TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING

Khwopa College Of Engineering
Libali, Bhaktapur
Department of Computer Engineering

A FINAL REPORT ON

DEEPFAKE VIDEO DETECTION SYSTEM

Submitted in partial fulfillment of the requirements for the degree

BACHELOR OF COMPUTER ENGINEERING

Submitted by

Jenish Twayana	KCE076BCT019
Pragya Shrestha	KCE076BCT024
Samshrita Ghimire	KCE076BCT039
Shristi Koju	KCE076BCT044

Under the Supervision of
Er.Mukesh Kumar Pokhrel
Lecturer
Department of Computer Engineering

Khwopa College of Engineering
Libali, Bhaktapur
2023-24

CERTIFICATE OF APPROVAL

This is to certify that this major project work entitled **"Deepfake Video Detection"** submitted by Jenish Twayana (KCE076BCT019), Pragya Shrestha (KCE076BCT024), Samshrita Ghimire (KCE076BCT039) and Shristi Koju (KCE076BCT044) has been examined and accepted as the partial fulfillment of the requirements for degree of Bachelor in Computer Engineering. The project was carried out under special supervision and within the time frame prescribed by the syllabus.

.....
Er. Bibha Sthapit
External Examiner
Assistant Professor
Department of Electronics and Computer Engineering,
IOE, Pulchowk

.....
Er. Mukesh Kumar Pokhrel
Project Supervisor
Lecturer
Department of Computer Engineering,
Khwopa College of Engineering

.....
Er. Dinesh Gothe
Head of Department,
Department of Computer Engineering,
Khwopa College of Engineering

Copyright

Acknowledgement

We extend our gratitude to the Department of Computer Engineering at Khwopa College of Engineering for affording us the opportunity to engage in this tangible project and for granting us access to all essential resources required for the fulfillment of our project.

We would like to extend our heartfelt appreciation to our Head of Department, Er. Dinesh Gothe, for his valuable advice, guidance, help and support for successful completion of our project.

We are greatly obliged to Er.Niranjan Bekoju offering training and imparting knowledge, enabling us to enhance our proficiency in LaTeX. Additionally, we are immensely thankful to Er. Dinesh Ghemosu for offering suggestions and providing motivation throughout this period.

Our special thanks are extended to Er. Mukesh Kumar Pokhrel for his invaluable assistance as our supervisor, providing us with various suggestions and insights that greatly contributed to the completion of our project. Gratitude to all our friends, well-wishers, and everyone who provided assistance throughout our project.

Jenish Twayana	KCE076BCT019
Pragya Shrestha	KCE076BCT024
Samshrita Ghimire	KCE076BCT039
Shristi Koju	KCE076BCT044

Abstract

The phrase "Seeing is believing" no longer holds true in the age of deepfake technology. With advancements, creating convincing deepfakes has become increasingly accessible, even via mobile apps. Detecting these manipulations has grown challenging, surpassing the capabilities of the human eye. Nonetheless, researchers are actively developing methods to recognize deepfakes, offering hope for mitigating their societal impact. Deepfake refers to manipulated digital media, such as images or videos, created using machine learning algorithms, particularly Generative Adversarial Networks (GANs). These algorithms are trained to swap elements of one video, such as a person's face, with those from another piece of content. Deepfake technology is used to stir up doubt about crucial matters, intentionally spread false information, and fuel unethical actions such as creating revenge porn. Henceforth, it is essential to identify such videos. This project aims to detect deepfakes using the ResNext Convolutional Neural Network (CNN) and Bidirectional Long Short Term Memory (BiLSTM) algorithms through a web application. Instead of writing the code from scratch, we have used the pre-trained ResNext CNN model to extract features at the frame level. The obtained features are then used to train an LSTM-based RNN to classify videos as real or manipulated, and the resulting confusion matrix provides us with the validation and testing accuracy. This system uses a diverse dataset comprising 14,823 videos from various sources, including Face-Forensic++, Deepfake Detection Challenge (DFDC), and Celeb-DF, after preprocessing to standardize each video to dimensions of 224 pixels, with a fixed duration of 5 seconds and a frame rate of 30 fps, resulting in a sequence of 150 frames. This approach contributes significantly to deepfake detection, establishing robust defenses against the malicious use of AI-generated content.

Keywords: *Deepfake, GAN, ResNeXt CNN, BiLSTM, Deep Learning*

Contents

List of Figures

List of Tables

List of Abbreviation

AI	Artificial Intelligence
CUDA	Compute Unified Device Architecture
CNN	Convolutional Neural Network
GAN	Generative Adversarial Network
GPU	Graphics Processing Unit
LSTM	Long Short-Term Memory
ML	Machine Learning
OpenCV	Open Source Computer Vision Library
ResNeXt	Residual Neural Network with NeXt
SDLC	Software Development Lifecycle
SQL	Structured Query Language
UI	User Interface
UML	Unified Modeling Language
VAEs	Variational Autoencoders

Chapter 1

Introduction

1.1 Background

With the technology becoming accessible to any user, lots of deepfake videos have been spread through social media. Deepfake refers to manipulated digital media such as images of videos where the image or video of a person is replaced with another person's likeness. In fact, deepfake is one of the increasingly serious issues in modern society. Deepfake has been frequently used to swipe faces of popular Hollywood celebrities over porn images. Videos deepfake was also used to produce misleading information and rumors for politicians [1].

In 2019, for instance, the U.S. intelligence community warned in its annual Worldwide Threat Assessment that "adversaries and strategic competitors probably will attempt to use deepfakes or similar machine-learning technologies to create convincing—but false—image, audio, and video files to augment influence campaigns directed against the United States and our allies and partners" [2].In line with this prediction, a deepfake surfaced on social media in March 2022 that showed Ukrainian president Volodymyr Zelenskyy telling his soldiers to surrender to Russian forces [3].The deepfake technique is menacing to world security when its creation methods can be used to generate leaders' videos with counterfeit speeches for falsification goals. Therefore, it can be abused to cause religious and political tensions between nations, deceive the public, affect the election results, and expose individuals, societies, and countries to danger [4].

Detecting political deepfakes using LSTM and ResNeXt models offers a promising solution to the growing challenge of identifying deceptivel content. As deepfake technology continues to evolve, it is imperative to develop advanced detection methods that keep pace with these advancements.The generation and proliferation of deepfake material will only get more competitive and by extension, more harmful in the future. In a world that is fast to adapt to technological advancements, but is slow to change socially and structurally, the effects of deepfake have the potential to be catastrophic. Even while we develop solutions to combat the rise and spread of deepfake, equally interested and competent parties are developing ways to generate more and more realistic, and sometimes even entirely convincing images of "fake humans" as generated by the StyleGAN [5] [6].

1.2 Motivation

In an era marked by the rampant spread of deepfake technology, the integrity of democratic processes and public discourse faces unprecedented threats. Manipulated digital media, particularly videos and images, have become pervasive tools

for spreading misinformation, manipulating public opinion, and eroding trust in institutions. Our project is motivated by the urgent need to develop advanced algorithms and methodologies capable of accurately detecting deepfakes, particularly focusing on political contexts. By leveraging cutting-edge techniques such as LSTM and ResNeXt models, we aim to establish a robust framework for distinguishing between authentic and manipulated media content. Our efforts extend across diverse domains, including news and journalism, the entertainment industry, and legal and forensic investigations, with the overarching goal of upholding truth, transparency, and trust in communication channels amidst the escalating threat posed by deepfake technology.

1.3 Problem Statement

The use of deepfake technology in images and videos poses a grave threat to the integrity of democratic processes, as it has the potential to spread misinformation, manipulate public opinion, and undermine trust in institutions. Deepfakes can be used to discredit opponents, manipulate people, and erode public trust in the authenticity of media content. The lack of effective tools and algorithms for detecting and preventing deepfakes exacerbates the challenge of safeguarding the credibility and transparency. It is imperative to address this problem by developing advanced algorithms, methodologies, and tools that can accurately and reliably detect deepfakes, mitigating their harmful impacts. The urgent need for robust deepfake detection mechanisms necessitates innovative research and technological solutions to effectively identify and counteract the proliferation of deepfakes, ensuring the preservation of truth, transparency, and trust in communication.

Existing deepfake detection methods often fall short in effectively identifying and distinguishing deepfakes from genuine video content, as creators continuously refine their techniques to produce increasingly convincing and undetectable manipulations. This escalating arms race between deepfake creation and detection exacerbates the urgency to develop advanced algorithms, methodologies, and tools capable of reliably detecting and mitigating the spread of political deepfakes.In summary, the pressing problem is the need for effective detection tools and strategies to identify and combat political deepfakes, preventing the dissemination of manipulated content and protecting the integrity of political communication and democratic processes.

1.4 Objective

The main aim of this project is:

- To detect whether the uploaded video is fake or real.

1.5 Scope and Applications

The major scope of this project involves developing algorithms and techniques to identify and distinguish manipulated media content from authentic sources, and

application of deepfake detection extend across various domains, including but not limited to:

- News and Journalism
- Entertainment Industry
- Legal and Forensic Investigations

Chapter 2

Literature Review

Each of these papers mentioned below contributes to the field of deepfake detection by proposing novel techniques or discussing potential approaches to identify manipulated content. By understanding the methods and concepts presented in these papers, we worked towards improving the detection and mitigation of deepfakes.

2.1 Exposing DeepFake Videos By Detecting Face Warping Artifacts

In this paper [7], synthesized videos are detected by exploiting the face warping artifacts resulted from the DeepFake production pipeline. The authors claims that the model can effectively identify DeepFake content without relying on specific examples of DeepFake-generated images for training. Four different CNN models-VGG16, ResNet50, ResNet101 and ResNet152 were trained and demonstrated AUC performance levels of 84.5%, 98.7%, 99.1%, 97.8% respectively.

2.2 Deepfake Detection with Deep Learning: CNN versus Transformers

This paper [8] presents single model deepfake detectors based on eight different deep learning architectures (four CNNs and four Transformers) and conducted same train-to-test and cross datasets evaluations.The author observes that CNNs models did better in same train-to-test dataset evaluations, and the Transformers models did better in cross datasets evaluations. The eight models achieved an average 88.46% accuracy and 96.66% AUC when tested on the Google DFD dataset.

2.3 A Hybrid CNN-LSTM model for Video Deepfake Detection

This project [9] has leveraged an optical flow based feature extraction approach to extract the temporal features, which are then fed to a hybrid model based on the combination of CNN and RNN architectures for classification.To reduce the computational constraints, the experiment was performed on a subset of frames as considering all the frames of the videos require higher computational power. However, from the experimentation it is observed that the model performed better with an increasing number of frames per video. This proposed method shows an

accuracy of 79.49% in Celeb-DF with a very reduced no. of sample size of ≤ 100 frames.

2.4 Generative Adversarial Nets

This seminal paper [10] introduces Generative Adversarial Networks (GANs), a fundamental technique in creating deep fake images. GANs consist of a generator network and a discriminator network, competing against each other in a game-theoretic framework. The generator aims to produce realistic images, while the discriminator aims to distinguish between real and fake images. GANs have since become a cornerstone of deep fake creation and detection research.

2.5 Exposing Deepfakes using Inconsistent Head Poses

In this paper [11], a method for detecting deepfake images by analyzing inconsistencies in head poses is introduced. By examining the variations in head orientation and movement, the authors identify unnatural patterns present in deepfake images. SVM classifier is trained with RBF kernels on the training data, with a grid search on the hyperparameters using 5 fold cross validation.

2.6 Hybrid Deepfake Detection Utilizing MLP and LSTM

In this paper [12], a new deepfake detection schema utilizing two deep learning algorithms: long short-term memory and multi-layer perceptron is proposed. The model is evaluated using a publicly available dataset named 140k Real and Fake Faces to detect images altered by a deepfake . using LSTM and MLP resulted in accuracies of 74.7% and 68% respectively. Regarding precision, both algorithms produced very similar results whereas recall was the lowest metric yielded by MLP at 61%, while it was the highest for LSTM at 81.4%.

2.7 A Review of Deep Learning-based Approaches for Deepfake Content Detection

This paper [13] presents a comprehensive review of recent studies for deepfake content detection using deep learning-based approaches. Authors aim to broaden the state-of-the-art research by systematically reviewing the different categories of fake content detection. Furthermore, it report the advantages and drawbacks of the examined works and future directions towards the issues and shortcomings still unsolved on deepfake detection.

2.8 DeepFakes and Beyond: A Survey of Face Manipulation and Fake Detection

This paper [14] y provides a thorough review of techniques for manipulating face images including DeepFake methods, and methods to detect such manipulations. In particular, four types of facial manipulation are reviewed: i) entire face synthesis, ii) identity swap (DeepFakes), iii) attribute manipulation, and iv) expression swap. It concludes current manipulations are usually based on GAN architectures such as StyleGAN, providing very realistic images.

2.9 In Ictu Oculi: Exposing AI Generated Fake Face Videos by Detecting Eye Blinking

This work [15] is based on detection of eye blinking in the videos, which is a physiological signal that is not well presented in the synthesized fake videos. Eye blinking is detected by quantifying the degree of openness of an eye in each frame in video using the Long-term Recurrent Convolutional Neural Networks (LRCN) model. This method is tested over benchmarks of eye-blinking detection datasets and also show promising performance on detecting videos generated with Deepfakes.

Table 2.1: Review Matrix with Research Paper and Summary of corresponding papers.

S.N	Reference	Summary
1	Exposing DeepFake Videos By Detecting Face Warping Artifacts [7]	The authors claims that the model can effectively identify DeepFake content without relying on specific examples of DeepFake-generated images for training. Four different CNN models- VGG16, ResNet50, ResNet101 and ResNet152 were trained and demonstrated AUC performance levels of 84.5%, 98.7%, 99.1%, 97.8% respectively.
2	Deepfake Detection with Deep Learning: CNN versus Transformers [8]	This paper presents single model deepfake detectors based on eight different deep learning architectures (four CNNs and four Transformers) and conducted same train-to-test and cross datasets evaluations.
3	A Hybrid CNN-LSTM model for Video Deepfake Detection [9]	This model has leveraged an optical flow based feature extraction approach to extract the temporal features, which are then fed to a hybrid model for classification showing an accuracy of 79.49%.
4	Generative adversarial nets [10]	This paper introduces Generative Adversarial Networks (GANs), a key technique used in creating deep fakes. The paper made the decision to select one evaluation measure over another depending on the application.

5	Exposing deep fakes using inconsistent head poses [11]	In this paper, a method for detecting deepfake images by analyzing inconsistencies in head poses is introduced. By examining the variations in head orientation and movement, model identifies unnatural patterns present in deepfake images.
6	Hybrid Deepfake Detection Utilizing MLP and LSTM [12]	In this paper, a new deepfake detection schema utilizing two deep learning algorithms: long short-term memory and multi-layer perceptron is proposed. The model is evaluated using a publicly available dataset named 140k Real and Fake Faces to detect images altered by a deepfake . using LSTM and MLP resulted in accuracies of 74.7% and 68% respectively.
7	A Review of Deep Learning-based Approaches for Deepfake Content Detection [13]	It presents a comprehensive review of recent studies for deepfake content detection using deep learning-based approaches with an aim to broaden the state-of-the-art research by systematically reviewing the different categories of fake content detection.
8	DeepFakes and Beyond: A Survey of Face Manipulation and Fake Detection [14]	This paper provides a thorough review of techniques for manipulating face images including DeepFake methods, and methods to detect such manipulations. In particular, four types of facial manipulation are reviewed: i) entire face synthesis, ii) identity swap (Deepfakes), iii) attribute manipulation, and iv) expression swap.
9	In Ictu Oculi: Exposing AI-generated fake face videos by detecting eye blinking [16]	This paper is based on detection of eye blinking in the videos, which is a physiological signal that is not well presented in the synthesized fake videos. Eye blinking is detected by quantifying the degree of openness of an eye in each frame in video using LCRN model.

Chapter 3

Requirement Analysis

3.1 SOFTWARE REQUIREMENTS

Software requirements for our system includes:

a. Python
b. Google Collaboratory
c. GitHub
d. StarUML
e. VS Code
f. OverLeaf
g. ClickUp
h. Slack
i. Microsoft Team
j. FastAPI
k. Google Calendar
l. Kaggle
m. Postman
n. Ngrok

3.2 HARDWARE REQUIREMENTS

Hardware requirements for our system includes:

a. Processor: To operate this system, an Intel-i5 or above processor is required. However, the operational feasibility in CPU is quite slow so, it is recommended to use CUDA enabled GPU.

b. Network Infrastructure: Deepfake detection systems may need to access and process large volumes of data from various sources, so a reliable network infrastructure with adequate bandwidth is essential.

c. Graphic Card: Google Colab provides the privilege to GPU (16GB NVIDIA Tesla T4 GPU) for training our model. In general, the requirement for processing power is CUDA enabled GPU with minimum of NVIDIA Graphics Card (4 GB,GTX 1650 Ti) and 8 GB RAM for smooth experience.

3.3 FUNCTIONAL REQUIREMENTS

3.3.1 Manipulated Videos Identification

Our system's core function is to accurately detect manipulated videos, particularly those containing propaganda or misinformation, with a focus on deepfake identification. This involves employing advanced algorithms, facial recognition, machine learning models, and frame-by-frame analysis to identify anomalies and patterns associated with deepfake generation. The goal is to provide a reliable solution for mitigating misinformation and propaganda in multimedia content.

3.3.2 Media Analysis

In the context of deepfake detection, "Media Analysis" refers to the system's capability to thoroughly examine video content and pinpoint any irregularities or inconsistencies that might indicate manipulation. This involves scrutinizing elements such as facial expressions, lip-syncing, and contextual details within the video frames to identify anomalies associated with deepfake creation. The goal is to provide a robust mechanism for recognizing manipulated media content.

3.3.3 Continuous Improvement

"Continuous Improvement" emphasizes the need for the detection system to evolve and enhance its capabilities over time. This involves accommodating user feedback, integrating new detection techniques, and leveraging advancements in deepfake detection research. The system should be adaptable, allowing for updates and improvements to stay ahead of emerging manipulation techniques, ensuring its effectiveness in identifying manipulated content remains high.

3.3.4 Reporting and Alerting

The "Reporting and Alerting" aspect focuses on the system's ability to generate detailed reports and alerts specifically for detected political deepfakes. These reports should offer comprehensive information about the analyzed media, detection outcomes, and relevant metadata. By providing transparent and informative reports, the system enables users to understand the nature of the identified deepfakes and take appropriate actions, contributing to the overall security and integrity of media content.

3.4 NON-FUNCTIONAL REQUIREMENTS

These requirements are not needed by the system but are essential for the better performance of the proposed system. The points below focus on the non-functional requirement of the system.

3.4.1 Reliability

The system should be reliable to use and have user friendly interfaces.

3.4.2 Maintainability

The system should be maintainable and it should be able to train on new input data and scalable to millions of data points.

3.4.3 Performance

The system should run smoothly and not consume too much processing power.

3.4.4 Robustness

The system should be able to handle unexpected errors in proper ways without crashing the system.

Chapter 4

FEASIBILITY STUDY

4.1 Economic Feasibility

The total expenditure of the project is just computational power. We used multiple sources for dataset collections, including diverse online platform such as Kaggle, new websites, etc. The data compilation includes information gathered from FaceForensic++, DFDC, and Celeb-DF. The computational power required can be fulfilled by using our own laptops alongside the Google Colab cloud computing service, so the project is economically feasible.

4.2 Technical Feasibility

The majority of the necessary software is readily accessible online. Any intensive processing and GPU demands were efficiently handled at no cost through the utilization of Google Colaboratory's cloud computing service. Basic processing tasks can be managed using commonly available devices.

4.3 Time Feasibility

The deadline for the project was met, as mentioned in the Gantt chart at **??**. Here is proper management of time as scheduled. The topmost priority was given to the optimum analysis of the algorithm to get higher accuracy and testing of models. Documentation remains as important as any other task, so throughout the whole project documenting each and every work was continued. Thus, the project was completed within the time frame, but lots of things can be further improved in future revised versions.

4.4 Operational Feasibility

The system is designed to be as easy to use as possible, with an interactive user interface (UI), making it feasible for users to become accustomed to it even without extensive training.

Chapter 5

System Design and Architecture

We had developed a system which receives a video as its input and processes it to investigate whether it is real or fake.

5.1 Usecase Diagram

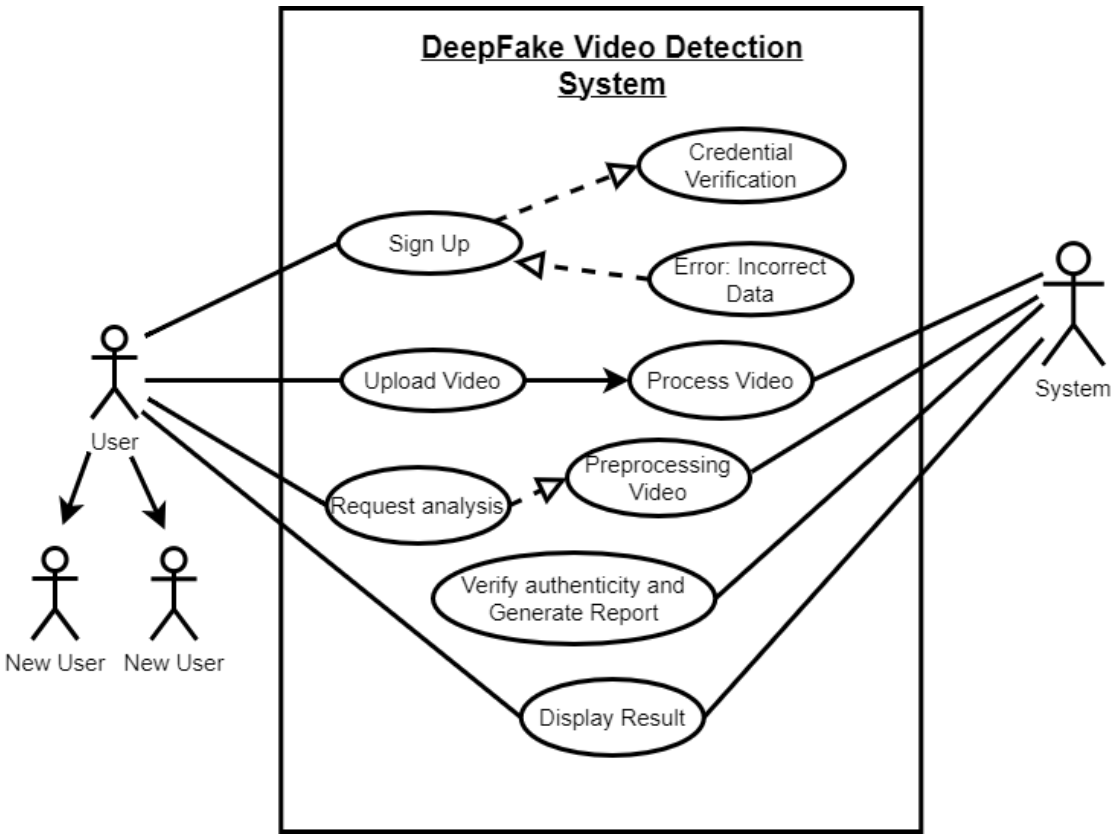


Figure 5.1: System Usecase Diagram

Actors:

- User: Represents an individual interacting with the system.
- New User: Refers to a user who is new to the system.
- Existing User: Represents a user who has previously used the system.

Usecases:

- Log in: The action of authenticating and gaining access to the system.

- Credential Verification: The process of checking user-provided credentials (such as username and password).
- Error Incorrect Credential: Indicates an error when incorrect credentials are provided during login.
- Upload Video: The action of submitting a video file for analysis. Check Input Format: Verifies that the uploaded video is in the correct format.
- Error Only Video Accepted: An error message displayed if a non-video file is uploaded.
- Request Analysis: Initiates the analysis process for the uploaded video.
- Display Result: Shows the analysis result to the user.
- Preprocessing: The initial steps taken to prepare the video for analysis.
- Detect Manipulation: Identifies any signs of video manipulation or deepfakes.
- Verify Authenticity: Determines whether the video is authentic or manipulated.
- Generate Report: Produces a report summarizing the analysis findings.

5.2 System Block Diagram

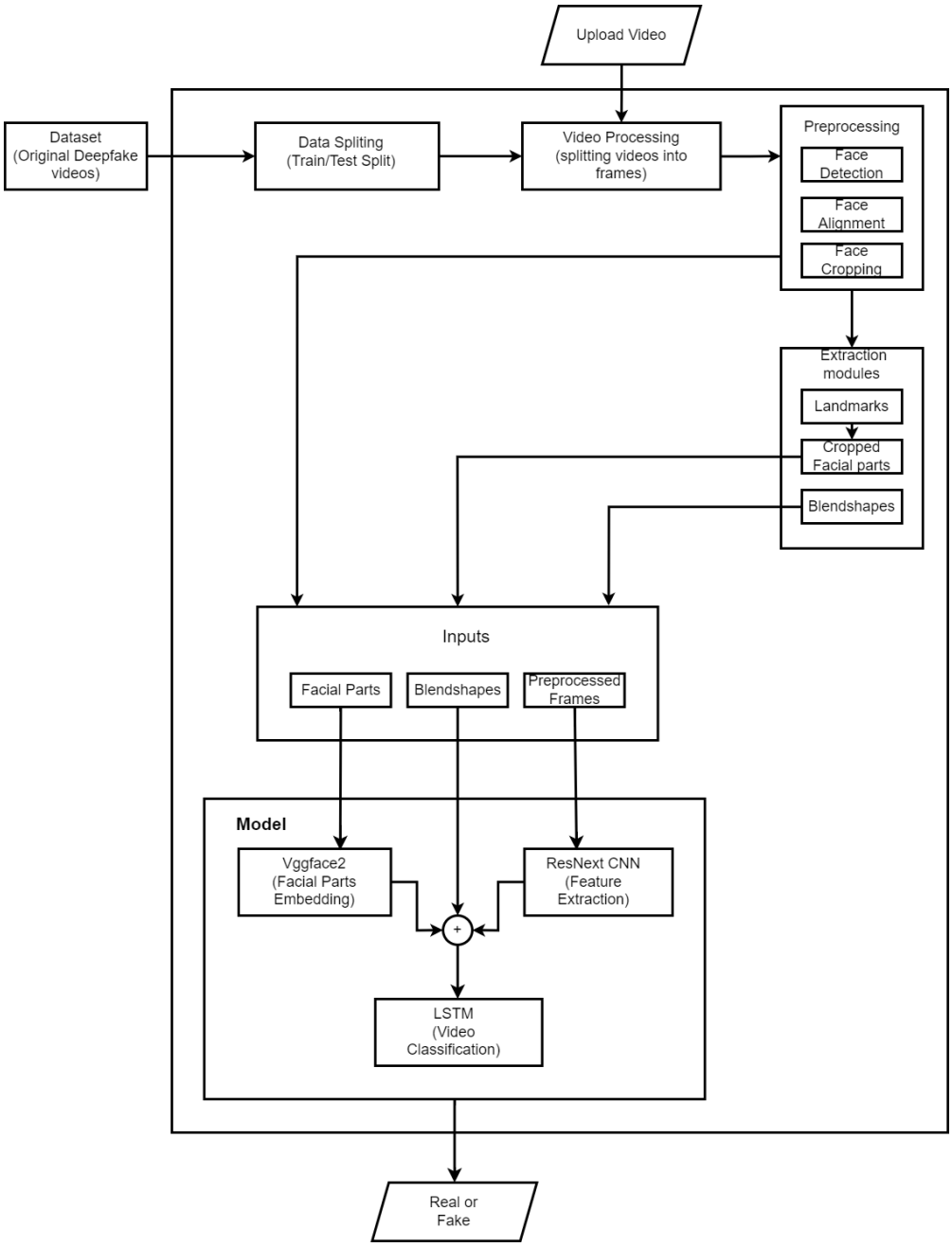


Figure 5.2: System Block Diagram

The flow of system block diagram is as below:

- When a user uploads video, it is splitted into frames and only initial 10 frames are selected for further analysis.
- Then, the selected frames are stored in an image array and sent for preprocessing.
- In preprocessing, for each frame, face is detected, aligned and cropped to make the input fit for classification model.

- The output of preprocessing is sent to the extraction module where the facial landmarks are extracted and cropped out. In this, the blendshapes helps to capture facial expressions and movements.
- Finally, the model receives three different inputs: facial parts, blendshapes and preprocessed frames.
- Vggface2 model receives input of facial parts wherein it embeds facial features.
- ResNext CNN model receives input of preprocessed frames and extracts relevant features from the frames.
- Finally, the LSTM model receives concatenated input of output of Vggface2, ResNext CNN and blendshapes and provides predictions as real or deepfake.

5.3 Workflow Diagram

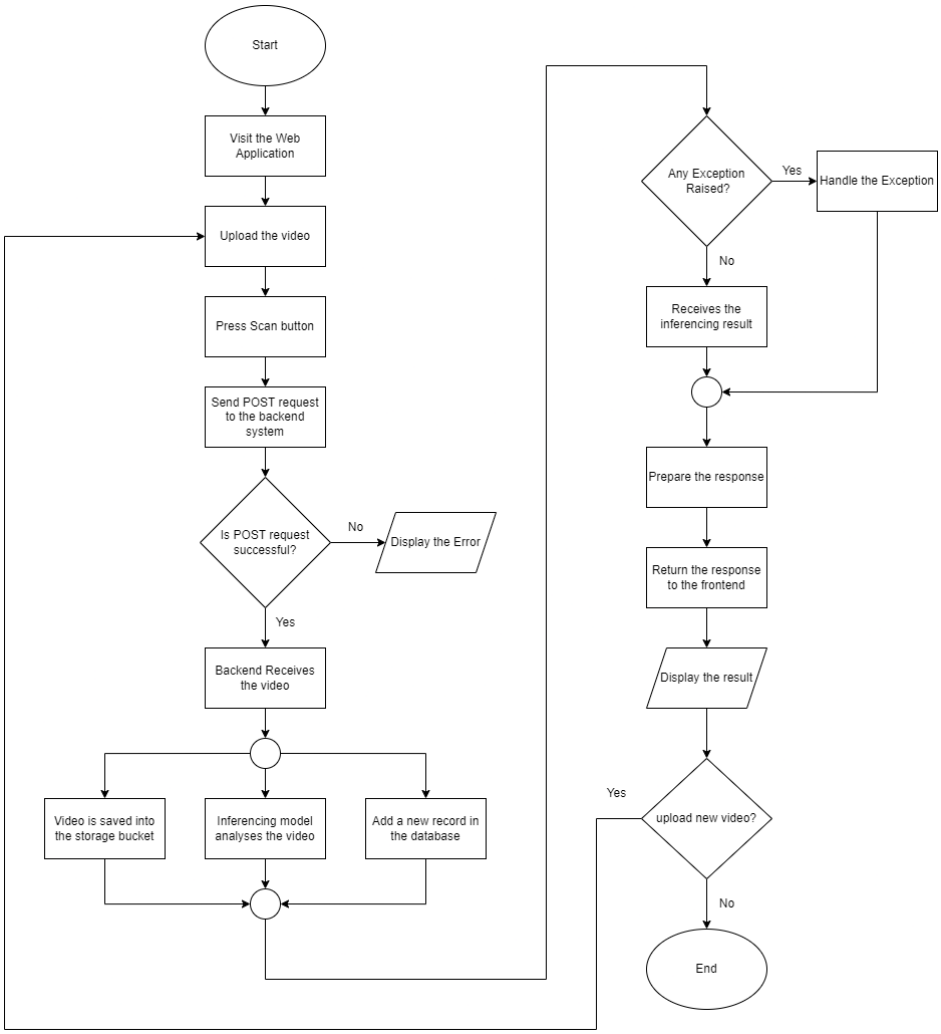


Figure 5.3: Workflow Diagram

The workflow of the system is as below:

- The user accesses the deepfake detection project through a web application.
- The user uploads the video into the input of webapp and presses scan button to request prediction.
- The POST request is sent to the server to update the uploaded video file.
- In case of unsupported video file, the POST request is unsucessful and error message is displayed to the user.
- If the POST request is successful, the backend system receives the video file for further processing.
- The backend system then stores the video in the storage bucket, sends the video to inferencing model for analysis and adds a new record in the database as metadata of the videofile.

- If the backend system doesnot encounter any exceptions, it receives an inferencing result as "FAKE" or "REAL" from the inferencing model.
- Then, the backend system returns the response to the frontend and the result is displayed to the user.
- The above steps are repeated for each video upload.

5.4 Sequence Diagram

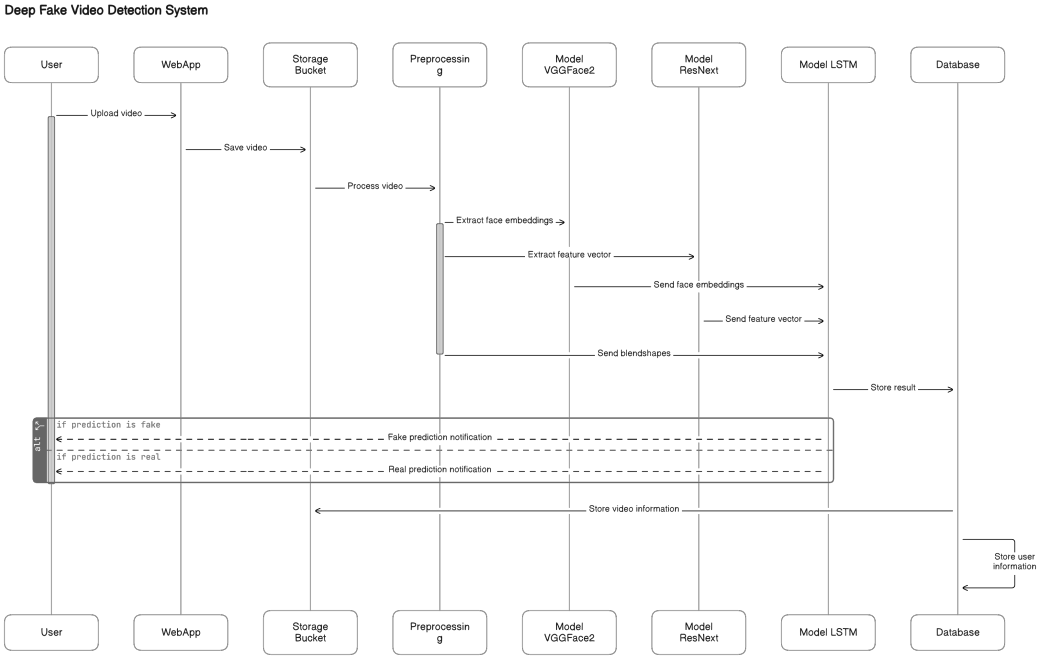


Figure 5.4: Sequence Diagram

The sequence diagram provides interactions among classes as follows:

- **User:** It initiates the process by sending a "LOAD URL" message to the WebApp.
- **WebApp:** It receives the "LOAD URL" message from the User and sends a "Load video by URL" message to the Storage component.
- **Storage:** It responds with a "Send binary data" message back to the WebApp.
- **WebApp:** After receiving the binary data, it sends it to the Preprocessor with an "Extract Metadata + Frames" message.
- **Preprocessor:** It processes the data and sends it back to the WebApp.
- **WebApp:** forwards the processed data to three different models: Model VGGFace2, Model FaceNet and Model LSTM.
- Each model processes the data and sends back their respective results as "Send extracted insights" messages to the WebApp.
- Finally, all these insights are stored in the Database after being sent from the WebApp with a "Store video information" message.

Chapter 6

Methodology

6.1 Software Methodology

The Agile methodology is a project management approach that involves breaking the project into phases and emphasizes continuous collaboration and improvement. Teams follow a cycle of planning, executing, and evaluating. By breaking down the project into manageable tasks and short iterations, progress is steady and responsive to feedback. Regular meetings with team members and supervisors foster collaboration and keep the team aligned. Agile's iterative nature allows for the incorporation of new findings and enhancements throughout the project life cycle. Inside Agile, the scrum framework has been used to manage all the projects through ClickUp. Ultimately, Agile methodology has enabled the delivery of a robust deepfake detection system that met the project's objectives while remaining flexible to changes and challenges.

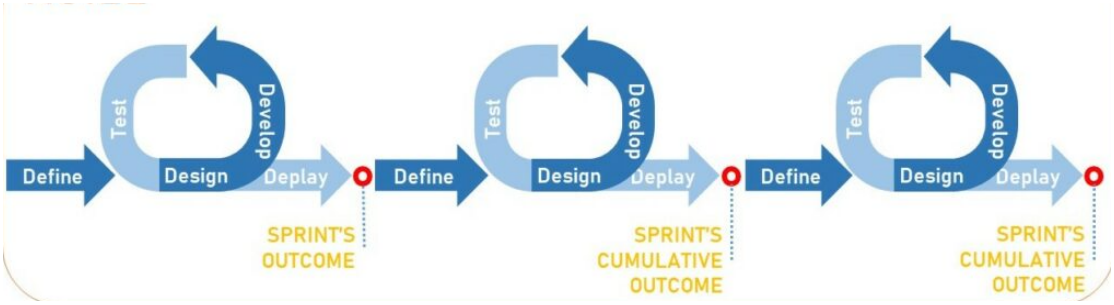


Figure 6.1: Agile methodology
source: https://www.wrike.com/agile-guide/

6.2 Scrum Framework

Scrum, an agile team collaboration framework, has been adopted in this software development, and it advocates breaking work into manageable goals to be completed within fixed time frames known as sprints. Typically lasting no more than a month and often around two weeks, each sprint is evaluated through brief stand-up meetings, called daily scrums, lasting up to 15 minutes. Upon sprint completion, the team conducts a sprint review to showcase the work to stakeholders (usually supervisor) and gather feedback, followed by an internal sprint retrospective for self-assessment and improvement.

It's essential to maintain a project backlog that encompasses all the tasks related to the entire project in one centralized location. Additionally, we also

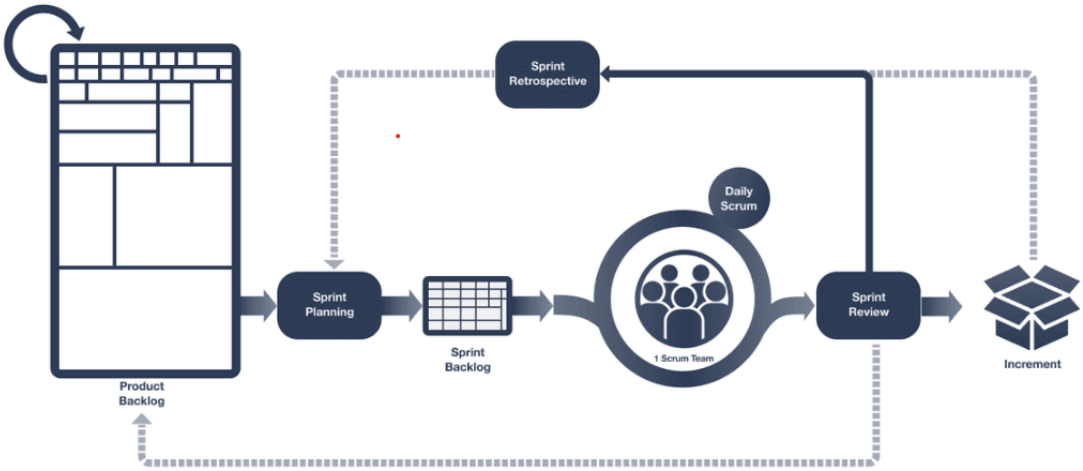


Figure 6.2: Scrum Framework
source: https://scrummate.com/agile-guide/scrum-framework/

maintain a sprint backlog that specifically includes backlogs relevant to the current sprint cycle, which is beneficial for maintaining focus on the immediate objectives.

6.2.1 ClickUp as Project Management Tool

We utilized ClickUP as our project management tool for organizing, distributing, managing, and tracking all work, as illustrated in Figure A.1. Tasks were generated as depicted in Figure A.2. Initially, we formulated a project plan and subsequently divided it into sprints, during which tasks from the project plan were executed.

6.2.2 Product Backlog

The product backlog is a prioritized list of work items that need to be accomplished to deliver a product in the Scrum framework. It serves as an integral planning tool within Scrum methodology, aiding the iterative and incremental software development activities. The product owner(in our context project supervisor or advisor) used to show up at the sprint planning meeting with the prioritized agile product backlog and describes the top items to the us. We then used to determine which items we can complete during the coming sprint. We teammates then used move items from the product backlog to the sprint backlog. In doing so, we expand each Scrum product backlog item into one or more sprint backlog tasks so they can more effectively share work during the sprint.Our product backlog includes following features:

1. Research and Analysis
2. Data Sourcing and Acquisition
3. Study and Research on Face Detection Model
4. Selection of Face Detection Model
5. Data Cleaning, Preprocessing and Augumentation

6. Experiment with different model configurations and hyperparameters
7. Model Selection and Development
8. Evaluation of BiLSTM model performance
9. Model Optimization and Enhancement
10. UI design
11. Frontend Development
12. Backend Development
13. Integration and Testing
14. System Evaluation and Visualization
15. Deployment in Web app

We have performed different tasks under below mentioned various sprints during project development:

6.2.2.1 Sprint 1-2: Research and Analysis

During these initial sprints, the focus was on laying the groundwork for the project. We delved into extensive research to understand the details of deepfake generation and existing detection techniques. Simultaneously, we explored various face detection models to identify the most suitable one for our project's objectives. These sprints were crucial for establishing a solid foundation upon which we built our deepfake detection system.

1. Conducted thorough research on deepfake generation techniques and existing detection methods.
2. Studied and analyzed various face detection models suitable for deepfake detection.
3. Selected the most appropriate face detection model based on research findings.

6.2.2.2 Sprint 3-4: Data Sourcing and Preprocessing

In these sprints, our primary goal was to gather the necessary data for training and testing our deepfake detection model. We sourced authentic and deepfake video datasets from reputable sources like Kaggle and ensured they are clean and properly formatted. Data preprocessing tasks such as resizing, cropping, rotating, standardizing different other data attributes for compatibility etc. were performed to prepare the datasets for model training.

1. Sourced and acquired deepfake and authentic video datasets from reliable sources.
2. Cleaned and pre-processed the collected datasets

6.2.2.3 Sprint 5-6: Model Development and Evaluation

In these sprints, our focus then shifted to model development consists of following tasks:

1. Explore various deep learning architectures suitable for deepfake detection (e.g., CNNs, GANs).
2. Experiment with different model configurations and hyperparameters.
3. Train initial deepfake detection models using labeled data.
4. Evaluate model performance using appropriate metrics (e.g., accuracy, precision, recall).

6.2.2.4 Sprint 7-8: Model Optimization and Enhancement

As we progress, our aim shifted to refine and enhance the BiLSTM model to ensure it achieves high accuracy in detecting deepfakes. Concurrently, we worked on developing the user interface design for our deepfake detection system, ensuring it is intuitive and user-friendly.It consists of following tasks:

1. Fine-tune model parameters to improve detection accuracy and reduce false positives.
2. Avoid overfitting of model by tuning of hyper-parameter carefully.

6.2.2.5 Sprint 9-10: Frontend and Backend Development

We focused on creating an engaging and responsive frontend interface for users to interact with, while also developing the necessary backend infrastructure to support the system's functionality.It consists of following tasks:

1. Implementing the frontend of the web application for the deepfake detection system.
2. Developing the backend, including server-side logic and database integration.

6.2.2.6 Sprint 11-12: Integration, Testing, and Evaluation

In the final stages of development, our priority was to integrate all components of the deepfake detection system seamlessly. We performed below mentioned activities:

1. Integrating the training models into a unified deepfake detection system.
2. Developing a user-friendly interface for interacting with the detection system.
3. Conducting thorough testing to validate system functionality and reliability.

6.2.2.7 Sprint 13-14: Deployment and Finalization

As we approach the end of the project, our focus had been on deploying the deepfake detection system in a production environment, making it accessible to users. We finalized all project documentation, ensuring it is comprehensive and well-documented. It consists of:

1. Deploying the deepfake detection system in a web application environment.
2. Finalizing project documentation, including system architecture, user manual, and technical specifications.
3. Addressing any remaining issues or feedback.

6.3 DATASET PREPARATION

6.3.1 Data Collection

To create a robust deepfake video detection system, we carried out an extensive data collection initiative that included a wide range of sources such as Kaggle, Youtube, and reputable news websites. Initially, we cast a wide net, gathering data from three primary sources: Face-Forensic++, Celeb-DF, and the Deepfake Detection Challenge (DFDC).

6.3.1.1 Data Sources

1. **Celeb-DF**: The Celeb-DF dataset contains 590 original videos and 5,639 synthesized videos, making it challenging for deepfake detection. The dataset includes subjects of various ages, ethnic groups, and genders, with the real videos chosen from publicly available YouTube videos featuring interviews of 59 celebrities. The dataset consists of over two million video frames, with an average video length of approximately 13 seconds and a standard frame rate of 30 frames per second. The real videos feature a diverse distribution in terms of gender, age, and ethnic groups.
2. **DFDC**:The DFDC (Deepfake Detection Challenge) dataset is a significant resource for deepfake detection, comprising over 100,000 videos sourced from 3,426 paid actors. This dataset was created to address the emerging threat of deepfakes and GAN-based face swapping methods by enabling the training of detection models and organizing the DFDC Kaggle competition.
3. **Face-Forensic++**: FaceForensics++ is a comprehensive dataset with 1000 manipulated video sequences using various methods like Deepfakes, Face2Face, FaceSwap, and NeuralTextures. It's designed for image and video classification, segmentation, and deep fake detection. The dataset includes 1000 deepfake models for data augmentation and a deepfake detection dataset from Google and JigSaw with over 3000 manipulated videos. It aims to standardize evaluation for facial manipulation detection with a hidden test set and a database of over 1.8 million manipulated images for research.

However, as we delved deeper into our project, we encountered challenges with the collected data. Concerns arose regarding its specificity and performance during our testing phases. Consequently, we made the strategic decision to streamline our dataset, focusing mostly on the DFDC dataset, which comprised approximately 42,912 videos and few dataset from Face-Forensic++ plus Celeb-DF.
Within the DFDC dataset, we found a distribution where roughly 14% of videos were authentic, while the remaining 86% were synthetic or "fake" videos generated using various techniques, including sophisticated manipulations like face swapping.
To ensure fairness and balance in our dataset, we undertook manual adjustments. As a result, we had around 11,796 videos, with an equal split between real and fake videos from DFDC only. We also collected 1164 videos (584 real and 580 fake) from Celeb-DF and 1863 videos from Face-Forensic++.

Table 6.1: Description of dataset

S.N.	Sources	Number of real videos	Number of fake videos	Total
1	Face-Forensic++	930	933	1863
2	Celeb-DF	584	580	1164
3	DFDC	5,898	5,898	11,796

Our approach to dataset refinement and balancing is pivotal to the development of a robust deepfake video detection system. This system represents a significant component of our college project, and we are optimistic that it will effectively discern and flag fake videos with a high degree of accuracy.

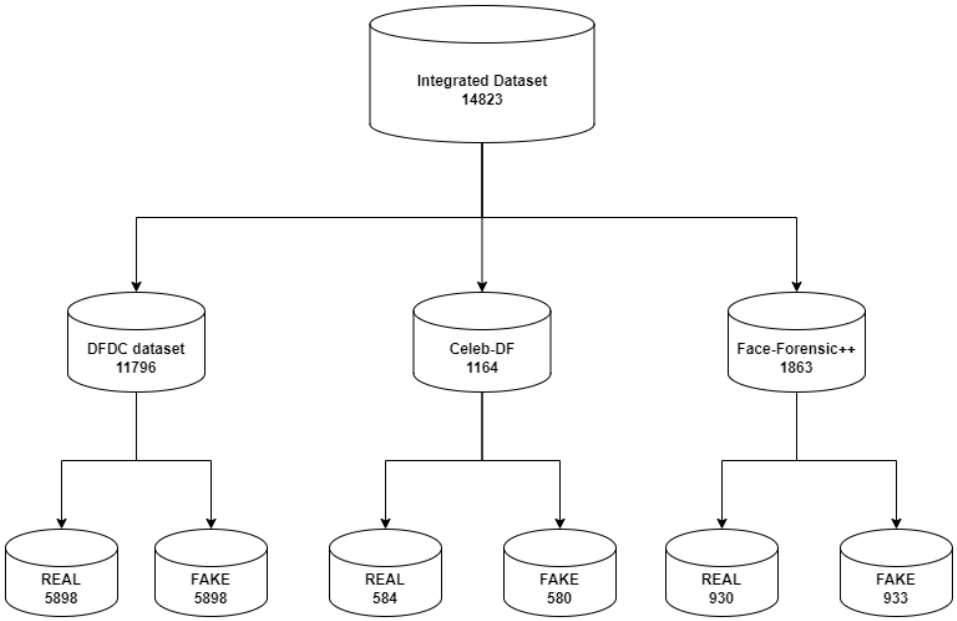


Figure 6.3: Datasets from different sources

6.3.2 Data Preprocessing

During the data preprocessing phase, we conducted a thorough examination of the aforementioned datasets to identify and address any anomalies. This meticulous review aimed to ensure the integrity and quality of the data used for subsequent analysis. Here are the key points of our data preprocessing procedures:

6.3.2.1 Abnormality Identification

As part of our comprehensive approach to deepfake detection, we carefully examined the datasets. This involved a thorough examination to identify any irregularities, inconsistencies, or outliers that could potentially impact the accuracy and reliability of our analysis. By carefully assessing the data, we aimed to gain a understanding of its distribution and structure, enabling us to anticipate and address potential challenges or biases that might arise during the detection process. Through this collaborative effort, we ensured that our deepfake detection models were built on a solid foundation of meticulously curated data.

6.3.2.2 Cleaning Procedures

To enhance the quality and integrity of the dataset, we implemented rigorous cleaning procedures. This involved the systematic removal of duplicate videos, irrelevant content, and any metadata that could introduce noise or bias into the data. By meticulously curating the dataset in this manner, we aimed to create a more refined and accurate representation of the underlying information. Our cleaning efforts were aimed at ensuring that the dataset accurately reflected the target domain, thus ensuring the reliability and effectiveness of our deepfake detection algorithms.

6.3.2.3 Standarization of Data Attributes

In order to facilitate seamless compatibility with deepfake detection algorithms, we undertook the standardization of various data attributes. This includes tasks such as standardizing file formats, resolutions, and other relevant attributes. By standardizing these attributes, we aimed for uniformity and consistency across the dataset. This standardized approach not only simplified the processing and interpretation of the data but also served to minimize potential sources of variability or error, thereby enhancing the robustness of our deepfake detection models.

6.3.2.4 Face Detection and Cropping

Models like MTCNN (Multi-task Cascaded Convolutional Networks) are used for accurate face detection.The output of MTCNN consists of the coordinates of the bounding box that tightly encloses the detected face. These coordinates include the (x, y) coordinates of the top-left corner of the bounding box and its width and height. Using these coordinates, we cropped the faces from the video frame and generated a new Face Cropped Video Dataset that specifically focused on facial imagery.

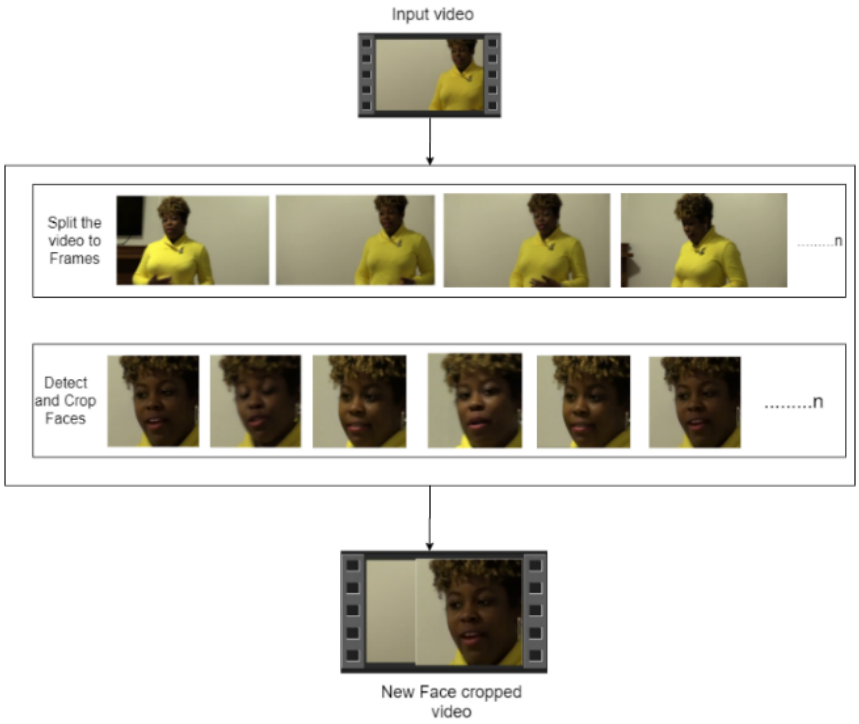


Figure 6.4: Preprocessing of Video with face cropping

6.3.2.5 Data Augmentation

Data augmentation is the process of artificially generating new data from existing data, primarily to train new machine learning (ML) models.It is most crucial part of data preprocessing which enhance the dataset to increasing its size and diversity. We performed it with a target of avoiding overfitting and improving and increasing

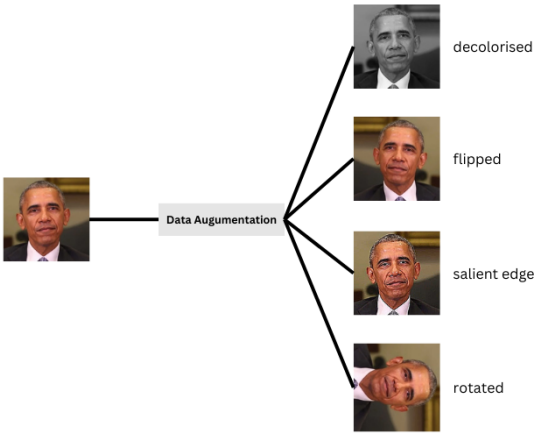


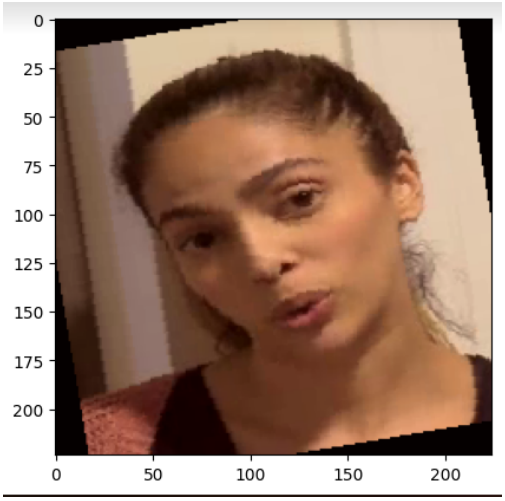
Figure 6.5: Some examples of Data Augumentation Techniques

model's ability to accurately detect unseen data. Some common data augmentation techniques include random cropping, rotation, flipping, scaling, and color jittering. Augmentation helps the model generalize better and improves its performance. We performed data augmentation to prevent overfitting and improves the model's ability to generalize to unseen data.

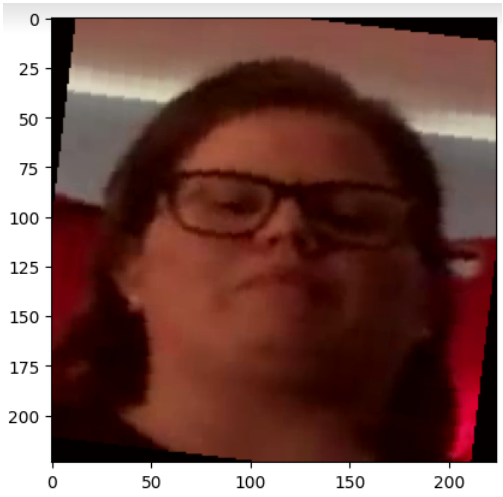
Common data augmentation techniques that we followed are:

1. **Rotation**:

   - It involves rotating the image by a certain angle, such as 90 or 15 degrees anticlockwise and clockwise.
   - It ensures that the detection model can detect deepfakes even if the orientation of the manipulated face or object differs from the original orientation in the training data.
   - We rotated the image by a certain angle such as 15° anticlockwise and clockwise which helped the model become invariant to orientation and improves its ability to recognize objects from different viewpoints.



(a) Anticlockwise Rotation by 15°

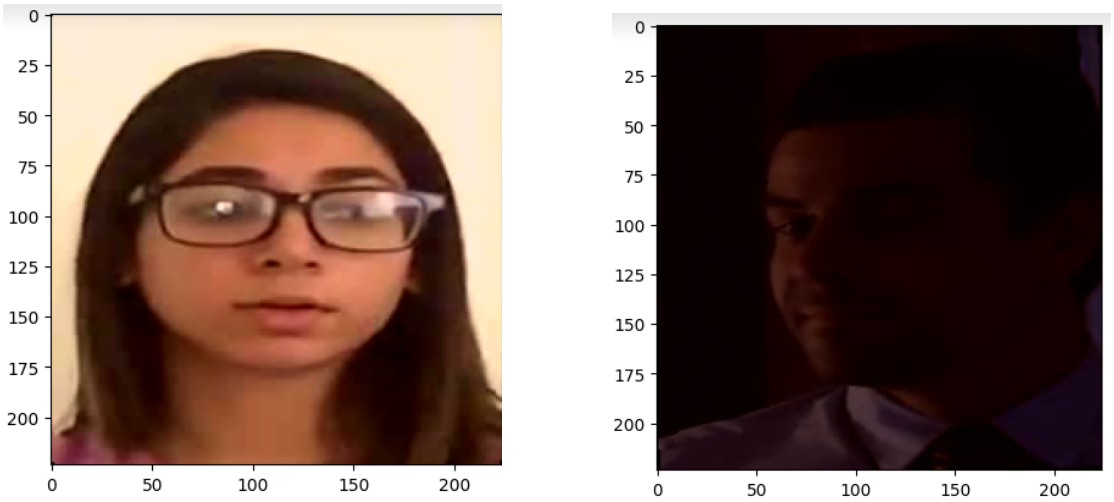


(b) Clockwise Rotation by 15°

Figure 6.6: Rotation as data augmentation technique

2. **Color Jittering**:

   - Deepfake videos may undergo color alterations during the manipulation process. Introducing random variations in the color properties of the images through techniques like changing brightness, contrast, saturation, or hue helps the model become more robust to such alterations.
   - By training the detection model with augmented data containing diverse color variations, it learns to detect deepfakes irrespective of changes in lighting conditions, color distributions, or other color-based manipulations.
   - It ensures that the detection model can effectively distinguish between genuine and manipulated content even when the manipulated regions exhibit variations in color or lighting that differ from the original content.
   - We changed brightness by various value such as 15%,18% etc.This helps the model become more robust to changes in lighting conditions and improves its ability to generalize across different environments.



(a) Increased brightness by 15%

(b) Brightness in negative value

Figure 6.7: Color Jittering as data augmentation technique

Thus, our data preprocessing efforts were driven by a commitment to quality, accuracy, and efficiency. By systematically identifying and rectifying abnormalities, standardizing data attributes, and prioritizing practicality in preprocessing techniques, we aimed to lay a solid foundation for subsequent deepfake detection and analysis tasks.
he preprocessing involves standardizing each video to dimensions of 224 pixels, with a fixed duration of 5 seconds and a frame rate of 30 frames per second (fps), resulting in a sequence of 150 frames. This consistent transformation ensures uniformity across the dataset, facilitating the subsequent stages of training and evaluation for the deepfake detection model.
After segregating a few corrupted videos and videos with less than 20 frames, we altogether had 14,712 videos with an equal split between real and fake videos.

6.3.3 Dataset Split

The dataset underwent a systematic division into training and test datasets, maintaining a specific ratio to optimize model training and evaluation. The split allocated 80% of the videos, totaling 11,768, to the training set and reserved the remaining 20%, or 2,944 videos, for the test set. This partitioning aims to strike a balance in the distribution of real and fake videos within each split.
Specifically, the balanced split entails ensuring that approximately 50% of the

S.N.	Splitting	Real videos	Fake videos	Total
1	Train Dataset	5,884	5,884	11,768
2	Test Dataset	1,472	1,472	2,944

Table 6.2: Description of dataset split

videos in both the training and test sets are authentic (real) videos, while the other 50% are synthetic (fake) videos.

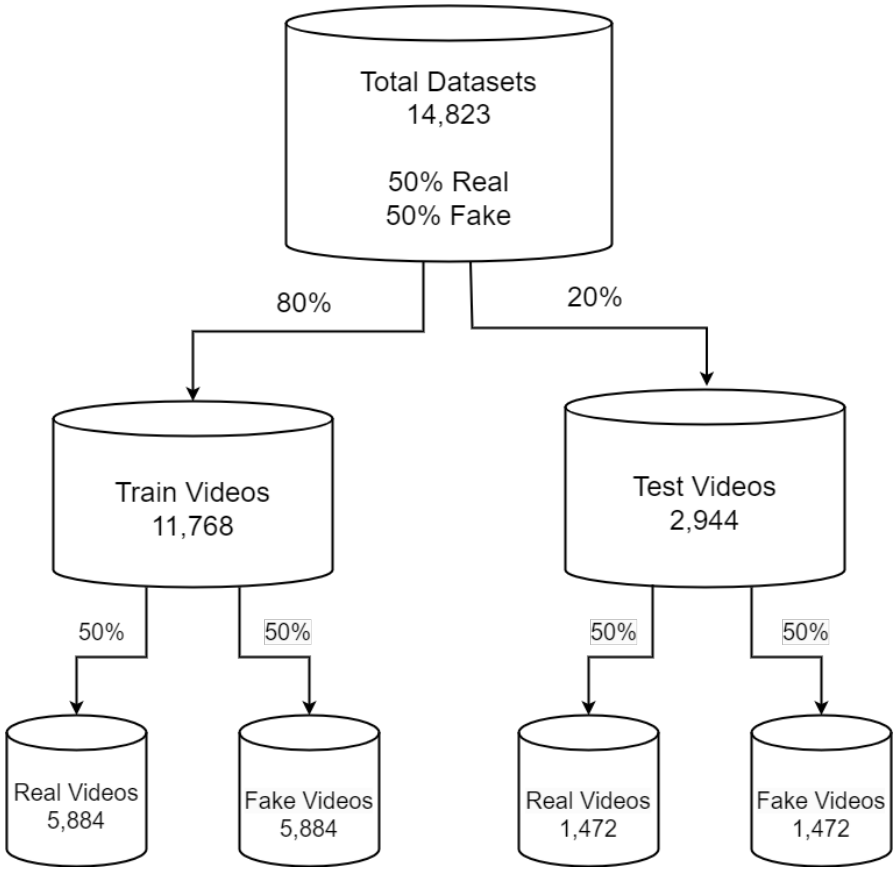


Figure 6.8: Splitting of datasets

6.4 Model Description and its training

The basic architecture to produce deepfake is the encoder-decoder architecture, where the encoder acquires the features of the target and the source face and the task of the decoder is to get encoding features of the target face and then generate fake video [10]. Only a small portion of the video is manipulated which means the deepfakes are shorter in time, therefore, the video is split into small frames and these frames are given as an input to detection model [17].
Analyzing temporal patterns in sequences of frames of such videos,and capturing subtle changes over time can indicate the presence of deepfake manipulation. The proposed model comprises of ResNeXt for feature extraction. These extracted features along with blendshapes and facial embeddings are provided as input to train a recurrent neural network i.e BiLSTM which is responsible for analyzing if the video has been put through manipulation or not, as it can effectively model temporal dependencies and patterns in sequential data, making it suitable for analyzing video frames. Moreover, it processes input sequences in both forward and backward directions, capturing information from past and future states.

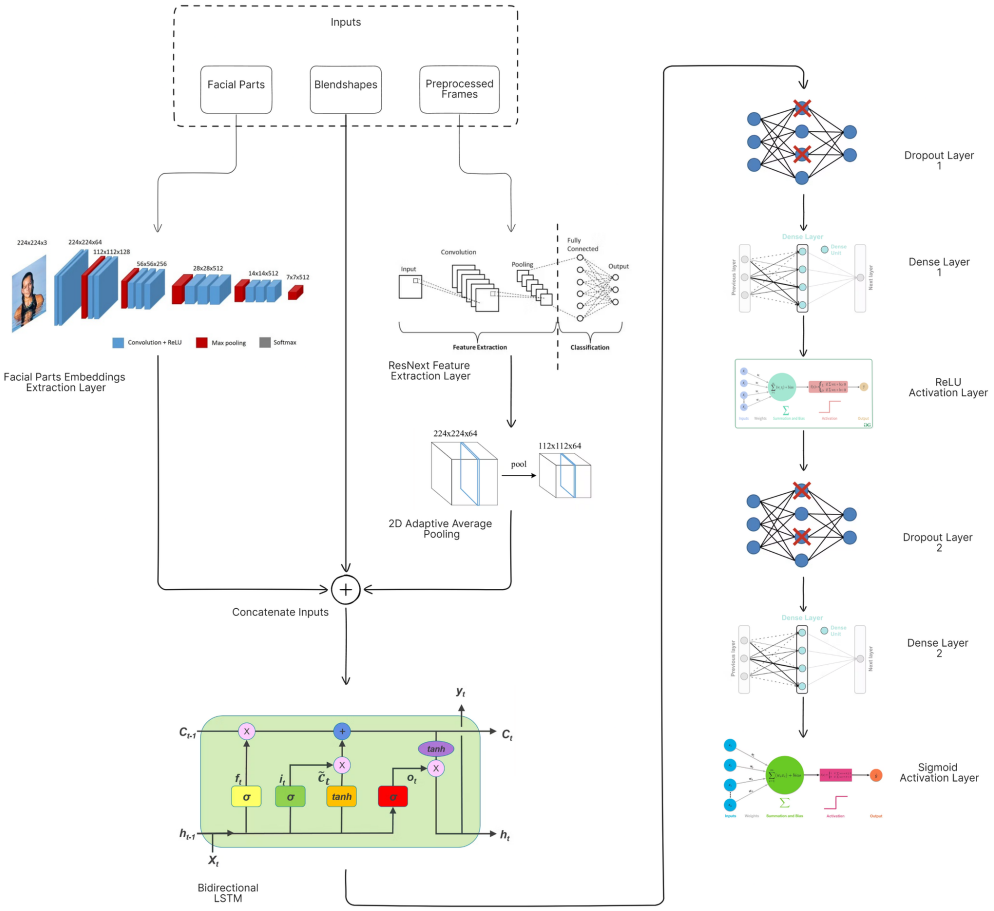
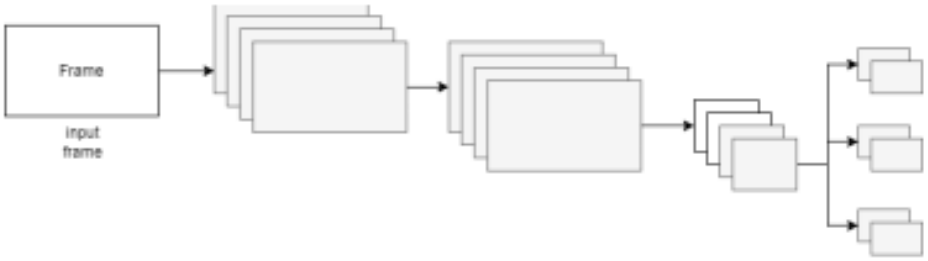


Figure 6.9: Illustrating the flow between different models and layers

6.4.1 ResNext Convolutional Neural Network (ResNext CNN)

Convolutional Neural Network (CNN) is a development of Multilayer Perceptron (MLP) designed to process two-dimensional data. CNN is included in the type of Deep Neural Network because of its high network depth and is widely applied to image data. In general, the convolutional layer detects edge features in the image, then the subsampling layer would reduce the dimensions of the features obtained from the convolutional layer, and finally passed them on to the output node through the forward propagation process [18].

ResNeXt inherits the fundamental idea of residual learning from ResNet. Residual learning involves the use of skip connections, or shortcuts, that allow the network to bypass one or more layers, facilitating the flow of gradients during training and alleviating the vanishing gradient problem. ResNeXt employs a bottleneck architecture in its residual blocks to reduce computational complexity.
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 experimental purpose we have used `ResNeXt50_32x4d` model. We have used a ResNext of 50 layers and 32 x 4 dimensions. Following, we have been fine-tuning the network by adding the extra required



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

Figure 6.10: Feature Extraction process using ResNext CNN
(Source: *https://paperswithcode.com/method/resnext)*

layers and selecting a proper learning rate to properly converge the gradient descent of the model. After passing each frame through ResNeXt, we had a set of feature maps that capture discriminative spatial information. These feature maps typically have variable spatial dimensions depending on the stride and pooling operations used within the ResNeXt architecture.

6.4.1.1 2D Adaptive Average Pooling

2D Adaptive Average Pooling to the output of ResNeXt before passing it to the BiLSTM layer is applied. It reduces the spatial dimensions of the feature maps to a fixed size, regardless of the input frame's dimensions. Adaptive pooling ensures that the model can handle inputs of varying spatial resolutions effectively. This allows the subsequent BiLSTM layer to focus on learning temporal patterns across frames, leading to improved deepfake detection performance.

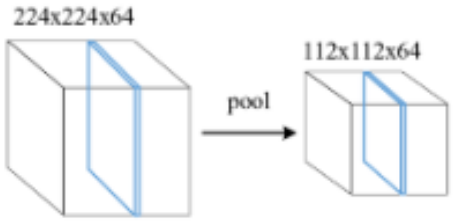


Figure 6.11: 2DAdaptive Average Pooling

6.4.2 Mediapipe

MediaPipe, developed by Google, offers a powerful framework for face landmark detection, providing precise identification and location of key points on the face. This technique involves predicting 478 3D facial landmarks and 52 blend shape scores in real-time, contributing to various applications like facial recognition, emotion analysis, virtual makeup, and augmented reality effects. We have used Mediapipe for following purposes:

1. Facial Landmark Extraction:
   MediaPipe enables the extraction of 468 landmark points for custom facial areas like eyes, eyebrows, lips,nose, and the outer face area, offering advanced capabilities for precise landmark detection and analysis. This detailed landmark extraction contributes to improving the depth and accuracy of facial feature recognition.

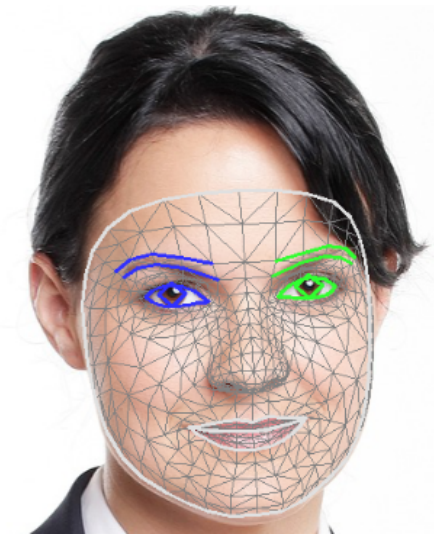


Figure 6.12: Facial Landmarks Detection

We applied the MediaPipe facial landmark detection model to each frame to obtain the coordinates of key facial landmarks.The output typically consist of a set of (x, y) coordinates for each detected landmark, representing points such as the corners of the eyes, the tip of the nose, and so on. We can convert the coordinates of facial landmarks into a suitable format for further processing. We used these landmarks coordinates to determine position of eyes, eyebrows, noses and lips and used them obtain two facial crops i.e eyes eyebrows and nose with lips, combined together in single canvas with

dimension of 124x124. So obtained output is then served as a valuable input for obtaining embeddings using models like VGGFace2.

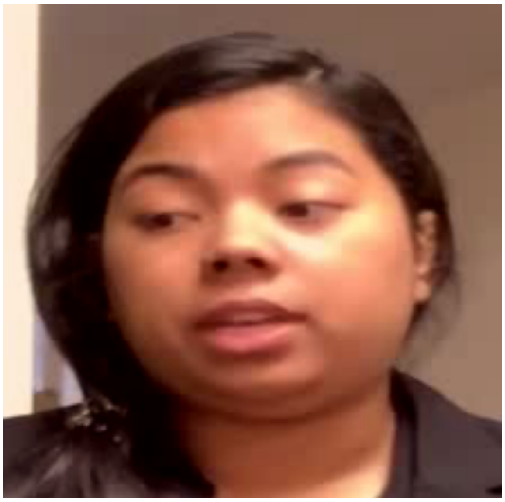
2. Blendshape Prediction Model:
   Blendshapes, also known as morph targets or shape keys, are a set of predefined facial expressions or deformations that can be applied to a 3D model of a face. A blendshape prediction model operates on the outputs from the face mesh model in MediaPipe, predicting 52 distinct blendshape scores that characterize different facial expressions. These blendshape scores are coefficients representing facial different expressions.Some of a few examples of the types of expressions that blendshape prediction models can capture: smile, frown, open mouth, closed eyes, raised eyebrows, squinting etc. This model enhances the understanding of emotions and movements in deepfake detection scenarios. We used these 52 distinct blendshape scores as input to BiLSTM Model after concatenating with other two results.

Figure 6.13: Mask obtained using Blendshape Coefficients

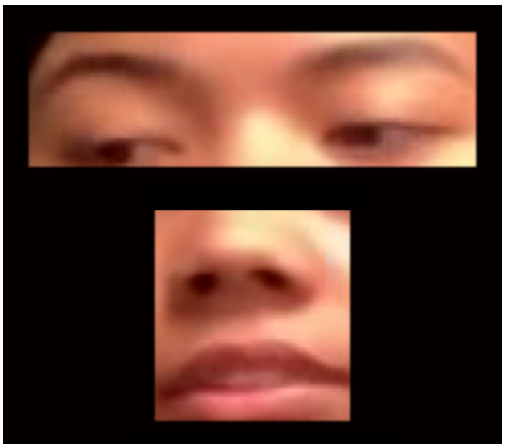
6.4.2.1 Facial Embeddings

Facial embeddings, in the context of our deepfake detection, refer to compact numerical representations of facial features extracted from video frames. These embeddings capture the unique characteristics of an individual's face in a high-dimensional vector space, enabling comparisons and analysis of facial similarities and differences. We extracted facial embeddings using InceptionResNetV1 pretrained on VGGFace2,along with facial landmarks extracted using MediaPipe. InceptionResNetV1 pretrained on VGGFace2 model is a powerful tool for facial embeddings that can be used for deepfake detection, offering advanced capabilities for face verification. Integrating the VGGFace2 model with Mediapipe's facial landmark extraction provides high-quality facial embeddings and spatial information about facial landmarks.

Facial landmarks extracted using MediaPipe for face in the video frame are used to crop facial parts of that frame i.e eyes with eyebrows and nose along with lips. These facial crops are included in same canvas and VGGFace2 uses it for extracting embeddings from this combined canvas providing output of 512 facial embeddings. These embeddings along with ResNext output and blendshape scores are combined together in appropriate format to serve as input to BiLSTM model.



(a) Original Image



(b) Combined canvas for two facial crops

Figure 6.14: Facial crops used for embeddings extraction

All these three different results: blendshapes, facial embeddings and output from ResNext passing through Adaptive pooling— are concatenated together as a single input to BiLSTM for temporal analysis and classification of videos.

6.4.3 Bidirectional Long Short-Time Memory(BiLSTM)

It is a bidirectional RNN that processes input sequences in both directions simultaneously. It consists of two LSTM layers: one processing the input sequence from past to future, and the other processing it from future to past. The outputs of both layers are concatenated at each time step. By doing so, BiLSTM effectively captures contextual information from both past and future states, enabling it to model complex temporal relationships with greater accuracy and depth.
In our system,the output of 2D adaptive average pooling, i.e., 2048-dimensional feature vectors, concatenated with 52 blendshape score and 512 facial embeddings resulting in a total of 2,612 feature vectors, is fed into the BiLSTM network to capture temporal dependencies across frames.BiLSTM processes the sequence of spatial features extracted by ResNeXt, learning temporal patterns indicative of deepfake manipulation. It utilizes the bidirectional nature of BiLSTM to capture both past and future context in the video sequence, resulting in the output of a 5224 units.
While experimenting in LSTM model, we obtained an output of 2612 units when provided with input from three different sources total of 2,612 dimensional feature vectors.
We applied dropout regularization to the output of the BiLSTM layer. It helps prevent overfitting by randomly setting a fraction of input units to zero during training.Then,the output of Dropout Layer 1 was passed through a fully connected Dense layer 1. This layer performs a linear transformation of the input features. ReLU activation fuction is used to introduce non-linearity to the model. Again, we applied another dropout layer after the Dense layer for additional regularization that passed its output through another fully connected Dense layer 2.Similar to Dense Layer 1, this layer performs a linear transformation of the input features,

potentially mapping them to an even lower-dimensional space. Finally, the output

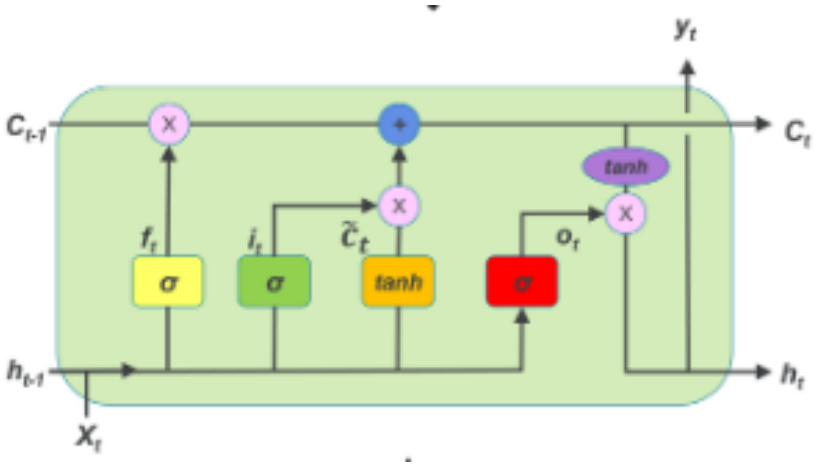


Figure 6.15: Internal structure of BiLSTM cell

of Dense Layer 2 is passed through a sigmoid activation function. Sigmoid squashes the output values to the range [0, 1], interpreting them as probabilities. This final output represents the probability that the input image is a deepfake. A value close to 1 indicates a high likelihood of the image being genuine, while a value close to 0 indicates a high likelihood of it being a deepfake.

6.4.3.1 Dropout Layer

Dropout is a regularization technique used to prevent overfitting in neural networks. During training, dropout layers randomly set a fraction of the input units to zero, effectively "dropping out" some neurons. This helps to prevent the model from relying too heavily on any particular set of features, forcing it to learn more robust representations. The dropout rate, which determines the probability of a neuron being dropped out, is typically set to a value between 0.2 and 0.5.

We have used two dropout layer in our model namely Dropout Layer 1 and

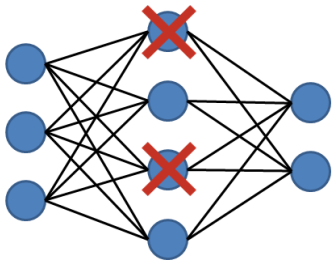


Figure 6.16: Dropout Layer
(Source: *https://towardsdatascience.com/coding-neural-network)*

Dropout Layer 2, each have dropout rate of 0.3 and 0.2 respectively. The input of 4096-dimensional feature vector is provided to Dropout Layer 1. The output of Dropout Layer 1 is passed through a fully connected Dense layer 1 that ensures linear transformation of the input features. And then Dropout Layer 2 is applied after Dense Layer 1 with input of 512 features for additional regularization. This

helps further prevent overfitting and improves the generalization performance of the model.

6.4.3.2 Dense Layer

A dense layer, also known as a fully connected layer, is a fundamental building block in neural network architectures. It's called "dense" because each neuron in the layer is connected to every neuron in the previous layer (or the input layer if it's the first layer in the network), creating a dense matrix of connections. Dropout serves as a form of regularization by adding noise to the network during training. Regularization techniques like dropout prevent the network from memorizing the training data and encourage it to learn more generalizable patterns. By placing a dense layer after dropout, the network can still adapt and learn from the noisy environment created by dropout while benefiting from regularization to prevent overfitting.

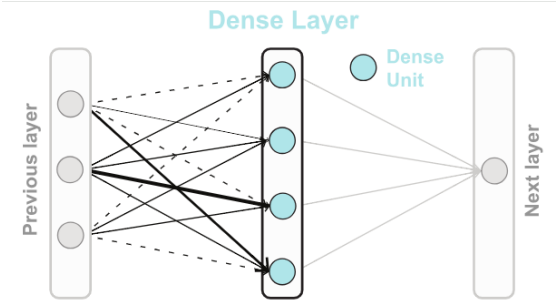


Figure 6.17: Dense Layer
(Source: *https://epynn.net/Dense.html)*

We have used two dense layer in our model namely Dense Layer 1 and Dense Layer 2. The result of Dropout Layer 1 acts as input to a fully connected Dense layer 1. Here, the dense layer acts as a feature extractor, transforming the high-dimensional output from the previous layers i.e 4096 features into a lower-dimensional space i.e 512 features. ReLU receives the output from dense layer 1 to introduces non-linearity to the model by outputting the input directly if it's positive, and zero otherwise. Another dropout layer is applied after Dense Layer 1 for additional regularization.The output from Dropout Layer 2 is passed through another dense layer. Similar to Dense Layer 1, this layer performs a linear transformation, potentially mapping the features to an even lower-dimensional space which is 2 output values. Finally, the output from Dense Layer 2 is passed through a sigmoid activation function.

6.4.4 Activation Function

An activation function is a mathematical function that is applied to the output of a neuron in a neural network. The activation function is used to introduce non-linearity into the output of the neuron, allowing the network to learn more complex and sophisticated patterns in the data.

6.4.4.1 Sigmoid Activation Function

The sigmoid activation function is a mathematical function commonly used in artificial neural networks. It maps any real-valued number to a value between 0 and 1, which makes it suitable for binary classification problems or as an output activation for probabilities.

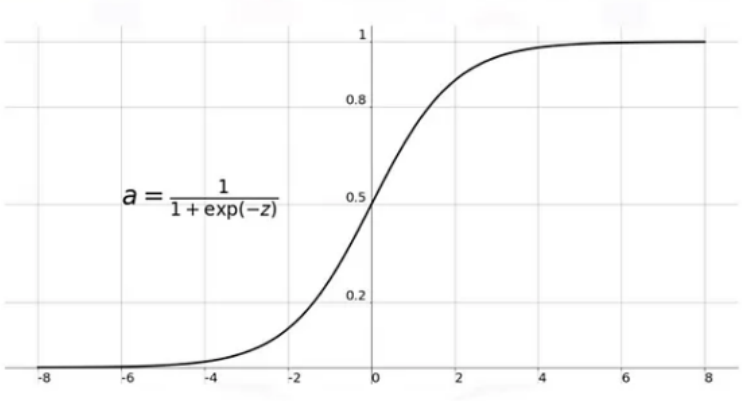


Figure 6.18: Graph respresenting Sigmoid Function
(Source:
https://medium.com/@toprak.mhmt/activation-functions-for-deep-learning)

The formula for the Sigmoid function is:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{6.1}$$

where,x is the input to the function whereas e is the base of the natural logarithm (approximately equal to 2.71828).
When the input to the sigmoid function is positive, the function returns a value closer to 1. When the input is negative, the function returns a value closer to 0. The sigmoid function has an S-shaped curve, which allows it to squash the input values into the range [0, 1]. In our system, its output represents the probability that the input image is a deepfake. A value close to 1 indicates a high likelihood of the image being genuine, while a value close to 0 indicates a high likelihood of it being a deepfake.
By combining these components in a sequential manner, the deepfake detection model transforms the input image through multiple layers of processing, gradually extracting and refining features to make a prediction about its authenticity. Adjustments to the architecture, hyperparameters, and training data can be made to optimize the performance of the model for the specific task at hand.

6.4.4.2 ReLU Activation Function

ReLU, which stands for Rectified Linear Unit, is a type of activation function used in artificial neural networks, particularly in deep learning models. It is one of the most commonly used activation functions due to its simplicity and effectiveness.

The ReLU function is defined as:

$$f(x) = max(0, x) \tag{6.2}$$

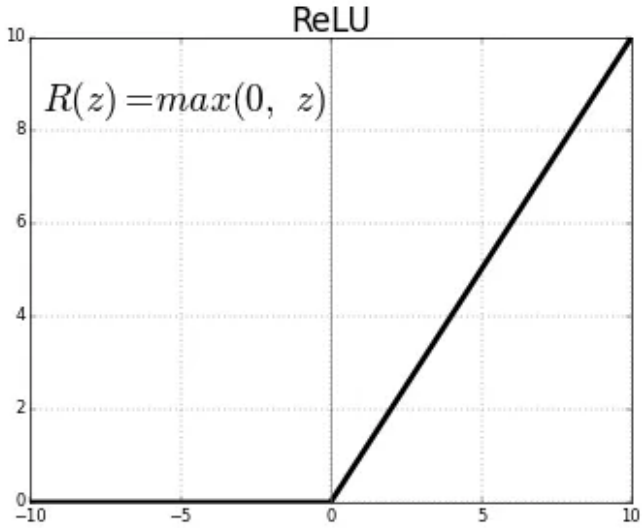


Figure 6.19: Graph respresenting ReLU Function
(Source: *hhttps://prateekvishnu.medium.com/activation-functions)*

In other words, it returns 0 if the input is less than 0, and returns the input value itself if it is greater than or equal to 0. Visually, this function looks like a piecewise linear function with a slope of 1 for positive values and a slope of 0 for negative values. ReLU function is non-linear, which means we can easily backpropagate the errors and have multiple layers of neurons being activated by the ReLU function. It does not activate all the neurons at the same time. Since the output of some neurons is zero, only a few neurons are activated making the network sparse, efficient and easy for computation.
The output from dense layer 1 is passed to ReLU activation fuction which is used to introduces non-linearity to the model by outputting the input directly if it's positive, and zero otherwise.

6.4.5 Loss Function

A loss function is typically defined as a mathematical expression that takes as input the model's predictions and the true labels (or target values) and outputs a single scalar value representing the "cost" or "loss" associated with the model's predictions. It quantifies how well the model's predictions match the actual observations during the training process. They provide a way to quantify the error or difference between predicted and actual outcomes, guiding the learning process to improve the model's performance. The choice of an appropriate loss function can significantly impact the training process and the final performance of the model. We used Binary Cross-Entropy Loss (BCE) for our model.

6.4.5.1 Cross-entropy loss function

Cross-entropy loss function measures the performance of a classification model whose output is a probability value between 0 and 1. Cross-entropy loss increases as the predicted probability diverges from the actual label. It plays a fundamental role in training models effectively by minimizing the dissimilarity between predicted and true values, ultimately enhancing model accuracy and performance in classification tasks.

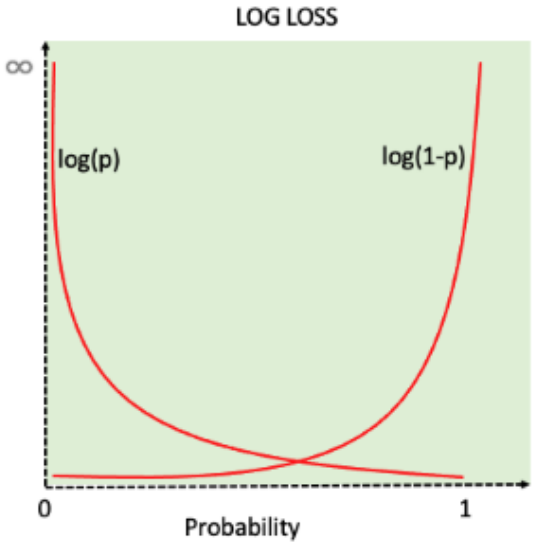


Figure 6.20: Binary Cross Entropy Loss for a single instance
(Source: *hhttps://www.geeksforgeeks.org/)*

In deepfake detection, cross-entropy loss is commonly used as the loss function when training models for binary classification tasks, where the goal is to distinguish between genuine and fake videos. The cross-entropy loss measures the dissimilarity between the predicted probability distribution and the true distribution of the target labels.Here's how cross-entropy loss is defined in the context of binary classification:

Let y be the true label (1 for genuine, 0 for fake), and p be the predicted probability of the positive class (i.e., the probability that the video is genuine).

$$L(y, p) = -\left(y \cdot \log(p) + (1 - y) \cdot \log(1 - p)\right) \tag{6.3}$$

where,y is the true label (0 for fake, 1 for genuine), p is the predicted probability of the positive class (i.e., the probability that the video is genuine) and log represents the natural logarithm.

6.4.6 Optimizers

An optimizer refers to an algorithm or method used to adjust the attributes of a model, typically its parameters, in order to minimize or maximize a certain objective function. The primary goal of an optimizer is to improve the performance of a machine learning model by iteratively updating its parameters based on the gradients of the loss function with respect to those parameters.
Optimizers work by iteratively updating the model parameters based on the gradients of the loss function with respect to those parameters. The gradient indicates the direction of steepest ascent, so optimizers typically move the parameters in the opposite direction of the gradient to decrease the loss.
We employed it during the training process to adjust the model's parameters in such a way that it learns to differentiate between authentic and manipulated content more effectively. The choice of optimizer can significantly impact the performance of the trained model. Selecting the appropriate optimizer and fine-tuning its hyperparameters is essential for achieving the best possible model performance.

6.4.6.1 Adam

The Adam optimizer, known as Adaptive Moment Estimation, is a highly effective optimization algorithm extensively utilized in machine learning for training deep learning models. It distinguishes itself through its adaptive learning rates, unlike conventional gradient descent methods where a uniform learning rate is applied to all parameters. This adaptability results in more efficient optimization, particularly in intricate models with numerous parameters.
At the core of Adam lies the concept of dynamically adjusting the learning rate during training by leveraging the first and second moments of the gradient. The first moment represents the average of the gradients, while the second moment denotes the average of the squared gradients. By monitoring these moments, Adam can estimate the variance and bias of the gradients, thereby enhancing the optimization process.
Essentially, Adam maintains a specific learning rate for each parameter and modifies it adaptively based on the estimated first and second moments of the gradients. Consequently, Adam possesses the capability to autonomously adjust the learning rate throughout the training phase, facilitating faster and more reliable convergence towards the minimum of the loss function.

6.5 Workflow Description

The project basically is composed of following intelligent models:

a. Face Detection Model
b. Landmarks and Blendshapes Extraction
c. Face Embeddings Extraction
d. ResNext CNN
e. Bidirectional LSTM

6.5.1 Training Workflow Description

a. Face Detection and Cropping:

- Utilized a pre-trained face detection model MTCNN to detect faces within the input images or videos.
- Cropped the detected faces to extract regions of interest (ROI) containing the face for further processing. This step ensures that only facial regions are considered for subsequent analysis.

b. Landmark Detection and Blendshapes:

- Employed a landmark detection model (such as MediaPipe FaceMesh) to accurately identify key facial landmarks (e.g., eyes, nose, mouth corners) within the cropped face regions.
- Obtained 52 blendshapes scores too as output from MediaPipe.

c. Facial Parts Cropped:

- After landmark detection, we cropped and extracted specific facial parts based on the detected landmarks. For example, crop the regions around the eyes and eyebrows and the nose and lips.
- Combine the cropped facial parts (such as eyes, nose, mouth) into a unified input representation for the VGGFace2 model.

d. ResNeXt as Feature Extraction CNN layer:

- Utilized a ResNeXt architecture (a variant of ResNet with improved performance) for feature extraction from video frames.
- Reduced the spatial dimensions of the feature maps to a fixed size using 2D Adaptive Average Pooling.

e. Input Preparation:

- Extracted facial embeddings using InceptionResNetV1 pre- trained on VGGFace2.
- Concatenated 512 facial embeddings, 52 blendshape scores and 2048-dimensional feature vectors as output from ResNext,feature extraction model.

f. Model Training, Evaluation and Hyper Parameters Tuning

- Incorporated a Bidirectional LSTM (Long Short-Term Memory) network as part of the detection pipeline, particularly for analyzing temporal patterns in video sequences.
- Trained the BiLSTM model using sequences of features extracted from consecutive frames of videos.
- Evaluated the trained deepfake detection model using appropriate metrics such as accuracy, precision, recall, and F1-score.

6.5.2 Inferencing Workflow Description

a. Model Loading:

- Begin by loading the pre-trained deepfake detection model.

b. Input Video Preprocessing:

- Face Detection, Alignment, and Cropping (RetinaFace): Utilize RetinaFace for accurate face detection, alignment, and cropping within the input video.

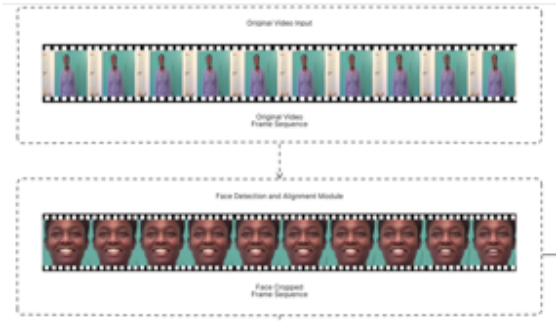


Figure 6.21: Input Video Preprocessing using RetinaFace

- Landmarks and Blendshapes Extraction (MediaPipe): Employ MediaPipe to extract facial landmarks and blendshapes from the cropped faces.

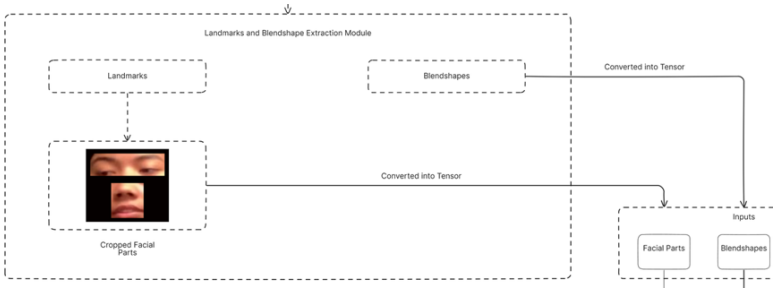


Figure 6.22: Landmarks and Blendshapes Extraction

- Facial Parts Cropping: Crop specific facial parts for detailed analysis.

6.5.2.1 RetinaFace

RetinaFace is a deep learning-based face detection model that has gained popularity for its high accuracy and efficiency. Developed by researchers at the Wuhan University and InsightFace, RetinaFace is specifically designed for face detection tasks in images and videos.
We have used it when processing an input video,for face detection and alignment, as RetinaFace excels at detecting faces within images or

video frames.This involves predicting bounding boxes around detected faces, along with confidence scores indicating the likelihood of a face being present. After detecting faces in the video frames, alignment becomes crucial for ensuring consistent and accurate analysis. Face alignment involves adjusting the detected faces to a canonical pose, such as frontal orientation, to facilitate further processing, such as feature extraction or comparison.

c. Input Data Transmission:

- Send the preprocessed data to Resnext for feature extraction
- Concatenate 2048-dimensional feature vectors with 52 blendshape scores and 512 facial embeddings obtained after using VGGFace2 provided with facial crops

d. Output Display:

- Utilize BiLSTM architectures within the deepfake detection model to process the input data and generate predictions regarding the video's authenticity.
- Display the output, indicating whether the input video is classified as real or a deepfake.

6.6 System Deployment

6.6.1 Web Application

Our web application for deepfake video detection seamlessly integrates frontend and backend technologies along with database and storage services to provide users with a robust and user-friendly experience.

6.6.1.1 Frontend Development

The frontend interface is developed using ReactJS framework technologies to create an intuitive and interactive platform for the users..

- **Drag and Drop Functionality:** Users can easily upload video files by dragging and dropping them onto the designated area.
- **Upload Button:** Alternatively, users can select video files from their device using the upload button.
- **File Validation:** Uploaded files undergo validation to ensure they are video files. If a non-video file is uploaded, an error message is displayed.
- **Scan and Reset Button:** Users can click the scan button to initiate the inferencing of the uploaded video. Reset button can be pressed to cancel the inferencing.

- **Sign In Functionality:** Users can sign in using the Google Account. Upon signing in, the uploaded videos are retained as the history and can be accessed later. It can be deleted too.
- **Displaying Result:** Once the backend completes its analysis, the result - whether the video is detected as fake or real - is displayed with additional details for more information.

6.6.1.2 Backend System

The backend functionality is powered by multiple technologies working together to provide the necessary features and functionalities.

- **API service:** FastAPI is the robust library on top of python programming language which serves the API service as a connection between frontend and backend system. It receives input data and requests from frontend and serves it the appropriate response, and handles the backend services.
- **Inferencing Service:** The inferencing service comprises of the mechanism to initialise and run the Deepfake detection model. It receives input video and preprocesses it to be fed into the detection model. The detection model generates the output result.
- **Database Service:** The database technology used is the PostgreSQL which is one of the robust and scalable relational database. The database stores the information about the authenticated user, and the outputs.
- **Storage Bucket:** The storage bucket is the storage container where the uploaded videos are saved. These videos can be accessed later.

6.6.1.3 Integration of Frontend with the Backend

The integration between the frontend and backend ensures seamless communication and smooth operation of the web application.

- **CORS Middleware:** FastAPI is configured with Cross-Origin Resource Sharing (CORS) middleware, enabling the frontend to interact with the backend seamlessly, even if they are hosted on different domains.
- **Error Handling:** The backend gracefully handles exceptions and returns appropriate HTTP status codes and error messages to the frontend, ensuring a reliable user experience.

In summary, our web application provides users with a convenient and reliable platform for detecting deepfake videos. The frontend offers an intuitive interface for uploading videos, while the backend system handles the server-side functionalities. This integration marks a significant milestone in our efforts to combat digital misinformation and enhance media authenticity.

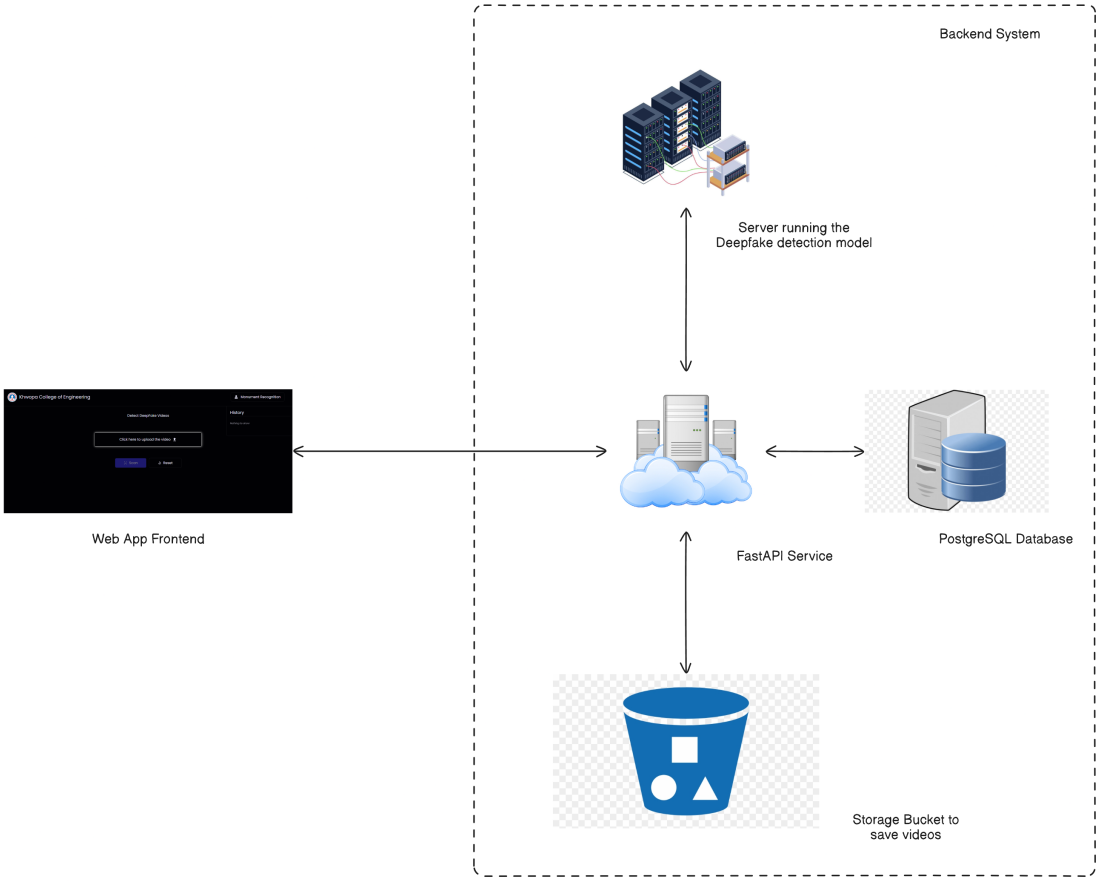


Figure 6.23: System Deployment

6.6.2 Test and Monitor

We tested our application thoroughly to ensure all components are functioning correctly and monitor our application for any issues or performance bottlenecks. The application is further optimized and designed in such a way that it could be scaled conveniently.

Chapter 7

Result and Discussion

7.1 Unit Tests

We performed tests on these units, and after inspecting the results of these tests, we made the necessary modifications and improvements, ensuring they would work as intended, before integrating all the units. Different test cases for our different units are shown below:

7.1.1 Face Detection and Alignment Module Unit

Test Scenario ID	Face Detection and Alignment Model Unit
Test Case Description	Check the module with image as input
Pre-conditions	Valid Image

Test execution steps include:

a. Initialise module.

b. Insert Image path in which face is to be detected.

c. Run the module.

S.N	**Input**	**Expected Output**	**Actual Output**	**Result**
1	Image containing a person	Cropped Image with face only	Appendix B.1 figure 1	Pass
2	Image containing an inclined face	Cropped Image with aligned face only	Appendix B.2 figure 2	Pass
3	Image containing multiple people	Raise "Multiple Faces Detected" Exception	Exception Raised and handled	Pass
4	Image containing no person	Raise "No Faces Detected" Exception	Exception Raised and handled	Pass

Table 7.1: Unit test cases of Face Detection and Alignment Module

7.1.2 Landmark and Blendshapes Extraction Unit

Test Scenario ID	Landmark and Blendshapes Extraction
Test Case Description	Extract landmarks and blendshapes from the image
Pre-conditions	Valid Image with a face

Test execution steps include:

a. Initialise module.

b. Insert an image path in which landmarks and blendshape of the face is to be extracted.

c. Pass the necessary argument to extract landmarks or blendshapes only or both.

d. Run the module.

S.N	**Input**	**Expected Output**	**Actual Output**	**Result**
1	Image of a person with face with landmarks only argument	Facial landmarks positions	Appendix B.3 figure 3	Pass
2	Image of a person with face with blendshapes only argument	Blendshape Coefficients	Appendix B.4 figure 4	Pass

Table 7.2: Unit test cases of Landmark and Blendshapes Extraction Module

7.1.3 Deepfake Detection Model Unit

Test Scenario ID	Deepfake Detection
Test Case Description	Classify whether the video is deepfake or real.
Pre-conditions	Valid input required by the model

Test execution steps include:

a. Initialise module.

b. Feed input to the module.

c. Run the module.

S.N	Input	Expected Output	Actual Output	Result
1	Input containing the preprocessed frames sequence, blendshapes and facial parts	Predicted label and confidence score	Predicted label and its confidence score	Pass

Table 7.3: Unit test case of Deepfake Detection Module

7.2 Integration Testing

Test Scenario ID	Integrated Deepfake Detection system
Test Case Description	Check the web application integration with the backend system and inferencing system, and inference the video
Pre-conditions	Function with all integrated modules that takes video as input and returns the label indicating whether the video is deepfake or real

Test execution steps include:

a. Open the web application

b. Upload the video

c. Press the scan button and wait for the response

d. Click on more details button

e. Check the detection output

f. Check other features like authentication etc.

S.N	Action	Expected Output	Actual Output	Result
1	Enter the webapp URL in the browser	Properly loaded homepage and responsive UI	Appendix B.5.1 figure 5	Pass
2	Press the sign in button	Google consent screen asking for authentication	Appendix figure B.5.2 figure 6	Pass
3	Upload the authentic video of a person	Inferencing server returns result of detection as label 'Real' and other information	Appendix figure B.5.3 figure 7	Pass
4	Upload the deepfake video of a person	Inferencing server returns result of detection as label 'Deepfake' and other information	Appendix figure B.5.4 figure 8	Pass
5	Upload the video with no person or face	Inferencing server returns the response as 'No Face Detected'	Appendix figure B.5.5 figure 9	Pass
6	Click on the video from the History tab	Inferencing is done on the video again and displays result	Appendix figure B.5.6 figure 10	Pass
7	Click on the delete button on the video from the History tab	Video is removed from the History tab	Appendix figure B.5.7 figure 11	Pass

Table 7.4: Integration test of Deepfake Detection System

7.3 Model Evaluation and Analysis

7.3.1 Model Evaluation Metrices

In the model evaluation process, the F1 score was employed, derived from the confusion matrix. The confusion matrix allows a detailed analysis of the model's performance by categorizing predictions into true positives, true negatives, false positives, and false negatives. The F1 score, a balance of precision and recall, provides a comprehensive assessment of the model's ability to correctly identify both authentic and deepfake videos while minimizing false classifications. This metric is particularly valuable in scenarios where class imbalances exist. The evaluation, driven by the confusion matrix and F1 score, offers insights into the model's effectiveness in accurately discerning between authentic and manipulated content, providing a nuanced understanding of its performance.

True Positive (TP): It refers to the number of predictions where the classifier correctly predicts the positive class as positive.

True Negative (TN): It refers to the number of predictions where the classifier correctly predicts the negative class as negative.

		Predicted	
		Negative	**Positive**
Actual	**Negative**	True Negative	False Positive
	Positive	False Negative	True Positive

Figure 7.1: Confusion Matrix

False Positive (FP): It refers to the number of predictions where the classifier incorrectly predicts the negative class as positive.

False Negative (FN): It refers to the number of predictions where the classifier incorrectly predicts the positive class as negative.

7.3.1.1 Precision

Precision is a good measure to determine, when the costs of False Positive is high. Precision talks about how precise/accurate a model is out of those predicted positive, how many of them are actual positive. The denominator is the Total Predicted Positive.

$$Precision = \frac{TP}{TP + FP} = \frac{TP}{Total\ Predicated\ Positive} \tag{7.1}$$

7.3.1.2 Recall

Recall actually calculates how many of the Actual Positives our model capture through labeling it as Positive (True Positive).

$$Recall = \frac{TP}{TP + FN} = \frac{TP}{Total\ Actual\ Positive} \tag{7.2}$$

7.3.1.3 Accuracy

It measures the proportion of correctly classified samples among all the samples in the test set, meaning the fraction of the total samples that were correctly classified by the classifier. The formula of the Accuracy considers the sum of True Positive and True Negative elements at the numerator and the sum of all the entries of the confusion matrix at the denominator.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} = \frac{Number\ of\ correctly\ classified\ samples}{Total\ number\ of\ samples} \tag{7.3}$$

7.3.1.4 F1 Score

F1 is a function of Precision and Recall. F1 Score is needed when you want to seek a balance between Precision and Recall.

$$F1Score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{7.4}$$

7.3.2 Progression of the Model

Over time, we've enhanced the model in successive versions. With each iteration, we have analysed its output and performance metrics and refined it further. Through careful attention and study, We identified the problems and incorporated the fixes in successive versions. These enhancements have resulted in a more effective and reliable model, better equipped to tackle various tasks and provide more accurate predictions or classifications. With each version, we've built upon previous successes, continually improving to deliver better results.

7.3.2.1 Initial Version

The initial version of our Deepfake Detection model was pretty simple in architecture and it incorporated only two main building blocks. i.e. Feature extraction layer ResNeXt and LSTM. The model was trained on smaller dataset.

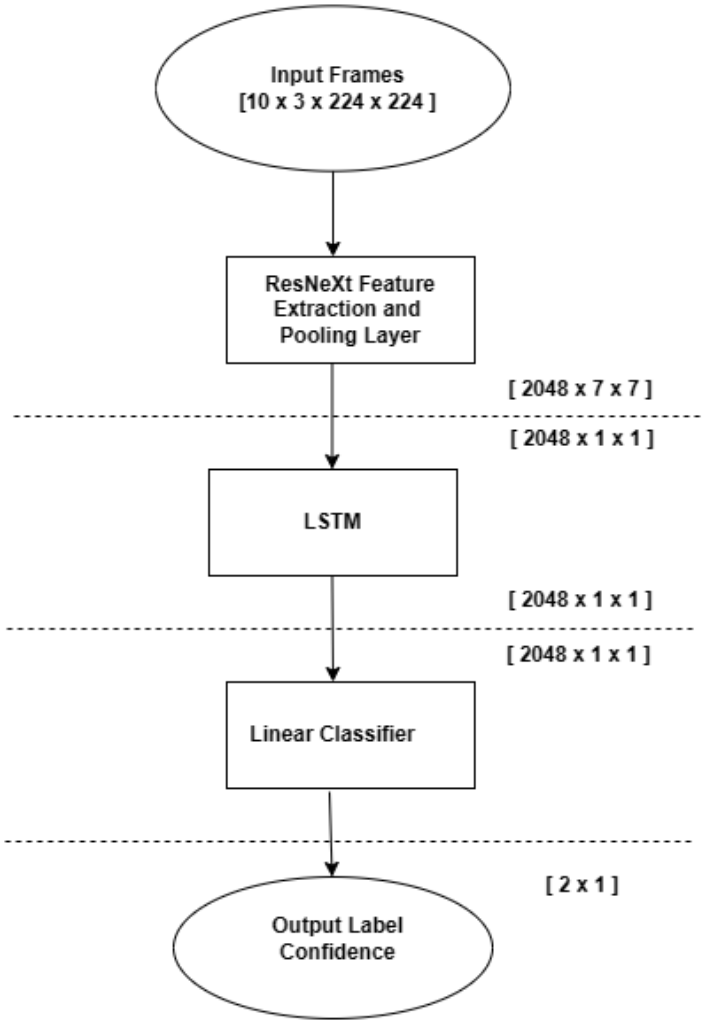
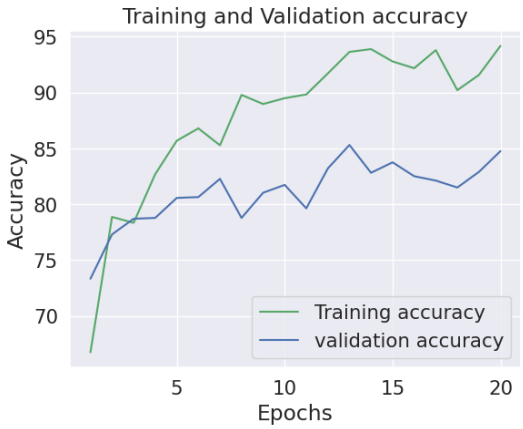


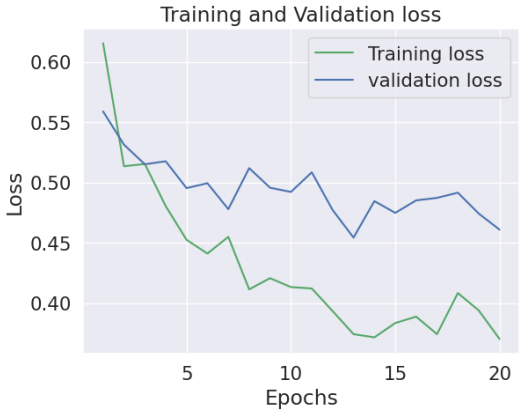
Figure 7.2: Illustrating the flow between different models and layers

S.N.	Hyper Parameters	Value
1	Batch Size	8
2	Input Frame Size	224 x 224
3	Sequence Length	10
4	Learning Rate	0.0001
5	Epochs	20

Table 7.5: Hyper Parameters used in Initial Version of model



(a) Accuracy Plot of Initial Version of model



(b) Loss Plot of Initial Version of model

Figure 7.3: Performance Graphs of Initial Version of model

From the above graphs, it can be seen that the model is overfitting since the train and validation curves are diverging as the epoch progresses. The suspected reason behind this is that the model is too simple for the task and the quality and quantity of dataset needs to be increased as well.

7.3.2.2 Mid Version

The mid version of our Deepfake Detection model was the successor of the Initial version and it incorporated two main building blocks i.e. Feature extraction layer ResNeXt and LSTM, along with the additional layers of dense layers, dropout layers and activation functions. The initial dataset was extended too and the mid version of the model was trained on the extended dataset.

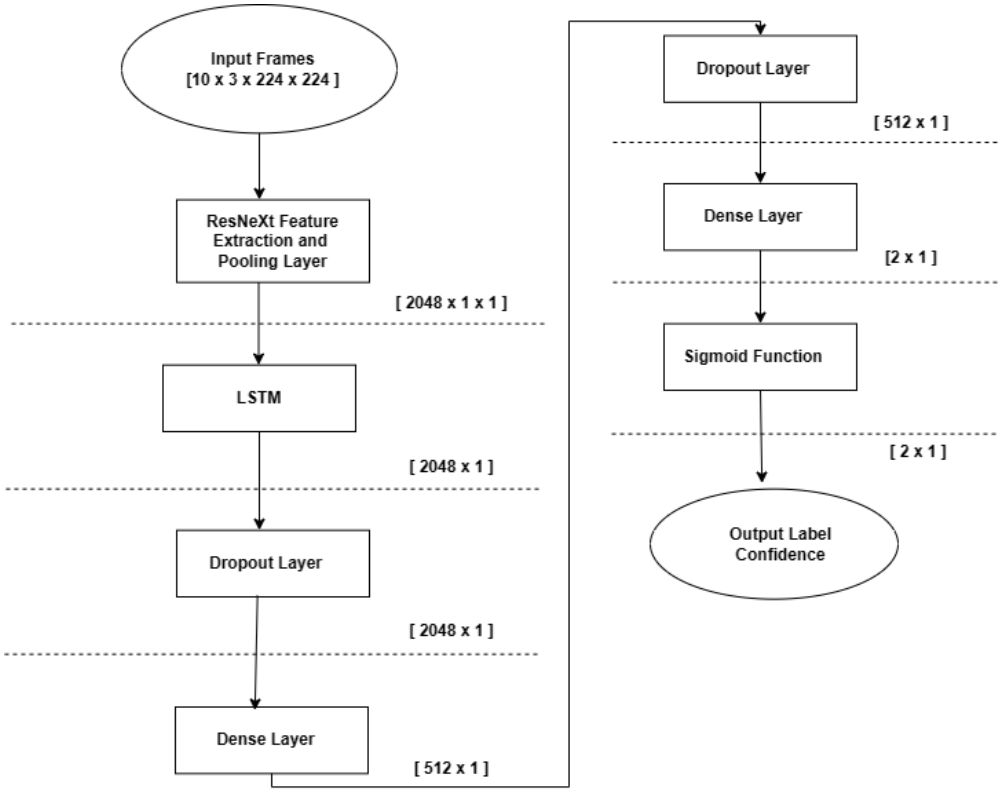
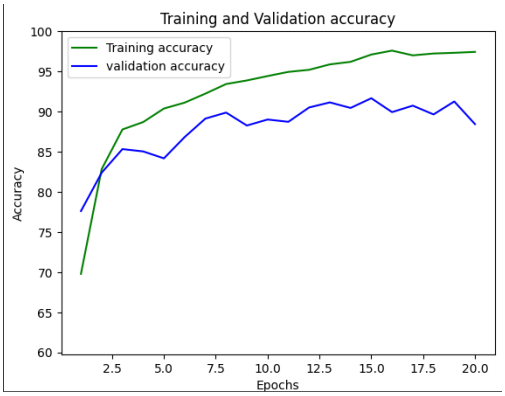


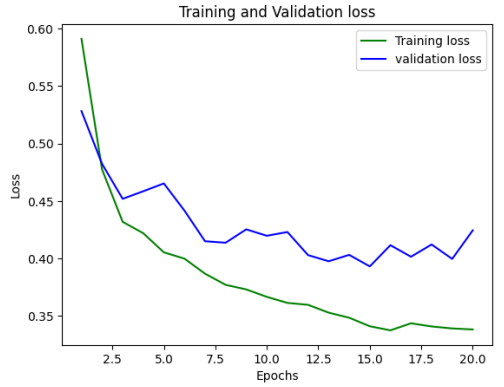
Figure 7.4: Illustrating the flow between different models and layers

S.N.	Hyper Parameters	Value
1	Batch Size	8
2	Input Frame Size	224 x 224
3	Sequence Length	10
4	Learning Rate	0.0001
5	Epochs	20

Table 7.6: Hyper Parameters used in Mid Version of model



(a) Accuracy Plot of Mid Version of model



(b) Loss Plot of Mid Version of model

Figure 7.5: Performance Graphs of Mid Version of model

From the above graphs, it can be seen that the model is improving than the Initial Version. Since, the model complexity is increasing as well as the larger number of dataset is used to train this version of model, the model seems to have improvement. However, the train and test plots are not yet converging and overfitting could also be anticipated.

7.3.2.3 Main Version

The Main version of our Deepfake Detection model was the successor of the Mid version and by far, it is the best we achieved. It incorporated multiple building block including Feature Extraction layer, Face Embedding Layers, BiLSTM and additional layers of dense layers, dropout layers and activation functions. The input to the model included facial crops as well as the blendshape scores too which further aided in improving the accuracy of the model. The dataset was extended further too.

z

S.N.	Hyper Parameters	Value
1	Batch Size	10
2	Input Frame Size	224 x 224
3	Sequence Length	10
4	Learning Rate	0.0001
5	Epochs	40

Table 7.7: Hyper Parameters used in Main Version of model

From the figure 7.7, it can be seen that the model is improving than the predecessor versions. In this version, we implemented the BiLSTM rather than the LSTM implementation that were used in previous versions. The BiLSTM implementation indicated the significant improvement. Further evaluation and comparison on BiLSTM and LSTM implementation is discussed in the later topics. Other factors like increasing model complexity and further features from the input like blendshapes, facial crops embeddings etc. also contributed to better result.

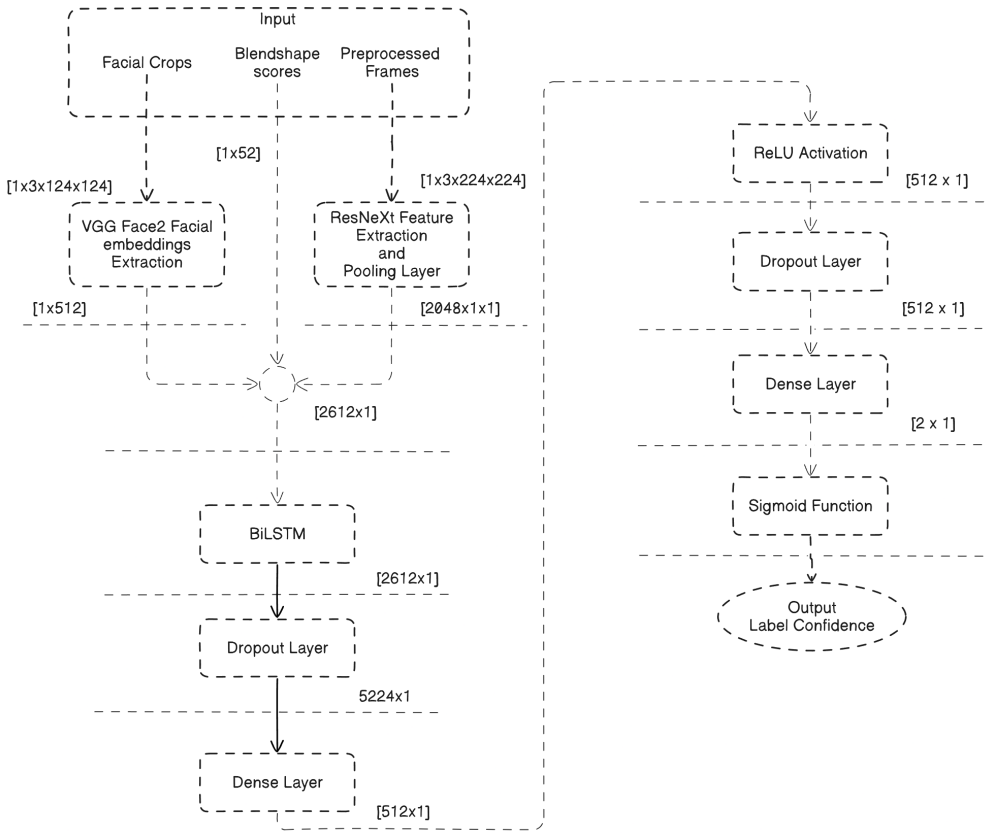
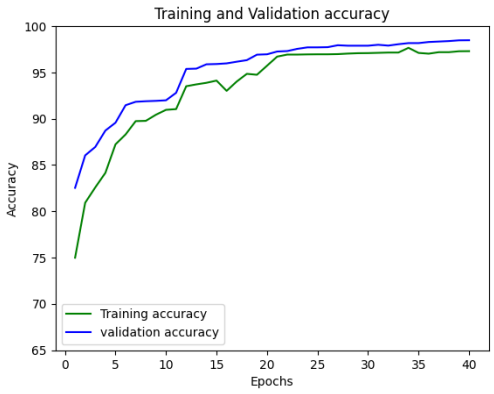
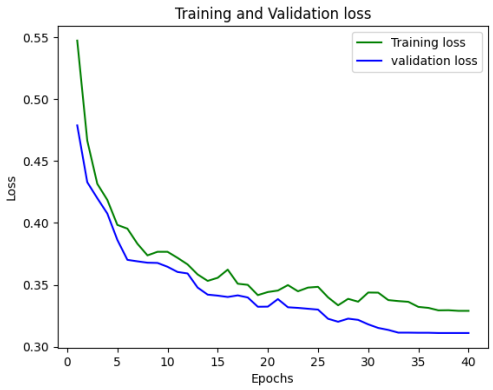


Figure 7.6: Illustrating the flow between different models and layers



(a) Accuracy Plot of Main Version of model



(b) Loss Plot of Main Version of model

Figure 7.7: Performance Graphs of Main Version of model

Table 7.8: Evaluation Results of different versions of Model

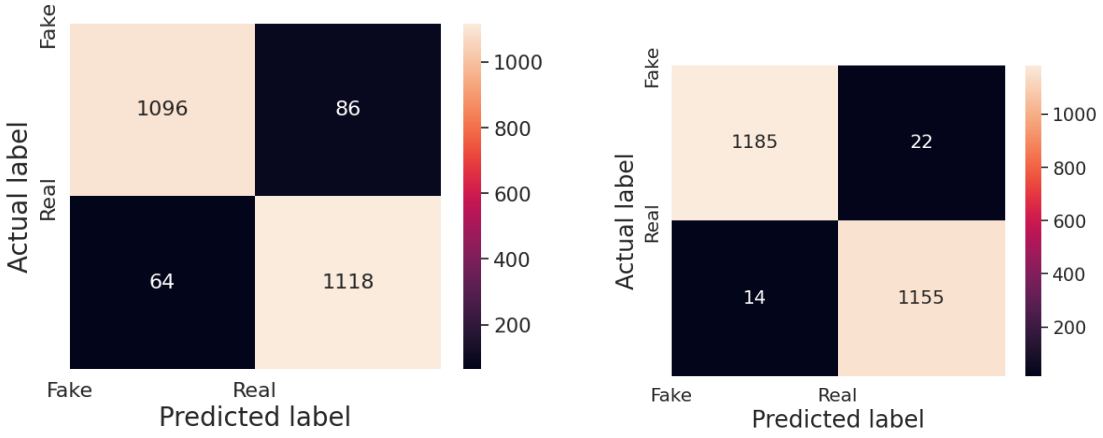
S.N.	Model	Dataset	Sequence length	Accuracy	Precision	Recall	F1-score
1	Initial	Cropped	10	84.74%	0.858	0.831	0.844
2	Mid	Cropped	10	90.96%	0.934	0.9569	0.944
3	Main	Cropped	10	98.5%	0.983	0.987	0.985

7.3.3 Evaluation of LSTM and BiLSTM

Our evaluation compares LSTM and BiLSTM models for deepfake video detection, providing insights into their effectiveness. By analyzing confusion matrices and training/validation metrics, we assess their predictive capabilities and generalization. The results contribute to combating deepfake proliferation and preserving digital trust. Evaluating both models allows us to understand their strengths and weaknesses, facilitating informed decision-making in deploying detection systems and adapting to evolving threats.

7.3.3.1 Confusion Matrix

Confusion matrices are vital for assessing the performance of deepfake detection systems, including those using ResNeXt, LSTM, and BiLSTM models. They break down model predictions against actual labels, aiding in the evaluation of true positives, true negatives, false positives, and false negatives. Through precision and recall calculations, they gauge a model's accuracy in distinguishing between authentic and manipulated content, guiding the refinement of detection algorithms to combat the proliferation of deceptive deepfake videos.



(a) Confusion Matrix of cropped dataset of LSTM

(b) Confusion Matrix of cropped dataset of BiLSTM

Figure 7.8: Confusion Matrices of LSTM and BiLSTM

Table 7.9: Evaluation Results of LSTM and BiLSTM Models

S.N.	Model	Dataset	Sequence length	Accuracy	Precision	Recall	F1-score
1	LSTM	Cropped	10	93.6%	0.929	0.946	0.937
2	BiLSTM	Cropped	10	98.5%	0.983	0.987	0.985

1. Calculation of evaluation metrics from confusion matrix for LSTM cropped dataset:

   True Positive(TP)= 1118
   True Negative(TN)= 1096

False Positive(FP)= 86
False Negative(FN)= 64

$$a.Precision = \frac{TP}{TP + FP} = \frac{1118}{1118 + 86} \approx 0.929$$
$$b.Recall = \frac{TP}{TP + FN} = \frac{1118}{1118 + 64} \approx 0.946$$
$$c.Accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{1118 + 1096}{1118 + 1096 + 86 + 64} \approx 0.936 = 93.6\%$$
$$d.F1Score = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2 * 0.929 * 0.946}{0.929 + 0.946} \approx 0.937$$

From the test dataset evaluation, accuracy of LSTM model for cropped dataset is calculated as 93.6%.

2. Calculation of evaluation metrics from confusion matrix for BiLSTM cropped dataset:

   True Positive(TP)= 1155
   True Negative(TN)= 1185
   False Positive(FP)= 20
   False Negative(FN)= 14

$$a.Precision = \frac{TP}{TP + FP} = \frac{1155}{1155 + 20} \approx 0.983$$
$$b.Recall = \frac{TP}{TP + FN} = \frac{1155}{1155 + 14} \approx 0.987$$
$$c.Accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{1155 + 1185}{1155 + 1185 + 20 + 14} \approx 0.985 = 98.5\%$$
$$d.F1Score = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2 * 0.983 * 0.987}{0.983 + 0.987} \approx 0.985$$

   From the test dataset evaluation, accuracy of BiLSTM model for cropped dataset is calculated as 98.5%.

   BiLSTM model outperforms the LSTM model in all evaluation metrics, including precision, recall, accuracy, and F1 score. It demonstrates higher accuracy and better ability to correctly classify both deepfake and authentic videos, making it a more effective choice for deepfake detection tasks.

7.3.3.2 Evaluation of BiLSTM model with different sequence lengths

We evaluated the BiLSTM model with different sequence lengths i.e 10, 20. So obtained accuracies of the model is tabulated below:

S.N.	Sequence Length	Accuracy
1	10	98.3%
2	20	98.7%

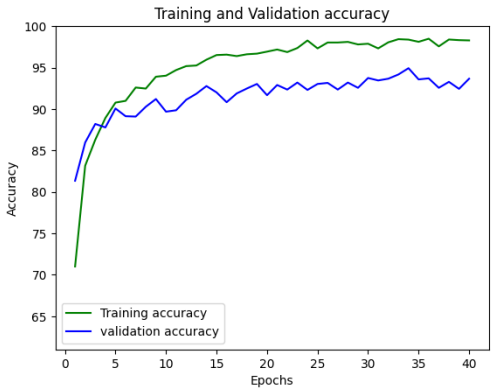
Evaluation Results of BiLSTM Model (Accuracy)

7.3.3.3 Training and Validation Accuracy

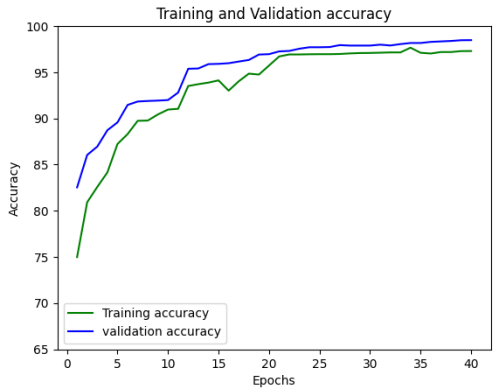
Training accuracy refers to the accuracy of a machine learning model on the data it was trained on. It measures how well the model fits the training data.

Validation accuracy, on the other hand, refers to the accuracy of the model on a separate dataset called the validation set. This dataset is not used for training and helps evaluate how well the model generalizes to new, unseen data.

In short, training accuracy evaluates how well the model learns from the training data, while validation accuracy assesses how well it performs on new data.



(a) Training and Validation Accuracy of LSTM



(b) Training and Validation Accuracy of BiLSTM

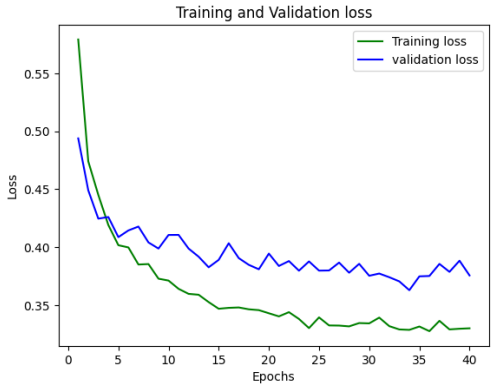
Figure 7.9: Training and Validation Accuracy of LSTM and BiLSTM

7.3.3.4 Training and Validation Loss

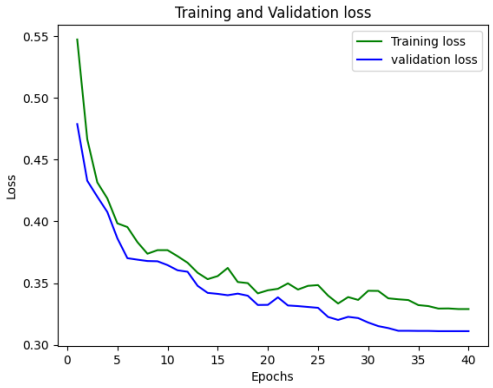
Training loss measures the error between the actual and predicted values during the training phase, indicating how well the model is learning from the data.

Validation loss is similar but computed on a separate dataset not seen during training, showing how well the model generalizes to new data.

Both are crucial in monitoring and optimizing the model's performance, with the goal of minimizing both training and validation loss for effective learning and generalization.



(a) Training and Validation Loss of LSTM



(b) Training and Validation Loss of BiLSTM

Figure 7.10: Training and Validation Loss of LSTM and BiLSTM

7.4 Inferencing Time

Following table represents the average inferencing time taken by the modules of the detection system on different devices, on the input video of sequence length 10.

S.N.	Device	Face Detection	Landmarks and Blendshapes Extraction	Model	Total
1	CPU (4 Cores)	3.8 sec	0.8 sec	12.5 sec	17.1 sec
2	GPU (Tesla T4)	2.2 sec	0.5 sec	0.2 sec	2.9 sec

Table 7.11: Inferencing time on different devices

7.5 Web Application

Users can visit https://deepfakedetection.me/ to access the system. Upon arrival, users will encounter an interface resembling the image provided below:

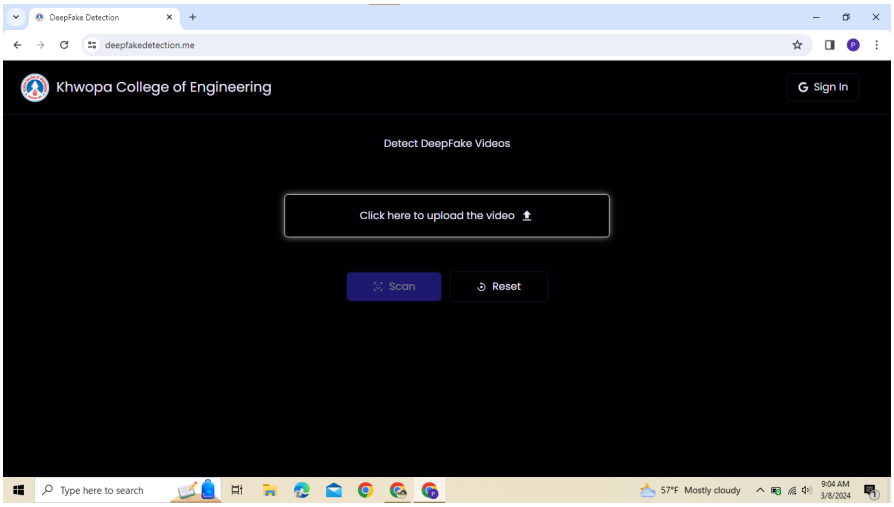


Figure 7.11: System Deployment

Chapter 8
Conclusion

The detection of deepfakes emerges as a pivotal battleground in the ongoing struggle against the dissemination of misinformation and the manipulation of digital content. With the continuous advancement of artificial intelligence algorithms, the generation of synthetic media reaches unprecedented levels of realism, underscoring the urgent necessity for correspondingly sophisticated detection mechanisms. Nevertheless, this endeavor is beset by formidable challenges stemming from the rapid evolution of deepfake techniques, as well as the adeptness of malicious actors in evading and subverting existing detection measures.

Detecting deepfake videos is crucial for verifying the authenticity of information and avoiding erroneous decision-making. This involves processing video data, training classification algorithms, and evaluating their performance on distinguishing between deepfake and genuine videos.In our project, we preprocessed each data into dimension of 224 pixels with 5 secs fixed duration in a sequence of 150 frames. Utilizing ResNext CNN for video feature extraction, resulting 2048-dimensional feature vector. This output acts as input to our BiLSTM model employed for image classification, with performance assessed using accuracy, precision, and recall metrics. Results from ResNext CNN-BiLSTM classification indicate a 98.5% accuracy, 98.3% precision, and 98.7% recall, and 98.5% F1-Score for 10-sequence length, while 20, and 30-sequence length exhibit higher performance at 98.7% and 99%.

Additionally, we also compared our BiLSTM model with LSTM model where LSTM classification indicate a 93.6% accuracy, 92.9% precision, and 94.6% recall, and 93.7% F1-Score. This concludes BiLSTM to be better because of obvious reasons.

As a conclusion, the detection of deepfakes remains a formidable challenge, concerted efforts towards innovation, collaboration, and education hold promise in mitigating their potential harms and safeguarding the integrity of digital content.

Chapter 9

Limitations and Future Enhancements

9.1 Limitation

There are some limitations to our project which could be summarized by the following points:

- May not generalize well to new and unseen types of deepfake content
- Susceptible to adversarial attacks
- Struggles to achieve good accuracy if the input is highly realistic content

9.2 Future Enhancements

As for the Future Enhancement for our project, the following implementation can be made:

- Utilize multi-modal information (e.g., audio, text) in addition to visual data to enhance the deepfake detection performance,
- Implement techniques for continuous learning to adapt your deepfake detection model to evolving deepfake generation techniques,
- Introduce adversarial training to make your model more robust against adversarial attacks, and
- Develop a browser extension that can automatically analyze videos during web browsing sessions for potential deepfake content.

Bibliography

[1] L. Nataraj, T. M. Mohammed, S. Chandrasekaran, A. Flenner, J. H. Bappy, A. K. Roy-Chowdhury, and B. Manjunath, "Detecting gan generated fake images using co-occurrence matrices," *arXiv preprint arXiv:1903.06836*, 2019.

[2] D. R. Coats, "Worldwide threat assessment," *US Intelligence Community*, 2019.

[3] M. Appel and F. Prietzel, "The detection of political deepfakes," *Journal of Computer-Mediated Communication*, vol. 27, no. 4, p. zmac008, 2022.

[4] T. T. Nguyen, Q. V. H. Nguyen, D. T. Nguyen, D. T. Nguyen, T. Huynh-The, S. Nahavandi, T. T. Nguyen, Q.-V. Pham, and C. M. Nguyen, "Deep learning for deepfakes creation and detection: A survey," *Computer Vision and Image Understanding*, vol. 223, p. 103525, 2022.

[5] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, "Analyzing and improving the image quality of stylegan," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 8110–8119.

[6] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 4401–4410.

[7] Y. Li and S. Lyu, "Exposing deepfake videos by detecting face warping artifacts," *arXiv preprint arXiv:1811.00656*, 2018.

[8] V. L. Thing, "Deepfake detection with deep learning: Convolutional neural networks versus transformers," *arXiv preprint arXiv:2304.03698*, 2023.

[9] M. F. Hashmi, B. K. K. Ashish, A. G. Keskar, N. D. Bokde, J. H. Yoon, and Z. W. Geem, "An exploratory analysis on visual counterfeits using conv-lstm hybrid architecture," *IEEE Access*, vol. 8, pp. 101 293–101 308, 2020.

[10] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, and Y. Bengio, "Generative adversarial nets," *Neural Networks*, pp. 50–65, 2014.

[11] X. Yang, Y. Li, and S. Lyu, "Exposing deep fakes using inconsistent head poses," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 30–45, 2019.

[12] J. Mallet, N. Krueger, R. Dave, and M. Vanamala, "Hybrid deepfake detection utilizing mlp and lstm," in *2023 3rd International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)*. IEEE, 2023, pp. 1–5.

[13] L. A. Passos, D. Jodas, K. A. da Costa, L. A. S. Júnior, D. Rodrigues, J. Del Ser, D. Camacho, and J. P. Papa, "A review of deep learning-based approaches for deepfake content detection," *arXiv preprint arXiv:2202.06095*, 2022.

[14] R. Tolosana, R. Vera-Rodriguez, J. Fierrez, A. Morales, and J. Ortega-Garcia, "Deepfakes and beyond: A survey of face manipulation and fake detection," *Information Fusion*, vol. 64, pp. 131–148, 2020.

[15] Y. Li, M.-C. Chang, and S. Lyu, "In ictu oculi: Exposing ai generated fake face videos by detecting eye blinking," *arXiv preprint arXiv:1806.02877*, 2018.

[16] Y. Li, M. C. Chang, and S. Lyu, "In ictu oculi: Exposing ai-generated fake face videos by detecting eye blinking," *Computer Vision and Image Understanding*, pp. 55–70, 2018.

[17] K. Raja, S. Venkatesh, R. Christoph Busch *et al.*, "Transferable deep-cnn features for detecting digital and print-scanned morphed face images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017, pp. 10–18.

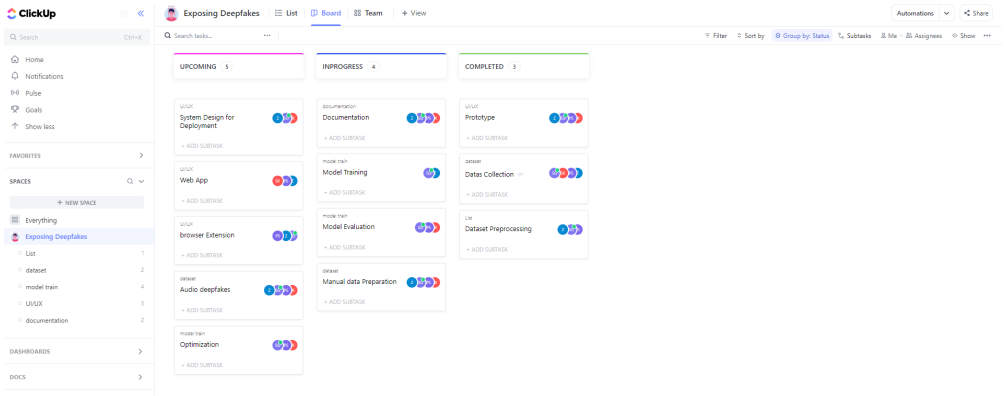
[18] M. A. Saleem, N. Senan, F. Wahid, M. Aamir, A. Samad, M. Khan *et al.*, "Comparative analysis of recent architecture of convolutional neural network," *Mathematical Problems in Engineering*, vol. 2022, 2022.

Chapter 10

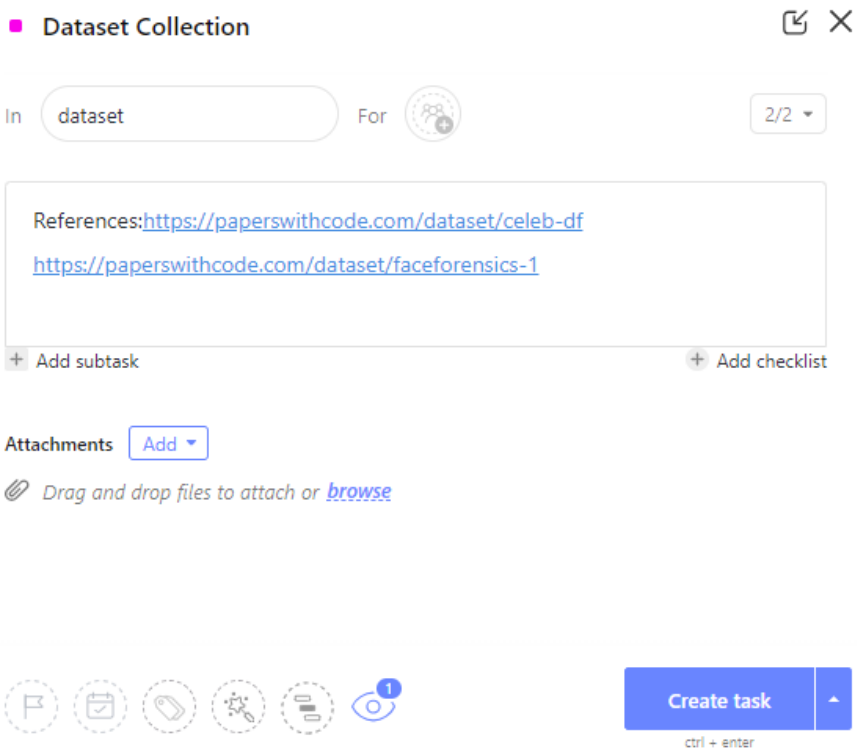
Appendix

A Snapshot

A.1 ClickUp



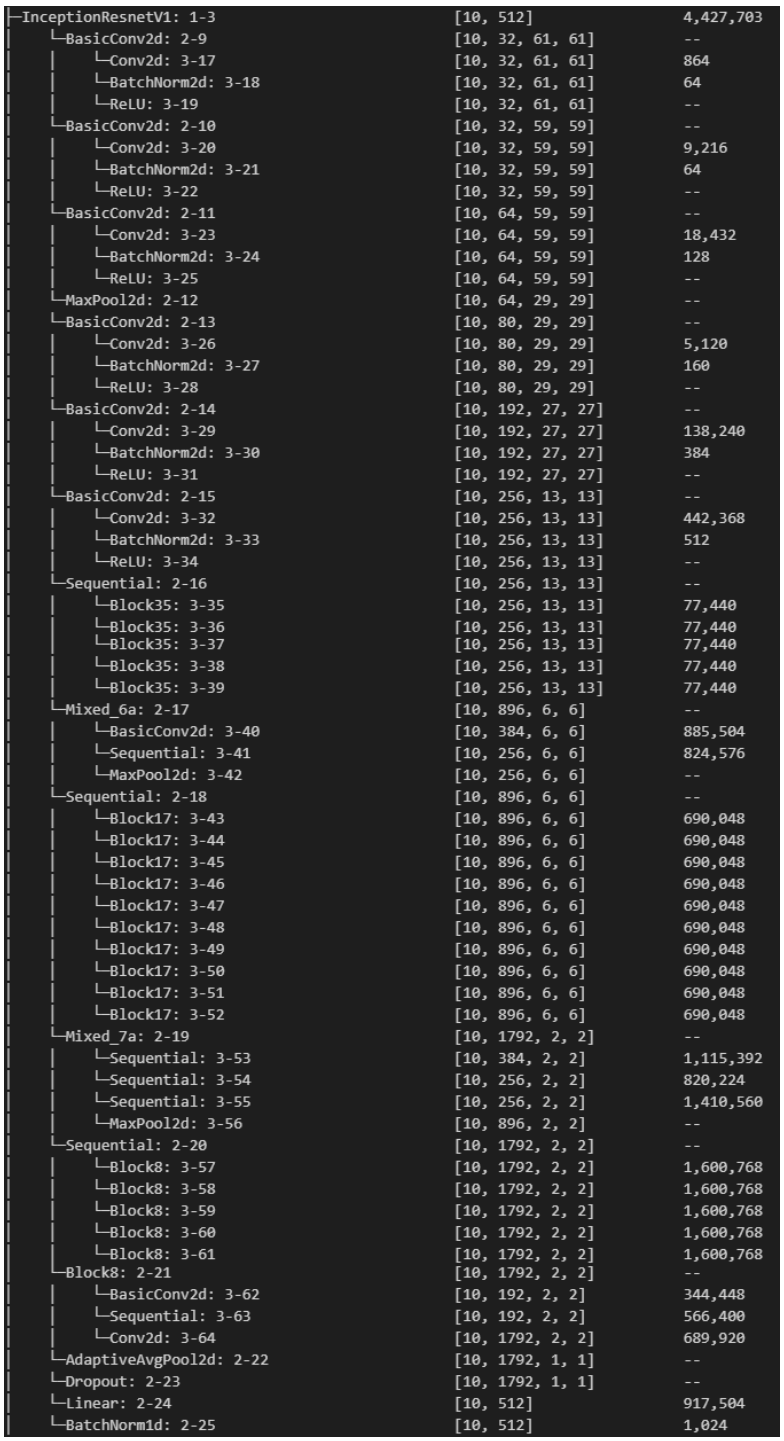
A.2 ClickUp



A.3 Model Summary of System

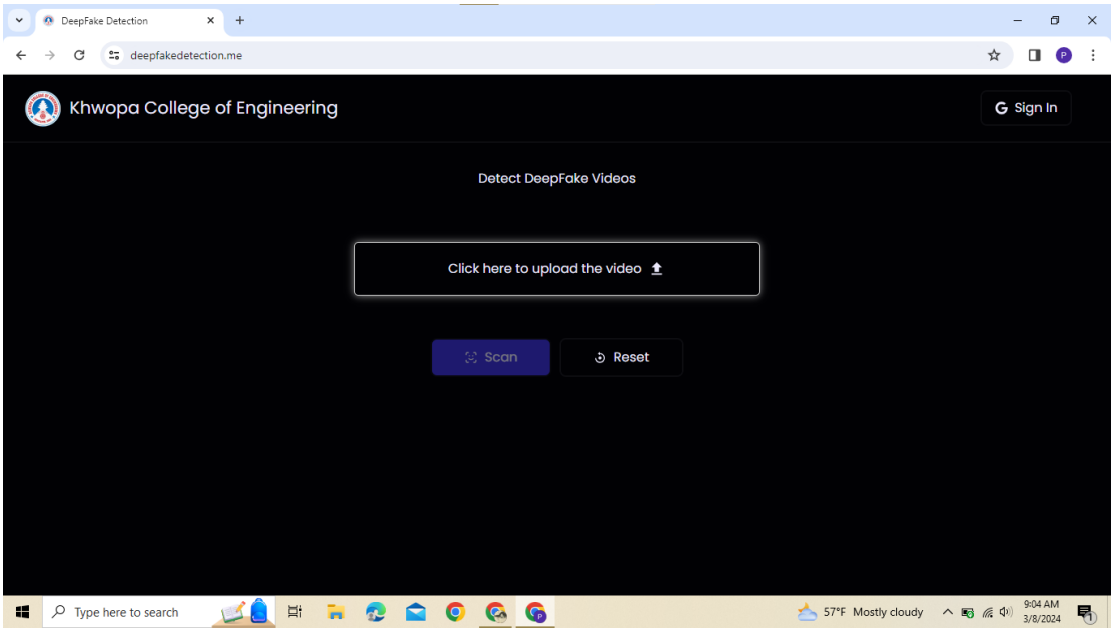
```
==========================================================================================
Layer (type:depth-idx)                             Output Shape              Param #
==========================================================================================
Model                                              --                        --
├─Sequential: 1-1                                  [10, 2048, 7, 7]          --
│    └─Conv2d: 2-1                                 [10, 64, 112, 112]        9,408
│    └─BatchNorm2d: 2-2                            [10, 64, 112, 112]        128
│    └─ReLU: 2-3                                   [10, 64, 112, 112]        --
│    └─MaxPool2d: 2-4                              [10, 64, 56, 56]          --
│    └─Sequential: 2-5                             [10, 256, 56, 56]         --
│    │    └─Bottleneck: 3-1                        [10, 256, 56, 56]         63,488
│    │    └─Bottleneck: 3-2                        [10, 256, 56, 56]         71,168
│    │    └─Bottleneck: 3-3                        [10, 256, 56, 56]         71,168
│    └─Sequential: 2-6                             [10, 512, 28, 28]         --
│    │    └─Bottleneck: 3-4                        [10, 512, 28, 28]         349,184
│    │    └─Bottleneck: 3-5                        [10, 512, 28, 28]         282,624
│    │    └─Bottleneck: 3-6                        [10, 512, 28, 28]         282,624
│    │    └─Bottleneck: 3-7                        [10, 512, 28, 28]         282,624
│    └─Sequential: 2-7                             [10, 1024, 14, 14]        --
│    │    └─Bottleneck: 3-8                        [10, 1024, 14, 14]        1,390,592
│    │    └─Bottleneck: 3-9                        [10, 1024, 14, 14]        1,126,400
│    │    └─Bottleneck: 3-10                       [10, 1024, 14, 14]        1,126,400
│    │    └─Bottleneck: 3-11                       [10, 1024, 14, 14]        1,126,400
│    │    └─Bottleneck: 3-12                       [10, 1024, 14, 14]        1,126,400
│    │    └─Bottleneck: 3-13                       [10, 1024, 14, 14]        1,126,400
│    └─Sequential: 2-8                             [10, 2048, 7, 7]          --
│    │    └─Bottleneck: 3-14                       [10, 2048, 7, 7]          5,550,080
│    │    └─Bottleneck: 3-15                       [10, 2048, 7, 7]          4,497,408
│    │    └─Bottleneck: 3-16                       [10, 2048, 7, 7]          4,497,408
├─AdaptiveAvgPool2d: 1-2                           [10, 2048, 1, 1]          --
├─LSTM: 1-4                                        [1, 10, 5224]             109,202,496
├─Dropout: 1-5                                     [1, 5224]                 --
├─Linear: 1-6                                      [1, 512]                  2,675,200
├─ReLU: 1-7                                        [1, 512]                  --
├─Dropout: 1-8                                     [1, 512]                  --
├─Linear: 1-9                                      [1, 2]                    1,026
├─Sigmoid: 1-10                                    [1, 2]                    --
==========================================================================================
Total params: 162,768,953
Trainable params: 162,768,953
Non-trainable params: 0
Total mult-adds (G): 51.36
==========================================================================================
Input size (MB): 7.87
Forward/backward pass size (MB): 2542.68
Params size (MB): 633.37
Estimated Total Size (MB): 3183.91
==========================================================================================
```

A.4 Model Summary of InceptionResNetV1



├─InceptionResnetV1: 1-3	[10, 512]	4,427,703
│ └─BasicConv2d: 2-9	[10, 32, 61, 61]	--
│ │ └─Conv2d: 3-17	[10, 32, 61, 61]	864
│ │ └─BatchNorm2d: 3-18	[10, 32, 61, 61]	64
│ │ └─ReLU: 3-19	[10, 32, 61, 61]	--
│ └─BasicConv2d: 2-10	[10, 32, 59, 59]	--
│ │ └─Conv2d: 3-20	[10, 32, 59, 59]	9,216
│ │ └─BatchNorm2d: 3-21	[10, 32, 59, 59]	64
│ │ └─ReLU: 3-22	[10, 32, 59, 59]	--
│ └─BasicConv2d: 2-11	[10, 64, 59, 59]	--
│ │ └─Conv2d: 3-23	[10, 64, 59, 59]	18,432
│ │ └─BatchNorm2d: 3-24	[10, 64, 59, 59]	128
│ │ └─ReLU: 3-25	[10, 64, 59, 59]	--
│ └─MaxPool2d: 2-12	[10, 64, 29, 29]	--
│ └─BasicConv2d: 2-13	[10, 80, 29, 29]	--
│ │ └─Conv2d: 3-26	[10, 80, 29, 29]	5,120
│ │ └─BatchNorm2d: 3-27	[10, 80, 29, 29]	160
│ │ └─ReLU: 3-28	[10, 80, 29, 29]	--
│ └─BasicConv2d: 2-14	[10, 192, 27, 27]	--
│ │ └─Conv2d: 3-29	[10, 192, 27, 27]	138,240
│ │ └─BatchNorm2d: 3-30	[10, 192, 27, 27]	384
│ │ └─ReLU: 3-31	[10, 192, 27, 27]	--
│ └─BasicConv2d: 2-15	[10, 256, 13, 13]	--
│ │ └─Conv2d: 3-32	[10, 256, 13, 13]	442,368
│ │ └─BatchNorm2d: 3-33	[10, 256, 13, 13]	512
│ │ └─ReLU: 3-34	[10, 256, 13, 13]	--
│ └─Sequential: 2-16	[10, 256, 13, 13]	--
│ │ └─Block35: 3-35	[10, 256, 13, 13]	77,440
│ │ └─Block35: 3-36	[10, 256, 13, 13]	77,440
│ │ └─Block35: 3-37	[10, 256, 13, 13]	77,440
│ │ └─Block35: 3-38	[10, 256, 13, 13]	77,440
│ │ └─Block35: 3-39	[10, 256, 13, 13]	77,440
│ └─Mixed_6a: 2-17	[10, 896, 6, 6]	--
│ │ └─BasicConv2d: 3-40	[10, 384, 6, 6]	885,504
│ │ └─Sequential: 3-41	[10, 256, 6, 6]	824,576
│ │ └─MaxPool2d: 3-42	[10, 256, 6, 6]	--
│ └─Sequential: 2-18	[10, 896, 6, 6]	--
│ │ └─Block17: 3-43	[10, 896, 6, 6]	690,048
│ │ └─Block17: 3-44	[10, 896, 6, 6]	690,048
│ │ └─Block17: 3-45	[10, 896, 6, 6]	690,048
│ │ └─Block17: 3-46	[10, 896, 6, 6]	690,048
│ │ └─Block17: 3-47	[10, 896, 6, 6]	690,048
│ │ └─Block17: 3-48	[10, 896, 6, 6]	690,048
│ │ └─Block17: 3-49	[10, 896, 6, 6]	690,048
│ │ └─Block17: 3-50	[10, 896, 6, 6]	690,048
│ │ └─Block17: 3-51	[10, 896, 6, 6]	690,048
│ │ └─Block17: 3-52	[10, 896, 6, 6]	690,048
│ └─Mixed_7a: 2-19	[10, 1792, 2, 2]	--
│ │ └─Sequential: 3-53	[10, 384, 2, 2]	1,115,392
│ │ └─Sequential: 3-54	[10, 256, 2, 2]	820,224
│ │ └─Sequential: 3-55	[10, 256, 2, 2]	1,410,560
│ │ └─MaxPool2d: 3-56	[10, 896, 2, 2]	--
│ └─Sequential: 2-20	[10, 1792, 2, 2]	--
│ │ └─Block8: 3-57	[10, 1792, 2, 2]	1,600,768
│ │ └─Block8: 3-58	[10, 1792, 2, 2]	1,600,768
│ │ └─Block8: 3-59	[10, 1792, 2, 2]	1,600,768
│ │ └─Block8: 3-60	[10, 1792, 2, 2]	1,600,768
│ │ └─Block8: 3-61	[10, 1792, 2, 2]	1,600,768
│ └─Block8: 2-21	[10, 1792, 2, 2]	--
│ │ └─BasicConv2d: 3-62	[10, 192, 2, 2]	344,448
│ │ └─Sequential: 3-63	[10, 192, 2, 2]	566,400
│ │ └─Conv2d: 3-64	[10, 1792, 2, 2]	689,920
│ └─AdaptiveAvgPool2d: 2-22	[10, 1792, 1, 1]	--
│ └─Dropout: 2-23	[10, 1792, 1, 1]	--
│ └─Linear: 2-24	[10, 512]	917,504
│ └─BatchNorm1d: 2-25	[10, 512]	1,024

A.5 Web Application Interface



B Unit Test

B.1 Face Detection with cropped Unit

Figure 1: Unit test 1 for detection module

B.2 Face Detection and Alignment Module Unit

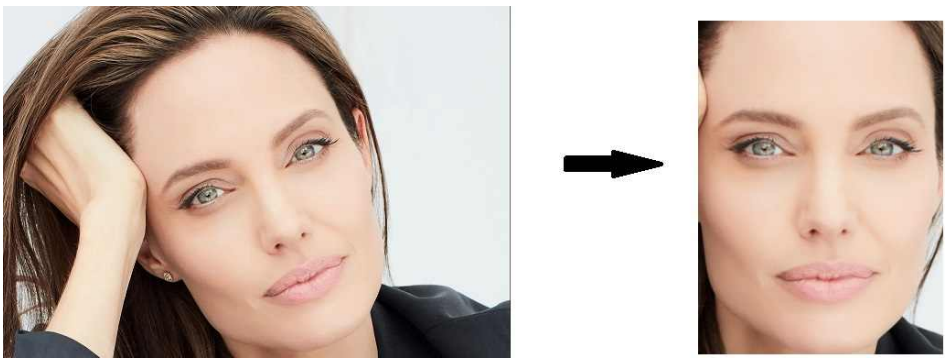


Figure 2: Unit test 1 for detection module

B.3 Facial Landmark

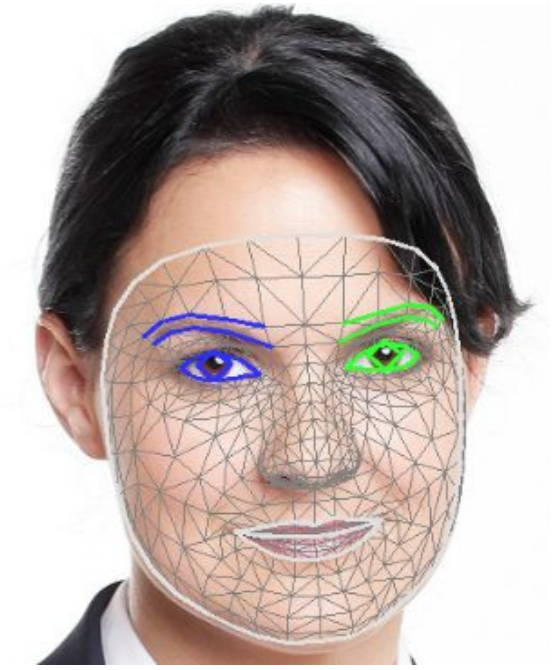


Figure 3: Facial Landmark

B.4 Blendshape Coefficients

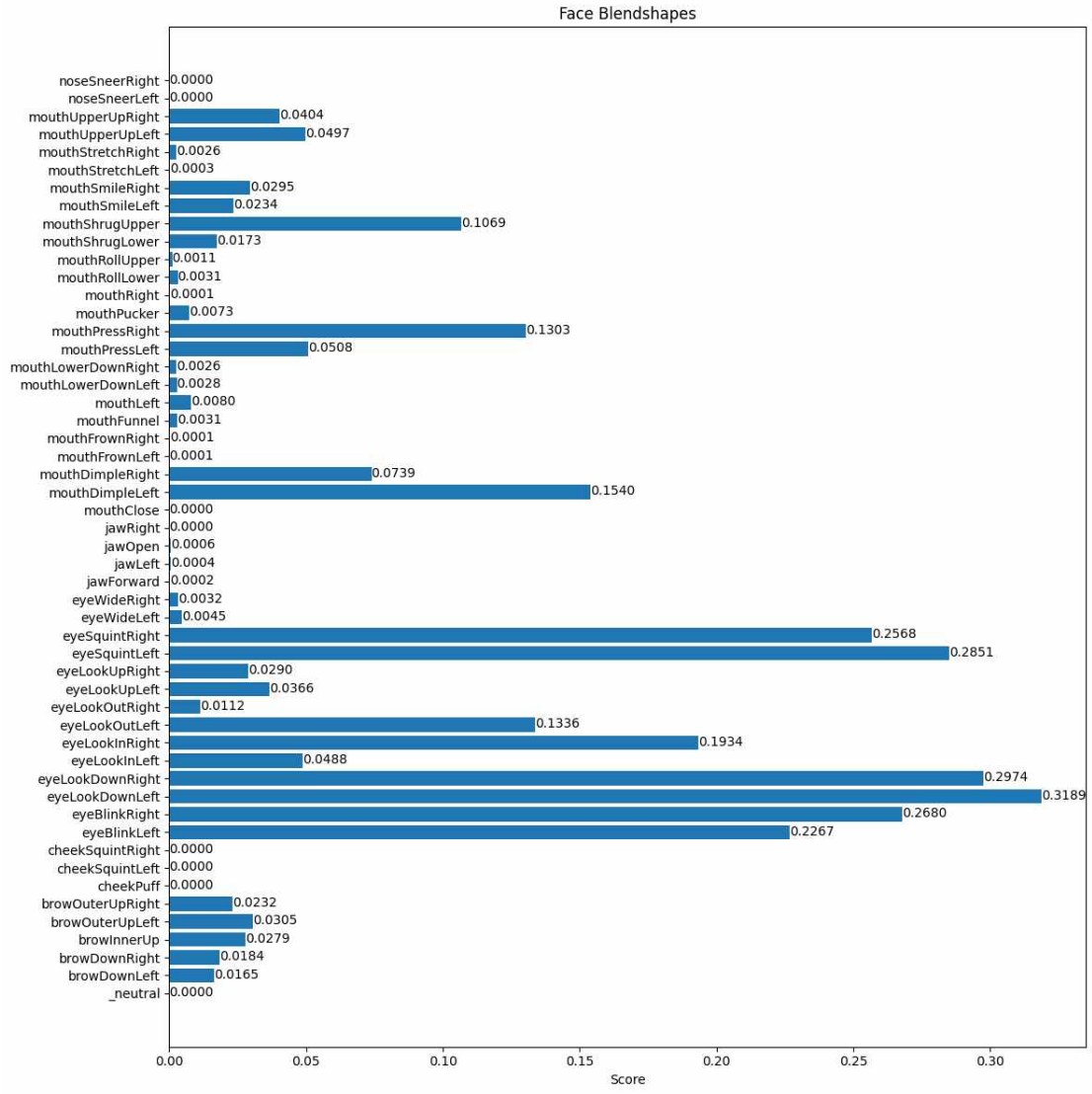


Figure 4: Blendshape Coefficients

B.5 Integration Test

B.5.1 Enter the webapp URL in the browser

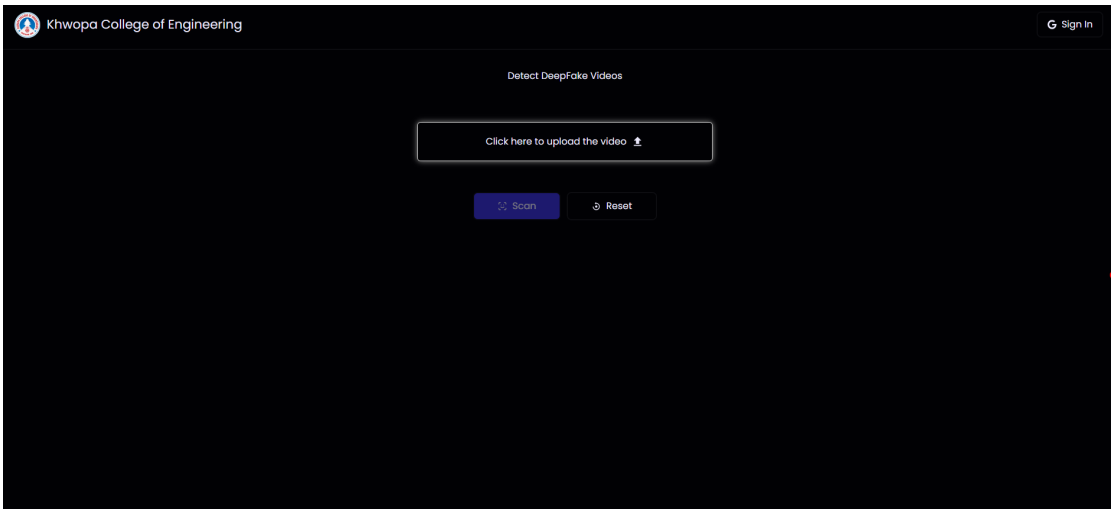


Figure 5: Integration Test 1

B.5.2 Press the sign in button

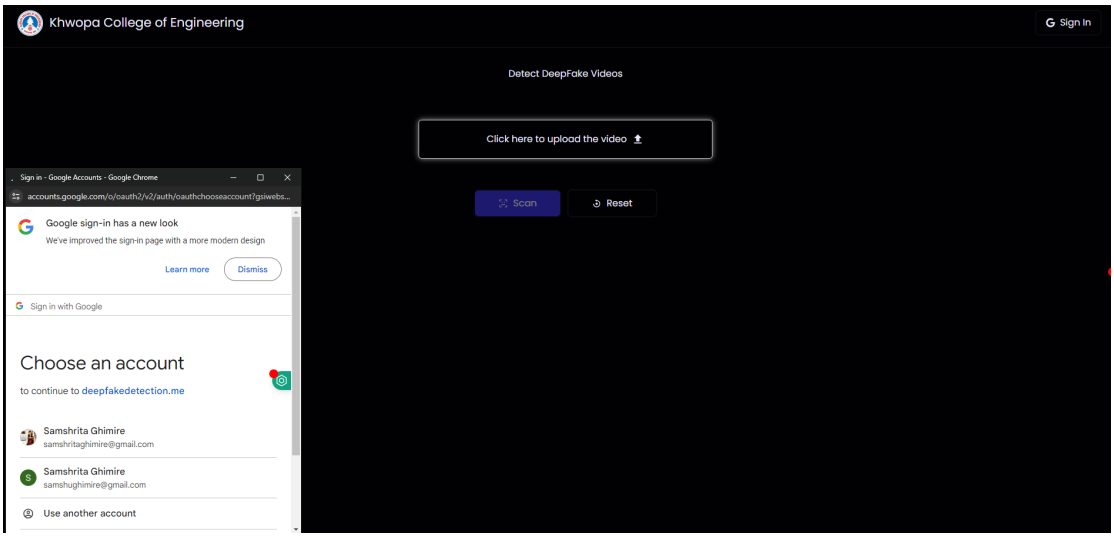


Figure 6: Integration Test 2

B.5.3 Upload the authentic video of a person

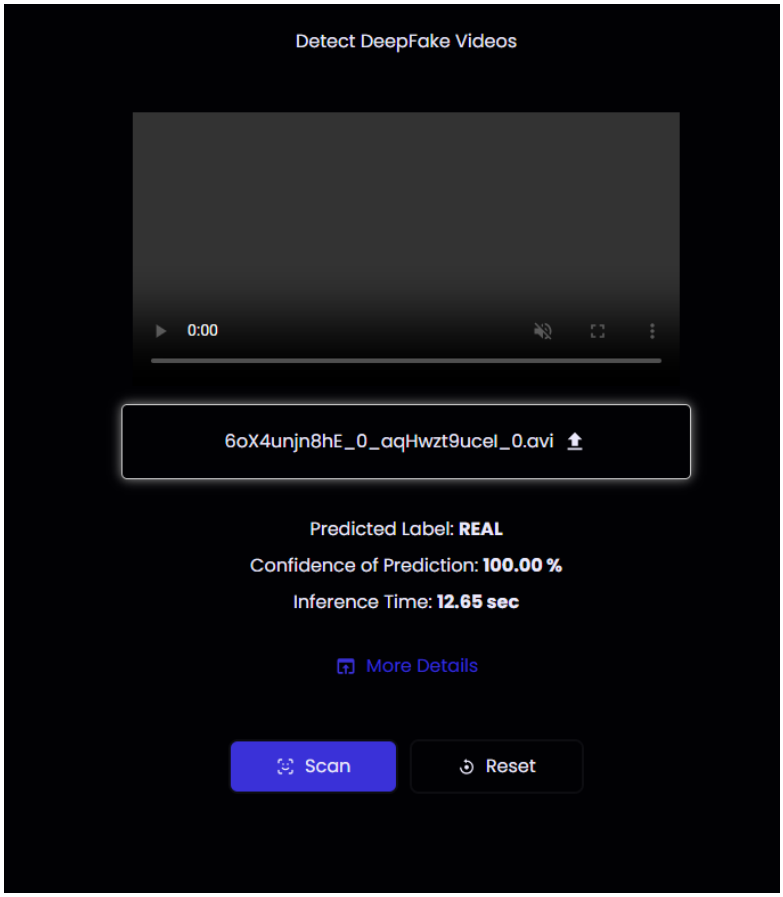


Figure 7: Integration Test 3

B.5.4 Upload the deepfake video of a person

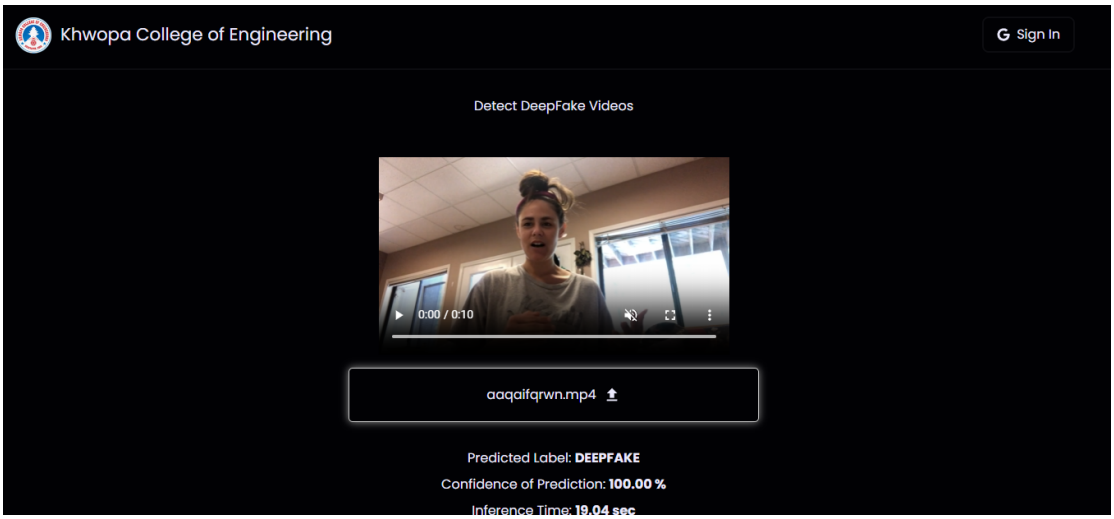


Figure 8: Integration Test 4

B.5.5 Upload the video with no person or face

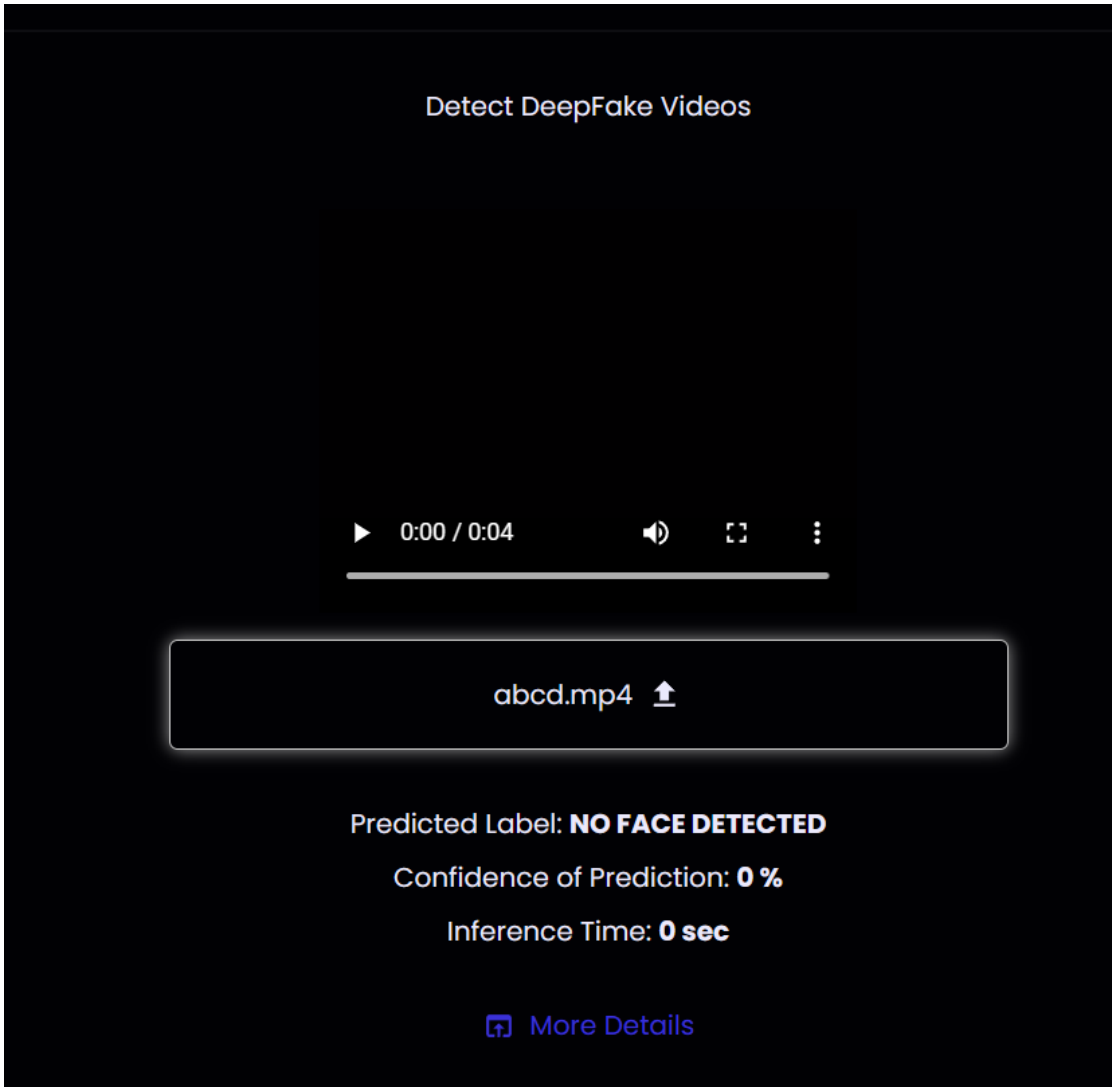


Figure 9: Integration Test 5

B.5.6 Click on the video from the History tab

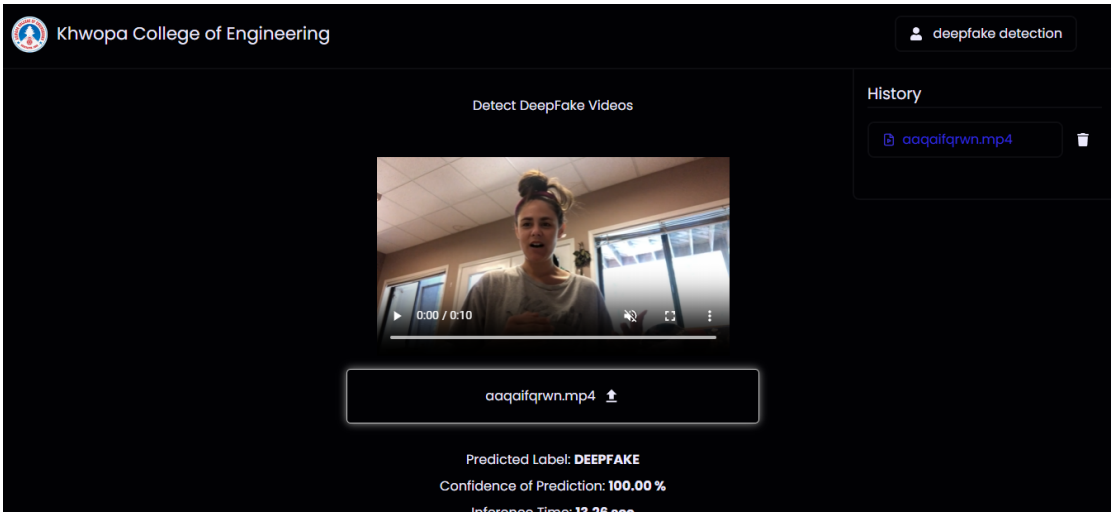


Figure 10: Integration Test 6

B.5.7 Click on the delete button on the video from the History tab

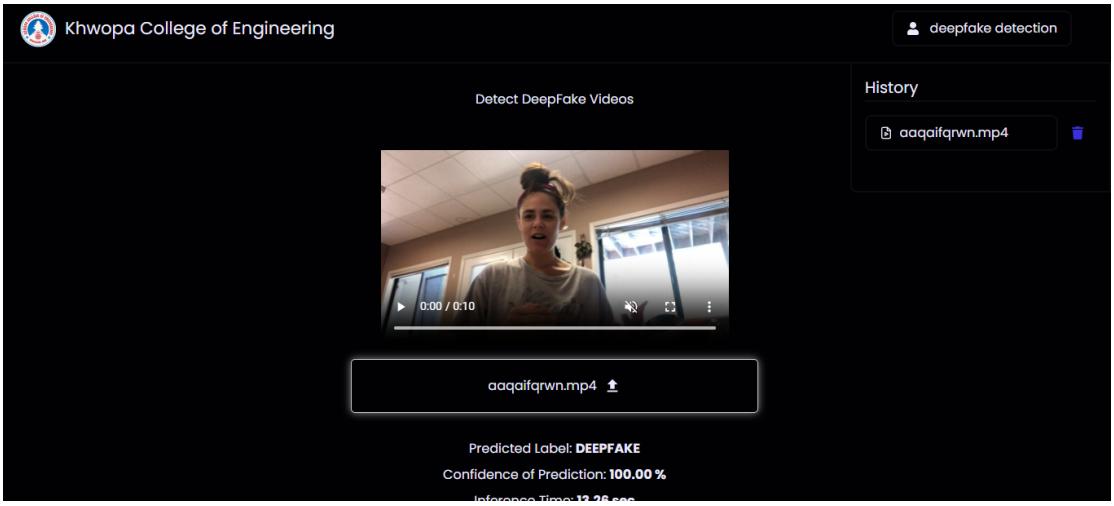


Figure 11: Integration Test 7

B.6 Gantt Chart of project