



**Faculty of Computers &
Artificial Intelligence**



Benha University

Deepfake Detection

**Departement
Scientific Computing**

Project Team

- 1- Ahmed Maher Gomaa Eraqy
- 2- Aliaa El-Sayed Mohamed Ahmed
- 3- Kariman Alaa El-Din El-Sayed Ibrahim
- 4- Marwa Ahmed Abdo Mohamed
- 5- Mayar Boghdady Ahmed Omar
- 6- Shrouq Saber Mohamed Salem

Under Supervision of

Dr. Tamer Ahmed Abassy

Benha, **JUNE 2023**

ACKNOWLEDGMENT

Firstly, the success of our project happens when we believe in ourselves, believe in doing the right thing to help other people, and the help of many people around us. Secondly, we would like to thank our supervisor, Dr. Tamer Ahmed Abassy for his encouragement, patience, guidance, and support when our project starts until we finished the project; thank you, Dr. Tamer Ahmed Abassy for helping us and for always wishing us the best. We feel that we have really been lucky to be working with someone like him.

In addition, many thanks go to our families for their inspirational thoughts, valuable guidance and assistance throughout this work and for their patience. Finally, we are very grateful to the many people who helped us with their contributions to complete this project.

DECLARATION

We hereby certify that this material, which we now submit for assessment on the program of study leading to the award of Bachelor of Computers and Artificial Intelligence in *(Deepfake Detection)* is entirely our own work, that we have exercised reasonable care to ensure that the work is original, and does not to the best of our knowledge breach any law of copyright, and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of our work.

Signed: _____

Signed: _____

Signed: _____

Signed: _____

Signed: _____

Signed: _____

Date: 14, February 2023.

Contents

1.	Abstract	7
2.	Introduction.....	7
2.1.	General Introduction and Overview	7
2.2.	Problem Definition	8
2.2.1.	Deepfake Recognition.....	8
2.2.2.	Deepfake Impact	9
2.2.3.	Deepfake Generation	9
2.2.4.	Deepfake Generation Applications	10
2.2.5.	Deepfake Detection Applications	11
2.2.6.	Deepfake Detection Methods.....	12
2.3.	Objectives	17
2.4.	Stakeholders	17
2.5.	Assumptions	18
2.6.	Scope	18
3.	Analysis and Design	19
3.1.	Interviews	19
3.2.	User and System Requirements.....	20
3.2.1.	Functional Requirements	20
3.2.2.	Non-Functional Requirements	20
3.3.	Block Diagram	21
3.4.	Sequence Diagram.....	21
3.5.	Context Diagram	22
3.6.	Data Flow Diagram (DFD)	22
3.7.	Activity Diagram.....	23
3.8.	Use Case Diagram.....	24
3.9.	State Diagram.....	25
3.10.	Used Tools and Technologies.....	26
3.10.1.	Tools	26
3.10.2.	Technologies	26
3.	Methodology	27
3.1.	Tools and Libraries.....	27
4.2.	Models.....	28

4.2.1.	InceptionResnet Model	28
4.2.2.	Densenet Model	31
4.3.	Application	33
4.4.	Connection	38
4.5.	Dataset	40
5.	Testing.....	41
5.1.	Image Testing.....	41
5.2.	Video Testing	42
6.	Conclusion	43
7.	Future Work	43
8.	References.....	44

List of Figures

Figure 1: GAN Architecture	10
Figure 2: Convolutional LSTM	13
Figure 3: Visualartefacts	14
Figure 4: Eye Blinking.....	16
Figure 5:Block Diagram	21
Figure 6: Sequence Diagram.....	21
Figure 7: Context Diagram	22
Figure 8: DFD	22
Figure 9: Activity Diagram.....	23
Figure 10: Use Case Diagram	24
Figure 11: State Diagram	25
Figure 12 : InceptionResnet architecture	28
Figure 13: model run.....	29
Figure 14: Training	29
Figure 15: Run Training.....	30
Figure 16: Train Vs. Validation for video	30
Figure 17: Denesenet Architecture	31
Figure 18: Densenet model	32
Figure 19: Training	32
Figure 20: Training Vs. Validation for image	33
Figure 21: Welcome Page	34
Figure 22: Sign Up Page	35
Figure 23: Sign in Page.....	35
Figure 24: Selecting Page	36
Figure 25: Upload image Page.....	37
Figure 26: Upload video Page.....	37
Figure 27: Firebase Architecture	38
Figure 28: JSON File	38
Figure 29: Cloud Firestore	39
Figure 30: Authentication	39
Figure 31: Remote Config	40
Figure 32: Test Fake Image	41
Figure 33: Test Real Image.....	41
Figure 34: Test Fake Video.....	42
Figure 35: Test Real Video	42

1. Abstract

Deep learning is an effective and useful technique that has been widely applied in a variety of fields, including computer vision, machine vision, and natural language processing. Deepfakes uses deep learning technology to manipulate images and videos of a person that humans cannot differentiate from the real one.

Recent studies show that deepfake videos and images have become heavily circulated through social channels. Detection of deepfake videos and images, therefore, has become increasingly critical and important. Therefore, we present this project to detect fake images and videos to help everyone who is exposed to this problem to prevent serious consequences.

2. Introduction

2.1. General Introduction and Overview

With the growth of social media, it has become very difficult to determine whether a photo/video (of any famous or political person) is real or fake, as photos and videos are manipulated with ease, but it is difficult to detect this forgery by the naked eye. Therefore, we need to determine whether the photos / videos are real or fake.

At the height of technological progress in the twenty-first century, which has become life for most people, there has been manipulation of visual and audio content that can easily deceive and has become a major risk factor for the world, so we wanted to direct even a small part of this progress to discover this manipulation and save human lives. Deepfakes (a portmanteau of "deep learning" and "fake") are synthetic media in which a person in an existing image or video is replaced with someone else's likeness. While the act of creating fake content is not new, deepfakes leverage powerful techniques from machine learning and artificial intelligence to manipulate or generate visual and audio content that can more easily deceive. The main machine learning methods used to create deepfakes are based on deep learning and involve training generative neural network architectures, such as autoencoders, or generative adversarial networks (GANs).

2.2. Problem Definition

Deep fake means that professional people using computer programs make video clips or modify an image, in which a person's face is replaced with another face created by a computer and certain programs, which often resembles another person, and the fake is convincing and very close to the truth and easy to believe. One of the well-known examples of the deep fake process is the account of Tom Cruise on the TikTok platform, as this account is only dedicated to publishing fake videos of the American actor Tom Cruise, and it has mastery in imitating the voice of the real actor and his behaviours, which made the followers convinced that he is the real Tom Cruise, in some videos , showing the fake actor doing his daily activities, such as washing his face in the morning, playing golf, or showing some simple magic tricks. Another example is the fake video of former US President Barack Obama, which was played by actor Jordan Peele. And there are also a lot of cases who have been exposed to this problem.

2.2.1. Deepfake Recognition

Through a set of tips and visual factors that can be audited by close monitoring:

1. **The movement of the eyes and eyelashes:** One of the most prominent ways for specialists to identify fake clips is the movement of the eyes and eyelashes. Slowness in this process may indicate deepfakes.
2. **Rigid facial expressions:** Another detail that can be relied upon is the observation of "stiffness" on facial movements, and its folds, especially at the corners of the mouth in scenes of smiles or laughter.
3. **Color tones:** One of the most important elements of deepfake recognition is the difficulty of integrating color tones into the human body. For example, when observing color tones between the face and neck or the face and arms, you will find a clear anomaly.
4. **Lower speed:** A popular trick that can be used to detect uneven "pixels" in photos and videos is to slow down the video or play it at a lower speed. When viewing fake clips at a slower speed, you will find that there are fine blank spaces.

5. **Teeth:** We will often find that the line of the teeth is not clear and looks “cartoon” unnatural, and without any luster like the real tooth, which often reflects light, regardless of the degree of color of the tooth.

2.2.2. Deepfake Impact

The types of deepfake risks can be categorized into:

1- **Political Risks:**

A clip of former US President "Barack Obama" spread, sparking great controversy and misleading public opinion, as well as a clip of current President Joe Biden, and this is just a simple example of the number of fake disasters that political circles can create.

2- **Moral Risks:**

Deep fake directly affects the reputation of individuals, especially celebrities: such as politicians, businessmen, art, and sports celebrities ... etc. This makes people's symptoms and reputations in the hands of unethical programmers.

2.2.3. Deepfake Generation

Generative adversarial networks (GANs) ^[1] are a form of deep neural network that has been commonly used to generate deepfake. One advantage of GANs is that it capable to learn from a set of training data set and create a sample of data with the same features and characteristics. For example, GANs can be used to swipe a “real” image or the video of a person with that of a “fake” one. ^[2]

The architecture of GANs consists of two parts: generator and discriminator.

- **The Generator:** generate new examples. The generated instances become negative(fake) training examples for the discriminator. ^[3]
- **The Discriminator:** learns to distinguish the generator's fake data from real data. The discriminator penalizes the generator for producing implausible results. ^[3]

When training begins, the generator produces obviously fake data, and the discriminator quickly learns to tell that it's fake, then the generator gets closer to producing output that can fool the discriminator. Finally, if generator training goes well, the discriminator gets worse at telling the difference between real and fake. It starts to classify fake data as real, and its accuracy decreases. [3] [8]

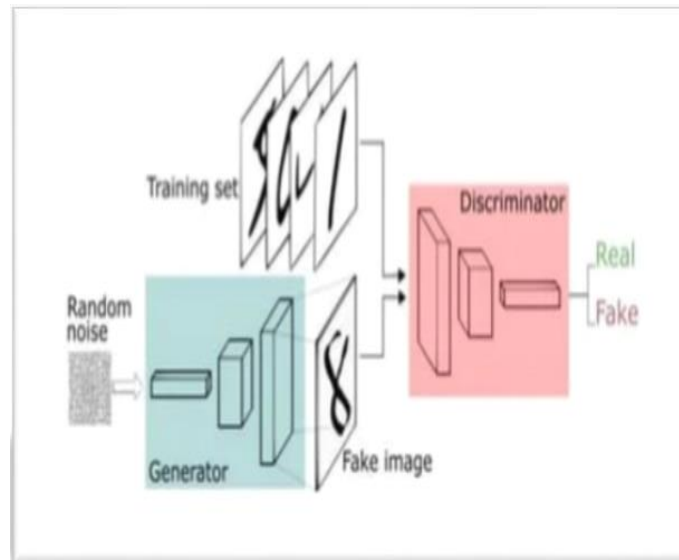


Figure 1: GAN Architecture

2.2.4. Deepfake Generation Applications

Many deepfake applications have already been around for quite a few years.

- 1- **Reface**: It was previously called Doublicat. One of its most important features is that it allows swapping faces or placing a face image in different videos. It is possible to create video files or GIFs and share them via WhatsApp, Telegram, and others. [4]
- 2- **Wombo**: An application that allows us to give movements to personal images, where the face image is uploaded, and then the system processes the image to present an animated video in a few seconds. Its AI algorithm converts still photos into fast video clips. Results may vary depending on the images uploaded. [4]

- 3- **FaceJoy**: An application that allows for the exchange of faces and also allows sex change and the integration of other types of filters. ^[4]
- 4- **FacePlay**: It contains a variety of short videos to change the face of the clip with one click. The first step: - Allow the system to scan the face and then select the video we want. The second step: - The AI technology does the rest of the work, which is creating the video clip with changing the face in a few seconds. ^[4]

2.2.5. Deepfake Detection Applications

- 1- **Sensity**: is an online platform dedicated to deepfake detection. It seeks to improve detection aspects such as face recognition, manual document check, as well as revealing media altered with deep learning and AI. ^[5]
- 2- **DeepWare Scanner** is an open-source forensic tool. ^[5]
- 3- **Microsoft Video Authenticator**: It can analyze both static images and videos in real-time mode. In the case of the video materials, it can spot blending boundary. ^[5]
- 4- **Deepfake-o-Meter** is an online deepfake detection platform. ^[5]

2.2.6. Deepfake Detection Methods

These methods fall into five categories based on the features they use. To begin with, detection based on general neural networks is commonly used in literature, where deepfake detection task is considered as regular classification tasks. Temporal consistency features are also exploited to detect discontinuities between adjacent frames of fake video.

To find distinguishable features, visual artefacts generated in blending process are exploited in detection tasks. Recently proposed approaches focused on more fundamental features, where camera fingerprint and biological signal-based schemes show great potential in detection tasks.

2.2.6.1. General-Network-based Methods

In this method, face images extracted from the detected video are used to train the detection network. Then, the trained network is applied to make predictions for all frames of this video. The predictions are finally calculated by averaging or voting strategy.

Consequently, the detection accuracy is highly dependent on the neural networks, without the need to exploit specific distinguishable features. In this section, we divide existing network-based methods into two types: ^[6]

2.2.6.1.1. Transfer Learning

Network-based detection methods should be the earliest method introduced for detection tasks. Shortly after the appearance of the first deepfake video, some early detection algorithms were proposed, mainly based on existing networks that performed well in image classification tasks. Transfer learning strategy could be easily found in the early studies. Combining steganalysis features and deep learning features.

Transfer learning is an optimization method, a shortcut to saving time or getting high performance.

There are many applications for transfer learning, as Inception, ResNet, InceptionResnet, Xception, VGG, DenseNet models.

2.2.6.1.2. Specially Designed Networks

With the advent of large-scale datasets and the development of detection algorithms, more attention is attracted. A new network is introduced to improve the performance of detection networks.

2.2.6.2. Temporal Consistency-based Methods

Time continuity is a unique feature for videos. Unlike images, video is a sequence composed of multiple frames, where adjacent frames have a strong correlation and continuity. When video frames are manipulated, the correlation between adjacent frames will be destroyed due to defects of deepfake algorithms, specifically expressed in the shift of face position and video flickering. [6][9]

2.2.6.2.1. CNN-RNN

Considering the time continuity in videos, first proposed to use RNN to detect deepfake videos. In their work, autoencoder was found to be completely unaware of previously generated faces because faces were generated frame-by-frame. This lack of temporal awareness results in multiple anomalies, which are crucial evidence for deepfake detection. To check the continuity between adjacent frames, an end-to-end trainable recurrent deepfake video detection system was proposed.

The proposed system is mainly composed of a convolutional long short-term memory (LSTM) structure for processing frame sequences. Two essential components are used in a convolutional LSTM structure, where CNN is used for frame feature extraction and LSTM is used for temporal sequence analysis. [7]

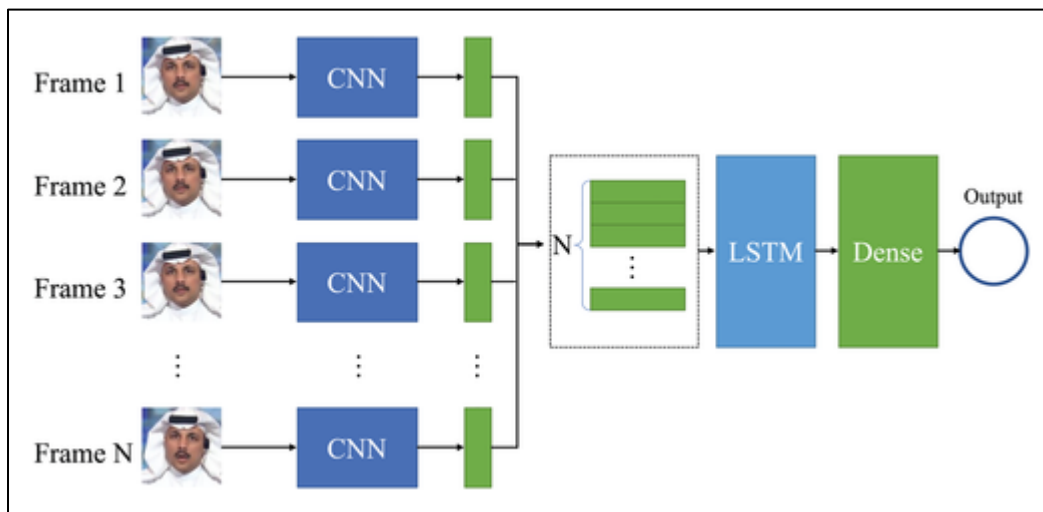


Figure 2: Convolutional LSTM

2.2.6.3. Visualartefacts-based Methods

In most existing deepfake methods, the generated face has to be blended into an existing background image, causing intrinsic image discrepancies on the blending boundaries. faces and background images come from different source images, giving rise to the abnormal behavior of the synthetic image, such as boundary anomaly and inconsistent brightness. These visual artefacts make deepfake videos fundamentally detectable. In this section, three main visual artefacts would be introduced. [6] [9]



Figure 3: Visualartefacts

2.2.6.3.1. Face warping artefacts

Based on the observations with inconsistency between faces and background, a new deep learning-based method. Face warping artefacts generated by blending process were used to detect fake videos.

In this case, there would be an obvious color difference and resolution inconsistency between the internal face and background areas. Since the purpose here is to detect inconsistency between the face region and background area, the negative samples are generated by a simplified process, where the face undergoes an affine warp back to the source image directly after smoothed.

2.2.6.3.2. Blending boundary

proposed a novel image representation, namely face X-ray, which was exploited to observe whether the input image can be decomposed into the foreground face and the background. Specifically, the blending boundary between the foreground manipulated face and the background was defined as face X-ray.

2.2.6.3.3. Head pose inconsistency

Observing that deepfake videos were created by splicing a synthesized face into the original image, Yang et al. proposed a new detection method based on 3D head poses. They argued that current generative neural networks could not guarantee landmark matching, causing that estimated 3D landmarks on the face-manipulated area were different from 3D landmarks estimated from the whole face area. In this method, the rotation matrix estimated using facial landmarks from the whole face and the one estimated using only landmarks in the central region are calculated to analyze the similarity between two pose vectors.

2.2.6.4. Camera-fingerprints-based methods

Camera fingerprints are a kind of noise with very weak energy, which plays an important role in forensic fields, especially source identification tasks. In general, camera fingerprints-based approaches have gone through three processes: the photo response nonuniformity (PRNU) patterns, noise print and recent video noise pattern.^[6]

2.2.6.5. Biological-Signals-based Methods

Detection based on biological signals is an interesting scheme that emerged in recent years. The core observation is that even though GAN is able to generate faces with high realism, the naturally hidden biological signals are still not easily replicate, making it difficult to synthesize human faces with reasonable behavior. Taking advantage of this abnormal behavior, several studies have been proposed. In this section, we will introduce two approaches based on biological signals: blinking frequency-based and heart rate-based detection approaches.

2.2.6.5.1. Heart Rate

A large difference was shown between original videos and deepfake videos when heart rate prediction was performed separately. However, this work only performed heart rate prediction of deepfake videos while lacked further experiments of deepfake detection. A large number of works have been carried out based on biological signals. A recently proposed approach, named Deep-Rhythm, utilized a dual-spatial-temporal attention mechanism to monitor the heartbeat rhythms, proving to generalize well over different datasets. Likewise, Deep-FakesON-Phys predicts the heart rate through changes in skin color, thereby considering the detection of deepfake videos.^[6]

2.2.6.5.2. Eye Blinking

Abnormalities with blink frequency were earlier identified as discriminable features in deepfake detection. This could be attributed to the fact that deepfake algorithms train models using a large number of face images obtained online. Most of the images show people with their eyes open, causing that a closed-eye view is difficult to generate in a manipulated video. Based on this finding, a deep neural network model, known as long-term recurrent CNN (LRCN), was introduced to distinguish open and close eye states.

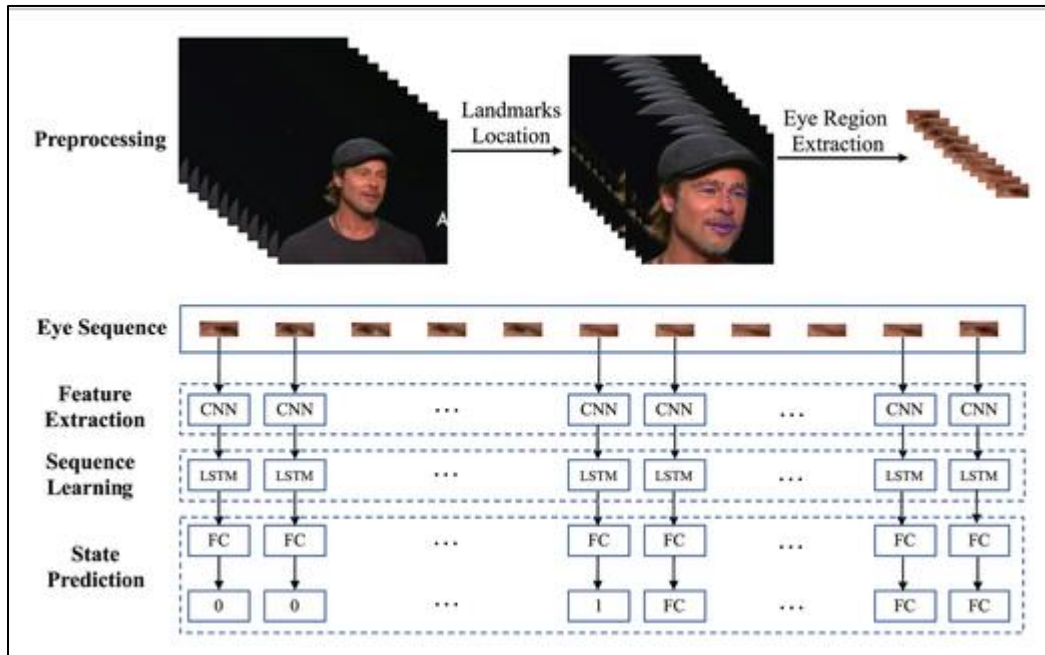


Figure 4: Eye Blinking

2.3. Objectives

There are many tools available for creating the deep fakes, but for deep fake detection there is hardly any tool available. Our approach for detecting the deep fakes will be great contribution in avoiding the percolation of the deep fakes over the world wide web. We will provide this app for the user to upload the image/video and classify it as fake or real. Even big applications like WhatsApp, Facebook can integrate this project with their application for easy pre-detection of deep fakes before sending to another user.

Deepfake technology may be used in many criminal acts such as:

1. Creating photos/videos of celebrities and politicians that may be satirical, spread false news, or show part of their daily lives.
2. Creating clips that threaten security in countries, such as a video about warning of a terrorist attack.
3. spread fake news through social media.
carry out fraud and theft of money.
4. Putting someone's face on the bodies of porn actors.

So, our main goal is to make an application to help and protect everyone who is exposed to these problems, and thus prevent any major consequences from occurring.

2.4. Stakeholders

- Anyone can use this app to check if the photo/video is fake or not.
- Applications' Programmers

2.5. Assumptions

Only name, email, phone number, and password are allowed to enter the system.
Android & web developers.
Providing the internet to operate and update the system.
The expected project's timeline can be met, and the project will complete within the expected time.
The data have been automatically updated in the data inventory and all information in the data inventory is synchronized like User's names, Phone numbers & E-mails.

2.6. Scope

Project Includes
The system requests to sign up/login in the 1st time.
System's functionality is to detect whether the image is fake or not.
The system will request to login once.
The system is available 24 hours.
User's profile functionality.

Project Excludes
Users fill in feedback form.
The system will request to login every time.

3. Analysis and Design

3.1. Interviews

1. Could the user be able to Create an account (with email or phone number) to log into the system?
 - Yes, the user could log into the system by his Gmail and phone number, and everyone should have a password to secure his information.
2. What are things that can be checked if they are fake or not?
 - Users can check if the photo/video is fake or not.
3. Who can use this system?
 - Anyone can use this app to check if the photo/video is fake or not.
4. To what range (areas) the system will be available in?
 - The system will be available in middle east, because their knowledge of this field is little, and therefore if someone is exposed to this problem, many serious consequences will occur.
5. On what platforms the system should be supported?
 - The system should be supported on mac, windows and Linux.
6. Does the photo/video quality have to be HD in order to be checked using this system?
 - No, the system can check any photo/video.
7. Do you have any notes or recommendations for the system design?
 - The GUI forms design must be simple, the number of colors on a single form must not exceed 5 colors, and the system allows the users to upload photo/video then check if fake or not.

3.2. User and System Requirements

3.2.1. Functional Requirements

1. **Register & Login:** The system must allow the user to register and login.
2. **Checking the image:** The system must check if the image is real or fake.
3. **Database:** The system database must contain all related info about users.
4. **Result:** The system must show the output of checking the image (Real/Fake).

3.2.2. Non-Functional Requirements

1. **Security:** Users' info should be secure.
2. **Appearance:** Simple look and good feel to use.
3. **Availability:** The system must be available all time.
4. **Response:** The system must be response to user in less than 1 second.

3.3. Block Diagram

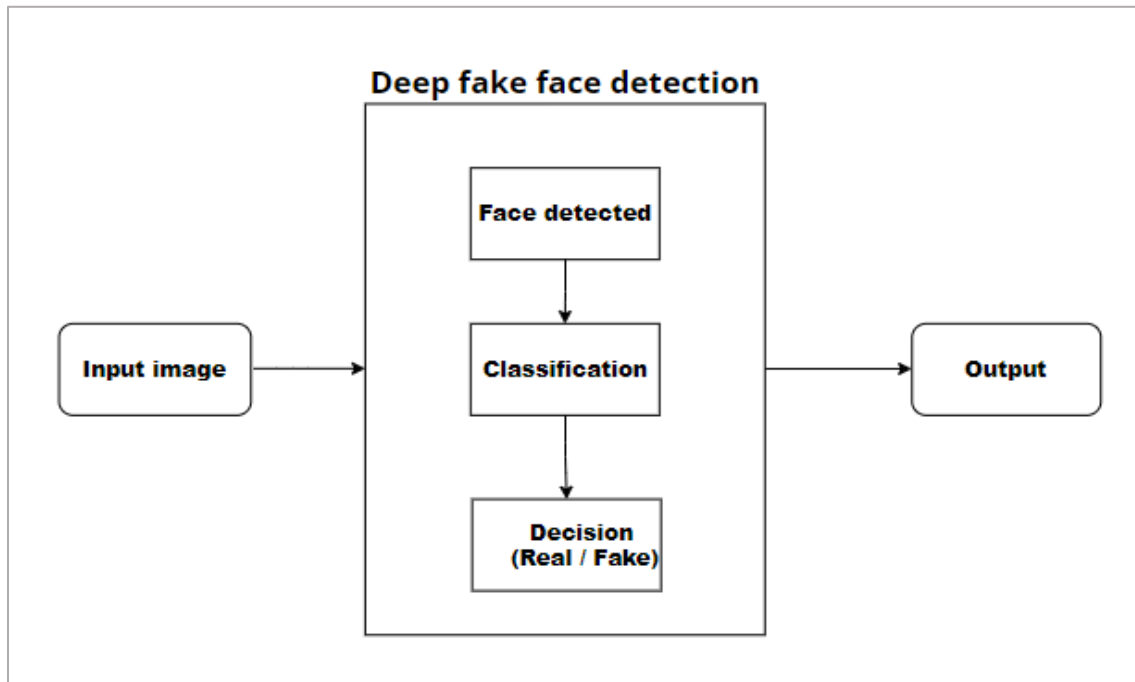


Figure 5:Block Diagram

3.4. Sequence Diagram

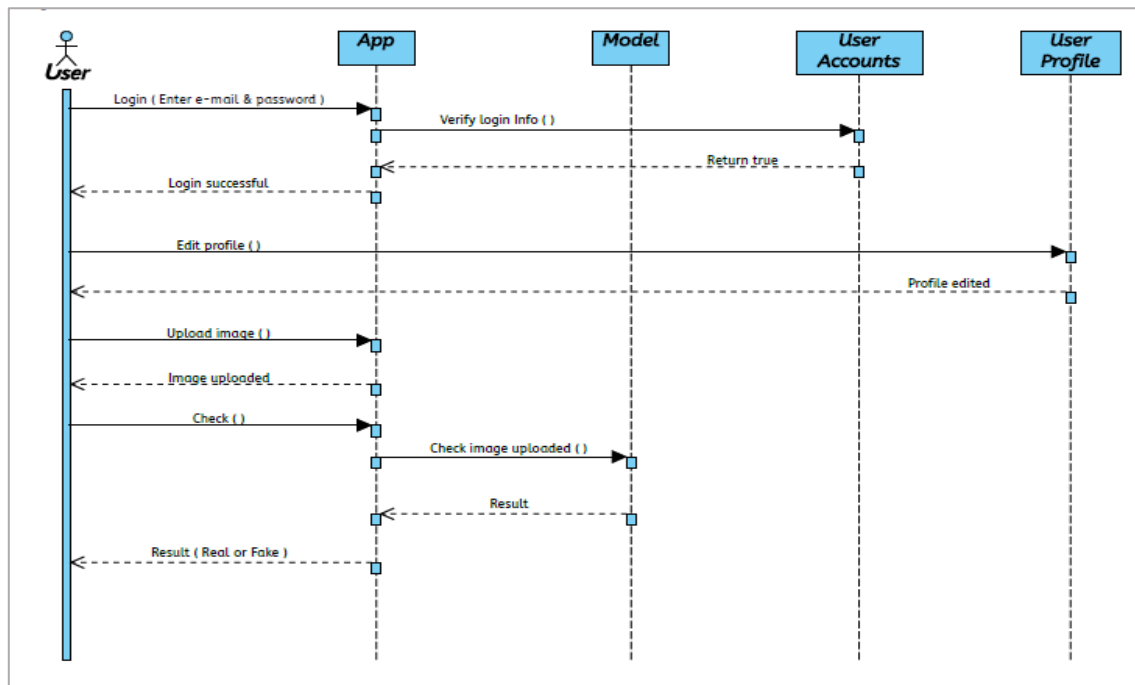


Figure 6: Sequence Diagram

3.5. Context Diagram

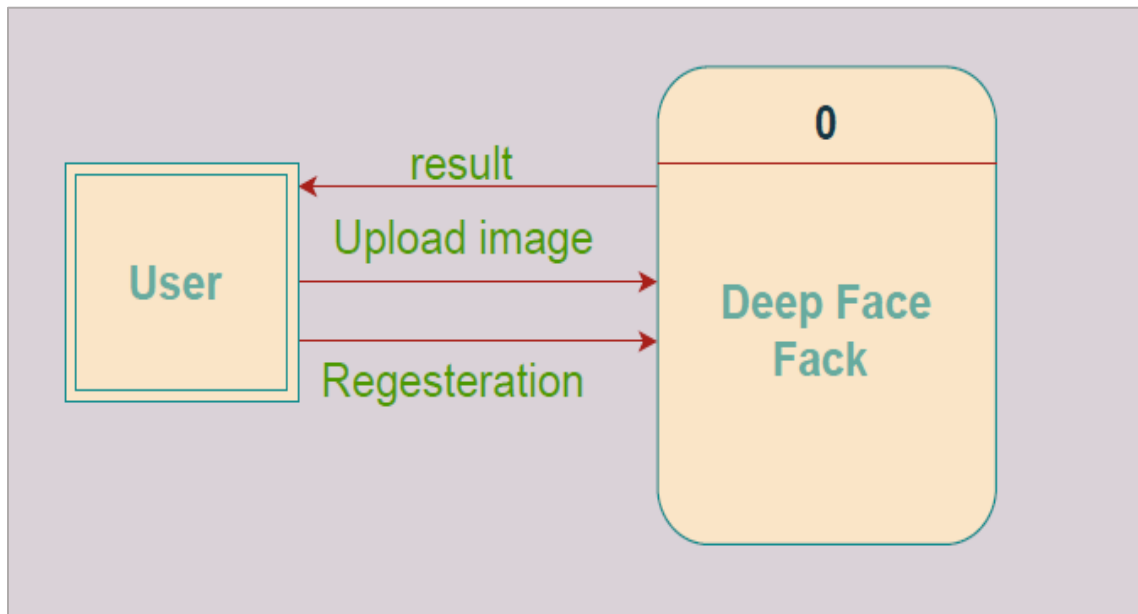


Figure 7: Context Diagram

3.6. Data Flow Diagram (DFD)

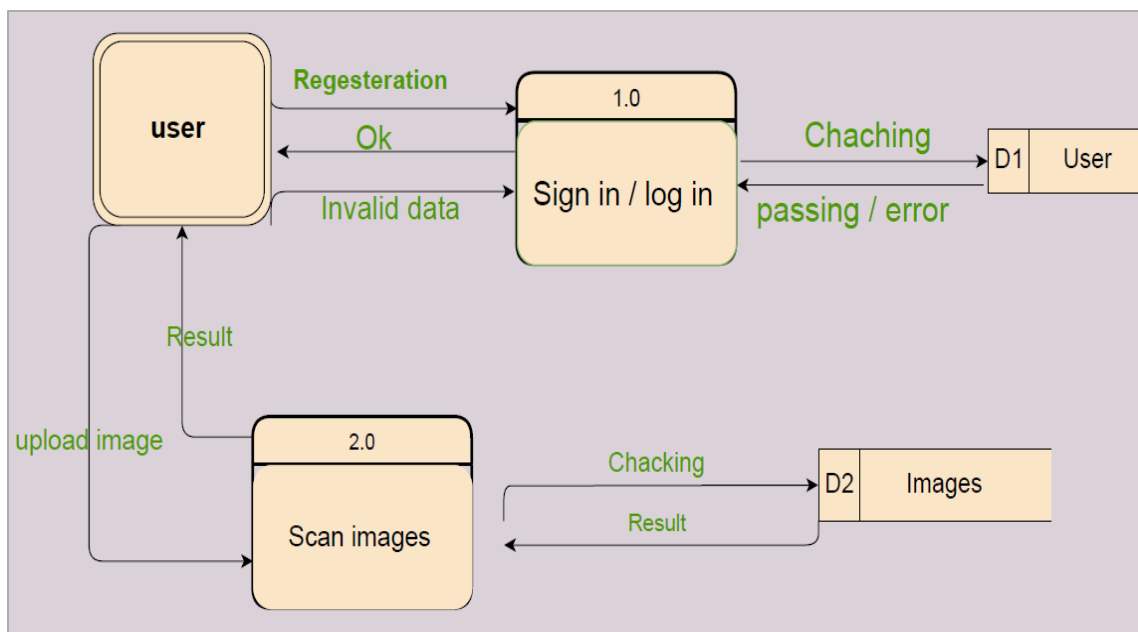


Figure 8: DFD

3.7. Activity Diagram

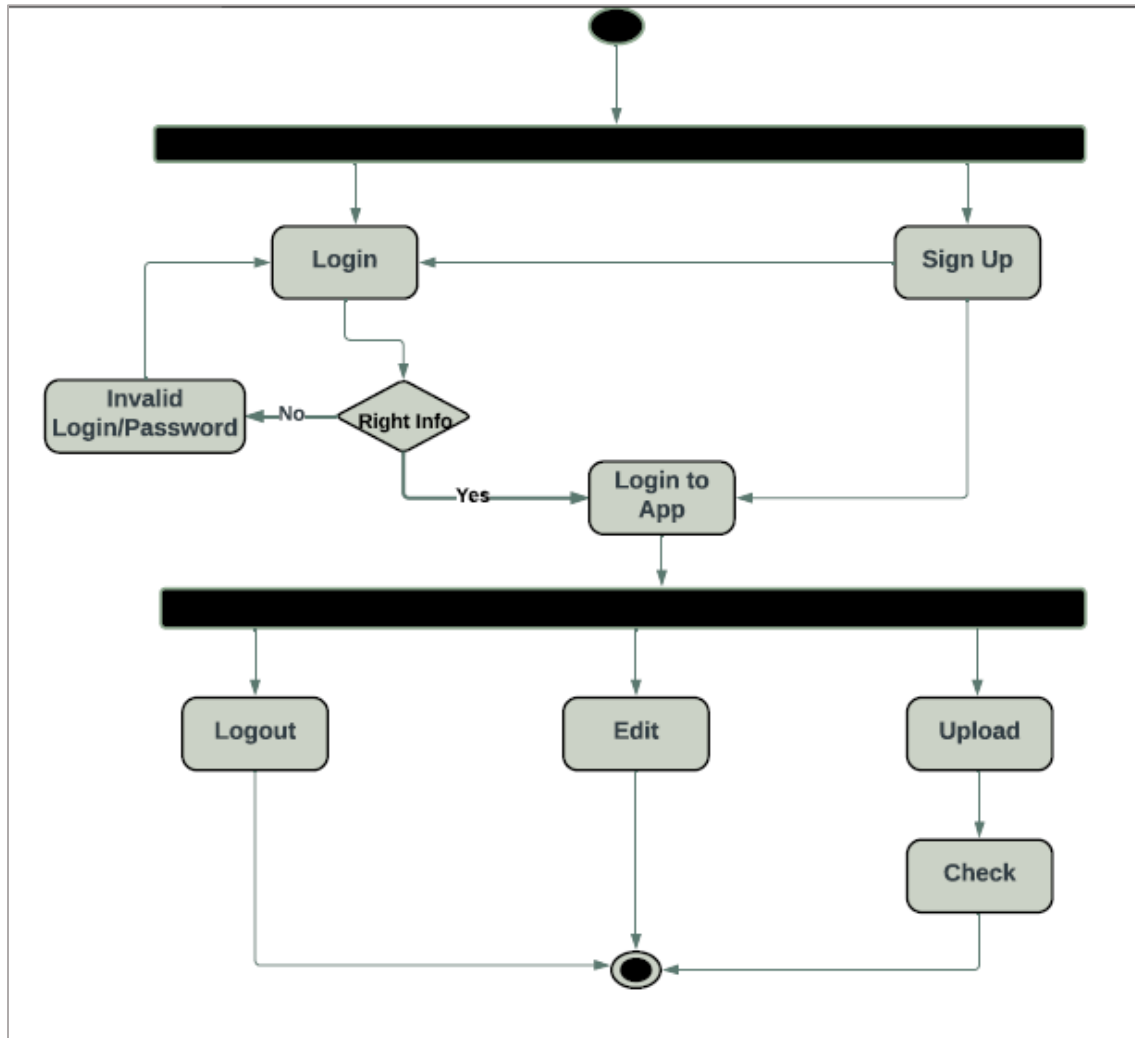


Figure 9: Activity Diagram

3.8. Use Case Diagram

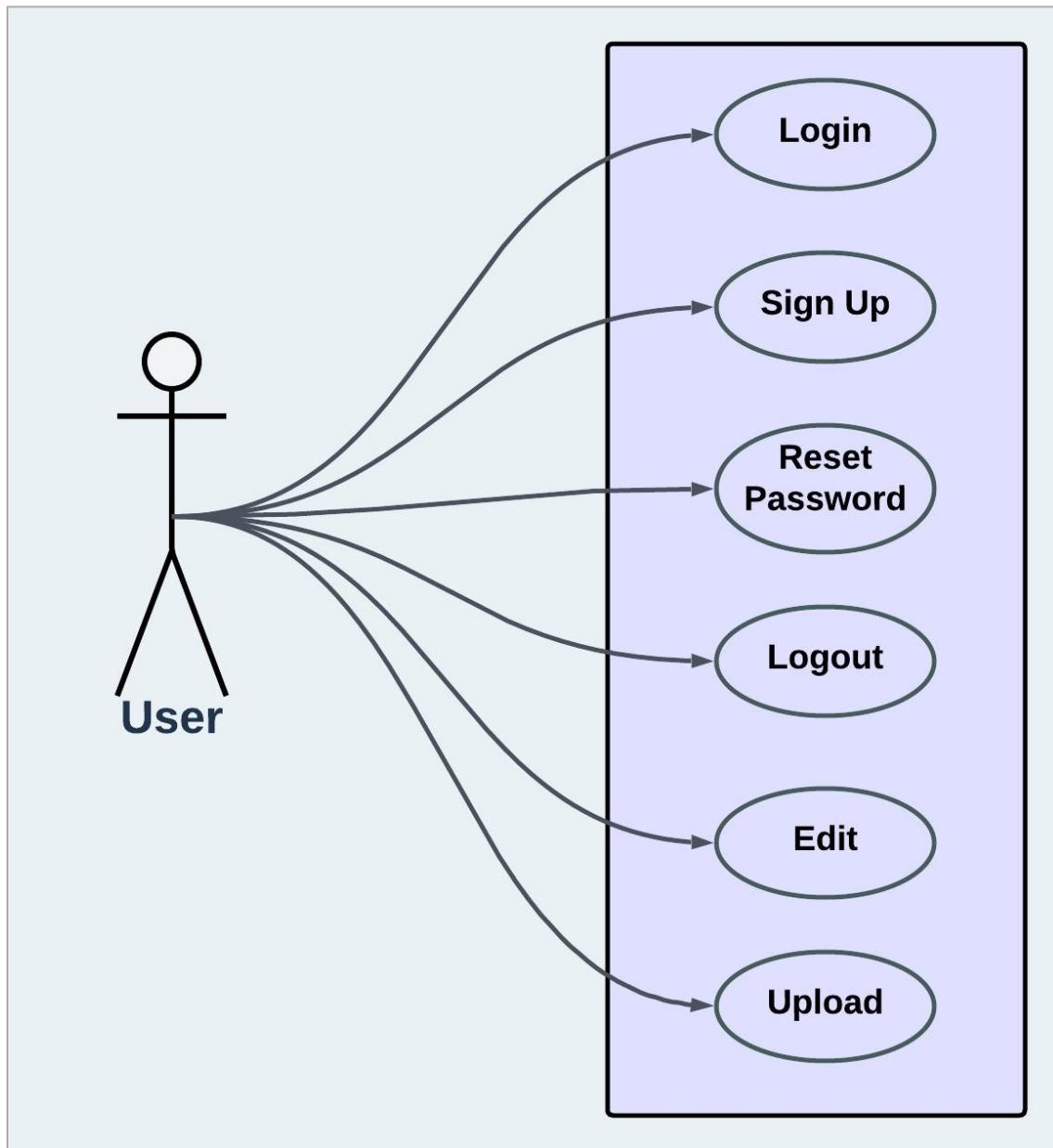


Figure 10: Use Case Diagram

3.9. State Diagram

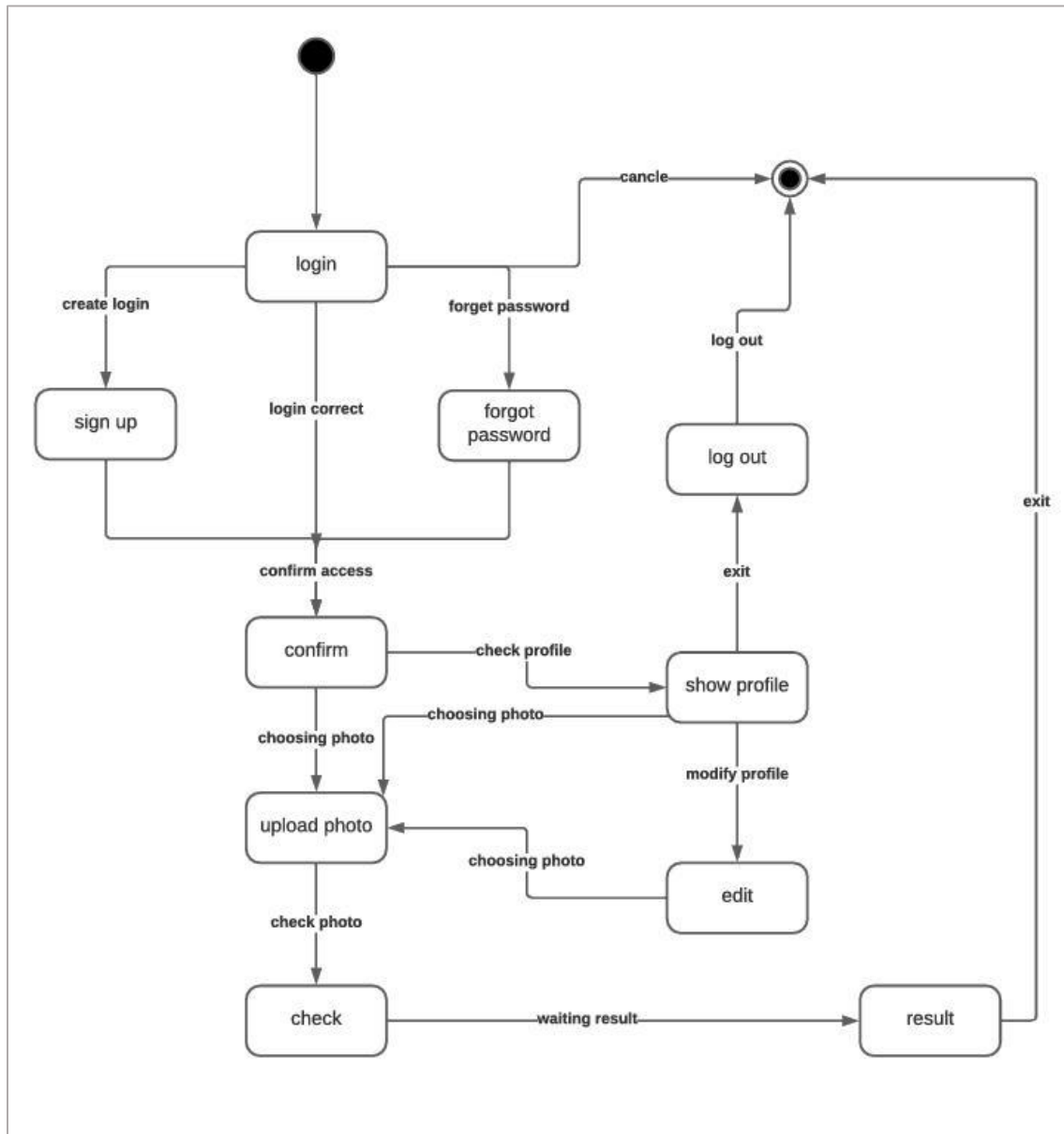


Figure 11: State Diagram

3.10. Used Tools and Technologies

3.10.1. Tools

1. XD
2. Visual Studio Code
3. Android Studio
4. Kaggle
5. Jupyter
6. Google Colab
7. Firebase

3.10.2. Technologies

1. Flutter
2. Transfer Learning (CNN models)

3. Methodology

3.1. Tools and Libraries

3.1.1. CV2

OpenCV is one of the most popular (open source) computer vision libraries. And now it plays a major role in real-time operations, which is very important in today's systems. We use this library for reading images and videos, extracting the Region of Interest (ROI), and do operations on images. In our project we use it to take **one frame per second**.

4.1.2. Dlib

Dlib is one of the most powerful and easy-to-go open-source libraries. Dlib is mostly used for face recognition purposes. They analyzed the object/face using the functions called HOG (Histogram of oriented gradients) and CNN (Convolutional Neural Networks).

4.1.3. OS

The OS module in Python provides functions for interacting with the operating system. This module provides a portable way of using operating system-dependent functionality. The “**os**” and “**os.path**” modules include many functions to interact with the file system (creating, removing, and copying files and reading data from files).

4.1.4. Matplotlib

Matplotlib is an open-source library, and we can use it freely. It is a low-level graph plotting library in python that serves as a visualization utility.

4.1.5. TensorFlow

TensorFlow is an end-to-end open-source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries, and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML-powered applications.

4.1.6. Keras

Keras is the high-level API of the TensorFlow platform: an approachable, highly productive interface for solving machine learning problems, with a focus on modern deep learning. It provides essential abstractions and building blocks for developing and shipping machine learning solutions with high iteration velocity.

4.2. Models

In our project we use General-Network-based methods (Transfer Learning)

4.2.1. InceptionResnet Model

Inception-ResNet-v2 is a convolutional neural architecture that builds on the Inception family of architectures but incorporates residual connections (replacing the filter concatenation stage of the Inception architecture)

Inception-ResNet-v2 is a convolutional neural network that is trained on more than a million images from the ImageNet database. The network is 164 layers deep and can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. As a result, the network has learned rich feature representations for a wide range of images. The network has an image input size of 299-by-299. [10]

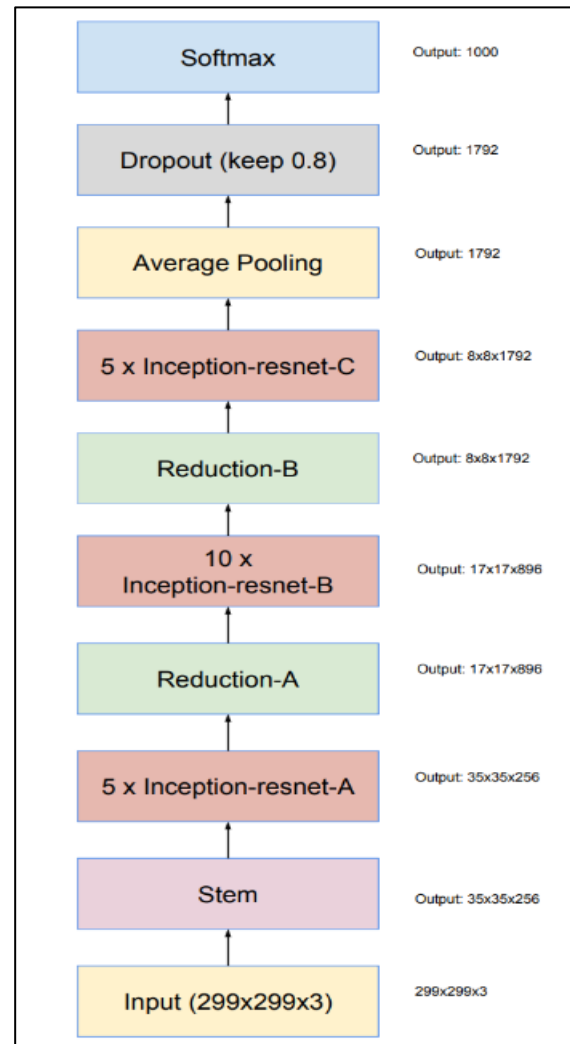


Figure 12 : InceptionResnet architecture

we use InceptionResnet model for video processing, and we resize the input image to 128*128 image size, it classifies images into 2 categories (**Real or Fake**) instead of 1000 categories.

```
model = InceptionResNetV2(  
    weights='imagenet',  
    include_top=False,  
    input_shape=(128,128,3)  
)  
model = build_model(densenet)  
model.summary()
```

Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/inception_resnet_v2/inception_resnet_v2_219055592/219055592 [=====] - 1s 0us/step
Model: "sequential"

Layer (type)	Output Shape	Param #
inception_resnet_v2 (Functional)	(None, 2, 2, 1536)	54336736
global_average_pooling2d (GlobalAveragePooling2D)	(None, 1536)	0
dense (Dense)	(None, 1)	1537

=====
Total params: 54,338,273
Trainable params: 54,277,729
Non-trainable params: 60,544

Figure 13: model run

We train this model within 20 epochs. The final accuracy is 98% for training and 95% for validation.

```
train_steps = 31228//64  
valid_steps = 2000 //64  
  
history = model.fit(  
    train_flow,  
    epochs = 20,  
    steps_per_epoch =train_steps,  
    validation_data =valid_flow,  
    validation_steps = valid_steps  
)
```

Figure 14: Training

```

Epoch 1/20
487/487 [=====] - 213s 248ms/step - loss: 0.3913 - accuracy: 0.8254 - val_loss: 0.5305 - val_accuracy: 0.8770
Epoch 2/20
487/487 [=====] - 121s 248ms/step - loss: 0.1937 - accuracy: 0.9213 - val_loss: 0.9682 - val_accuracy: 0.6870
Epoch 3/20
487/487 [=====] - 117s 240ms/step - loss: 0.1288 - accuracy: 0.9469 - val_loss: 0.1761 - val_accuracy: 0.9294
Epoch 4/20
487/487 [=====] - 116s 238ms/step - loss: 0.1109 - accuracy: 0.9526 - val_loss: 0.1901 - val_accuracy: 0.9209
Epoch 5/20
487/487 [=====] - 118s 243ms/step - loss: 0.0909 - accuracy: 0.9613 - val_loss: 0.2568 - val_accuracy: 0.9098
Epoch 6/20
487/487 [=====] - 119s 244ms/step - loss: 0.0847 - accuracy: 0.9635 - val_loss: 0.3584 - val_accuracy: 0.9249
Epoch 7/20
487/487 [=====] - 117s 241ms/step - loss: 0.0706 - accuracy: 0.9681 - val_loss: 0.4271 - val_accuracy: 0.8780
Epoch 8/20
487/487 [=====] - 118s 243ms/step - loss: 0.0614 - accuracy: 0.9732 - val_loss: 0.4517 - val_accuracy: 0.8765
Epoch 9/20
487/487 [=====] - 119s 244ms/step - loss: 0.0665 - accuracy: 0.9709 - val_loss: 0.1619 - val_accuracy: 0.9425
Epoch 10/20
487/487 [=====] - 119s 243ms/step - loss: 0.0587 - accuracy: 0.9741 - val_loss: 0.5983 - val_accuracy: 0.9062
Epoch 11/20
487/487 [=====] - 116s 239ms/step - loss: 0.0565 - accuracy: 0.9745 - val_loss: 0.4060 - val_accuracy: 0.8740
Epoch 12/20
487/487 [=====] - 118s 241ms/step - loss: 0.0517 - accuracy: 0.9761 - val_loss: 0.1848 - val_accuracy: 0.9405
Epoch 13/20
...
Epoch 19/20
487/487 [=====] - 117s 239ms/step - loss: 0.0327 - accuracy: 0.9849 - val_loss: 0.2222 - val_accuracy: 0.9168
Epoch 20/20
487/487 [=====] - 118s 242ms/step - loss: 0.0369 - accuracy: 0.9840 - val_loss: 0.1586 - val_accuracy: 0.9491

```

Figure 15: Run Training

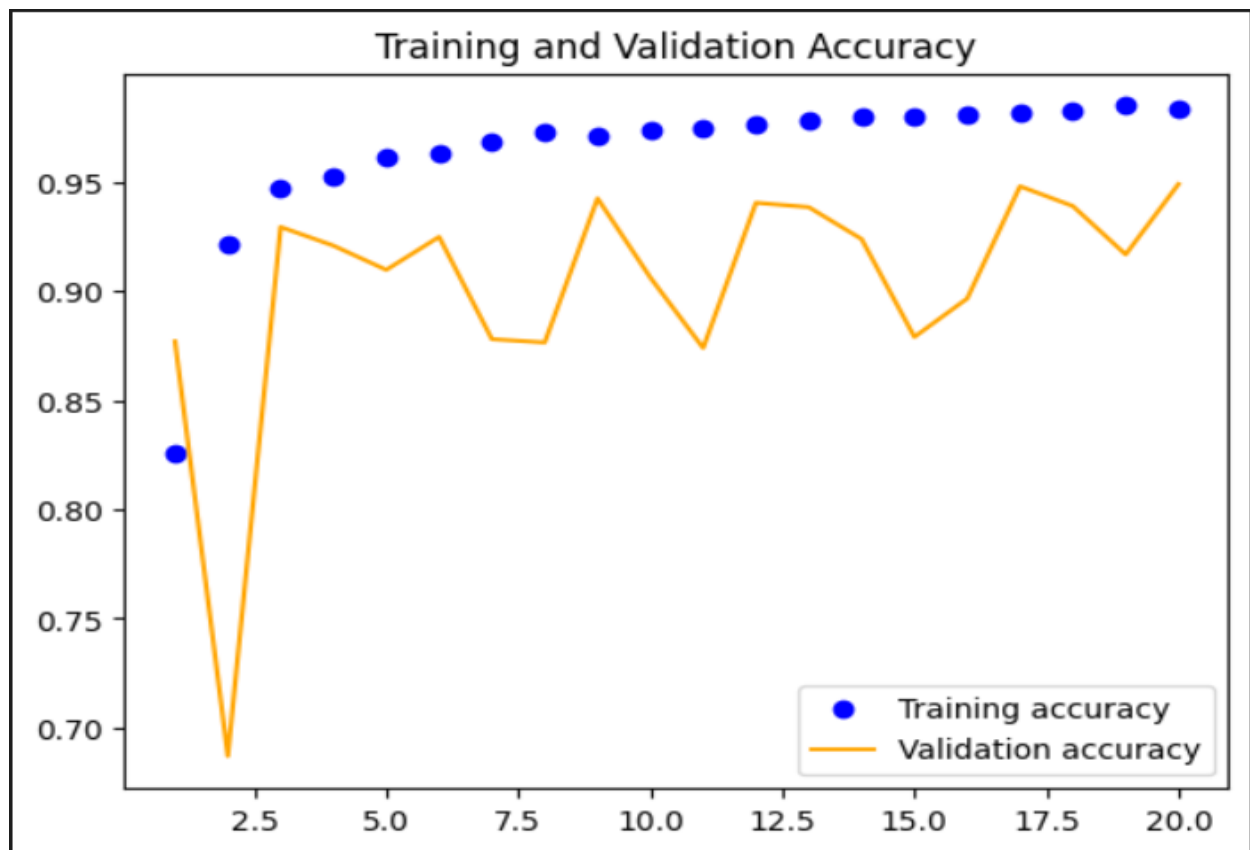


Figure 16: Train Vs. Validation for video

4.2.2. Densenet Model

In a traditional feed-forward Convolutional Neural Network (CNN), each convolutional layer except the first one (which takes in the input), receives the output of the previous convolutional layer, and produces an output feature map that is then passed on to the next convolutional layer. Therefore, for 'L' layers, there are 'L' direct connections; one between each layer and the next layer.^[11]

However, as the number of layers in the CNN increases, the 'vanishing gradient' problem arises. This means that as the path for information from the input to the output layers increases, it can cause certain information to 'vanish' or get lost which reduces the ability of the network to train effectively.^[11]

Densenet resolves this problem by modifying the standard CNN architecture and simplifying the connectivity pattern between layers. In a Densenet architecture, each layer is connected directly with every other layer, hence the name Densely Connected Convolutional Network. For 'L' layers, there are $L(L+1)/2$ direct connections.^[11]

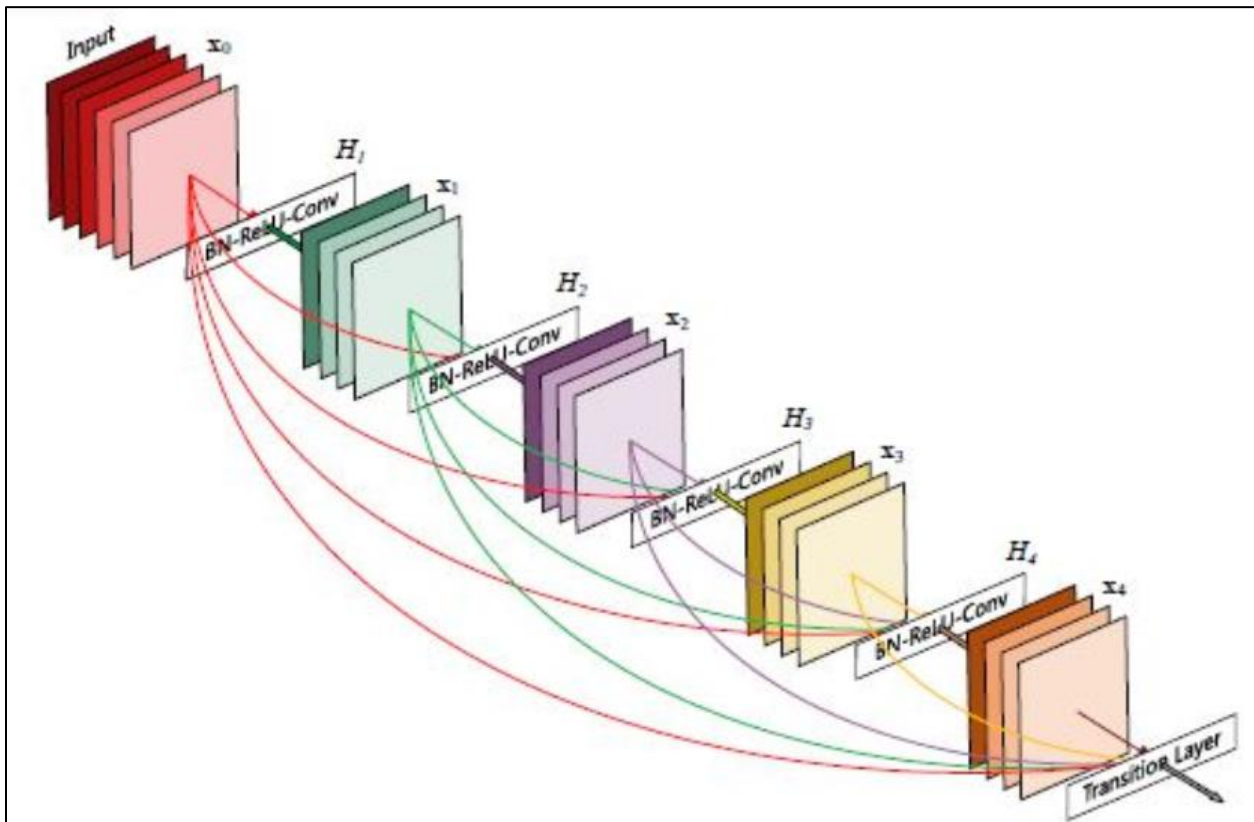


Figure 17: Denseset Architecture

It consists of 1 7*7 convolution layers, 58 3*3 convolution layers, 61 1*1 convolution layers, 4 AvgPool layers, 1 Fully Connected layer.

We use this model for image processing. The size of input image to model is 224*224.

```
densenet = DenseNet121( weights=None, include_top=False, input_shape=(224,224,1) )

model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
densenet121 (Functional)	(None, 7, 7, 1024)	7031232
global_average_pooling2d (Gl	(None, 1024)	0
dense (Dense)	(None, 1)	1025

Total params: 7,032,257
 Trainable params: 6,948,609
 Non-trainable params: 83,648

Figure 18: Densenet model

We train this model within 10 epochs. The final accuracy is 99% for training and 97.5% for validation.

```
history = model.fit(
    train_generator,
    steps_per_epoch = (100000//100),
    validation_data = valid_generator,
    validation_steps = (20000//100),
    epochs = 10
)
```

Epoch 1/10
 1000/1000 [=====] - 950s 932ms/step - loss: 0.4942 - accuracy: 0.7545 - val_loss: 0.7210 - val_accuracy: 0.6277
 Epoch 2/10
 1000/1000 [=====] - 557s 556ms/step - loss: 0.2515 - accuracy: 0.8950 - val_loss: 0.3341 - val_accuracy: 0.8543
 Epoch 3/10
 1000/1000 [=====] - 555s 555ms/step - loss: 0.1338 - accuracy: 0.9475 - val_loss: 0.3512 - val_accuracy: 0.8537
 Epoch 4/10
 1000/1000 [=====] - 556s 556ms/step - loss: 0.0820 - accuracy: 0.9685 - val_loss: 1.2437 - val_accuracy: 0.6237
 Epoch 5/10
 1000/1000 [=====] - 555s 555ms/step - loss: 0.0559 - accuracy: 0.9792 - val_loss: 0.3351 - val_accuracy: 0.8924
 Epoch 6/10
 1000/1000 [=====] - 556s 556ms/step - loss: 0.0401 - accuracy: 0.9851 - val_loss: 0.1160 - val_accuracy: 0.9567
 Epoch 7/10
 1000/1000 [=====] - 555s 555ms/step - loss: 0.0330 - accuracy: 0.9872 - val_loss: 0.1173 - val_accuracy: 0.9561
 Epoch 8/10
 1000/1000 [=====] - 558s 558ms/step - loss: 0.0269 - accuracy: 0.9904 - val_loss: 0.1541 - val_accuracy: 0.9487
 Epoch 9/10
 1000/1000 [=====] - 557s 557ms/step - loss: 0.0235 - accuracy: 0.9917 - val_loss: 0.0991 - val_accuracy: 0.9666
 Epoch 10/10
 1000/1000 [=====] - 557s 557ms/step - loss: 0.0195 - accuracy: 0.9930 - val_loss: 0.0737 - val_accuracy: 0.9750

Figure 19: Training

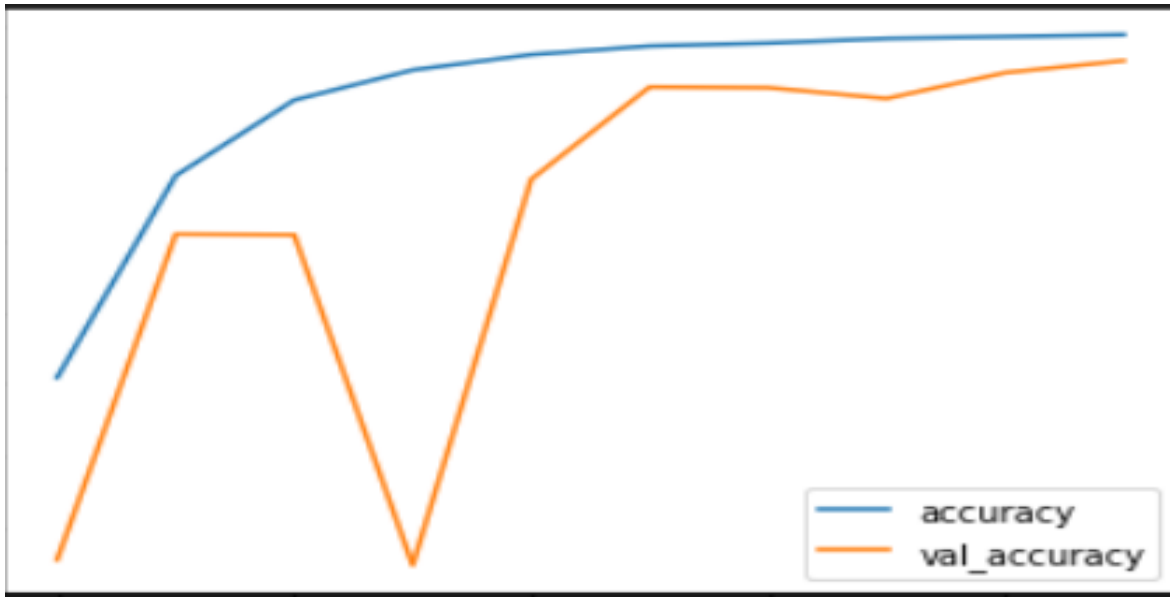


Figure 20: Training Vs. Validation for image

4.3. Application

Flutter is an open-source UI software development kit created by Google. It is used to develop cross-platform applications for Android, iOS, Linux, macOS, Windows, and the web from a single codebase.

Flutter consists of two important parts: ^[12]

- **SDK (Software Development Kit):** A collection of tools that are going to help us develop our applications. This includes tools to compile our code into native machine code (code for iOS and Android).
- **Framework (UI Library based on widgets):** A collection of reusable UI elements (buttons, text inputs, sliders, and so on) that we can personalize for our own needs.

To develop with Flutter, we will use a programming language called Dart. Dart focuses on front-end development, and we can use it to create mobile and web applications.

Flutter is simple to use, compiles applications (updates) quickly, and has a good documentation, so we develop our application with flutter.

Our application consists of 6 main pages. Firstly, a **welcome page** where users can sign up or sign in, Once choosing one of the two previous options, a new page opens for each of them, for the user to register his data.

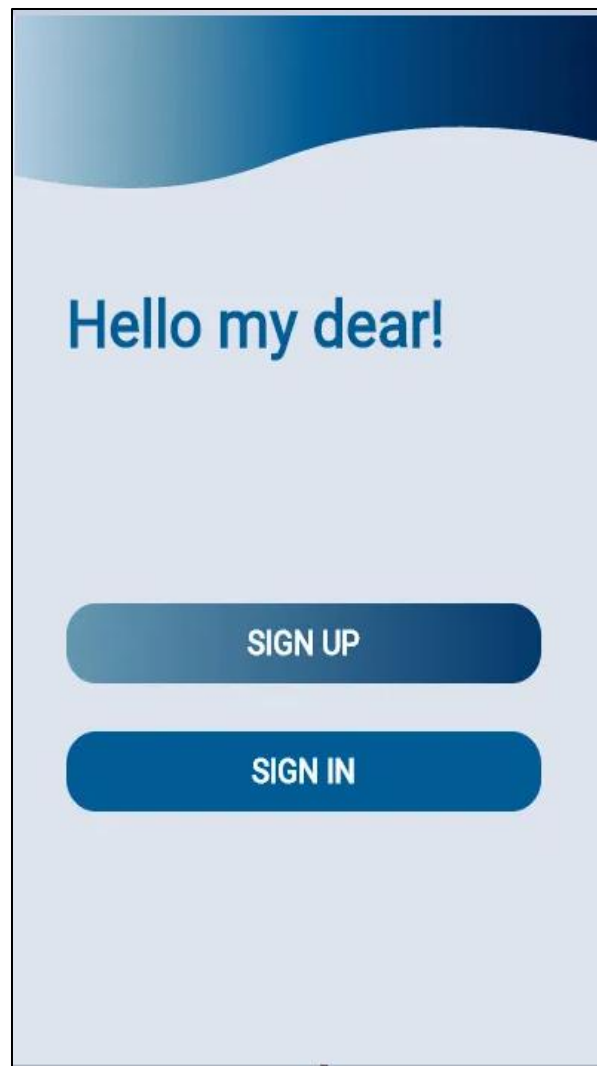
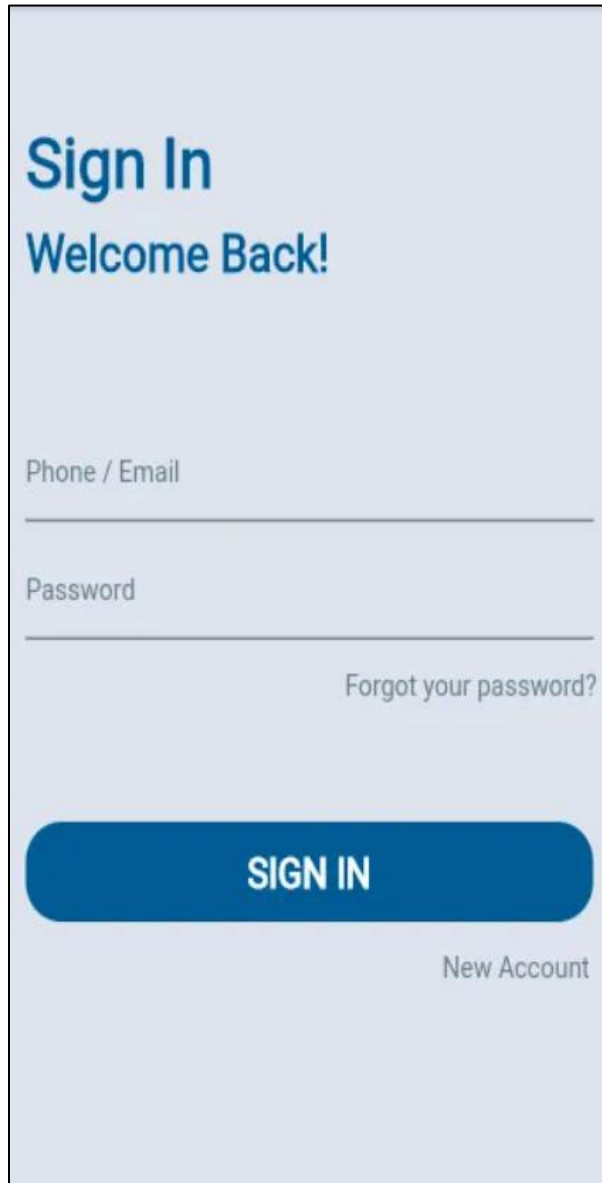


Figure 21: Welcome Page

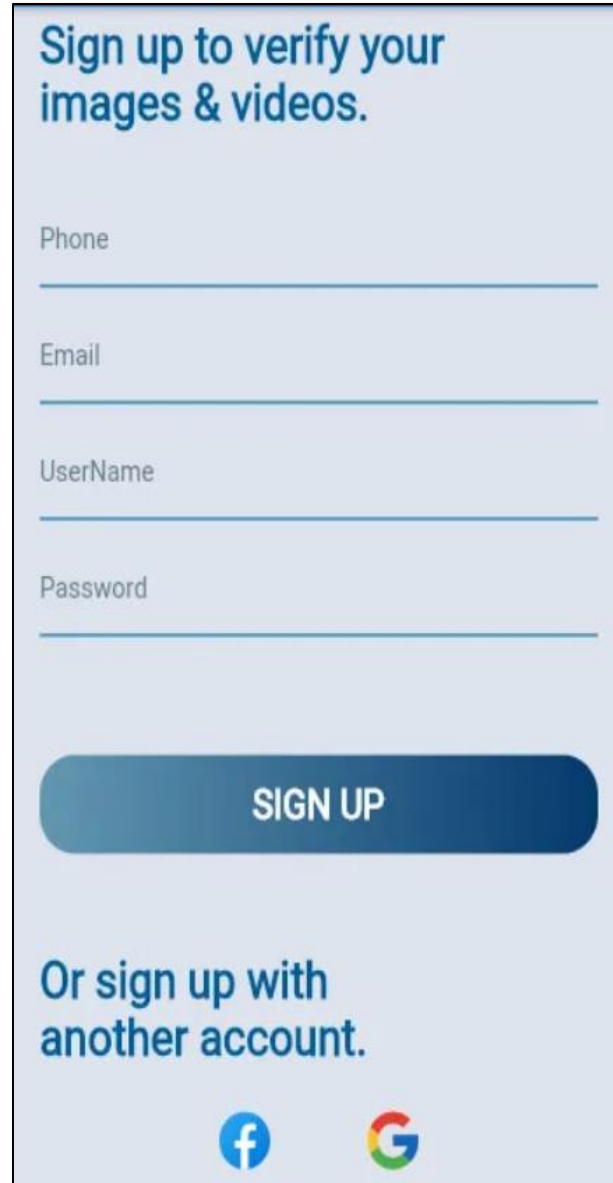
If the user does not have any account, then select **sign up** option and enter his phone number, email, username, and password. The user can sign up by using his Facebook account or Gmail account also.

If the user already has an account, then he selects **sign in** option and enter his phone number or email and password.



The Sign In page features a light blue background. At the top, the text "Sign In" is in a large, bold, dark blue font, followed by "Welcome Back!" in a slightly smaller, bold, dark blue font. Below this, there are two input fields: "Phone / Email" and "Password", both with light blue borders and placeholder text. A link "Forgot your password?" is positioned to the right of the Password field. At the bottom, there is a large, rounded, dark blue button with the text "SIGN IN" in white, and a link "New Account" in a smaller, dark blue font to its right.

Figure 23: Sign in Page



The Sign Up page features a light blue background. At the top, the text "Sign up to verify your images & videos." is in a bold, dark blue font. Below this, there are four input fields: "Phone", "Email", "UserName", and "Password", all with light blue borders and placeholder text. A large, rounded, dark blue button with the text "SIGN UP" in white is centered below the input fields. At the bottom, the text "Or sign up with another account." is in a bold, dark blue font, followed by two social media icons: Facebook (blue circle with white 'f') and Google (multi-colored 'G').

Figure 22: Sign Up Page

After signing in or signing up, users need to upload photos or videos to check if they are fake or real, so they have two options **Image** or **Video**.

Once the user chooses one of the two options, a page will be opened to upload the image or video he wants.

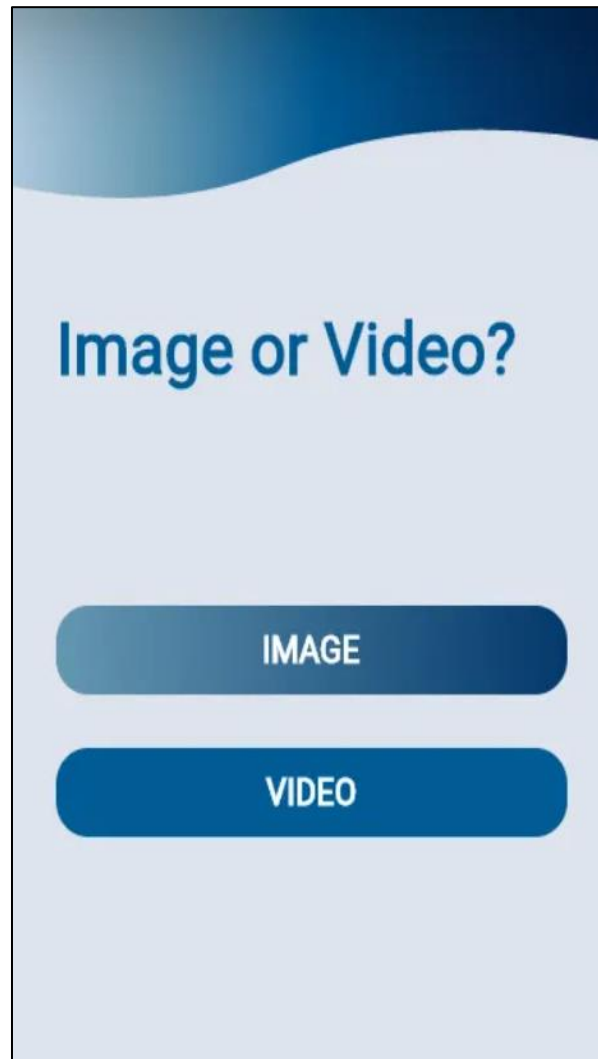
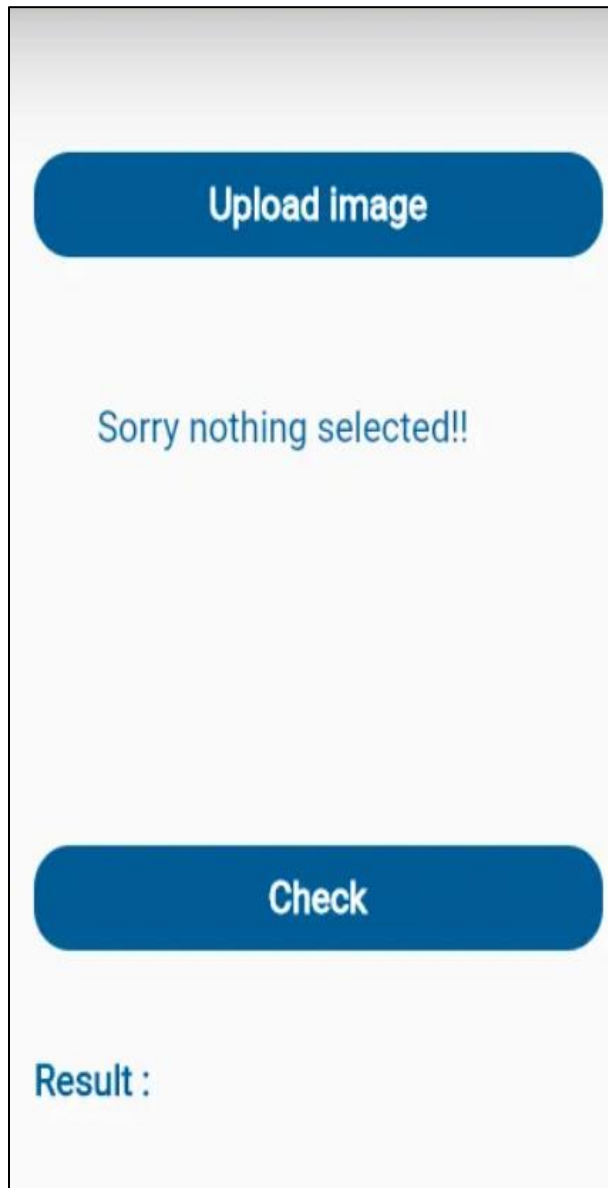


Figure 24: Selecting Page

The user uploads image or video and click on **check** button to get the result (**fake** or **real**)



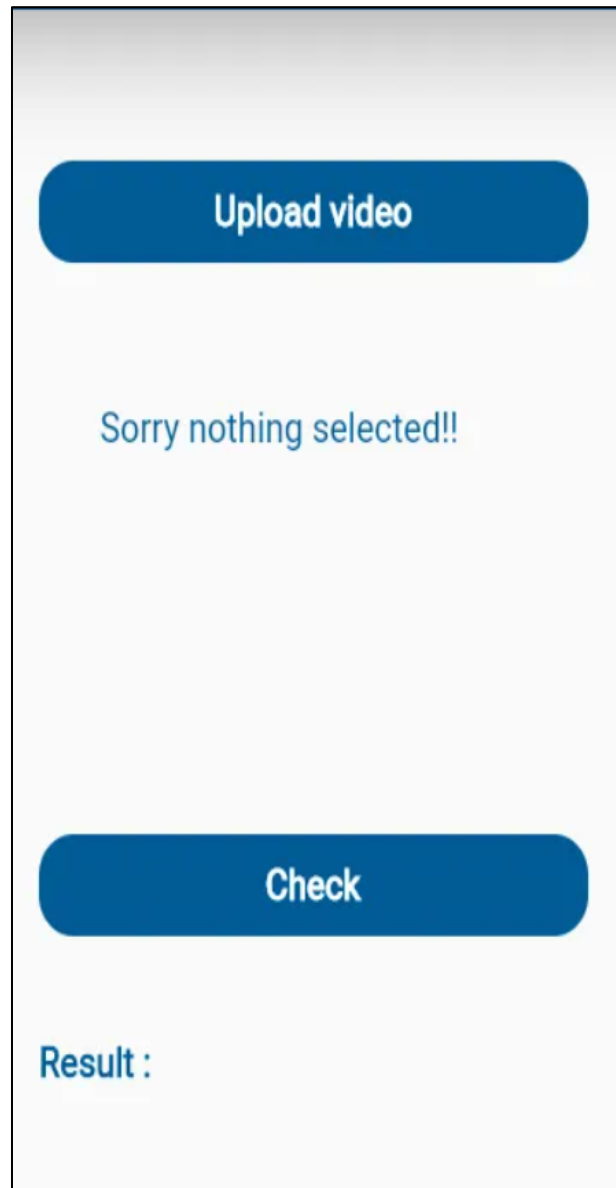
Upload image

Sorry nothing selected!!

Check

Result :

Figure 25: Upload image Page



Upload video

Sorry nothing selected!!

Check

Result :

Figure 26: Upload video Page

4.4. Connection

Firebase is a product of Google which helps developers to build, manage, and grow their apps easily. It helps developers to build their apps faster and in a more secure way. No programming is required on the firebase side, which makes it easy to use its features more efficiently. It provides services to android, iOS, web, and unity. It provides cloud storage. It uses NoSQL for the database for the storage of data. ^[13]

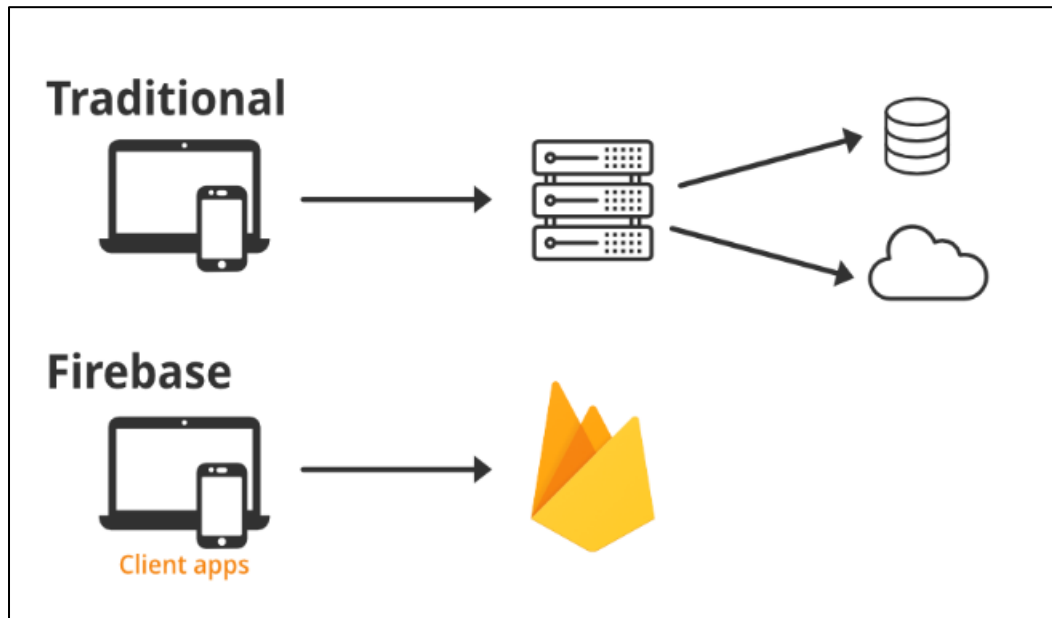


Figure 27: Firebase Architecture

There are many backend services that help developers to build and manage their applications in a better way. These services as:

- **Realtime Database:** The Firebase Realtime Database is a cloud-based NoSQL database that manages your data at the blazing speed of milliseconds (**Big JSON File**). ^[13]

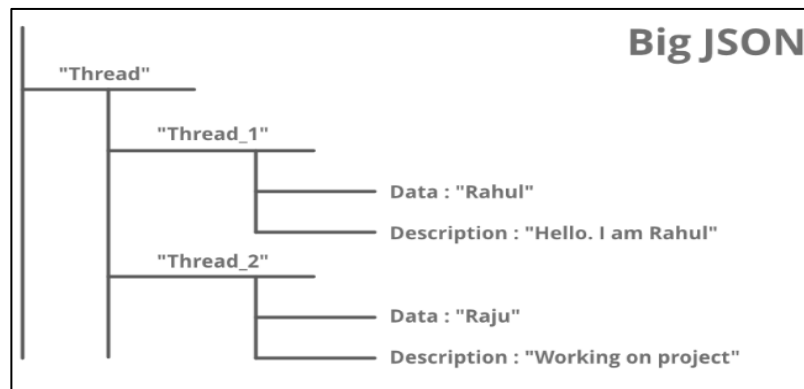


Figure 28: JSON File

- **Cloud Firestore:** The cloud Firestore is a NoSQL document database that provides services like store, sync, and query through the application on a global scale. It stores data in the form of objects also known as Documents. ^[13]

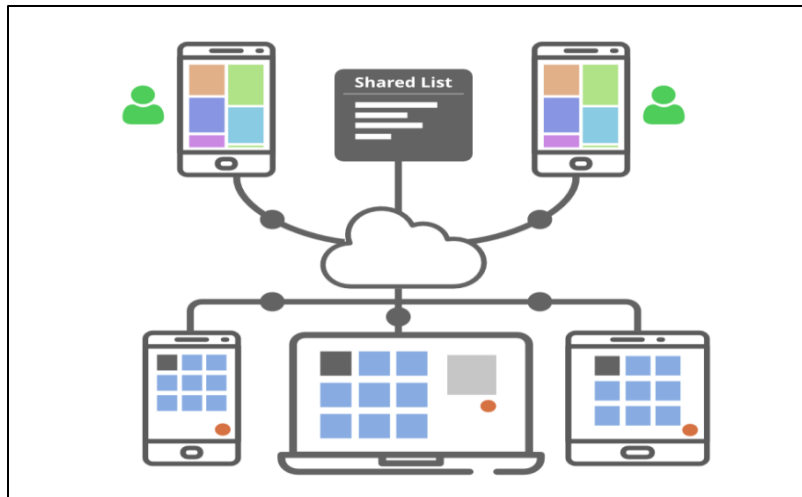


Figure 29: Cloud Firestore

- **Authentication:** Firebase Authentication service provides easy to use UI libraries and SDKs to authenticate users to your app. It reduces the manpower and effort required to develop and maintain the user authentication service. It even handles tasks like merging accounts, which if done manually can be hectic. ^[13]

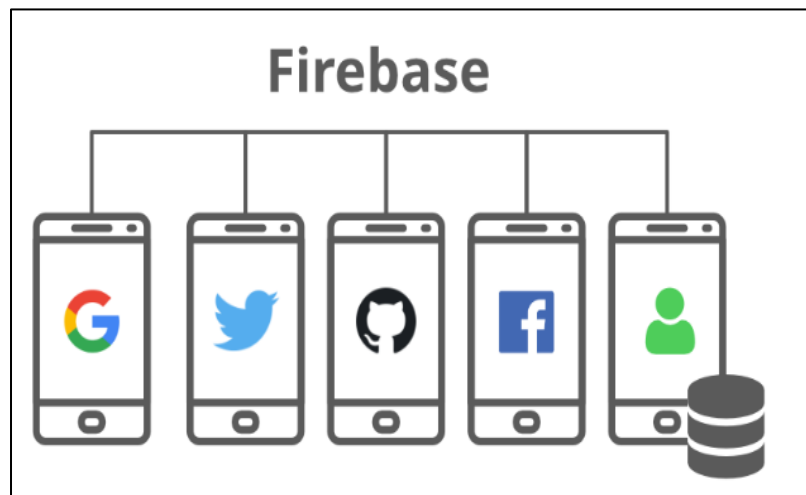


Figure 30: Authentication

- **Remote Config:** The remote configuration service helps in publishing updates to the user immediately. The changes can range from changing components of the UI to changing the behavior of the applications. These are often used while publishing seasonal offers and contents to the application that has a limited life.

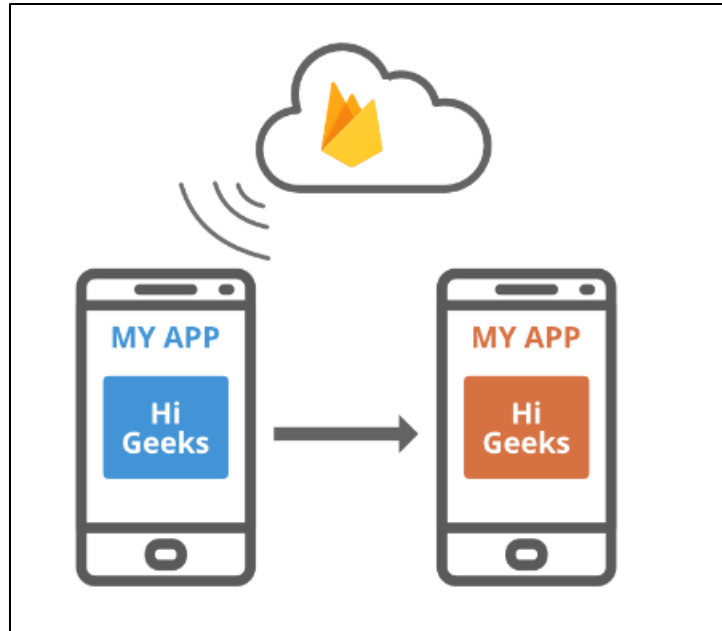


Figure 31: Remote Config

Based on these advantages, we have decided to use Firebase in our project.

4.5. Dataset

- **FaceForensics++** dataset for videos ^[12], We use 900 real videos and 900 fake videos from this dataset.
- **140k Real and Fake Faces** dataset for images ^[13]

5. Testing

5.1. Image Testing

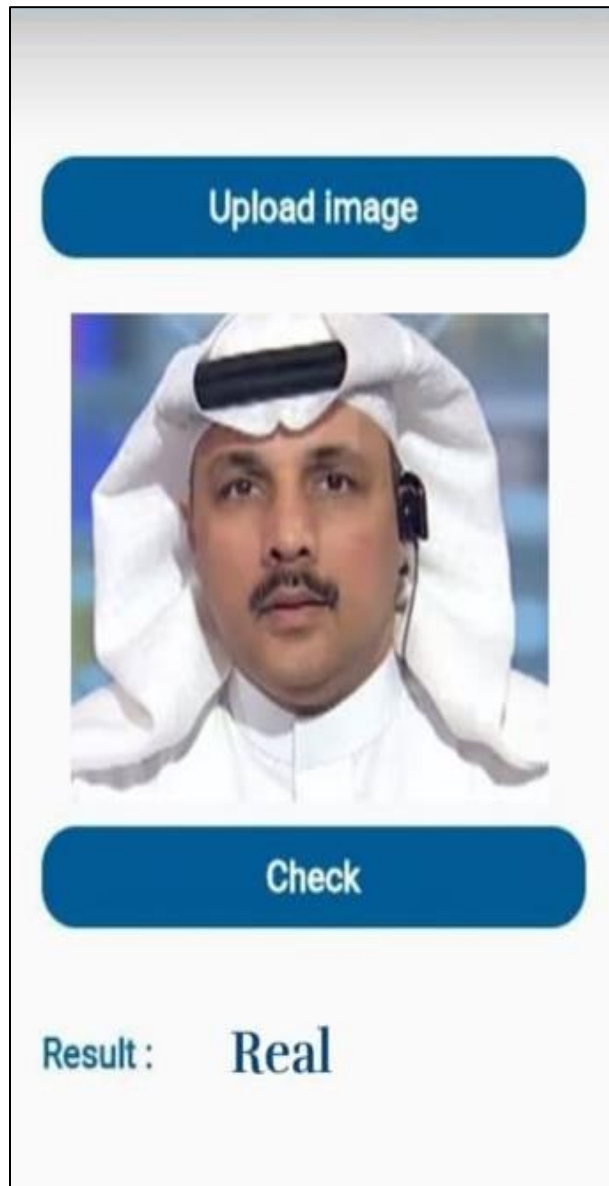


Figure 33: Test Real Image

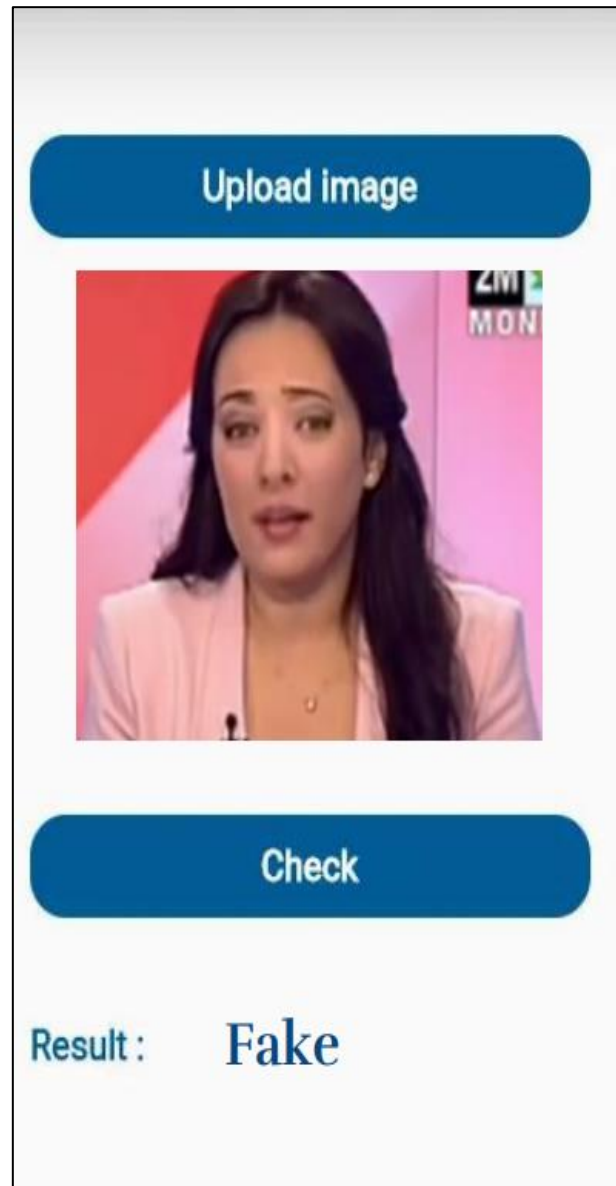


Figure 32: Test Fake Image

5.2. Video Testing



Figure 35: Test Real Video

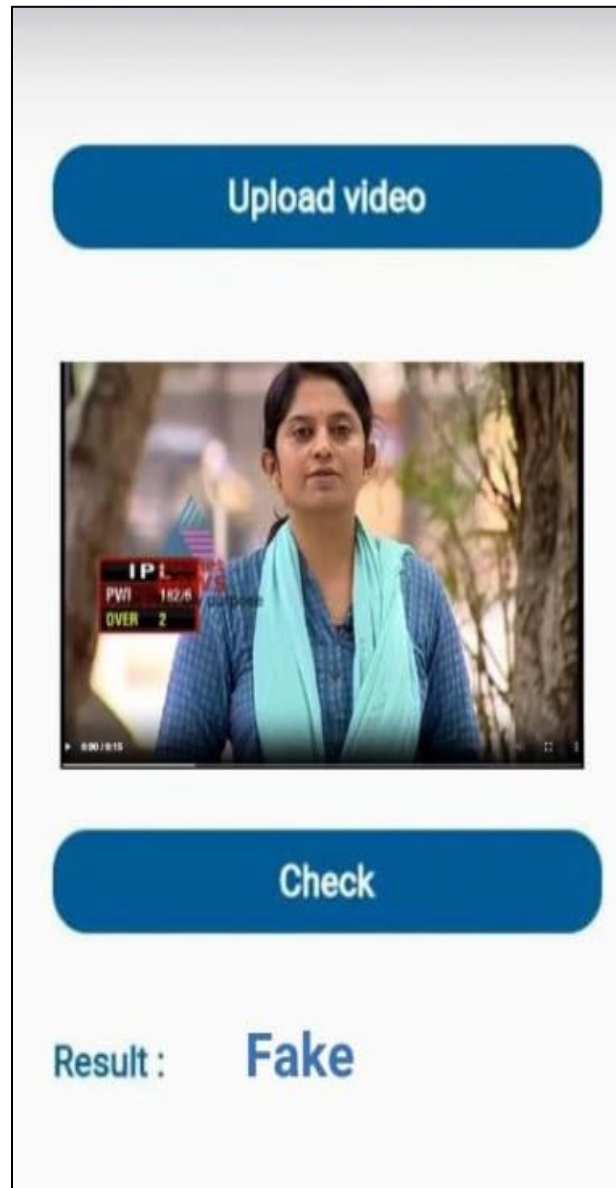


Figure 34: Test Fake Video

6. Conclusion

Due to the spread of social media and its great impact on our lives, as well as the development of technology that we have reached now, it has become easy for anyone to make a fake photo or video for anyone, in addition to publishing it on a large scale, which leads to harming them and exposing their lives to problems and hurt their status in society, it may be sometimes leads to suicide.

This problem was not only exposed to the general public, but also exposed to many important or famous personalities such as US President Barack Obama and actor Tom Cruise.

Deepfake technology has become a source of gain for some people in return for harming others and exhausting their lives.

Fake images or videos are not always complete and completely correct, despite the development of technology, but rather that there are many gaps that can be discovered through them, such as the difference in skin color between the original image and the image superimposed on it, or the eye not moving regularly as usual, or the appearance of teeth as if it was cartoonish, or not moving the head regularly and in wrong directions, and other signs.

So we suggested that we help people who are exposed to this problem and make an application that helps them know whether this image or video is real or fake. This will help our societies eliminate these harmful models of people.

7. Future Work

We hope to spread this application among people about this problem (marketing) and make them aware, in addition to detecting other things that can be faked, such as audio, signatures, sales and purchase contracts, currencies and official papers.

8. References

- [1] Mirza, M. and Osindero, S. (2014) Conditional Generative Adversarial Nets.
- [2] Nataraj, L., et al. (2019) Detecting GAN Generated Fake Images Using Co-Occurrence Matrices. Electronic Imaging, 2019, 532-1-532-7.
<https://doi.org/10.2352/ISSN.2470-1173.2019.5.MWSF-532>
- [3] https://developers.google.com/machine-learning/gan/gan_structure
- [4] <https://www.infobae.com/en/2022/03/30/with-these-5-applications-you-can-create-deep->
- [5] [https://antispoofing.org/Deepfake Detection Software: Types and Practical Application](https://antispoofing.org/Deepfake%20Detection%20Software%20Types%20and%20Practical%20Application)
- [6] <https://ietresearch.onlinelibrary.wiley.com/doi/10.1049/bme2.12031?af=R>
- [7] T. T. Nguyen, D. T. Nguyen, Q. V. H. Nguyen, C. M. Nguyen, D. Nguyen, (26 April 2021) Deep Learning for Deepfakes Creation and Detection
https://www.researchgate.net/publication/336058980_Deep_Learning_for_Deepfakes_Creation_and_Detection_A_Survey
- [8] Tjon, Eric C., "Detecting DeepFakes with Deep Learning" (2020). Master's Projects. 971.
https://scholarworks.sjsu.edu/etd_projects/971
- [9] Almars, A.M. (2021) Deepfakes Detection Techniques Using Deep Learning: A Survey. Journal of Computer and Communications, 9, 20-35.
<https://doi.org/10.4236/jcc.2021.95003>
- [10] <https://www.mathworks.com/help/deeplearning/ref/inceptionresnetv2.html>
- [11] <https://iq.opengenus.org/architecture-of-densenet121/>

[12] <https://www.freecodecamp.org/news/what-is-flutter-and-why-you-should-learn-it-in-2020/>

[13] <https://www.geeksforgeeks.org/firebase-introduction/>

[14] <https://www.kaggle.com/datasets/sorokin/faceforensics>

[15] <https://www.kaggle.com/datasets/xhlulu/140k-real-and-fake-faces>