

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

Contents

1	Synopsis	1
2	Technical Keywords	2
2.1	Area of Project	2
2.2	Technical Keywords	2
3	Introduction	3
3.1	Project Idea	3
3.2	Motivation of the Project	4
3.3	Literature Survey	5
4	Problem Definition and scope	7
4.1	Problem Statement	7
4.1.1	Goals and objectives	7
4.1.2	Statement of scope	8
4.2	Major Constraints	8
4.3	Methodologies of Problem solving	9
4.3.1	Analysis	9
4.3.2	Design	10
4.3.3	Development	10
4.3.4	Evaluation	10
4.4	Outcome	10
4.5	Applications	10
4.6	Hardware Resources Required	10
4.7	Software Resources Required	11
5	Project Plan	12
5.1	Project Model Analysis	12
5.1.1	Reconciled Estimates	13
5.1.2	Cost Estimation using COCOMO(Constructive Cost) Model	13

5.2	Risk Management w.r.t. NP Hard analysis	14
5.2.1	Risk Identification	14
5.2.2	Risk Analysis	14
5.3	Project Schedule	16
5.3.1	Project task set	16
5.3.2	Timeline chart	16
6	Software requirement specification	17
6.1	Introduction	17
6.1.1	Purpose and Scope of Document	17
6.1.2	Use Case View	17
6.2	Functional Model and Description	18
6.2.1	Data Flow Diagram	18
6.2.2	Activity Diagram:	20
6.2.3	Non Functional Requirements:	21
6.2.4	Sequence Diagram	22
7	Detailed Design Document	23
7.1	Introduction	23
7.1.1	System Architecture	23
7.2	Architectural Design	25
7.2.1	Module 1 : Data-set Gathering	25
7.2.2	Module 2 : Pre-processing	26
7.2.3	Module 3: Data-set split	27
7.2.4	Module 4: Model Architecture	28
7.2.5	Module 5: Hyper-parameter tuning	29
8	Project Implementation	31
8.1	Introduction	31
8.2	Tools and Technologies Used	32
8.2.1	Planning	32
8.2.2	UML Tools	32
8.2.3	Programming Languages	32
8.2.4	Programming Frameworks	32
8.2.5	IDE	32
8.2.6	Versioning Control	32
8.2.7	Cloud Services	32

8.2.8	Application and web servers:	33
8.2.9	Libraries	33
8.3	Algorithm Details	33
8.3.1	Dataset Details	33
8.3.2	Preprocessing Details	33
8.3.3	Model Details	34
8.3.4	Model Training Details	37
8.3.5	Model Prediction Details	39
9	Software Testing	40
9.1	Type of Testing Used	40
9.2	Test Cases and Test Results	41
10	Results and Discussion	42
10.1	Screen shots	42
10.2	Outputs	46
10.2.1	Model results	46
11	Deployment and Maintenance	48
11.1	Deployment	48
11.2	Maintenance	48
12	Conclusion and Future Scope	50
12.1	Conclusion	50
12.2	Future Scope	50
A	References	51
B	Project Planner	53
C	Paper Published Summary	55
D	Participation Certificate	59
E	Plagiarism Report	60
F	Information of Project Group Members	61

List of Figures

5.1 Spiral Methodology SDLC	12
6.1 Use case diagram	17
6.2 DFD Level 0	18
6.3 DFD Level 1	19
6.4 DFD Level 2	19
6.5 Training Workflow	20
6.6 Testing Workflow	21
6.7 Sequence Diagram	22
7.1 System Architecture	23
7.2 Deepfake generation	24
7.3 Face Swapped deepfake generation	24
7.4 Dataset	26
7.5 Pre-processing of video	27
7.6 Train test split	28
7.7 Overview of our model	29
8.1 ResNext Architecture	34
8.2 ResNext Working	35
8.3 Overview of ResNext Architecture	35
8.4 Overview of LSTM Architecture	36
8.5 Internal LSTM Architecture	36
8.6 Relu Activation function	36
8.7 Dropout layer overview	37
8.8 Softmax Layer	38
10.1 Home Page	42
10.2 Uploading Real Video	43
10.3 Real Video Output	43
10.4 Uploading Fake Video	44

10.5 Fake video Output	44
10.6 Uploading Video with no faces	45
10.7 Output of Uploaded video with no faces	45
10.8 Uploading file greater than 100MB	46
10.9 Pressing Upload button without selecting video	46
B.1 Complete Project Plan	53
B.2 Project Plan 1	54
B.3 Project Plan 2	54
C.1 Abhijit Jadhav	56
C.2 Hitendra Patil	56
C.3 Jay Patel	57
C.4 Abhishek Patange	57
C.5 Manjushri Mahajan	58
D.1 Participation Certificate	59
E.1 Plagiarism Report	60

List of Tables

4.1	Hardware Requirements	11
5.1	Cost Estimation	13
5.2	Risk Description	15
5.3	Risk Probability definitions	15
9.1	Test Case Report	41
10.1	Trained Model Results	47

Chapter 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 combination 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-Forensics++[1], Deepfake detection challenge[2], and Celeb-DF[3] data-sets. We also evaluated our model against the large amount of real time data like YouTube data-set to achieve competitive results in the real time scenario's.

Chapter 2

Technical Keywords

2.1 Area of Project

Our project is a Deep learning project which is a sub branch of Artificial Intelligence and deals with the human brain inspired neural network technology. Computer 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.

2.2 Technical Keywords

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

Chapter 3

Introduction

3.1 Project Idea

In the world of ever growing Social media platforms, Deepfakes are 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 use 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 will upload the video. The video will be processed by the model and the output will be rendered back to the user with the classification of the video as deepfake or real and confidence of the model.

3.2 Motivation of the Project

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

Like 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 platform. 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.

3.3 Literature Survey

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 can not 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 pre-trained model. Their process used the HOHO [19] dataset consisting of just 600 videos.

Their dataset consists small number of videos and same type of videos, which may not perform very well on the real time data. We will be training our 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 4

Problem Definition and scope

4.1 Problem Statement

Convincing manipulations of digital images and videos have been demonstrated for several decades through the use of visual effects, recent advances in deep 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.

4.1.1 Goals and objectives

Goal and Objectives:

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

4.1.2 Statement of scope

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

4.2 Major Constraints

- **User:** User of the application will be able detect the whether the uploaded video is fake or real, Along with the model confidence of the prediction.
- **Prediction:** The User will be able to see the playing video with the output on the face along with the confidence of the model.
- **Easy and User-friendly User-Interface:** Users seem to prefer a more 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.

4.3 Methodologies of Problem solving

4.3.1 Analysis

- Solution Requirement

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

- Solution Constraints

We analysed the solution in terms of cost, speed of processing, requirements, level of expertise, availability of equipment's.

- Parameter Identified

1. Blinking of eyes
2. Teeth enchantment
3. Bigger distance for eyes
4. Moustaches
5. Double edges, eyes, ears, nose
6. Iris segmentation
7. Wrinkles on face
8. Inconsistent head pose
9. Face angle
10. Skin tone
11. Facial Expressions
12. Lighting
13. Different Pose
14. Double chins
15. Hairstyle
16. Higher cheek bones

4.3.2 Design

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

4.3.3 Development

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

4.3.4 Evaluation

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

4.4 Outcome

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

4.5 Applications

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

4.6 Hardware Resources Required

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

- **Client-side Requirements:** Browser: Any Compatible browser device

Table 4.1: Hardware Requirements

Sr. No.	Parameter	Minimum Requirement
1	Intel Xeon E5 2637	3.5 GHz
2	RAM	16 GB
3	Hard Disk	100 GB
4	Graphic card	NVIDIA GeForce GTX Titan (12 GB RAM)

4.7 Software Resources Required

Platform :

1. Operating System: Windows 7+
2. Programming Language : Python 3.0
3. Framework: PyTorch 1.4 , Django 3.0
4. Cloud platform: Google Cloud Platform
5. Libraries : OpenCV, Face-recognition

Chapter 5

Project Plan

5.1 Project Model Analysis

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

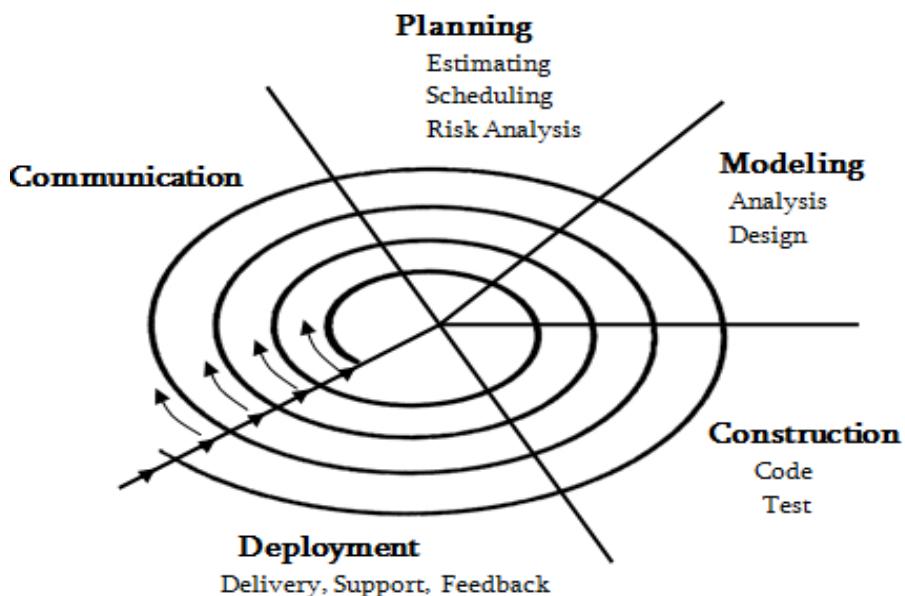


Figure 5.1: Spiral Methodology SDLC

5.1.1 Reconciled Estimates

1. **Cost Estimate :** Rs 11,600

Table 5.1: Cost Estimation

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

2. **Time Estimates :** 12 Months (refer Appendix B)

5.1.2 Cost Estimation using COCOMO(Constructive Cost) Model

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

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

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

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

$$E = 57.2206PM$$

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

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

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

$$D = 11.6M$$

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

$$PeopleRequired(P) = \frac{E}{D}$$

$$P = \frac{57.2206}{11.6}$$

$$P = 4.93$$

5.2 Risk Management w.r.t. NP Hard analysis

5.2.1 Risk Identification

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

1. Remove any picture frames that contain more than one person.
2. Make sure you have an abundance of video footage. Extract facial pictures contain different pose, face angle, and facial expressions.
3. Some resembling of both persons may help, like similar face shape.

5.2.2 Risk Analysis

In Deepfakes, it creates a mask on the created face so it can blend in with the target video. To further eliminate the artifacts

1. Apply a Gaussian filter to further diffuse the mask boundary area.
2. Configure the application to expand or contract the mask further.
3. Control the shape of the mask.

Table 5.2: Risk Description

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

Table 5.3: Risk Probability definitions

Probability	Value	Description
High	Probability of occurrence is	> 75%
Medium	Probability of occurrence is	26 – 75%
Low	Probability of occurrence is	< 25%

5.3 Project Schedule

5.3.1 Project task set

Major Tasks in the Project stages are

- Task 1: Data-set gathering and analysis

This task consists of downloading the dataset. Analysing the dataset and making the dataset ready for the preprocessing.

- Task 2 : Module 1 implementation

Module 1 implementation consists of splitting the video to frames and cropping each frame consisting of face.

- Task 3: Pre-processing

Pre-processing includes the creation of the new dataset which includes only face cropped videos.

- Task 4: Module 2 implementation

Module 2 implementation consists of implementation of DataLoader for loading the video and labels. Training a base line model on small amount of data.

- Task 5 : Hyper parameter tuning

This task includes the changing of the Learning rate, batch size, weight decay and model architecture until the maximum accuracy is achieved.

- Task 6 : Training the final model

The final model on large dataset is trained based on the best hyper parameter identified in the Task 5.

- Task 7 : Front end Development

This task includes the front end development and integration of the back-end and front-end.

- Task 8 : Testing

The complete application is tested using unit testing,

5.3.2 Timeline chart

Please refer Annex C for the planner

Chapter 6

Software requirement specification

6.1 Introduction

6.1.1 Purpose and Scope of Document

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

6.1.2 Use Case View

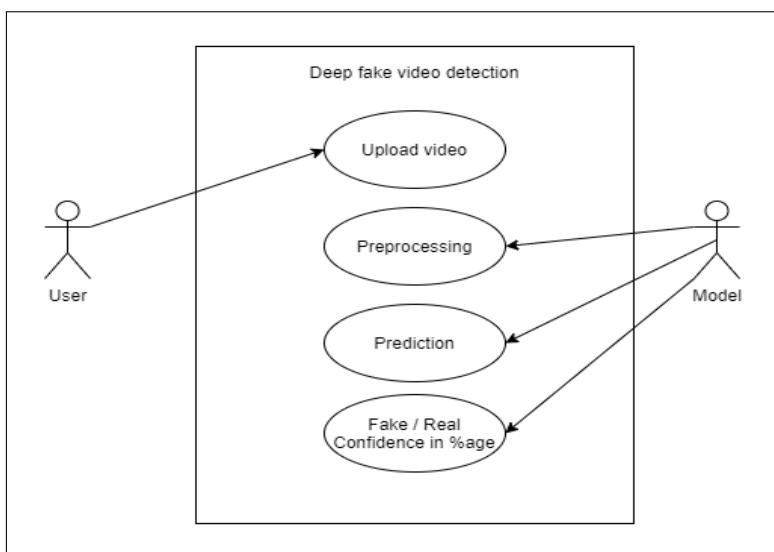


Figure 6.1: Use case diagram

6.2 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.

6.2.1 Data Flow Diagram

DFD Level-0

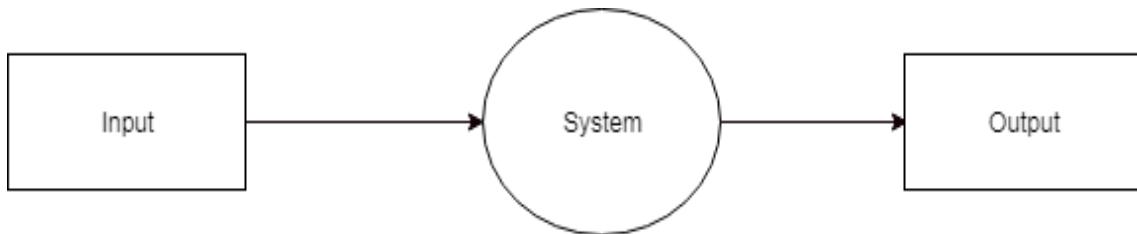


Figure 6.2: DFD Level 0

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

- 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

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

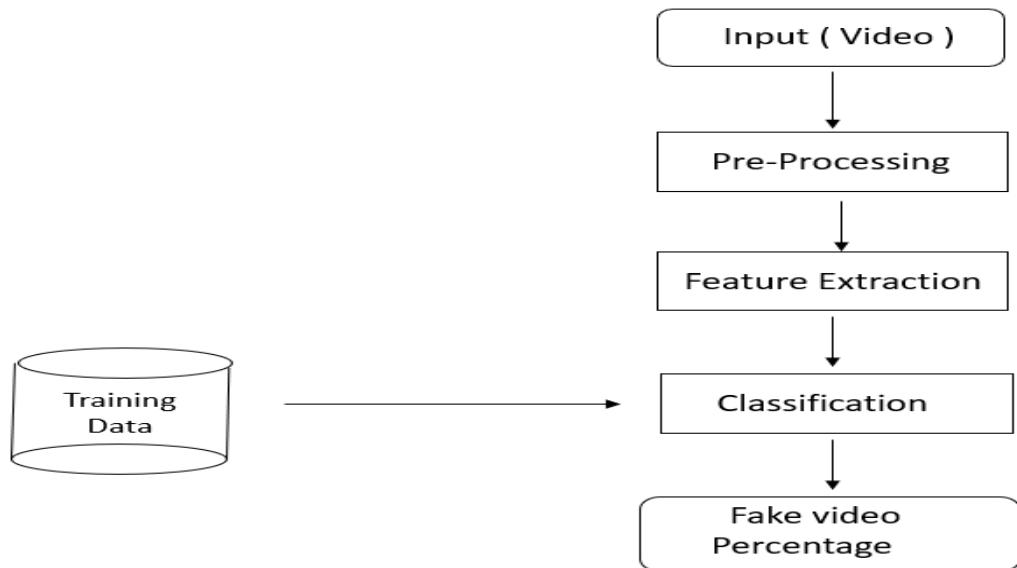


Figure 6.3: DFD Level 1

DFD Level-2

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

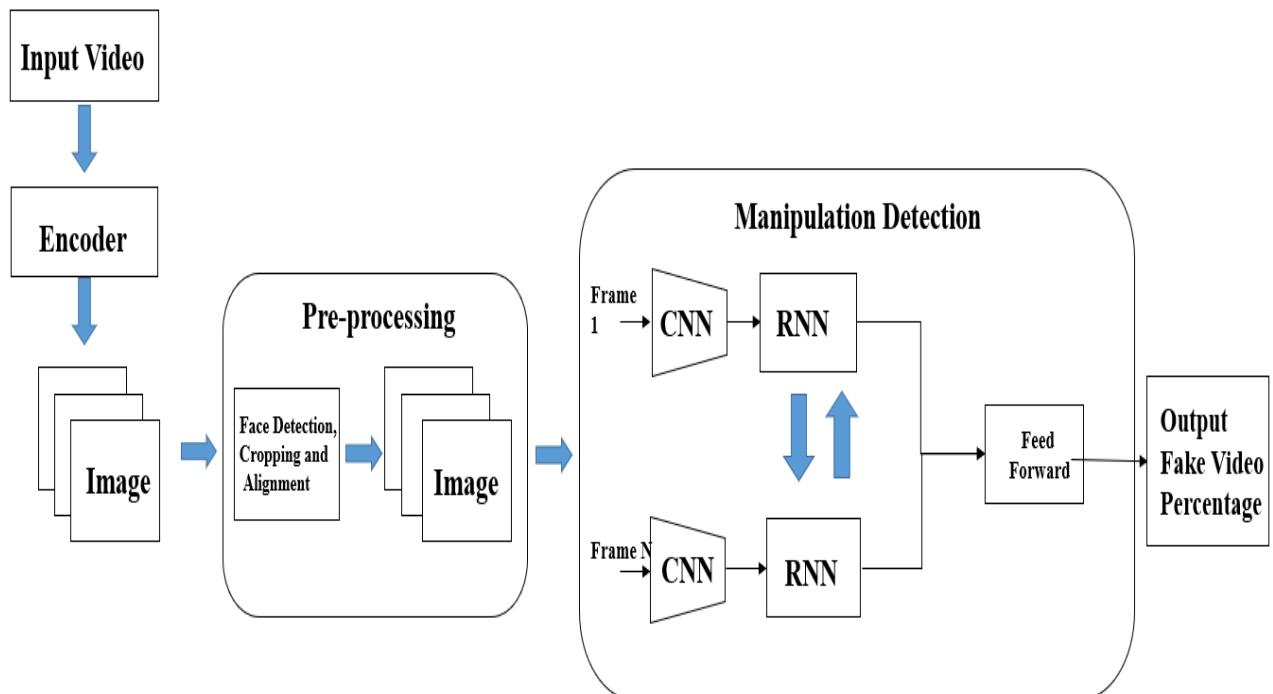


Figure 6.4: DFD Level 2

6.2.2 Activity Diagram:

Training Workflow:

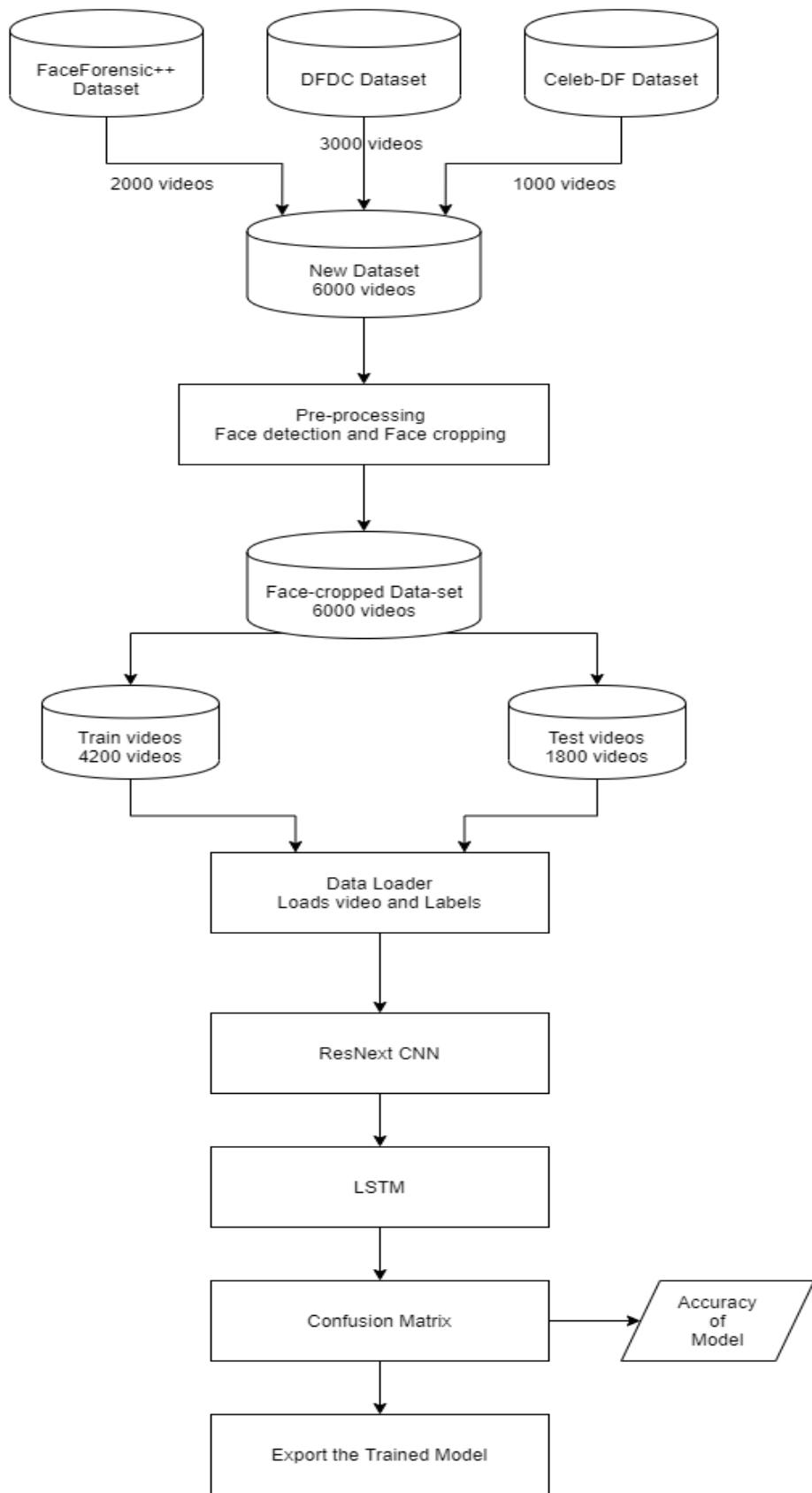


Figure 6.5: Training Workflow

Testing Workflow:

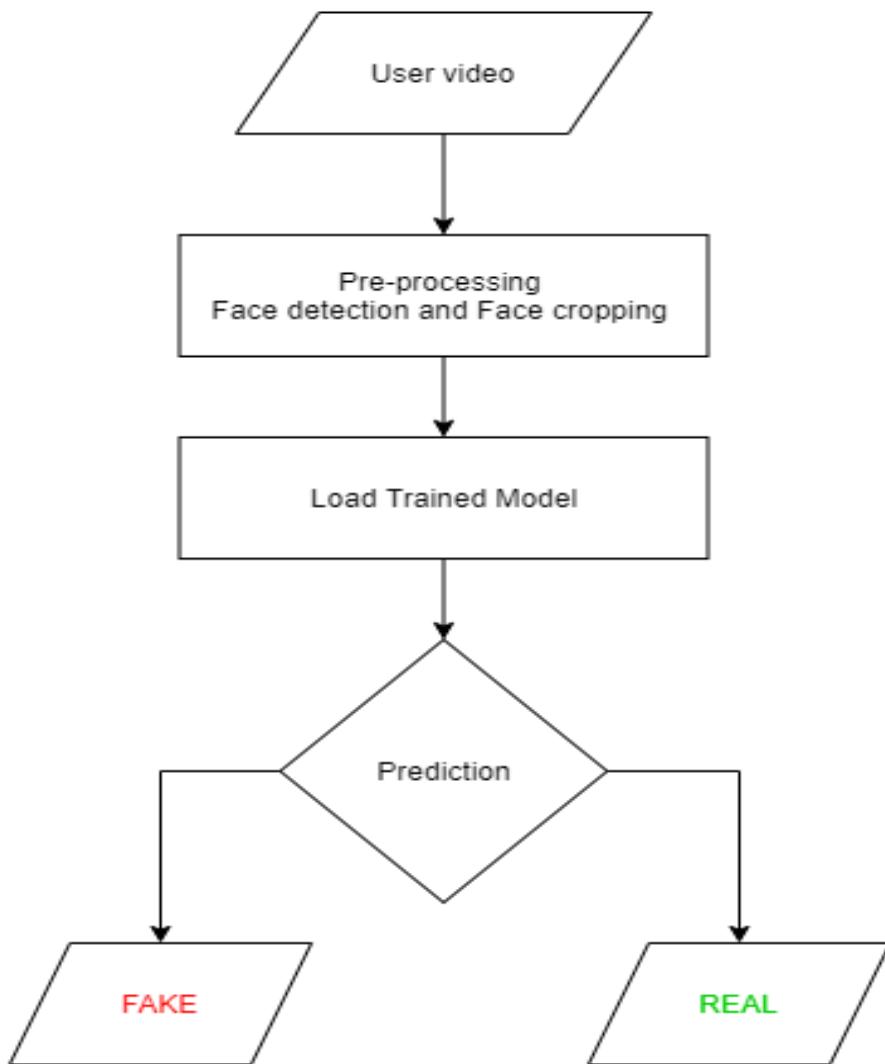


Figure 6.6: Testing Workflow

6.2.3 Non Functional Requirements:

Performance 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 able for human manipulation.
- To extent the safety of the videos uploaded by the user will be deleted after 30 min from the server.

Security Requirement

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

6.2.4 Sequence Diagram

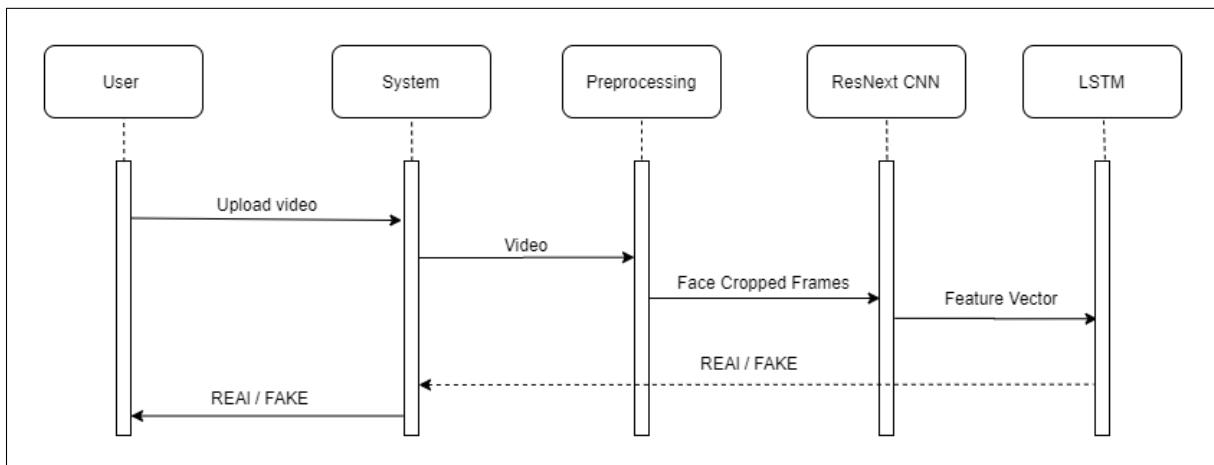


Figure 6.7: Sequence Diagram

Chapter 7

Detailed Design Document

7.1 Introduction

7.1.1 System Architecture

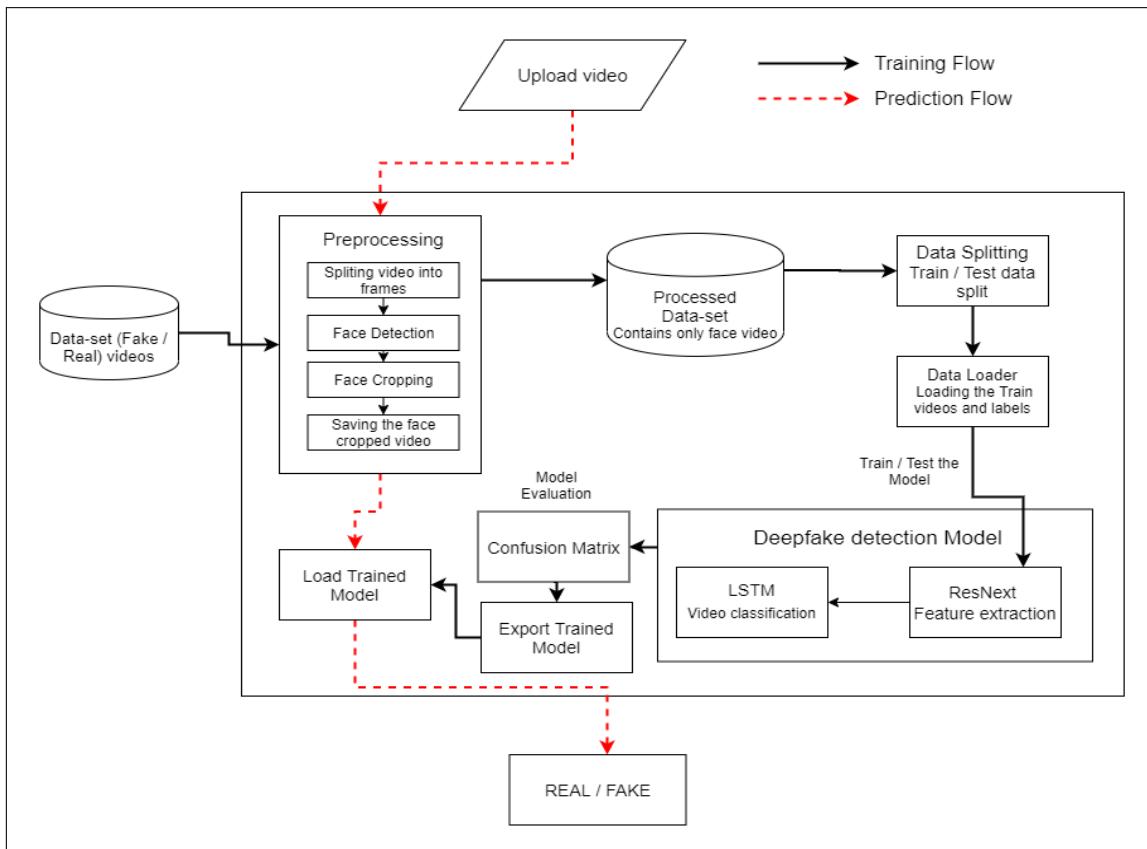


Figure 7.1: System Architecture

In this system, we have trained our PyTorch deepfake detection model on equal number of real and fake videos in order to avoid the bias in the model. The system architecture of the model is showed in the figure. In the development phase, we 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.

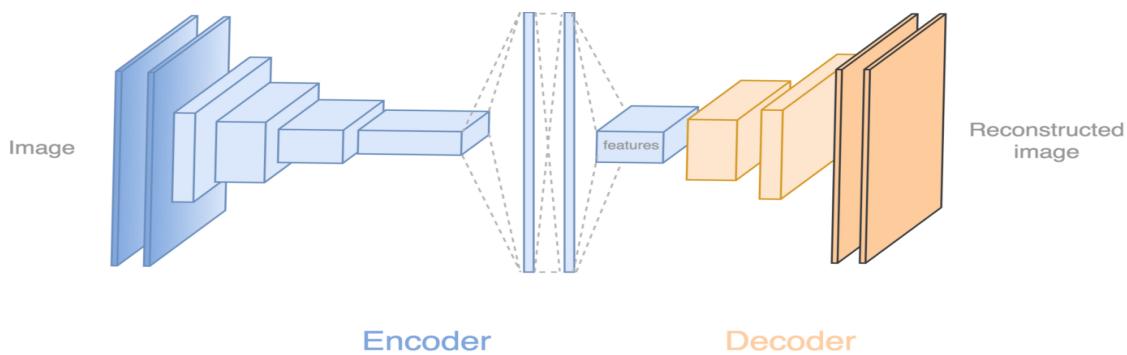


Figure 7.2: Deepfake generation



Figure 7.3: Face Swapped deepfake generation

Tools for deep fake creation.

1. Faceswap
2. Faceit
3. Deep Face Lab
4. Deepfake Capsule GAN
5. Large resolution face masked

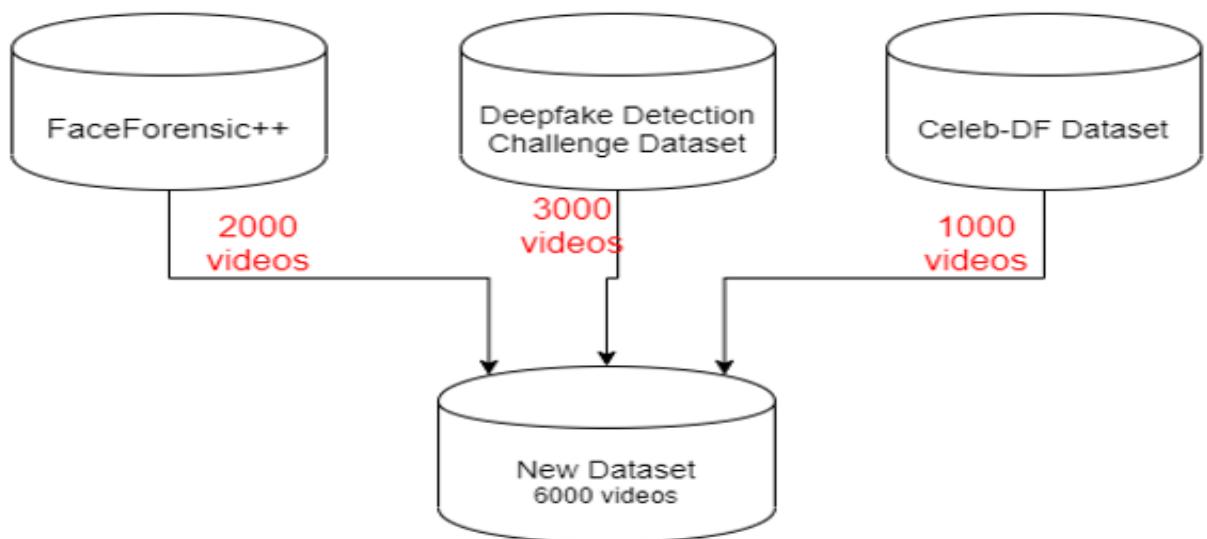
7.2 Architectural Design

7.2.1 Module 1 : Data-set 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 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 7.4: Dataset**

7.2.2 Module 2 : Pre-processing

In this step, the videos are preprocessed and all the unrequired and noise is removed from videos. Only the required portion of the video i.e face is detected and cropped.

The first steps in the preprocessing of the video is to split the video into frames. After splitting the video into frames the face is detected in each of the frame and the frame is cropped along the face. Later the cropped frame is again converted to a new video by combining each frame of the video. The process is followed for each video which leads to creation of processed dataset containing face only videos. The frame that does not contain the face is ignored while preprocessing.

To maintain the uniformity of number of frames, we have selected a threshold value based on the mean of total frames count of each video. Another reason for selecting a threshold value is limited computation power. As a video of 10 second at 30 frames per second(fps) will have total 300 frames and it is computationally very difficult to process the 300 frames at a single time in the experimental environment. So, based on our Graphic Processing Unit (GPU) computational power in experimental environment we have selected 150 frames as the threshold value. While saving the frames to the new dataset we have only saved the first 150 frames of the video to the new video. To demonstrate the proper use of Long Short-Term Memory (LSTM) we have considered the frames in the sequential manner i.e. first 150 frames and not randomly. The newly created video is saved at frame rate of 30 fps and resolution of 112 x 112.

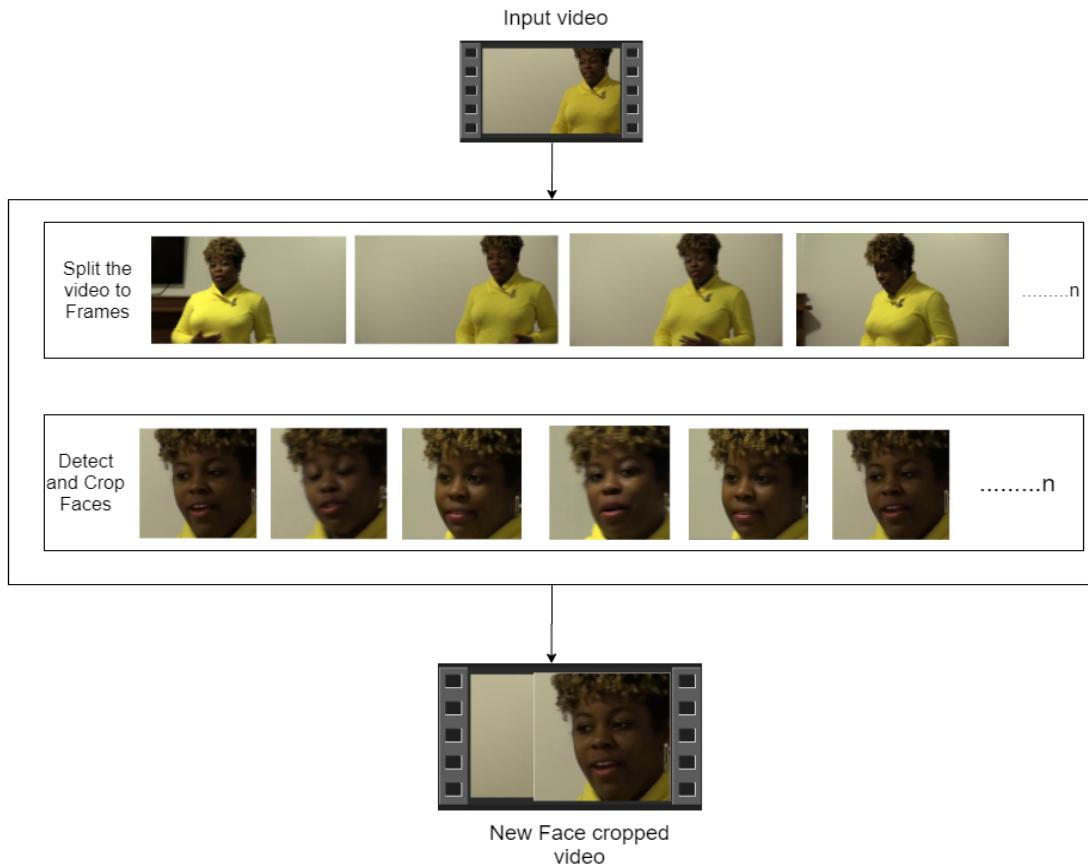
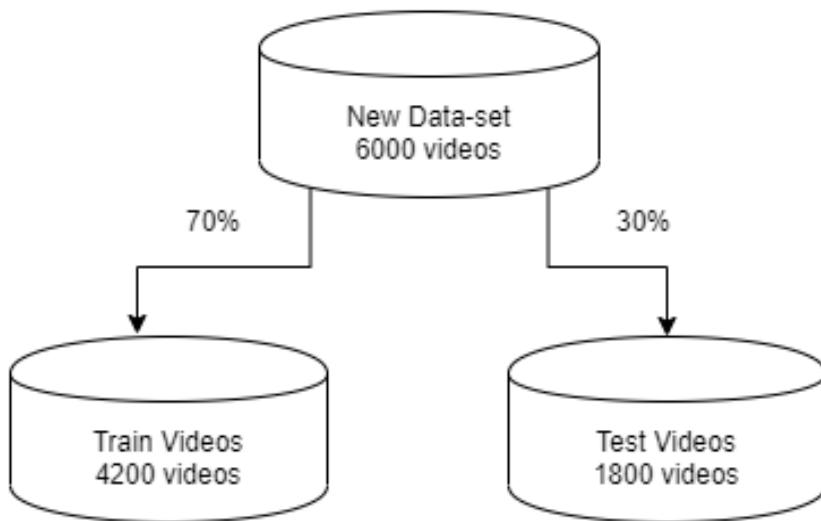


Figure 7.5: Pre-processing of video

7.2.3 Module 3: Data-set split

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

**Figure 7.6: Train test split**

7.2.4 Module 4: Model Architecture

Our model is a combination of CNN and RNN. We have used the Pre-trained 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 target output size of the image of the form H x W. For sequential processing of the frames a Sequential Layer is used. The batch size of 4 is used to perform the batch training. A SoftMax layer is used to get the confidence of the model during predication.

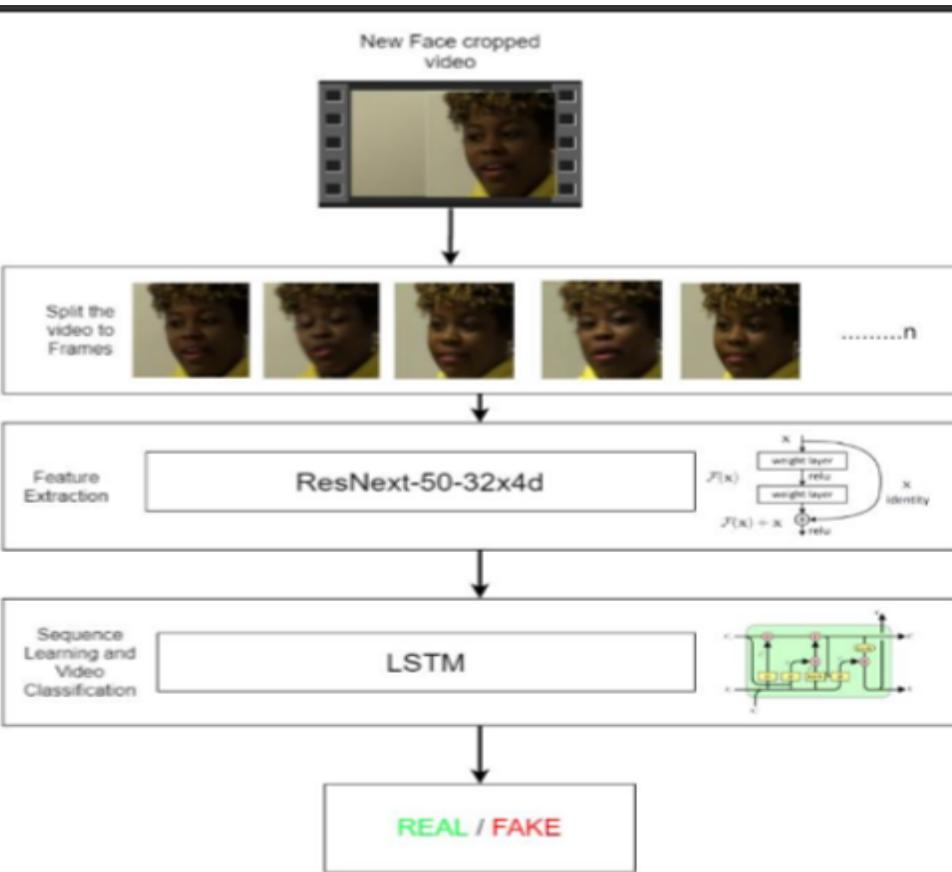


Figure 7.7: Overview of our model

7.2.5 Module 5: Hyper-parameter tuning

It is the process of choosing the perfect hyper-parameters for achieving the 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.

Chapter 8

Project Implementation

8.1 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 introduced 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.

8.2 Tools and Technologies Used

8.2.1 Planning

1. OpenProject

8.2.2 UML Tools

1. draw.io

8.2.3 Programming Languages

1. Python3
2. JavaScript

8.2.4 Programming Frameworks

1. PyTorch
2. Django

8.2.5 IDE

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

8.2.6 Versioning Control

1. Git

8.2.7 Cloud Services

1. Google Cloud Platform

8.2.8 Application and web servers:

1. Google Cloud Engine

8.2.9 Libraries

1. torch
2. torchvision
3. os
4. numpy
5. cv2
6. matplotlib
7. face_recognition
8. json
9. pandas
10. copy
11. glob
12. random
13. sklearn

8.3 Algorithm Details

8.3.1 Dataset Details

Refer 7.2.1

8.3.2 Preprocessing Details

- Using glob we imported all the videos in the directory in a python list.
- cv2.VideoCapture is used to read the videos and get the mean number of frames in each video.

- To maintain uniformity, based on mean a value 150 is selected as idea value for creating the new dataset.
- The video is split into frames and the frames are cropped on face location.
- The face cropped frames are again written to new video using VideoWriter.
- The 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.

8.3.3 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 (32x4d)
conv1	112×112	$7 \times 7, 64, \text{stride } 2$
		$3 \times 3 \text{ max pool, stride } 2$
conv2	56×56	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128, C=32 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3	28×28	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256, C=32 \\ 1 \times 1, 512 \end{bmatrix} \times 4$
conv4	14×14	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512, C=32 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$
conv5	7×7	$\begin{bmatrix} 1 \times 1, 1024 \\ 3 \times 3, 1024, C=32 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	1×1	global average pool 1000-d fc, softmax
	# params.	25.0×10^6

Figure 8.1: ResNext Architecture

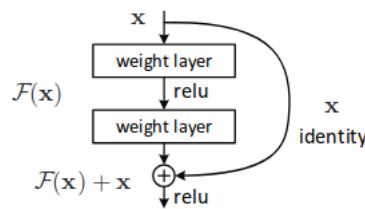


Figure 8.2: ResNext Working

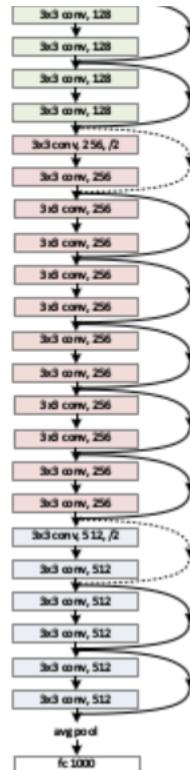


Figure 8.3: Overview of ResNext Architecture

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

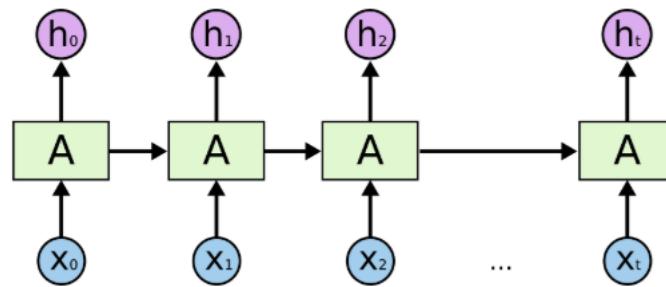


Figure 8.4: Overview of LSTM Architecture

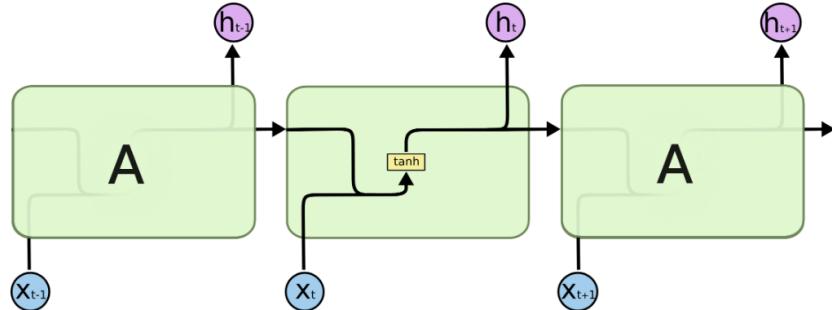


Figure 8.5: Internal LSTM Architecture

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

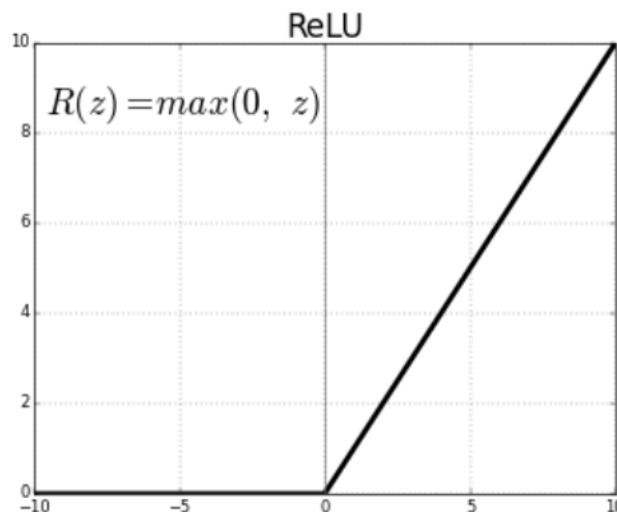


Figure 8.6: Relu Activation function

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

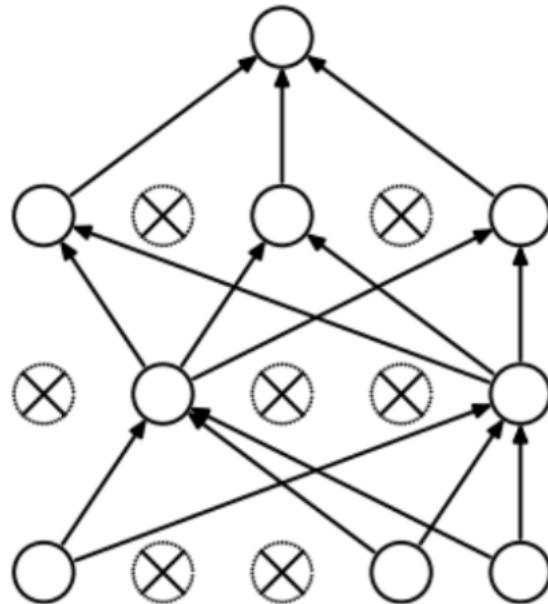


Figure 8.7: Dropout layer overview

- **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.

8.3.4 Model Training Details

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

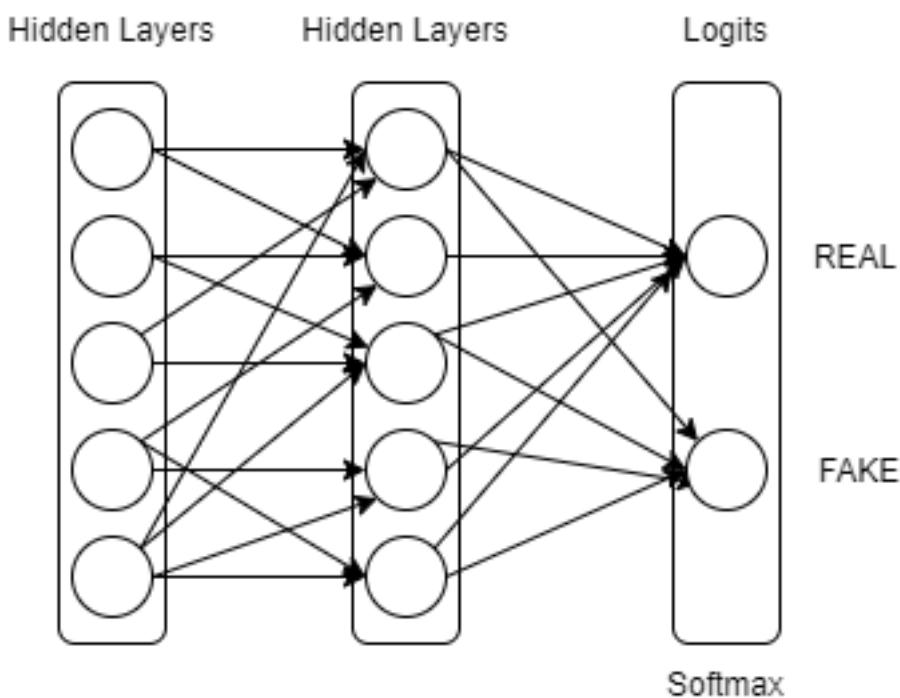


Figure 8.8: Softmax Layer

- **Confusion Matrix:** A confusion matrix is a summary of prediction results on a classification problem. The number of correct and incorrect predictions are summarized with count values and broken down by each class. This is the key to the confusion matrix. The confusion matrix shows the ways in which 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.

8.3.5 Model Prediction Details

- The model is loaded in the application
- The new video for prediction is preprocessed(refer 8.3.2, 7.2.2) and passed to the loaded model for prediction
- The trained model performs the prediction and return if the video is a real or fake along with the confidence of the prediction.

Chapter 9

Software Testing

9.1 Type of Testing Used

Functional Testing

1. Unit Testing
2. Integration Testing
3. System Testing
4. Interface Testing

Non-functional Testing

1. Performance Testing
2. Load Testing
3. Compatibility Testing

9.2 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

Chapter 10

Results and Discussion

10.1 Screen shots

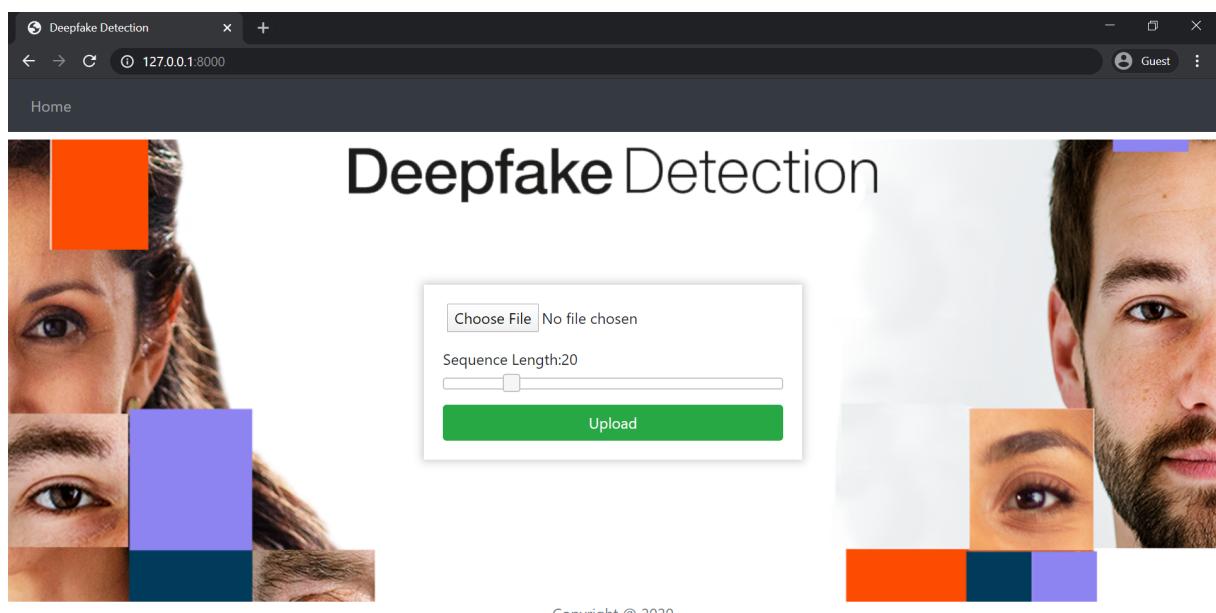


Figure 10.1: Home Page

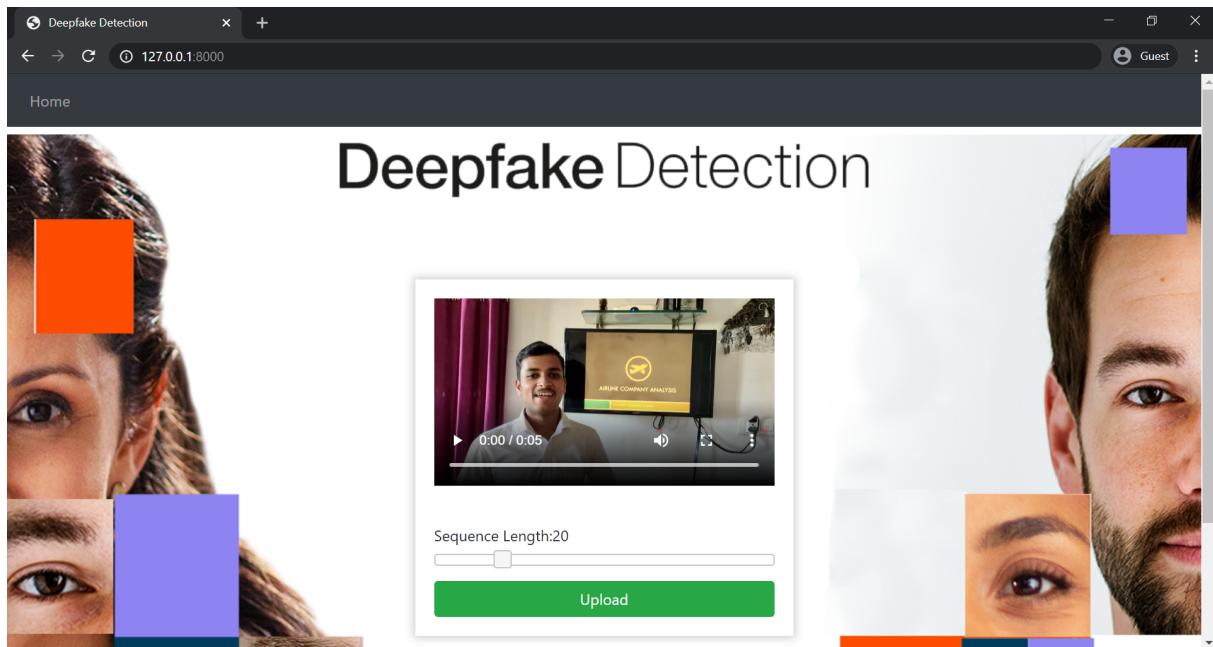


Figure 10.2: Uploading Real Video

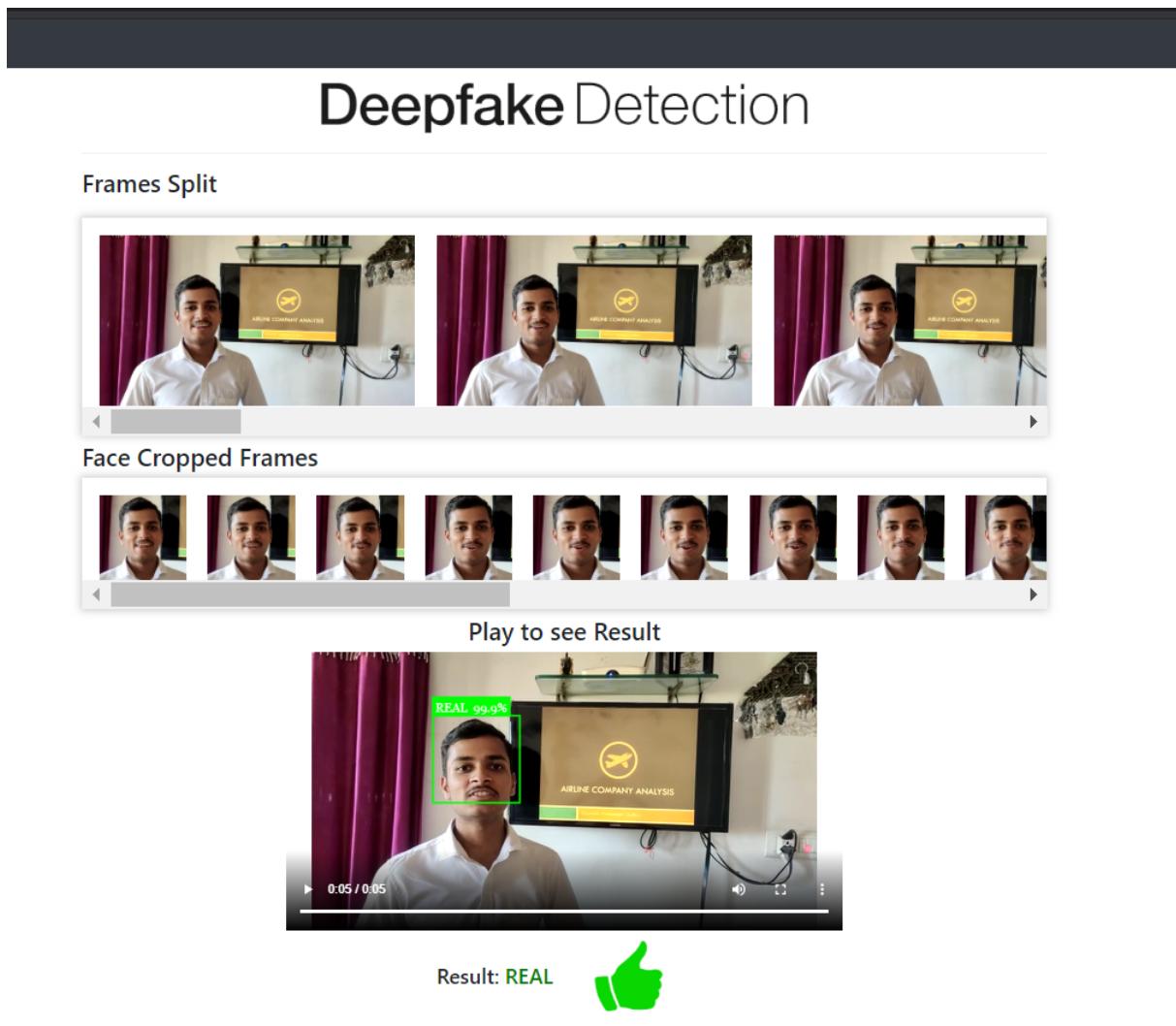


Figure 10.3: Real Video Output

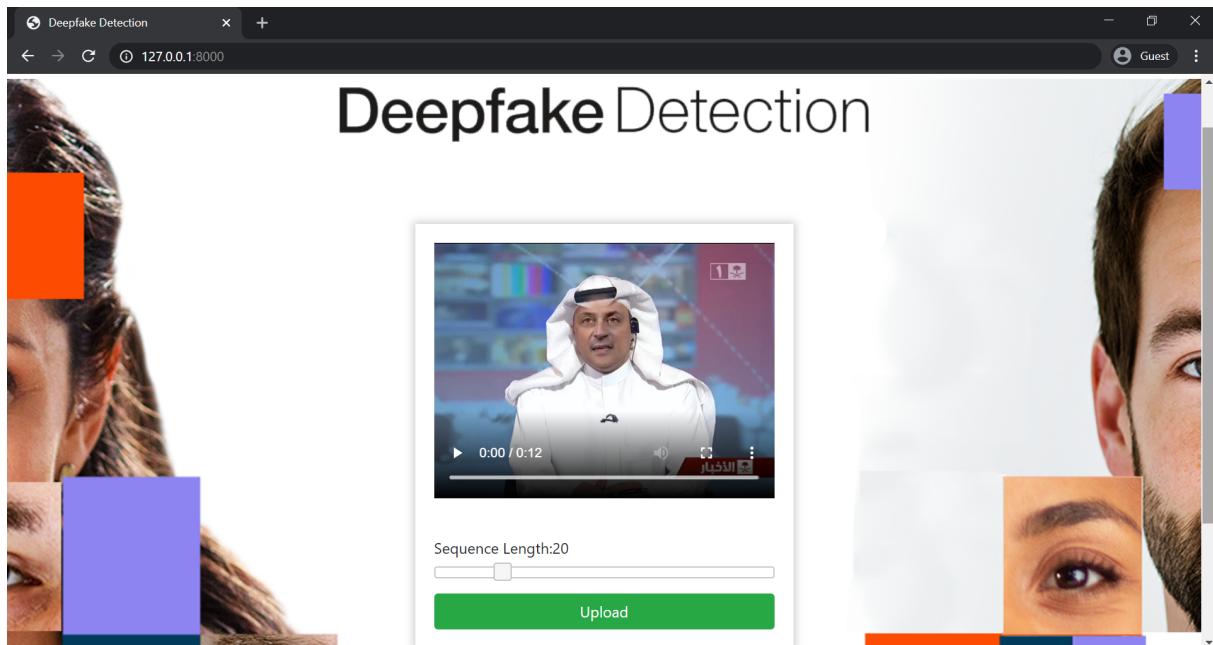


Figure 10.4: Uploading Fake Video

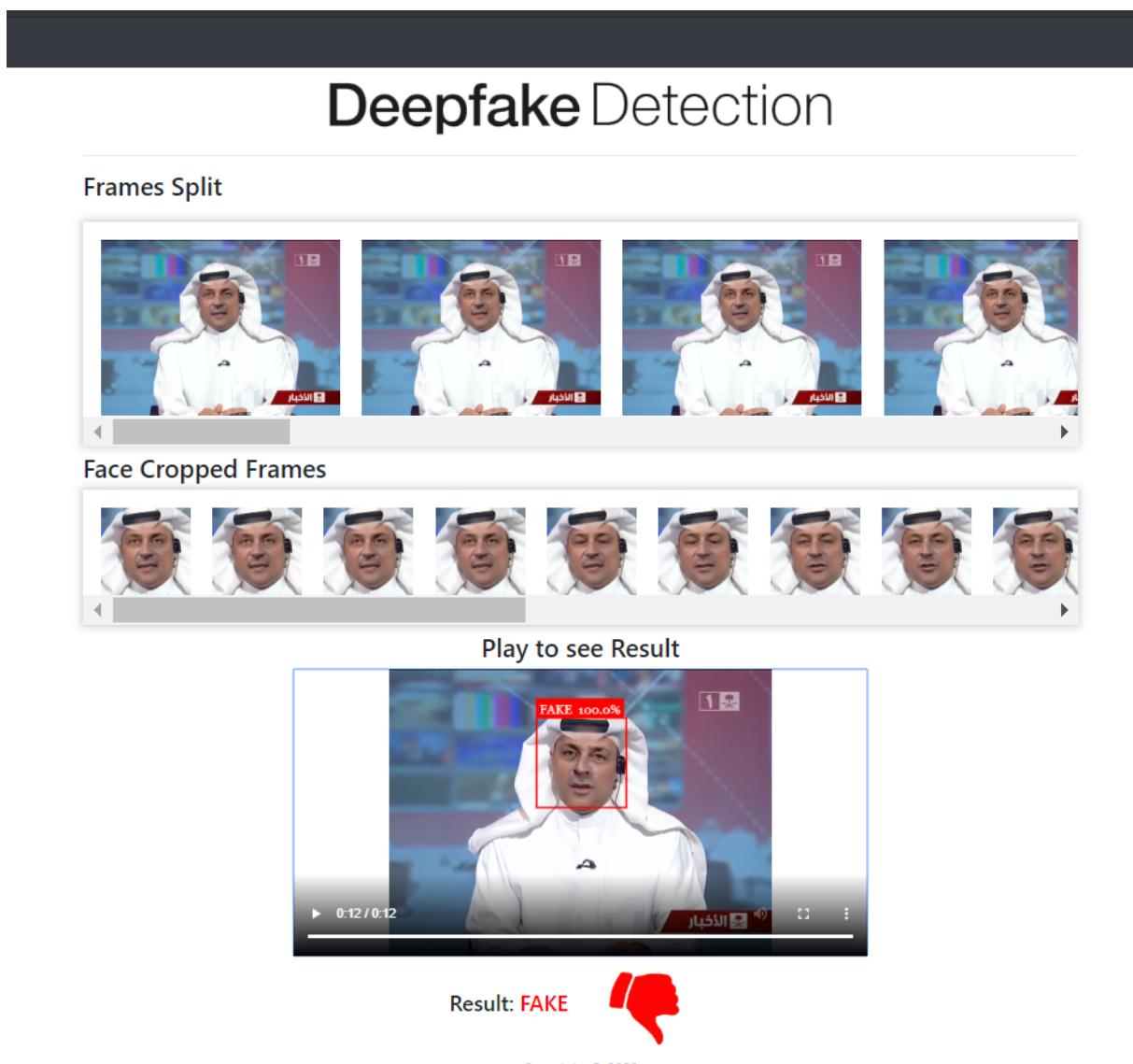


Figure 10.5: Fake video Output

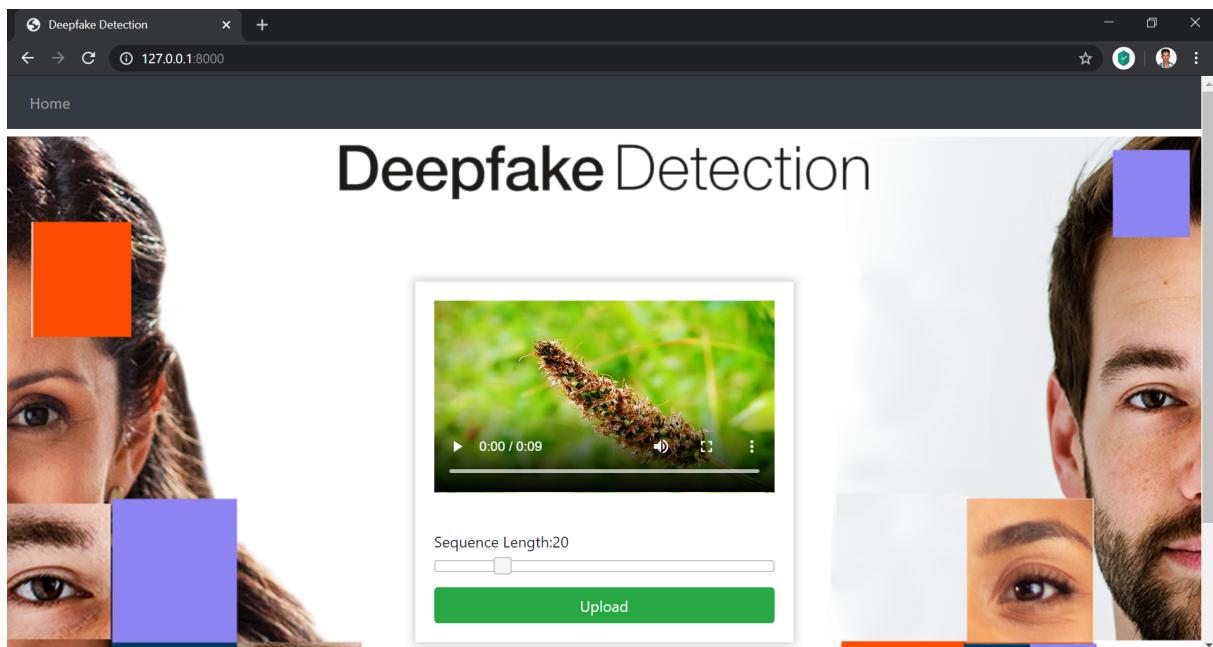


Figure 10.6: Uploading Video with no faces



Figure 10.7: Output of Uploaded video with no faces

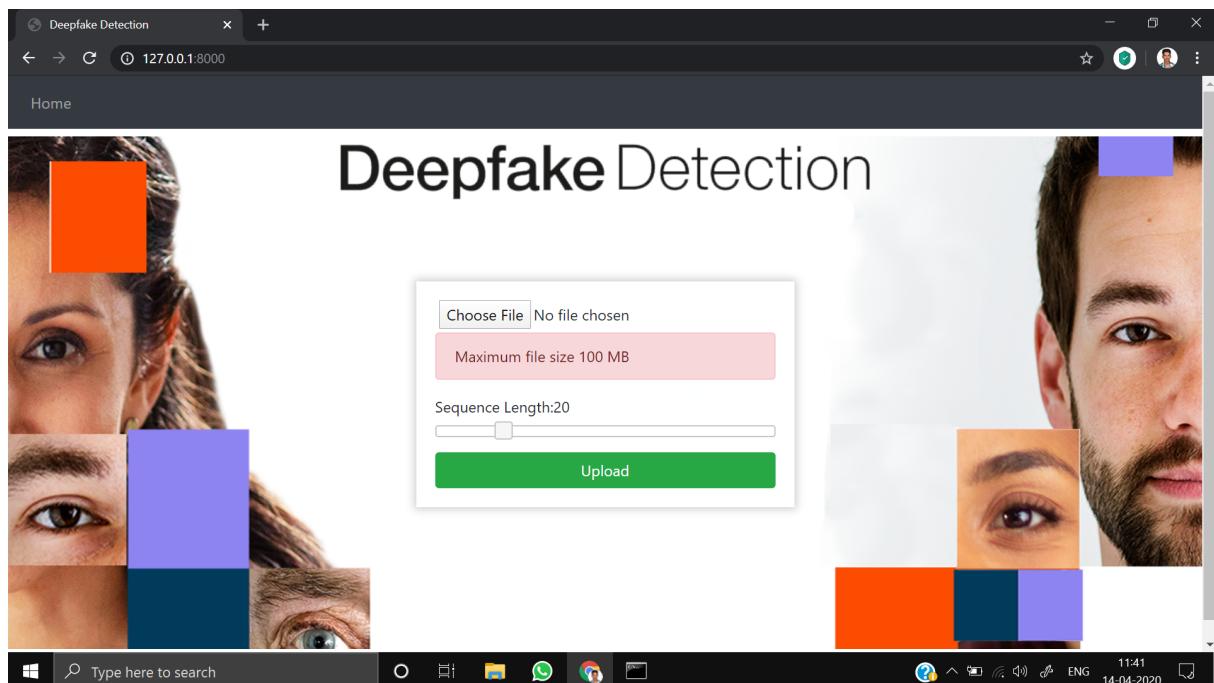


Figure 10.8: Uploading file greater than 100MB

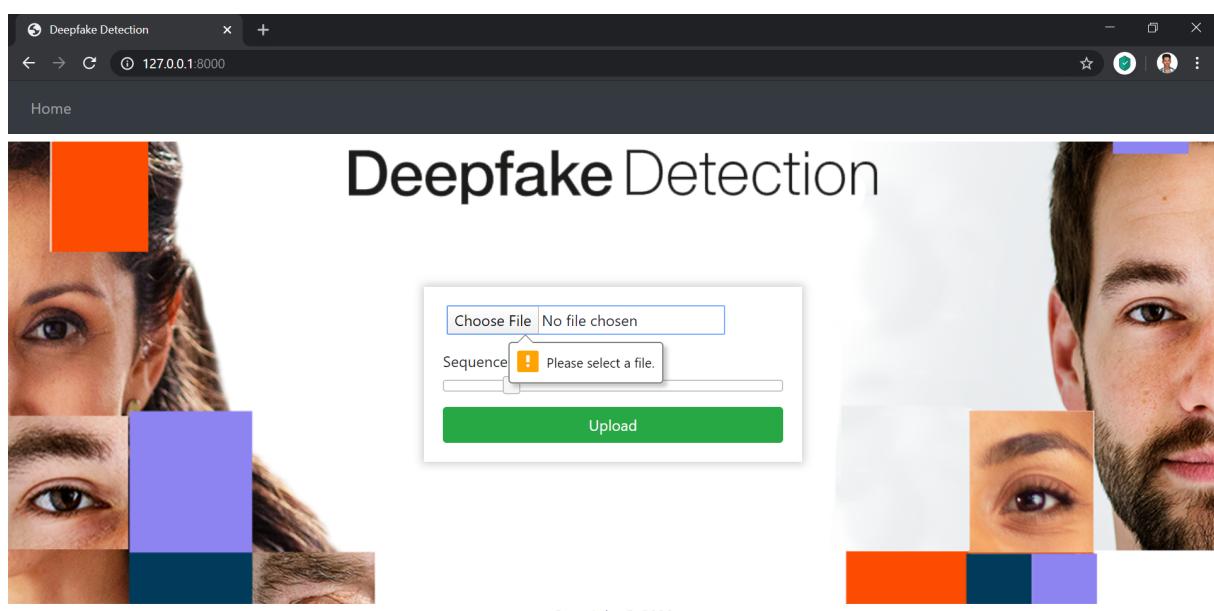


Figure 10.9: Pressing Upload button without selecting video

10.2 Outputs

10.2.1 Model results

Table 10.1: Trained Model Results

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

Chapter 11

Deployment and Maintenance

11.1 Deployment

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

1. clone the repository using the below command.
`git clone https://github.com/abhijitjadhav1998/Deefake_detection_Django_app.git`
Note: As its a private repository only authorized users will be able see the code and do the process of deployment
2. `pip install torch==1.4.0 torchvision==0.5.0 -f https://download.pytorch.org/whl/torch_stable.html`
3. `pip install -r requirement.txt`
4. `python manage.py migrate`
5. Copy all the trained models into models folder.
6. `python manage.py runserver 0.0.0.0:8000`

11.2 Maintenance

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

1. Stop the production server using `Ctrl + C`.
2. `git pull`
Note: As its a private repository only authorized users will be able see the code and do the process of deployment
3. `pip install -r requirement.txt`

4. `python manage.py migrate`
5. Copy all the trained models into models folder(optional: Do this if any new models are added).
6. `python manage.py runserver 0.0.0.0:8000`

Chapter 12

Conclusion and Future Scope

12.1 Conclusion

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

12.2 Future Scope

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

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

Appendix A

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 Artifacts,” 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 networks 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 Signal Based Surveillance (AVSS), Auckland, New Zealand, 2018, pp. 1-6.
- [19] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human 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 on 06 April 2020
- [23] <https://www.geeksforgeeks.org/software-engineering-cocomo-model/> Accessed on 15 April 2020
- [24] Deepfake Video Detection using Neural Networks
<http://www.ijsrd.com/articles/IJSRDV8I10860.pdf>
- [25] International Journal for Scientific Research and Development <http://ijsrd.com/>

Appendix B

Project Planner

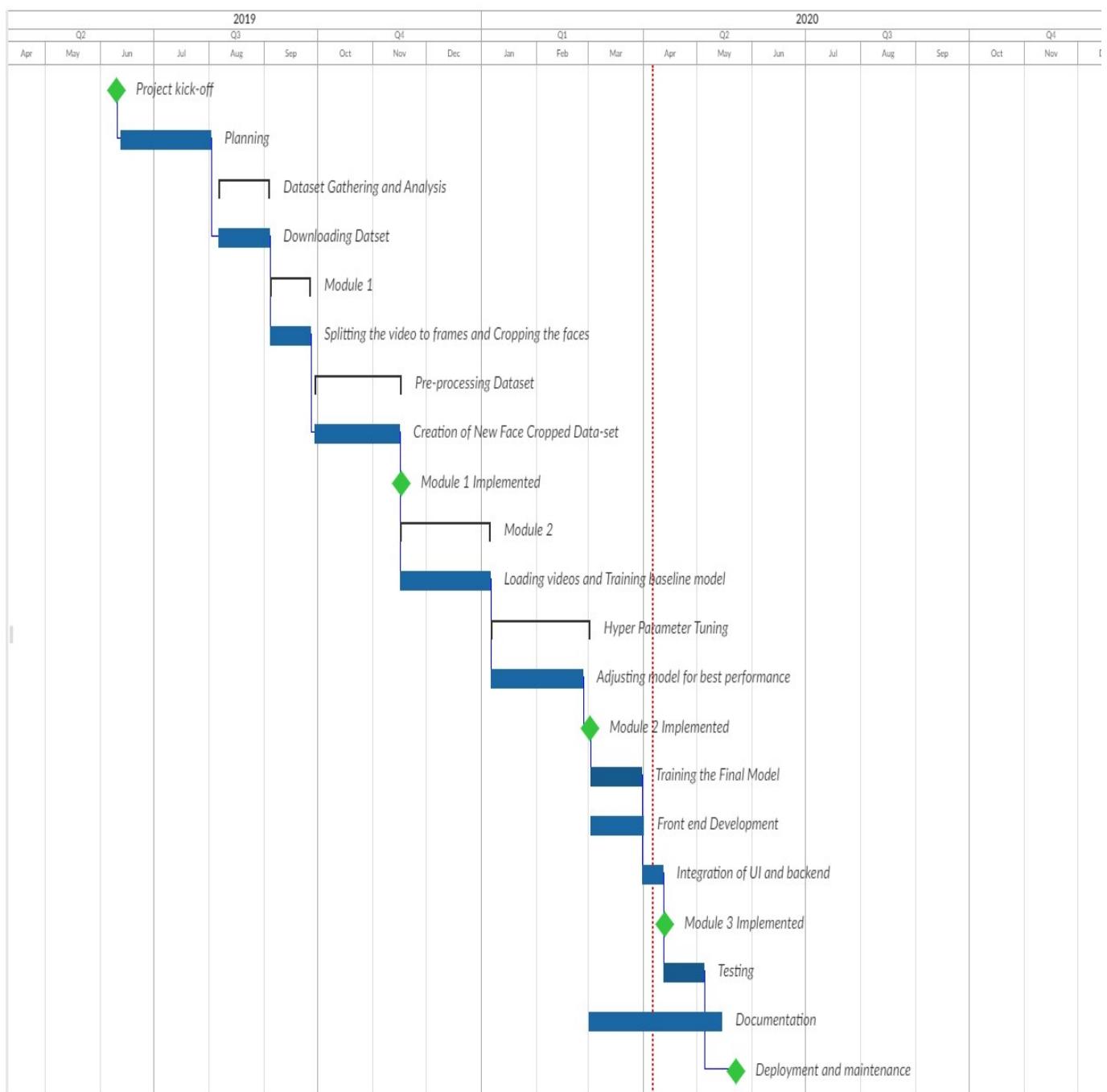
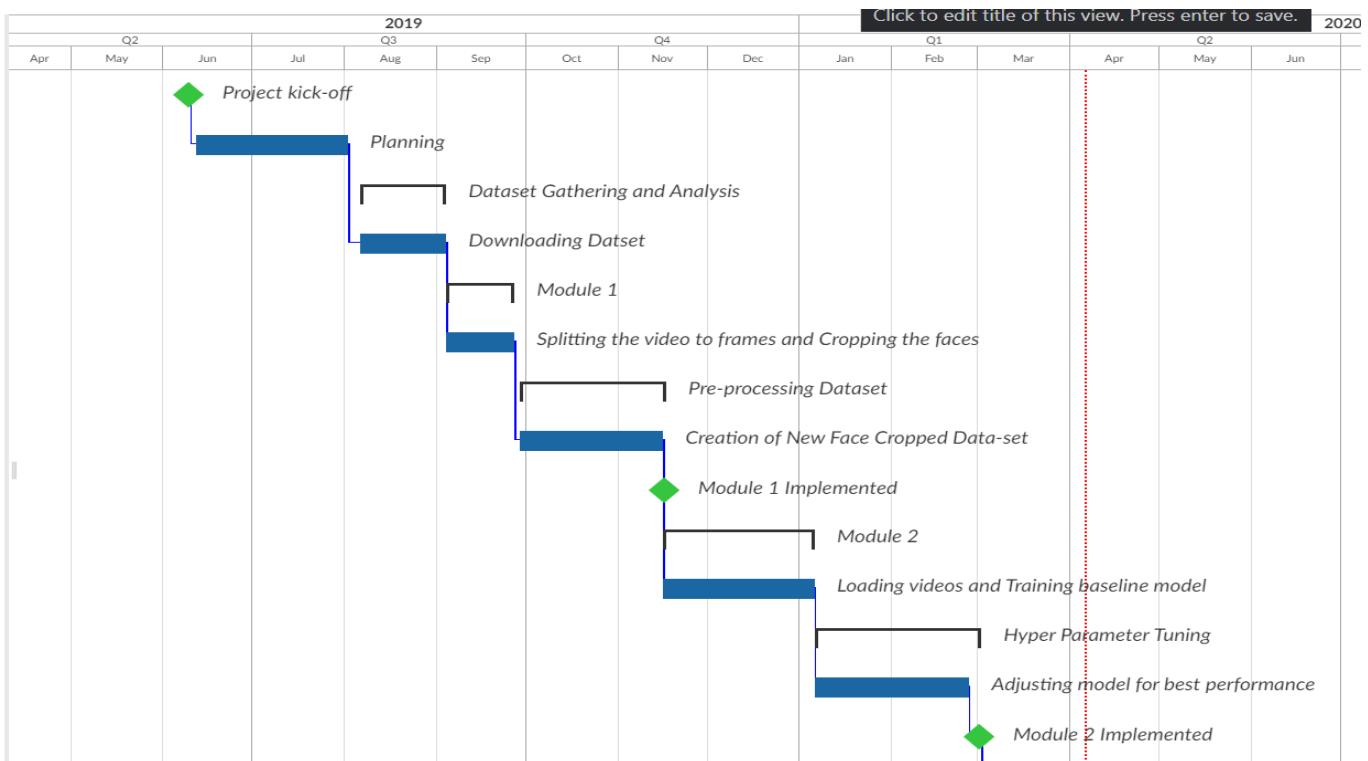
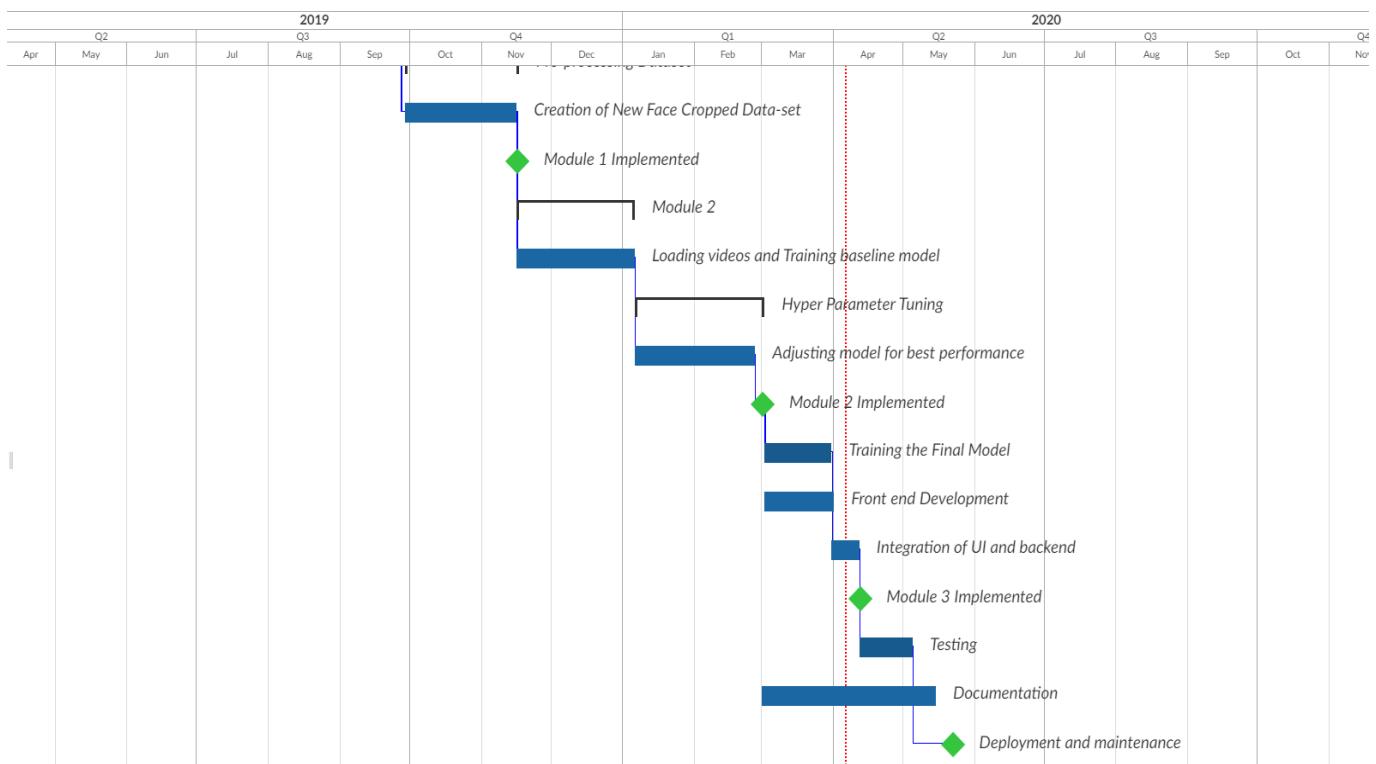


Figure B.1: Complete Project Plan

**Figure B.2: Project Plan 1****Figure B.3: Project Plan 2**

Appendix C

Paper Published Summary

1. Paper Title: Deepfake Video Detection using Neural Networks[24]
2. Name of the Journal : International Journal for Scientific Research and Development(IJSRD)[25].
3. Paper status : Accepted
4. Publication Details:
 - Paper ID: IJSRDV8I10860
 - Published in: Volume : 8, Issue : 1
 - Publication Date: 01/04/2020
 - Page(s): 1016-1019
 - Impact factor: 4.396
 - ISSN (online): 2321-0613