A Project-1 Report
on

PIRATE VIDEO RECOGNITION (DEEP FAKES) USING DEEP LEARNING

Submitted to
The Department of Computer Science and Engineering

In partial fulfillment of the academic requirements of
Jawaharlal Nehru Technological University
For
The award of the degree of

Bachelor of
Technology in
Computer Science and Engineering
(2017 – 2021)

By

BIKUMALLA SAIKIRAN	17311A05Q2
A. SOMASEKHAR GOUD	17311A05R5
AMMANAGARI VIKAS	17311A05U9

Under the Guidance of
Mrs. M. Yellamma
Assistant Professor

Sreenidhi Institute of Science and Technology

(An Autonomous Institution)
Yamnampet, Ghatkesar, R.R. District, Hyderabad - 501301

Department of Computer Science and Engineering

Sreenidhi Institute of Science and Technology

CERTIFICATE

This is to certify that this Project-1 report on "**PIRATE VIDEO RECOGNITION (DEEP FAKES) USING DEEP LEARNING**", submitted by BIKUMALLA SAIKIRAN (17311A05Q2), A.SOMASEKHAR GOUD (17311A05R5), AMMANAGARI VIKAS (17311A05U9) in the year 2021 in partial fulfillment of the academic requirements of Jawaharlal Nehru Technological University for the award of the degree of Bachelor of Technology in Computer Science and Engineering, is a bonafide work that has been carried out by them as part of their **Project-1 during Fourth Year First Semester**, under our guidance. This report has not been submitted to any other institute or university for the award of any degree.

Internal guide	**Project Coordinator**	**Head of Department**
Dr. K. Basavaraju	Mrs. M. Yellamma	Dr. Aruna Varanasi
Professor	Assistant Professor	Professor & HOD
Department of CSE	Department of CSE	Department of CSE

External Examiner

Date:-

DECLARATION

We **BIKUMALLA SAIKIRAN (17311A05Q2), A.SOMASEKHAR GOUD (17311A05R5), AMMANAGARI VIKAS (17311A05U9)** students of **SREENIDHI INSTITUTE OF SCIENCE AND TECHNOLOGY, YAMNAMPET, GHATKESAR,** studying IVth year Ist semester, **COMPUTER SCIENCE AND ENGINEERING** solemnly declare that the Project-1 work, titled **"PIRATE VIDEO RECOGNITION (DEEP FAKES) USING DEEP LEARNING"** is submitted to **SREENIDHI INSTITUTE OF SCIENCE AND TECHNOLOGY** for partial fulfillment for the award of degree of Bachelor of technology in **COMPUTER SCIENCE AND ENGINEERING**.

It is declared to the best of our knowledge that the work reported does not form part of any dissertation submitted to any other University or Institute for award of any degree.

ACKNOWLEDGEMENT

I would like to express my gratitude to all the people behind the screen who helped me to transform an idea into a real application.

I would like to express my heart-felt gratitude to my parents without whom I would not have been privileged to achieve and fulfill my dreams. I am grateful to our principal, **DR. T. Ch. Siva Reddy,** who most ably runs the institution and has had the major hand in enabling me to do my project.

I profoundly thank **Dr. Aruna Varanasi**, Head of the Department of Computer Science & Engineering who has been an excellent guide and also a great source of inspiration to my work.

I would like to thank our Coordinator **Mrs. M. Yellamma** & my internal guide **Dr. K. Basavaraju** for Project-1 for their technical guidance, constant guidelines, encouragement and support in carrying out my project on time at college.

The satisfaction and euphoria that accompany the successful completion of the task would be great but incomplete without the mention of the people who made it possible with their constant guidance and encouragement crowns all the efforts with success. In this context, I would like to thank all the other staff members, both teaching and non-teaching, who have extended their timely help and eased my task.

BIKUMALLA SAIKIRAN **17311A05Q2**

A. SOMA SEKHAR GOUD **17311A05R5**

AMMANAGARI VIKAS **17311A05U9**

Abstract

Deep learning networks such as GAN's (Generative Adversarial Networks) are used to generate pirate videos (fake videos), or an autoencoder (encoder + decoder) is used to encode media using deep learning networks thereby reconstructing the image to generate a pirate video (fake video). The pirate media generated of this kind are called deep fakes. These Realistic AI-generated videos of people doing and saying fictional things by the use of deep fake techniques, have a significant impact on the user or people's legitimacy of their information provided online. These content generation and modification techniques may affect the quality of public discourse and the safeguarding of human rights, especially given that deepfakes may be used maliciously as a source of misinformation, manipulation, harassment, and persuasion. The effectiveness in misuse of the manipulated or AI-generated media has a greater threat on the confidentiality and legitimacy of the original content. Thus manipulated media identification is a technically demanding and rapidly evolving challenge across the Tech industry and requires collaborations beyond it. Ironically, AI has got a solution to detect the pirate videos, the same AI which has been used to generate these Deep fakes can be used to detect the deep fakes. This project uses deep learning methods such as convolutional neural networks and transfer learning approach to tackle the pirate videos and detect. The Project is an application which verifies whether a video is pirate video (deep fake). The goal of the project is to build an innovative deep learning application that can help detect deep fakes and manipulated media.

LIST OF FIGURES			
S.No	**Fig No**	**Title of Figure**	**Page No**
1	4.1	Architectural Design	08
2	4.2	Use Case Diagram	09
3	4.3	Class Diagram	10
4	4.4	Sequence Diagram	11
5	4.5	Activity Diagram	12
6	5.1	Data preparation	16
7	5.2	Import resources	17
8	5.3	Data preparation	17
9	5.4	Model Building	18
10	5.5	Model Summary	18
11	5.6	Performance Analysis	19
12	5.7	Performance graphs	19
13	5.8	Confusion Matrix	19
14	5.9	Test data preparation	20
15	5.10	Flask Deployment	20
16	5.11	File.html	21
17	5.12	success.html	21
18	5.4.1	Landing Page	22
19	5.4.2	Browse video	22
20	5.4.3	Upload video	23
21	5.4.4	Running instance of flask	23
22	5.4.5	Fake video detected	23
23	5.4.6	Files structure before execution	24
24	5.4.7	Browse video (ii)	24
25	5.4.8	Real Video detected	25
26	5.4.9	Files structure after execution	25

1. INTRODUCTION

Pirate videos also known as fake videos or AI-generated videos are a form of altered multimedia. Altered multimedia compromises one's secrecy and integrity. Pirate video is one in which the dialogue can be modified i.e, the original speech of the person in video is manipulated and replaced with some other dialogue or speech which sinks with lip movements and facial nodes. This kind of video altering can be done using deep learning networks such as GAN's (Generative adversarial networks) or an autoencoder (encoder + decoder) pair. These kinds of videos are generally called deep fakes. These videos can be detected by using the same Artificial Intelligence, and we are currently doing that.This project uses deep learning methods such as convolutional neural networks and transfer learning approach to tackle the pirate videos and detect. The same AI is used to detect the fake videos.

1.1 Motivation

Pirated videos are a trending topic nowadays. The original videos are manipulated by the usage of some video editing tools or using Artificial Intelligence techniques and the same replica of the original videos are made which seem to be true but in fact are fake videos.

Recently an indian politician named Manoj Tiwari from BJP (Bharatiya Janata Party) used deep fakes technology to win new votes. The fake video went viral in Whatsapp just ahead of legislative assembly elections in Delhi. He always used to campaign in Haryanvi language. But he wanted to target the hindi speaking voters in the area. His original video was manipulated using deepfake technology and videos in Hindi and English language were created accordingly as per the movements of his mouth.

There are many more instances where this technology has been used. Many have used this technology to manipulate the videos of popular or renowned personalities like Donald Trump, Mark Zukerberg etc. As people don't have the idea of recognising whether it is a real one or fake one, we have decided to develop a deep learning model which can detect the originality of the video. The Video editing tools altered videos can be recognized somewhat easily by humans by looking extensively, but the videos altered by Artificial Intelligence techniques and algorithms can't be recognized by just looking at them.

1.2 Problem Definition

The word deepfake combines the terms “deep learning” and “fake,” and is a form of artificial intelligence. Simply, deep fakes are false videos created with the help of deep learning. Deep learning is the subset of machine learning and machine learning is the subset of artificial intelligence. But because of this technology people are made to believe something which is false. Here the superimposing of faces of a target person to an other video in order to fake the video. The imitation of the person is done in the fake video perfectly and the fake videos cannot be found easily and they spread in the internet like a wildfire. The movements and the analogous expressions are captured and the videos are just realistic videos that one cannot differentiate them. This technology can be applied to both images and videos. The first time when the deepfakes is used to swap the face of a celebrity with a porn star. It is really intimidating to the security of the world. It can cause havoc to the secrecy and integrity of one's security.

1.3 Objective of Project

The objective of the project is to find out whether the video which is uploaded is real or fake one. There are many tools to detect the originality of the video but they are not reliable, we want to make a model which gives high accuracy and is consistent. We want to provide an Flask interface to our project which is a web interface. It consists of two buttons one is used to upload the video whereas the other is used to submit the video.When the video is submitted the video goes as the input to the already trained model. The process here goes as, each and every frame of the video is compared with that of fake ones. If they are thought to be fake by the model ,then one is returned and if they are thought to be not fake then zero is returned. The statistical mode of the data (all zeroes and ones returned by model processing different frames of video) is taken, if the result is one then the video is fake else if the result is zero then the video is real. The output is displayed in the next web page which gives the information about the file name and nature of the video i.e whether the video is real or fake. The video along with its tag (either fake or real) can also be saved to the database.

2. LITERATURE SURVEY

2.1 Existing System

There are many pirate video detection tools but the usage of them isn't that trustworthy. Most of them are private video detection that can be done easily as the model cannot detect the eye blinking of the person. If you cannot find the blinking his eyes forever in the video then it can tend to be a fake video.

Poor lip syncing in the video and bad image quality of the videos are the litmus test to find the originality of the video. Strange light effects, badly affected lightening and jewelry are used to spot the nature of the video. The above features are tested by the above mentioned tools. Inconsistent illuminations and reflections are two other factors which effect the quality of the video. There are some machine learning approaches which try to detect whether an image is fake or not by using facial nodes.

2.2 Proposed System

The transfer learning approach is used here to find good accuracy of the model.

The inception_resnet model is used and with the help of transfer learning the model is created and the final two layers of the network are removed and manipulated as desired. The pooling layer is added with the required number of trainable inputs and the final softmax layer consists of two trainable features that are real and fake.

The data is divided into large chunks of data and the model is trained on the data. Before training the model, the input data is in the form of video, it is converted into different frames (images). These images are then processed and are used to build and train the model. The model is loaded and the input to this model is given by uploading the video with the help of the upload button which will be created using the flask interface. When the submit button is pressed the video gets processed and the output is given in the next web page whether the video uploaded is real or fake.

3. SYSTEM ANALYSIS

This System Analysis is closely related to requirements analysis. It is also an explicit formal inquiry carried out to help someone (referred to as the decision maker) identify a better course of action and make a better decision than he might otherwise have made. This step involves breaking down the system in different pieces to analyze the situation, analyzing project goals, breaking down what needs to be created and attempting to engage users so that definite requirements can be defined.

3.1 Functional Requirement Specifications

The system after careful analysis has been identified to be present with the following modules.

1. Data Gathering Module:

This module deals with gathering of the training and test data from the internet. The gathered videos from the internet are then wrangled. Transfer learning approach or any multimedia manipulation cannot be done on videos, instead videos are converted into images based on their frame rates (32fps etc). Hence in this module initially the gathered videos consisting of both fake videos and original videos are stored as fake and real folders. Then the wrangling is done on the videos to generate images from them and the images are finally stored with labels as fake and real, which are used to train the deep learning model.

2. Model Development module:

This module deals with model building. After gathering and segregating the data in the gathering phase, a deep learning model is built. Here we use the transfer learning approach to build a model using an existing or pre built model **Inception ResNet v2.** Transfer learning refers to adding extra layers at the end of an existing network according to the usage and purpose. Here we add different layers such as global average pooling layer and dense layer. After the model is built it is then trained in the training module by fitting data on it.

3. Training Module:

This module includes creation of training data from the data set which is fed to the model. This module is used to train the model which is built earlier in the development module. The model is trained on the training data which contains different fake and real videos for different number of epochs. Cross validation set is added to ensure the model doesn't overfit by running on extra epochs. The number of epochs is dependent on the cross validation set accuracy and cross validation set loss. Training is stopped at a certain epoch by looking at the training and cross validation accuracy and loss.

4. Testing and Deployment Module:

This module includes testing the model on the test data and then deploying it using the flask interface. The model is loaded and the input to this model is given by uploading the video with the help of the upload button which will be created using the flask interface. When the submit button is pressed the video gets processed and the output is given in the next web page whether the video uploaded is real or fake.

3.2 Performance Requirements:

Performance is measured in terms of the output provided by the application. Requirement specification plays an important part in the analysis of a system. Only when the requirement specifications are properly given, it is possible to design a system, which will fit into the required environment. It rests largely with the users of the existing system to give the requirement specifications because they are the people who finally use the system. This is because the requirements have to be known during the initial stages so that the system can be designed according to those requirements. It is very difficult to change the system once it has been designed and on the other hand designing a system, which does not cater to the requirements of the user is of no use.

The requirement specification for any system can be broadly stated as given below:

- The system should be accurate

3.3 Software Requirements

- ➢ **Operating System:** Microsoft Windows / Linux / MacOS
- ➢ **Technology :** Artificial Intelligence - Deep Learning
- ➢ **Tools:** Jupyter Notebook / Spyder, Flask framework
- ➢ **Platform:** Anaconda Distribution or Google Colab

To run a software it should satisfy minimum system requirements. For accomplishment of this project we used ANACONDA Distribution. In Anaconda Distribution, either Jupyter Notebooks or Spyder can be used. If the system doesn't have GPU acceleration training, building and fitting the training data to the model would become very choppy, due to high computational expense. In such cases Google Colabs can be used, as it provides a decent GPU which is quite enough for running this project. For deploying the project, Flask is used. Flask is a micro web framework which is written in python.

Jupyter Notebooks

Jupyter notebook or formerly known as IPython notebooks comes with Anaconda Distribution or even can be installed Individually. Jupyter notebooks is a web based interactive computational interface which supports various languages such as Python, Ruby, R, Julia etc. Jupyter notebooks can be converted to various standard output formats such as HTML, pdf, Python files etc.

Spyder:

Spyder is an Interactive Development environment (IDE) for python programming especially for scientific computational programming. Spyder integrates with python's scientific and computational packages such as numpy, pandas, scikit-learn, Scipy, pandas, Matplotlib etc. Spyder is an editor and supports features such as automatic code completion or suggestion, support of multiple IPython consoles parallely.

Flask Framework:

Flask is a web framework written in python language. It is a lightweight framework and it does not require a particular set of libraries or packages. Flask framework can be used to embed machine learning and deep learning algorithms with web pages (HTML, bootstrap etc). The flask framework is an interface between the backend machine learning or neural network model and the input providing GUI of the end user.

3.4 Hardware Requirements:

Processor

- **Minimum :** AMD x86 or Any intel - 64 bit processor
- **Recommended:** Any Intel or AMD processor with at least 8 logical cores (logical CPUs) and hyperthreading capability. Preferably Intel i5 8th gen equivalent or above.

Disk

- **Minimum:** 3 GB of HDD for anaconda distribution and at least 50 GB to store training and testing data chunk wise.
- **Recommended:** 300 GB free space to store entire data. Having an SSD is an advantage.

RAM

- **Minimum:** 8 GB DDR4 RAM
- **Recommended:** 8 - 16 GB DDR4 RAM or 8 GB DDR5 RAM

Graphics Processing Unit

- GPU acceleration or having GPU makes computation very fast
- GPU support is an added advantage
- 4 GB of graphics card is more than enough.

4. SYSTEM DESIGN

System design can be said as the process in which the architecture, interfaces, components, modules and data for a system are defined such that requirements specified are satisfied. In other views it can be seen as application of theory of systems for product development. Generally, the most widely used methods for systems design are Object-oriented analysis and design methods

4.1 Architectural Design:

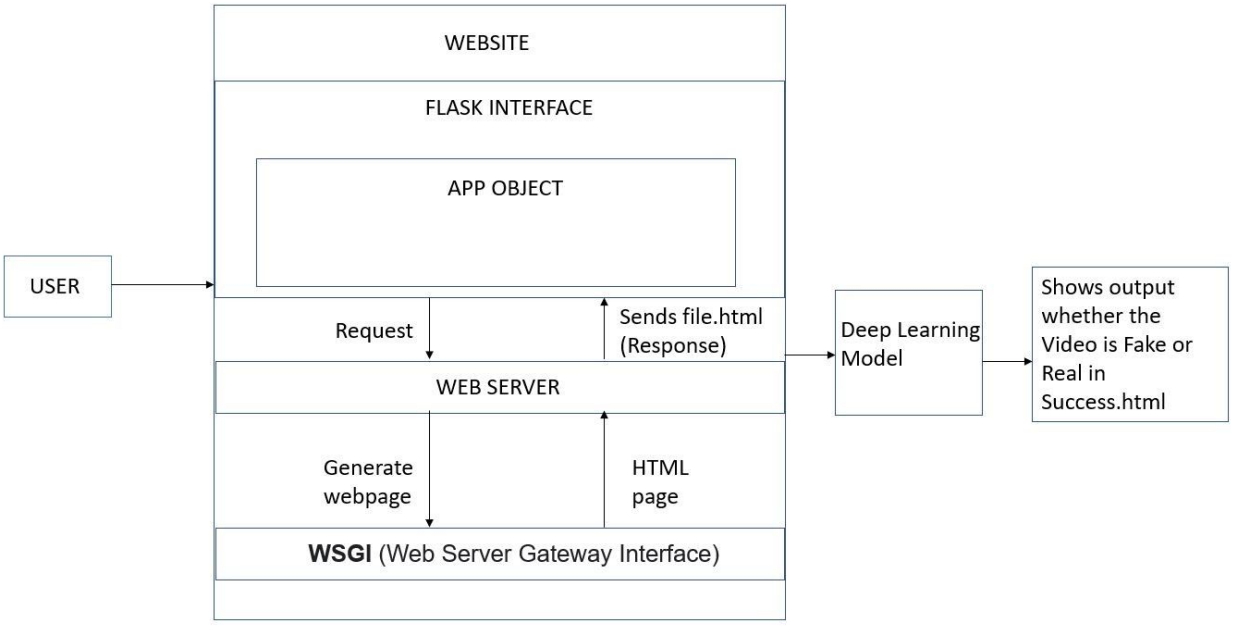


Fig 4.1 Architectural Design

In the proposed system, the user interacts with the system using Flask interface which is a web interface.The web design consists of two files named file and success. The request is first given to the web server and the webpage is generated which returns the html page to the webserver and it returns the response to the system and the web page is displayed. The first file consists of two buttons. The upload button is used to upload the video and when the submit button is pressed the video is fed to the deep learning model after necessary preprocessing. The video is then input to the deep learning model which will give the result whether the video is fake or original. The output is then displayed in the second webpage.

4.2 UML Diagrams

4.2.1 Use Case Diagram

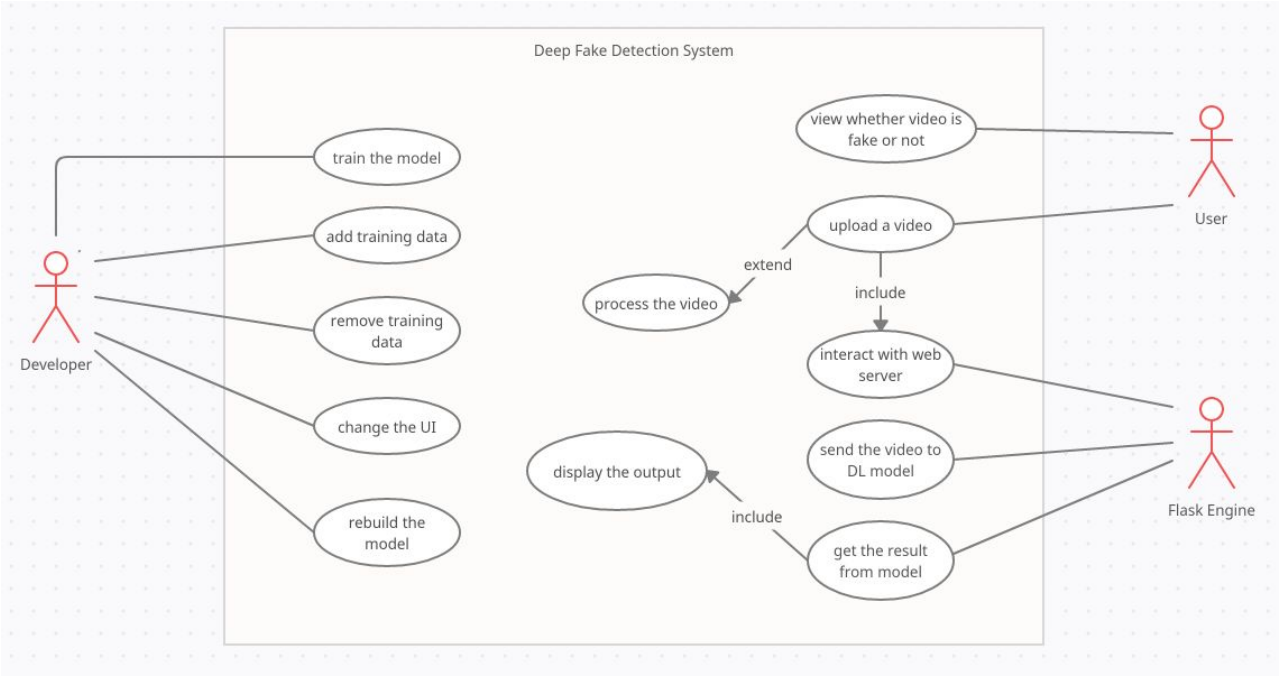


Fig 4.2 Use Case Diagram

Developer:

The developer here collects the large chunks of data and trains the model and saves the model.

The video which is taken as input from the User interface is given as input to the model and the model gives us the result whether the video is fake or real.

User:

The User here can upload the video and can interact with the User interface. In the system, we can observe that the user interacts with the application through a graphical user interface.

Flask Engine:

The Flask engine sends the uploaded video to the deep learning model as input which also captures the output and displays it in the next web page.

4.2.2 Class Diagram

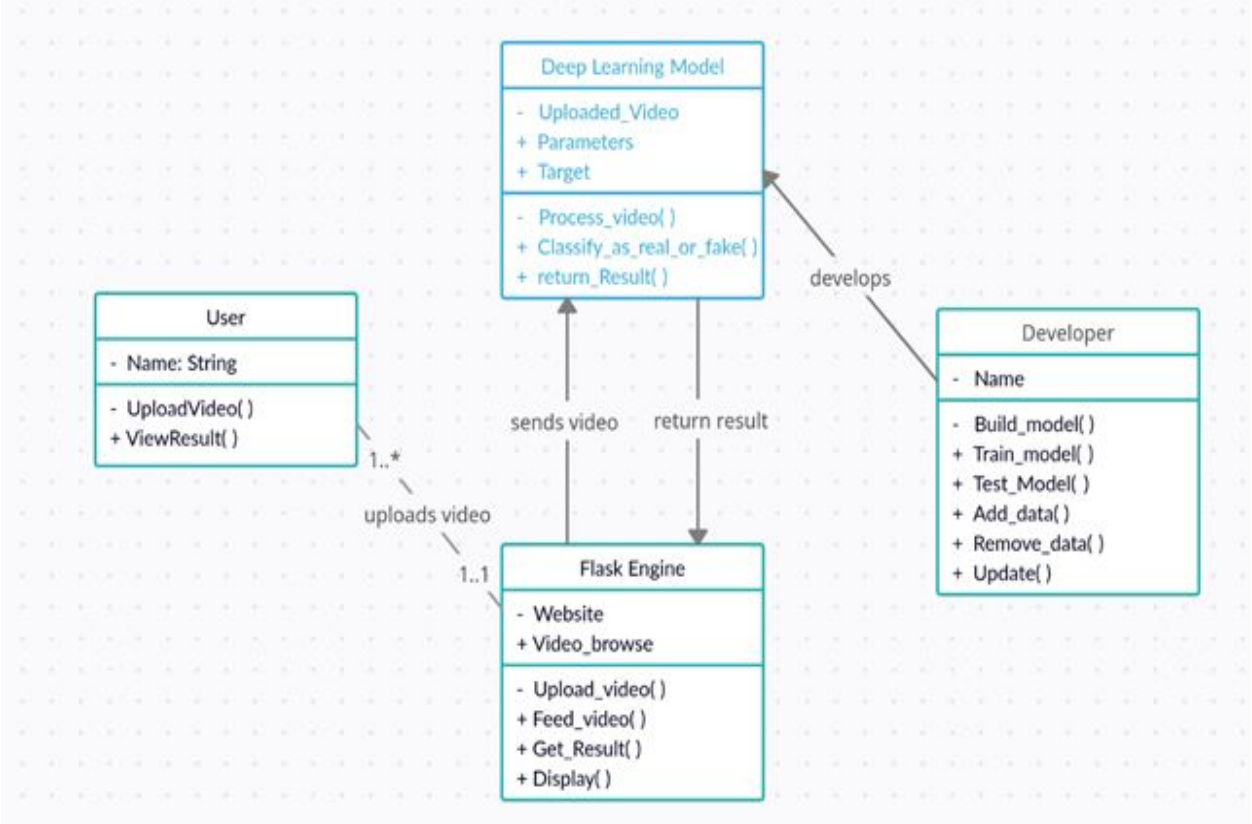


Fig 4.3 Class Diagram

There are four classes: User, Flask Engine, Developer and Deep learning model.

The User can Upload the video and view the result whether the video is fake or real.

The Flask Engine provides the interface such that the videos can be updated. The videos can be uploaded and the results are displayed. It also provides the functionality of displaying the web pages.

The developer has the functionalities like collecting the data, building the model, training the model, testing the model and updating the data if required.

The deep learning model takes the video and processes the video and finally gives the result (real or fake).

4.2.3 Sequence Diagram

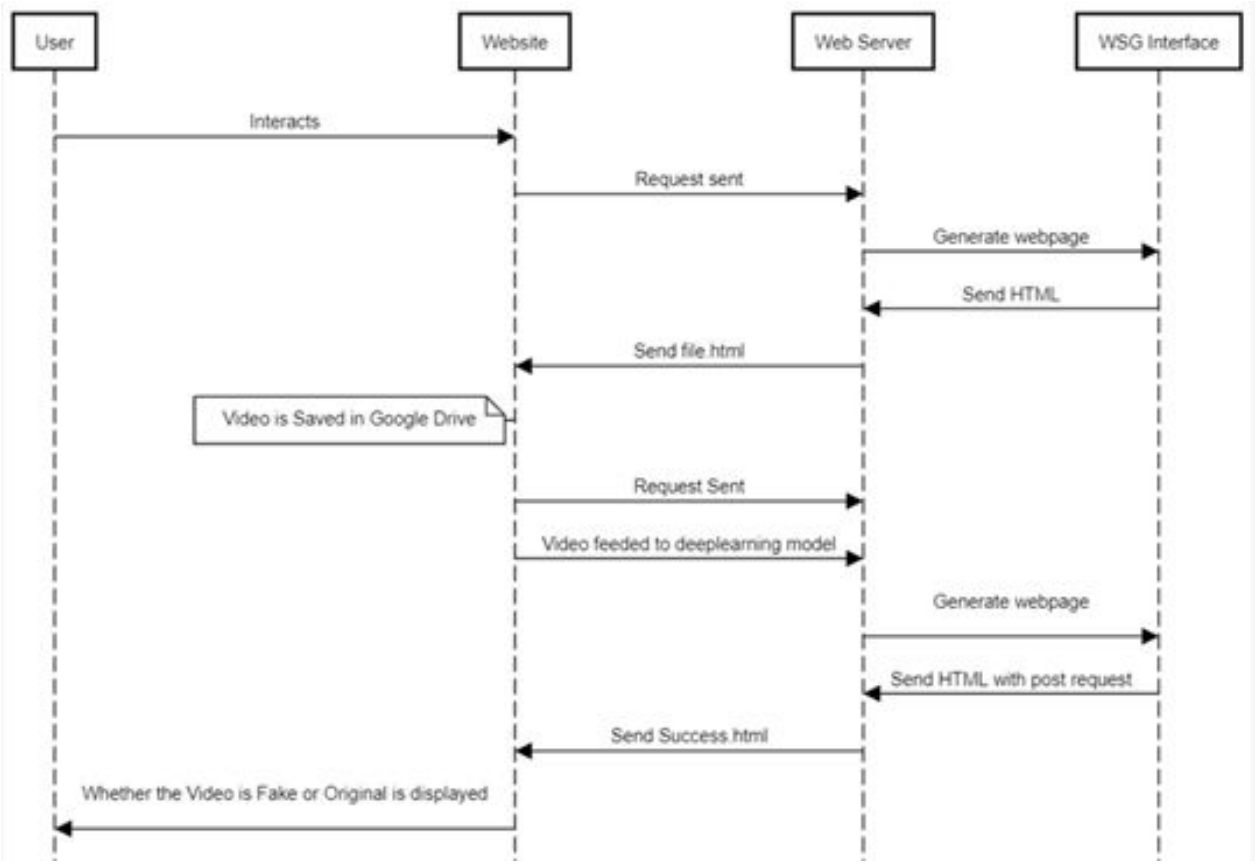


Fig 4.4 Sequence Diagram

The User above interacts with the website by uploading the video and hits on submit button and the request is send to the web server and then the generation of web page is done and the WSGI sends the html page and the file as response is send and the video is also saved in the google drive and the uploaded video is given as input to deep learning model and the model gives the result whether the video is real or fake.The nature of the video is passed to the next web page.The Web Server generates the webpage and the WSGI sends the webpage along with the post request and the web server sends the response to the website and the content is displayed about the nature of the video. The user can then see the nature of the video.

4.2.4 Activity Diagram

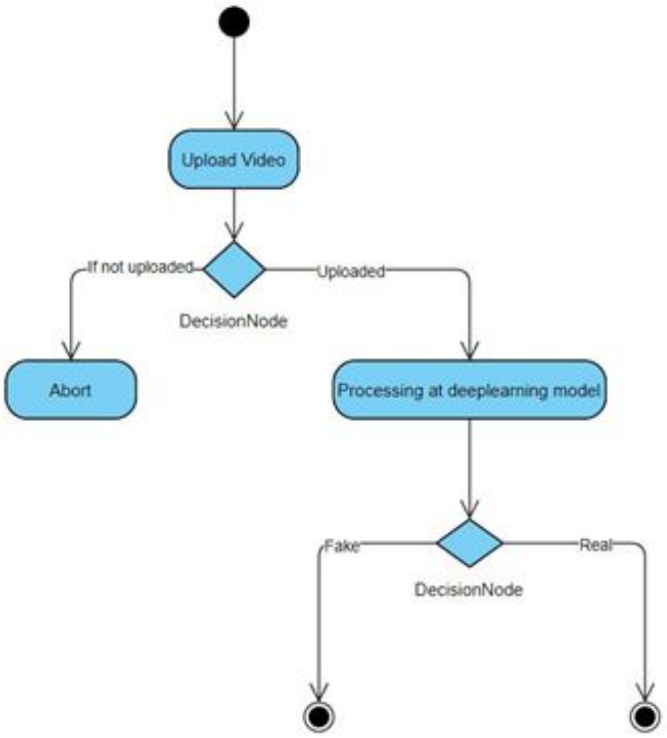


Fig 4.5 Activity Diagram

The Video is uploaded using the upload button in the webpage if the video is not uploaded it goes to the abort state if uploaded it is feeded as input to the deep learning model and at the final stage the decision is made whether the video is fake or real.

4.3 Modules

1. Data Gathering Module:

This module deals with gathering of the training and test data from the internet. The gathered videos from the internet are then wrangled. Transfer learning approach or any multimedia manipulation cannot be done on videos, instead videos are converted into images based on their frame rates (32fps etc). Hence in this module initially the gathered videos consisting of both

fake videos and original videos are stored as fake and real folders. Then the wrangling is done on the videos to generate images from them and the images are finally stored with labels as fake and real, which are used to train the deep learning model.

2. Model Development module:

This module deals with model building. After gathering and segregating the data in the gathering phase, a deep learning model is built. Here we use the transfer learning approach to build a model using an existing or pre built model **Inception ResNet v2.** Transfer learning refers to adding extra layers at the end of an existing network according to the usage and purpose. Here we add different layers such as global average pooling layer and dense layer. After the model is built it is then trained in the training module by fitting data on it.

3. Training Module:

This module includes creation of training data from the data set which is fed to the model. This module is used to train the model which is built earlier in the development module. The model is trained on the training data which contains different fake and real videos for different number of epochs. Cross validation set is added to ensure the model doesn't overfit by running on extra epochs. The number of epochs is dependent on the cross validation set accuracy and cross validation set loss. Training is stopped at a certain epoch by looking at the training and cross validation accuracy and loss.

4. Testing and Deployment Module:

This module includes testing the model on the test data and then deploying it using the flask interface. The model is loaded and the input to this model is given by uploading the video with the help of the upload button which will be created using the flask interface. When the submit button is pressed the video gets processed and the output is given in the next web page whether the video uploaded is real or fake.

5. IMPLEMENTATION AND RESULTS

5.1 Language / Technology Used :

Python language is used to write the code. Python provides a wide variety of libraries for scientific and computational usage. Libraries such as scikit learn, scipy, numpy, keras, tensorflow etc are used to detect the fake videos. The main functionalities in the project are

1. **Data Preparation:** Converting the videos into sequence of frames (images)
2. **Model Building:** Building a deep learning model (neural network)
3. **Model Training:** Training the model by fitting the training data to it.
4. **Testing and deployment**: Making the model test ready and deploying it in a web server.

The libraries or packages used for the above functionalities are :

Data Preparation:

- opencv library is used to capture the video and generate frames
- json and os libraries are used to read the labels, rename the images and store them
- PIL (python image library) is used to manipulate the images

Model building and Training:

- opencv library is used for image processing
- os and json libraries are used to retrieve images and perform operations on them
- tensorflow is a Artificial Intelligence library used to create multi layer models, neural networks and is used for purposes such as classification, prediction etc
- keras is a library to create and manage neural networks.
- matplotlib and seaborn are data visualization libraries.
- sklearn is a library which is used for machine learning and statistical modelling of data
- pandas is a data management library

Testing and Deployment:

- numpy is a library used for numerical calculations for image pixel data manipulation
- flask and gevent are used to deploy the model in a web framework

Technology used in this project is Artificial Intelligence. Deep learning is a part of Artificial intelligence, Artificial Neural Networks are created to detect the deep fakes. Transfer learning approach, which is a deep learning algorithm, is used. Deep learning algorithms are AI that learn from both structured and unstructured data without any human supervision.

Convolutional Neural Network is a deep learning algorithm which is used for analyzing images and generating insights from it by learning and processing the image. Deep learning algorithms are generally designed inorder to mimic the functioning of the human brain.

5.2 Algorithms Implemented (Transfer Learning):

Transfer learning, a deep learning algorithm is used in this project. Transfer learning refers to storing the knowledge gained while solving one problem (trained on that problem data) and applying the stored knowledge on a different but similar problem. For example knowledge gained while recognizing handwritten digits can be used to recognize handwritten characters by slightly modifying the existing model and using the pre-gained knowledge.

Transfer learning approach involves referring a pre trained neural network and tweaking or modifying the neural network in order to use the knowledge gained on a new or different problem based on size of the dataset of new problem (small or large) and similarity of the data with respect to the original or pre-trained dataset.

Based on the size and similarity of the dataset being used and pre trained data the existing or pre trained model can be fine tuned, or fine tuned and retrained, or add required layers such as global average pooling, dense etc at the end of the neural network, or add layers at the start of network.

In general the steps in implementing the transfer learning approach are :

1. import the resources, load the data set and explore it
2. create pipeline: pretrained model might have some dimensionality and size of data and the new dataset might have different, hence the data is normalised or scaled to the requirements of the original data set or pre-trained model.
3. build the model by making necessary changes to the existing model such as adding layers
4. Compile the newly built model and fit the training data to train the model
5. Check the predictions on the new model

5.3 Basic Code

Project directory structure and the important files are:

__init__.py	an empty file that helps python find your actions
capture_img.ipynb	code to generate images from the dataset containing videos
dataset / real	directory containing all the images with the label: real
dataset / fake	directory containing all the images with the label: fake
detection_train.ipynb	code to implement transfer learning approach and build the model
model_play.ipynb	code to deploy model to a web page using flask framework
file.html	landing page of the user interface
success.html	web page which displays the output

Table 5.1 Project files list

capture_img.ipynb :

```
import dlib
import cv2
import os
import re
import json
from pylab import *
from PIL import Image, ImageChops, ImageEnhance

train_frame_folder = '/content/drive/My Drive/project/train_sample_videos'
with open(os.path.join(train_frame_folder, 'metadata.json'), 'r') as file:
    data = json.load(file)
list_of_train_data = [f for f in os.listdir(train_frame_folder) if f.endswith('.mp4')]
detector = dlib.get_frontal_face_detector()
for vid in list_of_train_data:
    count = 0
    cap = cv2.VideoCapture(os.path.join(train_frame_folder, vid))
    frameRate = cap.get(5)
    while cap.isOpened():
        frameId = cap.get(1)
        ret, frame = cap.read()
        if ret != True:
            break
        if frameId % ((int(frameRate)+1)*1) == 0:
            face_rects, scores, idx = detector.run(frame, 0)
            for i, d in enumerate(face_rects):
                x1 = d.left()
                y1 = d.top()
                x2 = d.right()
                y2 = d.bottom()
                crop_img = frame[y1:y2, x1:x2]
                if data[vid]['label'] == 'REAL':
                    a = cv2.imwrite('/content/drive/My Drive/project/dataset/real/'+vid.split('.')[0]+'_'+str(count)+'.png', cv2.resize(crop_img, (128, 128)))
                    if a: print(".",end="")
                elif data[vid]['label'] == 'FAKE':
                    b = cv2.imwrite('/content/drive/My Drive/project/dataset/fake/'+vid.split('.')[0]+'_'+str(count)+'.png', cv2.resize(crop_img, (128, 128)))
                    if b: print("*",end="")
                count+=1
```

Fig 5.1 data preparation

deepfake_detection_train.ipynb :

```
import os
import cv2
import json
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sn
import pandas as pd
from tensorflow.keras.preprocessing.image import ImageDataGenerator, img_to_array, load_img
from tensorflow.keras.utils import to_categorical
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning: panda
  import pandas.util.testing as tm
```

Fig 5.2 import resources

```
input_shape = (128, 128, 3)
data_dir = '/content/drive/My Drive/project/dataset'

real_data = [f for f in os.listdir(data_dir+'/real') if f.endswith('.png')]
fake_data = [f for f in os.listdir(data_dir+'/fake') if f.endswith('.png')]

X = []
Y = []

for img in real_data:
    X.append(img_to_array(load_img(data_dir+'/real/'+img)).flatten() / 255.0)
    Y.append(1)
for img in fake_data:
    X.append(img_to_array(load_img(data_dir+'/fake/'+img)).flatten() / 255.0)
    Y.append(0)

Y_val_org = Y

#Normalization
X = np.array(X)
Y = to_categorical(Y, 2)

#Reshape
X = X.reshape(-1, 128, 128, 3)

#Train-Test split
X_train, X_val, Y_train, Y_val = train_test_split(X, Y, test_size = 0.2, random_state=5)
```

Fig 5.3 Data preparation

```python
from tensorflow.keras.applications import InceptionResNetV2
from tensorflow.keras.layers import Conv2D
from tensorflow.keras.layers import MaxPooling2D
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import InputLayer
from tensorflow.keras.layers import GlobalAveragePooling2D
from tensorflow.keras.models import Sequential
from tensorflow.keras.models import Model
from tensorflow.keras import optimizers
from tensorflow.keras.callbacks import ReduceLROnPlateau, EarlyStopping

googleNet_model = InceptionResNetV2(include_top=False, weights='imagenet', input_shape=input_shape)
googleNet_model.trainable = True
model = Sequential()
model.add(googleNet_model)
model.add(GlobalAveragePooling2D())
model.add(Dense(units=2, activation='softmax'))
model.compile(loss='binary_crossentropy',
              optimizer=optimizers.Adam(lr=1e-5, beta_1=0.9, beta_2=0.999, epsilon=None, decay=0.0, amsgrad=False),
              metrics=['accuracy'])
model.summary()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/inception_resnet_v2/inception_resnet_v2_weights_tf_dim_ordering_tf_kernels_notop.h5
219062272/219055592 [==============================] - 7s 0us/step
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
inception_resnet_v2 (Functio (None, 2, 2, 1536)        54336736
_________________________________________________________________
global_average_pooling2d (Gl (None, 1536)              0
_________________________________________________________________
dense (Dense)                (None, 2)                 3074
=================================================================
Total params: 54,339,810
Trainable params: 54,279,266
Non-trainable params: 60,544
_________________________________________________________________
```

Fig 5.4 Model Building

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
inception_resnet_v2 (Functio (None, 2, 2, 1536)        54336736
_________________________________________________________________
global_average_pooling2d (Gl (None, 1536)              0
_________________________________________________________________
dense (Dense)                (None, 2)                 3074
=================================================================
Total params: 54,339,810
Trainable params: 54,279,266
Non-trainable params: 60,544
_________________________________________________________________
```

Fig 5.5 Model Summary

```
[ ]  f, (ax1, ax2) = plt.subplots(1, 2, figsize=(20, 4))
     t = f.suptitle('Pre-trained InceptionResNetV2 Transfer Learn with Fine-Tuning & Image Augmentation Performance ', fontsize=12)
     f.subplots_adjust(top=0.85, wspace=0.3)

     epoch_list = list(range(1,EPOCHS+1))
     ax1.plot(epoch_list, history.history['accuracy'], label='Train Accuracy')
     ax1.plot(epoch_list, history.history['val_accuracy'], label='Validation Accuracy')
     ax1.set_xticks(np.arange(0, EPOCHS+1, 1))
     ax1.set_ylabel('Accuracy Value')
     ax1.set_xlabel('Epoch #')
     ax1.set_title('Accuracy')
     l1 = ax1.legend(loc="best")

     ax2.plot(epoch_list, history.history['loss'], label='Train Loss')
     ax2.plot(epoch_list, history.history['val_loss'], label='Validation Loss')
     ax2.set_xticks(np.arange(0, EPOCHS+1, 1))
     ax2.set_ylabel('Loss Value')
     ax2.set_xlabel('Epoch #')
     ax2.set_title('Loss')
     l2 = ax2.legend(loc="best")
```

Fig 5.6 Performance Analysis

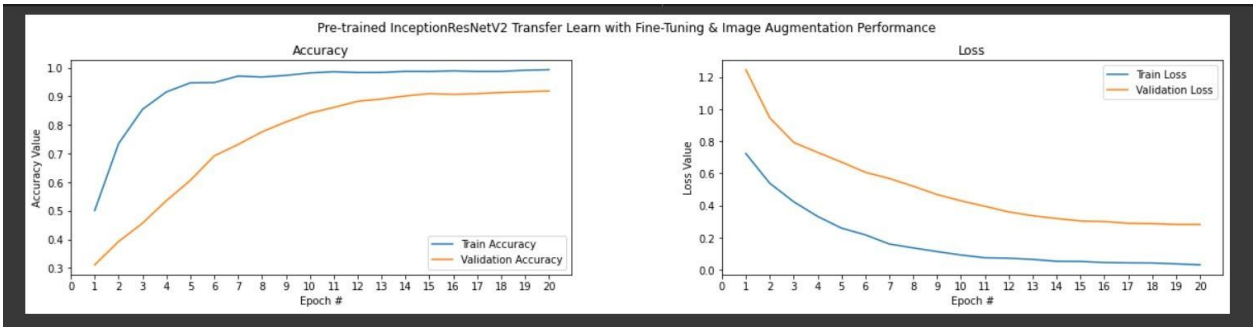


Fig 5.7 Performance graphs

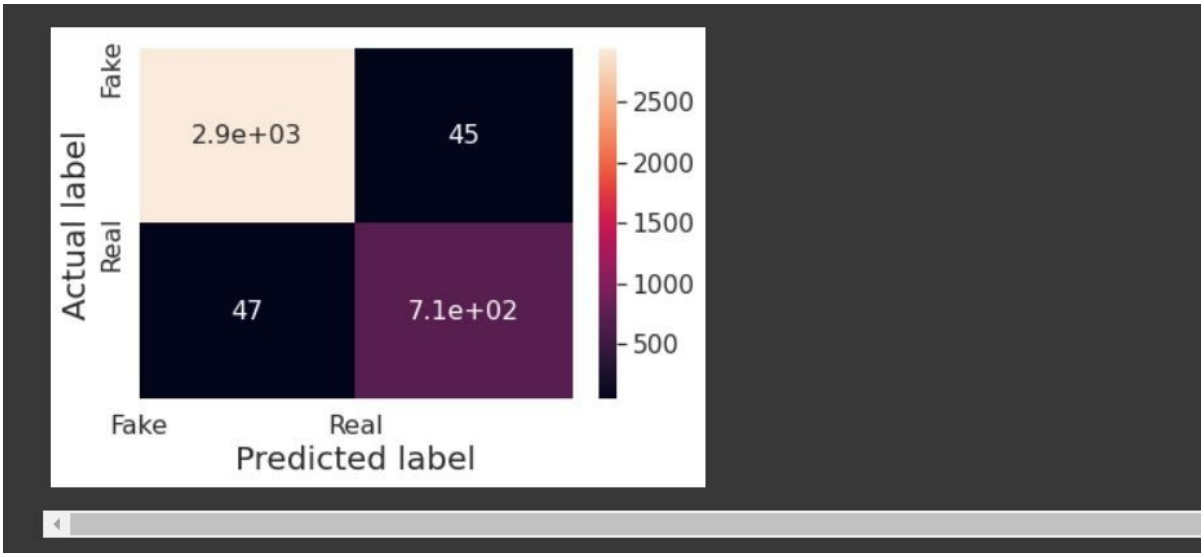


Fig 5.8 Confusion Matrix

model_play.ipynb :

```
[5] input_shape = (128, 128, 3)
    pr_data = []
    detector = dlib.get_frontal_face_detector()
    cap = cv2.VideoCapture('/content/drive/My Drive/project/real.mp4')
    frameRate = cap.get(5)
    temp = []
    while cap.isOpened():
        frameId = cap.get(1)
        ret, frame = cap.read()
        if ret != True:
            break
        if frameId % ((int(frameRate)+1)*1) == 0:
            face_rects, scores, idx = detector.run(frame, 0)
            for i, d in enumerate(face_rects):
                x1 = d.left()
                y1 = d.top()
                x2 = d.right()
                y2 = d.bottom()
                crop_img = frame[y1:y2, x1:x2]
                data = img_to_array(cv2.resize(crop_img, (128, 128))).flatten() / 255.0
                data = data.reshape(-1, 128, 128, 3)
                temp.append(model.predict_classes(data))
    temp = mode([int(i) for i in temp])
    print(temp)
```

Fig 5.9 test data preparation

```
            frameId = cap.get(1)
            ret, frame = cap.read()
            if ret != True:
                break
            if frameId % ((int(frameRate) + 1) * 1) == 0:
                face_rects, scores, idx = detector.run(frame, 0)
                for i, d in enumerate(face_rects):
                    x1 = d.left()
                    y1 = d.top()
                    x2 = d.right()
                    y2 = d.bottom()
                    crop_img = frame[y1:y2, x1:x2]
                    data = img_to_array(cv2.resize(crop_img, (128, 128))).flatten() / 255.0
                    data = data.reshape(-1, 128, 128, 3)
                    y.append(model.predict_classes(data))
        templ = mode([int(i) for i in y])
        if templ==0:
          g=f.filename+"---"+"fake video"
        else:
          g=f.filename+"---"+"real video"
        return render_template("success.html", name=g)

import portpicker
port = portpicker.pick_unused_port()
from google.colab import output
output.serve_kernel_port_as_window(port)

from gevent.pywsgi import WSGIServer
host='localhost'
app_server = WSGIServer((host, port), app)
app_server.serve_forever()

if __name__ == '__main__':
    app.run(debug=True,use_reloader=False)
```

Fig 5.10 Flask Deployment

File.html

file.html

```
<html>
<head>
    <title>upload</title>
</head>
<body>
    <form action = "/success" method = "post" enctype="multipart/form-data" align="center">
        <input type="file" name="file" />
        <br>
        <br>
        <br>
        <input type = "submit" value="Upload">
    </form>
</body>
</html>
```

Fig 5.11 File.html

Success.html

success.html

```
<html>
<head>
<title>success</title>
</head>
<body>
<p>File uploaded and processed successfully</p>
<p>File Name: {{name}}</p>
</body>
</html>
```

Fig 5.12 success.html

5.4 Results / Output Screens

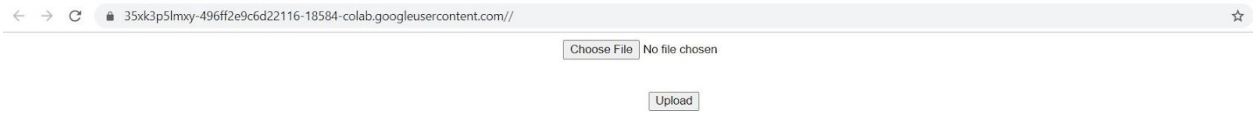


Fig 5.4.1 Landing Page

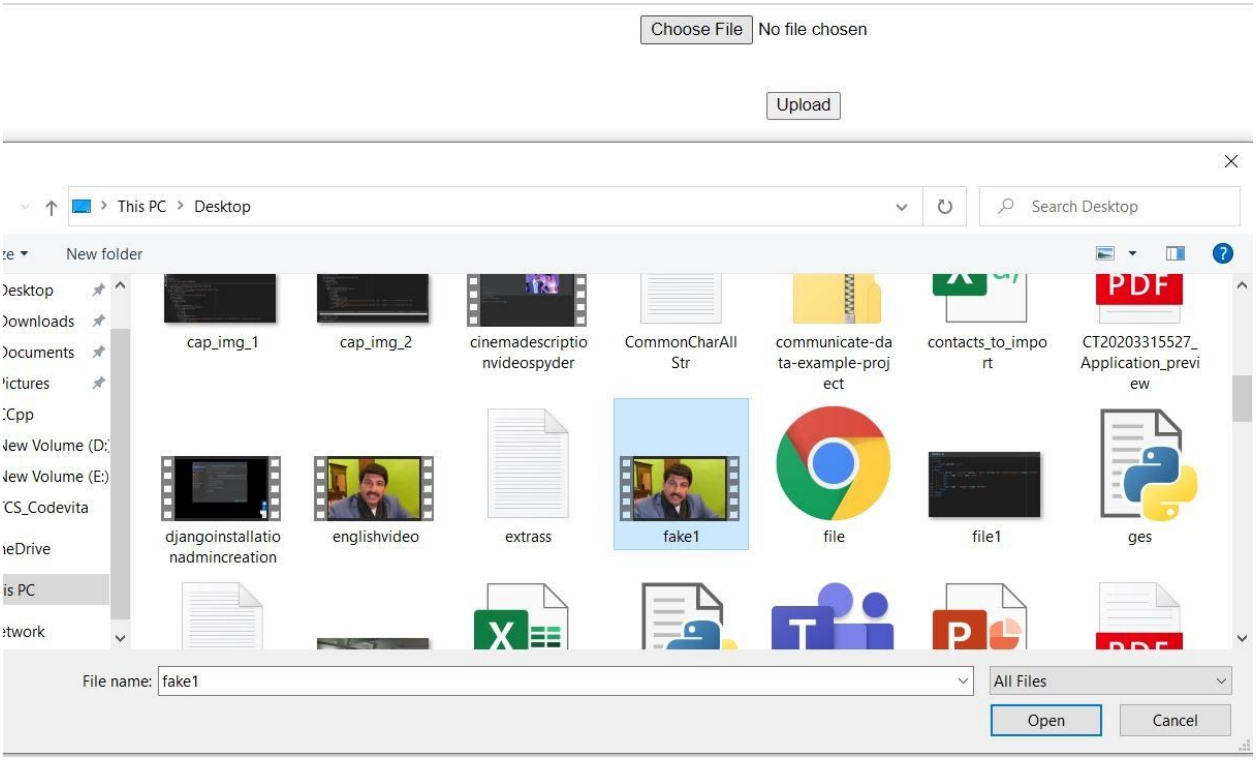


Fig 5.4.2 Browse video

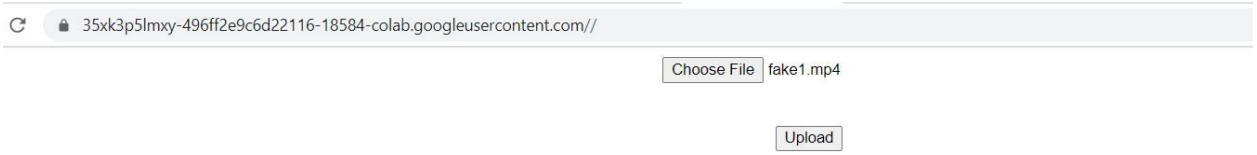


Fig 5.4.3 Upload video

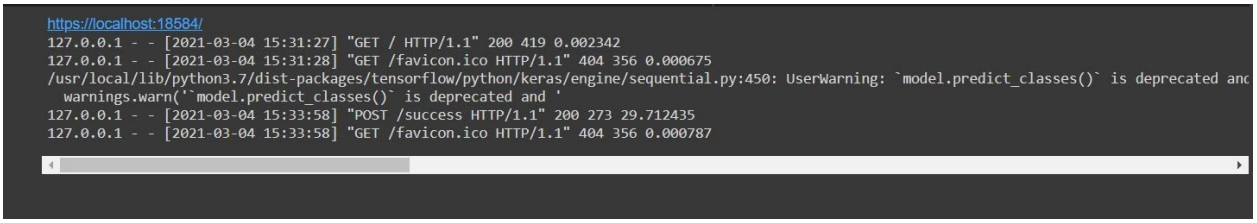


Fig 5.4.4 Running instance of flask

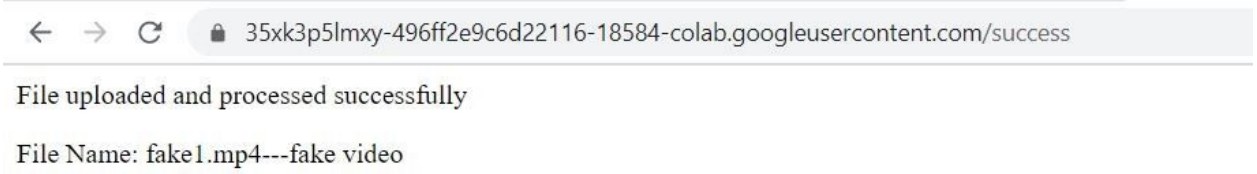


Fig 5.4.5 Fake video detected

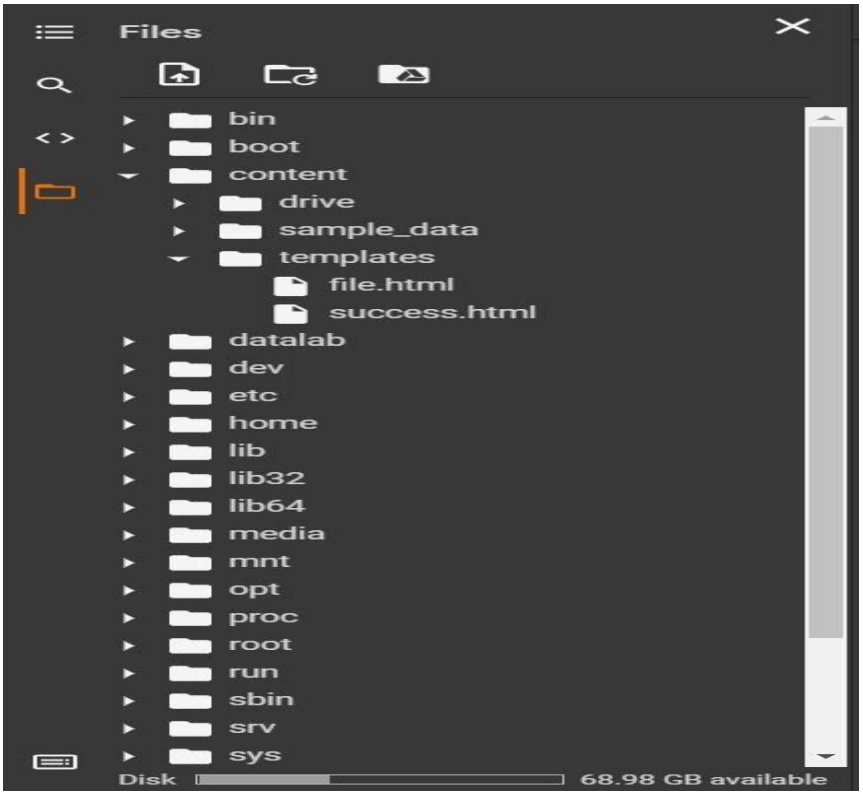


Fig 5.4.6 Files structure before execution

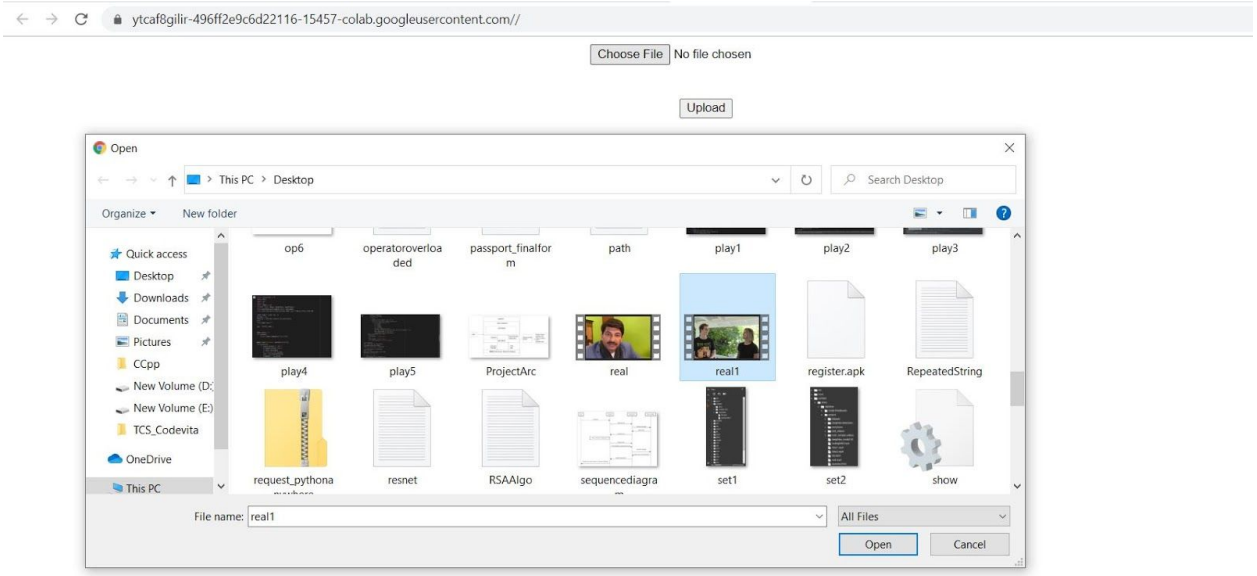


Fig 5.4.7 Browse video (ii)

Fig 5.4.8 Real Video detected

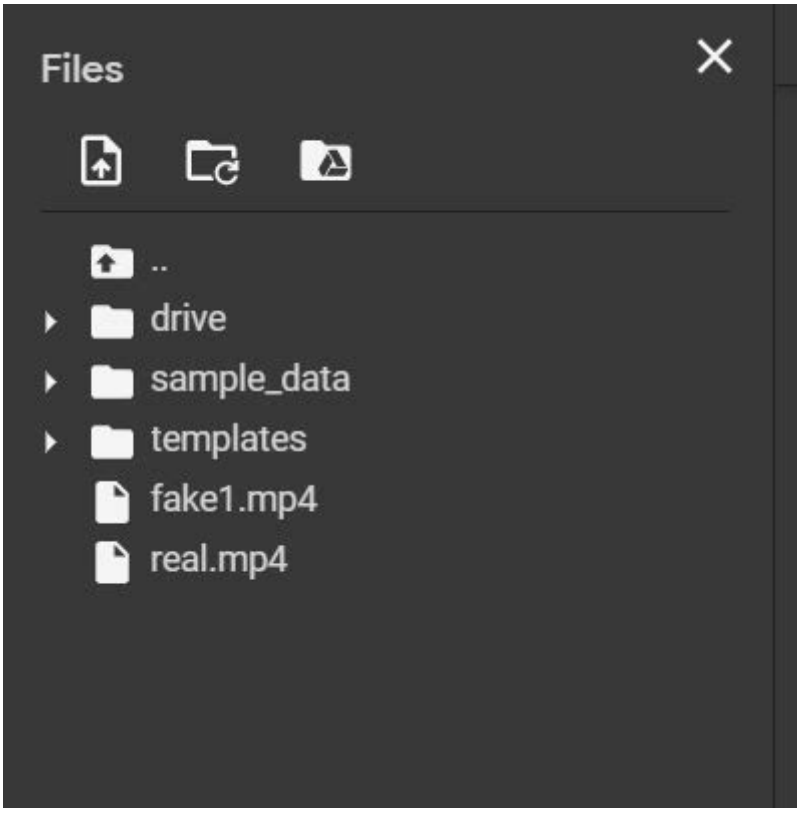


Fig 5.4.9 Files structure after execution

The files which are passed as input are saved with titles as fake or real in the project directory and can be stored in a database if database connectivity is provided.

6. TESTING

6.1 Types of Testing :

We have implemented integration testing and Regression testing methodologies for testing the working of the project. In the integration testing we tested the code for each module and then integrated them based on the hierarchy to top and tested the final code to verify it passes all of the test cases. We identified a few bugs in the final code and then we modified the code and retested it again and again until it satisfied all test cases.

6.2 Test Cases :

Test Scenario	Test Case	Expected Results	Actual Results	Pass / Fail
check browse functionality	check whether the browsing of file from PC is working	File browsed successfully	File Browsed Successfully	Pass
check upload functionality	check whether uploading of file from PC is working	File uploaded successfully	File uploaded successfully	Pass
check backend functionality	check whether results are being directed or displayed on webpage	File uploaded and processed successfully	File uploaded and processed successfully	Pass
check the inputs	check what happens on giving wrong input file format	Invalid file format	Invalid File format	Pass
check the inputs	check what happens on giving wrong file with correct extension	Invalid file format	File uploaded and processed successfully	Fail

Table 6.1 Test Case general template

7. CONCLUSION AND FUTURE SCOPE

Conclusion:

In the end, deep fakes using deep learning have increased a lot, they are extensively used these days. They are used in the fields like politics, education, artistic expression, criminal forensics, film production etc. There are both benefits and risks from this. In the field of politics, the pirate videos are created in different languages for campaigning in the different areas of that state, if the candidate doesn't know the languages spoken in that region.

The risks which arise over here is that photo morphing can be done and fake videos can be done, that is your face can be morphed with the face of a porn star. And different irrelevant videos can be made with the help of already existing real videos.

Deepfakes is a trending topic and it needs a lot of attention. Because it can cause havoc to the secrecy and integrity of one's life. Because once the fake videos are made they spread in the digital media and social media like a wildfire, they have to be detected as soon as possible inorder to stop the spreading of malicious content.

Our model that is inception_resnet model has given 91% accuracy which is better than many other fake and untrustworthy tools which are present online. The model is working fine and giving correct results when given different videos as input.

Future Scope:

As per today there are only a few features like poor lip syncing, blinking eyes, strange light effects, badly affected lightening, jewelry, bad image quality, inconsistent illuminations, reflections etc are used to find whether a video is fake or real.

The Aim of the future project is to find many more factors which determine the nature of the video and if the video is found to be fake then it should be reported and the ip address from which the video is uploaded should be found.

By doing this we have two advantages, one is the fake video which is uploaded is reported as spam or fake, then people can know that the video is not authentic and the second one is the fraud who has uploaded the video can be found easily and he can be suitably punished.

BIBLIOGRAPHY

References

1. https://www.tensorflow.org/tutorials/images/transfer_learning
2. https://keras.io/api/applications/
3. https://ai.googleblog.com/2016/08/improving-inception-and-image.html
4. https://keras.io/api/applications/inceptionresnetv2/
5. https://www.pyimagesearch.com/2018/06/18/face-recognition-with-opencv-python-and-deep-learning/

Data set links

1. https://www.kaggle.com/c/deepfake-detection-challenge/data
2. https://www.kaggle.com/c/deepfake-detection-challenge/data?select=train_sample_videos
3. https://www.kaggle.com/c/deepfake-detection-challenge/data?select=test_videos
4. https://ai.facebook.com/datasets/dfdc/
5. https://www.technologyreview.com/2019/09/25/132884/google-has-released-a-giant-database-of-deepfakes-to-help-fight-deepfakes/

APPENDIX-A: Building Neural Networks with Tensorflow & Keras

Deep neural networks are generally massive in nature with hundreds of layers, hence the term "deep" is used. Tensorflow is a python library which contains the modules and functions which allow the creation, building or modification of large neural networks. Keras is an API provided by Tensorflow which can be used to efficiently build large Neural Networks.

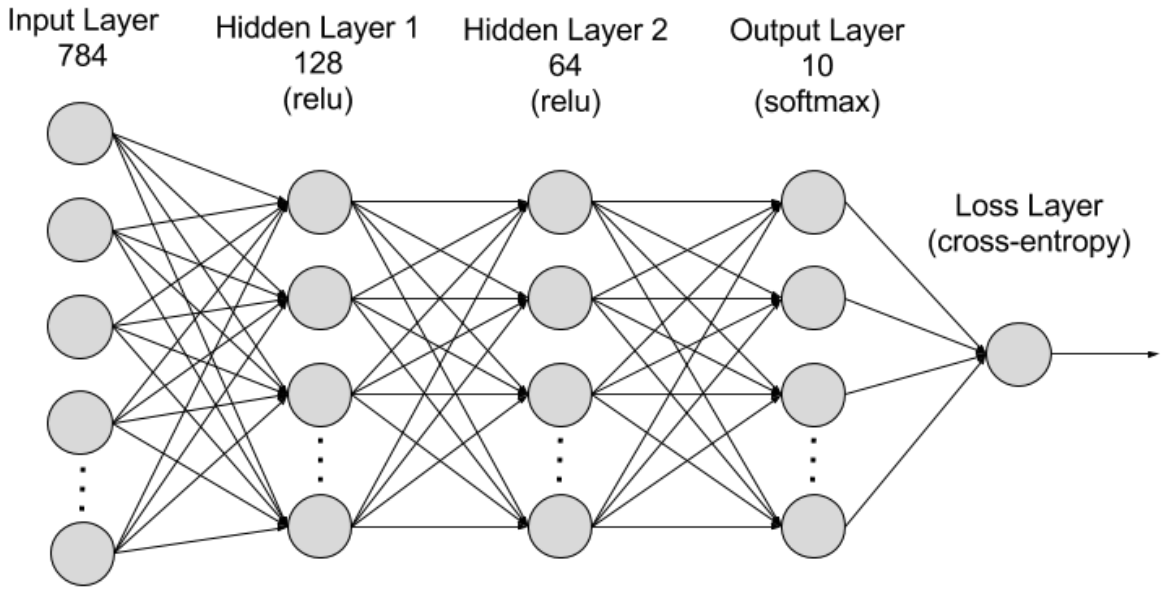
keras API can be used by importing *tensorflow.keras* from tensorflow. In keras, or generally any neural network consists of configurable and connecting blocks (nodes) called as layers. All the layers stacked upon each other forms a type of neural network model called Sequential Model. Sequential model can be implemented in tensorflow and keras as *tensorflow.keras.Sequential* . In general a neural network contains three types of layers: input layer, hidden layers and output layer.

input layer: tf.keras.Sequential method can be used to create a model with arguments as *tensorflow.keras.Layers.Flatten(input_shape=(, ,))* as one of the arguments in it. The tensorflow.keras.Layers.Flatten is used to know the input shape to manage input tensors, input_shape is used to specify the input tensors shape as a tuple.

hidden layers: input layer in a neural network contains a flatten layer which is used to flatten or normalize the input shape, whereas hidden layers contain dense layers (fully connected layers). In this layer, the number of neurons in the layer and the activation function used in neurons has to be specified. Creating a dense layer using tensorflow can be done using *tensorflow.keras.Layers.Dense(no_neurons , activation = ' ').*

output layer: output layer is also a dense layer which has the connection from all the hidden layers. output layer must give the output in the range of 0 to 1 and the sum of all the outputs from the outputs must be 1 (since they indicate the probability or confidence of the output). The activation function which can be used in this layer is *softmax.* output layer with 10 output neurons and softmax function as its activation function can be added using tensorflow as *tensorflow.keras.Layers.Dense(10, activation = 'softmax').*

Let's build a sample neural network with an input layer, 2 hidden layers and an output layer with a number of neurons and activation functions as shown in below figure.



sample_nn = tensorflow.keras.Sequential([

tensorflow.keras.layers.Flatten (input_shape=(28,28,1)),

tensorflow.keras.layers.Dense(128 ,activation='relu'),

tensorflow.keras.layers.Dense(64, activation='relu'),

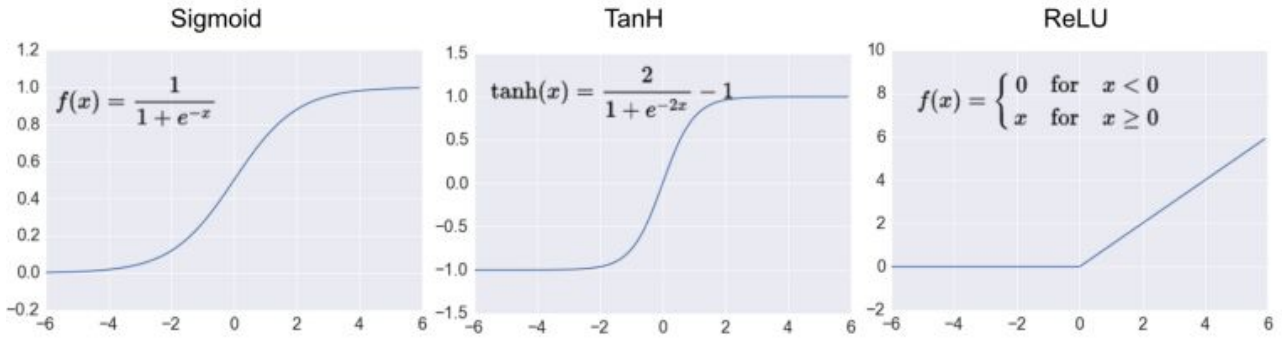
tensorflow.keras.layers.Dense(10, activation='softmax')])

A neural network with a sequential model will be created.

The summary of the model created can be obtained by using the method *model_name.summary()*.

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_1 (Flatten)          (None, 784)               0
_________________________________________________________________
dense_2 (Dense)              (None, 128)               100480
_________________________________________________________________
dense_3 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_4 (Dense)              (None, 10)                650
=================================================================
Total params: 109,386
Trainable params: 109,386
Non-trainable params: 0
```

Activation Functions :



Among the different nonlinear activation functions specified in the above figure, RELU is used extensively as activation function for the hidden layers.

A neural network has neurons, an activation function for each neuron, but the most important metrics of a neural network are weights and biases. weights and biases are the adjustable parameters which hold the knowledge gained by a neural network.

Weights and parameters of a model can be known by using the get_weights method. get_weights method returns a list of 2 numpy arrays: One array contains all the weights and another one contains the biases of the model.

weights_biases = sample_nn.get_weights()

We can even get the weights and biases of any layer specifically. For example to get the weights and biases of first layer of a sequential model as two separate arrays:

weights = sample_nn.get_layer(index=0).get_weights()[0]

biases = sample_nn.get_layer(index=0).get_weights()[0]

or

to get both of them as a single list we can use

weights_biases = sample_nn.get_layer(index=0).get_weights()

APPENDIX-B : TRANSFER LEARNING

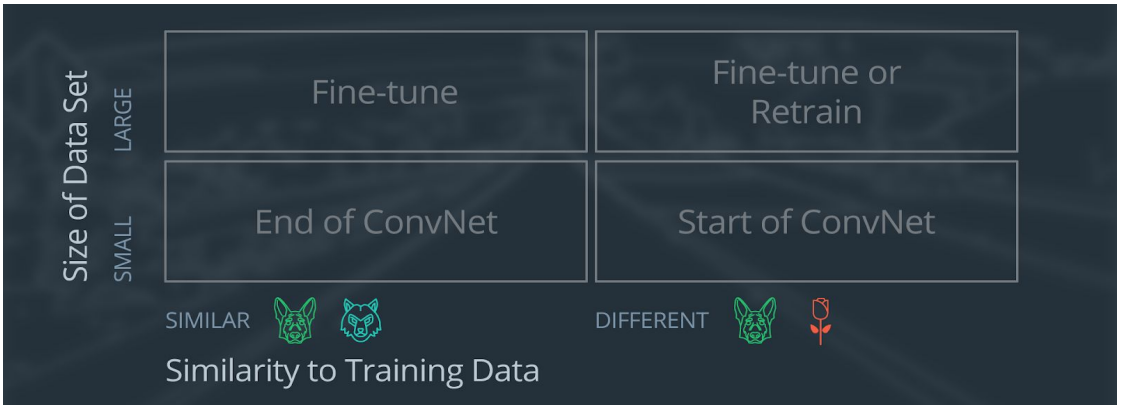
Transfer learning, a deep learning algorithm is used in this project. Transfer learning refers to storing the knowledge gained while solving one problem (trained on that problem data) and applying the stored knowledge on a different but similar problem. For example knowledge gained while recognizing handwritten digits can be used to recognize handwritten characters by slightly modifying the existing model and using the pre-gained knowledge.

Transfer learning approach involves referring a pre trained neural network and tweaking or modifying the neural network in order to use the knowledge gained on a new or different problem based on size of the dataset of new problem (small or large) and similarity of the data with respect to the original or pre-trained dataset.

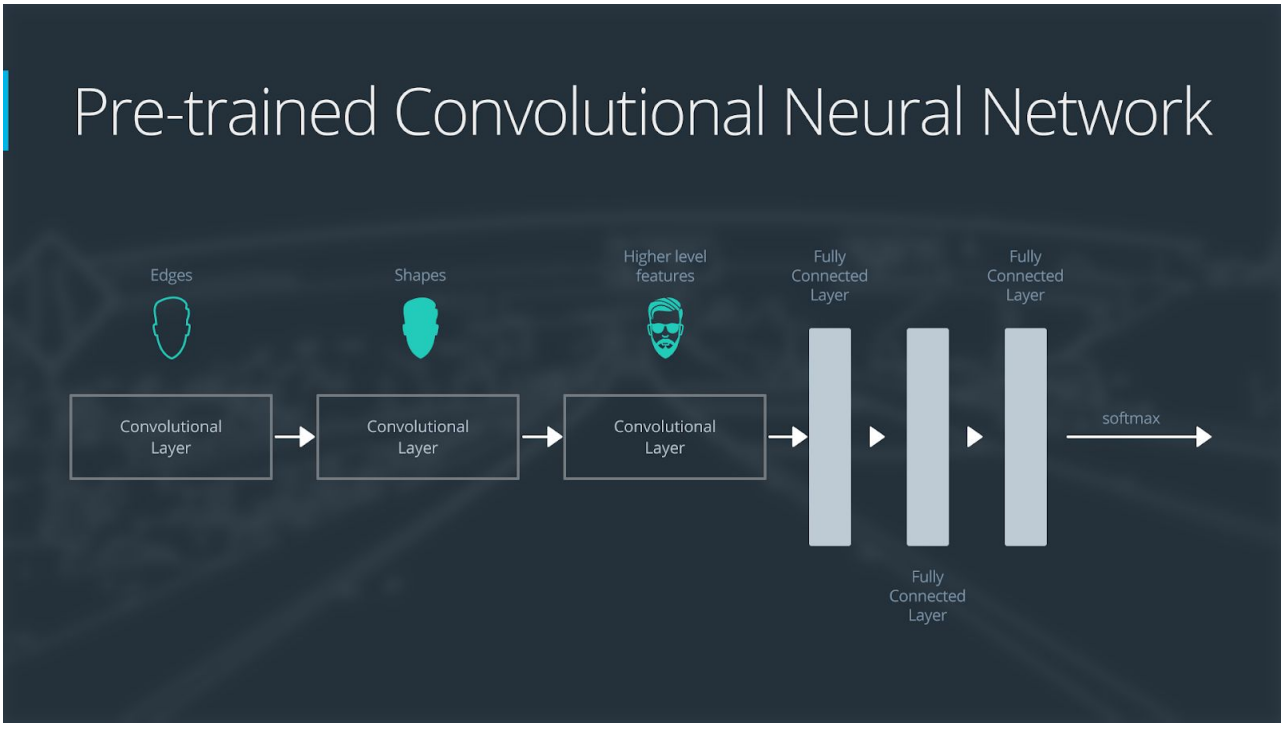
In general the steps in implementing the transfer learning approach are :

1. import the resources, load the data set and explore it
2. create pipeline: pretrained model might have some dimensionality and size of data and the new dataset might have different, hence the data is normalised or scaled to the requirements of the original data set or pre-trained model.
3. build the model by making necessary changes to the existing model such as adding layers
4. Compile the newly built model and fit the training data to train the model
5. Check the predictions on the new model

Based on the size and similarity of the dataset being used and pre trained data the existing or pre trained model can be fine tuned, or fine tuned and retrained, or add required layers such as global average pooling, dense etc at the end of the neural network, or add layers at the start of network.



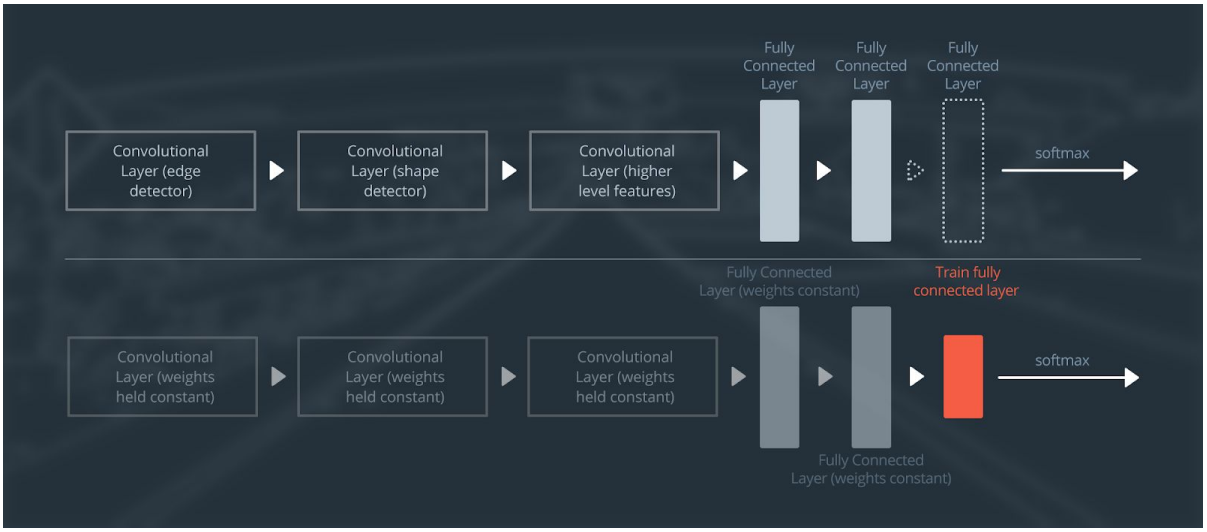
Let’s consider a sample convolutional neural network for demonstrating the four cases of transfer learning and the approaches to apply transfer learning.



Now let’s look into the different cases and the approach to apply transfer learning.

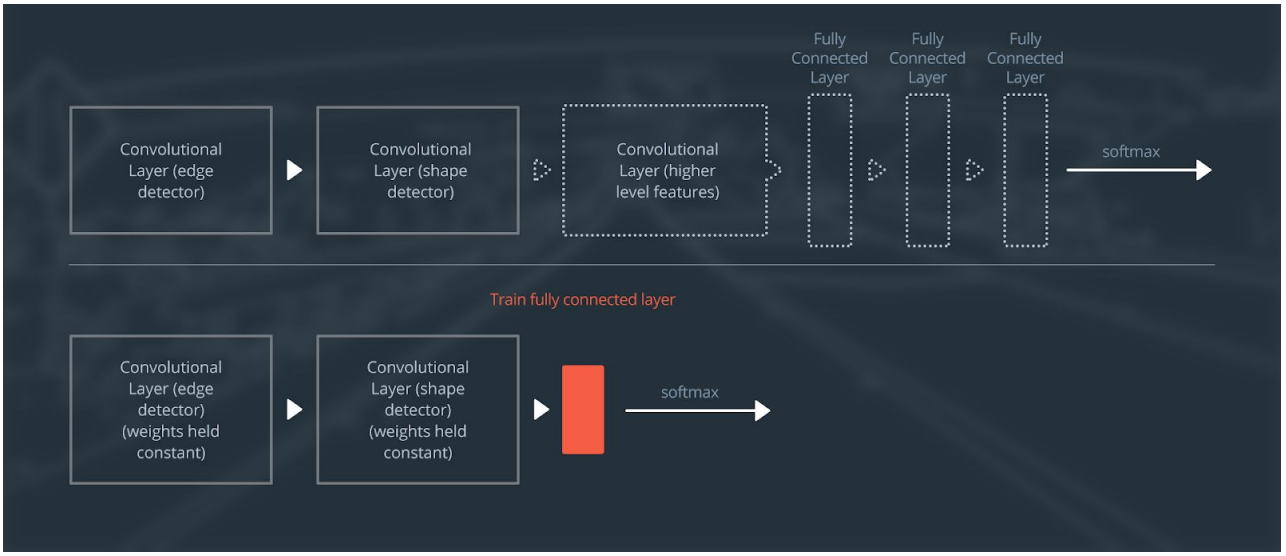
1. Small dataset with Similar data (to that of pretrained data):

Remove the last layer of pretrained network and add a new fully connected layer according to the requirements. Assign random weights to the newly added layer, freeze the old network layers weights and train the model on the new data again to update the newly random assigned weights.



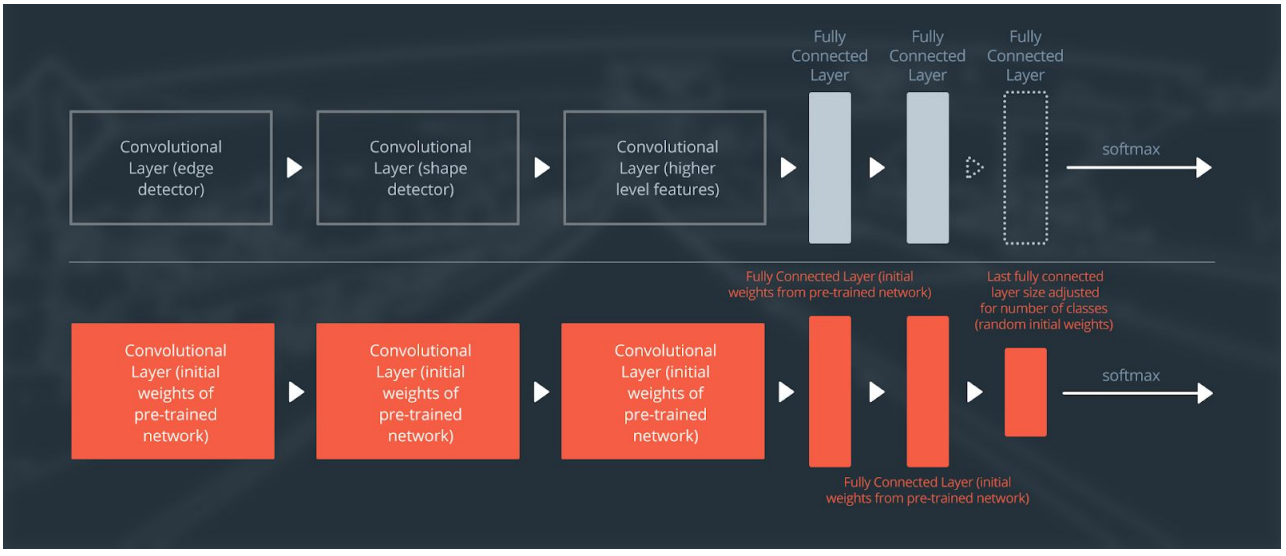
2. Small dataset with different data (to that of pretrained data):

Remove most of the beginning layers of the pre-trained network. Add a new fully connected layer to the existing pretrained layers. Assign random weights to the new fully connected layer freezing the weights of the pre-trained network. Train the model to update the weights of the new layer.



3. Large Dataset with similar data (to that of pretrained data):

Remove the last layer of the pre-trained network. Assign random weights to the new fully connected layer. Now train the entire network and update the weights of the entire network on new dataset.



4. Large Dataset with different data (to that of pretrained data):

Remove the last layer of the pre-trained network i.e, last fully connected layer. Now retrain the entire network on randomly initialised weights. In the previous case the weights were updated, but the pre-trained layers weights were not randomly initialised, instead they were updated from their values of the original pre-trained network. But here, the weights are randomly initialised and updated while training on the dataset. Alternatively, we can use the approach of previous case.