

TrustWave AI: A Deepfake Detection System

A PROJECT REPORT

Submitted by

Aryan Soni (22BCE11274)

Abhay Pratap Singh (22BCE11210)

Siddhant Singh Bisht (22BCE10779)

Prateek Narain (22BCE10291)

Vrinda Devadas (22BCE11227)

*in partial fulfillment for the award of the degree
of*

BACHELOR OF TECHNOLOGY

in

PROGRAM OF STUDY



SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

VIT BHOPAL UNIVERSITY

KOTRIKALAN, SEHORE

MADHYA PRADESH - 466114

April 2024

BONAFIDE CERTIFICATE

Certified that this project report titled “**TrustWave AI: A Deepfake Detection System**” is the bonafide work of “**Aryan Soni(22BCE11274), Abhay Pratap Singh(22BCE11210), Siddhant Singh Bisht(22BCE10779), Prateek Narain (22BCE10291), Vrinda Devadas(22BCE11227)**” who carried out the project work under my supervision. Certified further that to the best of my knowledge the work reported here does not form part of any other project / research work on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

PROJECT SUPERVISOR

Dr. Pavithra R., Assistant Professor
School of Computer Science and Engineering
VIT BHOPAL UNIVERSITY

The Project Exhibition II Examination is held on 30/04/24

ACKNOWLEDGEMENT

First and foremost I would like to thank the Lord Almighty for His presence and immense blessings throughout the project work.

I wish to express my heartfelt gratitude to Dr.POONKUNTRAN S, Head of the Department, School of Computing Science and Engineering for much of his valuable support encouragement in carrying out this work.

I would like to thank my internal guide Dr.Pavithra R., for continually guiding and actively participating in my project, giving valuable suggestions to complete the project work.

I would like to thank all the technical and teaching staff of the School of Computer Science, who extended directly or indirectly all support.

Last, but not the least, I am deeply indebted to my parents who have been the greatest support while I worked day and night for the project to make it a success.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
1	INTRODUCTION <ul style="list-style-type: none"> ● 1.1.Introduction ● 1.2.Motivation of the project ● 1.3.Problem Statement ● 1.4.Goal and objective ● 1.5.Statement of scope ● 1.6.Methodologies of Problem solving 	8-12
2	LITERATURE SURVEY <ul style="list-style-type: none"> ● 2.1.Challenges ● 2.2.Core area of the project ● 2.3.Comparing TrustWave AI with Existing Platforms 	13-14
3	PROJECT PLAN <ul style="list-style-type: none"> ● 3.1.Project Model Analysis ● 3.2.Risk management w.r.t NP Hard analysis <ul style="list-style-type: none"> ○ 3.2.1.Risk Identification ○ 3.2.2.Risk Analysis ● 3.3. Project Task Set 	15-17
4	SOFTWARE REQUIREMENT SPECIFICATION <ul style="list-style-type: none"> ● 4.1.Introduction <ul style="list-style-type: none"> ○ 4.1.1.Purpose and scope of document ○ 4.1.2.Use case View ● 4.2.Functional Model and Description <ul style="list-style-type: none"> ○ 4.2.1.Data Flow Diagram ○ 4.2.2.Activity Diagram ○ 4.2.3.Non Functional Requirements 	18-22

	○ 4.2.4.Sequence Diagram	
5	DETAILED DESIGN DOCUMENT <ul style="list-style-type: none"> ● 5.1.Introduction <ul style="list-style-type: none"> ○ 5.1.1.System Architecture ● 5.2.Architectural Design <ul style="list-style-type: none"> ○ 5.2.1.Module1: Data-set Gathering ○ 5.2.2.Module2: Pre-Processing ○ 5.2.3.Module3: Data-set Splitting ○ 5.2.4.Module4: Model Architecture ○ 5.2.5.Module5: Hyper-parameter tuning 	23-28
6	PROJECT IMPLEMENTATION <ul style="list-style-type: none"> ● 6.1.Introduction ● 6.2.Tools and technologies Used 	29-32
7	SOFTWARE TESTING <ul style="list-style-type: none"> ● 7.1.Types of testing used ● 7.2.Test cases and Test Results 	33
8	RESULT <ul style="list-style-type: none"> ● 8.1.Results Achieved: 	34-38
9	FUTURE ENHANCEMENT AND CONCLUSION: <ul style="list-style-type: none"> ● 9.1.Conclusion ● 9.2.Future Scope 	39-40
10	REFERNCES	41-42

List of Tables

3.1 Risk Description.	17
3.2. Risk Probability definitions.	17
7.1 Test Case Report.	34

List of Figures

3.1 Spiral Methodology SDLC	15
4.1 Use case diagram	19
4.2 DFD Level 0.	20
4.3 DFD Level 1.	20
4.4 DFD Level 2.	21
4.5 Training Workflow.	21
4.6 Testing Workflow.	22
4.7 Sequence Diagram.	23
5.1 System Architecture.	24
5.2 Deepfake generation.	25
5.4 Dataset.	25
5.5 Pre-processing of video.	26
5.6 Train test split.	27
5.7 Overview of our model.	29

Chapter 1

Introduction

1.1. Introduction

This work proposes a novel deep learning-based method to effectively distinguish between real and AI-generated videos. Our system can detect both "replacement" and "reenactment" deepfakes, essentially using artificial intelligence (AI) to combat AI.

The System's Architecture:

- **Feature Extraction:** A Res-Next convolutional neural network efficiently extracts features from each frame of the video.
- **Classification:** These extracted features are then fed into a Long Short-Term Memory (LSTM) based recurrent neural network (RNN). The LSTM analyses the temporal sequence of features, ultimately classifying the video as real or manipulated (deepfake).

The term “Deepfake” is derived from “Deep Learning (DL)” and “Fake,” and it describes specific photo-realistic video or image contents created with DL’s support. This word was named after an anonymous Reddit user in late 2017, who applied deep learning methods for replacing a person’s face in pornographic videos using another person’s face and created photo-realistic fake videos. To generate such counterfeit videos, two neural networks:

- i. a generative network
- ii. a discriminative network with a FaceSwap technique were used

The generative network creates fake images using an encoder and a decoder.

The discriminative network defines the authenticity of the newly generated images.

The combination of these two networks is called Generative Adversarial Networks (GANs), proposed by Ian Goodfellow.

Based on a yearly report in Deepfake, DL researchers made several related breakthroughs in generative modeling. For example, computer vision researchers proposed a method known as Face2Face for facial re-enactment. This method transfers facial expressions from one person to a real digital 'avatar' in real-time.

In 2017, researchers from UC Berkeley presented CycleGAN to transform images and videos into different styles.

Another group of scholars from the University of Washington proposed a method to synchronize the lip movement in video with a speech from another source . Finally, in November 2017, the term “Deepfake” emerged for sharing porn videos, in which celebrities’ faces were swapped with the original ones.

In January 2018, a Deepfake creation service was launched by various websites based on some private sponsors. After a month, several websites, including Gfycat , Pornhub, and Twitter, banned these services. However, considering the threats and potential risks in privacy vulnerabilities, the study of Deepfake emerged super-fast.

Rossler et al. introduced a vast video dataset to train the media forensic and Deepfake detection tools called FaceForensic in March 2018.

After a month, researchers at Stanford University published a method, “Deep video portraits” that enables photo-realistic re-animation of portrait videos.

UC Berkeley researchers developed another approach for transferring a person’s body movements to another person in the video.

NVIDIA introduced a style-based generator architecture for GANs for synthetic image generation.

According to report, Google search engine could find multiple web pages that contain Deepfake related videos. We found the following additional information from this report:

- The top 10 pornographic platforms posted 1,790+ Deepfake videos, without concerning pornhub.com, which has removed ’Deepfakes’ searches.
- Adult pages post 6,174 Deepfake videos with fake video content.
- 3 New platforms were devoted to distributing Deepfake pornography.
- In 2018, 902 articles were published in arXiv, including the keyword GAN either in titles or abstracts.
- 25 Papers published on this subject, including non-peer reviews, are investigated, and DARPA funded 12 of them.

1.2. Motivation of the project

The increasing computational power has made deep learning algorithms incredibly powerful, enabling the creation of indistinguishable human-synthesized videos, commonly known as deepfakes. These realistic face-swapped deepfakes can be misused in scenarios such as creating political distress, spreading fake terrorism events, revenge porn, and blackmail. The motivation for this study arises to address this challenge. Here are some key points:

1. **Scale of Disruption:** Deep fakes can significantly disrupt trust and information integrity. For instance, consider the infamous BuzzFeed video titled “You Won’t Believe What Obama Says in This Video!” It demonstrated how deep fakes could convincingly imitate global leaders, leading to a break of trust between leaders and the public.
2. **Catalyst for Mayhem:** When a deep fake portrays a nation’s leader saying or doing something false, it can have far-reaching consequences. The resulting loss of trust can act as a catalyst for chaos and confusion within society.
3. **Defending Against Proliferation:** Detecting deep fakes accurately is crucial to prevent their widespread dissemination. By identifying features indicative of deep fake generation, we can develop systems that distinguish between real and manipulated content.
4. **Technological Advancements:** As deep fakes become more sophisticated, the need for robust detection systems grows. Researchers and engineers collaborate to create effective methods for recognizing distorted media, even as deep fake technology evolves rapidly

1.3. Problem Statement

For decades, manipulating digital media has been possible through visual effects. However, recent advancements in deep learning have dramatically increased the realism and accessibility of creating fake content, also known as deepfakes. While generating deepfakes with AI tools has become easier, detecting them remains a significant challenge.

The widespread use of deepfakes poses a serious threat:

- **Spreading misinformation:** Deepfakes can be used to manipulate public opinion and sow discord by creating fake news and political propaganda.

- **Harming individuals:** They can be used to create revenge porn, defame individuals, or blackmail them.

Therefore, detecting deepfakes is crucial to prevent their spread and harmful impacts. This project proposes a deep learning approach using a Long Short-Term Memory (LSTM) based artificial neural network to address this challenge.

1.4. Goal and objective

Deepfakes are created using machine learning algorithms to manipulate or replace parts of an original video or image, often altering a person's face or other features. These manipulated videos can be maliciously distributed, leading to political attacks or social problems. The objective of a deepfake detection system is to ensure transparency, objectivity, and accuracy in identifying such content.

- **Unmask the truth:** Our project aims to discern the genuineness of videos and expose deepfakes.
- **Combat misinformation:** We aim to reduce the spread of misleading content and protect internet users from online manipulation.

Accurate classification: Our system will effectively distinguish between real videos and deepfakes.

1.5. Statement of scope

There are many tools available for creating the deep fakes, but for deep fake detection there is hardly any tool available. Our approach for detecting the deepfakes will be great contribution in avoiding the percolation of the deepfakes over the world wide web. We will be providing a web-based platform for the user to upload the video and classify it as fake or real. This project can be scaled up from developing a web-based platform to a browser plugin for automatic deepfake detection's. Even big application like WhatsApp, Facebook can integrate this project with their application for easy pre-detection of deep fakes before sending to another user. A description of the software with Size of input, bounds on input, input validation, input dependency, i/o state diagram, major inputs, and outputs are described without regard to implementation detail.

1.6. Methodologies of Problem solving

To mimic real-world situations and improve performance, the system is trained on a large, balanced, and mixed dataset. This dataset combines various existing resources like:

- **Celeb-DF**: Contains real and deepfake videos of celebrities.
- **Deepfake Detection Challenge**: A large-scale dataset with diverse deepfake videos.
- **FaceForensics++**: Includes manipulated videos generated using various techniques.

This diverse training approach contributes to the system's robustness and competitive accuracy, achieved with a relatively simple and reliable architecture.

Chapter 2

Literary survey

2.1. Challenges

1. Variability in Deepfake Quality - The quality and diversity of datasets used for training detection models significantly impact their performance. Datasets should cover a wide range of contexts, variations, and scenarios to ensure robustness.
2. Limited availability of training data - Deepfake detection models require large-scale datasets for training. Availability of such datasets is crucial for improving the performance of detection algorithms.
3. Resource constraints - collecting and curating large-scale datasets can be resource-intensive.
4. Advancement in deepfake generation techniques - The field of deepfakes is constantly evolving. New techniques and tools emerge frequently. Detection systems need to keep pace with the latest developments to remain effective. Researchers must stay informed about the most recent advancements in deepfake generation and detection technologies.
5. Interpretability of model decisions - These models learn intricate patterns and features from data, but understanding their internal decision-making process is challenging.
6. Accuracy - Deepfakes can impact legal proceedings, potentially prolonging trials and leading to false assumptions. Detecting deepfakes accurately is crucial for maintaining the integrity of legal processes.

2.2. Core area of the project

Our project is a Deep learning project which is a sub branch of Artificial Intelligence and deals with the human brain inspired neural network technology. Computer vision plays an important role in our project. It helps in processing the video and frames with the help of Open-CV, we can perform tasks like object detection, tracking, and image manipulation. A PyTorch trained model is a classifier to classify the source video as deepfake or pristine.

2.3. Comparing TrustWave AI with Existing Platforms

- **Innovative Approach:**
 - Your system introduces an innovative deep learning-based technique specifically designed for differentiating between AI-generated fake videos (deepfakes) and genuine ones.
 - This approach showcases your commitment to staying at the forefront of research and development.
- **ResNext CNN and LSTM-Based RNN:**
 - The central component of your system leverages a ResNext Convolutional Neural Network (CNN).
 - This CNN extracts features from individual frames of videos.
 - These features are then used to train a Long Short-Term Memory (LSTM)-based Recurrent Neural Network (RNN).
 - The LSTM-RNN helps classify videos, determining if they are manipulated (deepfakes) or genuine recordings.
- **Robustness and Real-Time Performance:**
 - Your model is evaluated on a large and balanced dataset that combines various existing datasets (e.g., FaceForensic++, Deepfake Detection Challenge, and Celeb-DF).
 - This ensures that your system performs well with real-time data and reflects real-world situations.
 - Achieving competitive results using a straightforward yet effective approach
- **Understanding the Implications:**
 - You recognize the profound implications of deepfake videos, including their impact on individuals, society, and political contexts.
 - By combating deepfakes using AI, you're addressing the challenges posed by your own creations.
- **Holistic Considerations:**
 - Your system doesn't just focus on technical aspects; it also considers ethical, legal, and societal implications.
 - This holistic approach sets it apart from platforms that may overlook these critical dimensions.

Chapter 3

Project Plan

3.1. Project Model Analysis

We've chosen to implement the Spiral model in our software development process due to its emphasis on collaboration, risk management, and adaptability. This model facilitates rapid changes and continuous evaluations, aligning with our goal of ensuring the product meets the desired outcomes. With our application being developed in modules, the Spiral model is well-suited to our needs. It allows for client involvement throughout the project, from feature prioritization to iteration planning and reviews, enabling transparency and frequent updates. However, it's important for clients to understand that they'll be witnessing a work in progress as part of this transparent approach. The Spiral model's capability to address risks aligns with the risk-intensive nature of our project, making it the ideal choice for our product development.

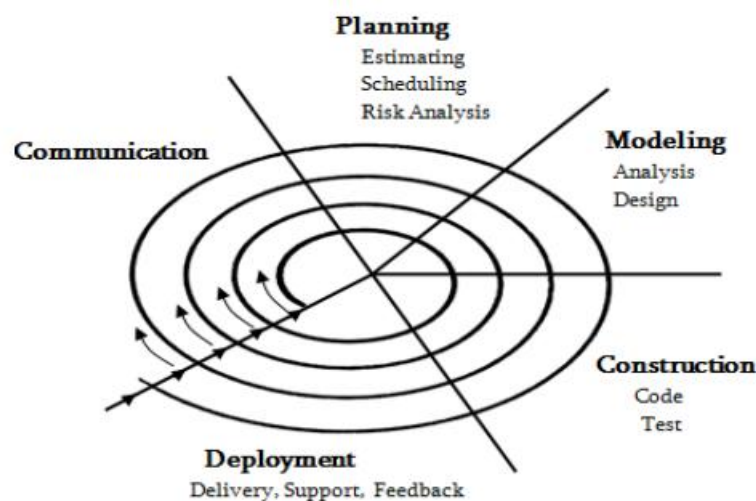


Figure 5.1: Spiral Methodology SDLC

Figure 3.1: Spiral Methodology SDLC

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

3.2.1 Risk Management

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

1. Remove any picture frames that contain more than one person.

2. Make sure you have an abundance of video footage. Extract facial pictures contain different pose, face angle, and facial expressions.
3. Some resembling of both persons may help, like similar face shape.

3.2.2 Risk Analysis

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

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

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

Table 3.1 Risk Description

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

Table 3.2 Risk Probability Definitions

3.3. Project Task set

Major task in the project stages are

Task 1: Data-set gathering and analysis

This task consists of downloading the dataset. Analyzing the dataset and making the dataset ready for the processing

Task 2 : Module 1 implementation

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

Task 3: Pre-processing

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

Task 4: Module 2 implementation

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

Task 5 : Hyper parameter tuning

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

Task 6 : Training the final model

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

Task 7 : 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 document lays out a project plan for the development of Deepfake video detection using neural network. The intended readers of this document are current and future developers working on Deepfake video detection using neural network and the sponsors of the project. The plan will include, but is not restricted to, a summary of the system functionality, the scope of the project from the perspective of the “Deepfake video detection” team (me and my mentors), use case diagram, Data flow diagram, activity diagram, functional and non- functional requirements, project risks and how those risks will be mitigated, the process by which we will develop the project, and metrics and measurements that will be recorded throughout the project.

4.1.2 Use Case view

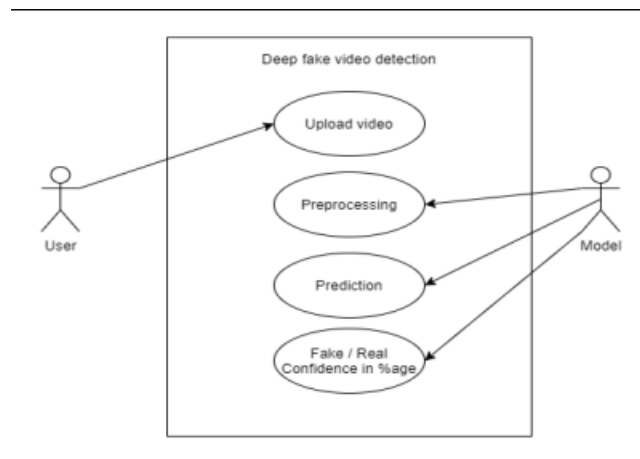


Figure 4.1: Use Case diagram

4.2 Functional Model and Description

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

4.2.1. Data Flow Diagram

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

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

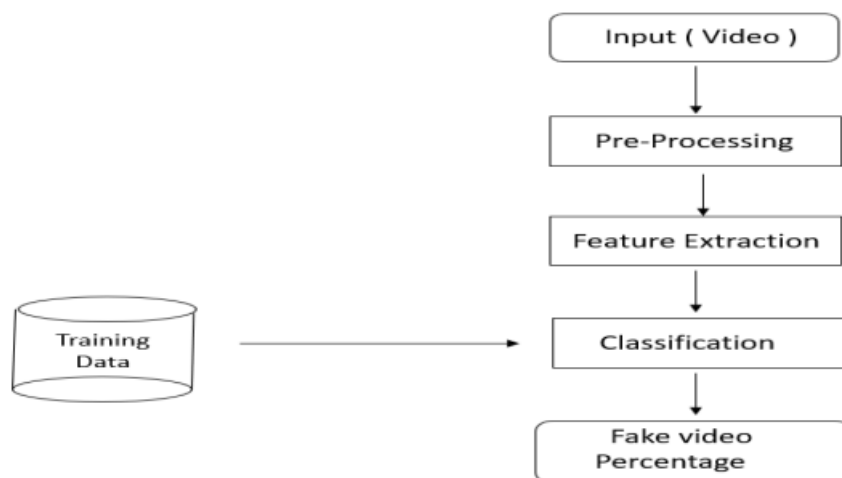


Figure 4.3: DFD Level 1

DFD Level-2

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

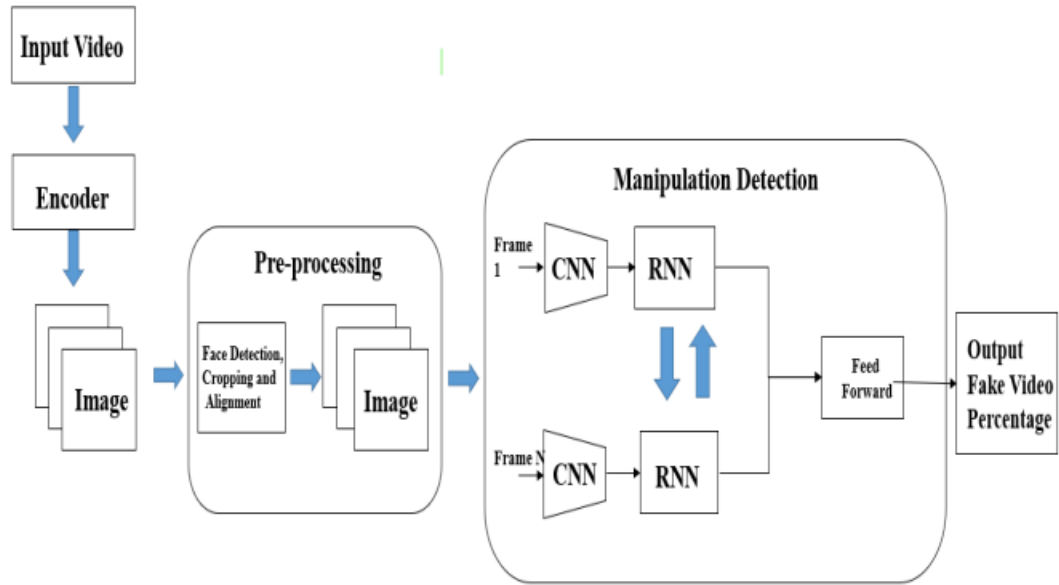


Figure 6.4: DFD Level 2

6.2.2 Activity Diagram

Training Workflow

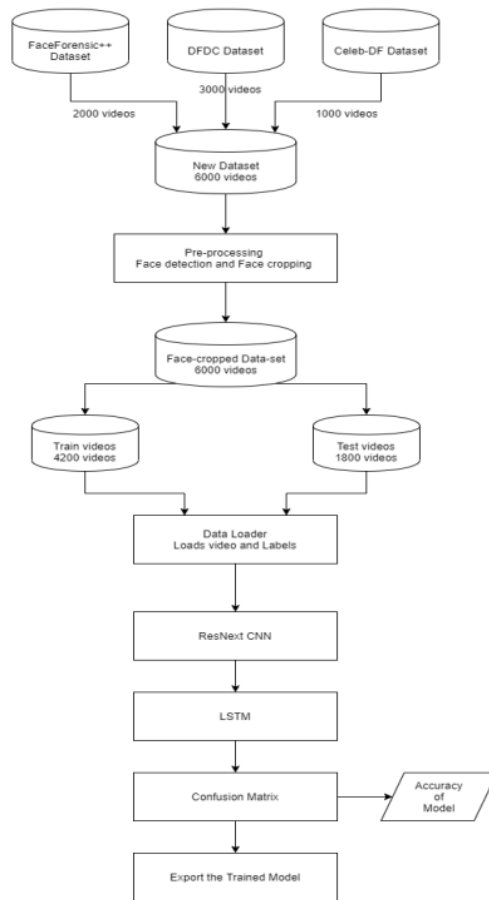


Figure 4.5: Training Workflow

Testing Workflow:

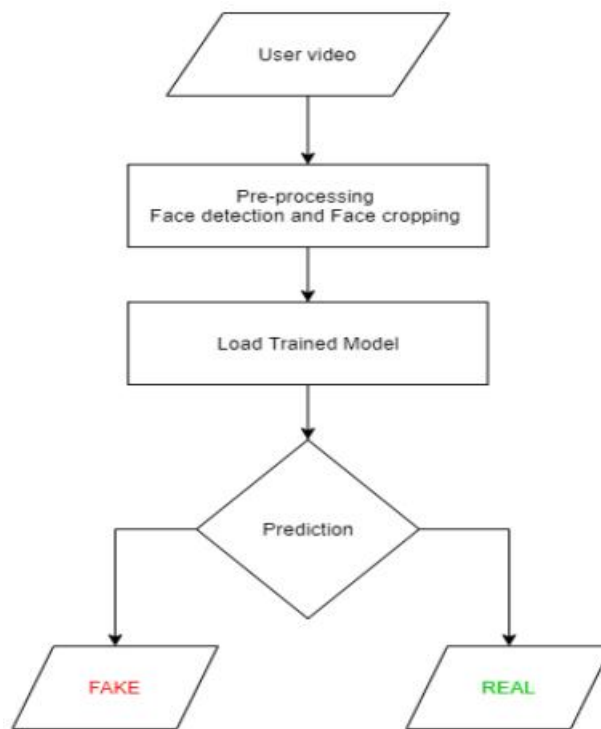


Figure 4.6: Testing Workflow

4.2.3 Non Functional Requirement

Performance Requirement

- The software should be efficiently designed so as to give reliable recognition of fake videos and so that it can be used for more pragmatic purpose.
- The design is versatile and user friendly.
- The application is fast, reliable and time saving.
- The system have universal adaptations.
- The system is compatible with future upgradation and easy integration.

Safety Requirement

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

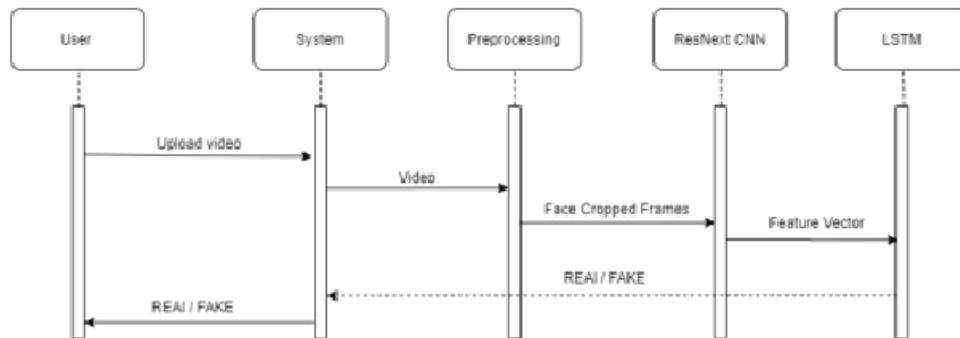


Figure 4.7: Sequence Diagram

Chapter 5

Detail Design Document

5.1. Introduction

5.1.1 System Architecture

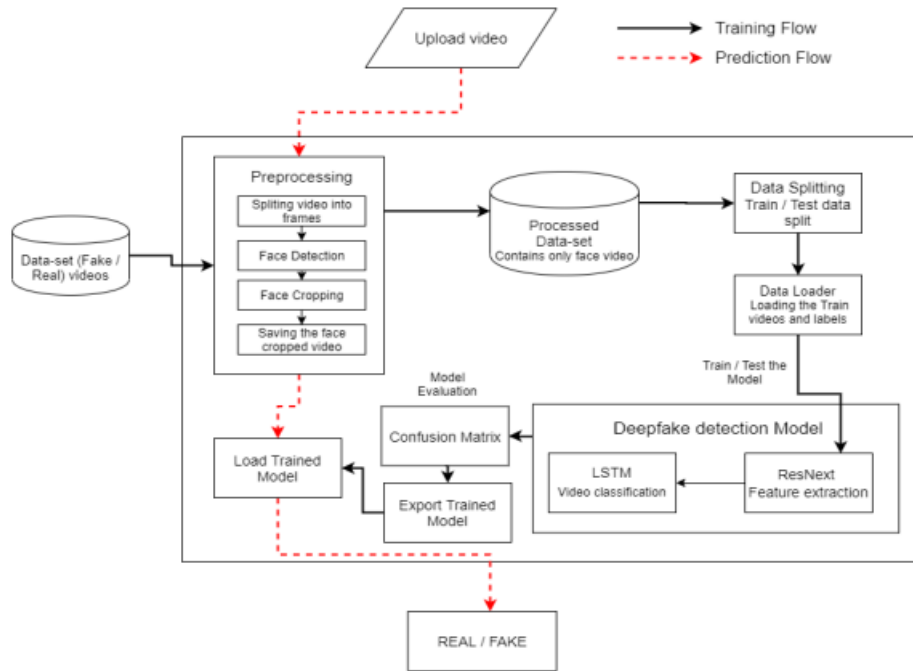


Figure 5.1 System Architecture

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

- **Creating deepfake videos**

To detect the deepfake videos it is very important to understand the creation process of the deepfake. Majority of the tools including the GAN and autoencoders takes a source image and target video as input. These tools split the video into frames , detect the face in the video and replace the source face with target face on each frame. Then the replaced frames are then combined using different pre-trained models. These models also enhance the quality of video my removing the left-over traces by the deepfake creation model. Which result in creation of a deepfake looks realistic in nature. We have also used the same approach to detect the deepfakes. Deepfakes created using the pretrained neural networks models are very realistic

that it is almost impossible to spot the difference by the naked eyes. But in reality, the deepfakes creation tools leaves some of the traces or artifacts in the video which may not be noticeable by the naked eyes. The motive of this paper to identify these unnoticeable traces and distinguishable artifacts of these videos and classified it as deepfake or real video.

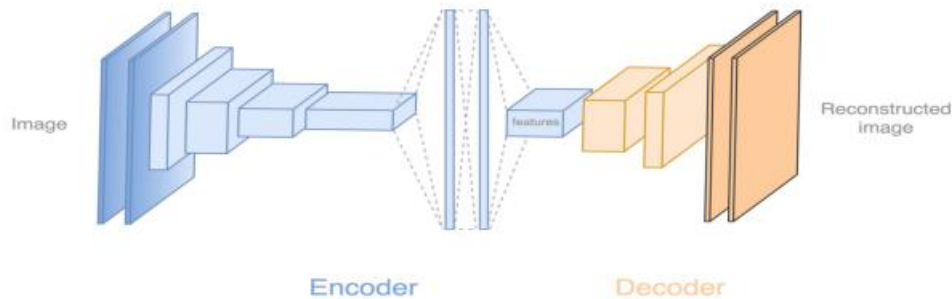


Figure 5.2 Deepfake generation

Tools for deep fake creation.

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

5.2. Architecture Desing

5.2.1 Module 1: Data-set Gathering

For making the model efficient for real time prediction. We have gathered the data from different available data-sets like FaceForensic++(FF), Deepfake detection challenge(DFDC), and Celeb-DF. Further we have mixed the dataset the collected datasets and created our own new dataset, to accurate and real time detection on different kind of videos. To avoid the training bias of the model we have considered 50% Real and 50% fake videos. Deep fake detection challenge (DFDC) dataset consist of certain audio alerted video, as audio deepfake are out of scope for this paper. We preprocessed the DFDC dataset and removed the audio altered videos from the dataset by running a python script. After preprocessing of the DFDC dataset, we have taken 1500 Real and 1500 Fake videos from the DFDC dataset. 1000 Real and 1000 Fake videos from the FaceForensic++(FF) dataset and 500 Real and 500 Fake videos from the CelebDF dataset. Which makes our total dataset consisting 3000 Real, 3000 fake videos and 6000 videos in total. Figure 2 depicts the distribution of the data-sets.

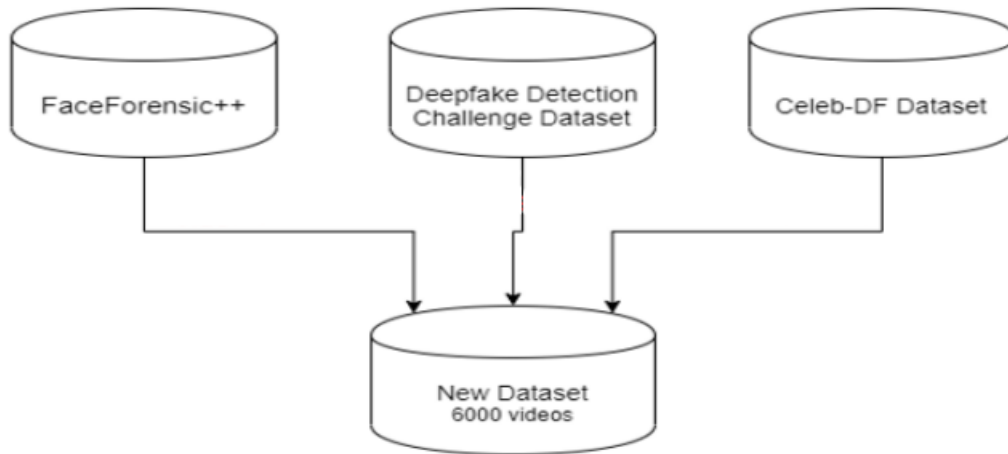


Figure 5.4: Dataset

5.2.2 Module 2: Pre-processing

In this step, the videos are preprocessed and all the unrequired and noise is removed from videos. Only the required portion of the video i.e. face is detected and cropped. The first steps in the preprocessing of the video is to split the video into frames. After splitting the video into frames the face is detected in each of the frame and the frame is cropped along the face. Later the cropped frame is again converted to a new video by combining each frame of the video. The process is followed for each video which leads to creation of processed dataset containing face only videos. The frame that does not contain the face is ignored while preprocessing. To maintain the uniformity of number of frames, we have selected a threshold value based on the mean of total frames count of each video. Another reason for selecting a threshold value is limited computation power. As a video of 10 second at 30 frames per second(fps) will have total 300 frames and it is computationally very difficult to process the 300 frames at a single time in the experimental environment. So, based on our Graphic Processing Unit (GPU) computational power in experimental environment we have selected 150 frames as the threshold value. While saving the frames to the new dataset we have only saved the first 150 frames of the video to the new video. To demonstrate the proper use of Long Short-Term Memory (LSTM) we have considered the frames in the sequential manner i.e. first 150 frames and not randomly. The newly created video is saved at frame rate of 30 fps and resolution of 112 x 112.

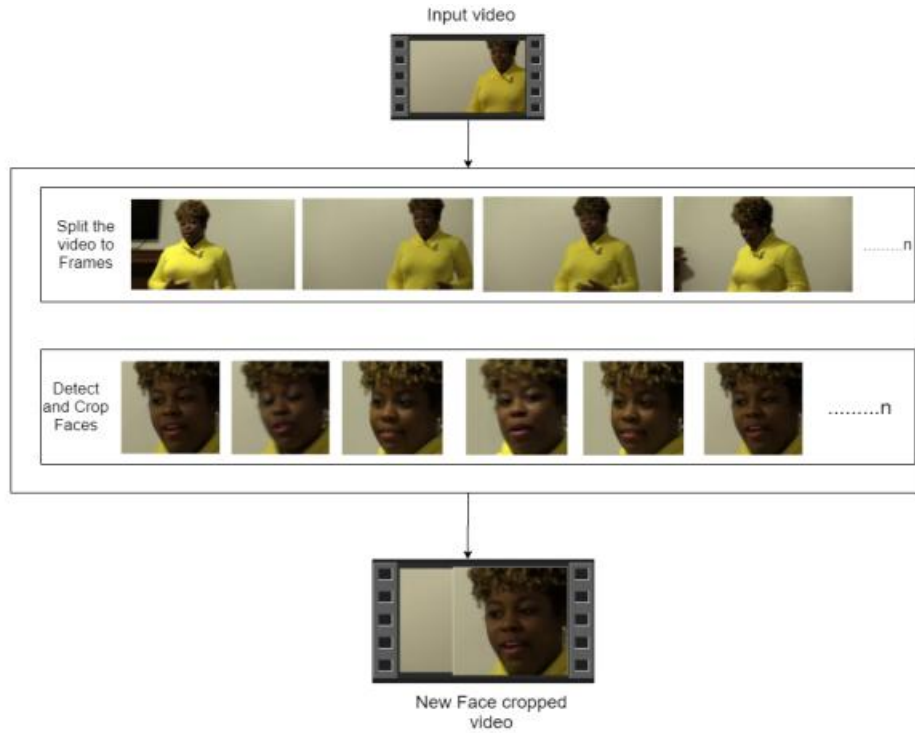


Figure 5.5: Pre-processing of video

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.

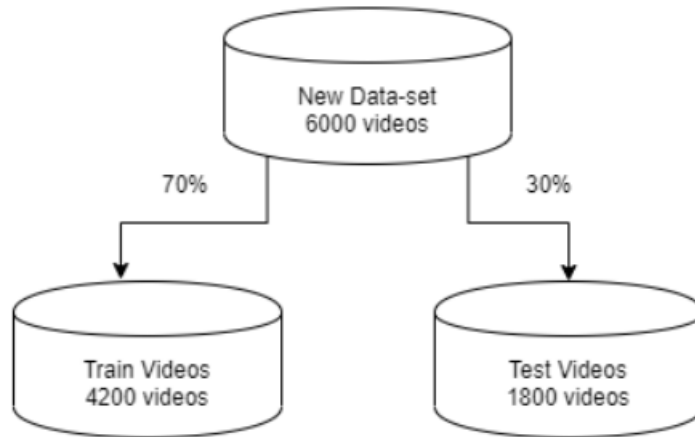


Figure 5.6: Train test split

5.2.4 Module 4: Model Architecture

Our model is a combination of CNN and RNN. We have used the Pre- trained ResNext CNN model to extract the features at frame level and based on the extracted features a LSTM network is trained to classify the video as deepfake or pristine. Using the Data Loader on

training split of videos the labels of the videos are loaded and fitted into the model for training.

ResNext :

Instead of writing the code from scratch, we used the pre-trained model of ResNext for feature extraction. ResNext is Residual CNN network optimized for high performance on deeper neural networks. For the experimental purpose we have used resnext50_32x4d model. We have used a ResNext of 50 layers and 32 x 4 dimensions. Following, we will be fine-tuning the network by adding extra required layers and selecting a proper learning rate to properly converge the gradient descent of the model. The 2048-dimensional feature vectors after the last pooling layers of ResNext is used as the sequential LSTM input.

LSTM for Sequence Processing:

2048-dimensional feature vectors is fitted as the input to the LSTM. We are using 1 LSTM layer with 2048 latent dimensions and 2048 hidden layers along with 0.4 chance of dropout, which is capable to do achieve our objective. LSTM is used to process the frames in a sequential manner so that the temporal analysis of the video can be made, by comparing the frame at 't' second with the frame of 't-n' seconds. Where n can be any number of frames before t. The model also consists of Leaky Relu activation function. A linear layer of 2048 input features and 2 output features are used to make the model capable of learning the average rate of correlation between eh input and output. An adaptive average polling layer with the output parameter 1 is used in the model. Which gives the target output size of the image of the form H x W. For sequential processing of the frames a Sequential Layer is used. The batch size of 4 is used to perform the batch training. A SoftMax layer is used to get the confidence of the model during predication.

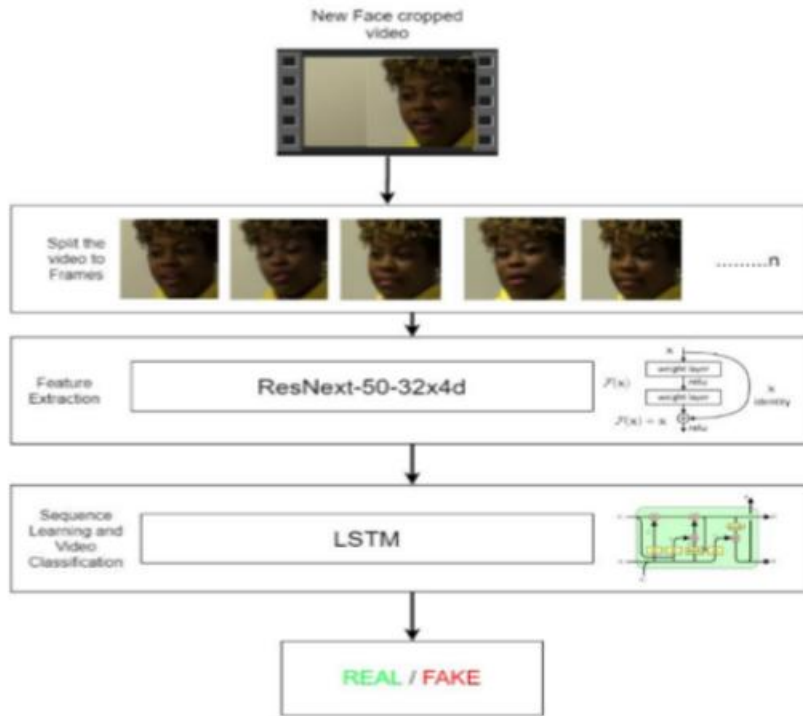


Figure 5.7: Overview of our model

5.2.5 Module 5: Hyper-parameter tuning

It is the process of choosing the perfect hyper-parameters for achieving the maximum accuracy. After reiterating many times on the model. The best hyper-parameters for our dataset are chosen. To enable the adaptive learning rate Adam optimizer with the model parameters is used. The learning rate is tuned to $1e-5$ (0.00001) to achieve a better global minimum of gradient descent. The weight decay used is $1e-3$. As this is a classification problem so to calculate the loss cross entropy approach is used. To use the available computation power properly the batch training is used. The batch size is taken of 4. Batch size of 4 is tested to be ideal size for training in our development environment. The User Interface for the application is developed using Django framework. Django is used to enable the scalability of the application in the future. The first page of the User interface i.e index.html contains a tab to browse and upload the video. The uploaded video is then passed to the model and prediction is made by the model. The model returns the output whether the video is real or fake along with the confidence of the model. The output is rendered in the predict.html on the face of the playing video.

Chapter 6

Project implementation

6.1 Introduction

There are many examples where deepfake creation technology is used to mislead the people on social media platform by sharing the false deepfake videos of the famous personalities like Mark Zuckerberg Eve of House A.I. Hearing, Donald Trump's Breaking Bad series where he was introduced as James McGill, Barack Obama's public service announcement and many more. These types of deepfakes create 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 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 that generate deepfake videos are being extensively used to create fake celebrity pornographic videos or revenge porn. Some of the examples are Brad Pitt, Angelina Jolie nude videos. The real looking nature of the deepfake videos makes the celebrities and other famous personalities the target of pornographic material, fake surveillance videos, fake news and malicious hoaxes. The Deepfakes are very much popular in creating the political tension. 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. Programming language

1. Python3

Backend Logic: Python is widely used for backend development due to its simplicity, readability, and extensive libraries. Here's how Python can contribute to our system:

- **Deep Learning Model:** Since our project involves deepfake detection, Python (especially with libraries like PyTorch) is ideal for training and deploying neural networks.
- **Data Preprocessing:** Python can handle data preprocessing tasks such as video frame extraction, resizing, and normalization.
- **APIs and Web Services:** Use Python to create RESTful APIs or GraphQL endpoints for communication between frontend and backend.
- **Database Interaction:** Python integrates seamlessly with databases (e.g., SQLite, PostgreSQL, MongoDB) for storing metadata or model weights.
- **Task Scheduling:** Python's cron jobs or libraries like schedule can automate tasks (e.g., model retraining)

2. JavaScript

Frontend Interaction: JavaScript is essential for frontend development. It runs directly in web browsers and allows dynamic interactions with users:

- **User Interface (UI):** Use JavaScript frameworks like React, Angular, or Vue.js to build interactive UI components.
- **Video Playback:** JavaScript can handle video playback directly in the browser using HTML5 video tags or third-party libraries (e.g., Video.js).
- **User Input Validation:** Validate user inputs (e.g., video file uploads) using JavaScript before sending them to the backend.
- **AJAX Requests:** Make asynchronous requests to our Python backend using JavaScript's fetch API or libraries like Axios.

Programming Frameworks

1. **PyTorch** - PyTorch is a popular deep learning framework that provides dynamic computation graphs and extensive support for neural network architectures.
2. **Django** - Django is a robust Python web framework that simplifies building web applications. It provides features like routing, database management, user authentication, and templating.

Version control

1. Git

Libraries

1. **torch and torchvision:**

- PyTorch is used to create, train, and deploy deep learning models (such as CNNs) for deepfake classification.
- torchvision provides pre-trained models (e.g., ResNet, DenseNet) and datasets (e.g., ImageNet) that can be fine-tuned for your specific task.

2. **os:**

- Used to manage file paths, create directories, and handle file I/O (e.g., reading video files).

3. **numpy:**

- NumPy handles numerical operations efficiently (e.g., matrix operations, statistical calculations).

4. **cv2 (OpenCV):**

- Reads video files, extract frames, and preprocess them (e.g., resizing, normalization).
- Performs face detection and alignment.

5. **matplotlib:**

- Visualizes training/validation curves, model performance metrics, and sample frames.

6. **face_recognition:**

- Detects faces in video frames.
- Extracts facial features (landmarks) for further analysis.
- Compares faces for similarity (e.g., to identify deepfake manipulations).

7. **json:**

- Used to store metadata about labels (e.g., deepfake or pristine) associated with video files.
- Maintains information about training/validation splits, model configurations, and hyperparameters.

8. **pandas:**

- Organizes and analyzes metadata (e.g., label information) in tabular form.
- Loads CSV files containing video metadata.

9. **copy:**

- Used to create copies of objects (e.g., model weights) without modifying the original.

10. **glob:**

- Used to search for video files in directories.
- Iterate through files for preprocessing and training.

11. **random:**

- Shuffle the dataset during training (e.g., random sampling of batches).

12. **sklearn (Scikit-learn):**

- Used for metrics (e.g., accuracy, precision, recall) evaluation.
- Implement preprocessing steps (e.g., feature scaling).

Chapter 7

Software testing

7.1 Type of testing used

Functions Testing

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

Non-functional Testing

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

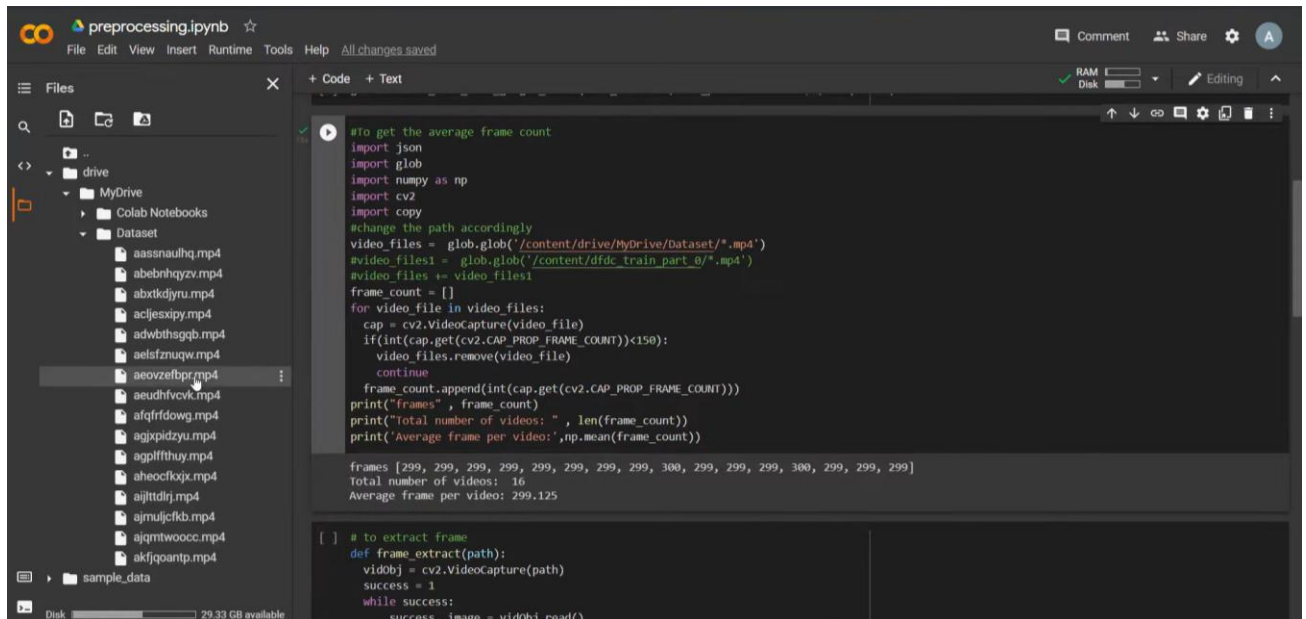
7.2 Test Cases and Test Results

Case ID	Test Case Description	Expected Result	Actual Result	Status
1	Upload a word file instead of video	Error message: Only video files allowed	Error message: Only video files allowed	Pass
2	Upload a 200MB video file	Error message: Max limit 100MB	Error message: Max limit 100MB	Pass
3	Upload a file without any faces	Error message: No faces detected. Cannot process the video.	Error message: No faces detected. Cannot process the video.	Pass
4	Videos with many faces	Fake./Real	Fake	Pass
5	Deepfake video	Fake	Fake	Pass
6	Enter /predict in URL	Redirect to /upload	Redirect to /upload	Pass
7	Press upload button without selecting video	Alert message: Please select video	Alert message: Phase 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

Result

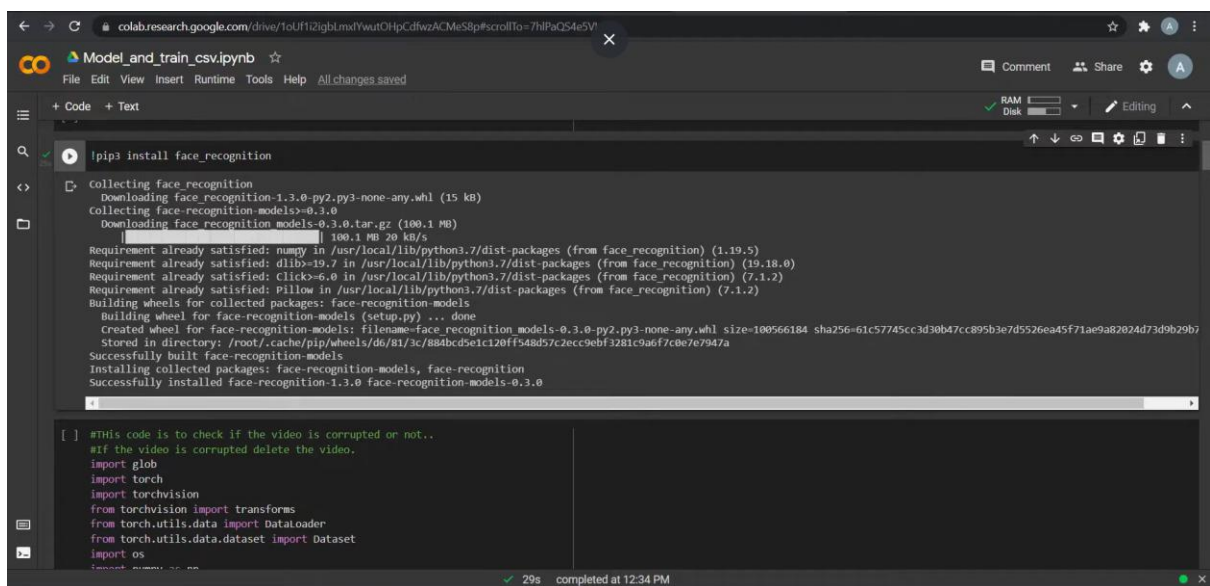


```
#to get the average frame count
import json
import glob
import numpy as np
import cv2
import copy

#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, 299, 299, 300, 299, 299, 299, 299, 299]
Total number of videos: 16
Average frame per video: 299.125

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



```
!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
  Downloading face_recognition_models-0.3.0.tar.gz (100.1 MB)
    100.1 MB 20 kB/s
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from face_recognition) (1.19.5)
Requirement already satisfied: dlib>=19.7 in /usr/local/lib/python3.7/dist-packages (from face_recognition) (19.18.0)
Requirement already satisfied: Click>=6.0 in /usr/local/lib/python3.7/dist-packages (from face_recognition) (7.1.2)
Requirement already satisfied: Pillow in /usr/local/lib/python3.7/dist-packages (from face_recognition) (7.1.2)
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=100566184 sha256=61c57745cc3d30b47cc895b3e7d5526ea45f71ae9a82024d73d9b29b7
  Stored in directory: /root/.cache/pip/wheels/d6/81/3c/884bcd5e1c120ff548d57c2ecc9ebf3281c9a6f7c0e7e7947a
Successfully built face_recognition_models
Installing collected packages: face_recognition_models, face_recognition
Successfully installed face_recognition-1.3.0 face_recognition_models-0.3.0

[ ] #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
```

```
Model_and_train_csv.ipynb ☆
File Edit View Insert Runtime Tools Help

+ Code + Text

transforms.ToPILImage(),
transforms.Resize((im_size,im_size)),
transforms.ToTensor(),
transforms.Normalize(mean,std))

video_fil = glob.glob('/content/drive/My Drive/Face_only_data/*.mp4')
print('Total no of videos :', len(video_fil))
print(video_fil)
count = 0
for i in video_fil:
    try:
        count+=1
        validate_video(i,train_transforms)
    except:
        print("Number of video processed: " , count , " Remaining : " , (len(video_fil) - count))
        print("Corrupted video is : " , i)
        continue
print((len(video_fil) - count))

... Total no of videos : 16
['/content/drive/My Drive/Face_only_data/aeovzefbpr.mp4', '/content/drive/My Drive/Face_only_data/akfjqoantp.mp4', '/content/drive/My Drive/Face_only_data/agplffthuy.mp4', '/content/dr
Number of video processed: 2 Remaining: 14
Corrupted video is : /content/drive/My Drive/Face_only_data/akfjqoantp.mp4

[ ] #to load preprocessed video to memory
import json
import glob
import numpy as np
import cv2
import copy
```

Executing (10s) Cell > validate_video() > frame_extract()

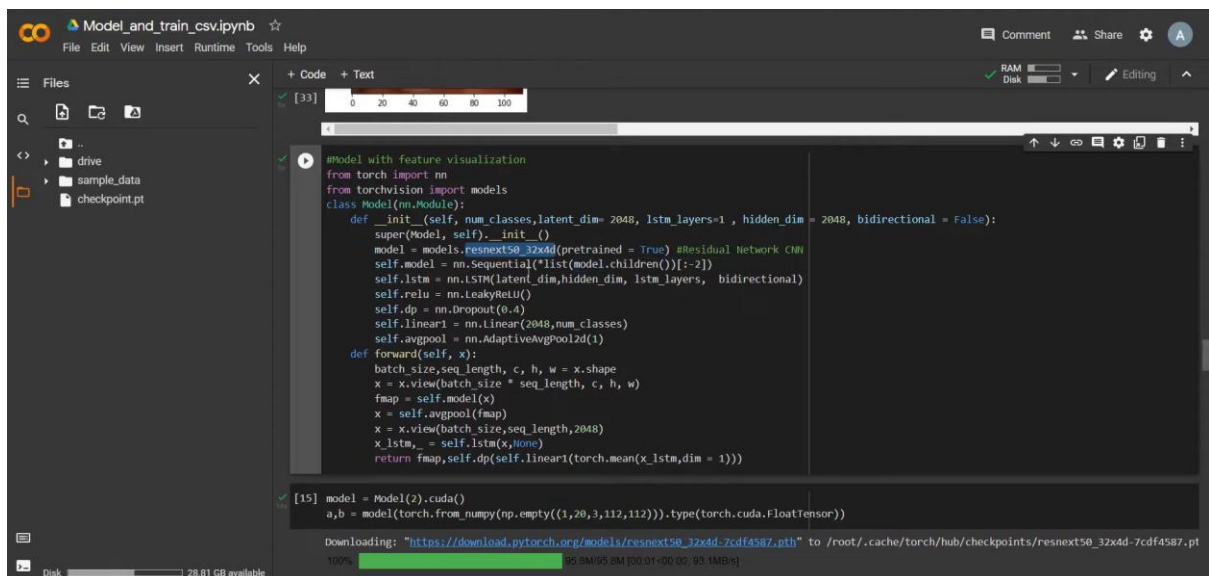
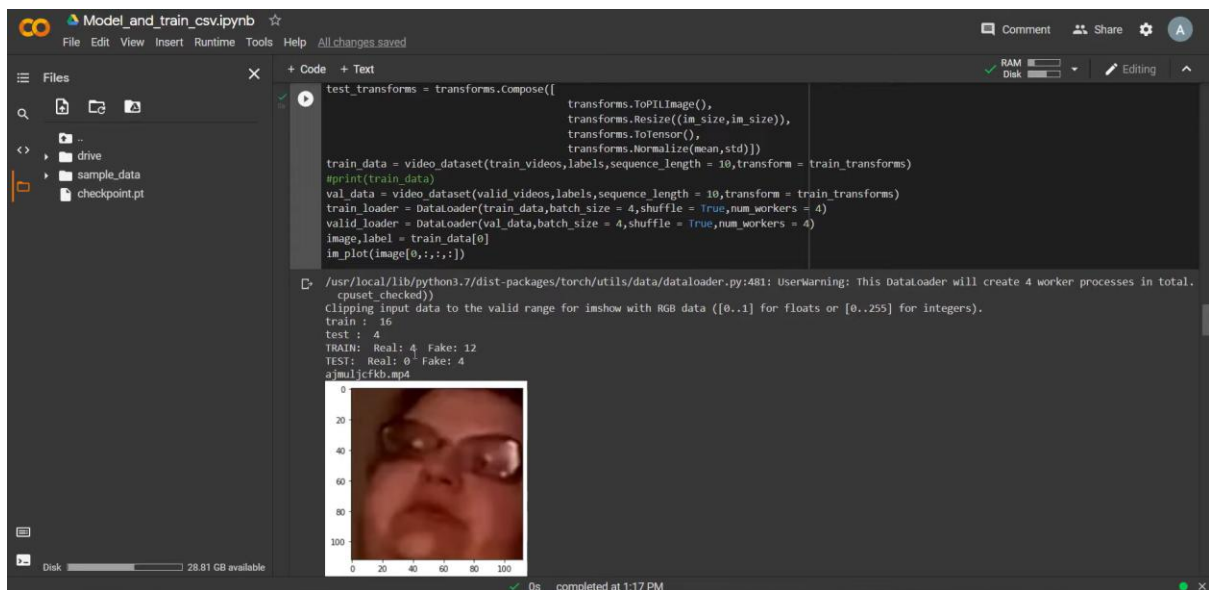
```
Model_and_train_csv.ipynb ☆
File Edit View Insert Runtime Tools Help

+ Code + Text

import numpy as np
import cv2
import copy
import random
video_files = glob.glob('/content/drive/My Drive/Face_only_data/*.mp4')
random.shuffle(video_files)
random.shuffle(video_files)
frame_count = []
for video_file in video_files:
    cap = cv2.VideoCapture(video_file)
    if(int(cap.get(cv2.CAP_PROP_FRAME_COUNT))<100):
        video_files.remove(video_file)
        continue
    frame_count.append(int(cap.get(cv2.CAP_PROP_FRAME_COUNT)))
print("frames are " , frame_count)
print("Total-no of video: " , len(frame_count))
print('Average frame per video:',np.mean(frame_count))

[ ] # 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
```

0s completed at 12:37 PM





Predict.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Code + Text

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

train_transforms = transforms.Compose([
    transforms.ToPILImage(),
    transforms.Resize((im_size,im_size)),
    transforms.ToTensor(),
    transforms.Normalize(mean,std)])

path_to_videos = ['/content/drive/My Drive/Balanced Face only data/aagfhgtpmv.mp4',
                  '/content/drive/My Drive/Balanced Face only data/aczrgyricp.mp4',
                  '/content/drive/My Drive/Balanced Face only data/apdkmztvby.mp4',
                  '/content/drive/My Drive/Balanced Face only data/abarnvbtwb.mp4']

path_to_videos = ['/content/drive/My Drive/Youtube Face only data/000_003.mp4',
                  '/content/drive/My Drive/Youtube Face only data/000.mp4',
                  '/content/drive/My Drive/Youtube Face only data/002_006.mp4',
                  '/content/drive/My Drive/Youtube Face only data/002.mp4']

]

path_to_videos= ["/content/drive/My Drive/DFDC_REAL Face only data/aabqyygbaa.mp4"]

video_dataset = validation_dataset(path_to_videos,sequence_length = 20,transform = train_transforms)
model = Model(2).cuda()
path_to_model = '/content/drive/My Drive/Models/model 87 acc 20 frames final data.pt'
model.load_state_dict(torch.load(path_to_model))
model.eval()
for i in range(0,len(path_to_videos)):
    print(path_to_videos[i])
```

0/0s completed at 1:32 PM

Predict.ipynb

File Edit View Insert Runtime Tools Help

Files

drive

MyDrive

Colab Notebooks

Dataset

Face_only_data

Global_metadata.csv

checkpoint.pt

sample_data

+ Code + Text

```

video_dataset = ValidationDataset(path_to_videos, sequence_length = 16, transform = train_transforms)
model = Model(2).cuda()
path_to_model = '/content/drive/My Drive/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")

```

/content/drive/My Drive/Dataset/abebnhqyzv.mp4

0

20

40

60

80

100

0

20

40

60

80

100

38

Chapter 9

Future enhancement and conclusion

9.1 Conclusion

We successfully addressed the critical task of deepfake detection using computer vision techniques. Our project aimed to build a system capable of identifying deepfakes accurately.

Computer vision techniques (specifically OpenCV) were crucial for processing video frames. We extracted frames from videos, resized them, and normalized pixel values

For deepfake detection, features related to facial expressions, textures, and spatial patterns were essential. These features served as input to our PyTorch-trained model.

Detecting and aligning faces within frames is critical for accurate feature extraction. OpenCV's face detection capabilities helped identify regions of interest (faces) in videos.

We trained the model using labeled video frames (both deepfake and pristine). The model learned to differentiate between real and manipulated content.

We evaluated the model's performance on validation and test sets. Metrics such as accuracy, precision, recall, and F1-score provided insights into its effectiveness.

9.2 Future Scope

1. Advanced Deep Learning Techniques:

- Capsule Networks: Investigate capsule networks, which aim to address limitations of traditional CNNs by capturing hierarchical relationships among features.
- Deep Reinforcement Learning: Explore reinforcement learning approaches for improving model performance and adaptability.
- Multi-Modal Learning: Combine computer vision with natural language processing (NLP) for more robust and context-aware deep learning models.

2. Adversarial Robustness:

- Deepfakes are susceptible to adversarial attacks. Research and implement techniques to make the model more robust against such attacks.

3. Real-Time Processing:

- Optimize the system for real-time video processing. Consider GPU acceleration and efficient model architectures.

4. Ethical Considerations:

- Continue addressing ethical concerns related to deepfakes, privacy, and consent.
- Ensure fairness and transparency in the model's predictions.

Chapter 10

References

- [1] Andreas Rossler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies, Matthias Nießner, “FaceForensics++: Learning to Detect Manipulated Facial Images” in arXiv:1901.08971.
- [2] Deepfake detection challenge dataset : <https://www.kaggle.com/c/deepfake-detectionchallenge/data> Accessed on 26 March, 2020
- [3] Yuezun Li , Xin Yang , Pu Sun , Honggang Qi and Siwei Lyu “Celeb-DF: A Large-scale Challenging Dataset for DeepFake Forensics” in arXiv:1909.12962
- [4] Deepfake Video of Mark Zuckerberg Goes Viral on Eve of House A.I. Hearing : <https://fortune.com/2019/06/12/deepfake-mark-zuckerberg/> Accessed on 26 March, 2020
- [5] 10 deepfake examples that terrified and amused the internet : <https://www.creativebloq.com/features/deepfake-examples> Accessed on 26 March, 2020
- [6] TensorFlow: <https://www.tensorflow.org/> (Accessed on 26 March, 2020)
- [7] Keras: <https://keras.io/> (Accessed on 26 March, 2020)
- [8] PyTorch : <https://pytorch.org/> (Accessed on 26 March, 2020)
- [9] G. Antipov, M. Baccouche, and J.-L. Dugelay. Face aging with conditional generative adversarial networks. arXiv:1702.01983, Feb. 2017
- [10] J. Thies et al. Face2Face: Real-time face capture and reenactment of rgb videos. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2387–2395, June 2016. Las Vegas, NV.
- [11] Face app: <https://www.faceapp.com/> (Accessed on 26 March, 2020) [12] Face Swap : <https://faceswaponline.com/> (Accessed on 26 March, 2020)
- [13] Deepfakes, Revenge Porn, And The Impact On Women : <https://www.forbes.com/sites/chenxiwang/2019/11/01/deepfakes-revenge-porn-andthe-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-ofthe-deepfake-and-the-threat-to-democracy>(Accessed on 26 March, 2020)
- [15] Yuezun Li, Siwei Lyu, “ExposingDF Videos By Detecting Face Warping Artifacts,” in arXiv:1811.00656v3.
- [16] Yuezun Li, Ming-Ching Chang and Siwei Lyu “Exposing AI Created Fake Videos by Detecting Eye Blinking” in arXiv:1806.02877v2.

- [17] Huy H. Nguyen , Junichi Yamagishi, and Isao Echizen “ Using capsule networks to detect forged images and videos ” in arXiv:1810.11215.
- [18] D. Güera and E. J. Delp, "Deepfake Video Detection Using Recurrent Neural Networks," 2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), Auckland, New Zealand, 2018, pp. 1-6.
- [19] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1–8, June 2008. Anchorage, AK
- [20] Umur Aybars Ciftci, İlke Demir, Lijun Yin “Detection of Synthetic Portrait Videos using Biological Signals” in arXiv:1901.02212v2
- [21] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. arXiv:1412.6980, Dec. 2014.
- [22] ResNext Model : https://pytorch.org/hub/pytorch_vision_resnext/ accessed on 06 April 2020
- [23] <https://www.geeksforgeeks.org/software-engineering-cocomo-model/> Accessed on 15 April 2020
- [24] Deepfake Video Detection using Neural Networks
<http://www.ijssrd.com/articles/IJSSRDV8I10860.pdf>
- [25] International Journal for Scientific Research and Development <http://ijssrd.com/>