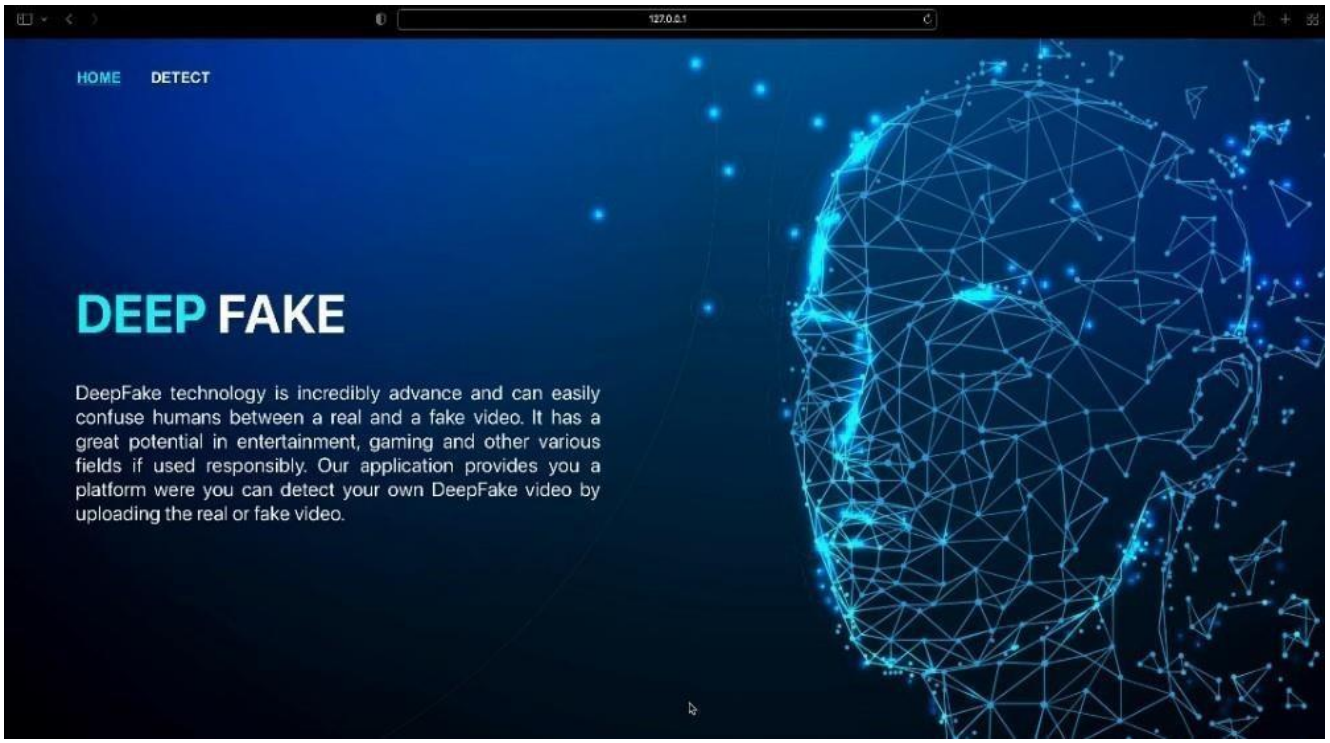


Fig 9.1 Home page

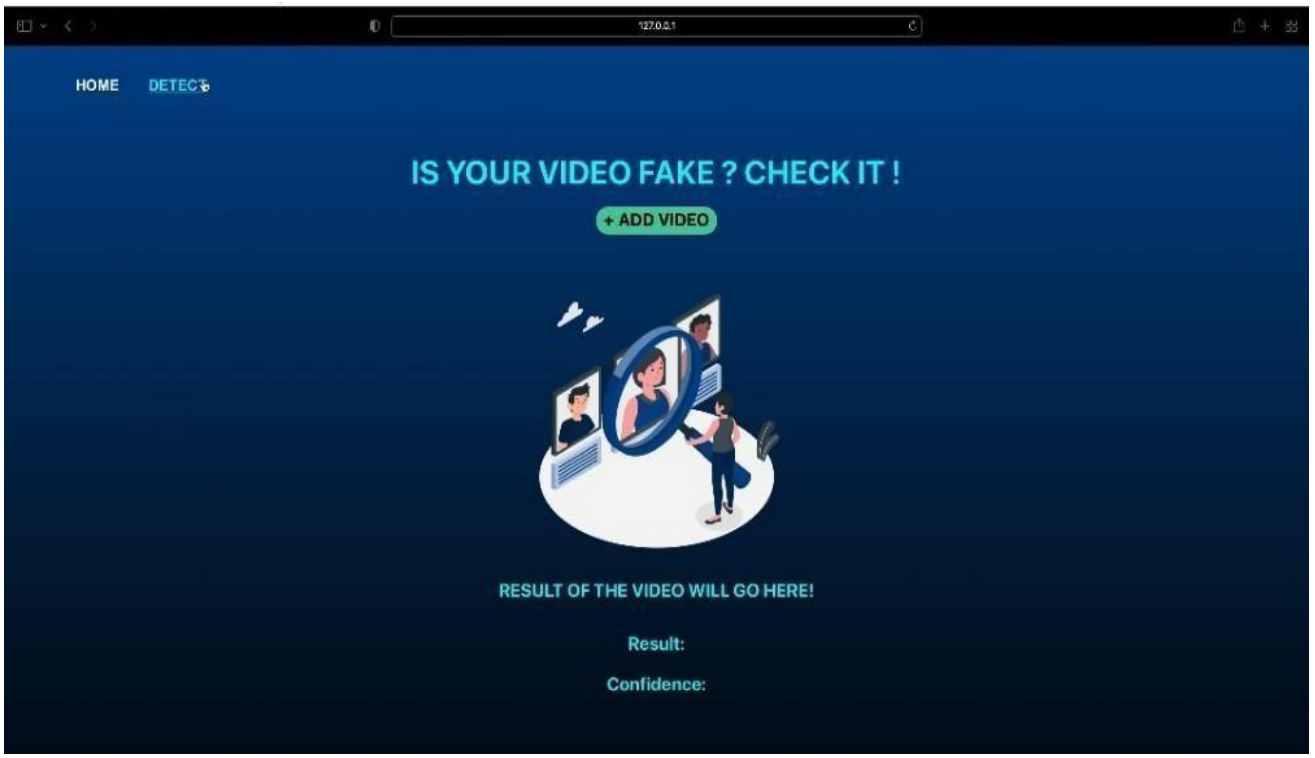


Fig 9.2 Detect: add Video

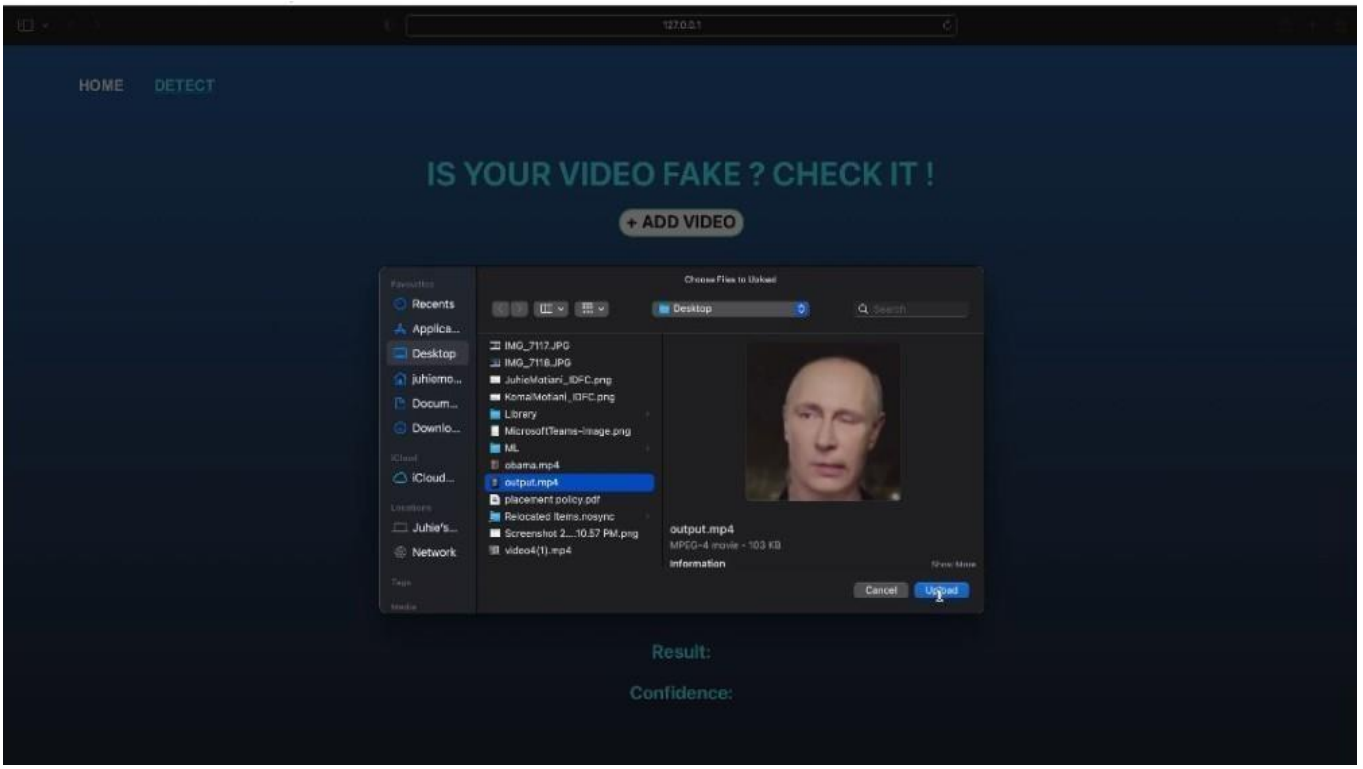


Fig 9.3 Detect: Upload Video

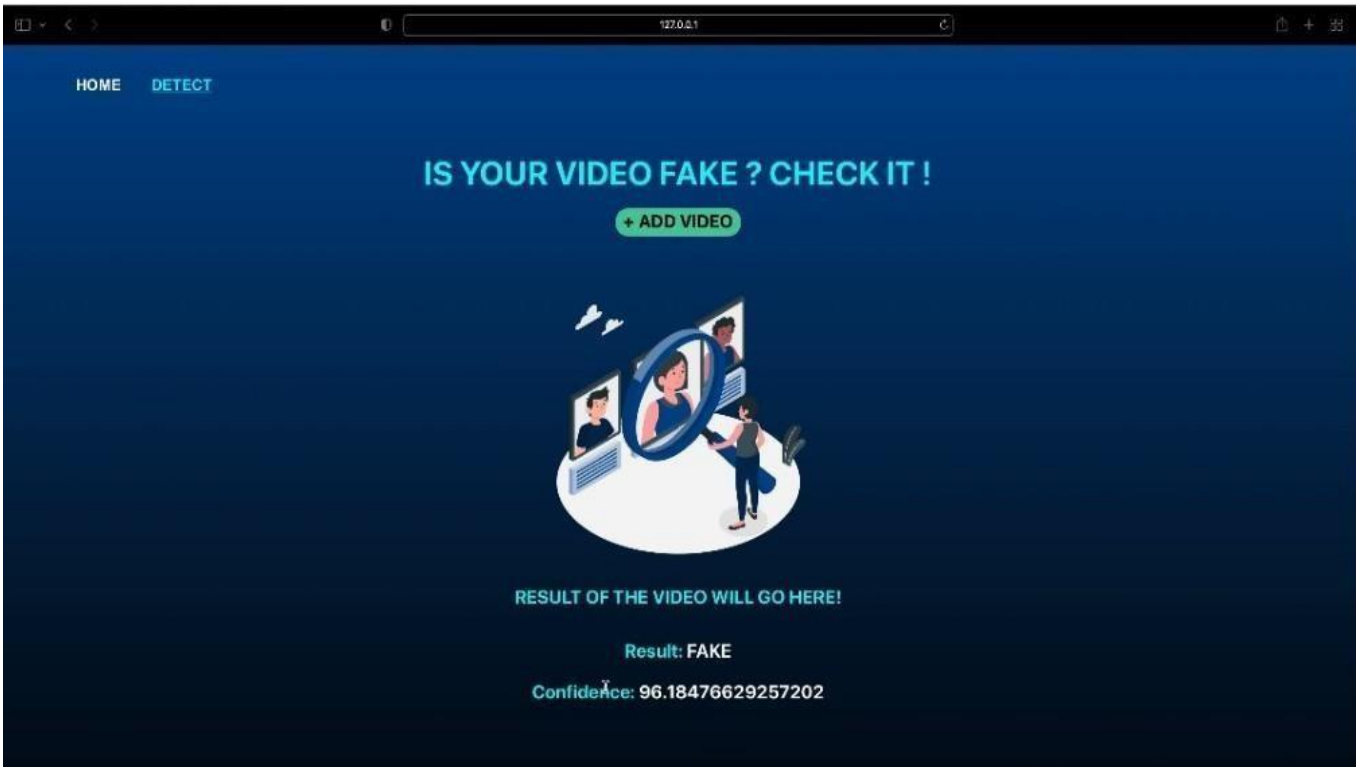


Fig 9.4 Detect: Result of Video

10. Conclusion and Future work

10.1 Conclusion:

In conclusion, our project has successfully addressed the challenge of detecting deepfake videos through the implementation of advanced deep learning techniques. We meticulously analyzed the problem statement, ensuring the feasibility of our proposed solution. By employing balanced training methodologies and rigorous evaluation techniques, we have developed robust deepfake detection models. The integration of our models into a user-friendly web-based application enhances accessibility, allowing users to easily verify the authenticity of videos. Furthermore, our comprehensive analysis of hardware and software requirements ensures the scalability and efficiency of our solution.

.

10.2 Future Work:

Enhanced Accessibility: Hosting the system on a dedicated domain name would make it easily accessible to users, improving usability and reach.

Enhanced Model Performance: Continuously refining and fine-tuning our deepfake detection models can lead to improved accuracy and reliability.

Real-time Detection: Investigating methods to enable real-time deepfake detection, potentially through optimization techniques and leveraging parallel processing capabilities.

Dataset Expansion: Expanding our dataset to encompass a broader range of deepfake variations and scenarios can further enhance the robustness and generalization of our models.

User Interface Enhancements: Iteratively improving the user interface of our web-based application to enhance usability and user experience, potentially incorporating additional features and functionalities.

11. References

[1] Andreas Rossler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies,Matthias Nießner, “FaceForensics++: Learning to Detect Manipulated Facial Images” in arXiv:1901.08971.

[2] Deepfake detection challenge dataset : https://www.kaggle.com/c/deepfake-detection-challenge/data Accessed on 26 March, 2020

[3] 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

[4] Deepfake Video of Mark Zuckerberg Goes Viral on Eve of House A.I. Hearing : https://fortune.com/2019/06/12/deepfake-mark-zuckerberg/ Accessed on 26 March, 2020

[5] 10 deepfake examples that terrified and amused the internet : https://www.creativebloq.com/features/deepfake-examples Accessed on 26 March, 2020

[6] TensorFlow: https://www.tensorflow.org/ (Accessed on 26 March, 2020)

[7] Keras: https://keras.io/ (Accessed on 26 March, 2020)

[8] PyTorch : https://pytorch.org/ (Accessed on 26 March, 2020)

[9] G. Antipov, M. Baccouche, and J.-L. Dugelay. Face aging with conditional generative adversarial networks. arXiv:1702.01983, Feb. 2017

[10] 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.

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

[12] Face Swap : https://faceswaponline.com/ (Accessed on 26 March, 2020)

[13] Deepfakes, Revenge Porn, And The Impact On Women : https://www.forbes.com/sites/chenxiwang/2019/11/01/deepfakes-revenge-porn-and-the-impact-on-women/

[14] The rise of the deepfake and the threat to democracy

https://www.theguardian.com/technology/ng-interactive/2019/jun/22/the-rise-of-the-deepfake-and-the-threat-to-democracy(Accessed on 26 March, 2020)

[15] Yuezun Li, Siwei Lyu, “ExposingDF Videos By Detecting Face Warping Arti- facts,” in arXiv:1811.00656v3.

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

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

[18] 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.

[19] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic hu- man actions from movies. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1–8, June 2008. Anchorage, AK

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

[21] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization.arXiv:1412.6980, Dec. 2014.

[22] ResNext Model : https://pytorch.org/hub/pytorch_vision_resnext/ accessed on06 April 2020

[23] https://www.geeksforgeeks.org/software-engineering-cocomo-model/ Accessedon 15 April 2020

[24] International Journal for Scientific Research and Development http://ijsrd.com/