Deep Fake Detection Using CNN and LSTM

A PROJECT REPORT

Submitted by

Mayank Chauhan - 1900520310031

Pawan Sharma - 1900520310044

Ved Prakash Tripathi - 1900520310064

Naman Tripathi - 1900520310039

Under the guidance of

Dr. Divya Sharma

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

In

ELECTRONICS AND COMMUNICATION ENGINEERING

INSTITUTE OF ENGINEERING AND TECHNOLOGY, LUCKNOW

DR. A. P. J. ABDUL KALAM TECHNICAL UNIVERSITY, LUCKNOW

ACKNOWLEDGEMENT

We would like to thank Dr. Divya Sharma, Assistant Professor, Department of Electronics and Communication Engineering, IET Lucknow, for her outstanding knowledge and enormous encouragement as our project guide. Dr. Subodh Wairya, Head of the Department, Department of Electronics and Communication Engineering, is to be thanked for providing us with the necessary resources to complete the project.

We would like to convey our gratitude to all teaching faculty in the Department of ECE, whose ideas during the evaluation process aided us in completing our project. We would like to express our gratitude to the Department of ECE, IET's non-teaching personnel for their invaluable assistance in the completion of our project.

We'd want to express our gratitude to our parents, friends, and classmates for their support throughout the project. Last but not least, we want to express our gratitude to everyone who has helped us complete this project successfully, whether directly or indirectly.

.

Mayank Chauhan - 1900520310031
Ved Prakash Tripathi - 1900520310064
Naman Tripathi - 1900520310039
Pawan Sharma - 1900520310044

CERTIFICATE

This is to certify that the project report "Deep fake detection using CNN and LSTM" submitted by Mayank Chauhan, Pawan Sharma, Ved Prakash Tripathi and Naman Tripathi in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Electronics and Communication Engineering of Institute of Engineering and Technology, Lucknow is a record of bona fide work carried out under my guidance and supervision.

Date:

Dr. Subodh Wairya
Head of the Department (ECE)

Dr. Divya Sharma
(Project Guide)

CANDIDATE DECLARATION

We, Mayank Chauhan, Pawan Sharma, Ved Prakash Tripathi and Naman Tripathi, final semester B.Tech. students in the Department of Electronics and Communication Engineering at IET Lucknow, hereby declare that the project work "Deepfake detection Using CNN and LSTM" was completed and submitted in partial fulfilment of the requirements for the award of Bachelor of Technology in Electronics and Communication Engineering during the academic year 2019-2023.

Date :

Place : IET Lucknow

Mayank Chauhan - 1900520310031

Ved Prakash Tripathi- 1900520310064

Naman Tripathi - 1900520310039

Pawan Sharma - 1900520310044

ABSTRACT

The rapid advancement in computational power has greatly amplified the capabilities of deep learning algorithms, enabling the creation of remarkably realistic synthetic videos known as deep fakes. These fabricated videos, featuring seamlessly swapped faces, have given rise to concerns regarding their potential use in political destabilization, falsification of terrorism events, dissemination of revenge porn, and extortion schemes. In our research, we present a novel deep learning-based methodology that effectively discerns AI-generated fake videos from genuine ones. Our approach proficiently identifies manipulated deep fakes involving face replacement and reenactment. Employing a Res-Next Convolutional Neural Network, we extract frame-level features, which are then utilized to train a Long Short-Term Memory (LSTM)-based Recurrent Neural Network (RNN). This RNN classifies videos, determining whether they have undergone any form of manipulation, thereby distinguishing between deep fakes and authentic videos. To ensure our model's real-world applicability and enhance its performance, we evaluate its efficacy on a substantial and diverse dataset carefully curated by blending various available datasets, such as Face-Forensic++[1], the Deepfake Detection Challenge[2], and Celeb-DF[3]. Additionally, we demonstrate that our system achieves competitive results through a straightforward and robust approach.

In summary, our research presents an innovative deep learning-based method for identifying AI-generated deep fake videos. By combining advanced neural networks and carefully curated datasets, we aim to tackle the challenges posed by deep fakes and contribute to the development of robust and reliable deep fake detection systems.

Keywords:

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

CONTENT

List of Figures

List of Abbreviations

Abbreviation	Meaning
CNN	Convolutional Neural Network
LSTM	Long Short Term Memory
RNN	Recurrent Neural Network
GAN	Generative Adversarial Network
PPG	Photoplethysmography
SVM	Support Vector Machine
GPU	Graphics Processing Unit
GCP	Google Cloud Platform
DFD	Deep Fake Detection
ReLU	Rectified Linear Unit

List of Tables

Chapter 1

Introduction

1.1 Project Idea

In today's rapidly expanding landscape of social media platforms, the emergence of Deepfakes poses a significant threat in the realm of artificial intelligence. Deepfakes, which involve the realistic swapping of faces, have become a prominent tool for orchestrating various malicious activities. These activities encompass a wide range of scenarios, including the creation of political turmoil, the fabrication of terrorism events, the dissemination of revenge pornography, and the extortion of individuals. One distressing example involves the unauthorized creation and circulation of nude videos purportedly featuring well-known figures such as Brad Pitt and Angelina Jolie.

Detecting the distinction between deepfake videos and authentic footage has become increasingly crucial. In this endeavor, we harness the power of artificial intelligence to combat the challenges posed by AI-generated deepfakes. Deepfakes are typically crafted using sophisticated tools such as FaceApp[4] and Face Swap[5], which leverage pre-trained neural networks like Generative Adversarial Networks (GANs) or Autoencoders to create these deceptive videos.

To address this issue, our proposed method employs a Long Short-Term Memory (LSTM) based artificial neural network. This network is specifically designed to analyze the sequential and temporal characteristics of video frames. Additionally, we incorporate a pre-trained Res-Next Convolutional Neural Network (CNN) to extract high-level features at the frame level. By combining these two components, our approach aims to effectively identify and differentiate between deepfakes and pristine videos.
The primary component of our system is the ResNext Convolutional Neural Network (CNN), which plays a crucial role in extracting features at the frame level from videos. These extracted features serve as valuable input for training our Long Short Term Memory (LSTM) based Recurrent Neural Network (RNN). The purpose of this trained RNN is to accurately classify videos as either Deepfake or real, based on the learned patterns and characteristics.

To ensure that our model performs well in real-time scenarios and effectively detects manipulation, we put considerable effort into training it using a diverse range of datasets. These datasets include FaceForensic++[1], Deepfake Detection Challenge[2], and Celeb-DF[3]. By combining and balancing these datasets, we can expose the model to various types of manipulation and genuine

videos, enabling it to learn the distinctive features and behaviors associated with both Deepfakes and real videos.
The training process involving these datasets helps our system to generalize its understanding of manipulation techniques and real video characteristics. This approach allows the model to make accurate predictions and classifications when confronted with new, unseen videos. By leveraging a combination of diverse datasets and training techniques, we strive to enhance the robustness and reliability of our system in detecting Deepfakes and preserving the integrity of video content.

Further to make the ready to use for the customers, we have developed a front end application where the user the user will upload the video .

1.2 Motivation of the Project

The rapid advancement of mobile camera technology and the widespread use of social media and media sharing platforms have significantly simplified the creation and dissemination of digital videos. This is further amplified by the remarkable progress in deep learning technology, which has enabled the development of previously unimaginable capabilities. Notably, generative models have emerged as a prominent example, capable of producing highly realistic images, speech, music, and even videos. These models have found application in diverse fields, ranging from enhancing accessibility through text-to-speech technologies to aiding in the generation of training data for medical imaging purposes.

As with any transformative technology, the rise of deep generative models has brought forth new challenges. Specifically, the emergence of "deep fakes," which are manipulated video and audio clips created using these models. Since their initial appearance in late 2017, numerous open-source methods and tools for generating deep fakes have emerged, resulting in a growing volume of synthesized media content. While some of these creations may be intended for amusement, others have the potential to inflict harm upon individuals and society as a whole.

In recent times, the prevalence of fake videos and their level of realism has surged due to the widespread availability of editing tools and the escalating demand for domain expertise in this field..

The proliferation of Deepfakes across various social media platforms has become alarmingly common, resulting in the dissemination of misleading information and spamming. Just envision a Deepfake video featuring our prime minister declaring a war against neighboring countries, or a

fabricated video of a renowned celebrity verbally abusing their fans. The potential consequences of such Deepfakes are terrifying, as they can pose threats, mislead the general public, and incite widespread panic.

In order to combat this growing issue, the detection of Deepfakes has become imperative. We propose a novel deep learning-based method specifically designed to effectively differentiate between AI-generated fake videos (Deep Fake Videos) and authentic videos. The development of such technology is of utmost importance to detect and identify Deepfakes accurately, thereby preventing their propagation and dissemination on the internet.

1.3 Literature Survey

To address the issue of face warping artifacts in deepfake videos, a study conducted by Face Warping Artifacts[6] proposed an approach that involves comparing the generated face regions and their surrounding areas using a dedicated Convolutional Neural Network (CNN) model. The researchers identified two distinct types of face artifacts in their work.

Their methodology is built upon the observation that existing deepfake algorithms often generate images with limited resolutions. Consequently, these images require additional transformations to align with the faces being replaced in the source video. However, it is worth noting that their approach does not account for the temporal analysis of frames, focusing primarily on the spatial comparison of specific facial regions and their context.

The paper titled "Detection by Eye Blinking" [7] introduces a novel method for deepfake detection based on the analysis of eye blinking as a crucial parameter. The approach utilizes a Long-term Recurrent Convolution Network (LRCN) for the temporal analysis of cropped eye blinking frames. However, it is acknowledged that as deepfake generation algorithms have become more advanced, relying solely on eye blinking as an indicator may no longer be sufficient. Additional parameters, such as teeth enhancement, facial wrinkles, incorrect placement of eyebrows, and other visual inconsistencies, need to be considered for effective deepfake detection.

Another approach presented in the paper "Capsule Networks to Detect Forged Images and Videos" [8] focuses on using capsule networks to detect manipulated and forged images and videos in various scenarios, including replay attack detection and computer-generated video detection. While the model showed promising performance in their dataset, the use of random noise during training

may impact its effectiveness on real-time data. In contrast, our proposed method aims to be trained on noiseless and real-time datasets for improved performance in real-world scenarios.
The paper on "Recurrent Neural Network (RNN) for Deepfake Detection" [9] utilizes RNN for sequential processing of frames combined with an ImageNet pre-trained model. However, their approach is based on the HOHO dataset, which consists of a small number of videos and may not perform optimally on real-time data. In contrast, our proposed method aims to train on a larger number of real-time data to enhance its performance and generalizability.

The paper titled "Synthetic Portrait Videos using Biological Signals" [11] focuses on extracting and analyzing biological signals from facial regions in both genuine and deepfake portrait video pairs. Spatial coherence and temporal consistency are calculated through various transformations to capture unique signal characteristics present in feature vectors and photoplethysmography (PPG) maps obtained from the videos. These extracted features are then used to train a probabilistic Support Vector Machine (SVM) and a Convolutional Neural Network (CNN).

To determine the authenticity of a video, average authenticity probabilities obtained from the trained models are utilized. If the average probability suggests a high likelihood of the video being a deepfake, it is classified as such; otherwise, it is considered pristine. The Fake Catcher approach demonstrates high accuracy in detecting fake content, regardless of the generator used, the content itself, or the resolution and quality of the video. However, the absence of a discriminator in their findings presents a challenge in preserving biological signals. Developing a differentiable loss function aligned with the proposed signal processing steps requires further exploration and development.

Chapter 2

Problem Definition and scope

2.1 Problem Statement

For many years, digital images and videos have been susceptible to manipulation using visual effects, but recent advancements in deep learning have greatly improved the authenticity and accessibility of fake content. This has given rise to the phenomenon known as deep fakes, where AI-generated media is created with remarkable realism.

The ease of creating deep fakes using artificial intelligence tools has raised concerns about their potential misuse. These fabricated videos have been utilized in various instances to generate political unrest, fabricate terrorism incidents, disseminate revenge pornography, and exploit individuals through blackmail [12]. Consequently, it is of utmost importance to develop robust techniques for detecting deep fakes and preventing their dissemination on social media platforms.In this context, we have taken a step forward by utilizing an LSTM-based artificial neural network for deep fake detection. The LSTM architecture, with its ability to analyze sequential data and capture temporal dependencies, proves valuable in distinguishing between genuine videos and deep fakes. By training the neural network on diverse datasets and exposing it to various manipulation techniques, we aim to enhance its ability to accurately identify and flag deep fakes.
The detection of deep fakes is a vital step in combating the negative impacts associated with their dissemination. By leveraging the power of artificial neural networks, such as the LSTM model, we strive to mitigate the potential harm caused by deep fakes and protect individuals, societies, and the integrity of digital content.
.

2.1.1 Goals and objectives

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

2.1.2 Statement of scope

The availability of numerous tools for creating deep fakes highlights the urgency for effective deep fake detection mechanisms. Our proposed approach for detecting deep fakes will make a significant contribution in mitigating their spread across the internet. We intend to develop a web-based platform that allows users to upload videos and classify them as either fake or real. This project has the potential to be scaled up, evolving from a web-based platform to a browser plugin that can automatically detect deep fakes.

Furthermore, integrating this project into major applications such as WhatsApp and Facebook can facilitate pre-detection of deep fakes before they are shared with other users. By incorporating this technology, users will have an added layer of protection against the dissemination of manipulated media.

In terms of software description, the input size, bounds, and validation will be specified to ensure accurate detection. An input dependency analysis will be conducted to identify the relevant factors that contribute to the deep fake detection process. An input/output state diagram will be created to illustrate the flow of data within the system. Additionally, the major inputs and outputs of the software will be defined, focusing on the core functionalities without delving into implementation details.

2.2 Major Constraints

User: The application allows users to determine whether an uploaded video is genuine or a deep fake, providing them with the prediction and the model's confidence level for that prediction.

Prediction: Users can view the uploaded video and observe the output displayed on the faces within the video, along with the corresponding confidence score generated by the model.

User-friendly Interface: To cater to user preferences for simplicity, the application incorporates an intuitive and user-friendly interface. It includes a browse tab that enables users to easily select the video they want to process, reducing complexity and enhancing the overall user experience.

Cross-platform Compatibility: Recognizing the importance of accessibility and reaching a wider audience, the application ensures cross-platform compatibility. As a server-side application, it can run on any device with a web browser, making it accessible across different platforms and devices.

2.3 Methodologies of Problem solving

2.3.1 Analysis

Solution Requirement:

After carefully analyzing the problem statement, we have determined the feasibility of the proposed solution. Extensive research has been conducted, including reviewing various research papers as mentioned in section 3.3. Following the feasibility analysis, the next step is to gather and analyze the dataset.

Data Set Analysis:

We have analyzed the dataset using different training approaches, such as training the model solely on fake or real videos. However, it was observed that this approach could introduce biases into the model, leading to inaccurate predictions. After thorough research, it has been concluded that balanced training of the algorithm is the most effective way to avoid bias and variance, resulting in improved accuracy.

Solution Constraints:

During the evaluation of the solution, several constraints have been considered, including cost, processing speed, requirements, level of expertise, and availability of equipment. These factors are crucial in determining the practical implementation and scalability of the solution.

Parameter Identified

1. Eyes Blinking
2. Enhancement in Teeth
3. Enhanced distance for Eyes
4. Moustaches different sizes
5. Double edges, eyes, ears, nose
6. Segmentation in Iris
7. Faces Wrinkles
8. Inconsistent and variation in head pose
9. Face with different angle
10. Tone of Skin
11. Expressions of FAce
12. Lighting
13. Different Pose

14. Double chins
15. Differ Hairstyle
16. Higher cheek bones

2.3.2 Design

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

2.3.3 Development

After careful analysis, we have made the decision to utilize the PyTorch framework in conjunction with the Python programming language for our project. PyTorch has been chosen for its excellent support for CUDA, which allows us to leverage the power of Graphics Processing Units (GPUs) for accelerated computations. This GPU support is crucial for training deep learning models efficiently. PyTorch also offers a high degree of flexibility and customization, allowing us to design and implement our deep fake detection algorithms with ease. Its intuitive interface and extensive documentation make it a popular choice for deep learning practitioners.

In terms of infrastructure, we have opted to utilize the Google Cloud Platform (GCP) for training our final model on a large dataset. GCP provides a scalable and reliable environment for handling big data and conducting computationally intensive tasks. Leveraging the resources and capabilities of GCP allows us to train our model efficiently and achieve better performance.

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.

Table 2.1: Hardware Requirements

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

2.7 Software Resources Needed

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 have chosen the Spiral model as our software development model for this project. The Spiral model focuses on the collaboration and interaction among team members, as well as effective risk management.

The reason for selecting the Spiral model is that it allows for quick and frequent changes to be made throughout the development process. This is achieved by conducting regular evaluations and assessments of the product against the desired outcomes. Given the nature of our application, which involves developing various modules, the Spiral model is well-suited for managing such complexity.

One of the advantages of the Spiral model is that it enables active client involvement throughout the project. Clients have the opportunity to prioritize features, participate in iteration planning, and engage in review sessions. This transparency allows clients to witness the progress and provide valuable feedback. However, it is important for clients to understand that they will be witnessing a work in progress and that this level of transparency comes with the benefit of their input.

Considering the risks involved in our project, the Spiral model's inherent ability to handle risks effectively makes it a suitable choice for our product development. By incorporating risk analysis and mitigation strategies at each iteration, we can address potential issues proactively and ensure a more successful outcome..

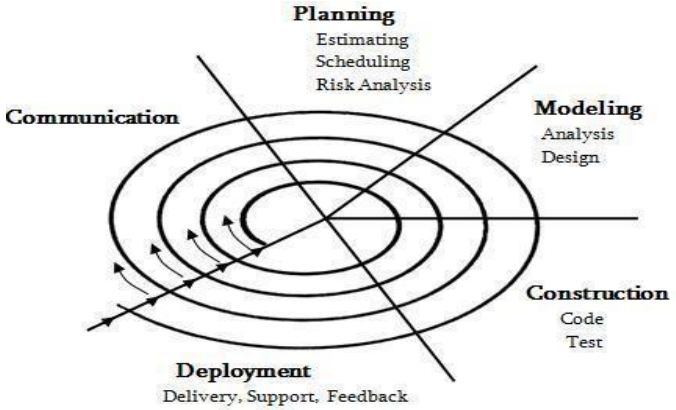

Figure 3.1: Spiral Methodology SDL

3.1.1 Reconciled Estimates

1. Cost Estimate : Rs 11,600

Table 3.1: Cost Estimation

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

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

3.1.2 Cost Estimation using COCOMO(Constructive Cost) Model

Since we have small team , less-rigid requirements, long deadline we are using the organic COCOMO[13] model.

1. Efforts Applied : The term "efforts applied" refers to the amount of labor or work required to complete a specific task or project. It quantifies the level of human resources, typically measured in person-months units, that will be needed to accomplish the desired objectives. Person-months represent the work completed by one person in a month, taking into account the individual's level of effort and the duration of the project..

$\text{EffortApplied}(E) = a_b(\text{KLOC})^{bb}$

$E = 2{:}4(20{:}5)^{1:05}$

$E = 57{:}2206\text{PM}$

2. Development Time: Development time refers to the duration required to complete a particular job or project. It is a measure of the time taken to accomplish the development tasks and is typically

measured in units of time such as weeks or months. The development time is directly proportional to the effort invested in the project.

The duration of the development time depends on various factors, including the complexity of the project, the size of the team working on it, the availability of resources, and any potential challenges or obstacles that may arise during the development process.

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

$D = 2:5(57:2206)^{0:38}$

$D = 11:6M$

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

$PeopleRequired(P) = E/D$

$P=57:2206/11:6$

$P = 4:93$

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

3.2.1 Risk Identification

Quality of Facial Pictures: Using a face detection library to scrape facial pictures from videos may provide a shortcut, but it may impact the quality of the final result. The accuracy of the face detection library and the clarity of the scraped images can affect the performance of the training process.

Multiple Persons in Picture Frames: Removing picture frames that contain more than one person is important to ensure accurate training and avoid confusion in the model. Failure to remove such frames may lead to incorrect facial recognition and classification.

Sufficient Video Footage: Having an abundance of video footage is crucial for training a robust model. Extracting facial pictures that encompass different poses, face angles, and facial expressions allows the model to learn and generalize better. Insufficient video footage may result in limited training data, which can affect the accuracy and performance of the model.

Resemblance of Persons: Similarities in face shape or other facial features between the persons being trained can be advantageous for the accuracy of the model. Incorporating facial images that exhibit resemblance can help improve the recognition and classification capabilities of the model.

To mitigate these risks:

1. Carefully evaluate the quality and accuracy of the face detection library used for scraping facial pictures.
2. Prioritize using high-quality video footage to ensure clear and representative facial images.
3. Implement a thorough filtering process to remove picture frames containing multiple persons.
4. Collect a diverse range of facial pictures to cover different poses, angles, and expressions.
5. Consider the resemblance factor between the persons being trained and include relevant facial images.
6. Perform regular quality checks and validation throughout the training process to identify and address any issues promptly.

3.2.2 Risk Analysis

1.Gaussian filter application:

Risk: Over-smoothing the mask boundary area may result in loss of detail and reduced overall quality of the deepfake.
Mitigation: Careful adjustment of filter parameters and thorough testing can help strike a balance and avoid undesirable blurring effects.

2.Mask expansion/contraction configuration:

Risk: Incorrect adjustments may lead to misalignment or unnatural appearance of the mask.
Mitigation: Implement precise controls, offer visual feedback, and provide options for fine-tuning to mitigate risks and prevent noticeable artifacts.

3.Mask shape control:

Risk: Drastic modifications may result in distorted or unrealistic facial proportions.
Mitigation: Provide intuitive tools and guidelines, while setting limits on extreme shape modifications to ensure natural-looking deepfakes.

A comprehensive risk analysis involves testing, user feedback, and continuous improvement to ensure high-quality deepfakes with minimal artifacts or inconsistencies.

Table 3.2: Risk Description

ID	Risk Description	Probability	Impact		
			Schedule	Quality	Overall
1	Does it over blur comparing with other non-facial areas of the video?	Low	Low	High	High
2	Does it flick?	High	Low	High	High
3	Does it have a change in the tone of skin near the edge of face?	Low	High	High	Low
4	Does the video exhibit double chin, double eyebrows, double edges on the face?	High	Low	High	Low
5	Does the video exhibit flickering or blurring when the face is partially obscured by hands or other objects?	High	High	High	High

Table 3.3: Risk Probability definitions

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

3.3 Project Schedule

3.3.1 Project task set

Major Tasks in the Project stages are

Task 1: Data-set gathering and analysis

This task consists of downloading the dataset. Analysing the dataset and mak-ing 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

During the pre-processing phase, a new dataset is created that exclusively contains videos with cropped faces. This process involves extracting the facial region from each frame of the original videos and saving only the cropped face frames. The purpose of this step is to focus solely on the facial features, disregarding the rest of the video content

Task 4: Module 2 implementation

Module 2 implementation consists of implementation of DataLoader for load-ing 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 mode

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

Software requirement specification

4.1 Introduction

4.1.1 Purpose and Scope of Document

This project proposal presents a comprehensive plan for developing a neural network-based solution for detecting Deepfake videos. The primary audience for this document comprises current and future developers involved in Deepfake video detection using neural networks, as well as the project sponsors. The plan encompasses various key components, including a concise overview of the system's functionality, an in-depth exploration of the project's scope from the perspective of the "Deepfake video detection" team (including mentors), and the utilization of use case diagrams, data flow diagrams, and activity diagrams to provide a holistic understanding of the system. Additionally, the document outlines both the functional and non-functional requirements that the solution must meet, addresses potential project risks, and proposes effective risk mitigation strategies. Furthermore, it delineates the development process that will be followed and highlights the essential metrics and measurements that will be employed to gauge project progress and success.

4.1.2 Use Case View

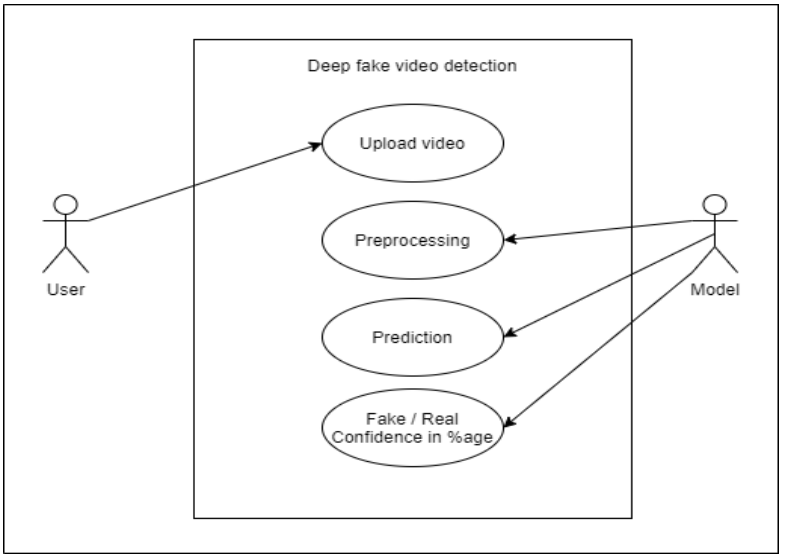

Figure 4.1: Use case diagram

4.2 Functional Model and Description

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

4.2.1 Data Flow Diagram

DFD Level-0

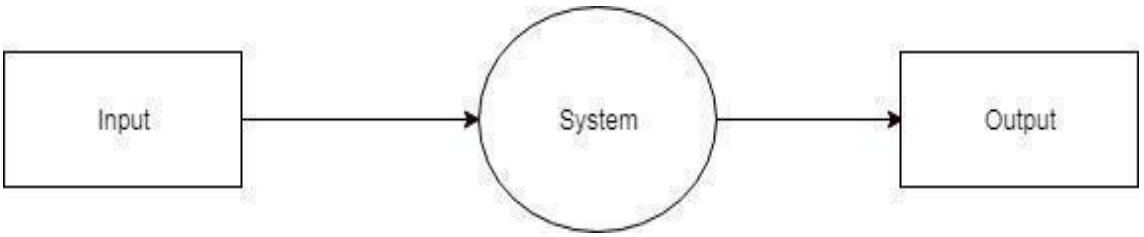

Figure 4.2: DFD Level 0

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

Input: Here input to the system is uploading video.

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

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

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

DFD Level-1

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

[2] Where system gives detailed information of the procedure taking place.

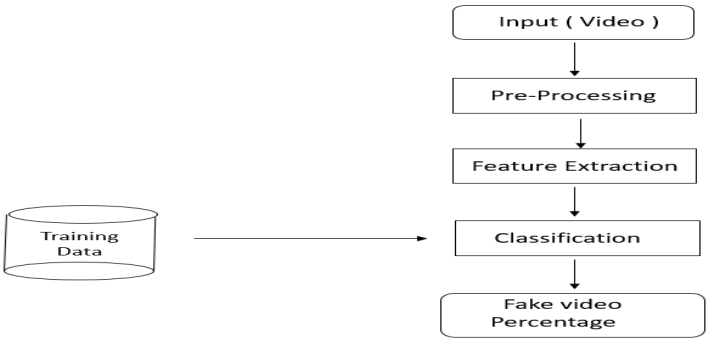

Figure 4.3: DFD Level 1

DFD Level-2

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

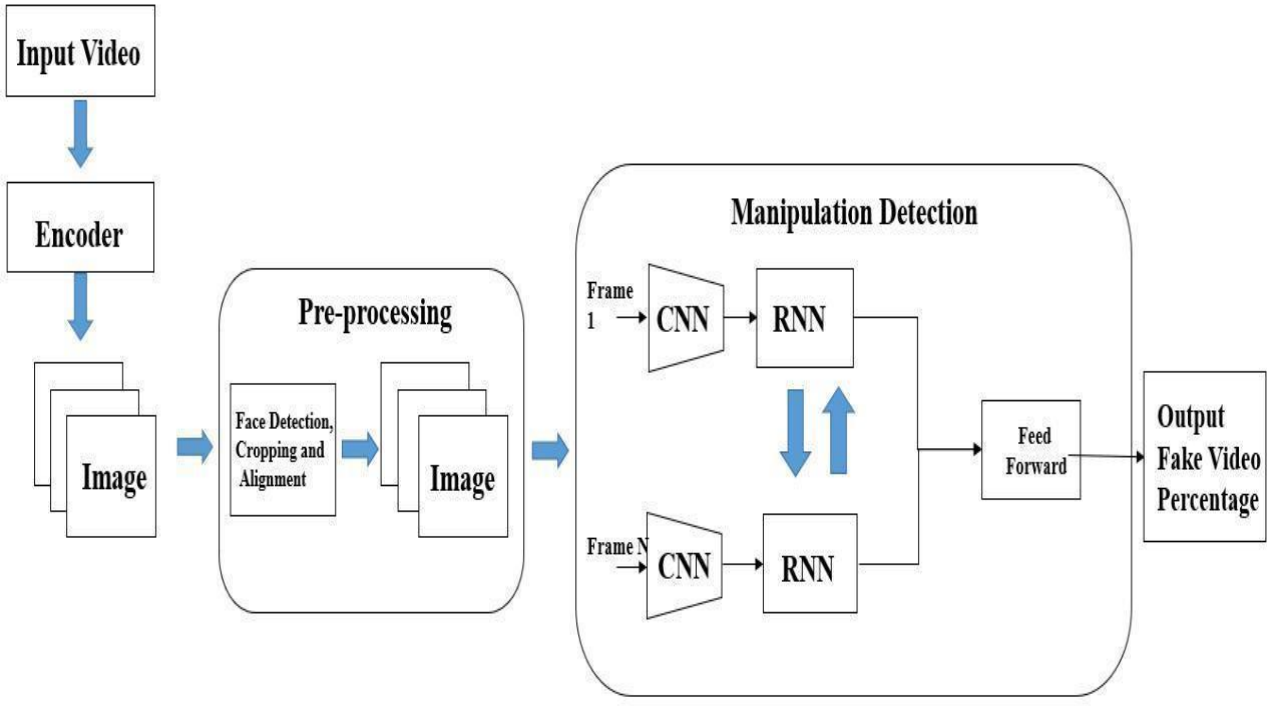

Figure 4.4: DFD Level 2

4.2.2 Activity Diagram:

Training Workflow:

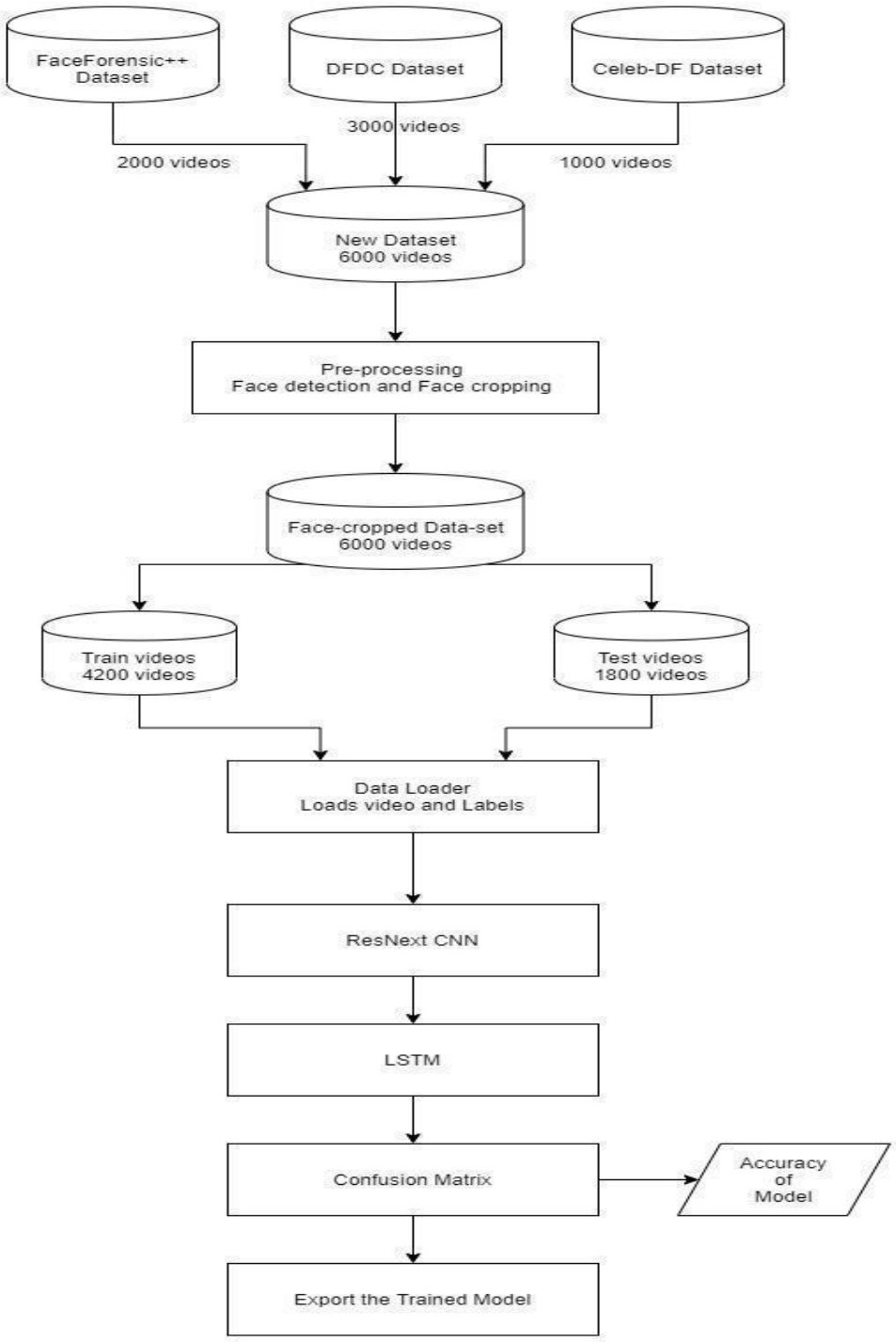

Figure 4.5: Training Workflow

Testing Workflow:

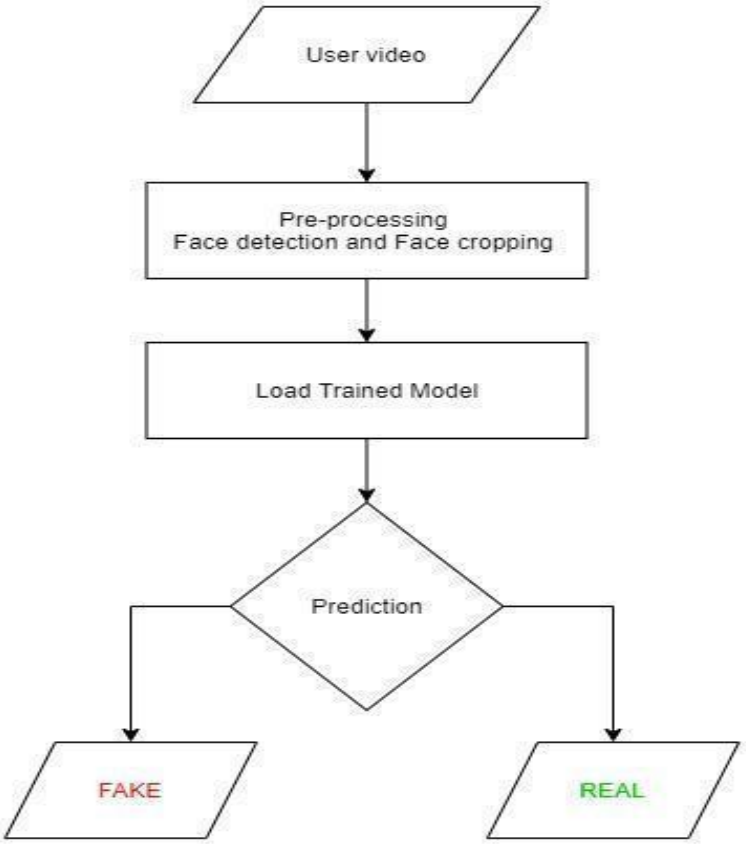

Figure 4.6: Testing Workflow

4.2.3 Non-Functional Requirements:

Performance Requirement

To ensure reliable recognition of fake videos and enhance the practical usability of the software, it is important to focus on efficient design principles. The design should prioritize accuracy and effectiveness in detecting deep fakes while considering practical applications.

Versatility and user-friendliness are key aspects of the design. The system should be able to handle various types of videos and adapt to different scenarios seamlessly. This versatility enables the software to cater to a wide range of user needs and effectively detect fake videos in diverse contexts. Simultaneously, the user interface should be intuitive, making it easy for users to interact with the software and interpret the results.

Speed, reliability, and time-saving capabilities are crucial considerations. The software should perform detection tasks efficiently, providing real-time or near real-time results. By minimizing processing time and maintaining high accuracy, the system becomes a valuable tool for users seeking quick and reliable verification of video authenticity.

Additionally, the design should allow for universal adaptations. It should be capable of handling various video formats, resolutions, and quality levels, ensuring compatibility with different sources and platforms. This adaptability broadens the scope of the software's applicability and makes it accessible to a wider user base.

Considering future advancements, it is essential to design the system with compatibility for future upgrades and enhancements in mind. The software should have the flexibility to incorporate new detection techniques, algorithms, and datasets as they become available. Easy integration with other systems and technologies is also beneficial, allowing for seamless integration into existing workflows or platforms.

Designing a system with compatibility for future upgrades and enhancements is crucial to ensure its long-term usability and adaptability. This involves adopting a modular approach, where different components can be easily replaced or upgraded without disrupting the entire system. Open standards and protocols should be employed to enable seamless integration with other systems and technologies, facilitating the incorporation of new detection techniques, algorithms, or datasets as they become available. Providing flexible APIs allows for interaction with external systems and easy integration with third-party tools. Extensibility should be prioritized, allowing for the addition of new functionalities or features in a modular and straightforward manner. Thorough documentation and version control ensure that future developers can understand and extend the system effectively. By considering these factors, a software solution can be designed to accommodate future advancements and enhancements in the field of deepfake detection.

Safety Requirement

The Data integrity is preserved. Once the video is uploaded to the system. It is only processed by the algorithm. The videos are kept secured from the human interventions, as the uploaded video is not are not able for human manipulation.

To extent the safety of the videos uploaded by the user will be deleted after 30 min from the server.

Security Requirement

While uploading the video, the video will be encrypted using a certain symmetric encryption algorithm. On server also the video is in encrypted format only.

The video is only decrypted from preprocessing till we get the output. After getting the output the video is again encrypted.

This cryptography will help in maintain the security and integrity of the video. SSL certification is made mandatory for Data security.

4.2.4 Sequence Diagram

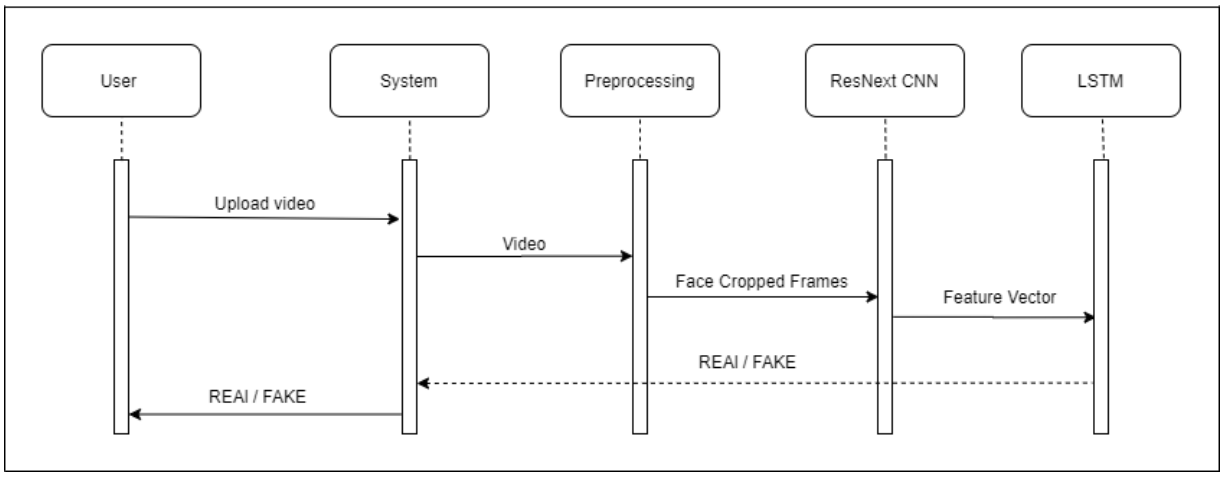

Figure 4.7: Sequence Diagram

Once the output is obtained, it is recommended to encrypt the video again before storing or transmitting it. This provides an additional layer of protection to the output data.

In terms of data security, it is also important to obtain SSL certification. SSL (Secure Sockets Layer) ensures that the communication between the client and server is encrypted and secure, preventing unauthorized interception or tampering of data.

By implementing these security measures, the system can maintain the security and integrity of the video data, protecting it from unauthorized access and maintaining the privacy of the users.

Chapter 5

Detailed Design Document

5.1 Introduction

5.1.1 System Architecture

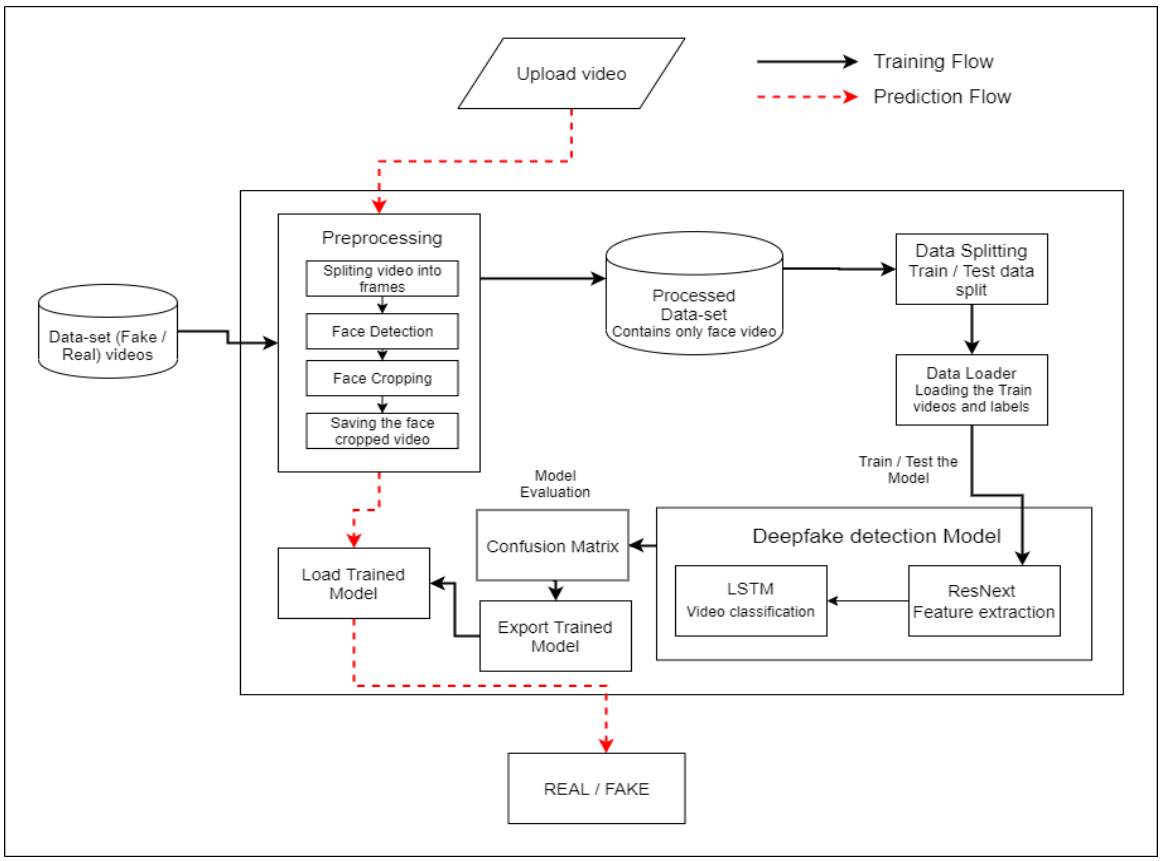

Figure 5.1: System Architecture

In the development of this system, a PyTorch deepfake detection model was trained to ensure a balanced representation of real and fake videos. This approach was adopted to mitigate any potential bias that may arise due to an uneven distribution of training samples. The system architecture of the model is depicted in the accompanying figure.

During the development phase, a dataset was acquired and subjected to preprocessing. As part of the preprocessing steps, the dataset underwent a transformation where only the face region of the

videos was cropped and extracted. This process resulted in the creation of a new processed dataset specifically focused on the facial features necessary for deepfake detection.

By concentrating on the face cropped videos, the model becomes adept at capturing the distinctive visual cues and patterns associated with deepfake manipulation, thus enhancing the accuracy of the detection process. The utilization of this processed dataset ensures that the model is trained on relevant and meaningful information for the task at hand.
This meticulous approach in dataset preprocessing and the training of the deepfake detection model serves to optimize the system's performance in accurately distinguishing between real and manipulated videos. By striking a balance in the training dataset and focusing on key facial features, the system is designed to effectively identify and flag deepfake content.

Creating deepfake videos

To effectively detect deepfake videos, it is essential to have a deep understanding of the typical process involved in their creation using popular deepfake generation tools like GANs and autoencoders. These tools typically take a source image and a target video as inputs to generate the deepfake video.
The process begins by breaking down the target video into individual frames, each containing a face. These faces are then detected and localized within each frame. Next, the source face is replaced with the target face using various techniques, such as face alignment and blending algorithms. This ensures that the substituted face seamlessly integrates into the video.
To further enhance the quality of the deepfake video, pre-trained models are utilized. These models employ advanced algorithms to refine the modified frames, reducing any remaining artifacts or traces left by the deepfake creation process. As a result, the final deepfake video appears highly realistic and can be visually indistinguishable from authentic videos.

However, even though deepfake videos may look convincing to human observers, they often contain subtle traces or artifacts that can be detected through computational analysis. The aim of the paper is to identify these imperceptible traces and distinguishable artifacts in deepfake videos. By analyzing these unique characteristics, it becomes possible to classify videos as either deepfake or real, providing a crucial tool in combating the spread of manipulated content.By developing methods to uncover and analyze these hidden traces and artifacts, researchers aim to enhance the detection of deepfakes, even those that appear visually indistinguishable from real videos. These techniques leverage computational algorithms and pattern recognition to identify patterns and inconsistencies that indicate the presence of deepfake manipulation. Ultimately, this research

contributes to the ongoing efforts to combat the spread of deepfakes and protect the integrity of digital content.

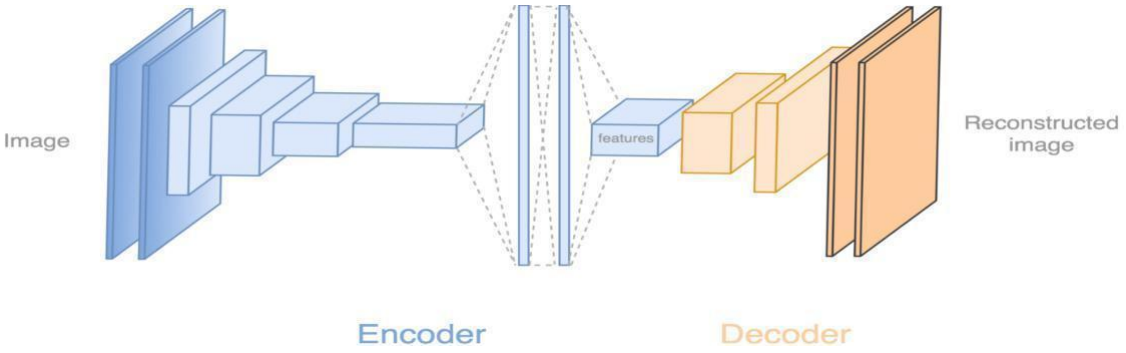

Figure 5.2: Deepfake generation

Figure 5.3: Face Swapped deepfake generation

Tools for deep fake creation.

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

5.2 Architectural Design

5.2.1 Module 1 : Data-set Gathering

To ensure the efficiency and real-time prediction capability of the deepfake detection model, a comprehensive dataset was assembled from various available sources. The dataset compilation involved gathering data from the FaceForensic++ (FF), Deepfake Detection Challenge (DFDC), and Celeb-DF datasets.

To create a balanced dataset and avoid training bias, equal proportions of real and fake videos were included. Specifically, 50% of the dataset comprised real videos, while the remaining 50% consisted of deepfake videos.

The DFDC dataset contained certain audio-alerted videos that were not within the scope of this research paper. To address this, a preprocessing step was performed. A Python script was employed to remove the audio-altered videos from the DFDC dataset, ensuring that the remaining videos were suitable for deepfake detection analysis.

From the preprocessed DFDC dataset, a subset of 1500 real videos and 1500 fake videos were carefully chosen. In addition to that, another set of 1000 real videos and 1000 fake videos were selected from the FF dataset. To further enrich the dataset, 500 real videos and 500 fake videos were specifically picked from the Celeb-DF dataset. All these efforts resulted in a comprehensive dataset comprising a total of 6000 videos. Out of these, 3000 videos are authentic, representing real scenarios, while the remaining 3000 videos are deepfake videos, artificially manipulated to deceive viewers. This diverse and sizable dataset serves as a valuable resource for training and evaluating deepfake detection algorithms.

The distribution of the datasets is depicted in Figure 2, illustrating the proportional representation of real and fake videos from each source. This comprehensive dataset, comprising a diverse range of videos, aims to improve the accuracy and robustness of the deepfake detection model, enabling.

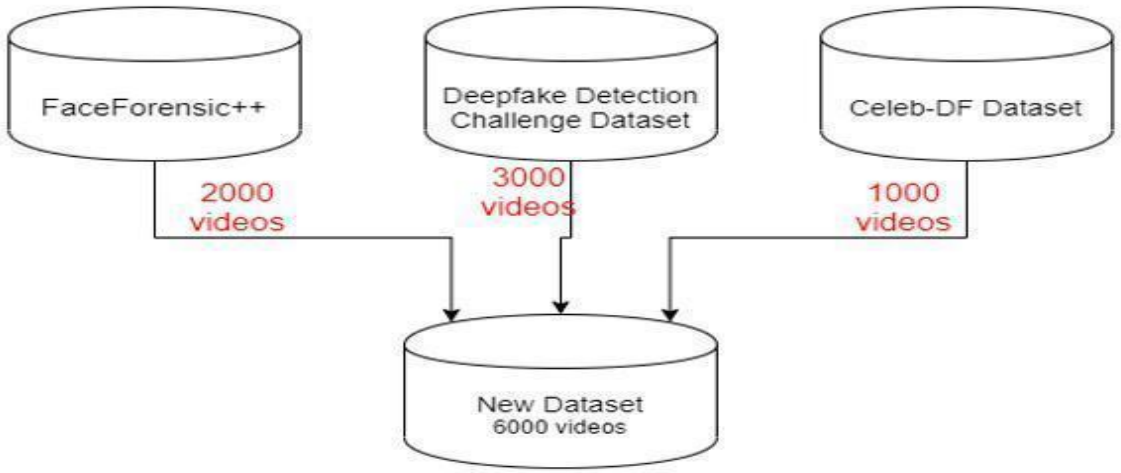

Figure 5.4: Dataset

5.2.2 Module 2 : Pre-processing

During the preprocessing step, the videos undergo several transformations to extract the necessary information and remove unwanted noise. The primary focus is on isolating and capturing the face region in each frame of the video.

The initial step in the preprocessing stage involves splitting the original video into individual frames, treating each frame as an independent image. Within each frame, a face detection algorithm is applied to identify the presence of a face. Once a face is detected, the frame is cropped to extract the face region, discarding the rest of the image. This process is repeated for every frame in the video.

The resulting cropped frames are then combined to create a new video where each frame solely contains the detected face region. Frames without a detected face are excluded during this preprocessing phase.

To maintain consistency and manage computational resources effectively, a threshold value is determined based on the mean frame count across all videos. This threshold ensures that all videos have a uniform number of frames. Considering computational limitations, such as GPU processing power, a threshold of 150 frames is chosen. Only the first 150 frames of each video are retained and saved in the processed dataset.

To leverage the effectiveness of Long Short-Term Memory (LSTM) models, the frames are arranged sequentially, preserving their original order. This sequential arrangement allows LSTM models to capture temporal dependencies and patterns present in the video data.

The newly processed videos are saved at a frame rate of 30 frames per second (fps) and a resolution of 112 x 112 pixels. This resolution strikes a balance between computational efficiency and retaining sufficient visual information for accurate deepfake detection purposes.

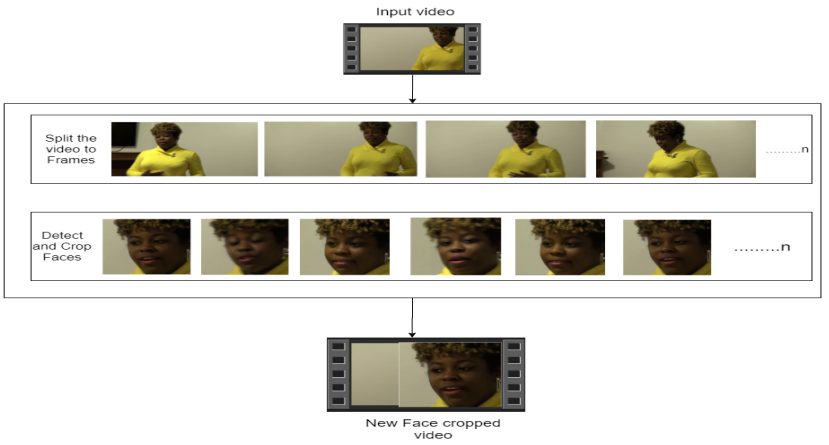

Figure 5.5: Pre-processing of video

5.2.3 Module 3: Data-set split

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

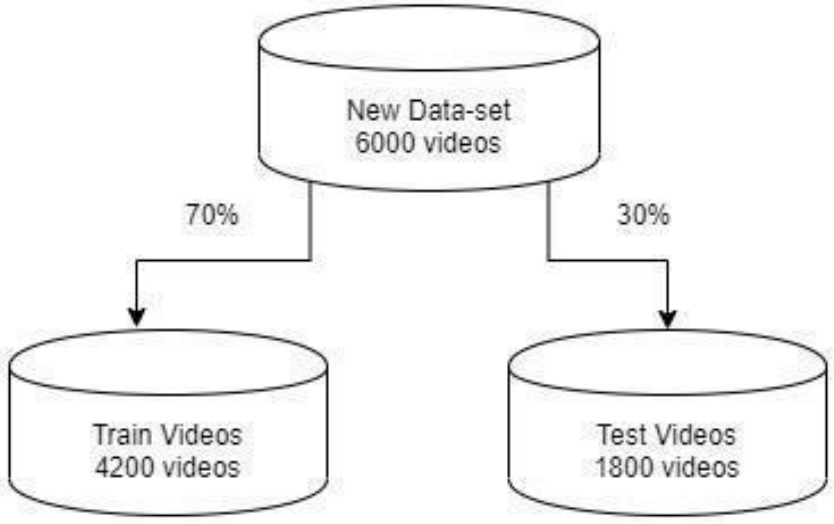

Figure 5.6: Train test split

5.2.4 Module 4: Model Architecture

In our deepfake detection model, we utilize a combination of Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), specifically the ResNext CNN model for feature extraction and a Long Short-Term Memory (LSTM) network for video classification.

For feature extraction, we employ a pre-trained ResNext model. ResNext is a residual CNN network designed to optimize performance on deeper neural networks. In our experiment, we specifically use the resnext50_32x4d model, which consists of 50 layers and dimensions of 32x4.

To optimize the network, we incorporate additional layers as needed and carefully select a suitable learning rate to ensure effective convergence of the gradient descent algorithm. The 2048-dimensional feature vectors, obtained after applying the last pooling layers of the ResNext model, are utilized as input for the sequential LSTM. By processing the sequence of feature vectors

in a sequential manner, the LSTM layer enables temporal analysis by comparing the frame at time 't' with the frames at previous time steps ('t-n', where 'n' can represent any preceding frames).

The model employs a Leaky ReLU activation function and a linear layer with 2048 input features and 2 output features. This architecture allows the model to capture the relationship between the input and output data.

To ensure the desired output size of the image, an adaptive average pooling layer is employed with an output parameter set to 1. This adjustment guarantees that the output size matches the desired H x W dimensions.

For handling the sequential processing of frames, a Sequential Layer is utilized, and a batch size of 4 is selected for training the model. Finally, a SoftMax layer is applied to obtain the model's prediction confidence, providing an indication of the likelihood of classifying a video as either a deepfake or a pristine video.

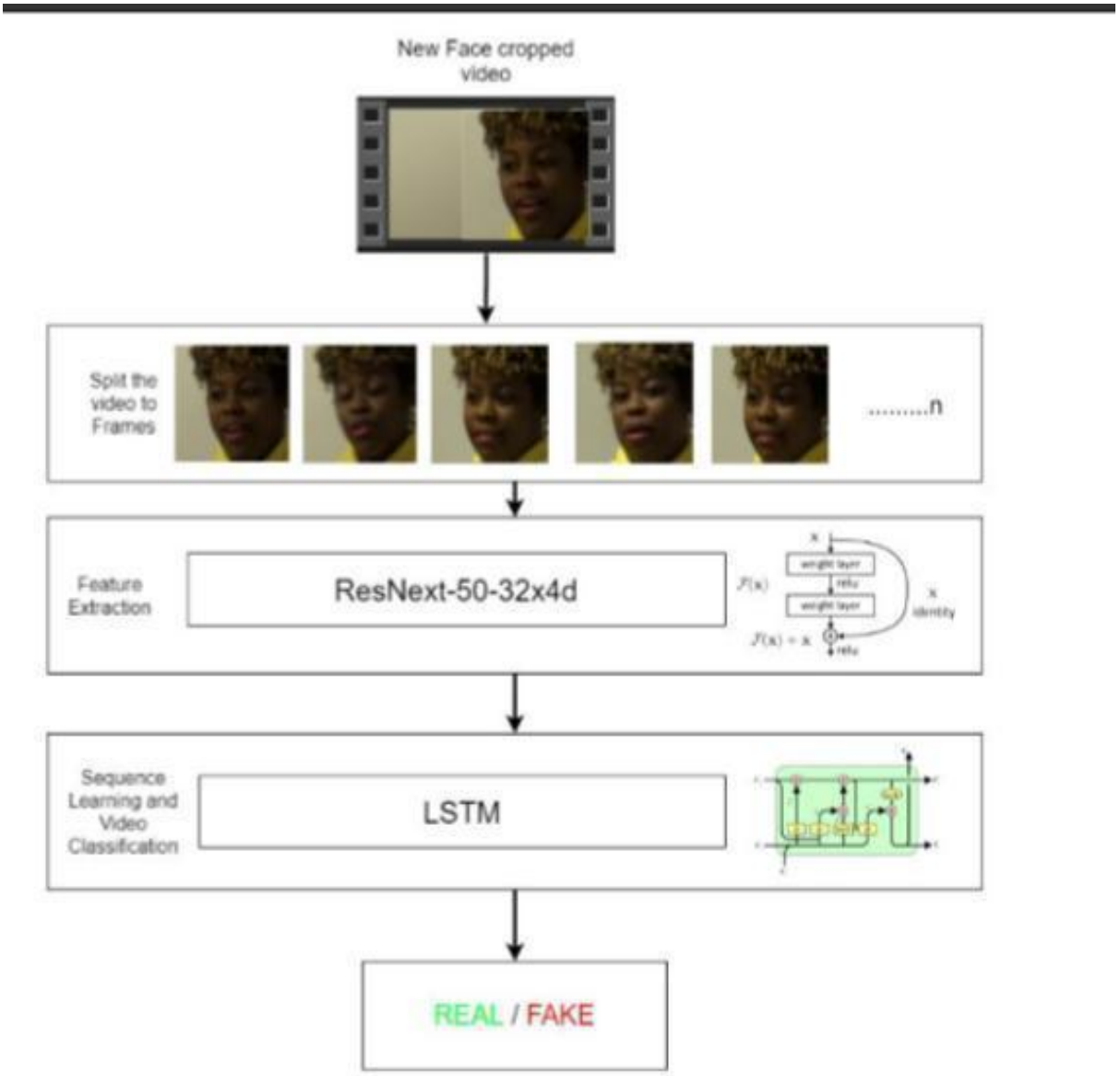

Figure 5.7: Overview of our model

5.2.5 Module 5: Hyper-parameter tuning

In order to achieve maximum accuracy, the process of hyperparameter tuning is crucial. After multiple iterations on the model, we have selected the best hyperparameters for our dataset. We utilize the Adam optimizer with a learning rate of 1e-5 (0.00001) to enable adaptive learning rate and achieve better convergence. A weight decay of 1e-3 is applied to prevent overfitting. Since this is a classification problem, we use the cross-entropy loss function.

To make efficient use of the available computation power, batch training is employed with a batch size of 4. Through experimentation, we have determined that a batch size of 4 is the ideal size for training in our development environment.

For the user interface of the application, we have utilized the Django framework. Django provides scalability and flexibility for the application's future development. The main page of the user interface, index.html, includes a tab that allows users to browse and upload a video. The uploaded video is then passed to the model for prediction. The model predicts whether the video is real or fake and provides the confidence of the model. The output is displayed on the predict.html page, overlaying the prediction on the playing video.

Chapter 6

Project Implementation

6.1 Introduction

There are many examples where deepfake creation technology is used to mis-lead the people on social media platform by sharing the false deepfake videos of the famous personalities like Mark Zuckerberg Eve of House A.I. Hearing, Don-ald Trump's Breaking Bad series where he was introduces as James McGill, Barack Obama's public service announcement and many more [15]. These types of deepfakes creates a huge panic among the normal people, which arises the need to spot these deepfakes accurately so that they can be distinguished from the real videos.

Latest advances in the technology have changed the field of video manipulation. The advances in the modern open source deep learning frameworks like TensorFlow, Keras, PyTorch along with cheap access to the high computation power has driven the paradigm shift. The Conventional autoencoders[16] and Generative Adversarial Network (GAN) pretrained models have made the tampering of the realistic videos and images very easy. Moreover, access to these pretrained models through the smartphones and desktop applications like FaceApp and Face Swap has made the deepfake creation a childish thing. These applications generate a highly realistic synthesized transformation of faces in real videos. These apps also provide the user with more functionalities like changing the face hair style, gender, age and other attributes. These apps also allow the user to create a very high quality and indistinguishable deepfakes. Although some malignant deepfake videos exist, but till now they remain a minority. So far, the released tools [4,5] that generate deepfake videos are being extensively used to create fake celebrity pornographic videos or revenge porn [17]. Some of the examples are Brad Pitt, Angelina Jolie nude videos. The real looking nature of the deepfake videos makes the celebrities and other famous personalities the target of pornographic material, fake surveillance videos, fakenews and malicious hoaxes. The Deepfakes are very much popular in creating the political tension [12]. Due to which it becomes very important to detect the deepfake videos and avoid the percolation of the deepfakes on the social media platforms.

6.2 Tools and Technologies Used

6.2.1 Planning

1. Open Project

6.2.2 UML Tools

1. draw.io

6.2.3 Programming Languages

1. Python3
2. JavaScript

6.2.4 Programming Frameworks

1. PyTorch
2. Django

6.2.5 IDE

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

6.2.6 Versioning Control

1. Git

6.2.7 Cloud Services

1. Google Cloud Platform

6.2.8 Application and web servers:

1. Google Cloud Engine

6.2.9 Libraries

1. torc
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

6.3 Algorithm Details

6.3.1 Dataset Details

Refer 5.2.1

6.3.2 Preprocessing Details

Using the 'glob' library, we import all the videos located in the specified directory and store them in a Python list. The 'cv2.VideoCapture' function is utilized to read the videos and calculate the mean number of frames in each video. To ensure consistency, we select the value of 150 as the ideal number of frames for creating the new dataset.

The video is then processed by splitting it into individual frames, and each frame is cropped to focus specifically on the area containing the face. This step involves employing face detection techniques to accurately identify and extract the face region from each frame.

The resulting cropped frames are then written to a new video using the 'VideoWriter' function. This allows us to create a new video comprised solely of the face-cropped frames. The new video is encoded in the MP4 format and has a frame rate of 30 frames per second, with a resolution of 112 x 112 pixels.

To ensure the effective utilization of LSTM for temporal sequence analysis, instead of randomly selecting frames, we choose the first 150 frames from each video and include them in the new video dataset

To ensure sequential processing of frames and to leverage the capabilities of the LSTM (Long Short-Term Memory) for temporal sequence analysis, we specifically select the first 150 frames of each video for inclusion in the new dataset. This sequential approach enables the LSTM to analyze and compare the frames at different time intervals, aiding in the detection of temporal patterns or changes within the video data.

6.3.3 Model Details

The model consists of following layers:

ResNext-CNN : The ResNext-CNN model, specifically the pre-trained variant called resnext50_32x4d()[18], is employed for deepfake detection. This model is composed of 50 layers and has a dimension of 32 x 4. The detailed implementation of the model can be visualized in the accompanying figure.

stage	output	**ResNeXt-50 (32×4d)**
conv1	112×112	7×7, 64, stride 2
conv2	56×56	3×3 max pool, stride 2 $\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128, C{=}32 \\ 1\times1, 256 \end{bmatrix} \times3$
conv3	28×28	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256, C{=}32 \\ 1\times1, 512 \end{bmatrix} \times4$
conv4	14×14	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512, C{=}32 \\ 1\times1, 1024 \end{bmatrix} \times6$
conv5	7×7	$\begin{bmatrix} 1\times1, 1024 \\ 3\times3, 1024, C{=}32 \\ 1\times1, 2048 \end{bmatrix} \times3$
	1×1	global average pool 1000-d fc, softmax
# params.		$\mathbf{25.0}\times10^6$

Figure 6.1: ResNext Architecture

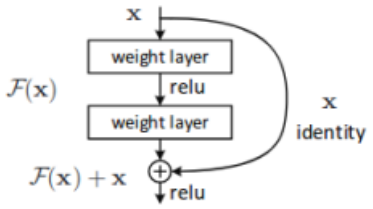

Figure 6.2: ResNext Working

ResNext, short for Residual Next, is a type of convolutional neural network (CNN) architecture that builds upon the concept of ResNet (Residual Network) models. ResNext models were specifically designed to handle the challenges of training deeper networks by introducing a new dimension called "cardinality."

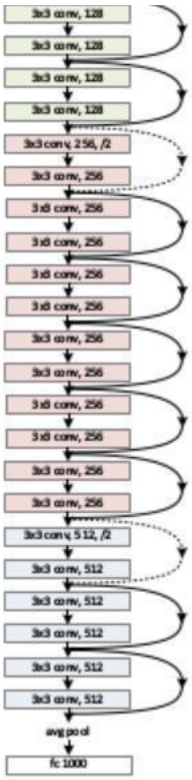

Figure 6.3: Overview of ResNext Architecture

Sequential Layer : The Sequential layer serves as a container for Modules, allowing them to be stacked and executed simultaneously. In the context of deepfake detection, the Sequential layer is utilized to store the feature vector generated by the ResNext model in an ordered manner. This enables the sequential passing of the feature vectors to the LSTM layer for further processing and

analysis. By maintaining the sequence of the feature vectors, the model can capture temporal dependencies and patterns within the video data.

LSTM Layer : The LSTM layer, or Long Short-Term Memory layer, plays a crucial role in the deepfake detection process. As a specialized type of recurrent neural network (RNN) layer, LSTM is designed to handle sequential data analysis tasks effectively. In the case of deepfake detection, the LSTM layer is utilized to capture the temporal dynamics and identify any discrepancies or variations between video frames.
In this particular implementation, the LSTM layer receives as input a sequence of 2048-dimensional feature vectors. These feature vectors are extracted from the video frames using the ResNext CNN model, which has been pre-trained to extract meaningful visual information. Each feature vector represents the visual characteristics of a specific frame, allowing the LSTM layer to analyze the sequence of these vectors and identify patterns or abnormalities over time.

The LSTM layer used in this model consists of one LSTM unit. The unit has 2048 latent dimensions, which means it can learn complex patterns and representations within the sequence. The LSTM layer also has 2048 hidden layers, which allows the network to capture and model the temporal dependencies in the video.
To prevent overfitting and enhance the generalization capability of the model, a dropout regularization technique is applied with a dropout rate of 0.4. Dropout randomly sets a fraction of the

input units to 0 during training, which helps prevent the network from relying too heavily on any specific set of features and promotes better generalization.

By processing the frames in a sequential manner, the LSTM layer can analyze the temporal changes between frames at different time steps. It compares the frame at time 't' with the frame at 't-n' seconds, where 'n' represents the number of frames before time 't'. This allows the model to capture and learn the temporal patterns and dynamics present in the video, which can be indicative of deepfake manipulations.

Overall, the LSTM layer plays a crucial role in analyzing the temporal aspects of the video data and contributes to the model's ability to detect deepfakes by capturing temporal inconsistencies and changes between frames.

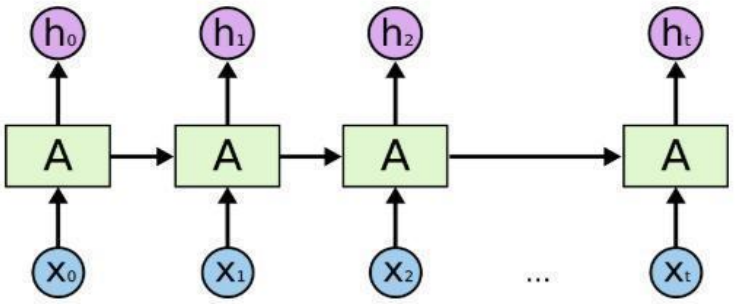

Figure 6.4: Overview of LSTM Architecture

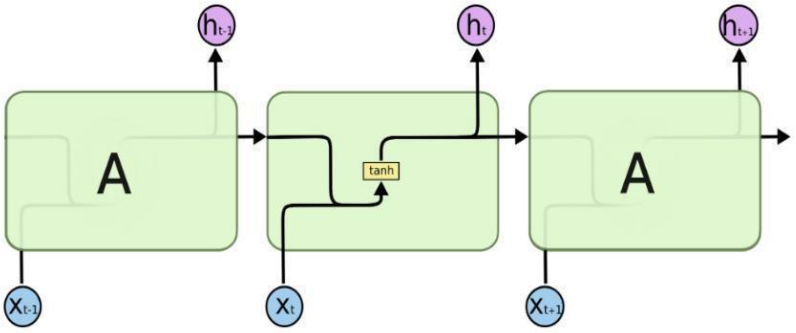

Figure 6.5: Internal LSTM Architecture

ReLU: The Rectified Linear Unit (ReLU) is a widely used activation function in neural networks. It serves a simple purpose: if the input value is negative, the output is set to 0; if the input value is non-negative, the output remains equal to the input value. This activation function effectively introduces non-linearity into the network, allowing it to learn complex patterns and make accurate predictions. By eliminating negative values and preserving positive values, ReLU helps the network to be more expressive and overcome the vanishing gradient problem often encountered in deep learning models.

ReLU is indeed a popular choice for activation functions in neural networks due to its several advantages. One of the key advantages is its non-linearity, which enables neural networks to capture complex and non-linear relationships in the data they are trained on. By introducing non-linearity,

ReLU allows the network to approximate more intricate functions and make more expressive predictions.

Another advantage of ReLU is its ability to alleviate the "vanishing gradient" problem that can occur with other activation functions like the sigmoid function. The vanishing gradient problem refers to the issue where gradients become extremely small during backpropagation in deep neural networks, hindering the learning process. Since ReLU has a simple derivative of either 0 or 1, it avoids the vanishing gradient problem and allows gradients to flow more freely during training, facilitating faster learning and better convergence.

Furthermore, ReLU is computationally efficient to compute compared to other activation functions, such as sigmoid or tanh, which involve exponential calculations. The ReLU function only requires a simple thresholding operation, making it faster to evaluate and more suitable for large-scale neural networks.

Overall, ReLU's non-linearity, gradient flow, and computational efficiency make it a widely adopted choice for activation functions in various deep learning applications.

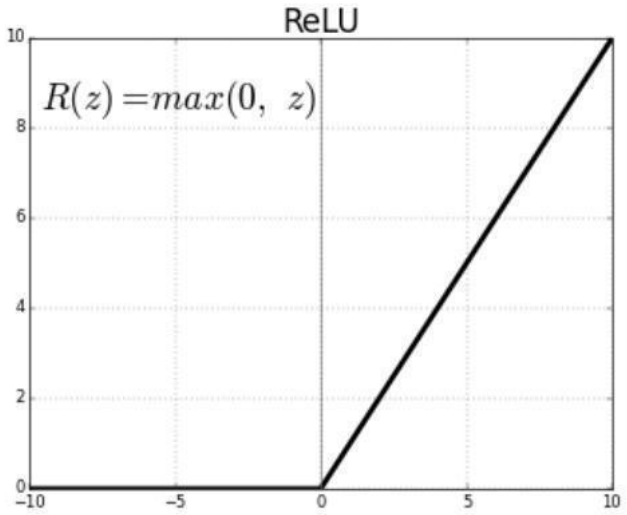

Figure 6.6: ReLu Activation function

Dropout Layer : The Dropout layer is a regularization technique commonly used in neural networks to prevent overfitting. It helps in improving the generalization ability of the model by randomly setting a fraction of the output values from the previous layer to zero during training. In this specific implementation, a Dropout layer with a value of 0.4 is utilized.

By setting a fraction of the neuron outputs to zero, the Dropout layer introduces a form of noise or randomness into the network. This noise disrupts the co-adaptation of neurons and encourages them to be more robust and independent. It prevents individual neurons from relying too heavily on specific input features, thus reducing the risk of overfitting.
During training, each neuron in the Dropout layer has a probability of being "dropped out" or deactivated, meaning its output is set to zero. The probability of dropout, in this case, is set to 0.4. As a result, the remaining active neurons need to compensate for the dropped-out neurons, leading to a more distributed representation of the input data.

By randomly dropping out neurons, the cost function becomes more sensitive to the neighboring neurons, forcing them to learn more reliable and diverse representations of the data. This helps to prevent overfitting and encourages the network to generalize better to unseen data.

During inference or prediction, when the Dropout layer is not active, the output of each neuron is multiplied by the retention probability (1 - dropout probability) to maintain the expected magnitude of the activations.

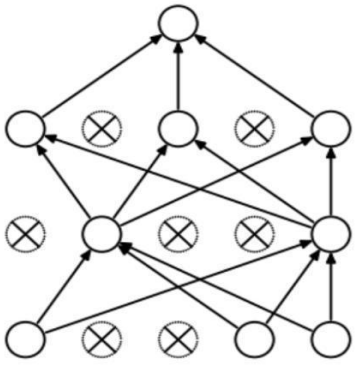

Figure 6.7: Dropout layer overview

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

6.3.4 Model Training Details

Train Test Split: The dataset is divided into two subsets: a training dataset and a testing dataset. The split is performed in a ratio of 70% for the training dataset and 30% for the testing dataset. This means that out of the total dataset consisting of 6,000 videos, 4,200 videos are assigned to the training dataset, while 1,800 videos are allocated to the testing dataset.

Furthermore, the train and test split is balanced, ensuring an equal representation of real and fake videos in both subsets. Specifically, each split contains an equal distribution of 50% real videos and 50% fake videos. This balanced split helps in training and evaluating the model's performance on an equal number of real and fake videos, allowing for a fair assessment of the deepfake detection capabilities.
Please refer to Figure 5.6 for a visual representation of the balanced train and test split.

Data Loader: It is used to load the videos and their labels with a batch size of 4.

Training: The training is done for 20 epochs with a learning rate of 1e-5 (0.00001),weight decay of 1e-3 (0.001) using the Adam optimizer.
Adam optimizer[19]: To enable the adaptive learning rate Adam optimizer with the model parameters is used.

Cross Entropy: Cross-entropy is a commonly used loss function in classification problems, including deep learning models. It is particularly suitable for scenarios where the output of the model represents probabilities or likelihoods of different classes.

The cross-entropy loss is derived from information theory and is defined as the average of the logarithmic loss over the training examples. It penalizes the model more heavily for confidently incorrect predictions and less for predictions that are close to the true labels. This property makes cross-entropy loss effective for training classification models, as it encourages the model to assign higher probabilities to the correct classes.

Softmax Layer The softmax layer serves as the final layer in the neural network model for the purpose of multi-class classification. Its function is to take the outputs from the preceding layers and transform them into a probability distribution across the available classes.

When given a vector of real-valued scores, referred to as logits, the softmax function calculates the exponentiation of each element and divides it by the sum of all exponentiated values. This normalization process ensures that the resulting values represent probabilities and add up to a total of 1. By applying the softmax function, we obtain a probability distribution over the possible classes.

In our specific case, the softmax layer has two output nodes: one for the "REAL" class and one for the "FAKE" class. The softmax layer assigns probabilities to each class, indicating the model's confidence in its prediction for a given input. The class with the highest probability is typically chosen as the predicted class during inference.

During training, the softmax layer is commonly used in conjunction with the cross-entropy loss function. This combination allows for the comparison of the predicted probabilities with the true labels, facilitating the learning process by minimizing the discrepancy between the predicted and actual class probabilities

.

In summary, the softmax layer in our model produces a probability distribution over the classes "REAL" and "FAKE," providing both the predicted class and a measure of confidence in the prediction.

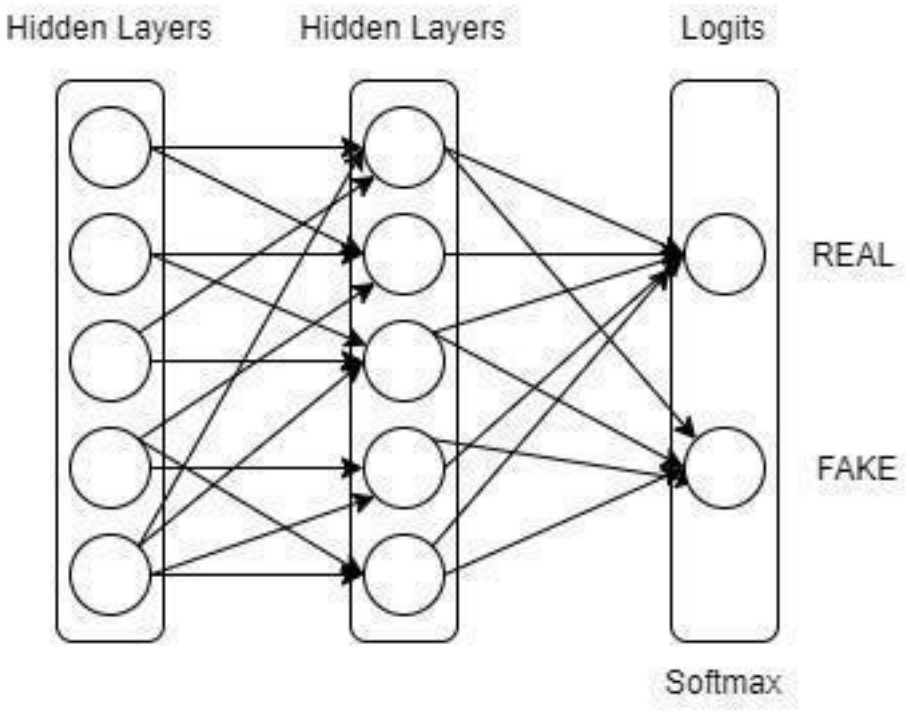

Figure 6.8: Softmax Layer

Confusion Matrix: The confusion matrix is a valuable tool for evaluating the performance of a classification model. It provides a detailed breakdown of the model's predictions by comparing them to the true labels. The matrix is organized into a grid-like structure, where each row represents the true labels of the data, and each column represents the predicted labels.

The main diagonal of the confusion matrix represents the correct predictions, where the true label matches the predicted label. Off-diagonal elements represent the incorrect predictions, indicating the misclassification of samples.
By examining the confusion matrix, we can calculate various evaluation metrics such as accuracy, precision, recall, and F1-score. Accuracy measures the overall correctness of the predictions, while precision focuses on the proportion of correctly predicted positive samples. Recall, also known as sensitivity or true positive rate, measures the proportion of actual positive samples that were correctly identified. The F1-score combines precision and recall to provide a balanced measure of a model's performance.
Overall, the confusion matrix helps us gain a deeper understanding of how well our classification model is performing and provides insights into the specific types of errors it is making. This information can be used to further refine and improve the model's accuracy and effectiveness.

Export Model: After the model is trained, we have exported the model. So that it can be used for prediction on real time data.

6.3.5 Model Prediction Details

The model is loaded in the application
The new video for prediction is preprocessed(refer 6.3.2, 5.2.2) and passed to the loaded model for prediction.

The trained model performs the prediction and return if the video is a real or fake along with the confidence of the prediction.

6.4 Code

The Below mentioned code explains the preprocessing , Model Training and Code used for Prediction of the given video as REAL or FAKE.

```python
# 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()
```

Python

Figure 6.9 : Preprocessing Step

```python
#Model with feature visualization
from torch import nn
from torchvision import models
class Model(nn.Module):
    def __init__(self, num_classes,latent_dim= 2048, lstm_layers=1 , hidden_dim = 2048, bidirectional = False):
        super(Model, self).__init__()
        model = models.resnext50_32x4d(pretrained = True) #Residual Network CNN
        self.model = nn.Sequential(*list(model.children())[:-2])
        self.lstm = nn.LSTM(latent_dim,hidden_dim, lstm_layers,  bidirectional)
        self.relu = nn.LeakyReLU()
        self.dp = nn.Dropout(0.4)
        self.linear1 = nn.Linear(2048,num_classes)
        self.avgpool = nn.AdaptiveAvgPool2d(1)
    def forward(self, x):
        batch_size,seq_length, c, h, w = x.shape
        x = x.view(batch_size * seq_length, c, h, w)
        fmap = self.model(x)
        x = self.avgpool(fmap)
        x = x.view(batch_size,seq_length,2048)
        x_lstm,_ = self.lstm(x,None)
        return fmap,self.dp(self.linear1(torch.mean(x_lstm,dim = 1)))
```

Python

```python
model = Model(2).cuda()
a,b = model(torch.from_numpy(np.empty((1,20,3,112,112))).type(torch.cuda.FloatTensor))
```

Python

Figure 6.10.1 : Model Training

```
                    accuracies.avg))
        torch.save(model.state_dict(),'/content/checkpoint.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):
```

Figure 6.10.2: Model Training

```
            for i,frame in enumerate(self.frame_extract(video_path)):
                #if(i % a == first_frame):
                faces = face_recognition.face_locations(frame)
                try:
                  top,right,bottom,left = faces[0]
                  frame = frame[top:bottom,left:right,:]
                except:
                  pass
                frames.append(self.transform(frame))
                if(len(frames) == self.count):
                    break
            """
            for i,frame in enumerate(self.frame_extract(video_path)):
                if(i % a == first_frame):
                    frames.append(self.transform(frame))
            """
            # if(len(frames)<self.count):
            #   for i in range(self.count-len(frames)):
            #         frames.append(self.transform(frame))
            #print("no of frames", self.count)
            frames = torch.stack(frames)
            frames = frames[:self.count]
            return frames.unsqueeze(0)

        def frame_extract(self,path):
          vidObj = cv2.VideoCapture(path)
          success = 1
          while success:
              success, image = vidObj.read()
              if success:
                  yield image

    def im_convert(tensor, video_file_name):
        """ Display a tensor as an image. """
        image = tensor.to("cpu").clone().detach()
        image = image.squeeze()
        image = inv_normalize(image)
```

Figure 6.11 : Detection using Trained model

Chapter 7

Software Testing

7.1 Type of Testing Used

Functional Testing

Functional testing is a type of software testing that focuses on verifying the functionality of a system or application. It involves testing the individual functions and features of the software to ensure that they work as intended and meet the specified requirements. The purpose of functional testing is to validate that the software behaves correctly and performs the tasks it is supposed to perform.

Functional testing involves designing test cases that cover various scenarios and inputs to test the different functions and features of the software. It typically includes activities such as test planning, test case creation, test execution, and result analysis. The tests are performed by simulating user interactions and inputting specific data to validate the expected behavior of the system.

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

Non-functional Testing

Non-functional testing, also known as quality attributes testing or performance testing, focuses on evaluating the non-functional aspects of a software system. Unlike functional testing, which checks the specific functions and features, non-functional testing is concerned with the overall performance, usability, reliability, and other characteristics of the software**.**

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

7.2 Test Cases and Test Results

Test Cases

Testing videos for different cases :-

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 7.1: Test Case Report

Chapter 8

Results and Discussion

8.1 Screen shots

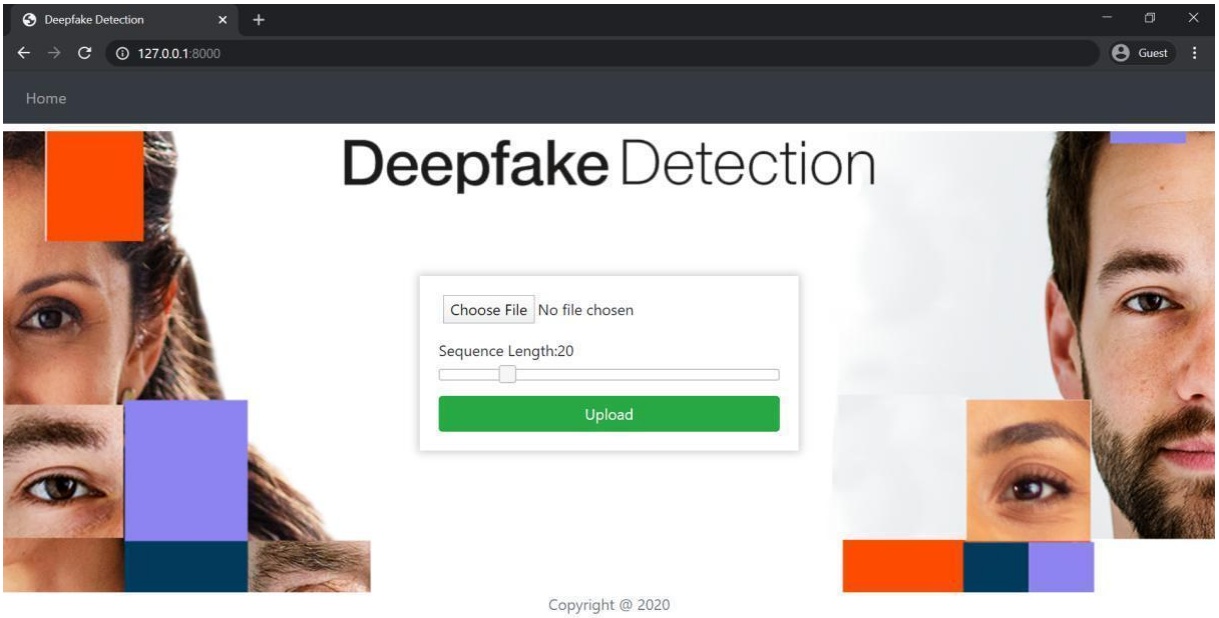

Figure 8.1: Home Page

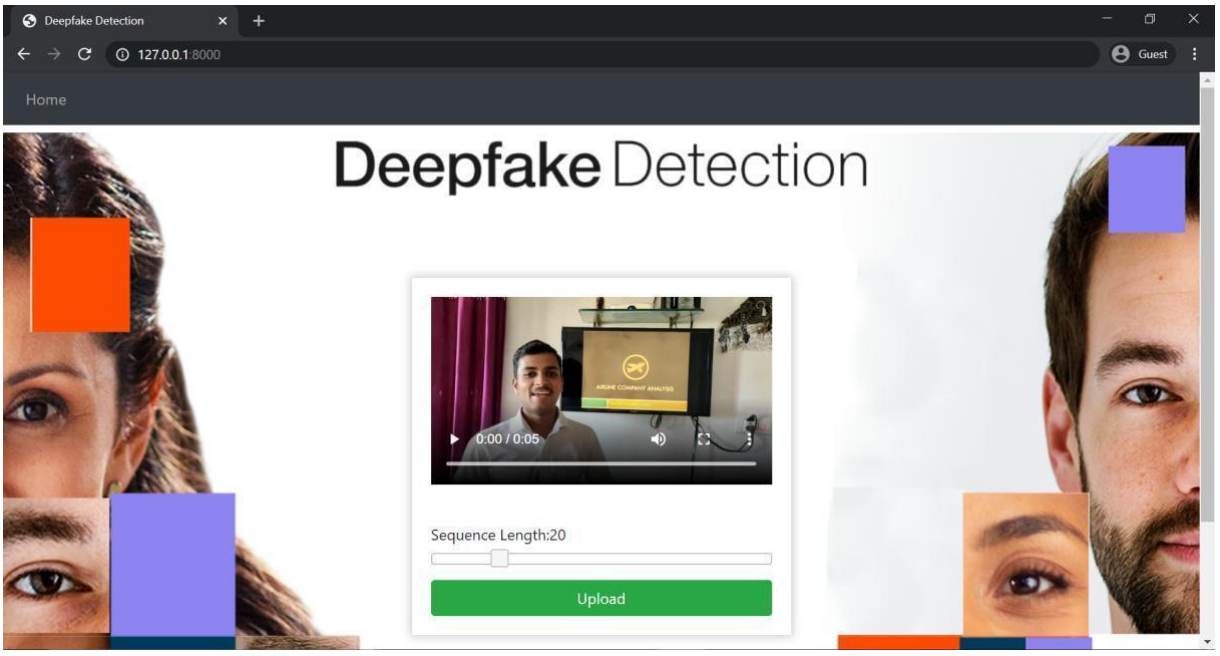

Figure 8.2: Uploading Real Video

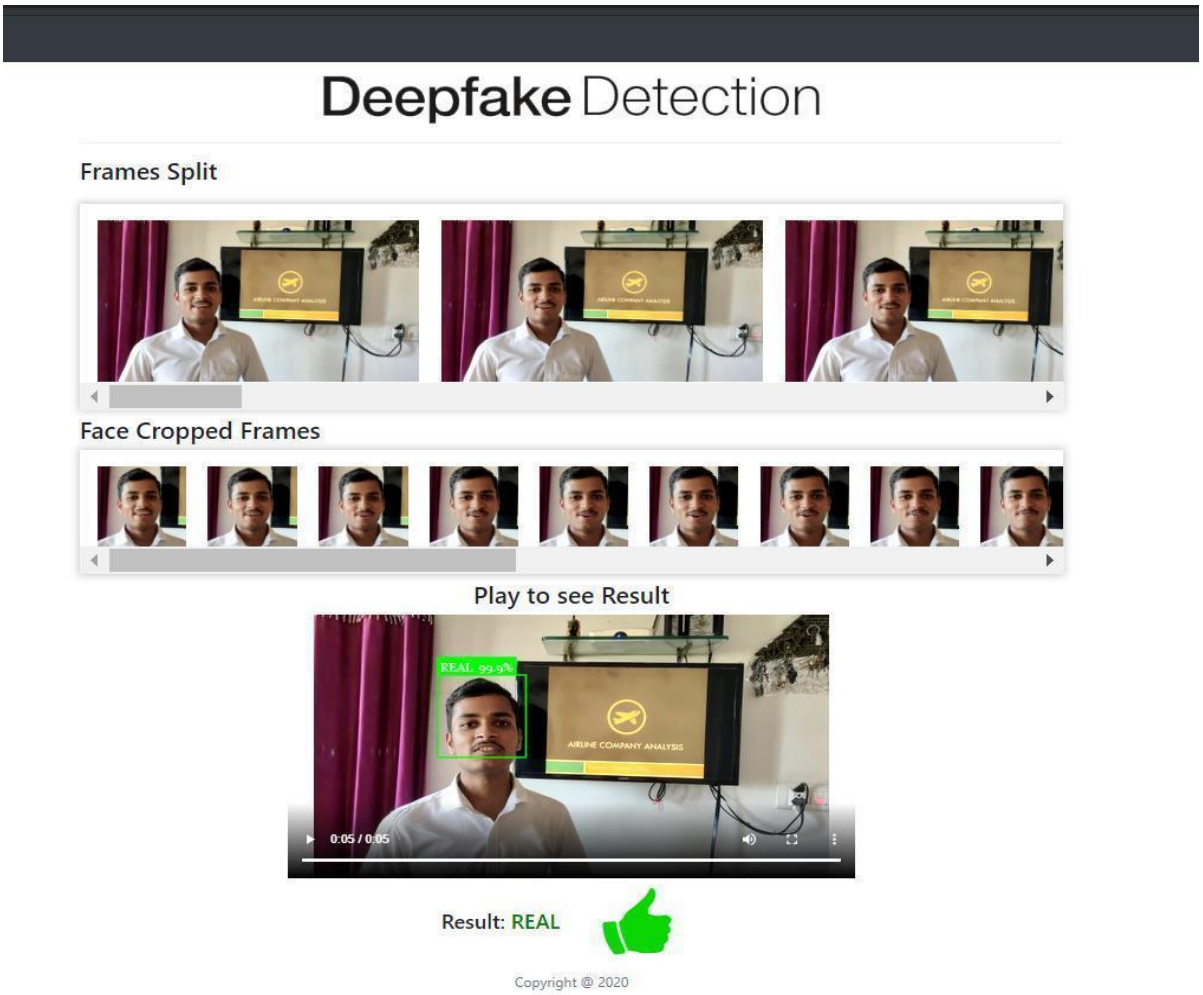

Figure 8.3: Real Video Output

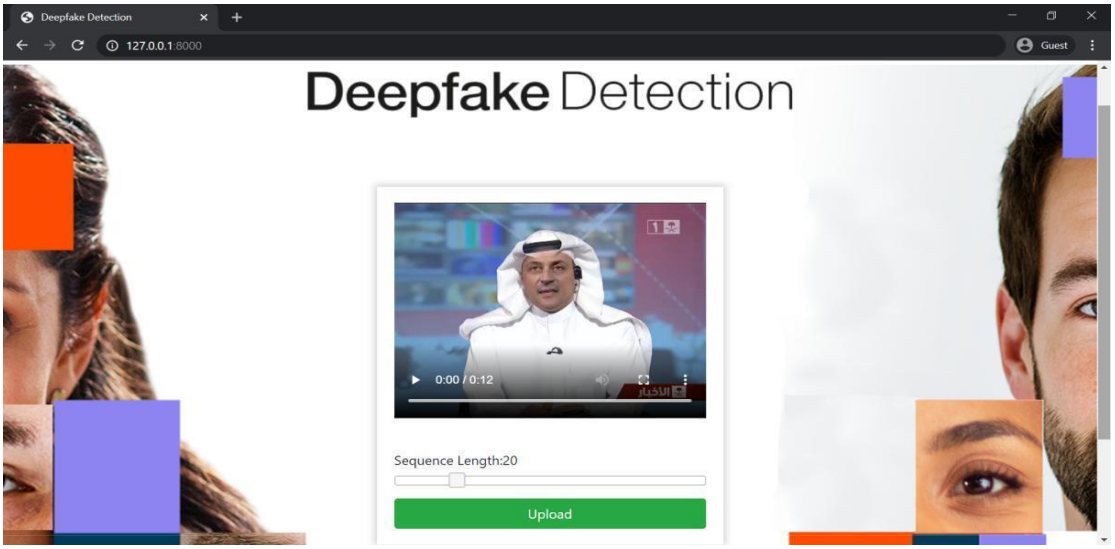

Figure 8.4: Uploading Fake Video

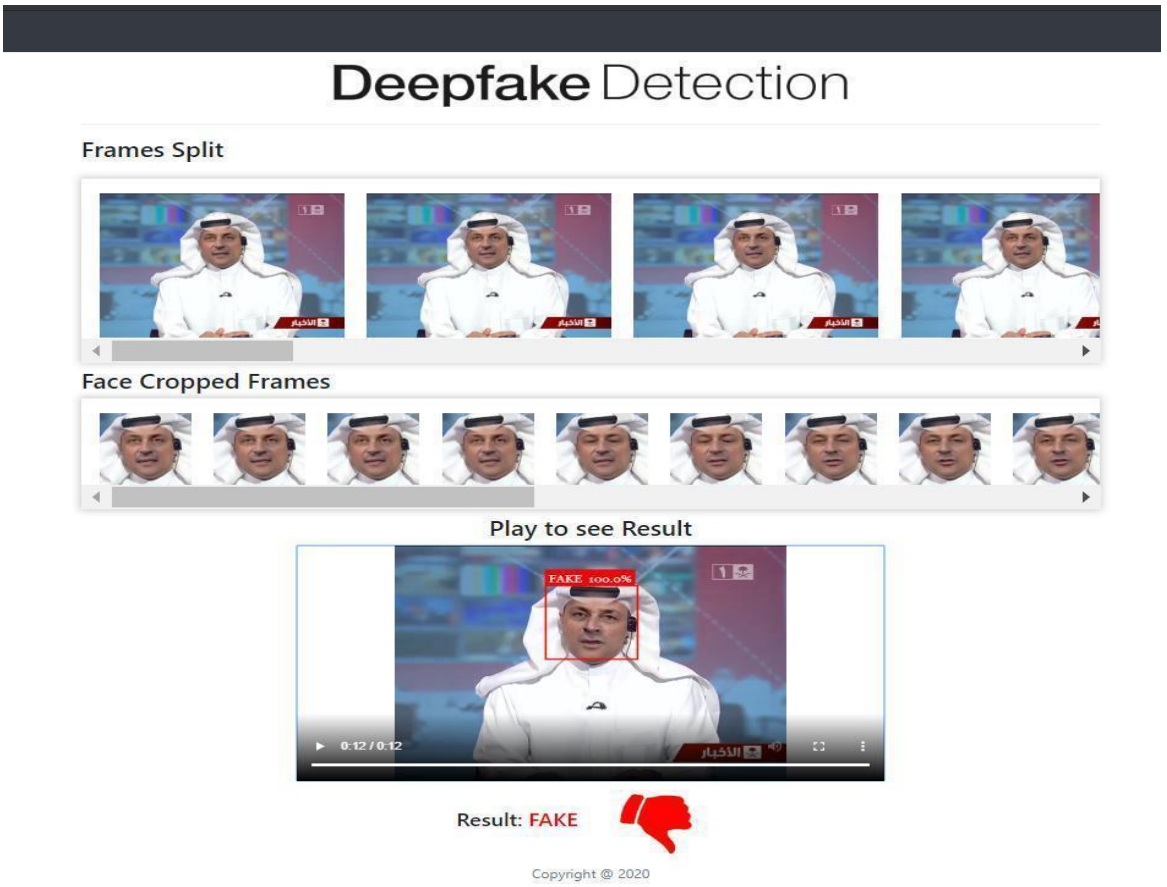

Figure 8.5: Fake video Output

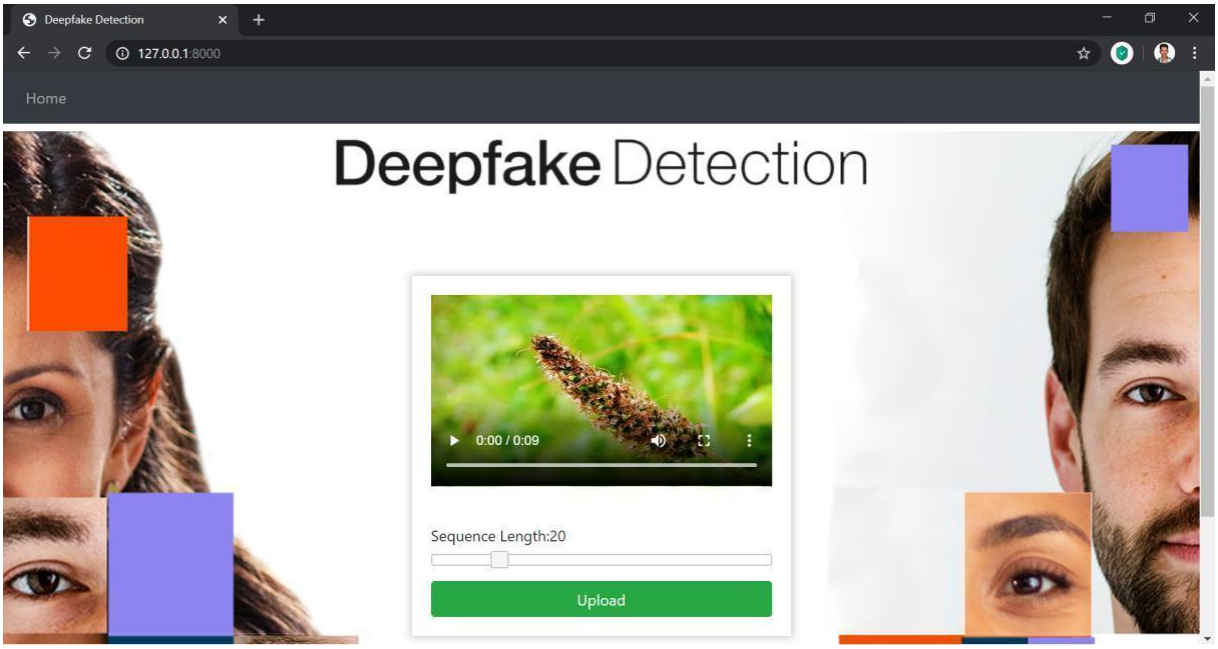

Figure 8.6: Uploading Video with no faces

Figure 8.7: Output of Uploaded video with no faces

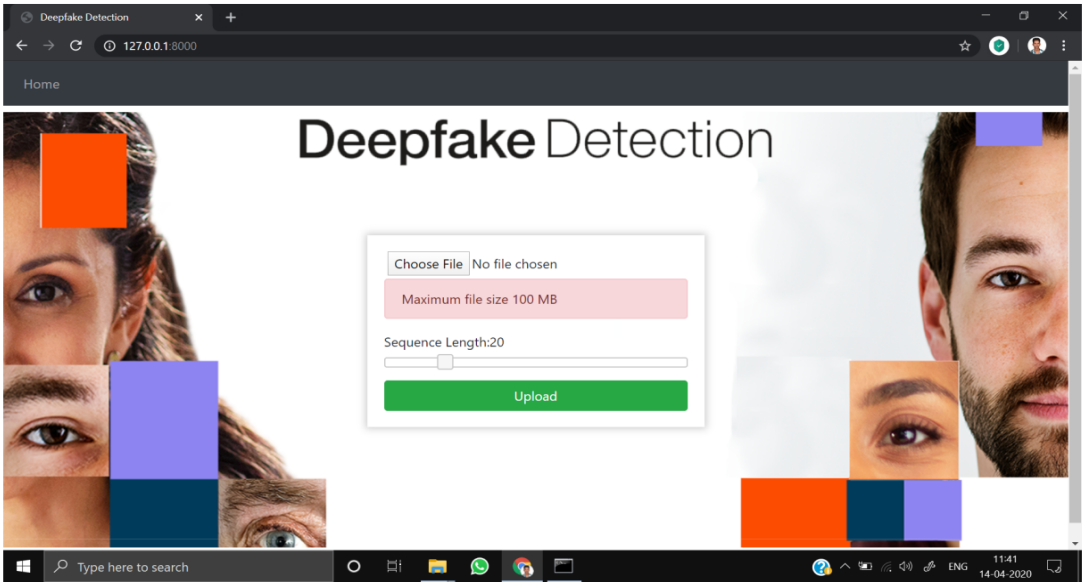

Figure 8.8: Uploading file greater than 100MB

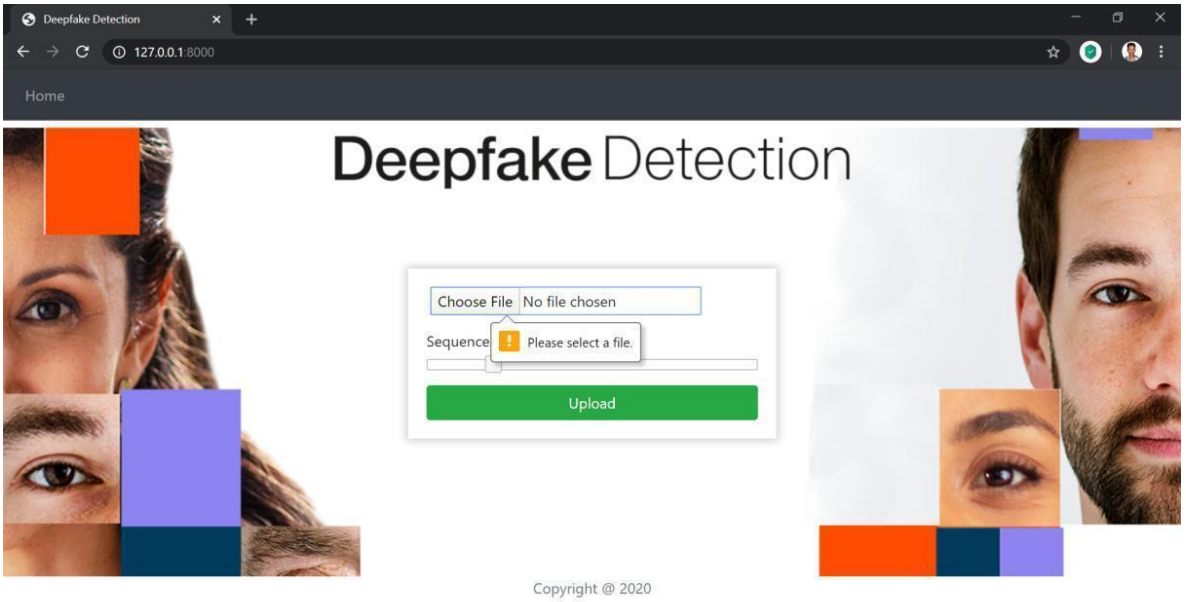

Figure 8.9: Pressing Upload button without selecting video

8.2 Outputs

8.2.1 Model results

Model Name	Dataset	No. of videos	Sequence length	Accuracy
model_90_acc_20_frames_FF_data	FaceForensic++	2000	20	90.95477
model_95_acc_40_frames_FF_data	FaceForensic++	2000	40	95.22613
model_97_acc_60_frames_FF_data	FaceForensic++	2000	60	97.48743
model_97_acc_80_frames_FF_data	FaceForensic++	2000	80	97.73366
model_97_acc_100_frames_FF_data	FaceForensic++	2000	100	97.76180
model_93_acc_100_frames_celeb_FF_data	Celeb-DF + FaceForensic++	3000	100	93.97781
model_87_acc_20_frames_final_data	Our Dataset	6000	20	87.79160
model_84_acc_10_frames_final_data	Our Dataset	6000	10	84.21461
model_89_acc_40_frames_final_data	Our Dataset	6000	40	89.34681

Table 8.1: Trained Model Result

Chapter 9

Conclusion and Future Scope

9.1 Conclusion

We have developed a neural network-based approach to classify videos as either deepfake or real, providing a measure of confidence in the predicted output. Our method demonstrates high accuracy by analyzing 1 second of video, equivalent to 10 frames per second.

The implementation of our model involves utilizing a pre-trained ResNext CNN model to extract features at the frame level. These features capture important visual information from each frame. We then employ an LSTM layer for temporal sequence processing, enabling the detection of changes between consecutive frames (t and t-1).

To optimize efficiency, our model processes video frames in specific sequences, including intervals of 10, 20, 40, 60, 80, and 100 frames. By considering these frame sequences, we effectively capture temporal dependencies and patterns within the video data.

Our approach offers a reliable and accurate solution for deepfake detection, providing both classification outcomes and confidence scores.

9.2 Future Scope

Continuous improvement and expansion are essential for any developed system, particularly when it leverages the latest technologies and holds significant potential for the future. In this case, the web-based platform can be further enhanced by transforming it into a browser plugin, providing users with convenient access to the functionality.

It is important to note that the current algorithm focuses solely on detecting face deepfakes. However, there is an opportunity to extend the capabilities of the algorithm to detect full-body deepfakes as well. By incorporating additional techniques and training data, the algorithm can be enhanced to identify manipulated videos that involve full-body movements and interactions.

By pursuing these enhancements, the system can remain at the forefront of deepfake detection technology, catering to evolving user needs and staying aligned with emerging trends in the field.

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/deepfakedetection-challenge/data Accessed on 26 March, 2023

[3] Yuezun Li , Xin Yang , Pu Sun , Honggang Qi and Siwei Lyu “Celeb-DF: A Largescale Challenging Dataset for DeepFake Forensics” in arXiv:1909.12962

[4] Face app: https://www.faceapp.com/ (Accessed on 26 March, 2023)

[5] Face Swap : https://faceswaponline.com/ (Accessed on 26 March, 2023)

[6] Yuezun Li, Siwei Lyu, “ExposingDF Videos By Detecting Face Warping Artifacts,” in arXiv:1811.00656v3.

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

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

[9] 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.

[10] 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

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

[12] The rise of the deepfake and the threat to democracy :Department of Electronics and Communication Engineering , IET Lucknow 51 Deepfake Video Detection

[13] https://www.geeksforgeeks.org/software-engineering-cocomo-model/ Accessed on 15 April 2023

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

[15] 10 deepfake examples that terrified and amused the internet : https://www.creativebloq.com/features/deepfake-examples Accessed on 26 March, 2023

[16] 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.

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

[18] ResNext Model : https://pytorch.org/hub/pytorch_vision_resnext/ accessed on 06 April 2023

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