DEEPFAKE FACE MORPHING VIDEO DETECTION USING ResNext CNN & LSTM

A Main project thesis submitted in partial fulfillment of requirements for the award of degree for

VIII semester

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

By

P RAVI KUMAR REDDY (20131A05J5)

M VAMSI KRISHNA (20131A05F0)

N N V S S L SIRI VAISHNAVI (20131A05G0)

M V G AASRITHA (20131A05F3)

Under the esteemed guidance of

Ms. P. POOJA RATNAM,

Assistant Professor,

Department of Computer Science and Engineering

GAYATRI VIDYA PARISHAD COLLEGE OF ENGINEERING(AUTONOMOUS)

(Affiliated to JNTU-K, Kakinada)

VISAKHAPATNAM

2023-2024

GAYATRI VIDYA PARISHAD COLLEGE OF ENGINEERING

(Autonomous)

Approved by AICTE & Affiliated to Andhra University, Visakhapatnam from 2022-23
(Affiliated to JNTUK, Kakinada upto 2021-22)
Accredited twice by NAAC with 'A' Grade with a CGPA of 3.47/4.00
Madhurawada, Visakhapatnam - 530048

CERTIFICATE

This is to certify that the main project entitled "Deepfake Face Morphing Video Detection Using Resnext CNN & LSTM" being submitted by

P RAVI KUMAR REDDY	(20131A05J5)
M VAMSI KRISHNA	(20131A05F0)
N N V S S L SIRI VAISHNAVI	(20131A05G0)
M V G AASRITHA	(20131A05F3)

in partial fulfilment for the award of the degree "Bachelor of Technology" in Computer Science and Engineering to the Jawaharlal Nehru Technological University, Kakinada is a record of bonafide work done under my guidance and supervision during VIII semester of the academic year 2023-2024.

The results embodied in this record have not been submitted to any other university or institution for the award of any Degree or Diploma.

GUIDE	**HEAD OF THE DEPARTMENT**
Ms. P. Pooja Ratnam	Dr. D. N. D. Harini
Assistant Professor	Associate Professor & H.O.D
Department of CSE	Department of CSE
GVPCE(A)	GVPCE(A)

EXTERNAL SIGNATURE

DECLARATION

We hereby declare that this project entitled "**DEEPFAKE FACE MORPHING VIDEO DETECTION USING RESNEXT CNN & LSTM**" is a bonafide work done by us and submitted to "**Department of Computer Science and Engineering, G.V.P College of Engineering (Autonomous) Visakhapatnam**", in partial fulfilment for the award of the degree of B. Tech is of our own and it is not submitted to any other university or has been published anytime before.

PLACE: VISAKHAPATNAM

DATE:

P RAVI KUMAR REDDY (20131A05J5)

M VAMSI KRISHNA (20131A05F0)

N N V S S L SIRI VAISHNAVI (20131A05G0)

M V G AASRITHA (20131A05F3)

ACKNOWLEDGEMENT

We would like to express our deep sense of gratitude to our esteemed institute **Gayatri Vidya Parishad College of Engineering (Autonomous)**, which has provided us anopportunity to fulfill our cherished desire.

We express our sincere thanks to our principal **Dr. A. B. KOTESWARA RAO, Gayatri Vidya Parishad College of Engineering (Autonomous)** for his encouragement to us during this project, giving us a chance to explore and learn new technologies in the form of mini projects.

We express our deep sense of Gratitude to **Dr. D. N. D. HARINI**, Associate Professor and **Head of the Department of Computer Science and Engineering, Gayatri Vidya Parishad College of Engineering (Autonomous)** for giving us an opportunity to do the project in college.

We express our profound gratitude and our deep indebtedness to our guide **Ms. P. Pooja Ratnam**, **Assistant Professor, Department of Computer Science and Engineering**, whose valuable suggestions, guidance and comprehensive assessments helped us a lot in realizing our project.

We also thank our coordinator, **Dr. CH. SITA KUMARI**, **Associate Professor, Department of Computer Science and Engineering**, for the kind suggestions and guidance for the successful completion of our project work.

P RAVI KUMAR REDDY (20131A05J5)
M VAMSI KRISHNA (20131A05F0)
N N V S S L SIRI VAISHNAVI (20131A05G0)
M V G AASRITHA (20131A05F3)

ABSTRACT

The growing computation power has made the deep learning algorithms so powerful that creating a indistinguishable human synthesized video popularly called as deepfake 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)

INDEX

CHAPTER 1

INTRODUCTION

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

1.3 Literature Survey

A. 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.

B. 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 deep- fakes like teeth enchantment, wrinkles on faces, wrong placement of eyebrows etc.

C. 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.

D. 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 out model on large number of Real time data.

E. 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 2

Problem Definition and Scope

2.1 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 deepfakes).Creating the deepfakes using the Artificially intelligent tools are simple task. But, when it comes to detection of these deepfakes, 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.

2.1.1 Goals and objectives

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

2.1.2 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.

2.2 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.

2.3 Methodologies of Problem solving

2.3.1 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 1.3. After checking the feasibility of the problem statement. The next step is the dataset 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 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

2.3.2 Design

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

2.3.3 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.

2.3.4 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.

2.4 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.

2.5 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.

2.6 Hardware Resources Required

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

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

Sr. No.	Parameter	Minimum Requirement
1	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)

Table 2.1: Hardware Requirements

2.7 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 3

Project Plan

3.1 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.

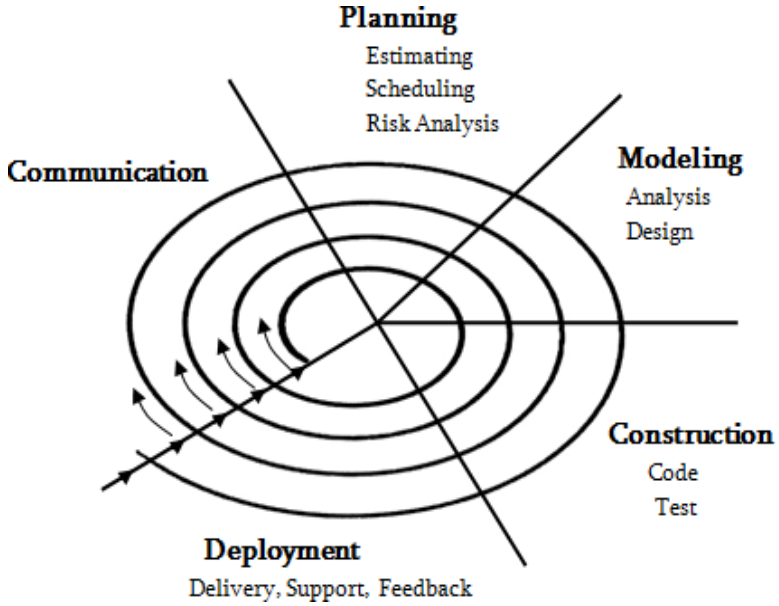


Figure 3.1: Spiral Methodology SDLC

3.2 Project Schedule

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 Data Loader 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.

CHAPTER 4

System Design

4.1 Introduction

4.1.1 Purpose and Scope of Document

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

4.1.2 Use Case View

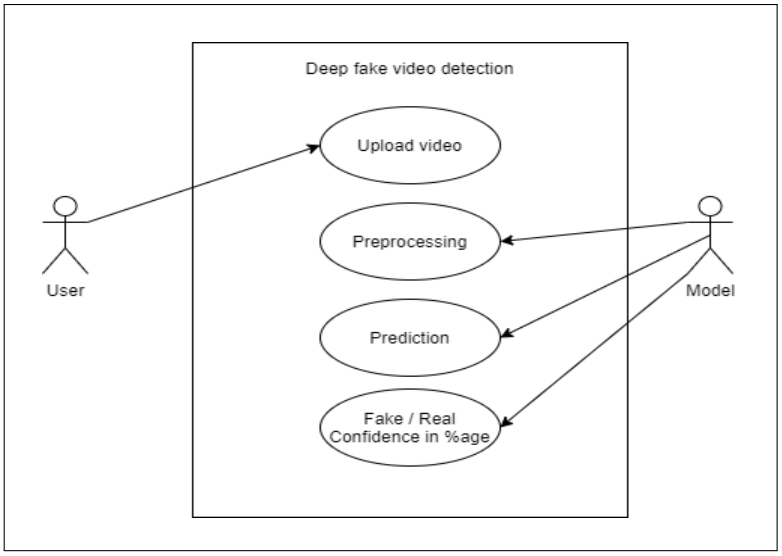


Figure 4.1: Use case diagram

4.2 Functional Model and Description

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

4.2.1 Data Flow Diagram

A. DFD Level-0:

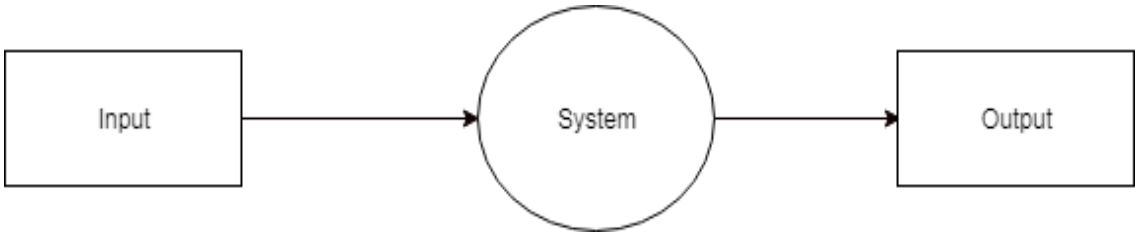


Figure 4.2: DFD Level 0

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

- 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.

B. 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.

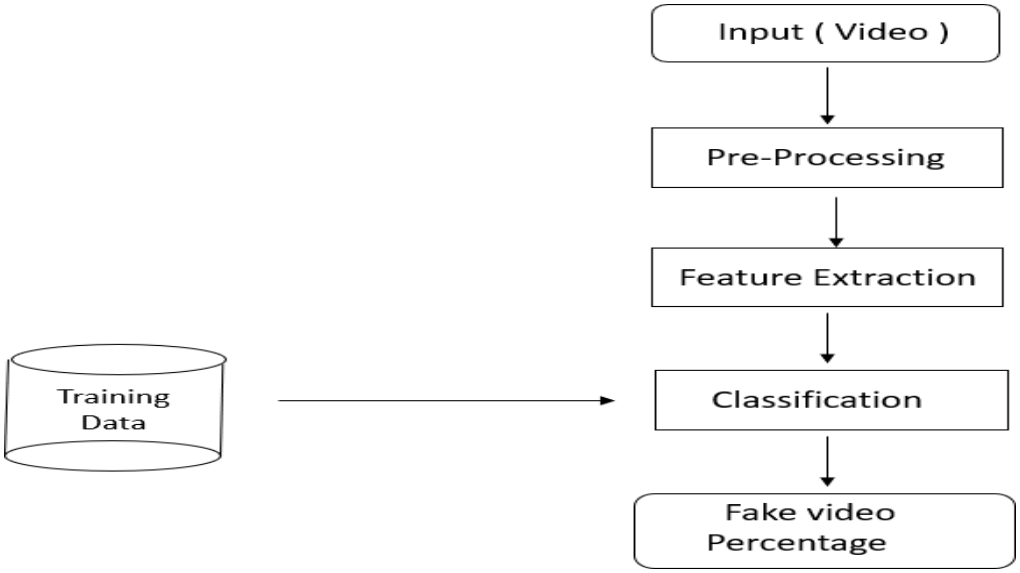


Figure 4.3: DFD Level 1

C. DFD Level-2: DFD level-2 enhances the functionality used by user etc.

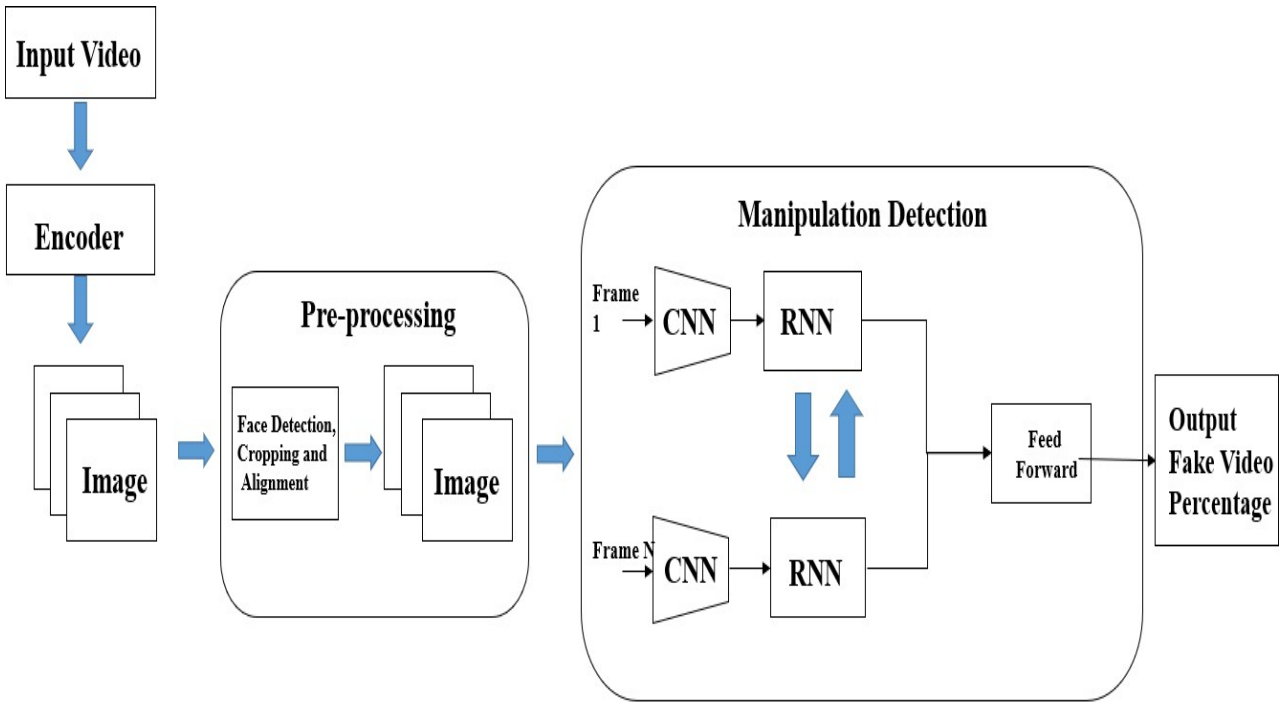


Figure 4.4: DFD Level 2

4.2.2 Activity Diagram:

Training Workflow:

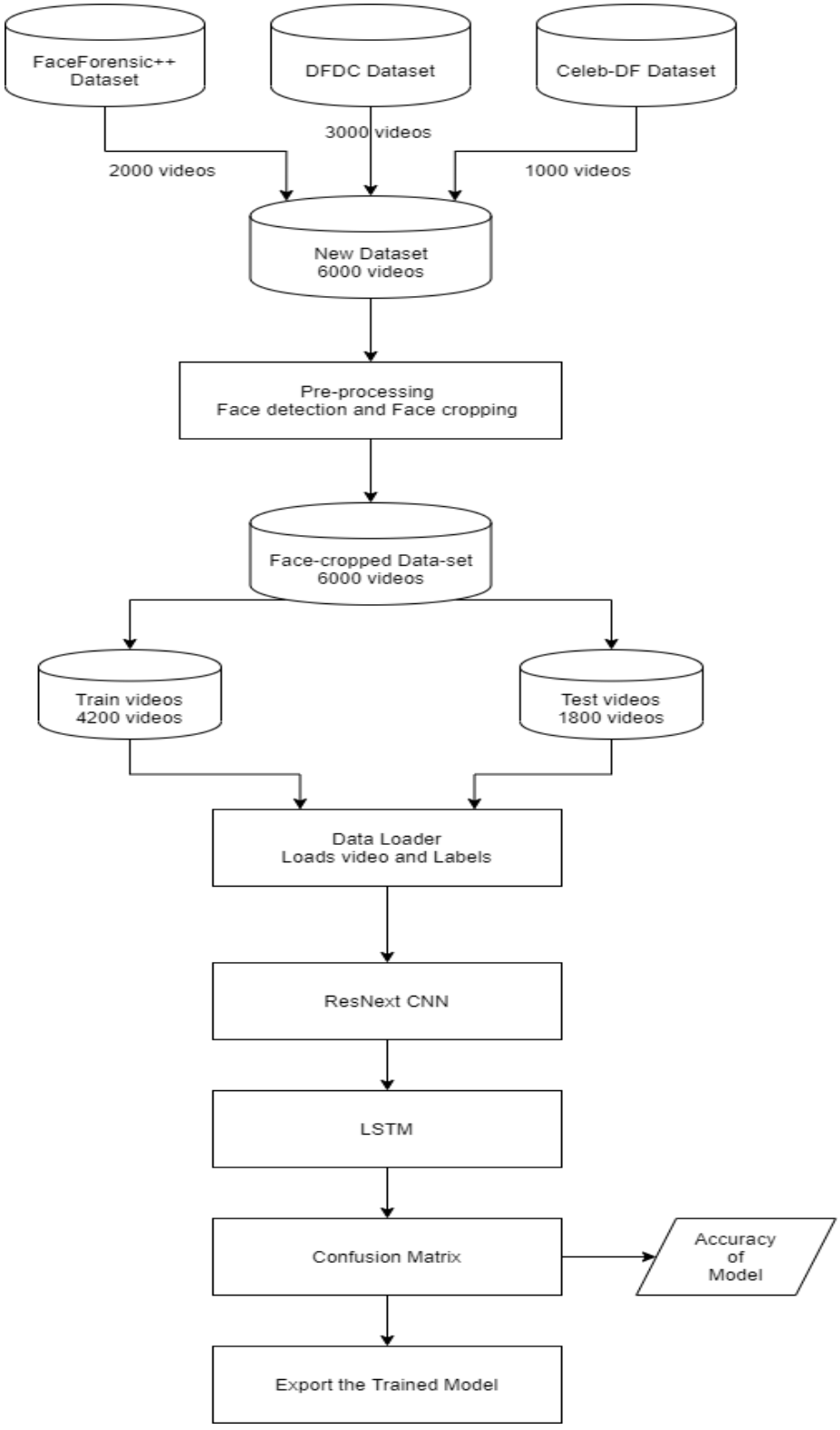


Figure 4.5: Training Workflow

Testing Workflow:

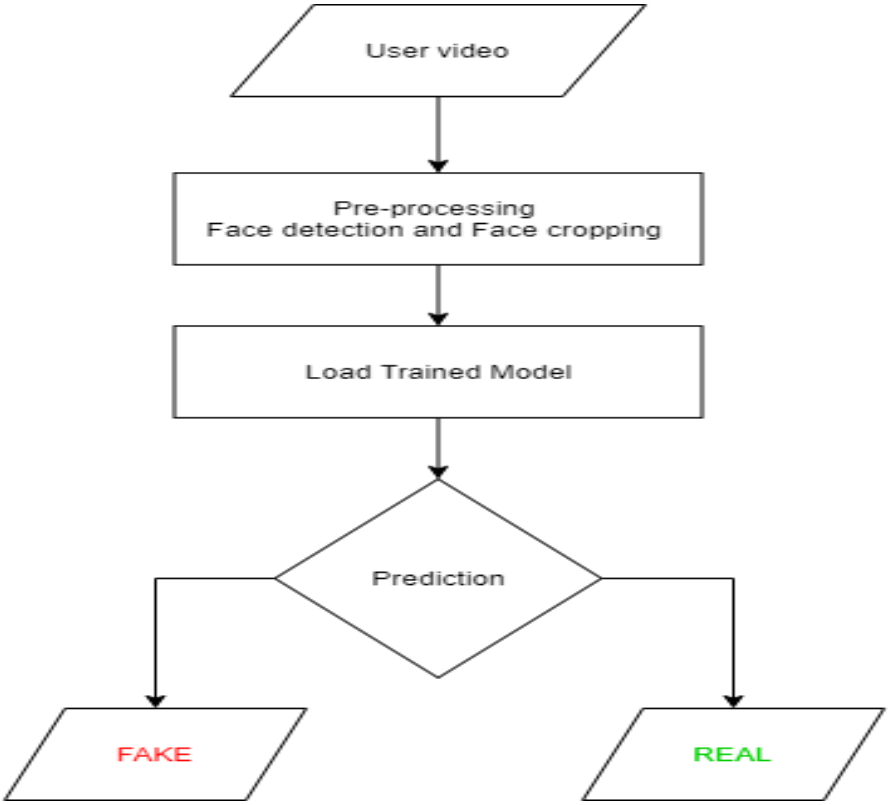


Figure 4.6: Testing Workflow

4.2.3 Sequence Diagram

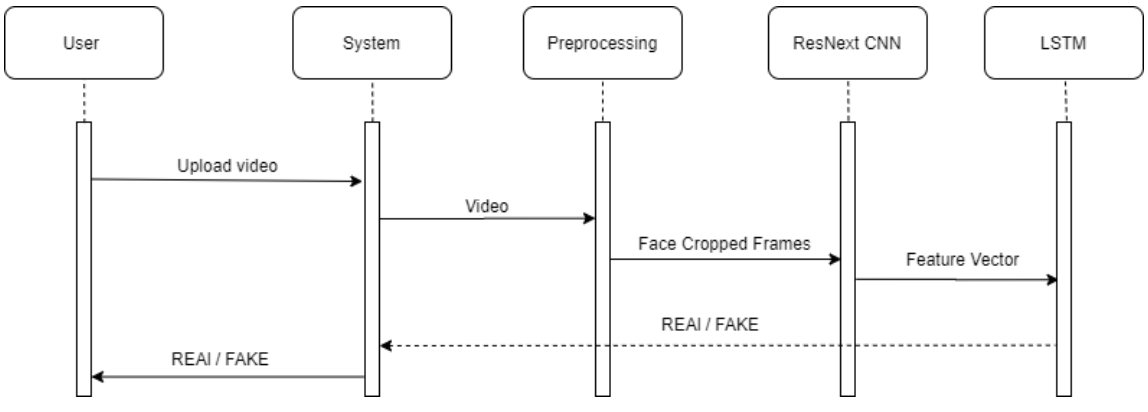


Figure 4.7: Sequence Diagram

CHAPTER 5

Project Implementation

5.1 Introduction

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

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

5.1.1 System Architecture

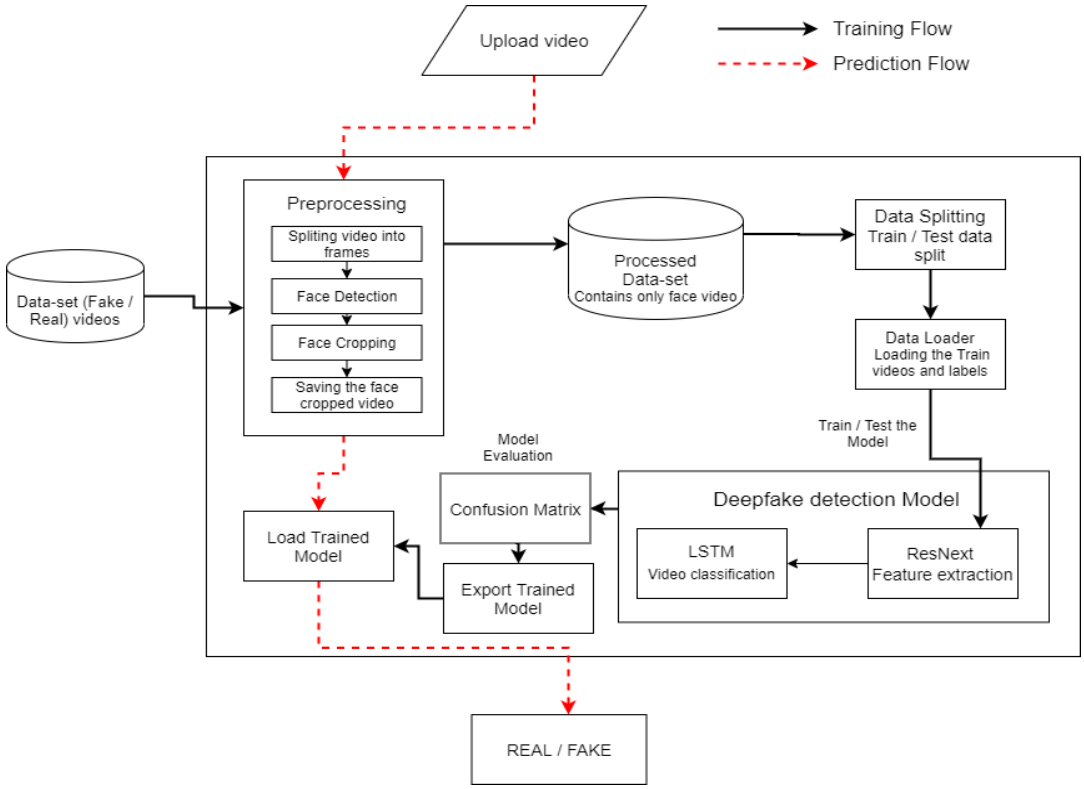


Figure 5.1: System Architecture

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

5.1.2 Creating DeepFake Video

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 by removing the left-over traces by the deepfake creation model. Which result in creation of a deepfake looks realistic in nature. We have also used the same approach to detect the deepfakes. Deepfakes created using the pretrained neural networks models are very realistic that it is almost impossible to spot the difference by the naked eyes. But in reality, the deepfakes creation tools leaves some of the traces or artifacts in the video which may not be noticeable by the naked eyes. The motive of this paper to identify these unnoticeable traces and distinguishable artifacts of these videos and classified it as deepfake or real video.

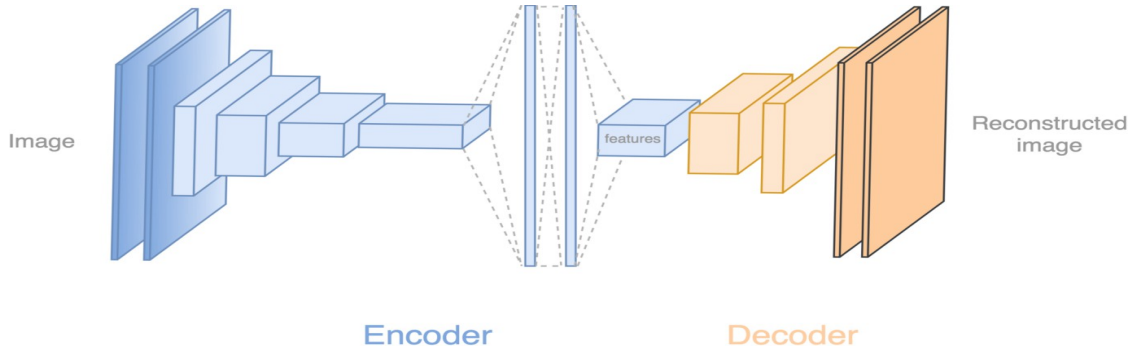


Figure 5.2: Deepfake generation

Figure 5.3: Face Swapped deepfake generation

5.1.3 About Model

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.		$\mathbf{25.0}\times10^6$

Figure 5.1: ResNext Architecture

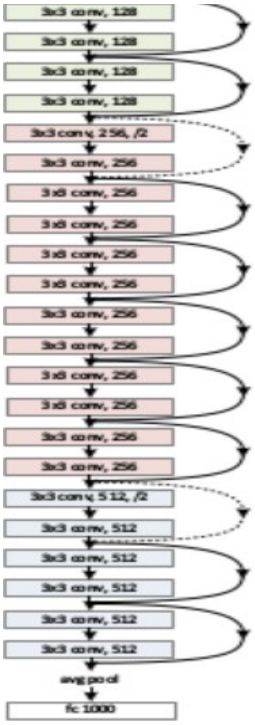


Figure 5.2: Overview of ResNext Architecture

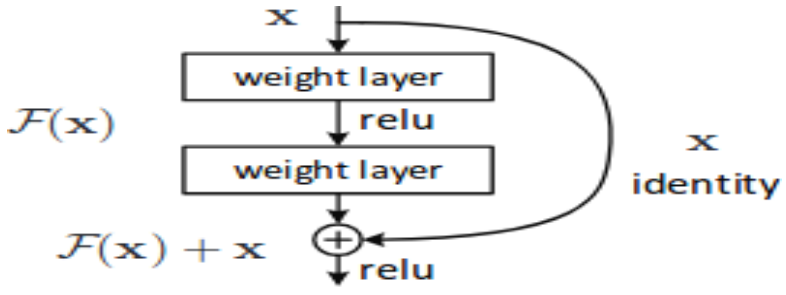


Figure 5.3: ResNext Working

- **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.

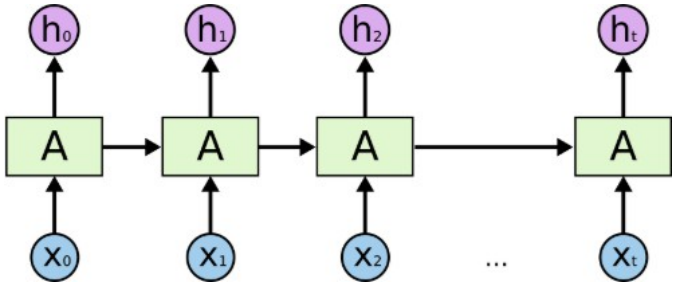


Figure 5.4: Overview of LSTM Architecture

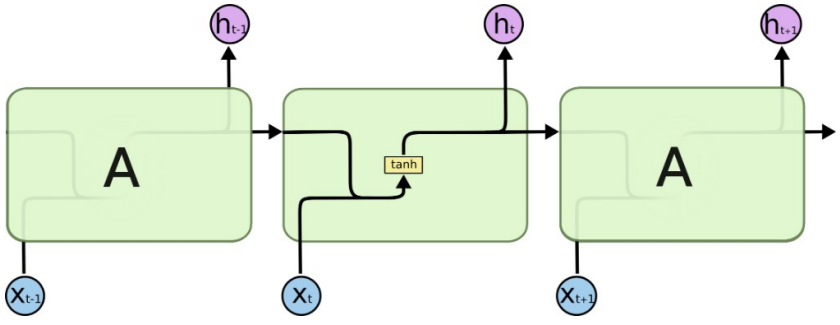


Figure 5.5: Internal LSTM Architecture

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

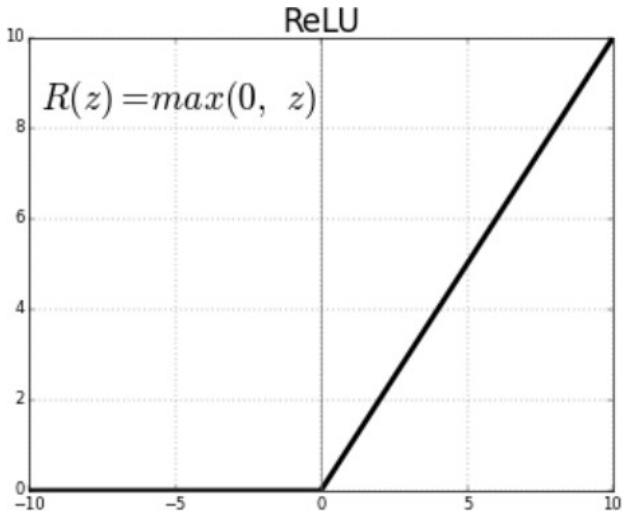


Figure 5.6: Relu Activation function

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

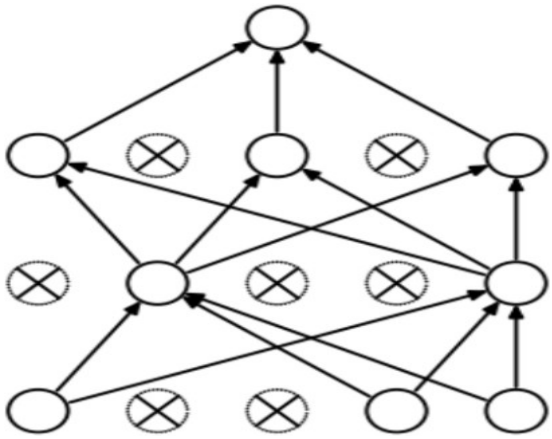


Figure 5.7: Dropout layer overview

- **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.

- **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.

- **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.

5.2 Architectural Design

5.2.1 Module 1: Data-set Gathering

For making the model efficient for real time prediction. We have gathered the data from different available data-sets like FaceForensic++(FF)[1], Deepfake detection challenge(DFDC)[2], and Celeb-DF[3]. 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.

Name of Dataset	Number of videos trained to model	
	Real	Fake
Deep fake detection challenge	1500	1500
FaceForensic++	1000	1000
Celeb-DF	500	500

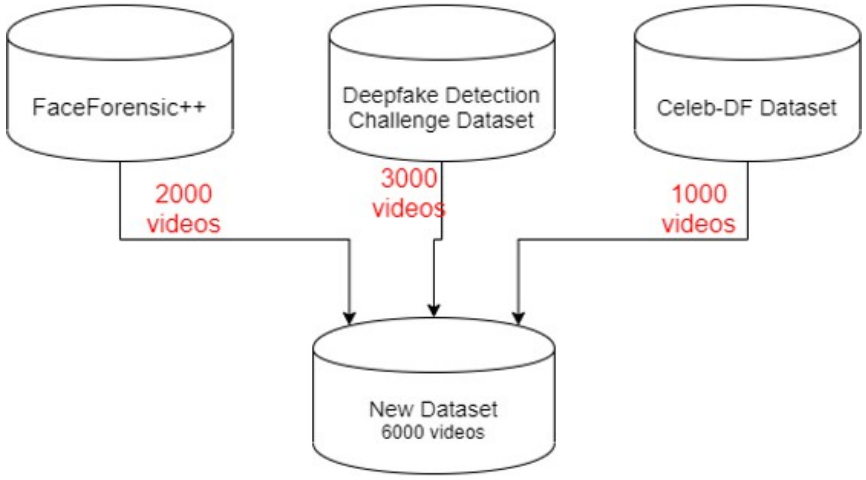


Figure 5.8: Dataset

5.2.2 Module 2 : Pre-processing

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

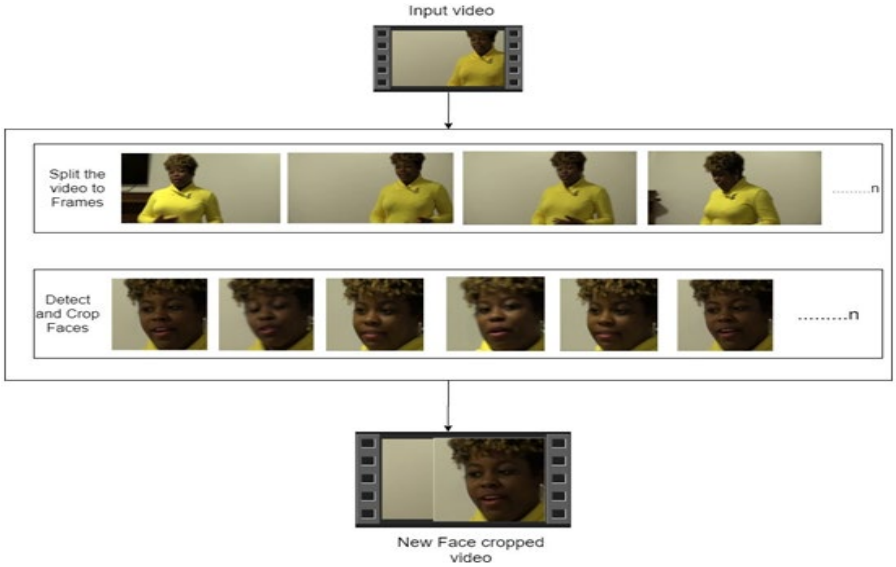


Figure 5.9: Pre-processing of video

5.2.3 Module 3: Data-set split

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

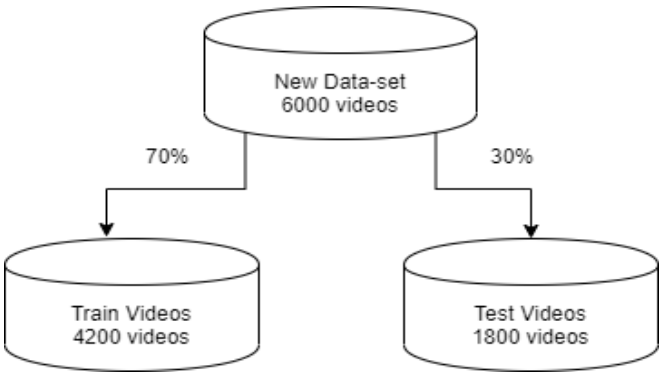


Figure 5.10: Train test split

5.2.4 Module 4: Model Architecture

Our model is a combination of CNN and RNN. We have used the Pre-trained 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, the labels of the videos from the training split are loaded and then fed 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 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 Soft Max layer is used to get the confidence of the model during predication.

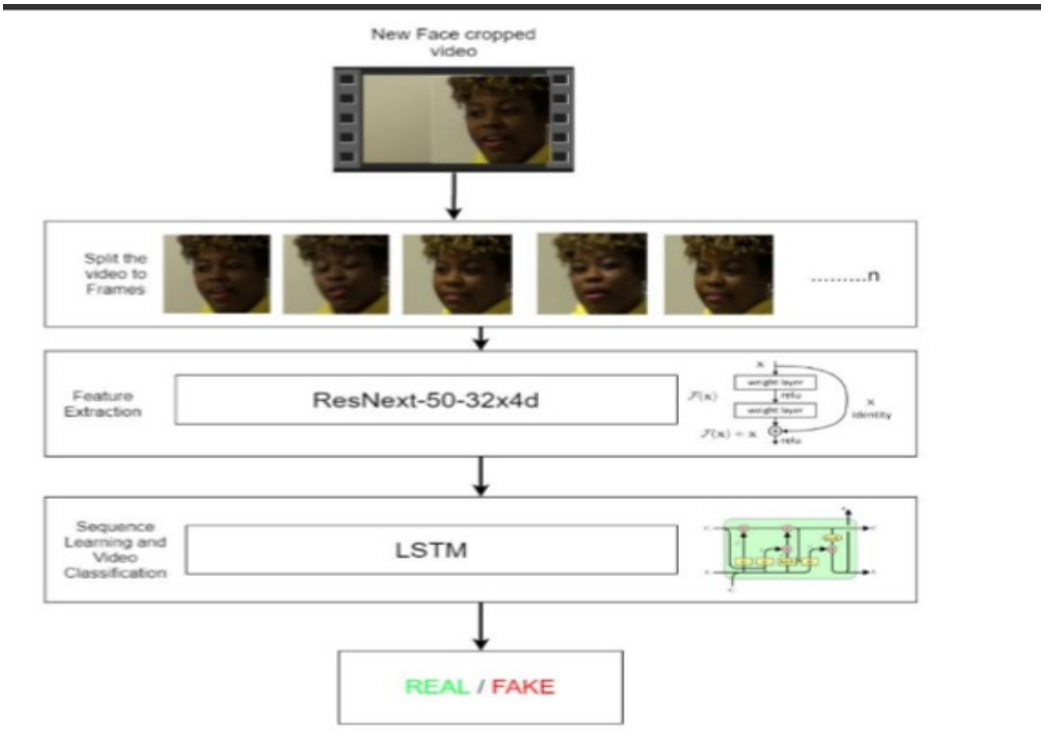


Figure 5.11: Overview of our model

5.2.5 Module 5: Hyper-parameter tuning

It is the process of choosing the perfect hyper-parameters for achieving the maximum accuracy. After re-iterating 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(0.001).

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.

5.3 MODEL EXECUTION

The execution of the model was done in Google Colab.

1. **Loading the Data Set From Google Drive**:

```
from google.colab import drive
drive.mount('/content/drive')
```

output: Mounted at /content/drive

2. **Checking the Number Of Frames in Each Video**:

```
import json
import glob
import numpy as np
import cv2
import copy
#change the path accordingly
video_files =  glob.glob('/content/drive/MyDrive/sampleDataset/*.mp4')
#video_files1 =  glob.glob('/content/dfdc_train_part_0/*.mp4')
#video_files += video_files1
frame_count = []
for video_file in video_files:
  cap = cv2.VideoCapture(video_file)
  if(int(cap.get(cv2.CAP_PROP_FRAME_COUNT))<150):
    video_files.remove(video_file)
    continue
  frame_count.append(int(cap.get(cv2.CAP_PROP_FRAME_COUNT)))
print("frames" , frame_count)
print("Total number of videos: " , len(frame_count))
print('Average frame per video:',np.mean(frame_count))
```

output: frames [300, 299, 300, 299, 300, 299, 300, 299, 300, 299]

Total number of videos: 10

Average frame per video: 299.5

3. **CROPING THE FACE ONLY AREA FROM EACH VIDEO**

```
# to extract frame
def frame_extract(path):
  vidObj = cv2.VideoCapture(path)
  success = 1
  while success:
      success, image = vidObj.read()
```

```
    if success:
      yield image
!pip3 install face_recognition
!mkdir '/content/drive/My Drive/FF_REAL_Face_only_data'
import torch
import torchvision
from torchvision import transforms
from torch.utils.data import DataLoader
from torch.utils.data.dataset import Dataset
import os
import numpy as np
import cv2
import matplotlib.pyplot as plt
import face_recognition
from tqdm.autonotebook import tqdm
# process the frames
def create_face_videos(path_list,out_dir):
 already_present_count =  glob.glob(out_dir+'*.mp4')
 print("No of videos already present " , len(already_present_count))
 for path in tqdm(path_list):
  out_path = os.path.join(out_dir,path.split('/')[-1])
  file_exists = glob.glob(out_path)
  if(len(file_exists) != 0):
   print("File Already exists: " , out_path)
   continue
  frames = []
  flag = 0
  face_all = []
  frames1 = []
  out = cv2.VideoWriter(out_path,cv2.VideoWriter_fourcc('M','J','P','G'), 30, (112,112))
  for idx,frame in enumerate(frame_extract(path)):
   #if(idx % 3 == 0):
   if(idx <= 150):
    frames.append(frame)
    if(len(frames) == 4):
     faces = face_recognition.batch_face_locations(frames)
     for i,face in enumerate(faces):
      if(len(face) != 0):
       top,right,bottom,left = face[0]
      try:
       out.write(cv2.resize(frames[i][top:bottom,left:right,:],(112,112)))
      except:
       pass
     frames = []
  try:
   del top,right,bottom,left
  except: pass
  out.release()
```

4. STORING THE CROPED VIDEOS INTO ANOTHER FILE

create_face_videos(video_files,'/content/drive/MyDrive/FF_REAL_Face_only_data/')

5. Check If The Video Is Corrupted Or Not. If The Video Is Corrupted Delete The Video:

```
import glob
import torch
import torchvision
from torchvision import transforms
from torch.utils.data import DataLoader
from torch.utils.data.dataset import Dataset
import os
import numpy as np
import cv2
import matplotlib.pyplot as plt
import face_recognition
#Check if the file is corrupted or not
def validate_video(vid_path,train_transforms):
    transform = train_transforms
    count = 20
    video_path = vid_path
    frames = []
    a = int(100/count)
    first_frame = np.random.randint(0,a)
    temp_video = video_path.split('/')[-1]
    for i,frame in enumerate(frame_extract(video_path)):
      frames.append(transform(frame))
      if(len(frames) == count):
        break
    frames = torch.stack(frames)
    frames = frames[:count]
    return frames
#extract a from from video
def frame_extract(path):
  vidObj = cv2.VideoCapture(path)
  success = 1
  while success:
      success, image = vidObj.read()
      if success:
          yield image

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)])
video_fil =  glob.glob('/content/drive/MyDrive/FF_REAL_Face_only_data/*.mp4')
#video_fil =  glob.glob('/content/drive/My Drive/Celeb_fake_face_only/*.mp4')
#video_fil += glob.glob('/content/drive/My Drive/Celeb_real_face_only/*.mp4')
#video_fil += glob.glob('/content/drive/My Drive/DFDC_FAKE_Face_only_data/*.mp4')
#video_fil += glob.glob('/content/drive/My Drive/DFDC_REAL_Face_only_data/*.mp4')
#video_fil += glob.glob('/content/drive/My Drive/FF_Face_only_data/*.mp4')
print("Total no of videos :" , len(video_fil))
print(video_fil)
count = 0;
for i in video_fil:
  try:
    count+=1
    validate_video(i,train_transforms)
  except:
    print("Number of video processed: " , count ," Remaining : " , (len(video_fil) - count))
    print("Corrupted video is : " , i)
    continue
print((len(video_fil) - count))
```

output: Total no of videos : 10

```
['/content/drive/MyDrive/FF_REAL_Face_only_data/ercqmajdid.mp4',
       '/content/drive/MyDrive/FF_REAL_Face_only_data/blszgmxkvu.mp4',
       '/content/drive/MyDrive/FF_REAL_Face_only_data/eyguqfmgzh.mp4',
       '/content/drive/MyDrive/FF_REAL_Face_only_data/lnuwkizkiw.mp4',
       '/content/drive/MyDrive/FF_REAL_Face_only_data/hypozprqhm.mp4',
       '/content/drive/MyDrive/FF_REAL_Face_only_data/mkzaekkvej.mp4',
       '/content/drive/MyDrive/FF_REAL_Face_only_data/uubgqnvfdl.mp4',
       '/content/drive/MyDrive/FF_REAL_Face_only_data/nfsztvjqpk.mp4',
       '/content/drive/MyDrive/FF_REAL_Face_only_data/wixbuuzygv.mp4',
       '/content/drive/MyDrive/FF_REAL_Face_only_data/xrhqtmxlvx.mp4']
0
```

6.Function that acess Global meta dataset .csv file and label the video whether it is real or fake :

```
#count the number of fake and real videos
def number_of_real_and_fake_videos(data_list):
  header_list = ["file","label"]
  lab = pd.read_csv('/content/drive/My Drive/Gobal_metadata.csv',names=header_list)
  fake = 0
  real = 0
```

```
 for i in data_list:
   temp_video = i.split('/')[-1]
   label = lab.iloc[(labels.loc[labels["file"] == temp_video].index.values[0]),1]
   if(label == 'FAKE'):
     fake+=1
   if(label == 'REAL'):
     real+=1
 return real,fake
```

7.Splitting the videos into Training and validation section ,also checking the cropped video

```
        # load the labels and video in data loader
import random
import pandas as pd
from sklearn.model_selection import train_test_split

header_list = ["file","label"]
labels = pd.read_csv('/content/drive/My Drive/Gobal_metadata.csv',names=header_list)
#print(labels)
train_videos = video_files[:int(0.8*len(video_files))]
valid_videos = video_files[int(0.8*len(video_files)):]
print("train : " , len(train_videos))
print("test : " , len(valid_videos))
# train_videos,valid_videos = train_test_split(data,test_size = 0.2)
# print(train_videos)

print("TRAIN: ", "Real:",number_of_real_and_fake_videos(train_videos)[0],"
          Fake:",number_of_real_and_fake_videos(train_videos)[1])
print("TEST: ", "Real:",number_of_real_and_fake_videos(valid_videos)[0],"
          Fake:",number_of_real_and_fake_videos(valid_videos)[1])



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)])

test_transforms = transforms.Compose([
                            transforms.ToPILImage(),
                            transforms.Resize((im_size,im_size)),
```

```
                                transforms.ToTensor(),
                                transforms.Normalize(mean,std)])
        train_data = video_dataset(train_videos,labels,sequence_length = 10,transform = train_transforms)
        #print(train_data)
        val_data = video_dataset(valid_videos,labels,sequence_length = 10,transform = train_transforms)
        train_loader = DataLoader(train_data,batch_size = 4,shuffle = True,num_workers = 4)
        valid_loader = DataLoader(val_data,batch_size = 4,shuffle = True,num_workers = 4)
        image,label = train_data[0]
        im_plot(image[0,:,:,:])
```

output: train : 8 test : 2

TRAIN: Real: 4 Fake: 4

TEST: Real: 1 Fake: 1

8. Functions of Resnext Model ,Training epoch,Testing epoch,Confusion matrix,Graphs of Loss and Accuracy:

```
#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)))
```

#train and test epoch functions

```
import torch
from torch.autograd import Variable
import time
import os
import sys
import os
def train_epoch(epoch, num_epochs, data_loader, model, criterion, optimizer):
    model.train()
    losses = AverageMeter()
    accuracies = AverageMeter()
    t = []
    for i, (inputs, targets) in enumerate(data_loader):
        if torch.cuda.is_available():
            targets = targets.type(torch.cuda.LongTensor)
            inputs = inputs.cuda()
        _,outputs = model(inputs)
        loss  = criterion(outputs,targets.type(torch.cuda.LongTensor))
        acc = calculate_accuracy(outputs, targets.type(torch.cuda.LongTensor))
        losses.update(loss.item(), inputs.size(0))
        accuracies.update(acc, inputs.size(0))
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
        sys.stdout.write(
                "\r[Epoch %d/%d] [Batch %d / %d] [Loss: %f, Acc: %.2f%%]"
                % (
                    epoch,
                    num_epochs,
                    i,
                    len(data_loader),
                    losses.avg,
                    accuracies.avg))
    torch.save(model.state_dict(),'/content/checkpoint1.pt')
    return losses.avg,accuracies.avg
def test(epoch,model, data_loader ,criterion):
    print('Testing')
    model.eval()
    losses = AverageMeter()
    accuracies = AverageMeter()
```

```
    pred = []
    true = []
    count = 0
    with torch.no_grad():
        for i, (inputs, targets) in enumerate(data_loader):
            if torch.cuda.is_available():
                targets = targets.cuda().type(torch.cuda.FloatTensor)
                inputs = inputs.cuda()
            _,outputs = model(inputs)
            loss = torch.mean(criterion(outputs, targets.type(torch.cuda.LongTensor)))
            acc = calculate_accuracy(outputs,targets.type(torch.cuda.LongTensor))
            _,p = torch.max(outputs,1)
            true +=
(targets.type(torch.cuda.LongTensor)).detach().cpu().numpy().reshape(len(targets)).tolist(
)
            pred += p.detach().cpu().numpy().reshape(len(p)).tolist()
            losses.update(loss.item(), inputs.size(0))
            accuracies.update(acc, inputs.size(0))
            sys.stdout.write(
                "\r[Batch %d / %d]  [Loss: %f, Acc: %.2f%%]"
                % (
                    i,
                    len(data_loader),
                    losses.avg,
                    accuracies.avg
                    )
                )
        print('\nAccuracy {}'.format(accuracies.avg))
    return true,pred,losses.avg,accuracies.avg
class AverageMeter(object):
    """Computes and stores the average and current value"""
    def __init__(self):
        self.reset()
    def reset(self):
        self.val = 0
        self.avg = 0
        self.sum = 0
        self.count = 0

    def update(self, val, n=1):
        self.val = val
        self.sum += val * n
        self.count += n
        self.avg = self.sum / self.count
def calculate_accuracy(outputs, targets):
    batch_size = targets.size(0)
```

```
    _, pred = outputs.topk(1, 1, True)
    pred = pred.t()
    correct = pred.eq(targets.view(1, -1))
    n_correct_elems = correct.float().sum().item()
    return 100* n_correct_elems / batch_size
```

#confusion matrix

```
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)
```

#graphs for loss and accuracy

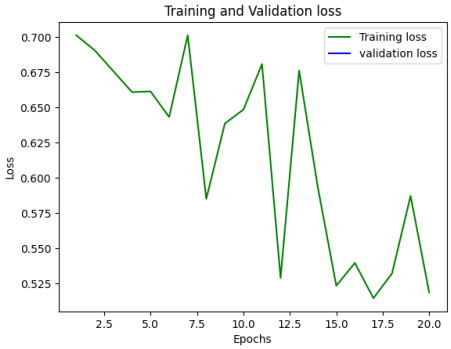
```
def plot_loss(train_loss_avg,test_loss_avg,num_epochs):
  loss_train = train_loss_avg
  loss_val = test_loss_avg
  print(num_epochs)
  epochs = range(1,num_epochs+1)
  plt.plot(epochs, loss_train, 'g', label='Training loss')
  plt.plot(epochs, loss_val, 'b', label='validation loss')
  plt.title('Training and Validation loss')
  plt.xlabel('Epochs')
  plt.ylabel('Loss')
  plt.legend()
  plt.show()
def plot_accuracy(train_accuracy,test_accuracy,num_epochs):
  loss_train = train_accuracy
  loss_val = test_accuracy
  epochs = range(1,num_epochs+1)
```

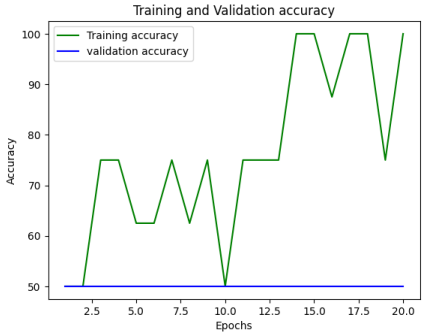
```
  plt.plot(epochs, loss_train, 'g', label='Training accuracy')
  plt.plot(epochs, loss_val, 'b', label='validation accuracy')
  plt.title('Training and Validation accuracy')
  plt.xlabel('Epochs')
  plt.ylabel('Accuracy')
  plt.legend()
  plt.show()

#execution
from sklearn.metrics import confusion_matrix
#learning rate
lr = 1e-5#0.001
#number of epochs
num_epochs = 20

optimizer = torch.optim.Adam(model.parameters(), lr= lr,weight_decay = 1e-5)

#class_weights = torch.from_numpy(np.asarray([1,15])).type(torch.FloatTensor).cuda()
#criterion = nn.CrossEntropyLoss(weight = class_weights).cuda()
criterion = nn.CrossEntropyLoss().cuda()
train_loss_avg =[]
train_accuracy = []
test_loss_avg = []
test_accuracy = []
for epoch in range(1,num_epochs+1):
   l, acc = train_epoch(epoch,num_epochs,train_loader,model,criterion,optimizer)
   train_loss_avg.append(l)
   train_accuracy.append(acc)
   true,pred,tl,t_acc = test(epoch,model,valid_loader,criterion)
   test_loss_avg.append(tl)
   test_accuracy.append(t_acc)
plot_loss(train_loss_avg,test_loss_avg,len(train_loss_avg))
plot_accuracy(train_accuracy,test_accuracy,len(train_accuracy))
print(confusion_matrix(true,pred))
print_confusion_matrix(true,pred)
```

output:

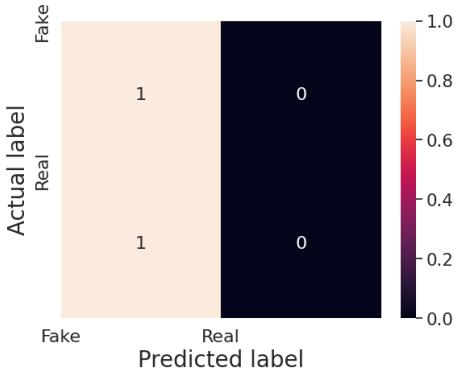
```
[Epoch 1/20] [Batch 1 / 2] [Loss: 0.701347, Acc: 50.00%]Testing
[Batch 0 / 1]  [Loss: nan, Acc: 50.00%]
Accuracy 50.0
[Epoch 2/20] [Batch 1 / 2] [Loss: 0.690577, Acc: 50.00%]Testing
[Batch 0 / 1]  [Loss: nan, Acc: 50.00%]
Accuracy 50.0
[Epoch 3/20] [Batch 1 / 2] [Loss: 0.675753, Acc: 75.00%]Testing
[Batch 0 / 1]  [Loss: nan, Acc: 50.00%]
Accuracy 50.0
[Epoch 4/20] [Batch 1 / 2] [Loss: 0.660965, Acc: 75.00%]Testing
[Batch 0 / 1]  [Loss: nan, Acc: 50.00%]
Accuracy 50.0
```

[Epoch 5/20] [Batch 1 / 2] [Loss: 0.661423, Acc: 62.50%]Testing
[Batch 0 / 1] [Loss: nan, Acc: 50.00%]
Accuracy 50.0
[Epoch 6/20] [Batch 1 / 2] [Loss: 0.643254, Acc: 62.50%]Testing
[Batch 0 / 1] [Loss: nan, Acc: 50.00%]
Accuracy 50.0
[Epoch 7/20] [Batch 1 / 2] [Loss: 0.701372, Acc: 75.00%]Testing
[Batch 0 / 1] [Loss: nan, Acc: 50.00%]
Accuracy 50.0
[Epoch 8/20] [Batch 1 / 2] [Loss: 0.585102, Acc: 62.50%]Testing
[Batch 0 / 1] [Loss: nan, Acc: 50.00%]
Accuracy 50.0
[Epoch 9/20] [Batch 1 / 2] [Loss: 0.638605, Acc: 75.00%]Testing
[Batch 0 / 1] [Loss: nan, Acc: 50.00%]
Accuracy 50.0
[Epoch 10/20] [Batch 1 / 2] [Loss: 0.648571, Acc: 50.00%]Testing
[Batch 0 / 1] [Loss: nan, Acc: 50.00%]
Accuracy 50.0
[Epoch 11/20] [Batch 1 / 2] [Loss: 0.680937, Acc: 75.00%]Testing
[Batch 0 / 1] [Loss: nan, Acc: 50.00%]
Accuracy 50.0
[Epoch 12/20] [Batch 1 / 2] [Loss: 0.528896, Acc: 75.00%]Testing
[Batch 0 / 1] [Loss: nan, Acc: 50.00%]
Accuracy 50.0
[Epoch 13/20] [Batch 1 / 2] [Loss: 0.676286, Acc: 75.00%]Testing
[Batch 0 / 1] [Loss: nan, Acc: 50.00%]
Accuracy 50.0
[Epoch 14/20] [Batch 1 / 2] [Loss: 0.593668, Acc: 100.00%]Testing
[Batch 0 / 1] [Loss: nan, Acc: 50.00%]
Accuracy 50.0
[Epoch 15/20] [Batch 1 / 2] [Loss: 0.523310, Acc: 100.00%]Testing
[Batch 0 / 1] [Loss: nan, Acc: 50.00%]
Accuracy 50.0
[Epoch 16/20] [Batch 1 / 2] [Loss: 0.539514, Acc: 87.50%]Testing
[Batch 0 / 1] [Loss: nan, Acc: 50.00%]
Accuracy 50.0
[Epoch 17/20] [Batch 1 / 2] [Loss: 0.514449, Acc: 100.00%]Testing
[Batch 0 / 1] [Loss: nan, Acc: 50.00%]
Accuracy 50.0
[Epoch 18/20] [Batch 1 / 2] [Loss: 0.532006, Acc: 100.00%]Testing
[Batch 0 / 1] [Loss: nan, Acc: 50.00%]
Accuracy 50.0
[Epoch 19/20] [Batch 1 / 2] [Loss: 0.587142, Acc: 75.00%]Testing
[Batch 0 / 1] [Loss: nan, Acc: 50.00%]
Accuracy 50.0
[Epoch 20/20] [Batch 1 / 2] [Loss: 0.518556, Acc: 100.00%]Testing
[Batch 0 / 1] [Loss: nan, Acc: 50.00%]
Accuracy 50.0
20





[[1 0]
[1 0]]
True positive = 1
False positive = 0
False negative = 1
True negative = 0

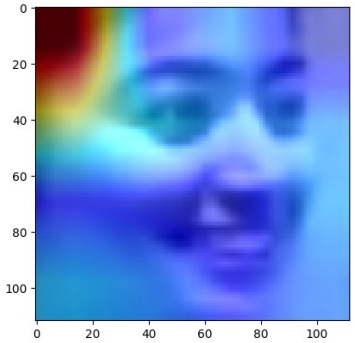


Calculated Accuracy 50.0

9. Code to make Predictions:

```
#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/MyDrive/TestPhase/sample4_fake.mp4"]

video_dataset = validation_dataset(path_to_videos,sequence_length = 40,transform =
train_transforms)
model = Model(2).cuda()
path_to_model =
'/content/drive/MyDrive/Models/model_93_acc_100_frames_celeb_FF_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")
```

Output: confidence of prediction: 87.55947709083557



FAKE

5.4 Tools and Technologies Used

SNO	FEILDS	NAME OF TOOLS AND TECHNOLOGIES
1	UMLTools	draw.io
2	Programming Languages	Python3 JavaScript
3	Programming Frameworks	PyTorch Django
4	IDE	Google Colab Jupyter Notebook Visual Studio Code
5	Versioning Control	git
6	Cloud Services	Google Cloud Platform
7	Libraries	Torch, torchvision, os, numpy,cv2, matplotlib face_recognition, json, pandas, copy, glob random, sklearn

CHAPTER 6

Software Testing

6.1 Type of Testing Used

Functional Testing

1. **Unit Testing**: Unit testing is a software testing method where individual units or components of a software are tested in isolation to ensure they function correctly as designed, typically through automated testing frameworks.

   Unit testing in deep fake detection involves verifying the functionality of individual components or algorithms within the detection system to ensure accurate identification of manipulated media.

2. **Integration Testing:** Integration testing verifies interactions between different components/modules of a system to ensure they work together correctly, typically involving testing interfaces and data flow.

   In deepfake detection, integration testing ensures seamless collaboration between various detection methods and modules to enhance accuracy and reliability of identifying manipulated content.

3. **System Testing:** System testing assesses the entire software system's behavior in accordance with specified requirements, ensuring its functionality, performance, and reliability meet expectations.

   In deepfake detection, system testing examines the overall performance and effectiveness of the detection system in identifying manipulated media across various scenarios and conditions.

4. **Interface Testing:** Interface testing evaluates the interactions between different software components, focusing on ensuring smooth communication and data exchange between interfaces, APIs, and modules.

   Interface testing specifically evaluates the integration and interaction between different algorithms, models, and components within the deepfake detection system, ensuring seamless communication and accurate data exchange to enhance detection accuracy.

Non-functional Testing

1. **Performance Testing:** Performance testing evaluates the speed, responsiveness, and resource usage of deepfake detection algorithms and systems to ensure efficient and timely identification of manipulated media.

   Performance testing in deepfake detection assesses algorithmic efficiency and system scalability to swiftly and accurately detect manipulated content.

2. **Load Testing:** Load testing involves evaluating a system's performance under anticipated load levels, assessing its ability to handle concurrent users and transactions while maintaining acceptable response times and resource utilization.

   Load testing in deepfake detection evaluates the algorithmic robustness and system scalability under varying levels of computational demand, ensuring efficient and accurate identification of manipulated media amidst increased processing loads.

3. **Compatibility Testing:** Compatibility testing ensures that software or systems are compatible with different environments, devices, and configurations, verifying seamless functionality across various platforms to guarantee a consistent user experience.

   Compatibility testing in deepfake detection ensures that detection algorithms and systems function effectively across different media formats, resolutions, and platforms, ensuring consistent and accurate identification of manipulated content across diverse sources and contexts.

6.2 Test Cases and Test Results

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

Table 6.1: Test Case Report

CHAPTER 7

Results and Discussion

7.1 Screen Shots

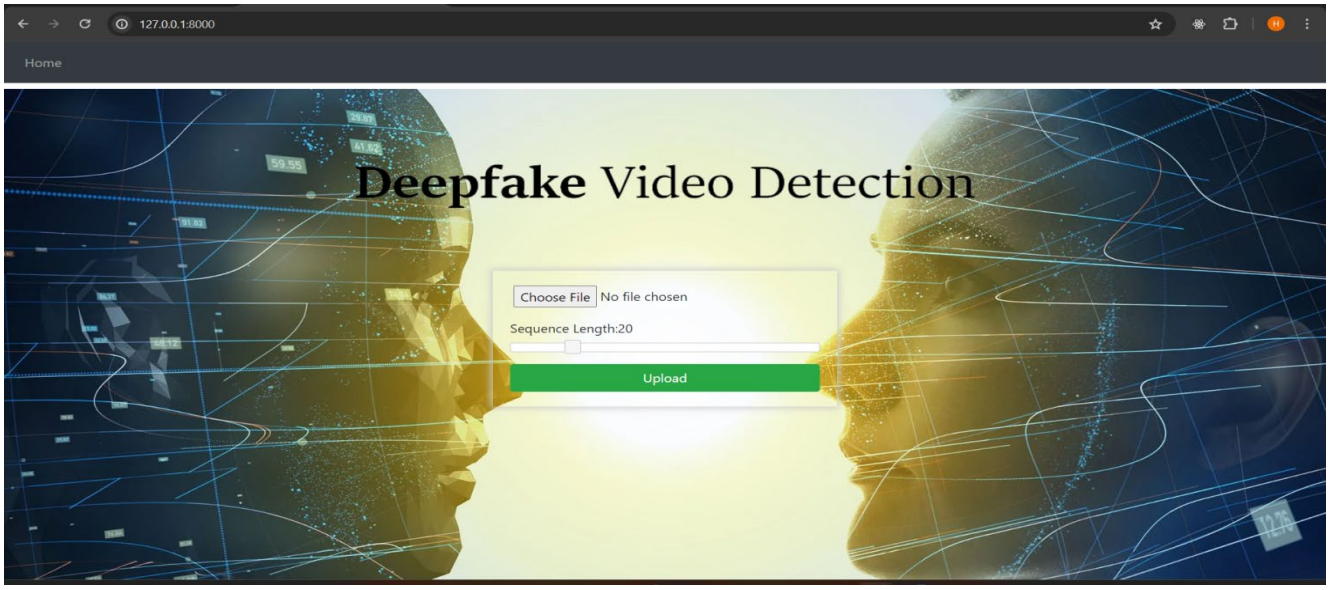


Figure 7.1: Home Page

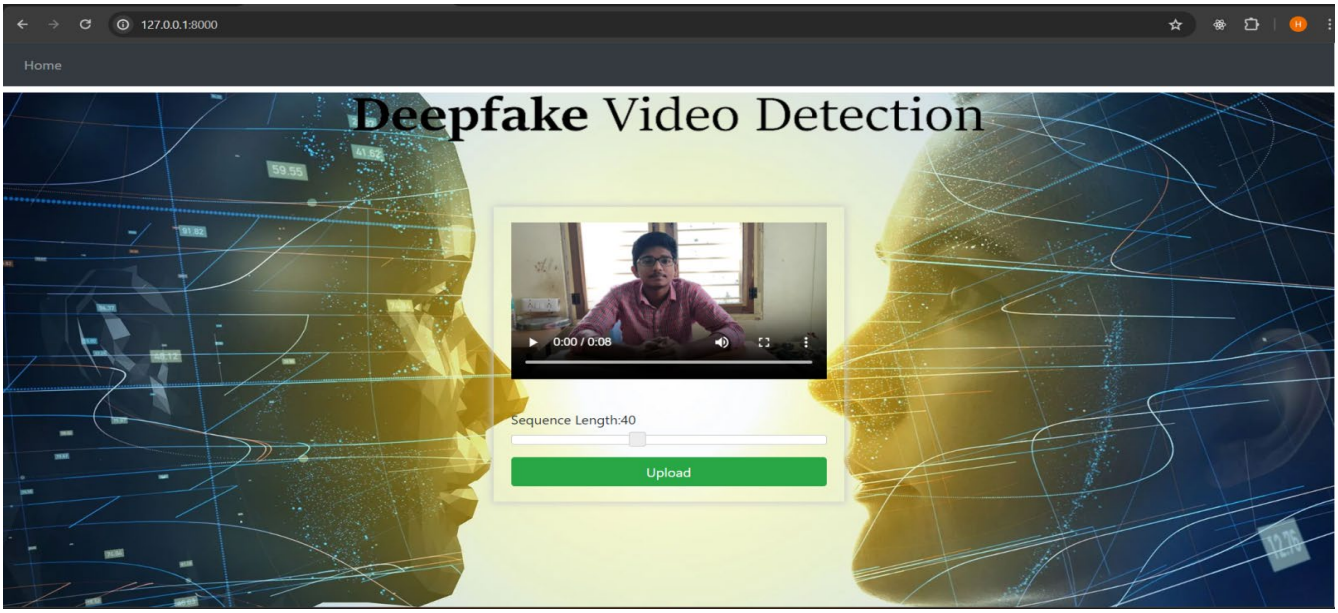


Figure 7.2: Uploading Real Video

Figure 7.3: Real Video Output

Figure 7.4: Uploading Fake Video

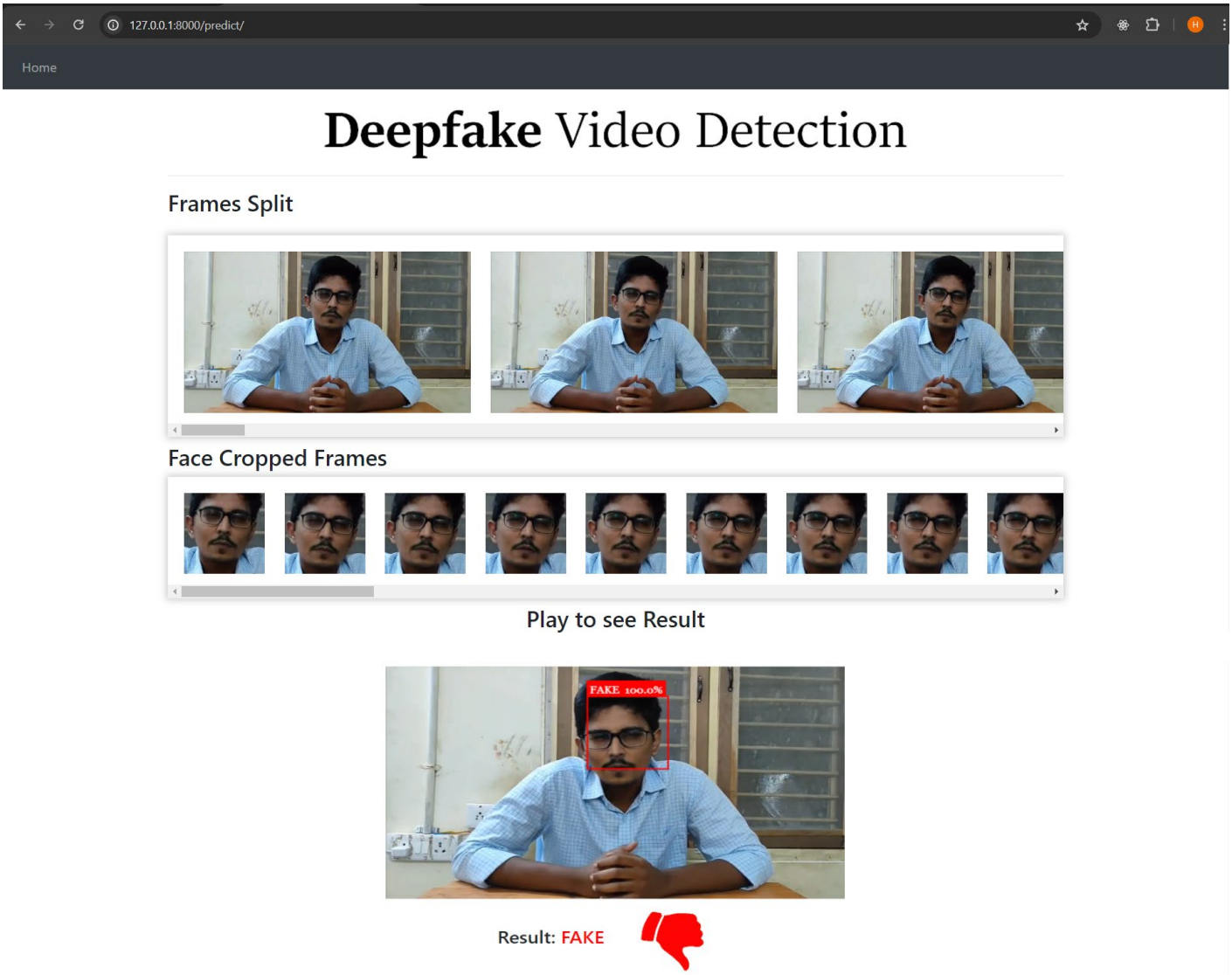


Figure 7.5: Fake video Output

Figure 7.6: Uploading Video with no faces

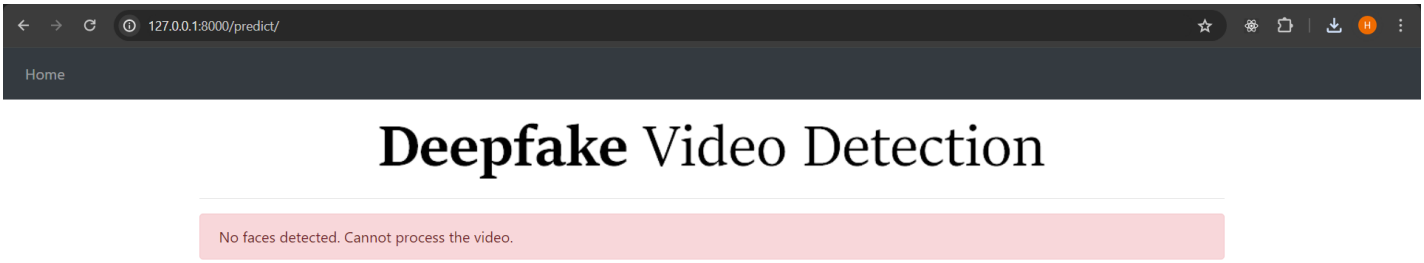


Figure 7.7: Output of Uploaded video with no faces

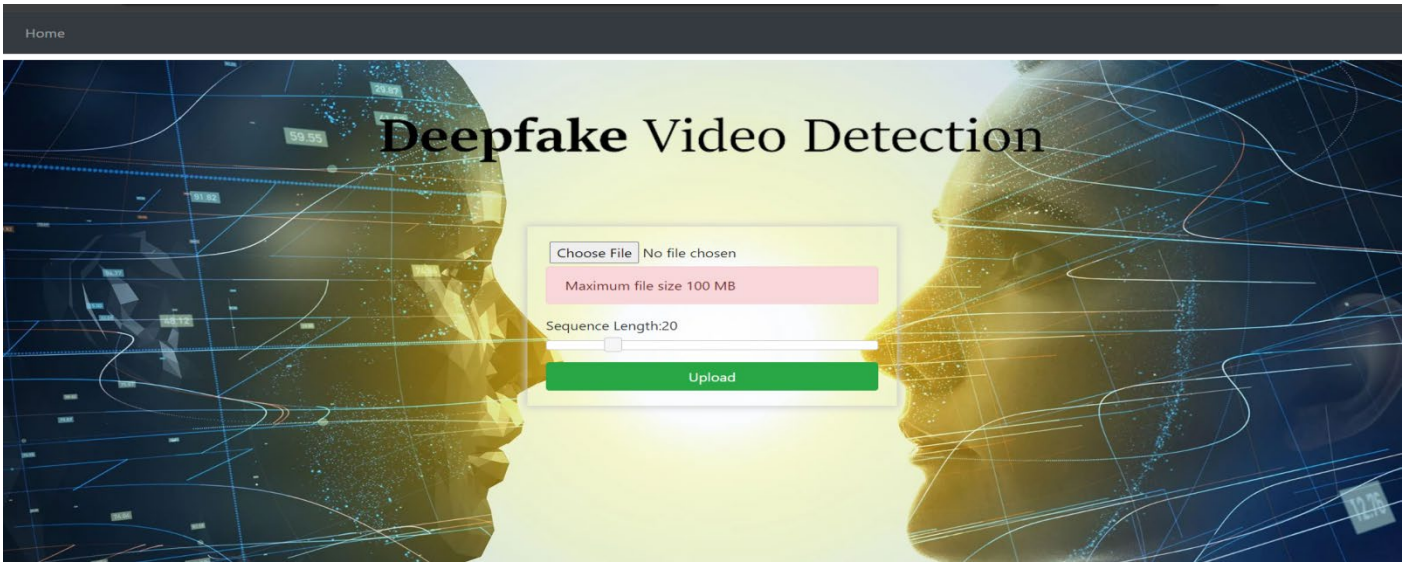


Figure 7.8: Uploading file greater than 100MB

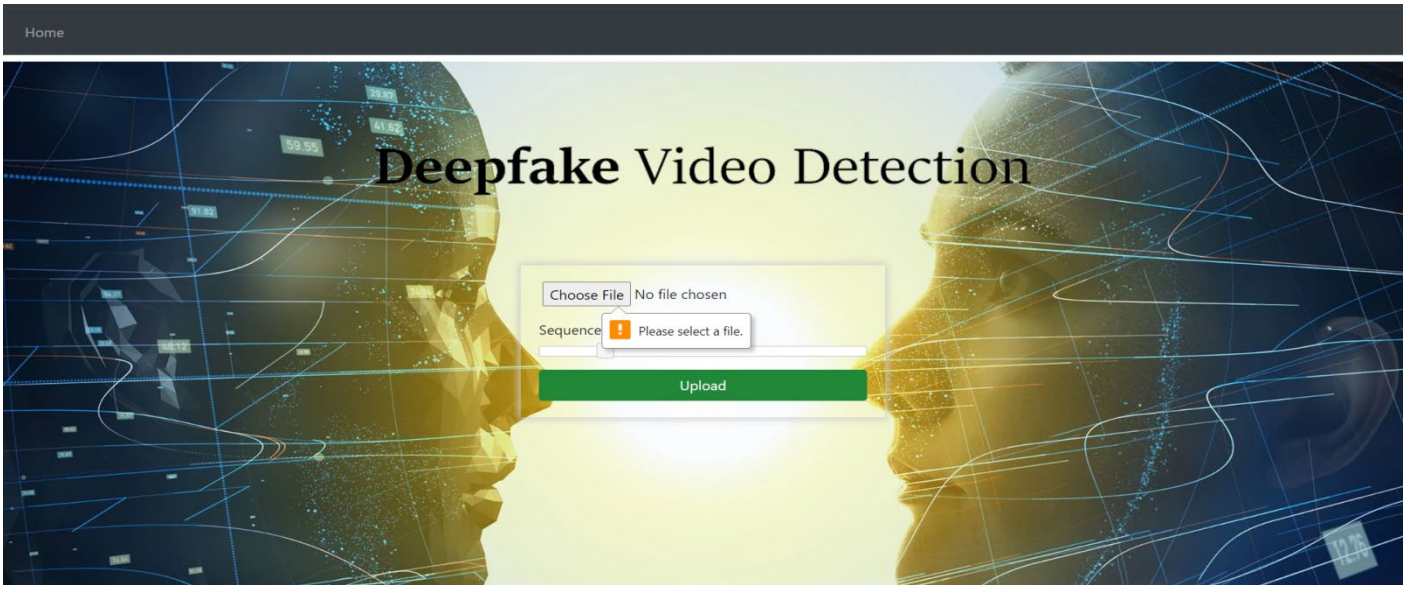


Figure 7.9: Pressing Upload button without selecting video

7.2 Outputs

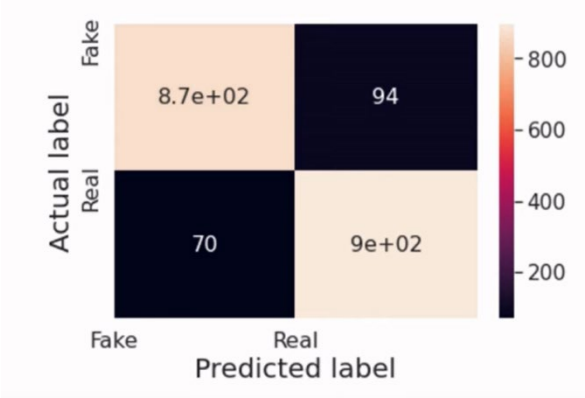
7.2.1 Model Results:

Model Name	Dataset	No. of videos	Sequence length	Accuracy
model_84_acc_10_frames_final_data	Our Dataset	6000	10	84.21461
model_87_acc_20_frames_final_data	Our Dataset	6000	20	87.79160
model_89_acc_40_frames_final_data	Our Dataset	6000	40	89.34681
model_91_acc_60_frames_final_data	Our Dataset	6000	60	91.59097
model_93_acc_100_frames_celeb_FF_data	Celeb-DF + FaceForensic++	3000	100	93.97781
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

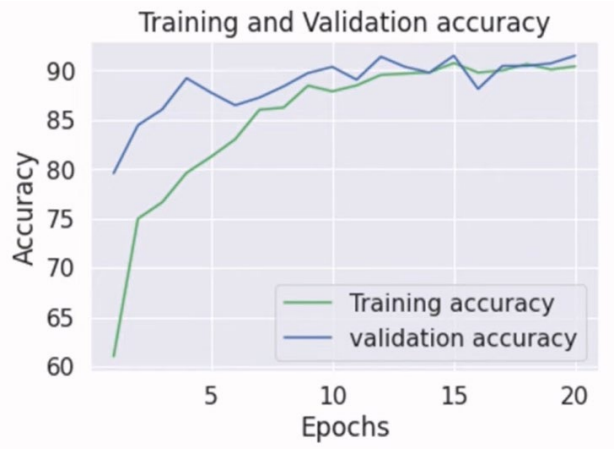
7.2.2 Other Parameters:

Graphs of Model 80 frames Sequence Length on Our Dataset

1. Confusion Matrix:



2. Training and Validation Accuracy:



3. Training and Validation Loss:

CHAPTER 8

Conclusion and Future Scope

8.1 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.

8.2 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.

CHAPTER 9

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

[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/

[5] 10 deepfake examples that terrified and amused the internet : https://www.creativebloq.com/features/deepfake-examples

[6] TensorFlow: https://www.tensorflow.org/

[7] Keras: https://keras.io/

[8] PyTorch : https://pytorch.org/

[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/

[12] Face Swap : https://faceswaponline.com/

[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.

[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 net- works to detect forged images and videos ” in arXiv:1810.11215.

[18] D. Güera and E. J. Delp, "Deepfake Video Detection Using Recurrent Neural Networks," 2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), Auckland, New Zealand, 2018, pp. 1-6.

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

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

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

[22] ResNext Model : https://pytorch.org/hub/pytorch_vision_resnext/

[23] https://www.geeksforgeeks.org/software-engineering-cocomo-model/

[24] DeepfakeVideo Detection using Neural Networks: http://www.ijsrd.com/articles/IJSRDV8I10860.pdf

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