A Final Project Report On

DeepFake Detection System

Submitted in the Partial Fulfillment of the
Requirements for the Degree of Bachelor of Computer Engineering Awarded
by Pokhara University

Submitted By:

Aakash Poudel (20070047)

Bikash G.C (20070057)

Nishan Baniya (20070073)

Safal Timsina (20070083)

Sangam Khadka (20070084)

Supervisor:

Er. Santosh Paudel

Assistant Professor

School of Engineering

Faculty of Science and Technology

POKHARA UNIVERSITY

August 2024

A Final Project Report On

DeepFake Detection System

Submitted in the Partial Fulfillment of the
Requirements for the Degree of Bachelor of Computer Engineering Awarded
by Pokhara University

Submitted By:

Aakash Poudel (20070047)

Bikash G.C (20070057)

Nishan Baniya (20070073)

Safal Timsina (20070083)

Sangam Khadka (20070084)

Supervisor:

Er. Santosh Paudel

Assistant Professor

School of Engineering

Faculty of Science and Technology

POKHARA UNIVERSITY

August 2023

COPYRIGHT

STUDENT'S DECLARATION

We hereby declare that this project work entitled **DeepFake Detection System** is based on our original work. All concepts, data, code, and any other work from external sources have been properly cited and referenced in accordance with the guidelines provided by School of Engineering, Pokhara University

We owe all the liabilities relating to the authenticity and originality of this project work and project report.

1. Aakash Poudel (20070047) ……..
2. Bikash GC (20070057) ……..
3. Nishan Baniya (20070073) ……..
4. Safal Timsina (20070083) ……..
5. Sangam Khadka(20070084) ……..

Date: ……

SUPERVISOR'S RECOMMENDATION

This is to certify that this project report entitled **DeepFake Detection System** prepared and submitted by below listed team of students in partial fulfillment of the requirements of the degree of Bachelor of Computer Engineering / Bachelor of Software Engineering awarded by Pokhara University, has been prepared and completed under my supervision.

I hereby recommend the same for acceptance by School of Engineering, Pokhara University.

1.Aakash Poudel (20070047) ..……

2 .Bikash G.C (20070057) …….

3.Nishan Baniya (20070073) ...……..

4. Safal Timsina(20070083) ..……

5.Sangam Khadka (20070084) ……..

……..

Er. Santosh Paudel
Assistant Proferssor
School of Engineering
Pokhara University

Date: …….

School of Engineering

POKHARA UNIVERSITY

EXTERNAL EXAMINER'S RECOMMENDATION

The undersigned certified that they have evaluated this project report entitled **DeepFake Detection System** submitted by **Aakash Poudel, Bikash G.C, Nishan Baniya, Safal Timsina, Sangam Khadka** and their oral presentation for partial fulfillment of the degree of Bachelor of Computer Engineering/ Bachelor of Software Engineering and recommended to the School of Engineering, Pokhara University for acceptance of this project work/report.

…….

Er. Santosh Panth

Assistant Professor

Gandaki College of Engineering & Science

Lamachaur, Pokhara, Kaski

Date: …….

School of Engineering

POKHARA UNIVERSITY

LETTER OF APPROVAL

This project report entitled Deepfake Detection System submitted by **Aakash Poudel, Bikash G.C, Nishan Baniya, Safal Timsina, Sangam Khadka** for partial fulfillment of the degree of Bachelor of Computer Engineering/ Bachelor of Software Engineering has been accepted by the School of Engineering, Pokhara University upon the recommendations of Supervisor and with the approval by the following examiner.

Er. Santosh Panth

Assistant Professor

Gandaki College of Engineering & Science

Lamachaur, Pokhara, Kaski

Er. Bhesh Bahadur Thapa

Assistant Professor

(Coordinator)

Bachelor of Computer Engineering

Bachelor of Software Engineerin

School of Engineering

Pokhara University

Date: …….

ABSTRACT

The growing computation power has made the deep learning algorithms so powerful that creating a indistinguishable human synthesized video popularly called as deep fakes have became very simple. Scenarios where these realistic face swapped deep fakes are used to create political distress, fake terrorism events, revenge porn, blackmail peoples are easily envisioned. 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*

ACKNOWLEDGEMENT

Firstly, we would like to extend our profound gratitude to our supervisor, Er. Santosh Paudel. His mentorship, and expertise were instrumental in shaping our understanding and approach to problem-solving. Despite our status as beginners, he skillfully directed our vision, helping us break down complex problems and apply systematic analysis. His dedication to our success has been exceptional, and we are deeply thankful for his invaluable contributions to this project. We express our sincere gratitude to the program coordinator, Dr. Udaya Raj Dhungana whose visionary idea laid the foundation for this project. Dr. Dhungana's invaluable feedback and insights played a pivotal role in elevating the quality of our work. We wish to convey our deep appreciation to our families for standing by us throughout the demanding phases of the project. Their unwavering support, understanding, and invaluable patience were a source of great strength and encouragement.

TABLE OF CONTENTS

Content

LIST OF FIGURES

LIST OF TABLE

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
ANN	Artificial Neural Network
CNN	Convolution Neural Network
DP	Deep Learning
FCN	Fully Connected layer
ReLU	Rectified Linear Network
SVM	Support Vector Machine

CHAPTER 1

INTRODUCTION

1.1Background

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 pristine 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. 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 combination 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.

1.2 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 learning have led to a dramatic increase in the realism of fake content and the accessibility in which it can be created. These so-called AI-synthesized media (popularly referred to as deep fakes). Creating the Deep Fakes using the Artificially intelligent tools are simple task. But, when it comes to detection of these Deep Fakes, it is major challenge. Already in the history there are many examples where the deepfakes are used as powerful way to create political tension[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.

1.3 Objectives

Here are some specific objectives of deep fake detection system:

- To prevent the spread of misinformation, protect individual privacy, and maintain the integrity of digital communications.

1.4 Scope and Limitations

1.4.1 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 implementation detail.

1.4.2 Limitations

The deepfake detection project, while innovative and effective, has several limitations. Firstly, the model's accuracy is highly dependent on the quality and diversity of the training data; any bias or lack of variation in the datasets could lead to inaccurate results, particularly in real-world scenarios where deepfakes vary significantly. Secondly, the computational resources required for processing and training are substantial, potentially limiting the system's scalability and accessibility. Additionally, the model may struggle with detecting deepfakes generated by more advanced or newer algorithms that produce fewer artifacts. Furthermore, the reliance on cloud platforms like Google Cloud may introduce concerns related to data privacy and latency. Finally, the project primarily focuses on video content, leaving out potential deepfakes in other formats, such as audio or text, and its effectiveness might diminish as deepfake technology continues to evolve.

1.5 Significance

Deepfake detection technology is a crucial tool in our digital age. It helps combat misinformation and social manipulation by identifying fabricated videos that can be used to spread false information or harm individuals. By protecting individuals and organizations from malicious attacks, deepfake detection enhances trust and transparency online. It also supports law enforcement in identifying and dismissing fabricated evidence, ensuring accurate investigations. While deepfakes pose challenges to freedom of expression, deepfake detection helps maintain a balance by allowing for creative expression while mitigating potential harm.

1.6 Contribution

This project significantly contributes to the advancement of deepfake detection technology. By employing a sophisticated model architecture that combines ResNext CNN and LSTM RNN, it explores a promising approach for accurately identifying

fabricated videos. The emphasis on training the model with a large-scale, balanced dataset, including real-time data, enhances its ability to generalize and perform effectively in real-world scenarios. Furthermore, the development of a user-friendly web-based platform promotes wider adoption of deepfake detection technology among the public. The potential for scaling the solution to browser plugins or integration within social media platforms further expands its reach and impact. Ultimately, this project helps to combat the spread of misinformation and protect individuals and organizations from the harmful effects of deepfakes.

1.7 Key Features

- **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 simplified 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.

1.8 Report Organization

This project report is structured into several chapters, each focusing on distinct aspects of the "Deepfake Detection system" web application development and implementation. The organization of the report is designed to offer a thorough understanding of the project's background, methodologies, and outcomes.

In the first chapter, we establish the foundation by discussing the background of the project, outlining the problem statement, setting clear objectives, and defining the scope. This chapter also highlights the significance of our work, contributions made, and key features of the "Deepfake Detection System" . Finally, the chapter concludes with an overview of the report's structure.

Chapter 2 provides a comprehensive literature review, presenting a background study on related legal technology and existing systems. This chapter delves into various challenges and tasks related to the development of legal assistant applications, offering a solid theoretical framework that underpins our work.

Chapter 3 is dedicated to system analysis and design, where we thoroughly analyze the system requirements, functionalities, and design considerations. This chapter includes detailed system architecture, interface design, and various diagrams such as use case, activity, sequence, and data flow diagrams. We also elaborate on the development methodology used

In Chapter 4, we describe the implementation and testing phases, covering the tools and technologies used, data collection processes, and testing strategies employed to ensure the application's effectiveness. This chapter also provides a detailed analysis of the test results and discusses the challenges encountered during implementation.

Finally, Chapter 5 presents the conclusions drawn from this project, summarizing the

key findings and reflecting on the limitations of our work. This chapter also offers recommendations for future enhancements aiming to improve its functionality and user experience.

CHAPTER 2

LITERATURE REVIEW

2.1 Background Study

Face Warping Artifacts [15] used the approach to detect artifacts by comparing the generated face areas and their surrounding regions with a dedicated Convolutional Neural Network model. In this work there were two-fold of Face Artifacts.

Their method is based on the observations that current deepfake algorithm can only generate images of limited resolutions, which are then needed to be further transformed to match the faces to be replaced in the source video. Their method has not considered the temporal analysis of the frames.

Detection by Eye Blinking [16] describes a new method for detecting the deepfakes by the eye blinking as a crucial parameter leading to classification of the videos as deepfake or pristine. The Long-term Recurrent Convolution Network (LRCN) was used for temporal analysis of the cropped frames of eye blinking. As today the deepfake generation algorithms have become so powerful that lack of eye blinking cannot be the only clue for detection of the deepfakes. There must be certain other parameters must be considered for the detection of deepfakes like teeth enchantment, wrinkles on faces, wrong placement of eyebrows etc.

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 approach of using RNN for sequential processing of the frames along with ImageNet pretrained 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 biological signals from facial regions on pristine and deepfake portrait video pairs. Applied transformations to compute the spatial coherence and temporal consistency, capture the signal characteristics in feature vector and photoplethysmography (PPG) maps, and further train a probabilistic Support Vector Machine (SVM) and a Convolutional Neural Network (CNN). Then, the average of authenticity probabilities is used to classify whether the video is a 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 3

METHODOLOGY

3.1 Requirement Analysis

3.1.1 Problem Domain Understanding

The problem domain of the project centers around the rapid advancement and proliferation of deepfake technology, which creates highly realistic but fabricated videos using deep learning techniques. As deepfakes become increasingly sophisticated, they pose significant threats to information integrity, with potential uses ranging from political manipulation to personal blackmail. The core challenge lies in distinguishing these artificially generated videos from genuine content, as traditional detection methods often fail to keep pace with evolving deepfake techniques. This project addresses this problem domain by developing an advanced deep learning-based detection system that utilizes Res-Next Convolutional Neural Networks and Long Short Term Memory networks to identify subtle artifacts and inconsistencies in video frames. By leveraging diverse datasets and real-time data, the project aims to provide an effective and scalable solution to combat the spread of misleading and potentially harmful deepfake content, thereby enhancing the reliability of digital media and contributing to the broader fight against digital misinformation.

3.1.2 Functional Requirements

Functional requirements specify the specific features and functionalities that the deepfake detection system must provide. For this project, functional requirements might include:

- **Real-time processing**: The system should be able to process videos in real-time or near real-time to enable timely detection.
- **High accuracy**: The system should achieve a high accuracy rate in distinguishing between real and deepfake videos.
- **User-friendly interface**: The system should be easy to use, even for non-technical users.

- **Scalability**: The system should be able to handle large volumes of videos and adapt to future growth.
- **Integration capabilities**: The system should be compatible with various platforms and integrate with existing systems.

3.1.3 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.

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.

3.1.4 Risk Analysis

In the project, risk analysis focuses on identifying and mitigating potential issues that could impact the effectiveness of deepfake detection. Key risks include the over-blurring of facial areas compared to non-facial regions, flickering or instability in video frames, changes in skin tone near the edges of faces, and artifacts such as double chins or eyebrows. These risks are evaluated based on their likelihood of occurrence and their potential impact on the project's schedule, quality, and overall success.Careful dataset preparation and validation are crucial to minimize noise and

ensure the model's robustness in real-time scenarios. This thorough risk management approach ensures that the deepfake detection system remains accurate and reliable, even in challenging conditions.

3.1.5 Feasibility Study

The feasibility study for the deepfake detection project confirms its viability across technical, operational, and economic dimensions. Technically, it employs advanced deep learning techniques like Res-Next CNNs and LSTM networks, supported by PyTorch and Google Cloud Platform, ensuring robust handling of the project's complexities. Operationally, the web-based application provides a user-friendly interface for video uploads and real-time classification, making it accessible and scalable. Economically, the use of open-source tools and cloud resources manages costs effectively, while legal and ethical considerations are addressed by ensuring the technology combats misinformation and respects privacy. Overall, the study validates that the project is both feasible and impactful.

3.2 System Analysis

3.2.1 Solution Requirement

We analyzed the problem statement and found the feasibility of the solution of the problem. We read different research paper as mentioned in 2.1. After checking the feasibility of the problem statement. The next step is the dataset gathering and analysis. We analyzed 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 analyzed 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

3.2.2 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.

3.2.3 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.

3.2.4 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.

3.2.5 Outcomes

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.

3.3 System Design

3.3.1 System Architecture

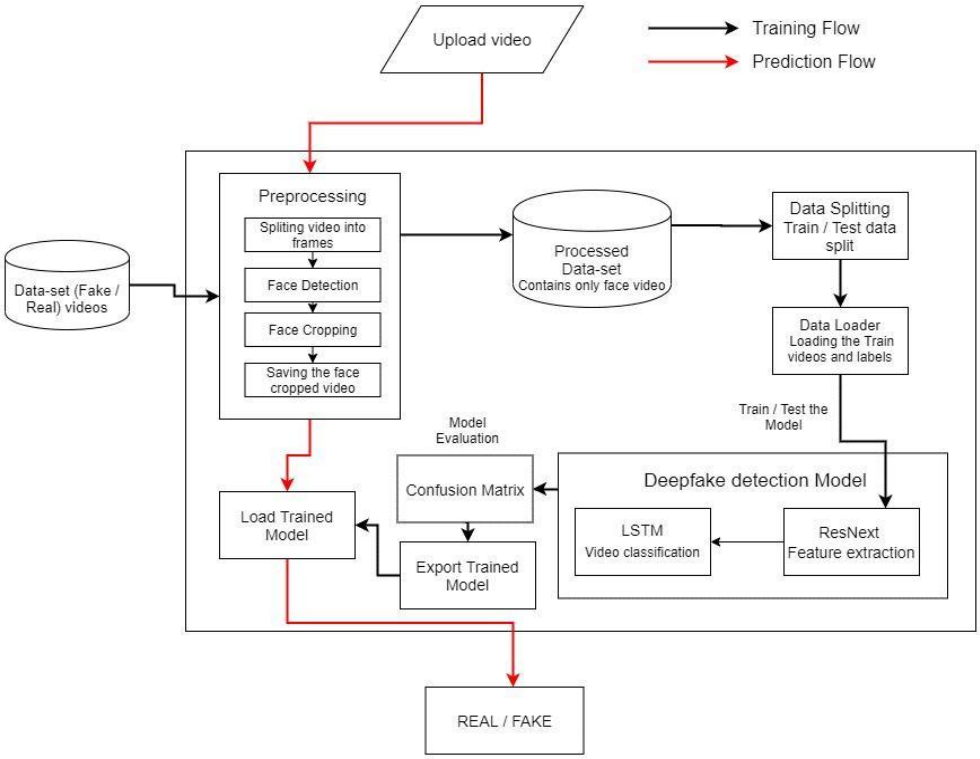


Figure 1: System Architecture

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.

• 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.

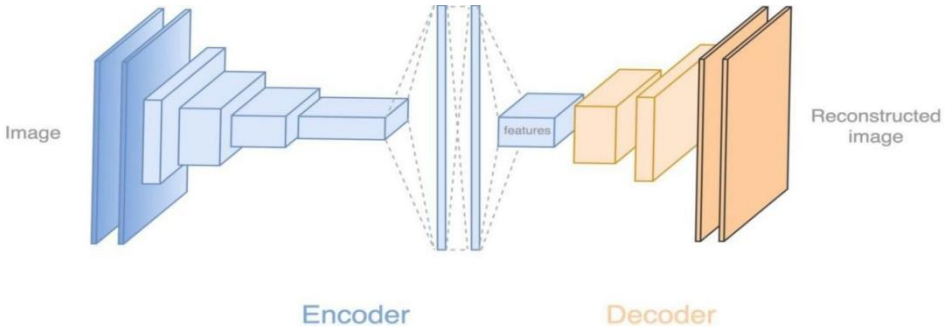


Figure 2: Deepfake generation

Figure 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

3.3.2 Architectural Design

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]. Futher we have mixed the dataset the collected datasets and created our own new dataset, to accurate and real time detection on different kind of videos. To avoid the training bias of the model we have considered 50% Real and 50% fake 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 DFDC dataset and removed the audio altered videos from the dataset by running a python script.

After preprocessing of the DFDC dataset, we have taken 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 CelebDF[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.

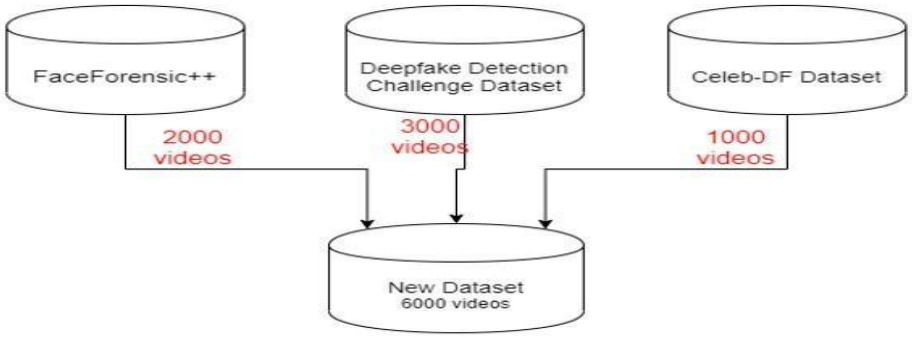


Figure 4: Dataset

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.

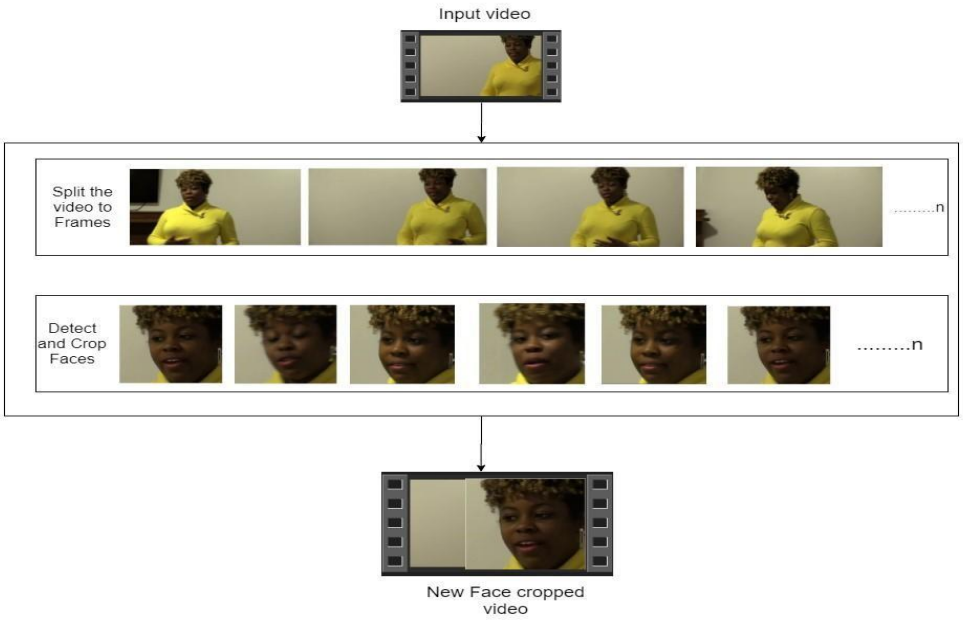


Figure 5: Pre-processing of video

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.

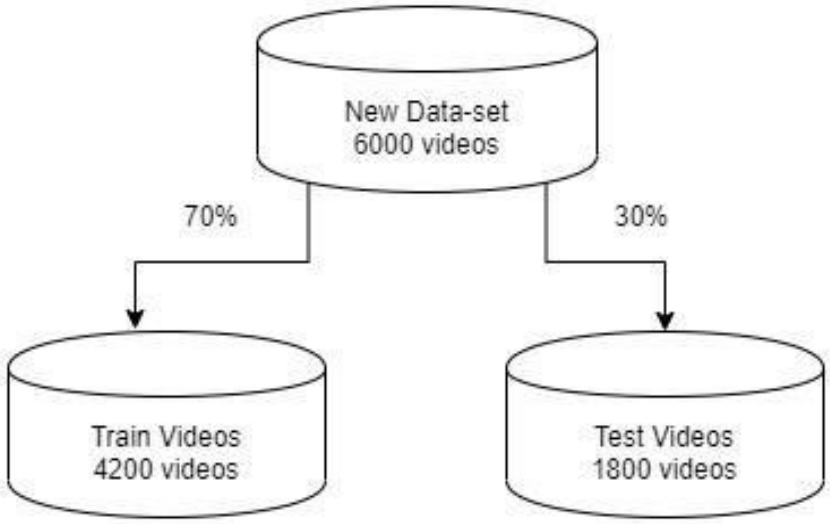


Figure 6: Train test split

Module 4: Model Architecture

Our model is a combination of CNN and RNN. We have used the Pre- trained ResNext 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 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 performance 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 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 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.

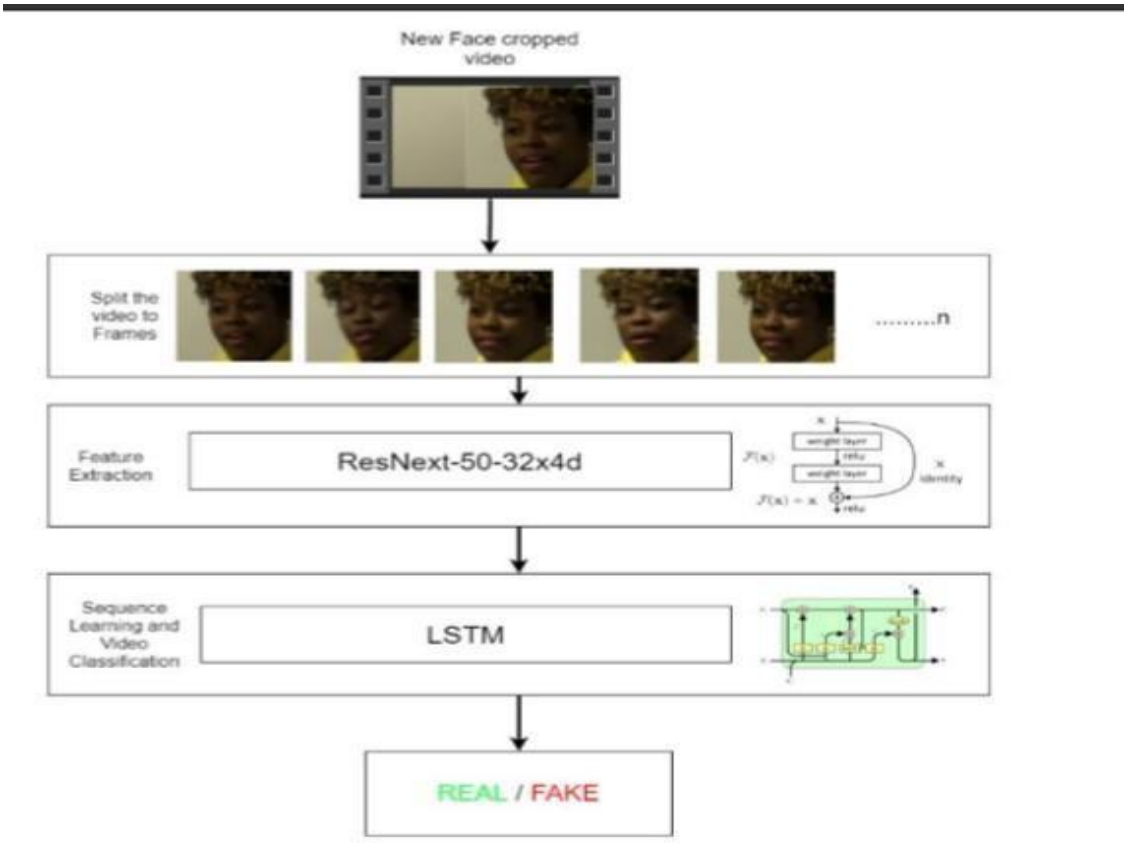


Figure 7: Overview of our model

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 hyperparameters 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.

3.4 Use Case Diagram

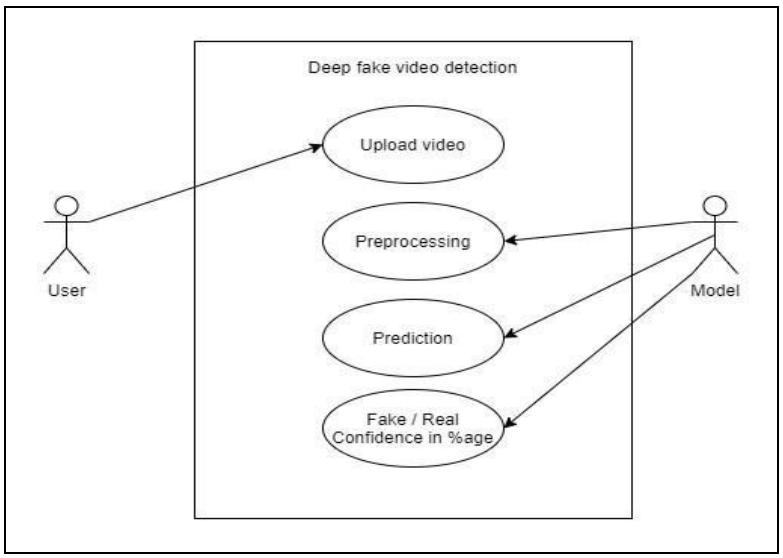


Figure 8: Use Case Diagram

3.5 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.

3.5.1 Data Flow Diagram

DFD Level-0

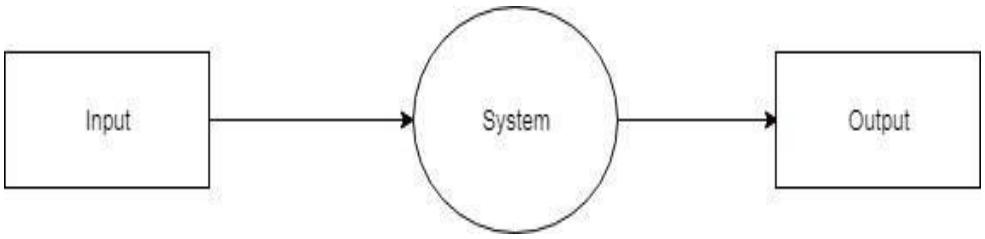


Figure 9: DFD Level 0

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

Input: Here input to the system is uploading video.

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

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

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

DFD Level-1

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

Where system gives detailed information of the procedure taking place.

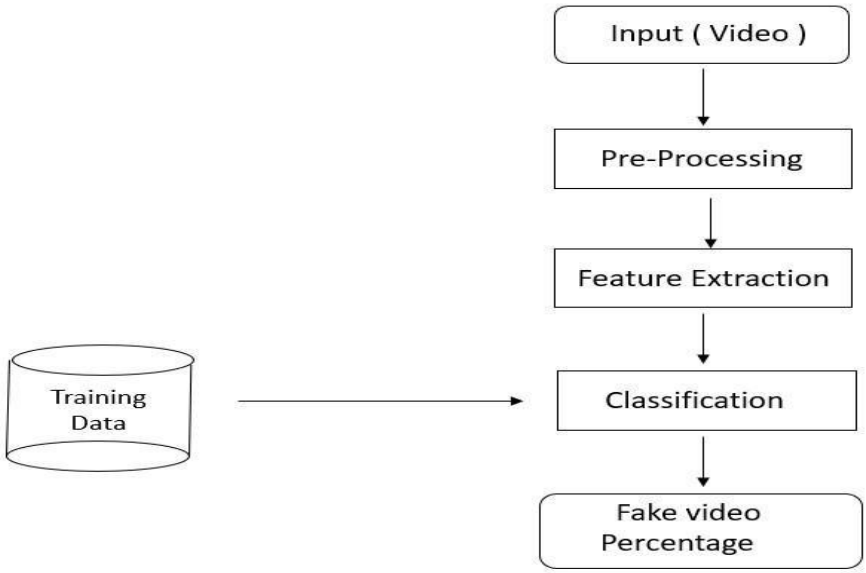


Figure 10: DFD Level 1

DFD Level-2

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

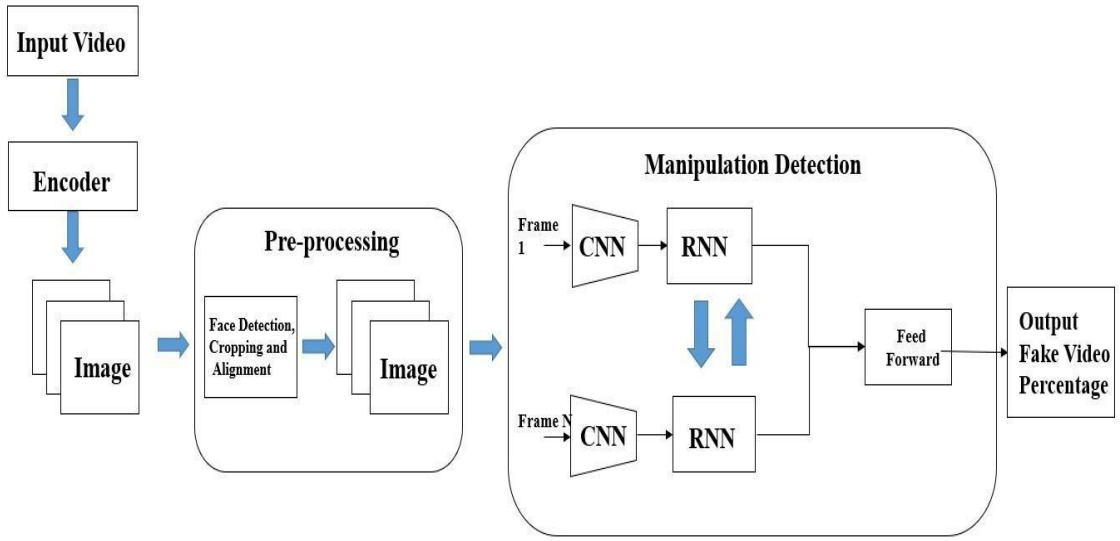


Figure 11: DFD Level 2

3.6 Activity Diagram:

Training Workflow:

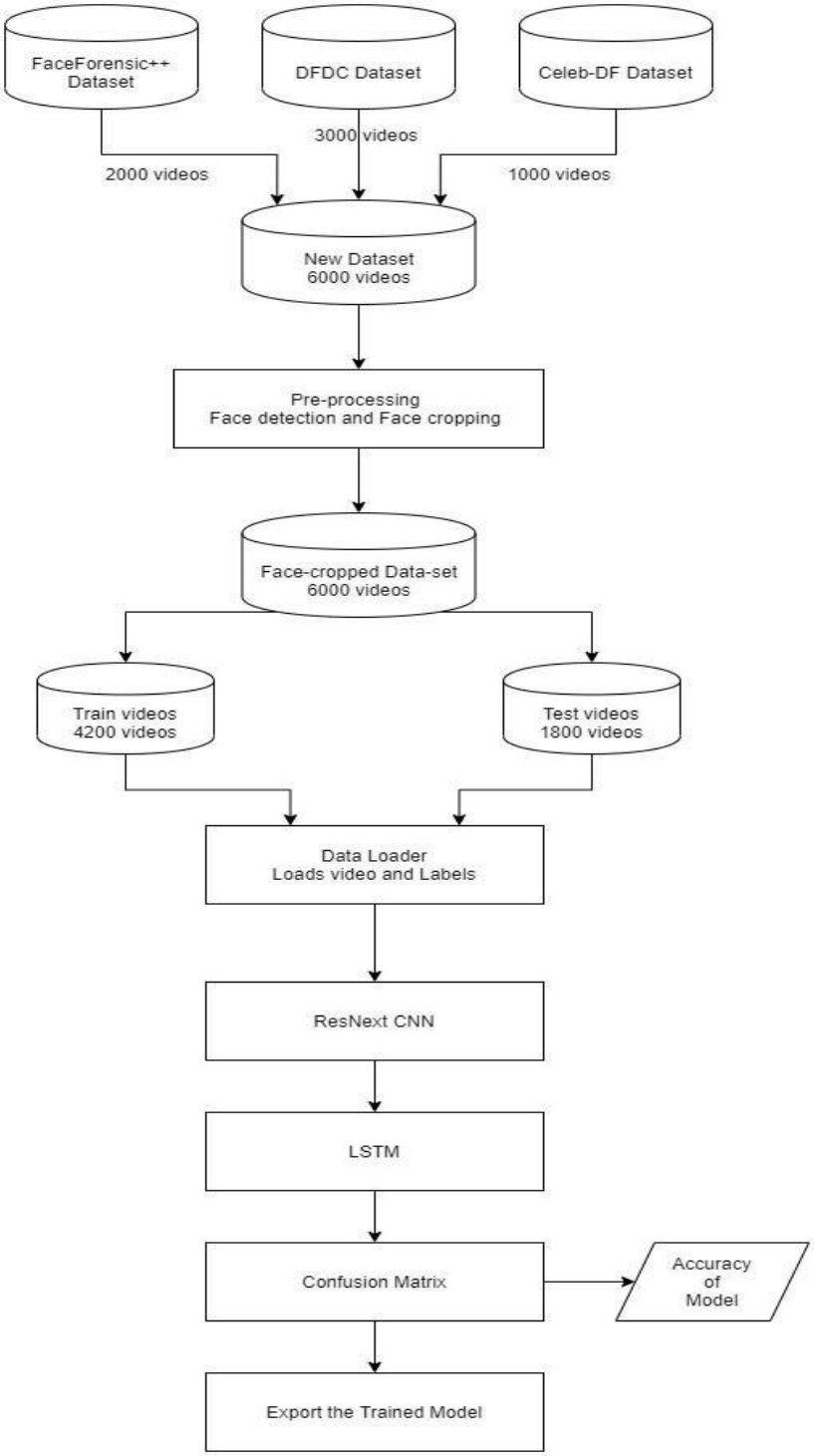


Figure12: Training Workflow

Testing Workflow:

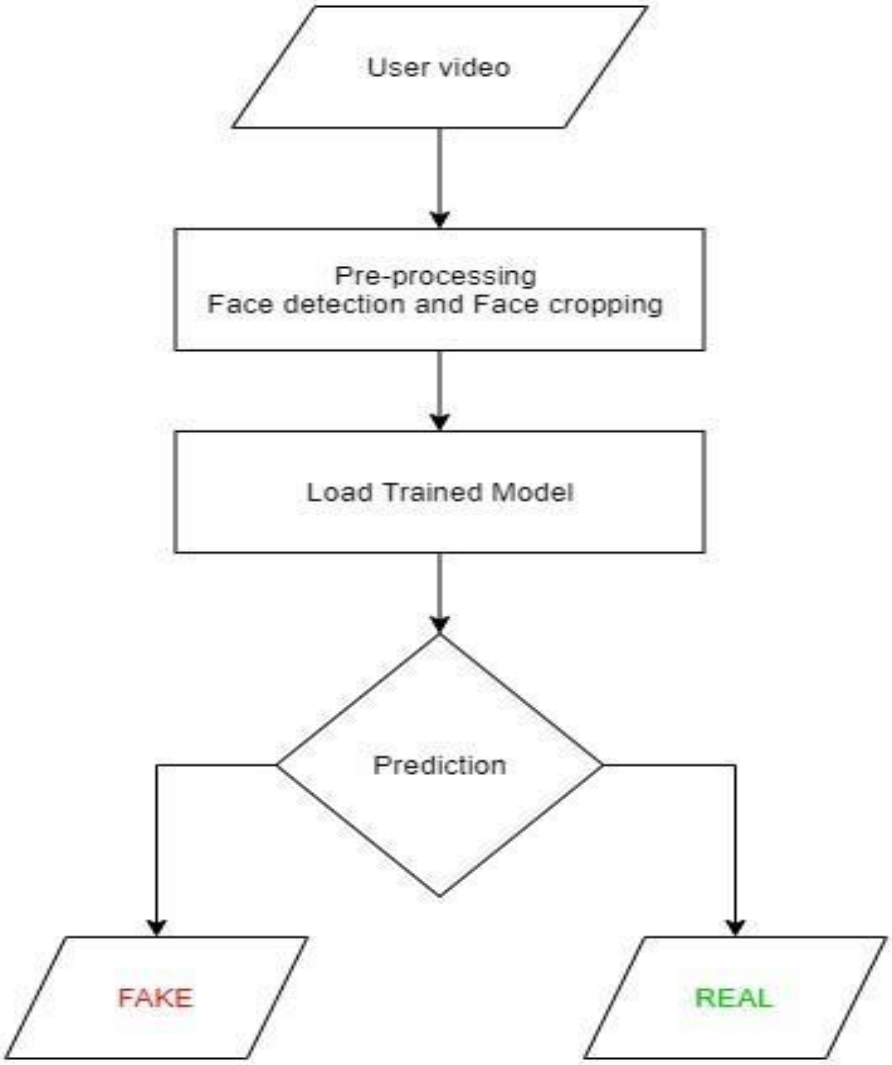
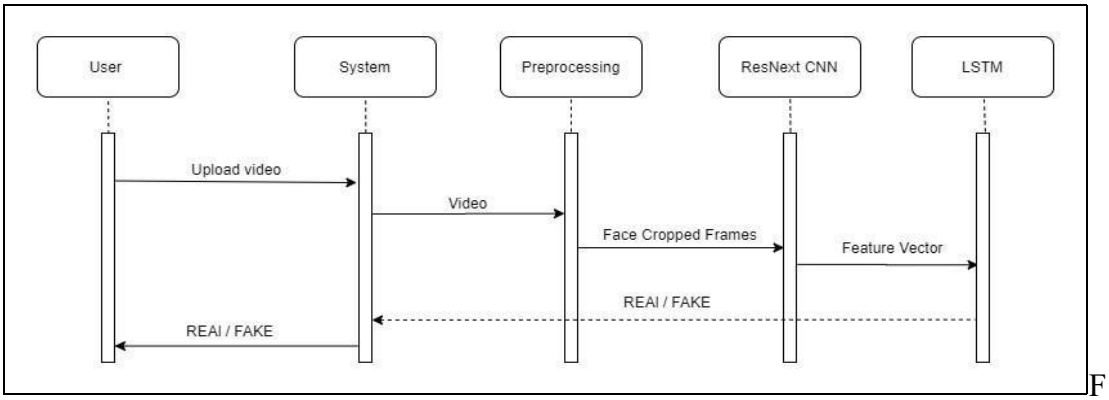


Figure 13: Testing Workflow

3.7 Sequence Diagram



F

Figure 14: Sequence Diagram

CHAPTER 4

IMPLEMENTATION AND TESTING

4.1 Introduction

There are many examples where deepfake creation technology is used to mislead 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 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 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.

4.2 Tools and Technologies Used

4.2.1 Planning

1. OpenProject

4.2.2 UML Tools

1. draw.io

4.2.3 Programming Languages

1. Python3
2. JavaScript

4.2.4 Programming Frameworks

1. PyTorch
2. Django

4.2.5 IDE

1. GoogleCollab
2. Jupyter Notebook
3. Visual Studio Code

4.2.6 Versioning Control

1. Git

4.2.7 Cloud Services

1. Google Cloud Platform

4.2.8 Application and web servers:

1. Google Cloud Engine

4.2.9 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

4.3 Algorithm Details

4.3.1 Dataset Details

Refer 3.3.2

4.3.2 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.

4.3.3 Model Details

The model consists of following layers:

ResNext CNN : The pre-trained model of Residual Convolution Neural Network 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 15: ResNext Architecture

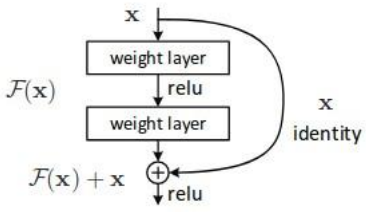


Figure 16: ResNext Working

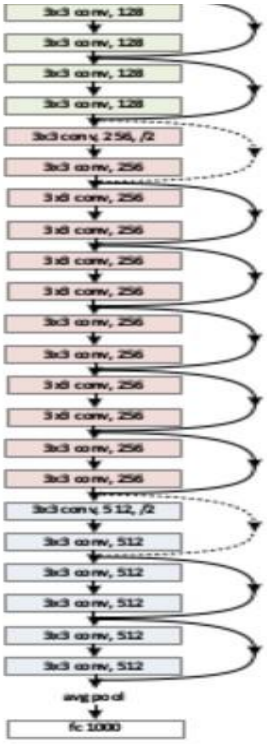


Figure 17: Overview of ResNext Architecture

- 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.

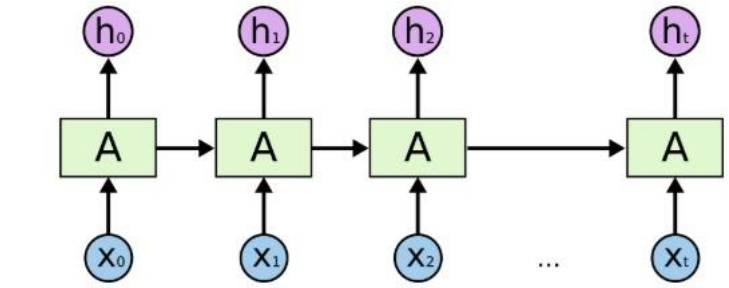


Figure 18: Overview of LSTM Architecture

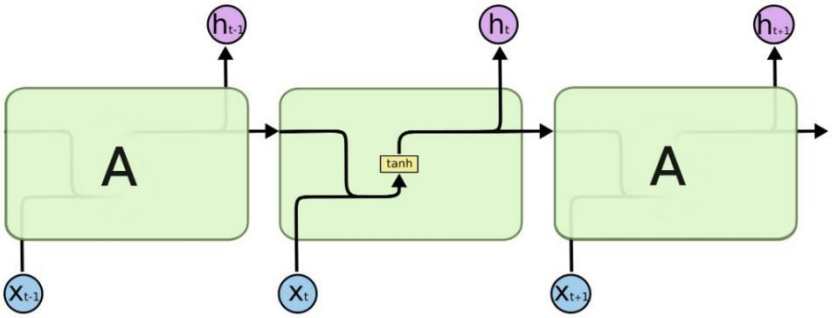


Figure 19: Internal LSTM Architecture

- 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.

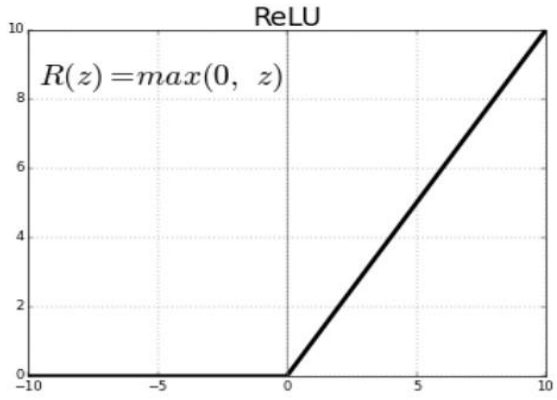


Figure 20: Relu Activation function

- Dropout Layer :Dropout layer with the value of 0.4 is used to avoid overfitting 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.

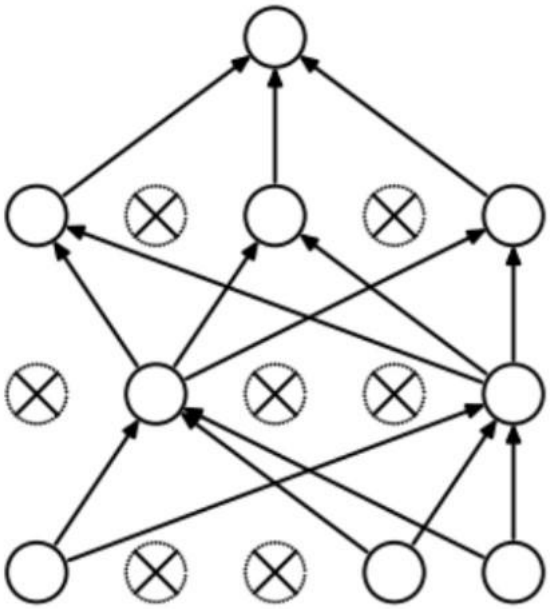


Figure 21: Dropout layer overview

- 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.

4.3.4 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.

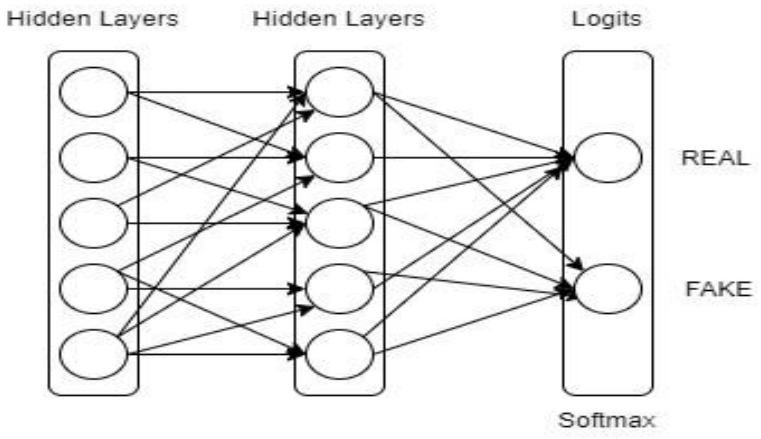


Figure 22: Softmax Layer

- 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.

4.3.5 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.

4.4 Type of Testing Used

- Functional Testing

- Unit Testing

- Integration Testing

- System Testing

- Interface Testing

- Non-functional Testing

- Performance Testing

- Load Testing

- Compatibility Testing

4.4.1 Test Cases and Test Results

Test Cases

Table 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

4.5 Outputs

4.5.1 Model results

Table 2: 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 + FaceForensic++	3000	100	93.97781
model_87_acc_20_frames_final_data	Our Dataset	6000	20	77.793
model_84_acc_10_frames_final_data	Our Dataset	6000	10	74.662
model_89_acc_40_frames_final_data	Our Dataset	6000	40	79.345
model_89_acc_60_frames_final_data	Our Dataset	6000	60	82.5974

model_89_acc _80_frames_ final_data	Our Dataset	6000	80	84.4908
model_89_acc _100_frames_ final_data	Our Dataset	6000	100	85.1023

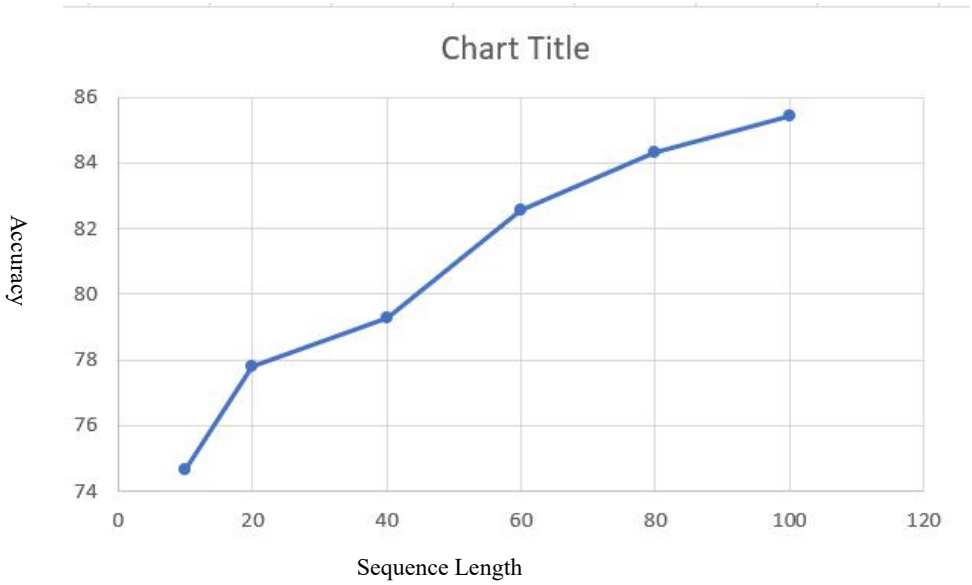


Figure 30. Output Graph Representation

CHAPTER 5

CONCLUSION AND FUTURE RECOMMENDATION

5.1 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 processing 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.

5.2 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.

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-detectionchallenge/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)

Appendix

- **User Interface**

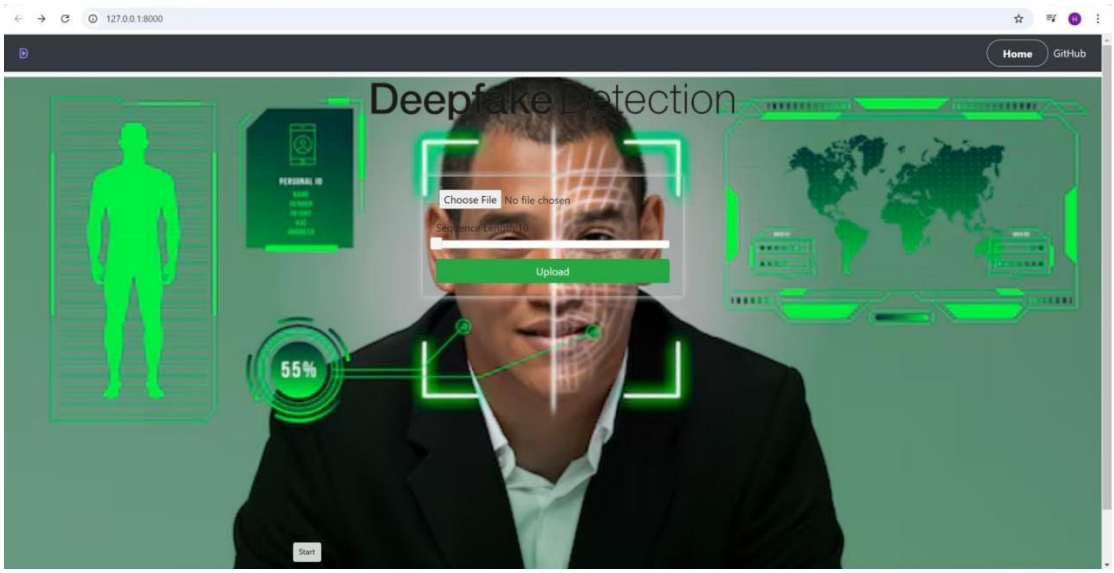


Figure 23: Home Page

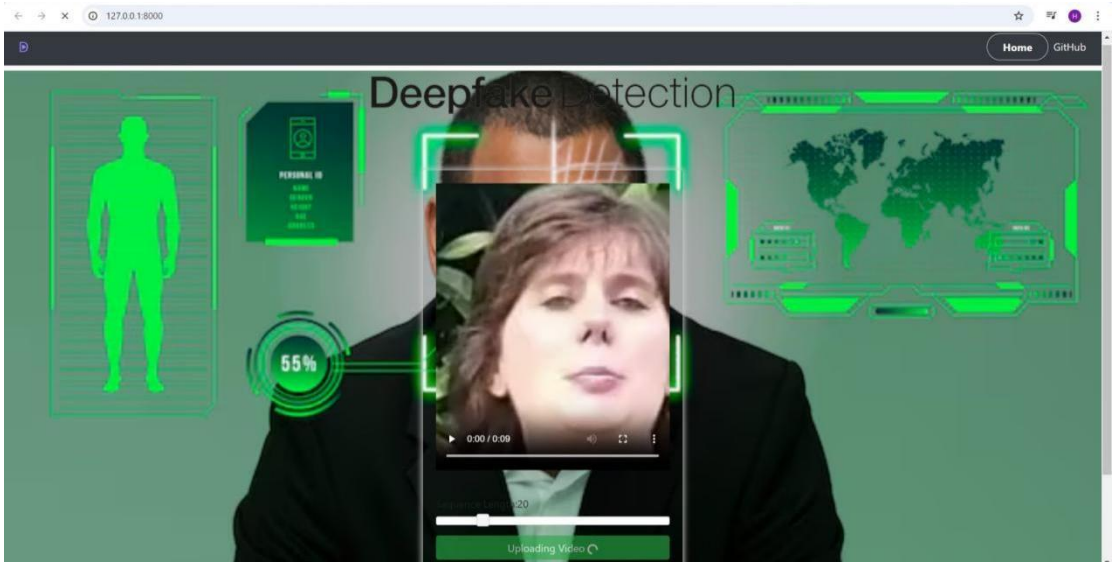


Figure 24: Uploading Real Video

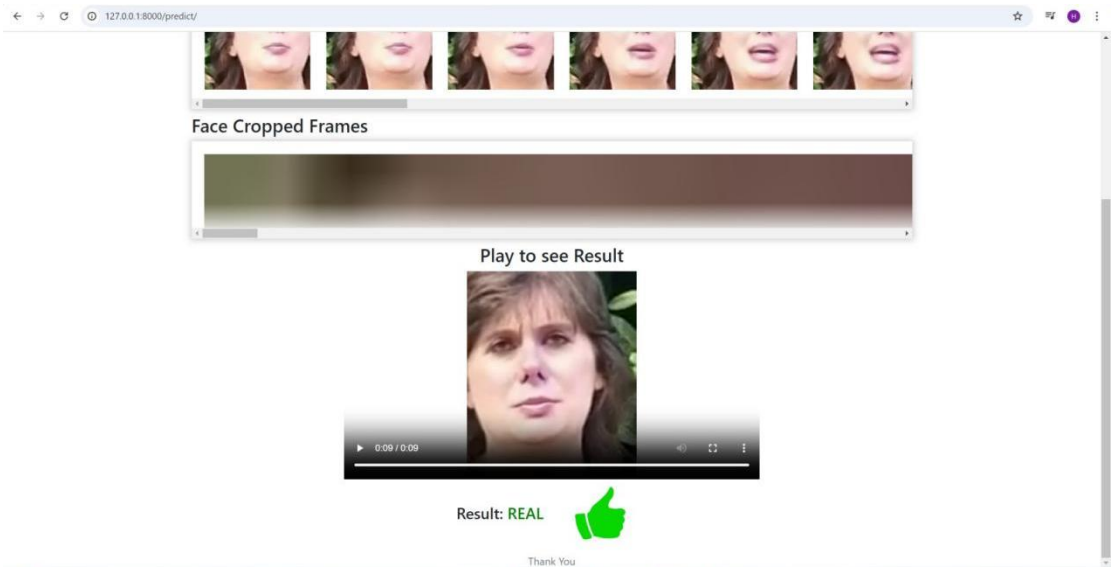


Figure 25: Real Video Output

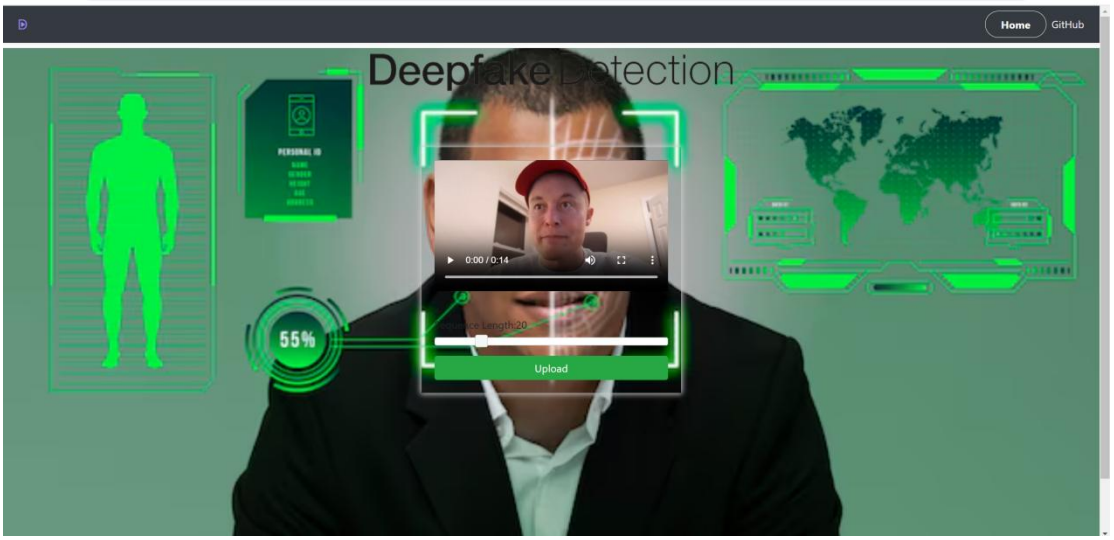


Figure 26: Uploading Fake Video

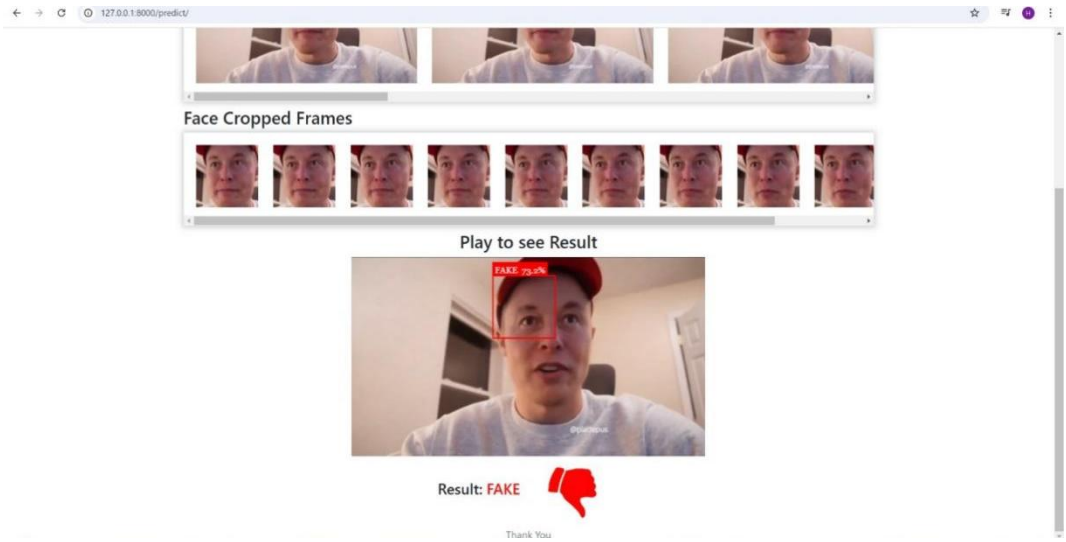


Figure 27: Fake video Output

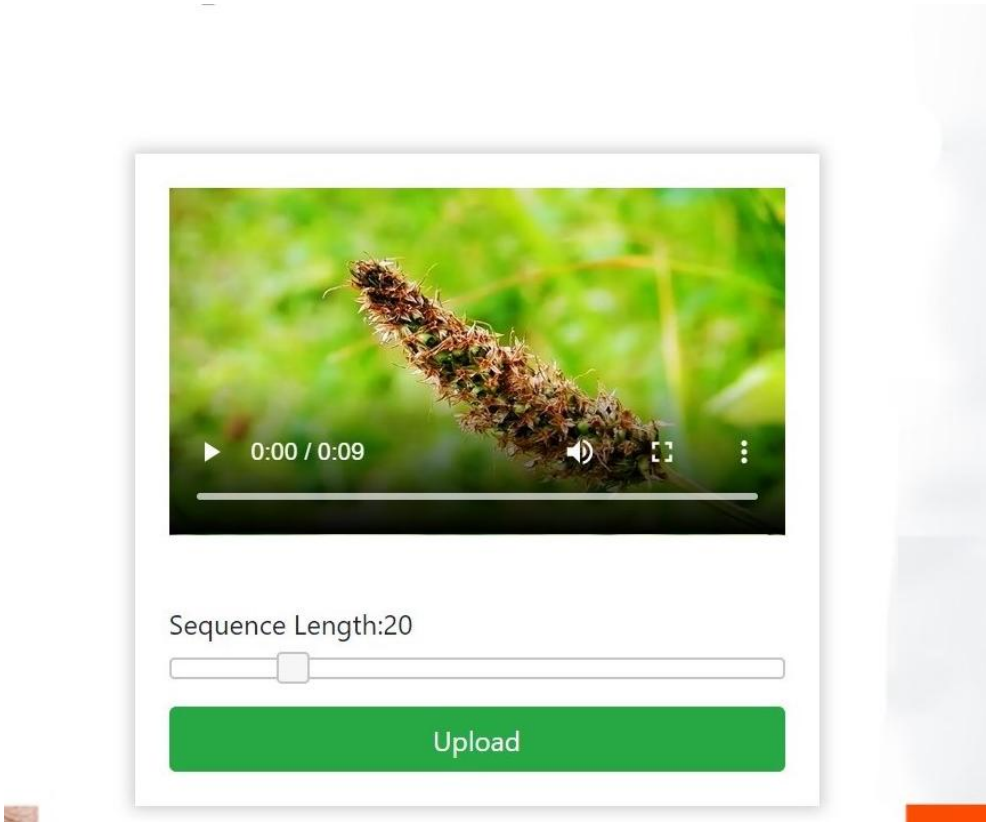


Figure 28: Uploading Video with no faces

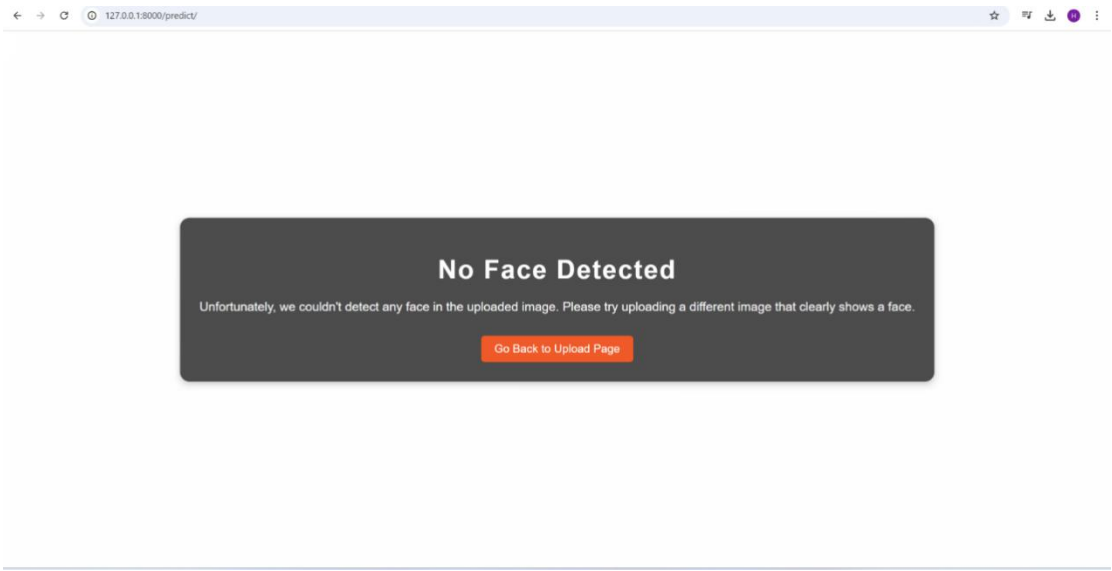


Figure 29: Output of Uploaded video with no faces

- **Code snippets:**

```
import seaborn as sn
#Output confusion matrix
def print_confusion_matrix(y_true, y_pred):
    cm = confusion_matrix(y_true, y_pred)
    print('True positive = ', cm[0][0])
    print('False positive = ', cm[0][1])
    print('False negative = ', cm[1][0])
    print('True negative = ', cm[1][1])
    print('\n')
    df_cm = pd.DataFrame(cm, range(2), range(2))
    sn.set(font_scale=1.4) # for label size
    sn.heatmap(df_cm, annot=True, annot_kws={"size": 16}) # font size
    plt.ylabel('Actual label', size = 20)
    plt.xlabel('Predicted label', size = 20)
    plt.xticks(np.arange(2), ['Fake', 'Real'], size = 16)
    plt.yticks(np.arange(2), ['Fake', 'Real'], size = 16)
    plt.ylim([2, 0])
    plt.show()
    calculated_acc = (cm[0][0]+cm[1][1])/(cm[0][0]+cm[0][1]+cm[1][0]+ cm[1][1])
    print("Calculated Accuracy",calculated_acc*100)
```

Figure 30: Confusion Matrix

```python
#Model with feature visualization
from torch import nn
from torchvision import models
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) #Residual Network CNN
        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(torch.mean(x_lstm,dim = 1)))
```

Figure 31: Pretrained RNN Model

```python
#Code for making prediction
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 = ['/content/drive/My Drive/Balanced_Face_only_data/aagfhgtpmv.mp4',
                                '/content/drive/My Drive/Balanced_Face_only_data/aczrgyricp.mp4',
                                '/content/drive/My Drive/Balanced_Face_only_data/agdkmztvby.mp4',
                                '/content/drive/My Drive/Balanced_Face_only_data/abarnvbtwb.mp4']

path_to_videos = ['/content/drive/My Drive/Youtube_Face_only_data/000_003.mp4',
                  '/content/drive/My Drive/Youtube_Face_only_data/000.mp4',
                  '/content/drive/My Drive/Youtube_Face_only_data/002_006.mp4',
                  '/content/drive/My Drive/Youtube_Face_only_data/002.mp4'


]

path_to_videos= ["/content/drive/My Drive/DFDC_REAL_Face_only_data/aabqyygbaa.mp4"]

video_dataset = validation_dataset(path_to_videos,sequence_length = 20,transform = train_transforms)
model = Model(2).cuda()
path_to_model = '/content/drive/My Drive/Models/model_87_acc_20_frames_final_data.pt'
model.load_state_dict(torch.load(path_to_model))
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")
```

Figure 32: Prediction process

```
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
def im_convert(tensor):
    """ 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)
    cv2.imwrite('./2.png',image*255)
    return image

def predict(model,img,path = './'):
  fmap,logits = model(img.to('cuda'))
  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)
  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[i
  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,:,:,:])
  result = heatmap * 0.5 + img*0.8*255
  cv2.imwrite('/content/1.png',result)
  result1 = heatmap * 0.5/255 + img*0.8
  r,g,b = cv2.split(result1)
  result1 = cv2.merge((r,g,b))
```

Figure 33: Prediction Function