A PROJECT REPORT ON

Robust Deepfake Detection through Res-Next CNN and LSTM-based RNN Fusion using Deep Learning

Mini project submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY
IN
INFORMATION TECHNOLOGY
(2021-2025)

R. HRITISH	**21241A12B4**
CHARAN LAKKAM	**21241A1277**
D. YASHWANTH	**21241A1281**

Under the esteemed guidance
Of
R. V. S. S. S. NAGINI
Associate Professor

DEPARTMENT OF INFORMATION TECHNOLOGY
GOKARAJU RANGARAJU INSTITUTE OF ENGINEERING AND TECHNOLOGY(AUTONOMOUS)
HYDERABAD

(2024-2025)

CERTIFICATE

This is to certify that it is a bonafide record of Mini Project work entitled **"ROBUST DEEPFAKE DETECTION THROUGH RES-NEXT CNN AND LSTM-BASED RNN FUSION USING DEEP LEARNING"** done by **R.HRITISH (21241A12B4), CHARAN. L(21241A1277), D. YASHWANTH (21241A1281)** of **B. Tech** in the Department of Information of Technology, **GOKARAJU RANGARAJU INSTITUTE OF ENGINEERING AND TECHNOLOGY** during the period 2020-2024 in the partial fulfillment of the requirements for the award of degree of **BACHELOR OF TECHNOLOGY IN INFORMATION TECHNOLOGY** from GRIET, Hyderabad.

Dr. R. V. S. S. S. Nagini
Associate Professor
(Internal Guide)

Dr. Y. J. Nagendra Kumar
Head of the Department
(Project-External)

ACKNOWLEDGEMENT

We take the immense pleasure in expressing gratitude to our Internal guide**, Dr. R.V.S.S.S.Nagini, Associate Professor, Dept of IT**, GRIET. We express our sincere thanks for his encouragement, suggestions and support, which provided the impetus and paved the way for the successful completion of the project work.

We wish to express our gratitude to **Dr. Y J Nagendra Kumar,** HOD IT, our ProjectCoordinators **Dr. R.V.S.S.S.Nagini** and **Mr. P.K. Abhilash** for their constantsupport during the project.

We express our sincere thanks to **Dr. Jandhyala N Murthy,** Director, GRIET**,** and **Dr. J. Praveen,** Principal, GRIET**,** for providing us the conductive environment for carrying through our academic schedules and project with ease.

We also take this opportunity to convey our sincere thanks to the teaching and non- teaching staff of GRIET College, Hyderabad.

Email: hritish4321@gmail.com
Contact No: 8500045573

Email: lakkamcharan123@gmail.com
Contact No: 8008475728

Email: yashwanthdaramalla@gmail.com
Contact No.: 8106226955

DECLARATION

This is to certify that the mini-project entitled "**ROBUST DEEPFAKE DETECTIONTHROUGH RES-NEXT CNN AND LSTM-BASED RNN FUSION USING DEEP LEARNING"** is a bonafide work done by us in partial fulfillment of the requirements for the award of the degree **BACHELOR OF TECHNOLOGY IN INFORMATIONTECHNOLOGY** from Gokaraju Rangaraju Institute of Engineering and Technology,Hyderabad.

We also declare that this project is a result of our own effort and has not been copied or imitated from any source. Citations from any websites, books and paper publications are mentioned in the Bibliography.

This work was not submitted earlier at any other University or Institute for the award of any degree.

R. HRITISH	**21241A12B4**
CHARAN.L	**21241A1277**
D. YASHWANTH	**21241A1281**

TABLE OF CONTENTS

12. LIST OF DIAGRAMS

ABSTRACT

The exponential growth in computational power has elevated the capabilities of deep learning algorithms, This makes creating indistinguishable human-synthesized videos, also known as deepfakes, incredibly. These sophisticated manipulations of realistic face-swapped deepfakes, aiming to fabricate political unrest, simulate terrorism events, propagate revenge porn, and coerce people by blackmail, have already become very common.

We propose a novel methodology for deep learning in view of an emerging challenge to efficiently distinguish these AI-generated fake videos from real ones.The approach is to focuson the automatic detection of replacement and reenactment deepfakes, utilizing Artificial Intelligence (AI) as a defense against AI-driven deception. At the core of our system lies a Res- Next CNN extracting frame-level features. These features are further used to train an LSTM- based RNN for video classification with regard to manipulation/non-manipulation, in essence making a distinction between deepfakes and genuine videos.

To ensure the real-time applicability of our model, we rigorously evaluate its performance on a substantial data set. This data set is meticulously crafted by amalgamating diverse sources, including Face-Forensics, Deep fake Detection Challenge, and Celeb-DF Datasets. Our results showcase the efficacy of a straightforward yet good approach

Keywords: Res-Next CNN, Recurrent Neural Network (RNN), LSTM-Long Short-TermMemory Computer vision

DOMAIN:

Machine Learning

1. INTRODUCTION

1.1 Introduction to Project

Now, with increasingly sophisticated hand-held mobile camera technology and the expanding reach of social-networking media, multi-media sharing websites, digital video sharing and creation are now simpler. A few deep-learning technologies spawned could have been thought impossible a mere handful of years ago. Among these are the sophisticated generative modelsof today embarking on producing remarkable life-like speech, music, images and videos even. These models have been applied on a wide range of environments from helping to provide training data for medical imaging to improving accessibility via text to speech.

Like any revolutionary technology, this has brought up new difficulties. Namely, the so-called "deep fakes" that have been made possible through the use of deep generative models produce fully realistic fake video and audio clips. Following their very first appearance less than a year ago, many open-source deep fake generation methods and tools are now available, and the number of synthesized media clips grows. While many of them are probably meant to be funny or entertaining, some of them could be inappropriate for individuals and society. Until recently, the number of fake videos along with degrees of realism has been increasing due to the availability of editing tools coupled with a high demand for domain expertise. Deep fakes over the social media platforms have grown very common, leading to spamming and peculating wrong information over the platform. Imagine a deep fake by our prime minister declaring war against neighboring countries, or a deep fake by any reputed celebrity abusing the fans. These types of deep fakes will be terrible and threat StringBuffer common people. Deep fake detection is very important for overcoming such a situation. Hence, we describe a new deep learning approach that differentiates between true and AI-generated fake video, in turn detecting the deep fake with increased accuracy. The technology must be developed to detect the fakes, which may help to find and stop the deep fakes from moving through the internet.

Goals and objectives:

- Our project seeks to unravel the deep fakes' distorted reality.
- Our project will reduce the Abuses' and misleading of the common people on the worldwide web.
- Our project will detect the label video as being either a deepfake or pristine.
- Provide a user-friendly system for uploading the video to the system and

distinguishing

- Whether the video is real or fabricated.

1.2 Existing System

One of the most challenges of NLP is that human dialect is complex and vague, making it troublesome for computers to get it and translate it precisely. NLP methods include the utilize of different scientific models and calculations, such as profound learning, measurable models, and rule-based approaches. A few of the well-known methods utilized in deepfake discovery incorporate - Opinion investigation, This strategy includes analyzing content information to decide the opinion communicated in it. Estimation investigation is regularly utilized to identify deepfake audits, comments, and social media posts. Named Substance Acknowledgment (NER), This procedure includes distinguishing and classifying named substances in content information, such as names of individuals, organizations, and areas. NLP methods have been utilized in different deepfake discovery approaches, such as recognizing fake audits, identifying fake news, and identifying fake social media posts. Be that as it may, deepfake discovery utilizing NLP is still a challenging errand, as deepfake methods are getting to be progressively modern, making it troublesome for NLP models to identify them precisely.

1.3 Proposed System

Deepfake is one of those DL-powered apps that has of late surfaced. So, deepfake frameworks can make fake pictures basically by substitution of scenes or pictures, motion pictures, and sounds that people cannot tell separated from genuine ones. Different advances have brought thecapacity to alter a engineered discourse, picture, or video to our fingers. Besides, video and picture fakes are presently so persuading that it is difficult to recognize between wrong and true substance with the bare eye.The comes about propose that the Ordinary Neural Systems (CNN) technique is the foremost regularly utilized DL strategy in distributions. Concurring to investigate, the lion's share of the articles are on the subject of video deepfake discovery. The lion's share of the articles centered on improving as it were one parameter, with the accuracy parameter accepting the foremost consideration.

2. REQUIREMENT ENGINEERING

2.1 Purpose and Scope of Document

A project plan for deepfake video detection using a neural network is provided in this document. This paper is intended for sponsors and upcoming developers of the Deepfake video detection by neural networks project. It will cover, among the other things, the system's functionality summary, the project'sscope as seen by the team, use case, activity and data flow diagrams and functional and non-functional requirements while evaluating project's risks and the management. This document describes the methodology used for developing the project as well as the measurement metrics that will be tracked all the way through.

2.1 Hardware Requirements

In the case of this project, a powerful enough computer will be needed. This project consumes a lot ofprocessing power because it involves image and video batch processing.

Client-side Requirements:

Type Browser: Any Compatible

browser deviceProcessor – i5 and

above (64-bit OS).

- RAM: 8GB or more.
- Storage – 500GB HDD or SSD (Higher specs are recommended for high performance).
- Input devices – Keyboard, Mouse.
- Network **-** Broadband internet connection for accessing APIs and online resources.
- Graphic card - NVIDIA GeForce GTX Titan (12GB RAM)

2.2 Software Requirements

1. OS: Windows 8 +
2. Python - Version 3.7 or higher.
3. Cloud platform: Google Cloud Platform
4. PyTorch 1.4,Django 3.0
5. Libraries: Face-recognition, Open CV

3. LITERATURE SURVEY

Confront Distorting Artifacts utilized the approach to distinguish artifacts by comparing the created confront regions and their encompassing districts with a devoted Convolutional Neural Arrange show. In this work there were two-fold of Confront Artifacts. Their strategy is based on the perceptions that current deepfake calculation can as it were produce pictures of constrained resolutions, which are at that point required to be encourage changed to coordinate the faces to be supplanted within the source video. Their strategy has not considered the transient investigation of the outlines.

The Long-term Repetitive Convolution Organize (LRCN) was utilized for transient investigation of the trimmed outlines of eye squinting. As nowadays the deepfake era calculations have ended up so capable that need of eye flickering can not be the as it were clue for location of the deepfakes. There must be certain other parameters must be considered for the location of profound- fakes like teeth charm, wrinkleson faces, off-base situation of eyebrows etc.

Capsule systems to identify manufactured pictures and recordings employments a strategy that employments a capsule arrange to identify manufactured, controlled pictures and recordings totallydifferent scenarios, like replay assault discovery and computer-generated video discovery.

In their method, they have utilized random noise within the preparing stage which isn't a great alternative. Still the show performed advantageous in their data set but may fall flat on genuine time data due to commotion in preparing. Our strategy is proposed to be prepared on silent and genuine time datasets.

Repetitive Neural Arrange (RNN) for deepfake detection used the approach of utilizing RNN for successive preparing of the outlines in conjunction with Ima- geNet pre-trained show. A HOHO dataset comprising of fair 600 recordings is utilized. We are going be preparing out demonstrate on expansive number of Realtime information.

Manufactured Representation Recordings utilizing Organic Signals [20] approach extricate bio-coherent signals from facial locales on flawless and deepfake representation video sets. Connected changes to compute the spatial coherence and transient consistency, capture the flag characteristics in highlight vector and photoplethysmography (PPG) maps, and advance prepare a probabilistic Bolster Vector Machine (SVM) and a Convolutional Neural Arrange (CNN). At that point, the average of realness probabilities is utilized to classify whether the video could be a deepfake or a pristine.

Fake Catcher identifies fake substance with tall precision, autonomous of the generator, substance, determination, and quality of the video. Due to need of discriminator driving to the misfortune in their findings to protect organic signals, defining a differentiable misfortune work that follows the proposed flag handling steps isn't straight forward handle.

4. Problem Definition and scope

4.1 Problem Statement

Persuading controls of computerized pictures and recordings have been illustrated for a few decades through the utilize of visual impacts, later progresses in profound learning have driven to a emotional increment within the authenticity of fake substance and the availability in which it can be made. These so-called AI-synthesized media (prevalently alluded to as profound fakes). Creating the Profound Fakes utilizing the Artifificially cleverly devices are straightforward errand. But, when it comes to location of these Profound Fakes, it is major challenge. As of now within the history there are numerous cases where the deepfakes are utilized as capable way to make political tension[14], fake fear mongering occasions, vindicate porn, extortion people groups etc.So it gets to be exceptionally critical to distinguish these deepfake and maintain a strategic distance from the permeation of deepfake through social media stages. We have taken a step forward in identifying the profound fakes utilizing LSTM based artifificial Neural organize.

4.1.1 Goals and Objectives:

- Our venture points at finding the misshaped truth of the profound fakes.
- Our extend will diminish the Abuses' and deluding of the common individuals on the world wide web.
- Our venture will recognize and classify the video as deepfake or flawless.
- Give a simple to utilize framework for utilized to transfer the video and recognize whether the video isgenuine or fake.

4.1.2 Statement of scope

There are numerous instruments accessible for making the profound fakes, but for profound fake locationthere's barely any device accessible. Our approach for identifying the profound fakes will be extraordinary commitment in maintaining a strategic distance from the permeation of the profound fakes over the world wide web. We are going be giving a web-based stage for

the client to transfer the video and classify it as fake or genuine. This extend can be scaled up from creating a web-based stage to a browser plugin for programmed profound fake detection's. Indeed huge application like WhatsApp, Facebook can coordinated this extend with their application for simple pre- detection of profound fakes some time recently sending to another client. A portrayal of the program withMeasure of input, bounds on input, input approval, input reliance, i/o state graph, Major inputs, and yieldsare depicted without respect to usage detail.

4.2 Major Constraints

• User: User of the application will be able identify the whether the uploaded video is fake or genuine,Along with the show confidence of the forecast.

• Prediction: The Client will be able to see the playing video with the yield on the confront at the side theconfidence of the show.

• Simple and User-friendly User-Interface: Clients appear to incline toward a more simplified handle ofProfound Fake video location. Subsequently, a straight forward and user-friendly interface is implemented.The UI contains a browse tab to choose the video for preparing. It diminishes the complications and at the same time improve the client encounter.

• Cross-platform compatibility: with an ever-increasing target market, accessibility should be your primary need. By empowering a cross-platform compatibility include, you'll be able increase your reachto over distinctive stages. Being a server side application it'll run on any gadget that includes a web browser introduced in it.

4.3 Methodologies of Problem solving

4.3.1 Analysis

• Solution Requirement

We analyzed the problem statement and found the feasibility of the solution of the problem. We read different research paper as mentioned in 3.3. After checking the feasibility of the problem statement. Thenext step is the data set gathering and analysis. We analyzed the data set in different approach of traininglike negatively or positively trained i.e Training the model with only fake or real videos may add extra bias in the model leading to incorrect predictions. So, we found from a lot of research that balanced training of the algorithm can avoid bias and variance in the algorithm, getting good accuracy.

• Solution Constraints

We analyzed the solution in terms of cost, Requirements, speed of processing, equipment's availability.

• Parameters Identified

1. Blinking of eyes

2. Teeth enchantment

3. distance for eyes(Bigger or Smaller)

4. Moustache

5. Edges, eyes, nose and ears

6. Iris segmentation

7. face Wrinkles

8. Head pose

9. Face angles

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 5. We decided the baseline architecture of the Model which includes the different layers and theirnumbers.

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 fifinal model on large number of data-set.

4.3.4 Evaluation

We assessed our show with a expansive number of genuine time information set which incorporate YouTube recordings information set. Disarray Lattice approach is utilized to assess the exactness of theprepared demonstrate.

4.4 Outcome

The outcome of the solution is trained deepfake detection models that will help the users to check if thenew 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 show will pre-process the video and anticipate whether the transferred video may be a deepfake or genuine video .

5. TECHNOLOGY

5.1 Programming Languages:

5.1.1 Python3

Python 3 could be a high-level, translated programming dialect known for its lucidness and ease of utilize. It bolsters numerous programming standards, counting procedural, object- oriented, and useful programming. Python 3 made strides upon Python 2 with highlights like way better Unicode support and more steady language structure. It encompasses a tremendous standard library and a flourishing biological system of third-party bundles, making it appropriate for a wide run of applications. Python 3 is broadly utilized in web advancement, information science, manufactured insights, and computerization. Its community-driven improvement guarantees persistent change and back.

It is used in various fields:

- Desktop GUI Applications
- Web Applications
- Console-based Application
- Software Improvement
- Audio-Video based Applications
- Scientific and Numeric Applications
- Business Applications
- 3D CAD Applications

5.1.2 JavaScript

JavaScript is a flexible, general programming language that is used mostly in the development of interactive web pages. It runs in the browser and gives a developer the ability to dynamically change HTML and CSS on the fly in reaction to events, thus refreshing the content dynamically and checking the forms for validity. JavaScript is also used on the server-side with environments like Node.js, thus allowing full-stack development. It supports object-oriented, imperative, and functional programming styles. Modern JavaScript supports ES6 modules, async/await, and Promises, among other features that make coding easier in terms of code handling and asynchronous operations. It can use libraries and frameworks from a big ecosystem, including React, Angular, and Vue.

- Server-Side Development
- Mobile App Development
- Desktop Applications
- Game Development

5.2 Programming Frameworks

1. PyTorch

PyTorch is an open-source deep learning framework developed by Facebook's AI Research Lab. Differentfrom TensorFlow, PyTorch provides users with dynamic computation graphs and through an enhanced ease-of-use interface where relating to the construction and training of neural networks are conducted.

PyTorch is used in research and production because of its great effectiveness in supporting GPU's andlarge libraries of built-in, pre-trained models.

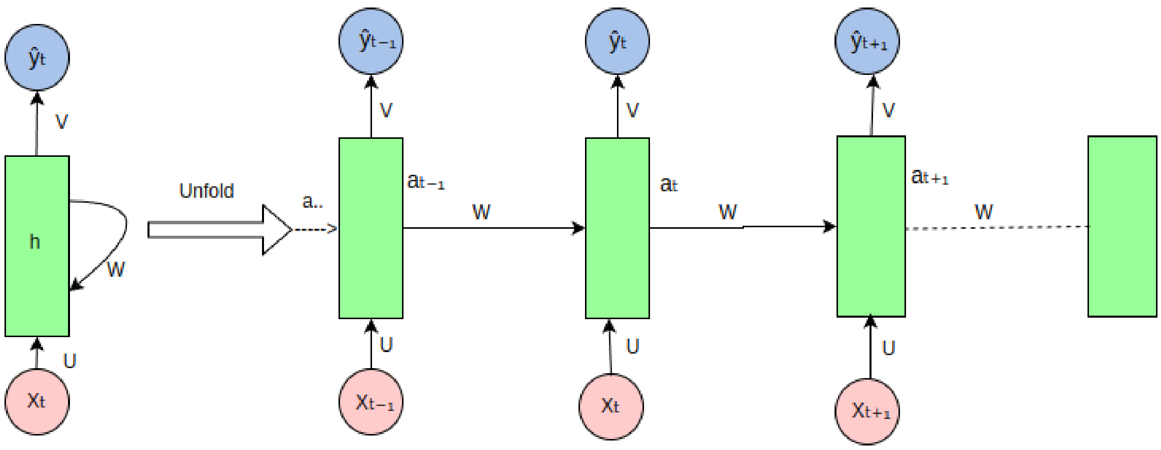
3. Django

Django is a high-level Python web framework that encourages rapid development and clean, pragmaticdesign. This system comes with an ORM, authentication, and an administration interface that enable a developer to build robust web applications, saving tremendous time. Django adheres to the concept "Don't Repeat Yourself," which encourages reuse and efficiency.

RNN:

RNN Architecture

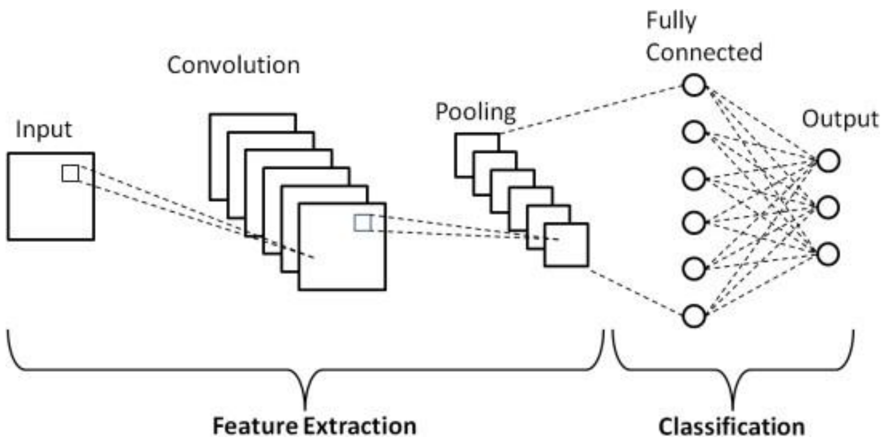
Unfolded RNN diagram to understand this RNN concept better.



A Recurrent Neural Network (RNN) is a type of neural network architecture designed for sequential dataprocessing, where each input's output influences the next step's prediction. Unlike traditional neural networks, which treat inputs and outputs independently, RNNs excel in tasks requiring predictions basedon contextual dependencies, such as predicting the next word in a sentence. This capability is facilitated by the network's hidden layer, which maintains a memory state across time steps. This hidden state, or memory state, retains information from previous inputs, enabling the network to capture temporal dependencies within sequential data.

Key to the RNN's functionality is its shared parameters across all time steps, including hidden layers,which ensures consistent processing of sequential inputs. This parameter sharing reduces complexity compared to other neural network architectures, making RNNs particularly effective for tasks involvingsequential data analysis and prediction.

CNN:



convolutional neural network (CNN) architecture

A Convolutional Neural Network (CNN) is a specialized architecture within deep learning that is widely utilized in Computer Vision, an area of Artificial Intelligence focused on enabling machines to interpret and understand visual information. In the realm of Machine Learning, Artificial Neural Networks (ANNs)have proven to be highly effective across various types of datasets including images, audio, and text.

Different types of neural networks are tailored to specific tasks; for instance, Recurrent Neural Networks (RNNs), particularly models like Long Short-Term Memory (LSTM), excel in sequential data predictiontasks such as language modeling.

In this blog post, we aim to construct a fundamental building block for CNNs. Unlike traditional neural networks, CNNs introduce specialized layers designed to exploit spatial hierarchies inherent in images: 1.Input Layer : This initial layer receives input data, such as images represented as pixel values. The number of neurons in the input layer corresponds to the total number of features in the data, which in thecase of images is typically the number of pixels.

2. Convolutional Layers : These layers apply convolution operations across the input data, extractingfeatures like edges and textures. Each convolutional layer consists of multiple filters (kernels) that convolve over the input data, producing feature maps that capture spatial patterns.

3. Pooling Layers : Following convolutional layers, pooling layers reduce the spatial dimensions of featuremaps while retaining important information. Common pooling operations include max pooling, which extracts the maximum value from each patch of the feature map.

4. Fully Connected Layers : These layers integrate the extracted features from convolutional and poolinglayers into a dense layer of neurons.

5. Output Layer : The final layer of the CNN utilizes activation functions such as sigmoid (for binaryclassification) or softmax (for multi-class classification) to compute class probabilities based on the learned features.

By leveraging these specialized layers, CNNs can effectively learn hierarchical representations of visualdata, making them indispensable tools in various applications of computer vision.

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

6. DESIGN REQUIREMENT ENGINEERING

CONCEPT OF UML:

The aim of those diagrams, which are based on UML, is to visually represent the machine as well as its primary players, roles, moves, objects, or training, with the intention of better understanding, manipulating, preserving, or filing statistics about the machine.

DIAGRAMS:

Unified Modelling Language is used to build models for various purposes and thereby, provides a more standardized means of visually representing a system's structure which is comparable to diagrams in other branches of designing. Essential to complicated applications, people interfaces should be always well-coordinated, especially if several teams are integrated. While he / she might not understand code, UML fills the gap of understanding this crevice by business people. It imparts essential framework structures, properties, and techniques to individuals who intend to be modest programmers. By means of the forms of use, the client intelligent, and decreased framework structure, UML saves the time of difference groups. Paralleled with the object-oriented plan and assessment, UML utilizes components andrelations to devise maps. According to the way UML charts are implemented they can be categorized in to structural and behavioral chart. It is extremely important to note that UML is intrinsically tied to

18object-oriented design and analysis, and hence employs objects like classes, objects, interfaces andrelationships in order to create detailed models

6.1 Data Flow Diagram:

DFD Level-0

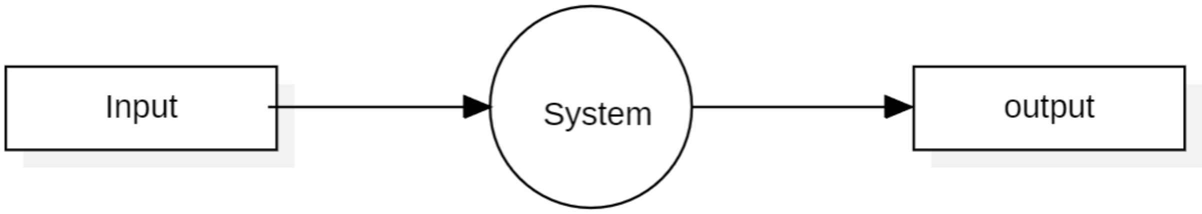


Fig 6.1.1 Level -0

DFD Level – 0 DFD indicates the basic flow of data in the system. In this System Input is given equalimportance 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 andoutput 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.

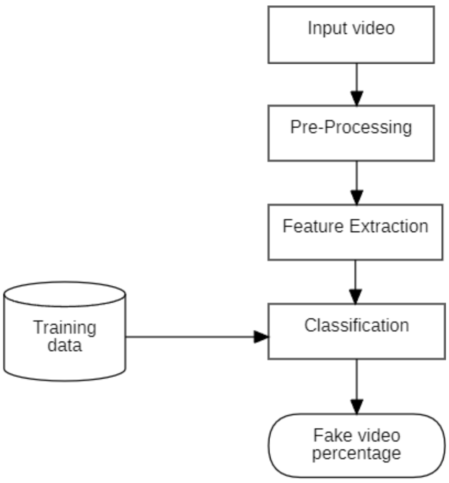


Fig 6.1.2: Level 1 DFD

DFD Level-2

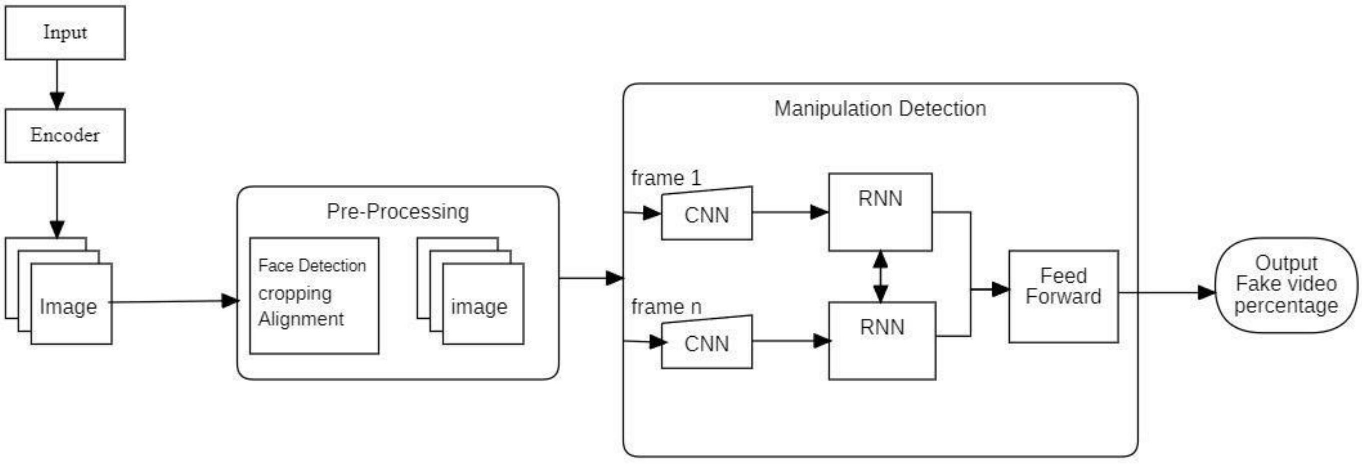


FiG 6.1.3: Level 2 DFD

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

6.2 Use Case Diagram:

One possibility is, that a behavior graph could possibly be a walk like a behavioral graph which describes a behavior that is easily understood to refer to the system functionalities, the members (performing artists), their goals, and the intelligent between these use cases. It outlines how the clients engaged with the framework making it easier to capture requirement and as well get clientneeds. This graph is helpful for drawing attention to the external interface of the system and the relation between the various use cases.

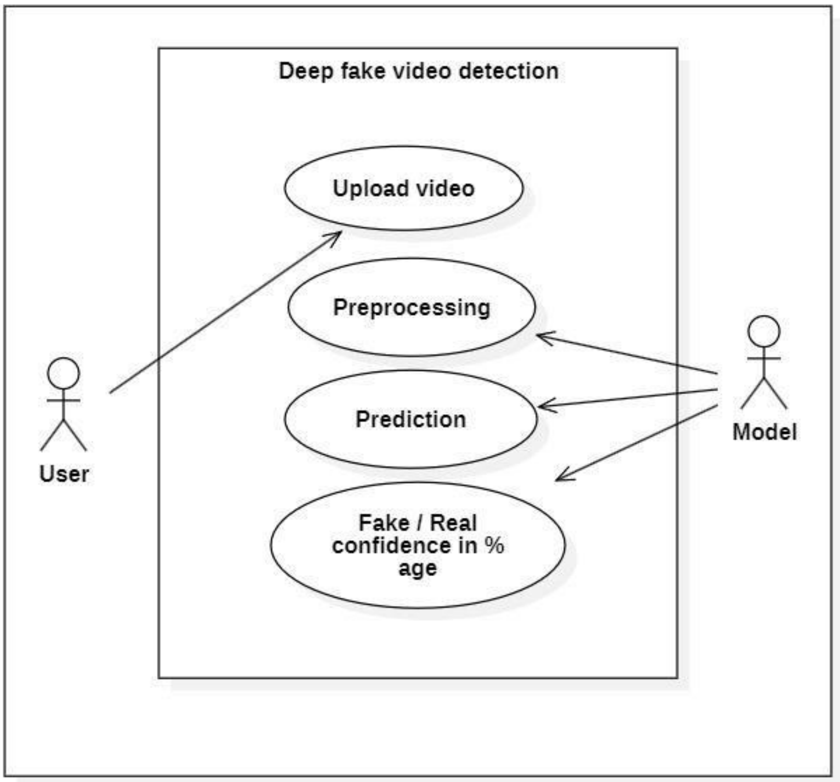


Fig 6.2.1 Use case diagram

6.3 Activity diagram:

An movement graph was probably be an enhanced version of the flowchart that mapped out the flow of data and control between activities. It describes how a set of exercises provides a benefit;this identifies the clustering of activities and the progression of control within the framework. In

particular, movement charts are very useful for now commerce forms and work-flows atmultiplier of thinking

Training Workflow:

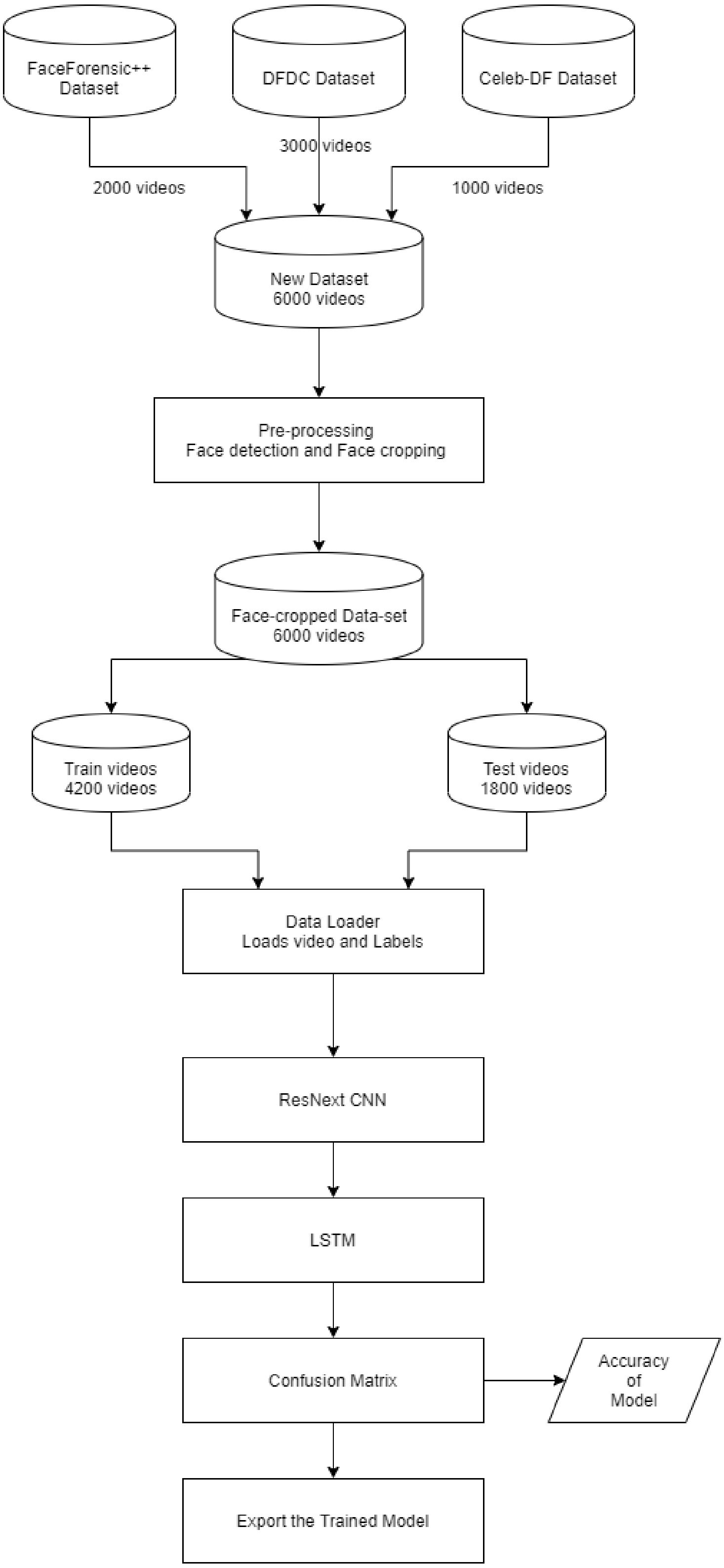


Fig 6.3: Training Workflow

6.4 Sequence Diagram:

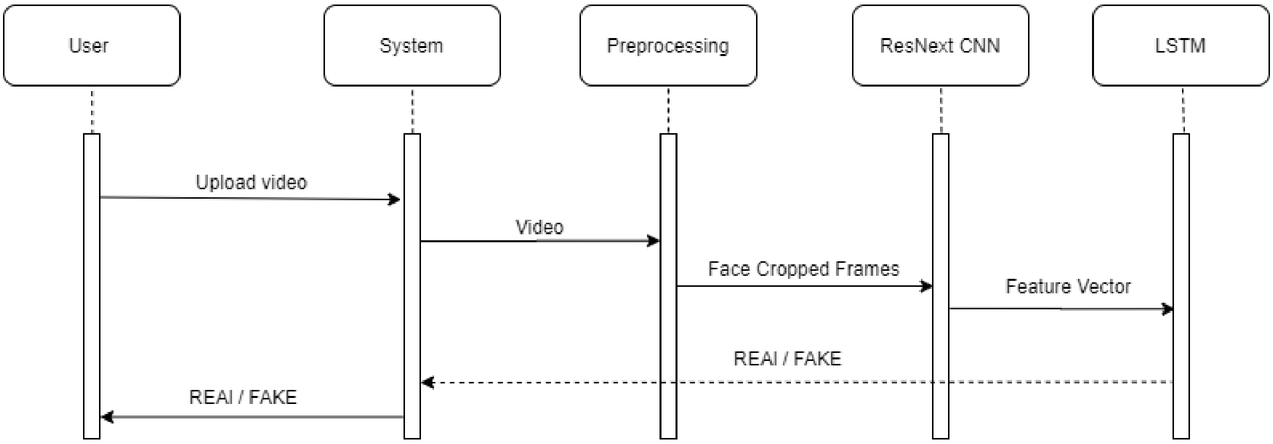


Figure 6.7: Sequence Diagram

6.5 TestingWorkflflow:

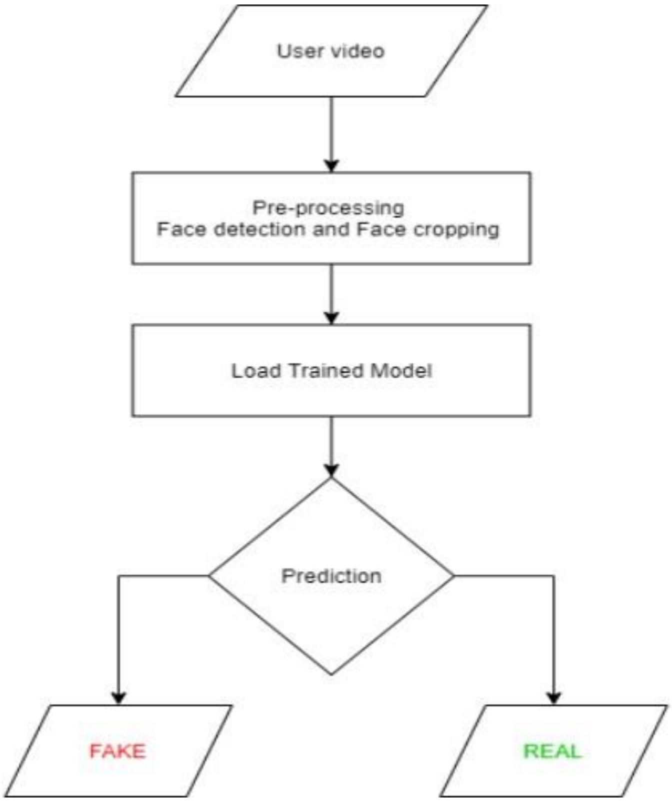


Figure 6.6: Testing Workflflow

6.6 System Architecture :

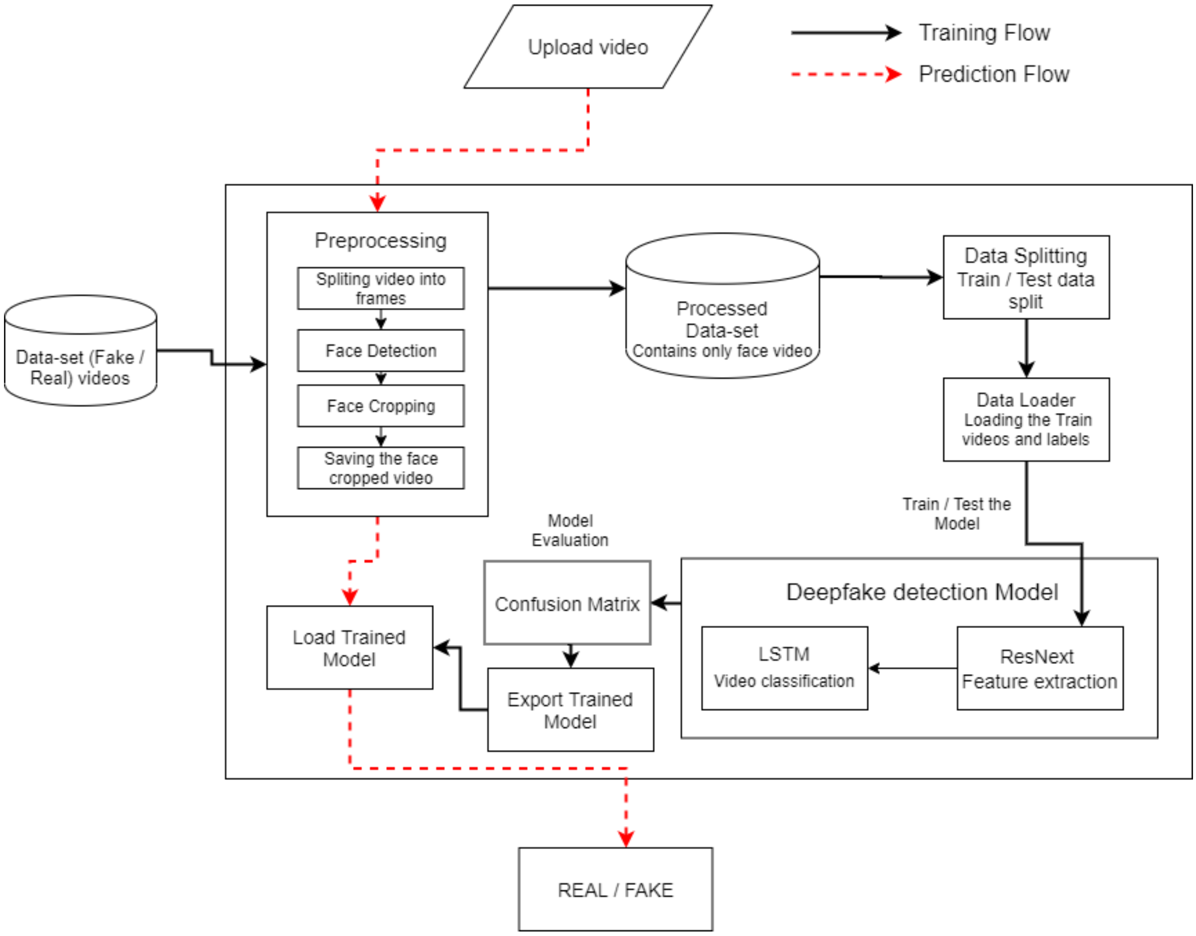


Fig 6.6 System Architecture

We have trained a model in PyTorch for Deepfake detection on this system with an equal number of realand fake videos to avoid bias in the model. The system architecture of the model is shown in the figure. At the development phase, we take a dataset, preprocess the dataset, create a new processed dataset containing only face-cropped videos.

• **Deepfakes video creation**.

First of all, it is very important to understand the creation process of the deepfake for the purpose of detecting the deepfake videos. Most of the tools, including GAN and autoencoders, take a source image and a target video as input. These tools break down the video into frames, detect the face in the video, andreplace the source face with the target face on each frame. These replaced frames further get combined using different pre-trained models. These models also perform a task in quality enhancement of video by removing the left-over traces by the deepfake creation model. This makes the deepfake look realistic in nature. Then, we have used the same approach to detect the deepfakes. These pre-trained neural networks models create very realistic deepfakes, making it almost impossible to spot any difference by naked eyes. However, in a real sense, the tools of creation of deepfakes leave some of the traces or artifacts in the video that may not be noticeable by naked eyes. This paper's motive is to identify these unnoticeabletraces and distinguishable artifacts of these videos and classify them as deepfakes or real videos..

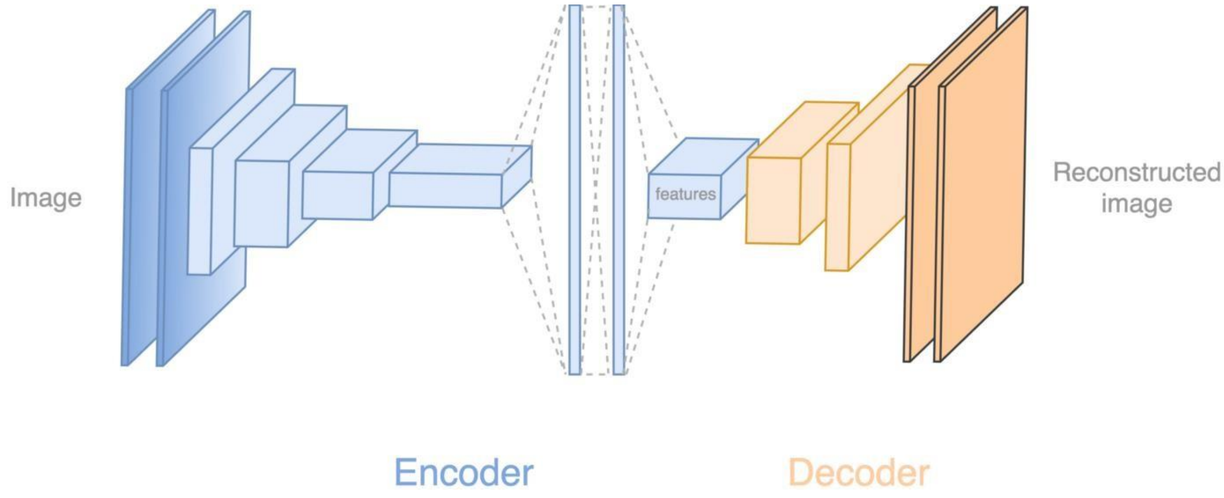


Figure 6.2: Deep fake generation

Figure 6.3: Face Swapped deep fake

Deep fake creation tools:

1.Faceit

2. Deep Face Lab

3. Faceswap

4.Large resolution Face Masked

5. Deepfake Capsule GAN

7. Implementation

There are many examples where deepfake creation technology is used to mislead the people on social media platform by sharing the false deepfake videos of the famous personalities like Mark Zuckerberg Eve of House A.I. Hearing, Donald Trump’s Breaking Bad series where he was introduces as James McGill, Barack Obama’s public service announcement and many more . 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 fifield 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 and GAN pretrained models which have made the destruction of the real 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 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 which generate deepfake videos are being extensively used to create fake pornography of famous people and celebrities.Some of the examples are Brad Pitt, Angelina Jolie nude videos. The deep-looking nature of Deepfake videos makes celebrities and other famous personalities a target for pornографіc material, fake surveillance videos, fake news, and malicious hoaxes. The Deepfakes are much popular in creating political tension. Due to which it becomes very important to detect the deepfake videos and avoid percolation of deepfakes on social media platforms.

7.1 Tools and Technologies Used

7.2.1 Planning

1. OpenProject

7.2.2 UML Tools

1. draw.io

7.2.3 Programming Languages

1. Python3
2. JavaScript

7.2.4 Programming Frameworks

1. PyTorch
2. Django

7.2.5 IDE

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

7.2.6 Versioning control

1. Git

7.2.7 Cloud Services

1. Google Cloud Platform

7.2.8 Application and web servers:

1. Google Cloud Engine

7.2.9 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

Preprocessing Details

• In a python list, we imported all the videos in the directory using glob.

• Reading the videos by cv2.VideoCapture to get the mean number of frames in each video.

• Based on mean value for uniformity 150 is selected as ideal value to create the new data set.

• The video is split into frames, and the frames crop on face location.

• Face-cropped frames once again write to the new video with the use of VideoWriter.

• The new video writes at 30 frames per second and with the resolution of 112 × 112 pixels inMP4format.

• Instead of selecting the random videos, the first 150 frames are written to the new video to make properuse of the LSTM for temporal sequence analysis.

Model Details

• ResNext CNN : Here, the used pre-trained model is Residual Convolution Neural Network. The usedmodel name is resnext50_32x4d(). This model consists of 50 layers and 32 x 4 dimensions. gives the detailed implementation of the model.

Below Figure gives the detailed information about the model

stage	output	ResNeXt-50 (32×4d)
conv1	112×112	7×7, 64, stride 2
conv2	56×56	3×3 max pool, stride 2 $\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128, C{=}32 \\ 1\times1, 256 \end{bmatrix} \times 3$
conv3	28×28	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256, C{=}32 \\ 1\times1, 512 \end{bmatrix} \times 4$
conv4	14×14	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512, C{=}32 \\ 1\times1, 1024 \end{bmatrix} \times 6$
conv5	7×7	$\begin{bmatrix} 1\times1, 1024 \\ 3\times3, 1024, C{=}32 \\ 1\times1, 2048 \end{bmatrix} \times 3$
	1×1	global average pool 1000-d fc, softmax
# params.		**25.0**$\times 10^6$

Fig 6.1: ResNeXt Architecture

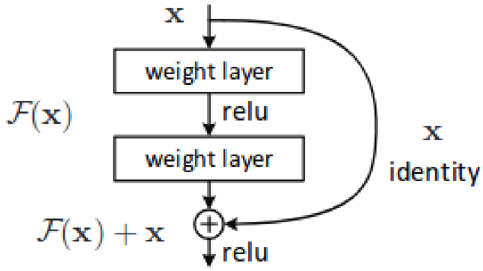


Fig 7.2: Res NeXt Working

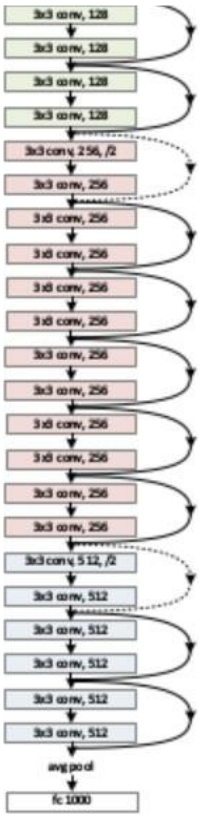


Fig 7.3: Res NeXt Architecture

- **Sequential Layer**: The Sequential layer acts as a container for modules that can be stacked and executed simultaneously. It is employed to organize and store feature vectors generated by the ResNext model in a structured sequence. This sequential arrangement facilitates passing thedata sequentially to the LSTM layer.

- **LSTM Layer**: LSTM (Long Short-Term Memory) is utilized for sequence processing, specifically to detect temporal changes across frames. It accepts 2048-dimensional feature vectors as input. We employ a single LSTM layer with 2048 latent dimensions and 2048 hiddenunits, supplemented by a dropout probability of 0.4. This configuration effectively supports ourobjective by processing video frames sequentially.

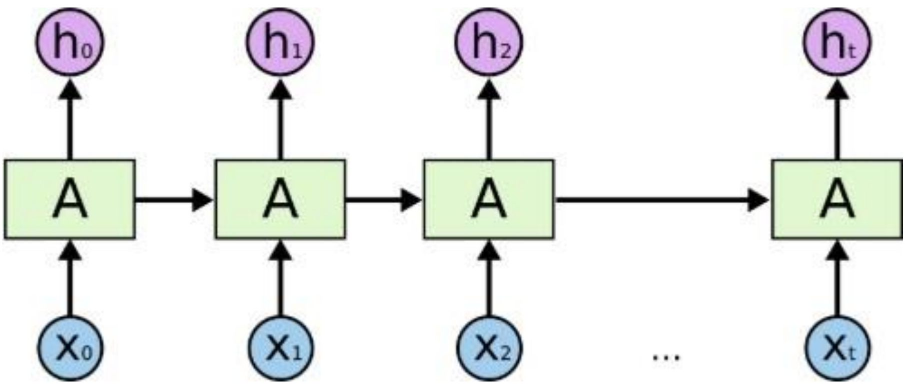


Fig 7.4: LSTM Architecture

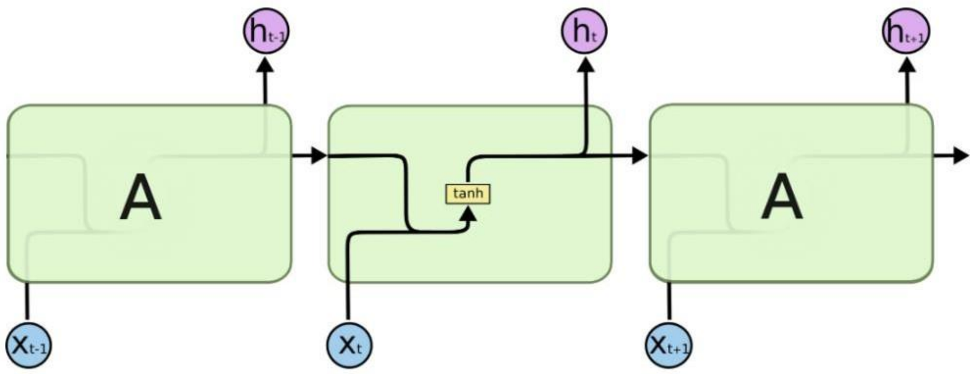


Fig 7.5: Internal Architecture of LSTM

Preprocessing Code:

Importing videos from data set and and calculating average frames per video

```python
#To get the average frame count
import json
import glob
import numpy as np
import cv2
import copy
#change the path accordingly
video_files =  glob.glob('/content/drive/MyDrive/dataset/*.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))
```

```
frames [299, 299, 299, 299, 299, 299, 299, 299, 300, 299, 299, 299, 299, 300, 299, 299, 299, 299, 300, 299, 299, 299]
Total number of videos:  22
Average frame per video: 299.1363636363636
```

Frame Extraction and Processing of Frames:

```python
# 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/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()
```

Preprocessing Output:

```
Collecting face_recognition
  Downloading face_recognition-1.3.0-py2.py3-none-any.whl (15 kB)
Collecting face-recognition-models>=0.3.0 (from face_recognition)
  Downloading face_recognition_models-0.3.0.tar.gz (100.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 100.1/100.1 MB 9.3 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Requirement already satisfied: Click>=6.0 in /usr/local/lib/python3.10/dist-packages (from face_recognition) (8.1.7)
Requirement already satisfied: dlib>=19.7 in /usr/local/lib/python3.10/dist-packages (from face_recognition) (19.24.4)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from face_recognition) (1.25.2)
Requirement already satisfied: Pillow in /usr/local/lib/python3.10/dist-packages (from face_recognition) (9.4.0)
Building wheels for collected packages: face-recognition-models
  Building wheel for face-recognition-models (setup.py) ... done
  Created wheel for face-recognition-models: filename=face_recognition_models-0.3.0-py2.py3-none-any.whl size=100566170 sha256=a237831d76e57100add33f14a
  Stored in directory: /root/.cache/pip/wheels/7a/eb/cf/e9eced74122b679557f597bb7c8e4c739cfcac526db1fd523d
Successfully built face-recognition-models
Installing collected packages: face-recognition-models, face_recognition
Successfully installed face-recognition-models-0.3.0 face_recognition-1.3.0
```

Creating a new folder consisting of only cropped face videos by using the existing data set:

```
create_face_videos(video_files,'/content/drive/My Drive/FF_REAL_Face_only_data/')

No of videos already present  0
100% 22/22 [20:12<00:00, 54.08s/it]
```

Model Training Information:

Train -Test Split: The dataset is divided into training and testing sets using a 70-30 split, comprising 4,200 training videos and 1,800 testing videos. This split ensures balance, with an equal distribution of50% real and 50% fake videos in both sets. Refer to Figure 7.6 for details.

Data Loader: The data loader function is employed to load videos along with their corresponding labels,utilizing a batch size of 4 for efficient processing.

Training: The model undergoes training for 20 epochs, employing a learning rate of 1e-5 (0.00001), aweight decay of 1e-3 (0.001), and the Adam optimizer. The Adam optimizer is chosen for its adaptive learning rate capabilities, which enhance training efficiency by adjusting learning rates for each parameter.

Adam Optimizer: This optimizer is selected for its effectiveness in optimizing model parameters byadapting learning rates based on the gradients of each parameter. It helps in efficiently converging towards minima in the loss landscape

Softmax Layer: The Softmax function acts as a squashing function, compressing outputs into a range of 0 to 1, thereby interpreting them as probabilities. In this context, it serves as the final layer in the neural network, having two output nodes representing classes (REAL or FAKE). The Softmax layer ensures thatthe outputs sum to 1, providing confidence scores (probabilities) for predictions.

Cross Entropy: Cross Entropy is utilized as the loss function, suitable for classification tasks. It measures the disparity between predicted and actual class probabilities, optimizing model parameters tominimize this difference.

.

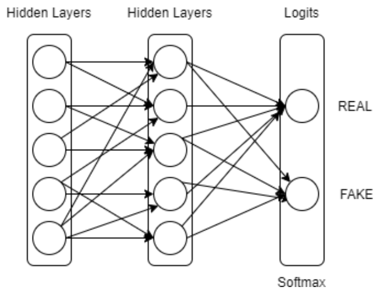


Fig 7.6: Softmax Layer

Confusion Matrix: A confusion matrix is a fundamental tool in evaluating the performance of aclassification model. It provides a detailed summary of prediction results, breaking down correctand incorrect predictions by each class. The matrix highlights how the classifier is confused when making predictions, offering insights into the types and frequency of errors. By analyzing

the confusion matrix, we not only assess the overall accuracy of the classifier but also gain valuable information about its strengths and weaknesses in distinguishing between differentclasses.

Export Model: Upon completing the training process, the model is exported to facilitate its deployment for real-time predictions on new data. Exporting the model allows it to be utilized outside of the training environment, enabling applications such as making predictions on live orstreaming data sources. This step ensures that the trained model can be effectively applied to practical scenarios beyond the training phase.

Model Training code:

Installing face_recognition module and checking if the video is corrupted or not:

```
[2] !pip3 install face_recognition
```

```
Collecting face_recognition
  Downloading face_recognition-1.3.0-py2.py3-none-any.whl (15 kB)
Collecting face-recognition-models>=0.3.0 (from face_recognition)
  Downloading face_recognition_models-0.3.0.tar.gz (100.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 100.1/100.1 MB 7.9 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Requirement already satisfied: Click>=6.0 in /usr/local/lib/python3.10/dist-packages (from face_recognition) (8.1.7)
Requirement already satisfied: dlib>=19.7 in /usr/local/lib/python3.10/dist-packages (from face_recognition) (19.24.4)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from face_recognition) (1.25.2)
Requirement already satisfied: Pillow in /usr/local/lib/python3.10/dist-packages (from face_recognition) (9.4.0)
Building wheels for collected packages: face-recognition-models
  Building wheel for face-recognition-models (setup.py) ... done
  Created wheel for face-recognition-models: filename=face_recognition_models-0.3.0-py2.py3-none-any.whl size=100566170 sha256=c5d3d424ef77e8c00
  Stored in directory: /root/.cache/pip/wheels/7a/eb/cf/e9eced74122b679557f597bb7c8e4c739cfcac526db1fd523d
Successfully built face-recognition-models
Installing collected packages: face-recognition-models, face_recognition
Successfully installed face-recognition-models-0.3.0 face_recognition-1.3.0
```

```python
#THis code is to 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
```

This code loads the video and labels the data(Real or Fake) from the provided csv file:

```
# load the video name and labels from csv
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
class video_dataset(Dataset):
    def __init__(self,video_names,labels,sequence_length = 60,transform = None):
        self.video_names = video_names
        self.labels = labels
        self.transform = transform
        self.count = sequence_length
    def __len__(self):
        return len(self.video_names)
    def __getitem__(self,idx):
        video_path = self.video_names[idx]
        frames = []
        a = int(100/self.count)
        first_frame = np.random.randint(0,a)
        temp_video = video_path.split('/')[-1]
        #print(temp_video)
        label = self.labels.iloc[(labels.loc[labels["file"] == temp_video].index.values[0]),1]
        if(label == 'FAKE'):
          label = 0
        if(label == 'REAL'):
          label = 1
        for i,frame in enumerate(self.frame_extract(video_path)):
          frames.append(self.transform(frame))
          if(len(frames) == self.count):
            break
        frames = torch.stack(frames)
        frames = frames[:self.count]
        #print("length:" , len(frames), "label",label)
        return frames,label
    def frame_extract(self,path):
      vidObj = cv2.VideoCapture(path)
      success = 1
      while success:
          success, image = vidObj.read()
          if success:
              yield image
#plot the image
def im_plot(tensor):
    image = tensor.cpu().numpy().transpose(1,2,0)
    b,g,r = cv2.split(image)
    image = cv2.merge((r,g,b))
    image = image*[0.22803, 0.22145, 0.216989] +  [0.43216, 0.394666, 0.37645]
    image = image*255.0
    plt.imshow(image.astype(int))
    plt.show()
```

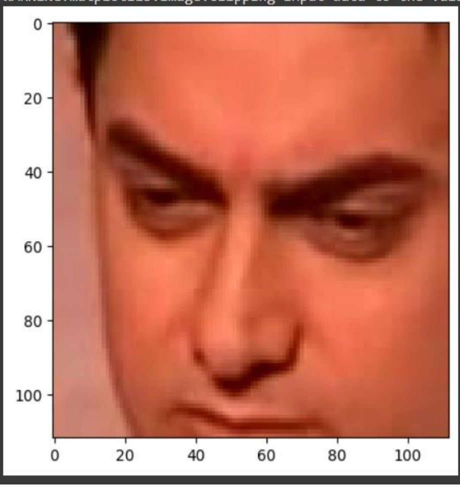
Model Training:

```
[8] #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
```

```python
# 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 :  9
test :  3
TRAIN:  Real: 5  Fake: 4
TEST:  Real: 1  Fake: 2
/usr/local/lib/python3.10/dist-packages/torch/utils/data/dataloader.py:558: UserWarning: This DataLoader will create 4 worker processes in total
  warnings.warn(_create_warning_msg(
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
```



Feature visualization:

```
#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)))
```

Plotting Validation loss:

```
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()
```

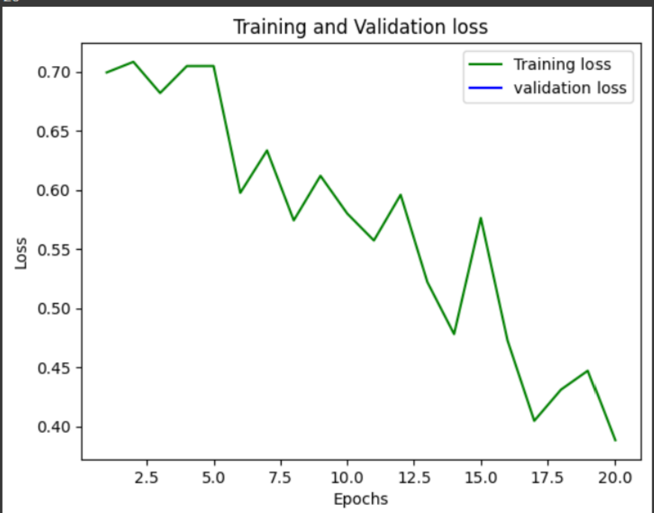
Using Adam optimizer for training the Model:

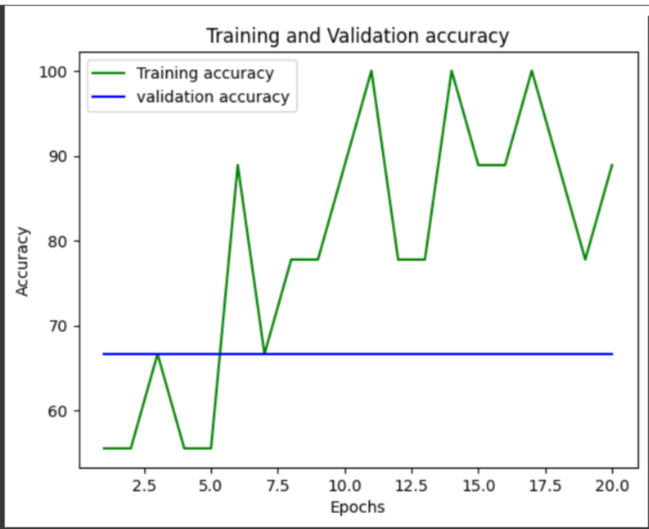
```
from sklearn.metrics import confusion_matrix
#learning rate
lr = 1e-5#0.001
#number of epochs
num_epochs = 20

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:

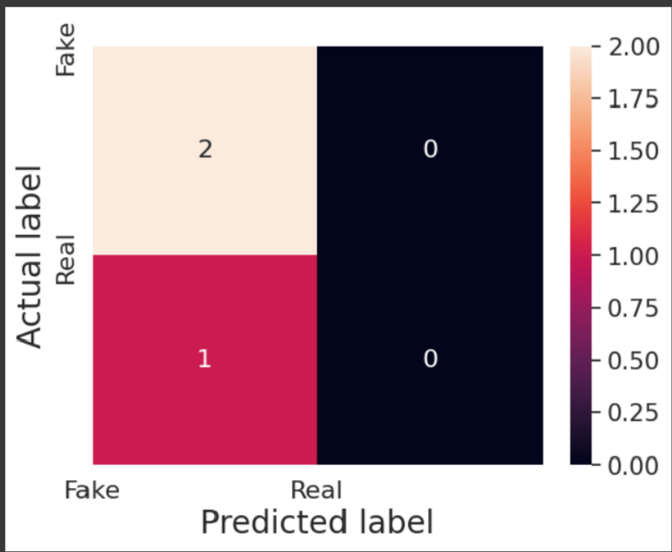
```
/usr/local/lib/python3.10/dist-packages/torch/utils/data/dataloader.py:558: UserWarning: This DataLoader will create 4 worker processes in tot
  warnings.warn(_create_warning_msg(
/usr/local/lib/python3.10/dist-packages/torch/nn/modules/conv.py:456: UserWarning: Plan failed with a cudnnException: CUDNN_BACKEND_EXECUTION_
  return F.conv2d(input, weight, bias, self.stride,
/usr/local/lib/python3.10/dist-packages/torch/autograd/graph.py:744: UserWarning: Plan failed with a cudnnException: CUDNN_BACKEND_EXECUTION_P
  return Variable._execution_engine.run_backward(  # Calls into the C++ engine to run the backward pass
[Epoch 1/20] [Batch 2 / 3] [Loss: 0.699339, Acc: 55.56%]Testing
[Batch 0 / 1]  [Loss: nan, Acc: 66.67%]
Accuracy 66.66666666666667
[Epoch 2/20] [Batch 2 / 3] [Loss: 0.708331, Acc: 55.56%]Testing
[Batch 0 / 1]  [Loss: nan, Acc: 66.67%]
Accuracy 66.66666666666667
[Epoch 3/20] [Batch 2 / 3] [Loss: 0.681980, Acc: 66.67%]Testing
[Batch 0 / 1]  [Loss: nan, Acc: 66.67%]
Accuracy 66.66666666666667
[Epoch 4/20] [Batch 2 / 3] [Loss: 0.704765, Acc: 55.56%]Testing
[Batch 0 / 1]  [Loss: nan, Acc: 66.67%]
Accuracy 66.66666666666667
[Epoch 5/20] [Batch 2 / 3] [Loss: 0.704810, Acc: 55.56%]Testing
[Batch 0 / 1]  [Loss: nan, Acc: 66.67%]
Accuracy 66.66666666666667
[Epoch 6/20] [Batch 2 / 3] [Loss: 0.597536, Acc: 88.89%]Testing
[Batch 0 / 1]  [Loss: nan, Acc: 66.67%]
Accuracy 66.66666666666667
[Epoch 7/20] [Batch 2 / 3] [Loss: 0.633371, Acc: 66.67%]Testing
[Batch 0 / 1]  [Loss: nan, Acc: 66.67%]
Accuracy 66.66666666666667
[Epoch 8/20] [Batch 2 / 3] [Loss: 0.574201, Acc: 77.78%]Testing
[Batch 0 / 1]  [Loss: nan, Acc: 66.67%]
```

```
[Epoch 9/20] [Batch 2 / 3] [Loss: 0.611939, Acc: 77.78%]Testing
[Batch 0 / 1]  [Loss: nan, Acc: 66.67%]
Accuracy 66.66666666666667
[Epoch 10/20] [Batch 2 / 3] [Loss: 0.580033, Acc: 88.89%]Testing
[Batch 0 / 1]  [Loss: nan, Acc: 66.67%]
Accuracy 66.66666666666667
[Epoch 11/20] [Batch 2 / 3] [Loss: 0.557207, Acc: 100.00%]Testing
[Batch 0 / 1]  [Loss: nan, Acc: 66.67%]
Accuracy 66.66666666666667
[Epoch 12/20] [Batch 2 / 3] [Loss: 0.595884, Acc: 77.78%]Testing
[Batch 0 / 1]  [Loss: nan, Acc: 66.67%]
Accuracy 66.66666666666667
[Epoch 13/20] [Batch 2 / 3] [Loss: 0.522022, Acc: 77.78%]Testing
[Batch 0 / 1]  [Loss: nan, Acc: 66.67%]
Accuracy 66.66666666666667
[Epoch 14/20] [Batch 2 / 3] [Loss: 0.478043, Acc: 100.00%]Testing
[Batch 0 / 1]  [Loss: nan, Acc: 66.67%]
Accuracy 66.66666666666667
[Epoch 15/20] [Batch 2 / 3] [Loss: 0.576272, Acc: 88.89%]Testing
[Batch 0 / 1]  [Loss: nan, Acc: 66.67%]
Accuracy 66.66666666666667
[Epoch 16/20] [Batch 2 / 3] [Loss: 0.472658, Acc: 88.89%]Testing
[Batch 0 / 1]  [Loss: nan, Acc: 66.67%]
Accuracy 66.66666666666667
[Epoch 17/20] [Batch 2 / 3] [Loss: 0.404669, Acc: 100.00%]Testing
[Batch 0 / 1]  [Loss: nan, Acc: 66.67%]
Accuracy 66.66666666666667
[Epoch 18/20] [Batch 2 / 3] [Loss: 0.431032, Acc: 88.89%]Testing
[Batch 0 / 1]  [Loss: nan, Acc: 66.67%]
Accuracy 66.66666666666667
```






```
[[2 0]
 [1 0]]
True positive =  2
False positive =  0
False negative =  1
True negative =  0
```




```
Calculated Accuracy 66.66666666666666
```

Model Prediction Details:

1.Loading the Model: The trained model is integrated into the application environment.

2.Preprocessing of New Video: Before prediction, the new video undergoes preprocessing stepsas outlined in sections 8.3.2 and 7.2.2 of the application's documentation. This preprocessing ensures that the video data is properly formatted and optimized for input into the model

3.Prediction Process: Once preprocessed, the video is passed to the loaded model for prediction.The model evaluates the video and provides a prediction indicating whether it is classified as realor fake. Additionally, the model outputs a confidence score associated with this prediction, reflecting the degree of certainty in the classification result.

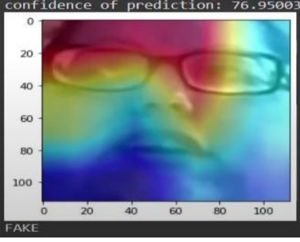
```python
im_size = 112
mean=[0.485, 0.456, 0.406]
std=[0.229, 0.224, 0.225]
sm = nn.Softmax()
inv_normalize =  transforms.Normalize(mean=-1*np.divide(mean,std),std=np.divide([1,1,1],std))
def im_convert(tensor):
    """ Display a tensor as an image. """
    image = tensor.to("cpu").clone().detach()
    image = image.squeeze()
    image = inv_normalize(image)
    image = image.numpy()
    image = image.transpose(1,2,0)
    image = image.clip(0, 1)
    cv2.imwrite('./2.png',image*255)
    return image

def predict(model,img,path = './'):
  fmap,logits = model(img.to('cuda'))
  params = list(model.parameters())
  weight_softmax = model.linear1.weight.detach().cpu().numpy()
  logits = sm(logits)
  _,prediction = torch.max(logits,1)
  confidence = logits[:,int(prediction.item())].item()*100
  print('confidence of prediction:',logits[:,int(prediction.item())].item()*100)
  idx = np.argmax(logits.detach().cpu().numpy())
  bz, nc, h, w = fmap.shape
  out = np.dot(fmap[-1].detach().cpu().numpy().reshape((nc, h*w)).T,weight_softmax[idx,:].T)
  predict = out.reshape(h,w)
  predict = predict - np.min(predict)
  predict_img = predict / np.max(predict)
  predict_img = np.uint8(255*predict_img)
  out = cv2.resize(predict_img, (im_size,im_size))
  heatmap = cv2.applyColorMap(out, cv2.COLORMAP_JET)
  img = im_convert(img[:,-1,:,:,:])
  result = heatmap * 0.5 + img*0.8*255
  cv2.imwrite('/content/1.png',result)
  result1 = heatmap * 0.5/255 + img*0.8
  r,g,b = cv2.split(result1)
  result1 = cv2.merge((r,g,b))
  plt.imshow(result1)
  plt.show()
  return [int(prediction.item()),confidence]
#img = train_data[100][0].unsqueeze(0)
#predict(model,img)
```

Model Prediction Code:

```python
#Code for making prediction
im_size = 112
mean=[0.485, 0.456, 0.406]
std=[0.229, 0.224, 0.225]

train_transforms = transforms.Compose([
                                        transforms.ToPILImage(),
                                        transforms.Resize((im_size,im_size)),
                                        transforms.ToTensor(),
                                        transforms.Normalize(mean,std)])
path_to_videos= ["/content/drive/My Drive/Face_only_data/aabqyygbaa.mp4"]

video_dataset = validation_dataset(path_to_videos,sequence_length = 20,transform = train_transforms)
model = Model(2).cuda()
path_to_model = '/content/drive/MyDrive/checkpoint.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:

```
/content/drive/My Drive/Dataset/abebnhqyzv.mp4
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:21: UserWarning: Implicit dimension choice for softmax has been deprecated.
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
confidence of prediction: 76.95003747940063
```



FAKE

8. SOFTWARE TESTING

Software testing is done before actual software is completely executed. The main objective for doing the software testing is the requirements of the expected output is free from errors and defects.

8.1 Unit Testing:

Unit testing is the first stage to test a module by testing each individual unit the module. Each module or method of a procedure is tested to get the expected output. It helps to fix the defects and errors the module of each unit.

Unit testing can be used to test the many parts or operations that make up the ship detection algorithm in the context of ship detection utilising remote sensing photos. The following are some instances of unit testing in this situation:

To make sure that remote sensing images are properly loaded, normalised, and ready for additional processing, test the image loading and preprocessing functions. Check that preprocessing operations like resizing, colour space conversion, and denoising are carried out correctly and that the photos are read in the expected format.

Implement and test the evaluation metrics for the ship detection algorithm to determine how wellit performs. Use sample test cases that cover a range of circumstances, such as various ship sizes, orientations, and image conditions, when conducting unit testing. Before including them in the overall ship detection system, you may make sure each individual component is accurate and functioning by extensively testing it.

8.2 Integration Testing:

Integration testing involves testing various parts of a system when other components are interconnected or combined. Here, we experimented with the connection between Excel and the SQLite database after exporting it from the upload interface and how the text input interface communicates with the Gemini Pro NLP model and translates those text searches into SQL statements . Integration points also included correctness of a SQL queries as it was expected to check that database returns correct values and correctness of its output displayed to the user. These tests enabled verifying that Framework components are integrated coherently providing seamless end-to-end functionality. The effective integration testing stage demonstrated that framework can accommodate real-life usage situations as needed for a client and ensure that they can pass files, enter queries, and obtain accurate results without errors. This stage was also crucial for approving the usefulness of work within the framework for combining integrations, across the adequacy of integration procedures, and the strength of the framework engineering.

8.3 Acceptance Testing:

Acceptance trying out is achieved with the consequences of system trying out as a starting point. this is completed to check that the anticipated and assumed necessities suit.

Examining the ship detection system to make sure it satisfies the intended requirements and acceptance criteria is the main goal of acceptance testing for ship detection utilising remote sensing pictures. During this testing step, the system's overall performance, including its correctness, dependability, and robustness, are evaluated. It might entail putting the system to the test using a variety of real-world remote sensing photos shot under various circumstances and settings. The objective is to confirm that the system can accurately and efficiently detect ships, reduce false positives, manage a variety of ship sizes and orientations, and work consistently in a variety of geographical locations and weather circumstances. Acceptance testing assists in ensuring that the ship detection system satisfies end-user requirements and is prepared for implementation in practical applications.

8.4 Testing on our System:

After integrated testing, machine checking out is finished. As a end result, each purposeful and nonfunctional trying out are included in the process. The incorporated checking out output is used as the input for system checking out. This checking out is accomplished on the gadget's design or behavior.

9. RESULTS

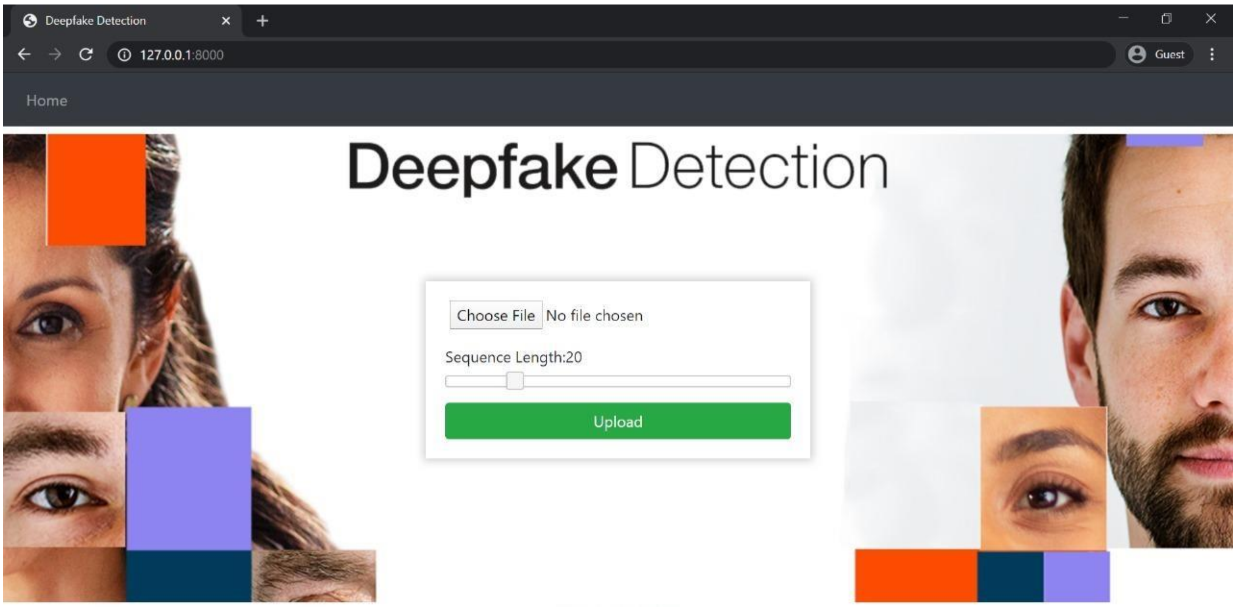


Fig 9.1: Home Page

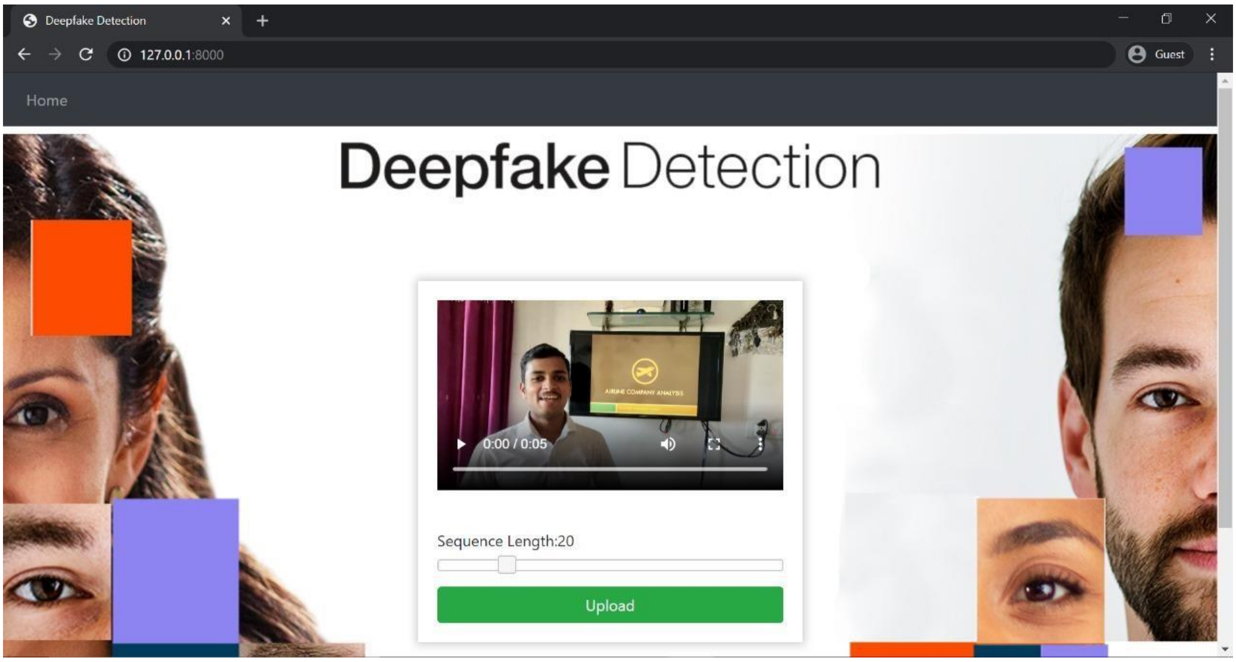


Fig 9.2: Uploading a Real Video

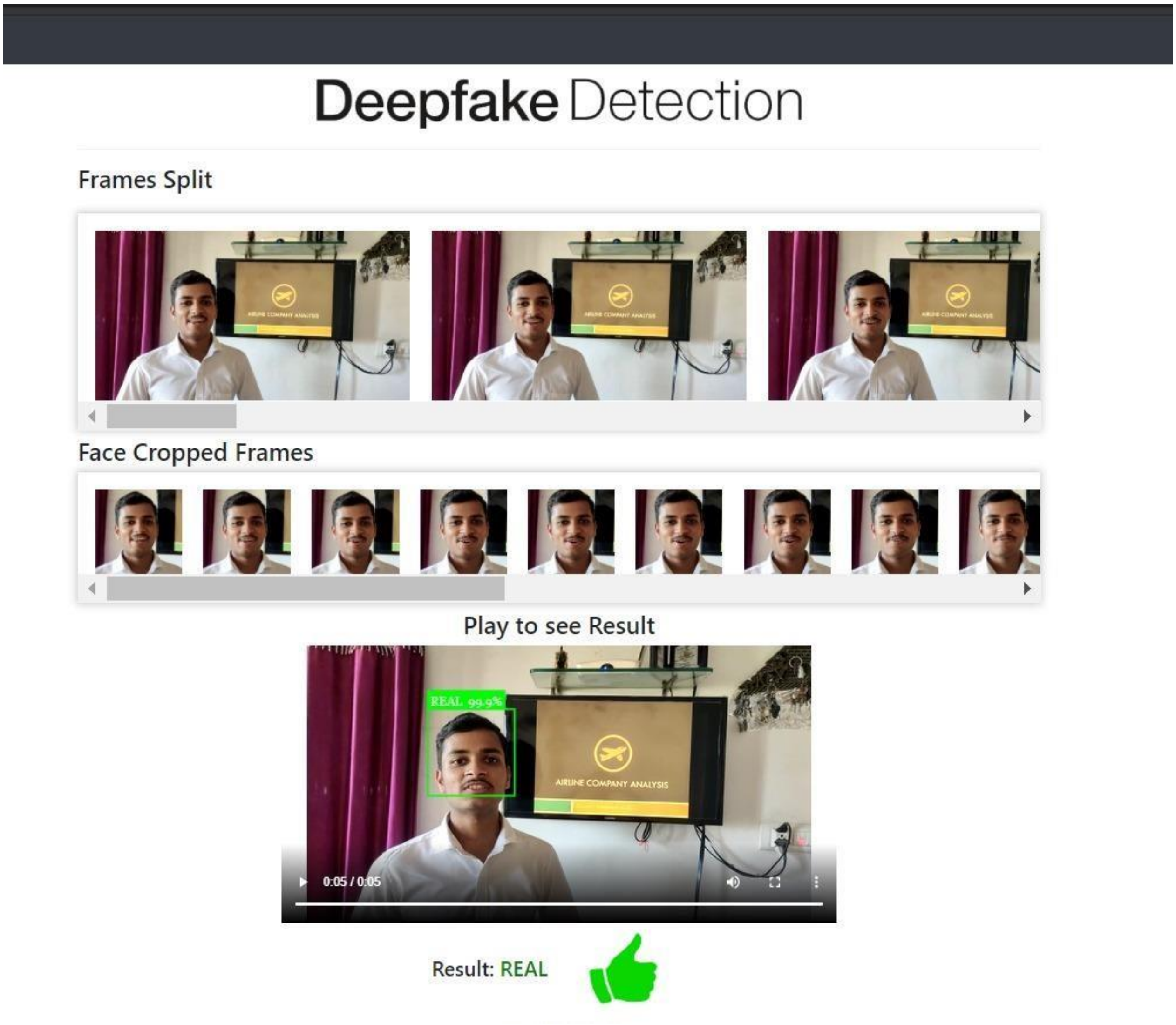


Fig 9.3: Output of Real Video

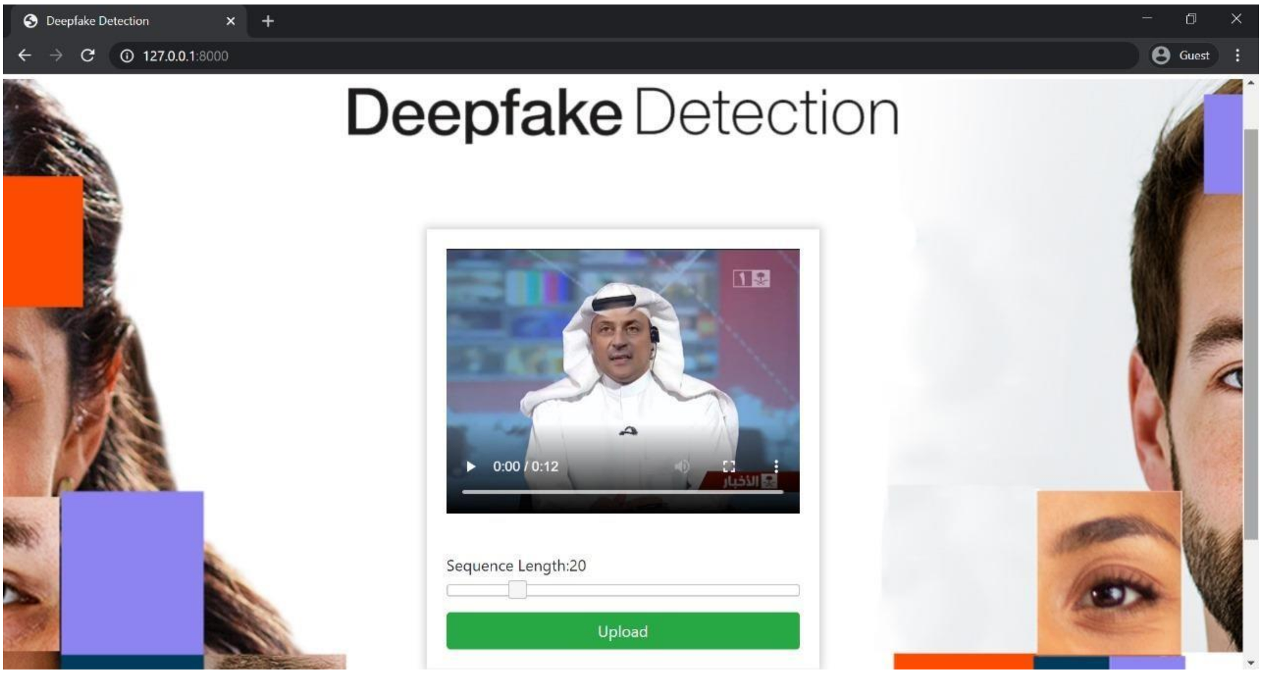


Fig 9.4: Uploading a Fake Video

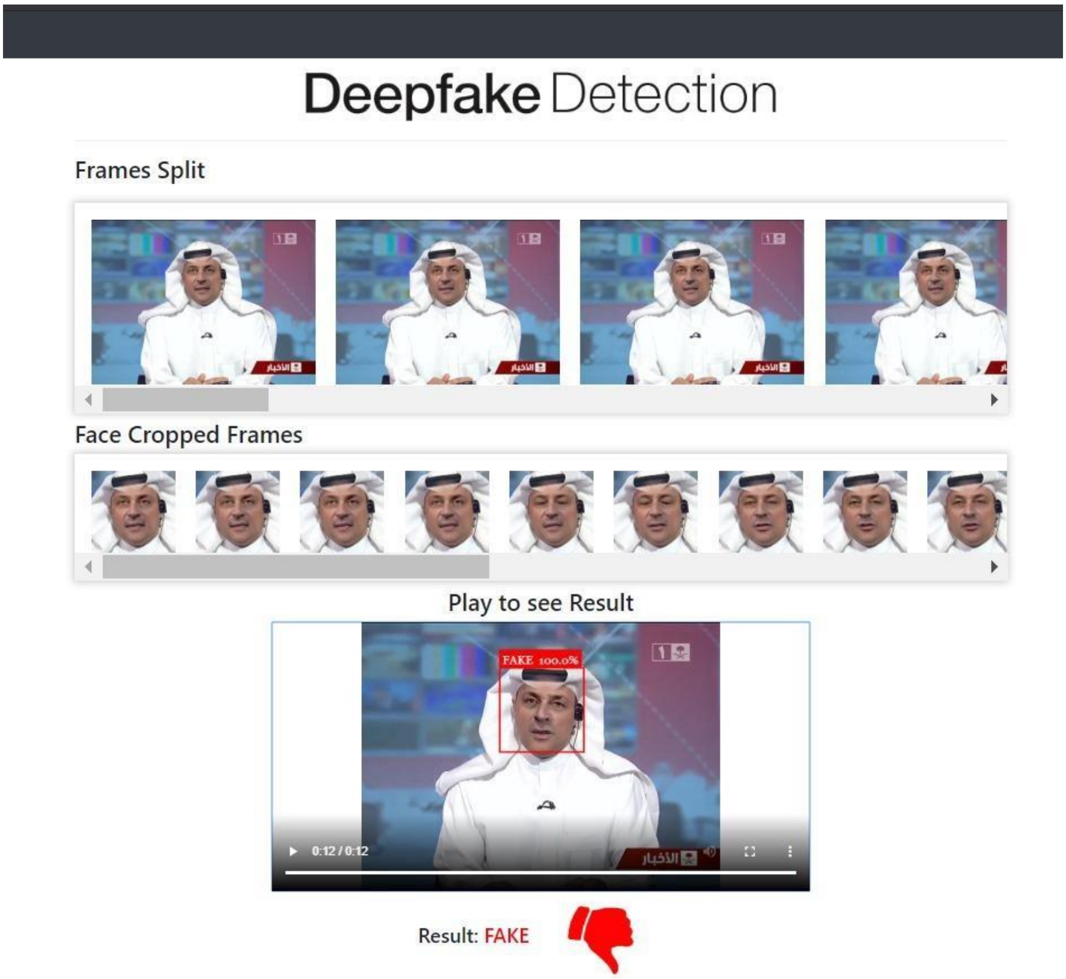


Fig 9.5: Output of Fake video

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

Table 9.6: Trained Model Result

10. CONCLUSION AND FUTURE ENHANCEMENTS

10.1 Conclusion

We have developed a neural network-based approach aimed at classifying videos as either deep fakes or real, providing a confidence score for our predictions. Our method excels at processing video data at a rate of 10 frames per second (1 second of video), achieving high accuracy in its predictions.Our model architecture utilizes a pre-trained ResNext CNN model to extract featuresat the frame level. This CNN model is adept at capturing intricate visual details and patterns within each frame of the video. Subsequently, we employ an LSTM (Long Short-Term Memory)network for temporal sequence processing. The LSTM is pivotal in discerning temporal changesbetween consecutive frames (t and t-1), enabling our model to detect subtle variations overtime.The sequence of frames processed by our model includes intervals such as 10, 20, 40, 60, 80, and 100 frames. This approach ensures that our model comprehensively analyzes the temporal evolution of video content, capturing changes and patterns across varying time scales.By leveraging these techniques, our model effectively integrates spatial and temporal information from videos, thereby enhancing its capability to distinguish between authentic and manipulated video content. This robust methodology enables accurate classification of videos as either deep fakes or real, accompanied by a reliable measure of prediction confidence.

10.2 Future Scope

Every developed system holds potential for enhancements, particularly when built using cutting-edge technology with promising future prospects.

Scaling to a Browser Plugin: The current web-based platform can be further expanded into abrowser plugin, enhancing accessibility and usability for users. This transition would simplifyaccess to the deep fake detection capabilities, making it readily available across various web environments.

Enhancing Detection Capabilities: While the current algorithm focuses on detecting facial deepfakes, there exists an opportunity to broaden its scope to include detection of full-body deep fakes. By extending the algorithm's capabilities beyond facial features, we can enhance its effectiveness in identifying manipulated content across different parts of the body within videos.

11. BIBLOGRAPHY

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

[2] Deepfake detection challenge data set : 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-scaleChallenging 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 gen- erativeadversarial networks. arXiv:1702.01983, Feb. 2017

[10] J. Thies et al. Face2Face: Real-time face capture and reenactment of rgb videos. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2387–2395, June 2016. Las Vegas, NV.

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

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

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

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

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

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

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

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

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

I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic hu- man actions from movies. Proceedings of the IEEE Conference on Computer Vision .andPattern recognition.

[19] Umur Aybars Ciftci, İ lke Demir, Lijun Yin “Detection of Synthetic Portrait Videos usingBiological Signals” in arXiv:1901.02212v2

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

[21] ResNext Model : https://pytorch.org/hub/pytorch_vision_resnext/ accessed on 06 April2020

[22] https://www.geeksforgeeks.org/software-engineering-cocomo-model/ Accessed on 15 April2020

[23] Deepfake Video Detection using Neural Networks http://www.ijsrd.com/articles/IJSRDV8I10860 .pdf

[24] International Journal for Scientific Research and Development

28% Overall Similarity

Top sources found in the following databases:

- 21% Internet database
- 2% Publications database
- Crossref Posted Content database
- 25% Submitted Works database

TOP SOURCES

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Pace University on 2024-05-08 Submitted works	6%
2	Educational Service District 105 on 2022-04-16 Submitted works	4%
3	ijiemr.org Internet	3%
4	ijarcce.com Internet	1%
5	coursehero.com Internet	1%
6	Rutgers University, New Brunswick on 2024-03-03 Submitted works	<1%
7	University of Southern Mississippi on 2023-02-21 Submitted works	<1%
8	gecgudlavalleru.ac.in Internet	<1%
9	ai.googleblog.com Internet	<1%

10	abhijithjadhav.medium.com Internet	<1%
11	github.com Internet	<1%
12	grietinfo.in Internet	<1%
13	ijircce.com Internet	<1%
14	Rutgers University, New Brunswick on 2024-03-03 Submitted works	<1%
15	Jawaharlal Nehru Technological University on 2024-07-11 Submitted works	<1%
16	Army Institute of Technology on 2023-10-26 Submitted works	<1%
17	Army Institute of Technology on 2024-05-27 Submitted works	<1%
18	Griffth University on 2024-05-19 Submitted works	<1%
19	gangw.cs.illinois.edu Internet	<1%
20	open-innovation-projects.org Internet	<1%
21	Rochester Institute of Technology on 2024-05-22 Submitted works	<1%

22	jetir.org Internet	<1%
23	1library.net Internet	<1%
24	exemplarycollegeessays184.blogspot.com Internet	<1%
25	Savitribai Phule Pune University on 2018-04-28 Submitted works	<1%
26	University of Sydney on 2021-10-14 Submitted works	<1%
27	diva-portal.org Internet	<1%
28	Savitribai Phule Pune University on 2017-06-08 Submitted works	<1%
29	University of Limerick on 2023-08-25 Submitted works	<1%
30	vdocuments.mx Internet	<1%
31	geeksforgeeks.org Internet	<1%
32	groundai.com Internet	<1%
33	his.diva-portal.org Internet	<1%

34	ijrpr.com Internet	<1%
35	ijsred.com Internet	<1%
36	Aligarh Muslim University, Aligarh on 2022-12-13 Submitted works	<1%
37	Sadia Syed, Eid Mohammad Albalawi. "Improved Skin Cancer Detectio... Crossref posted content	<1%
38	Shri Guru Gobind Singhji Institute of Engineering and Technology on 2... Submitted works	<1%
39	University of Pittsburgh on 2023-11-17 Submitted works	<1%
40	Visvesvaraya Technological University on 2021-10-01 Submitted works	<1%
41	mdpi.com Internet	<1%
42	California State University, Sacramento on 2021-11-24 Submitted works	<1%
43	General Sir John Kotelawala Defence University on 2023-11-10 Submitted works	<1%
44	Higher Education Commission Pakistan on 2014-12-08 Submitted works	<1%
45	Karunya University on 2024-02-19 Submitted works	<1%

46	MIT-ADT University on 2024-06-04 Submitted works	<1%
47	Savitribai Phule Pune University on 2017-09-27 Submitted works	<1%
48	godok.id Internet	<1%
49	meeraacademy.com Internet	<1%
50	ibm.com Internet	<1%
51	123dok.com Internet	<1%
52	Trident University International on 2018-11-22 Submitted works	<1%
53	fastercapital.com Internet	<1%
54	ijritcc.org Internet	<1%
55	repository.charlotte.edu Internet	<1%
56	sist.sathyabama.ac.in Internet	<1%
57	analyticsvidhya.com Internet	<1%

58	semuelyp.com Internet	<1%