Mini Project Report

on

FAKE AND REAL FACE DETECTION USING DEEP LEARNING (USING SEQUENTIAL MODEL)

By

SAANVI MAHAPATRA (Reg. No.-201900424)

HARSHITA GAUTAM (Reg. No.-201900394)

KSHAMIKA GHIYA (Reg. No.-201900436)

Group Id – 03

In partial fulfilment of requirements for the award of degree in Bachelor of Technology in Computer Science and Engineering (2022)

Under the Project Guidance of

Mr. Sanjoy Ghatak

Assistant Professor 1, Department of CSE

Sikkim Manipal Institute of Technology, Majitar

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SIKKIM MANIPAL INSTITUTE OF TECHNOLOGY

(A constituent college of Sikkim Manipal University)

MAJITAR, RANGPO, EAST SIKKIM – 737136

PROJECT COMPLETION CERTIFICATE

This is to certify that the below mentioned students of Sikkim Manipal Institute of Technology have worked under my supervision and guidance from 17th January 2022 to 21th May 2022 and successfully completed the project entitled "FAKE AND REEAL FACE DETECTION USING DEEP LEARNING" in partial fulfilment of the requirements for the award of Bachelor of Technology in Computer Science and Engineering.

University Registration No	Name of Student	Course
201900424	**Saanvi Mahapatra**	**B.Tech (CSE)**
201900394	**Harshita Gautam**	**B.Tech (CSE)**
201900436	**Kshamika Ghiya**	**B.Tech (CSE)**

Sanjoy Ghatak

Assistant Professor 1

Department of Computer Science and Engineering

Sikkim Manipal institute of Technology

Majhitar, Sikkim – 737136

PROJECT REVIEW CERTIFICATE

This is to certify that the work recorded in this project report entitled **"Fake and Real Face Detection Using Learning (using sequential model)"** has been jointly carried out by **Saanvi Mahapatra (Reg. 201900424), Harshita Gautam (Reg. 201900394) and Kshamika Ghiya (Reg. 201900436)** of Computer Science & Engineering Department of Sikkim Manipal Institute of Technology in partial fulfilment of the requirements for the award of Bachelor of Technology in Computer Science and Engineering. This report has been duly reviewed by the undersigned and recommended for final submission for Mini Project Viva Examination.

Sanjoy Ghatak

Assistant Professor 1

Department of Computer Science and Engineering

Sikkim Manipal institute of Technology

Majhitar, Sikkim – 737136

CERTIFICATE OF ACCEPTANCE

This is to certify that the below mentioned students of Computer Science & Engineering Department of Sikkim Manipal Institute of Technology (SMIT) have worked under the supervision of **Mr. Sanjoy Ghatak** of Assistant Professor 1, Department of Computer Science and Engineering from **17th January 2022 to 21st May 2022** on the project entitled **"Fake and Real Face Detection Using Learning (using sequential model)".**

The project is hereby accepted by the Department of Computer Science & Engineering, SMIT in partial fulfilment of the requirements for the award of Bachelor of Technology in Computer Science and Engineering.

University Registration No	Name of Student	Project Venue
201900424	**Saanvi Mahapatra**	**SMIT**
201900394	**Harshita Gautam**	**SMIT**
201900436	**Kshamika Ghiya**	**SMIT**

Prof. (Dr.) Kalpana Sharma

Professor & Head of Department

Computer Science & Engineering Department

Sikkim Manipal Institute of Technology

Majhitar, Sikkim – 737136

DECLARATION

We, the undersigned, hereby declare that the work recorded in this project report entitled **"Fake and Real Face Detection Using Learning (using sequential model)"** in partial fulfilment for the requirements of award of B.Tech (CSE) from Sikkim Manipal Institute of Technology (A constituent college of Sikkim Manipal University) is a faithful and bonafide project work carried out at "**SIKKIM MANIPAL INSTITUTE OF TECHNOLOGY**" under the supervision and guidance of **Mr. Sanjoy Ghatak,** Assistant Professor, Department of Computer Science and Engineering.

The results of this investigation reported in this project have so far not been reported for any other Degree or any other Technical forum.

The assistance and help received during the course of the investigation have been duly acknowledged.

SAANVI MAHAPATRA (Reg. No.-201900424)

HARSHITA GAUTAM (Reg. No.-201900394)

KSHAMIKA GHIYA (Reg. No.-201900436)

ACKNOWLEDGMENT

We take this opportunity to acknowledge indebtedness and deep sense of gratitude to my guide **Mr. Sanjoy Ghatak** whose valuable guidance and kind supervision gave us throughout the course which shaped the present work as it shows.

We pay our deep sense of gratitude to **Prof. (Dr.) Kalpana Sharma, H.O.D, Computer Science & Engineering Department, Sikkim Manipal Institute of Technology** for giving us the opportunity to work on this project and provided all support required.

We are obliged to our project coordinators **Dr. Sandeep Gurung and Mr. Biraj Upadhyaya** for elevating, inspiration and supervising in completion of our project.

We would also like to thank any other staff of **Computer Science & Engineering Department, Sikkim Manipal Institute of Technology** for giving us continuous support and guidance that has helped us in completion of our project.

SAANVI MAHAPATRA (Reg. No.-201900424)

HARSHITA GAUTAM (Reg. No.-201900394)

KSHAMIKA GHIYA (Reg. No.-201900436)

DOCUMENT CONTROL SHEET

1	REPORT NUMBER	CSE/Mini Project/Internal/B.Tech/C/3/2022
2	TITLE OF THE REPORT	FAKE AND REAL FACE DETECTION USING DEEP LEARNING (USING SEQUENTIAL MODEL)
3	TYPE OF REPORT	Technical
4	AUTHOR	SAANVI MAHAPATRA (201900424) HARSHITA GAUTAM (201900394) KSHAMIKA GHIYA (201900436)
5	ORGANIZING UNIT	Sikkim Manipal Institute of Technology
6	LANGUAGE OF THE DOCUMENT	English
7	ABSTRACT	High reliability over a wide range of input sources, deep learning is swiftly becoming a major subset of machine learning. Convolutional neural networks are a great method for applying deep learning to photo classification (CNN). The Keras Python module simplifies the process of generating a CNN. Face detection might potentially be used to recognize faces in pictures, videos, and other media. A CNN model built with Keras sequential model is utilized to determine if a face is real or fake.
8	SECURITY CLASSIFICATION	General
9	DISTRIBUTION STATEMENT	General

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Deep learning is becoming a very popular subset of machine learning due to its high-level performance across many types of data. A great way to use deep learning to classify images is to build a convolutional neural network (CNN). The Keras library in Python makes it simple to build a CNN.

Fake faces are a problem in the real world today. It can be overcome by building a software program to detect fake and real faces in images. Face detection could also be a way of detecting faces from pictures, video footages, etc. There are various face detection algorithms, one of the most popular ones being the viola jones algorithm for object detection whose success rate is 78.4%.

In this paper, we will be using Sequential model made using CNN model of Keras to predict real or fake face. The model will be trained using Keras helper functions without defining a separate testing and validation set. By the end, the application will be able to predict if an entered image is fake or real.

INTRODUCTION

Computers see images using pixels. Pixels in images are the basic unit of programmable colour on a computer display or in a computer image. For example, a certain group of pixels may signify an edge in an image or some other pattern. Convolutions are a mathematical way of combining two signals to form a third signal, which can be used to help identify images. Convolution provides a way of multiplying together two arrays of numbers generally of different sizes, but of the same dimensionality, to produce a third array of numbers of the same dimensionality.

A convolution in this project, multiplies a matrix of pixels with a filter matrix or 'kernel' and sums up the multiplication values. Then the convolution slides over to the next pixel and repeats the same process until all the image pixels have been covered.

The model type which we will be using to build a convolutional neural network (CNN) is Sequential from Keras. Sequential model of keras allows to build a model layer by layer using the add () function. fit() function can be used to train the model with the following parameters: training data (train_X), target data (train_Y), validation data and the number of epochs.

The number of epochs is a hyperparameter that defines the number of times that the learning algorithm will work through the entire dataset. For example: one epoch means that each sample in the training dataset has had an opportunity to update the internal model parameters.

Steps per epoch is batches of samples to train. It is used to define how many batches of samples to use in one epoch. It is used to declaring one epoch finished and starting the next epoch.

Using a predict function, we will be able to see actual predictions made by our model for the test data. We can use the existing dataset, or we can input new data into the prediction function to see predictions our model makes on the new data.

The dataset is taken from Kaggle (https://www.kaggle.com/ciplab/real-and-fake-face-detection). This dataset contains 80,000 of expert-generated high-quality photoshopped face images. These images are composite of different faces, separated by eyes, nose, mouth, or whole face. In case of Generative Adversarial Networks (GAN) it is very easy to generate fake face images.

Figure 1. Dataset in a glance

This project contains:

1. Keras is used to build a convolutional neural network (CNN)
2. Matplotlib is used for visualization and analysis
3. The coding language used in this project is Python programming language.
4. The project has been coded on Google Colab
5. TensorFlow is used for backend development of application

LITERATURE SURVEY

SR. NO.	AUTHOR	PAPER AND PUBLICATION DETAILS	TITLE	RELEVANCE TO PROJECT
1	1) Zhengzhe Liu, Xiaojuan Qi1 2) Philip H. S. Torr	University of Oxford The University of Hong Kong	Global Texture Enhancement for Fake Face Detection In the Wild	Understanding fake images and enhancing fake face recognition in the right direction.
2	1) Ana Bertran, Huanzhou Yu 2) 2)Paolo Sacchetto	Stanford University	Face Detection Project Report	Build a highly efficient algorithm with the highest number of face detections and the lowest number of false alarms in terms of computational complexity
3	1) H.A. Rowley, Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA 2) S. Baluja, Department of Computer Science and the Robotics Institute, Carnegie Mellon University, USA 3) T. Kanade, Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA		Neural network-based face detection	The results of comparisons with various state-of-the-art face detection systems reveal that our system has comparable detection and false-positive rates.

Table 1. Literature Survey

PROBLEM DEFINITION

Detecting faces in a photograph is easily solved by humans, although has historically been challenging for computers given the dynamic nature of faces. For example, faces must be detected regardless of orientation or angle they are facing, light levels, clothing, accessories, hair colour, facial hair, makeup, age, and so on.

Over the past decade face detection and recognition have transcended from esoteric to popular areas of research in computer vision and one of the better and successful applications of image analysis and algorithm-based understanding. Because of the intrinsic nature of the problem, computer vision is not only a computer science area of research, but also the object of neuroscientific and psychological studies also, mainly because of the general opinion that advances in computer image processing and understanding research will provide insights into how our brain work and vice versa. A general statement of the face recognition problem (in computer vision) can be formulated as follows: given still or video images of a scene, identify or verify one or more persons in the scene using a stored database of faces.

SOLUTION STRATEGY

The challenge of fake face detection is not only posed to the biometric systems but also to the general media perception on social media. Thus, it is of paramount importance to detect faked face representations to reduce the vulnerability of biometrics systems and to reduce the impact of manipulated social media content.

In the project:

1. Data gets augmented by ImageDataGenerator function offered by the Keras API.
2. Data is augmented while minimally supplementing the data using rotation, zooms, mirroring, and shifts to ensure that the data is not skewed.
3. Convolutional Neural Network is considered as the methodology that is used with data augmentation in this research.
4. The augmented dataset is fed into the defined CNN model in order to predict the class.

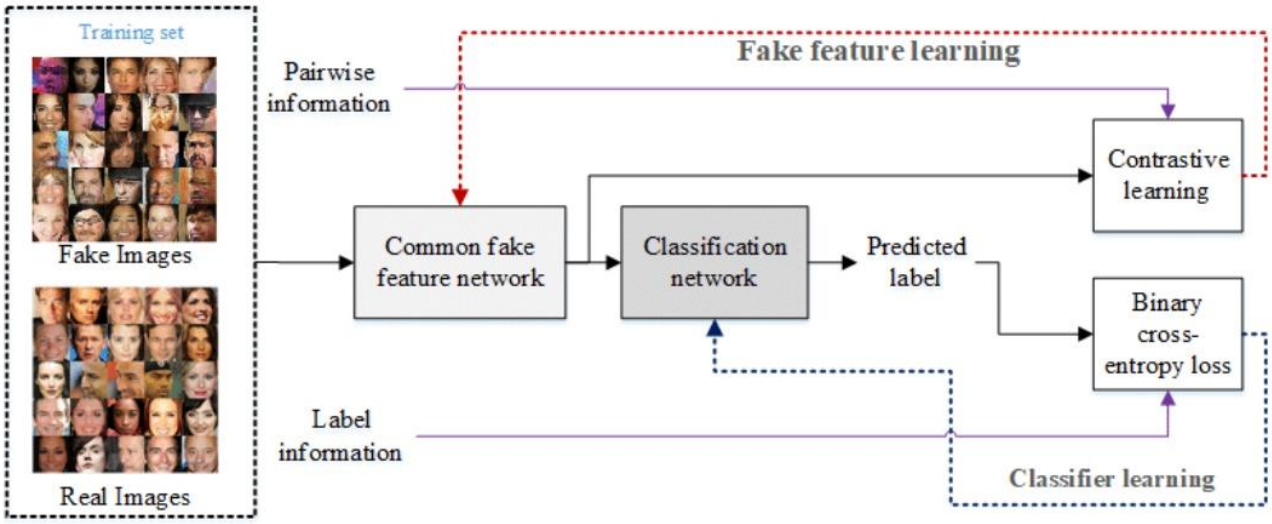

Figure 2. Flowchart of the proposed fake face detectors based on the proposed common fake feature network with the two-step learning approach

BLOCK DIAGRAM

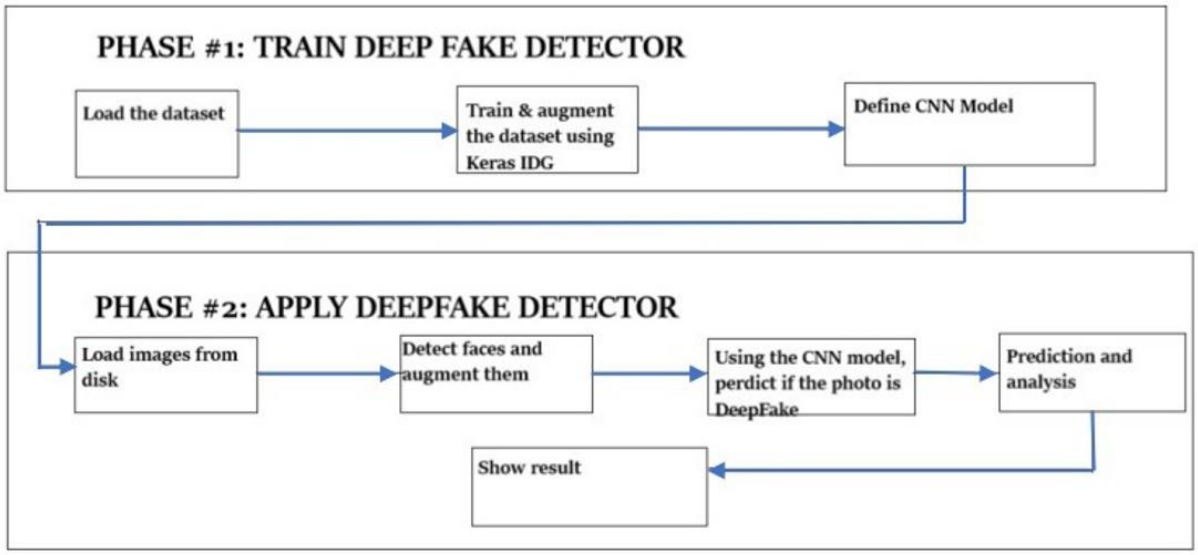

Figure 3. Block diagram for Training and Testing

There are mainly two phases to the project. In the first phase, the dataset is loaded onto jupyter notebook. Then the images in the dataset are trained and augmented. Then a CNN model for prediction and analysis is built. In the second phase of the project, the detector is applied to an image and the result given by the detector is checked.

IMPLEMENTATION DETAILS

Input: Face images

Output: Distinguish between real and fake image

STEPS:

1. Start.
2. Read the datasets (test and training) using Keras ImageDataGenerator while minimally supplementing the data using rotation, zooms, mirroring, and shifts to ensure that the data is not skewed.
3. Plot a histogram showing number of instances w.r.t class.
4. From the training set print 16 images in cmp = grey with defined units and measurements, along with title “label”

   *2, 3, 4 → Making the data reasonable)

5. Define a CNN model using Sequential algorithm to train and use. Compile the model using metric accuracy.
6. Train model using Keras helping function, without separate testing and validation set.

RESULT AND DISCUSSION

1. Import necessary libraries.

```
from keras.models import Sequential
from keras.layers import Conv2D
from keras.layers import Flatten
from keras.layers import Dense
from keras.layers import MaxPool2D
import numpy as np
from keras.preprocessing import image
from keras.callbacks import EarlyStopping, ModelCheckpoint
import tensorflow as tf
from keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt
%matplotlib inline
```

```
[2]
try:
    from tensorflow.python.util import module_wrapper as deprecation
except ImportError:
    from tensorflow.python.util import deprecation_wrapper as deprecation
deprecation._PER_MODULE_WARNING_LIMIT = 0
```

Figure 4. Code snippet 1

2. Using keras ImageDataGenerator, read the datasets (test and training) while slightly augmenting the data with rotation, zooms, mirroring, and shifts, which makes sure that the data is not biased.

   In the following piece of code, the following is performed on the dataset: re-shift, rotate, rescale, zoom the images. Moreover, the data is augmented using width shift range and height shift range.

   Augmenting the data means that there will be various modified copies of the data. It mainly helps in reducing overfitting when training a model.

```
[3]
nbatch = 128
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=10.,
    width_shift_range=0.1,
    height_shift_range=0.1,
    zoom_range=0.2,
    horizontal_flip=True)

test_datagen = ImageDataGenerator(rescale = 1./255)

training_set = train_datagen.flow_from_directory('dataset/training',
                                                 target_size=(128,128),
                                                 batch_size =nbatch,
                                                 class_mode = 'binary')

test_set = test_datagen.flow_from_directory('dataset/test',
                                            target_size=(128,128),
                                            batch_size =nbatch,
                                            class_mode = 'binary')
```

```
Found 817 images belonging to 3 classes.
Found 311 images belonging to 3 classes.
```

Figure 5. Code Snippet 2

3. Plot a histogram showing number of instances w.r.t class for the training set and the test set.

   In the following histogram diagram, the x-axis represents class and the y-axis represents the number of instances. The blue zone shows the number of training sets and the red zone shows the test set.

```
h1 = plt.hist(training_set.classes, bins=range(0,3), alpha=0.8, color='blue', edgecolor='black')
h2 = plt.hist(test_set.classes,  bins=range(0,3), alpha=0.8, color='red', edgecolor='black')
plt.ylabel('# of instances')
plt.xlabel('Class')
```

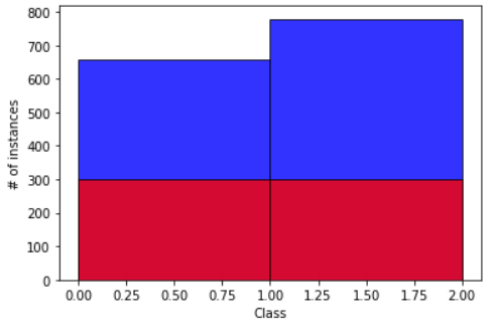

Figure 6. Code Snippet 3

4. From the training set print 16 images in cmp = grey with defined units and measurements, along with title “label”

   In this piece of code, the colour of the images is changed to black and white, and the photo size is changed to 16x16 pixels, so that all the images are of the same size. In the same way the axis, subplot, and the title are added to the images. 16 images are printed/shown as an example.

```
[5]  1 for X, y in training_set:
     2   print(X.shape, y.shape)
     3   plt.figure(figsize=(16,16))
     4   for i in range(16):
     5     plt.subplot(4,4,i+1)
     6     plt.axis('off')
     7     plt.title('Label: ')
     8     img = np.uint8(255*X[i,:,:,0])
     9     plt.imshow(img, cmap='gray')
    10   break
```

Figure 7. Code Snippet 4

5. Define a CNN model to train and use.

   The model type used in this project is Sequential model. It is the easiest way to build a model in Keras. It allows to build a model layer by layer.

   - Add() function adds layers to the model.
   - Convo2D layers deal with input images, which are seen as 2-dimensional matrices, 64 nodes in the first layer and 32 nodes in each layer. This can be changed depending on size of dataset.
   - Kernel size is the size of the filter matrix for convolution.
   - Dense is a standard layer type that is used in many cases for neural networks.
   - Flatten is a layer in between the Convo2D and dense layer. Flatten layer serves as a connection between the convolution and dense layers.
   - Activation function used in dense layer are RELU- Rectified Linear Activation and sigmoid.

```
model = Sequential()

model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=(128,128,3)))

model.add(MaxPool2D(pool_size=(2,2)))

model.add(Conv2D(64, kernel_size=(3, 3),
                 activation='relu'))

model.add(MaxPool2D(pool_size=(2,2)))
model.add(Conv2D(128, kernel_size=(3, 3),
                 activation='relu'))

model.add(MaxPool2D(pool_size=(2,2)))

model.add(Flatten())

model.add(Dense(activation="relu",
                units=256))

model.add(Dense(activation="sigmoid",
                units=1))

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 126, 126, 32)      896

max_pooling2d (MaxPooling2D  (None, 63, 63, 32)        0
)

conv2d_1 (Conv2D)            (None, 61, 61, 64)        18496

max_pooling2d_1 (MaxPooling  (None, 30, 30, 64)        0
2D)

conv2d_2 (Conv2D)            (None, 28, 28, 128)       73856

max_pooling2d_2 (MaxPooling  (None, 14, 14, 128)       0
2D)

flatten (Flatten)            (None, 25088)             0

dense (Dense)                (None, 256)               6422784

dense_1 (Dense)              (None, 1)                 257

=================================================================
Total params: 6,516,289
Trainable params: 6,516,289
Non-trainable params: 0
```

Figure 8. Code Snippet 5

6. Compile the CNN model using ‘adam’ optimiser, loss of ‘binary crossentropy’, and metric accuracy.

```
[7]  1 model.compile(optimizer = 'adam',
     2              loss = 'binary_crossentropy',
     3              metrics = ['accuracy'])
```

Figure 9. Code Snippet 6

7. Creating a callback list. A callback is a set of functions to be applied at given stages of the training procedure. Callbacks are used to get a view on internal states and statistics of the model during training.

```
callbacks_list = [
   EarlyStopping(monitor='val_loss', patience=10),
   ModelCheckpoint(filepath='model_checkpoint.hdf5', monitor='val_loss', save_best_only=True, mode ='max'),
]
```

Figure 10. Code Snippet 7

8. Train model using keras helping function, without separate testing and validation set.

```
history = model.fit_generator(
      training_set,
      steps_per_epoch=65,
      epochs=50,
      validation_data=test_set,
      validation_steps=28,
      callbacks = callbacks_list
  )
```

```
Epoch 1/50
65/65 [==============================] - 791s 12s/step - loss: 0.9067 - accuracy: 0.5695
Epoch 2/50
65/65 [==============================] - 463s 7s/step - loss: 0.7532 - accuracy: 0.5929
Epoch 3/50
65/65 [==============================] - 468s 7s/step - loss: 0.7176 - accuracy: 0.5982
Epoch 4/50
65/65 [==============================] - 470s 7s/step - loss: 0.6942 - accuracy: 0.5929
Epoch 5/50
65/65 [==============================] - 467s 7s/step - loss: 0.6997 - accuracy: 0.6046
Epoch 6/50
65/65 [==============================] - 466s 7s/step - loss: 0.6622 - accuracy: 0.6363
Epoch 7/50
65/65 [==============================] - 468s 7s/step - loss: 0.6702 - accuracy: 0.6319
Epoch 25/50
65/65 [==============================] - 462s 7s/step - loss: 0.2112 - accuracy: 0.9371
Epoch 26/50
65/65 [==============================] - 459s 7s/step - loss: 0.2054 - accuracy: 0.9400
Epoch 27/50
65/65 [==============================] - 463s 7s/step - loss: 0.1767 - accuracy: 0.9391
Epoch 28/50
65/65 [==============================] - 465s 7s/step - loss: 0.1855 - accuracy: 0.9449
Epoch 29/50
65/65 [==============================] - 465s 7s/step - loss: 0.1847 - accuracy: 0.9449
Epoch 30/50
65/65 [==============================] - 464s 7s/step - loss: 0.1576 - accuracy: 0.9542
Epoch 31/50
65/65 [==============================] - 463s 7s/step - loss: 0.1601 - accuracy: 0.9576
Epoch 32/50
65/65 [==============================] - 462s 7s/step - loss: 0.1336 - accuracy: 0.9605
Epoch 43/50
65/65 [==============================] - 474s 7s/step - loss: 0.0877 - accuracy: 0.9761
Epoch 44/50
65/65 [==============================] - 468s 7s/step - loss: 0.0837 - accuracy: 0.9746
Epoch 45/50
65/65 [==============================] - 467s 7s/step - loss: 0.0837 - accuracy: 0.9761
Epoch 46/50
65/65 [==============================] - 468s 7s/step - loss: 0.0814 - accuracy: 0.9761
Epoch 47/50
65/65 [==============================] - 470s 7s/step - loss: 0.0789 - accuracy: 0.9810
Epoch 48/50
65/65 [==============================] - 468s 7s/step - loss: 0.0970 - accuracy: 0.9815
Epoch 49/50
65/65 [==============================] - 469s 7s/step - loss: 0.0743 - accuracy: 0.9834
Epoch 50/50
65/65 [==============================] - 471s 7s/step - loss: 0.0750 - accuracy: 0.9805
```

Figure 11. Code Snippet 8

9. Fake will be 1 and real will be 2 as class indices

```
[11]  1 training_set.class_indices
```

```
{'.ipynb_checkpoints': 0, 'fake': 1, 'real': 2}
```

Figure 12. Code Snippet 9

10. Plot the train test models (loss to no. of epochs and accuracy to no. of epoch)

    The following snippet of code produces 2 graphs, where the blue line indicates the test set and the red line indicates the training set.

    a) Loss to the number of epochs

    b) Accuracy to the number of epochs

```
plt.figure(figsize=(16,6))
plt.subplot(1,2,1)
nepochs=len(history.history['loss'])
plt.plot(range(nepochs), history.history['loss'],     'r-', label='train')
plt.plot(range(nepochs), history.history['val_loss'], 'b-', label='test')
plt.legend(prop={'size': 20})
plt.ylabel('loss')
plt.xlabel('# of epochs')
plt.subplot(1,2,2)
plt.plot(range(nepochs), history.history['acc'],     'r-', label='train')
plt.plot(range(nepochs), history.history['val_acc'], 'b-', label='test')
plt.legend(prop={'size': 20})
plt.ylabel('accuracy')
plt.xlabel('# of epochs')
```

```
Text(0.5, 0, '# of epochs')
```

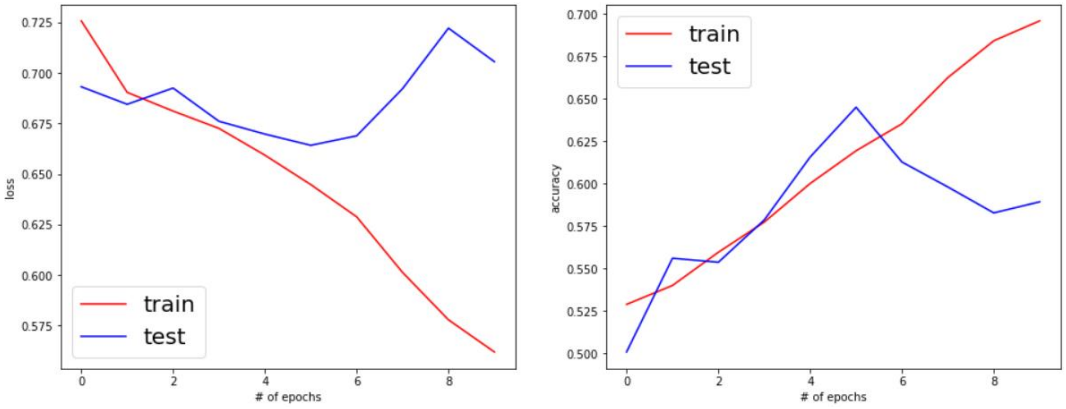

Figure 13. Code Snippet 10

11. Check the model by entering images of own.

    A function ImagePrediction is defined which takes an input about the location of an image and predicts if the face in the image is real or fake.

```
def ImagePrediction(loc):
    test_image = image.load_img(loc, target_size = (128,128))
    plt.axis('off')
    plt.imshow(test_image)
    test_image = image.img_to_array(test_image)
    test_image = np.expand_dims(test_image, axis =0)
    result = model.predict(test_image)
    if result[0][0] == 1:
        predictions = 'Real'
    else:
        predictions = 'Fake'
    print('Prediction: ',predictions)
```

```
img = input("Enter Location of Image to predict: ")
test_image_1 = ImagePrediction(img)
```

```
Enter Location of Image to predict: dataset/face_pred/check.jpg
Prediction:  Real
```

Fig: 7, Real Face

```
Enter Location of Image to predict: dataset/face_pred/check2.jpg
Prediction:  Fake
```

Fig: 8, Fake Face

Figure 14. Code Snippet 11

LIMITATIONS OF THE PROJECT

Generalizability

The great majority of existing face manipulation detection algorithms have been tested on a single database or on recognised forms of face manipulations. To put it another way, the reported empirical findings demonstrated detector performance under the identical train and test manipulation type/database. However, when assessed in a cross-manipulation situation, their performance generally suffers dramatically (where train and test sets are not from the same manipulation type or database). As a result, published detection rates are too optimistic.

Interpretability

Few research has attempted to investigate the interpretability and trustworthiness of face manipulation detectors till now. Due to the black box nature of deep learning approaches, many detection methods, particularly those based on deep neural networks, lack explainability. Understanding and trusting the opinion of a false detector is critical for humans; nevertheless, the human expert is at the end of the processing chain and so wants to understand the conclusion. Some crucial applications, like as journalism and law enforcement, will not accept a numerical score or label that is not backed up by sound reasoning and insights.

Furthermore, it is not easy to characterize the intent of a face manipulation. So far, learning-based detectors cannot distinguish between malicious and benign processing operations. For example, it is impossible to tell if a change of illumination was carried out only for the purpose of enhancement or to better fit a swapped face in a video. In general, characterizing the intent of a manipulation is extremely difficult and it will become even harder with the spread of deep learning-based methods in almost all the enhancement operations. In a world where most of the media are processed using deep learning-based tools, it is increasingly likely that something be manipulated, and the key to forensic performance is learning the intent behind a manipulation.

Vulnerabilities

State-of-the-art detection methods make heavy use of deep learning, i.e., deep neural network models serve as the most popular backbone. Prior studies have demonstrated that detectors based on neural networks are most susceptible to adversarial attacks

FUTURE SCOPE

Deepfake detection has several possible future applications. They are classified into two types: blacklist applications and whitelist apps. Blacklist applications include security and surveillance applications, as well as criminal identification applications. Other apps, such as attendance tracking and access control, are designated as whitelist applications.

Due to the sheer benefits, it offers over previous surveillance techniques like as biometrics, facial recognition has grown in favour in recent years. The face recognition sector is fast growing into new industries such as government, healthcare, security, retail, marketing, airport boarding, entertainment, and many more. The car industry is capitalising on facial recognition's potential by putting it into smart autos that start only after recognising the driver. Furthermore, networking apps employ this strategy to connect people who share similar features.

It is used in offices, government, real-estate, manufacturing for Control and record employee, visitor, vendor, and maintenance access to places.

Detecting and preventing ATM fraud. In India, a database of all ATM cardholders can be created, and facial recognition systems installed. So, every time a user enters an ATM, his image is captured and compared to a stored photo in the database to provide access.

In India, duplicate voters are being reported. This can be overcome by using fake face detectors.

This technology may also be used to verify passports, visas, driver’s license. It may be utilised to improve monitoring and security in the defence ministry, airports, and other key locations. It may also be utilised by the police force to identify offenders.

CONCLUSION

DeepFake technologies' rapid development has posed significant challenges to the authenticity of video content. It is critical to develop DeepFake detection methods, with three-dimensional (3D) convolution neural networks (CNN) attracting widespread interest and achieving satisfactory results. However, there are only a few 3D CNNs designed for DeepFake detection, and their parameters are large, resulting in high memory and storage consumption. In this project we used a simple Sequential model of CNN for detection of real and fake face images.

The number of epochs we run is **50** with steps per epoch as **65** on a large dataset of **80,000** images which gives an accuracy of **0.98**, which means that when we run our code on 4 input images, 3 out of 4 images are detected correctly as fake/real face.

With this we can conclude that the sequential model is one of the optimal ways to detect fake and real face images.

This project can be further developed into a software for fake and real face detection for live video feeds.

GANTT CHART

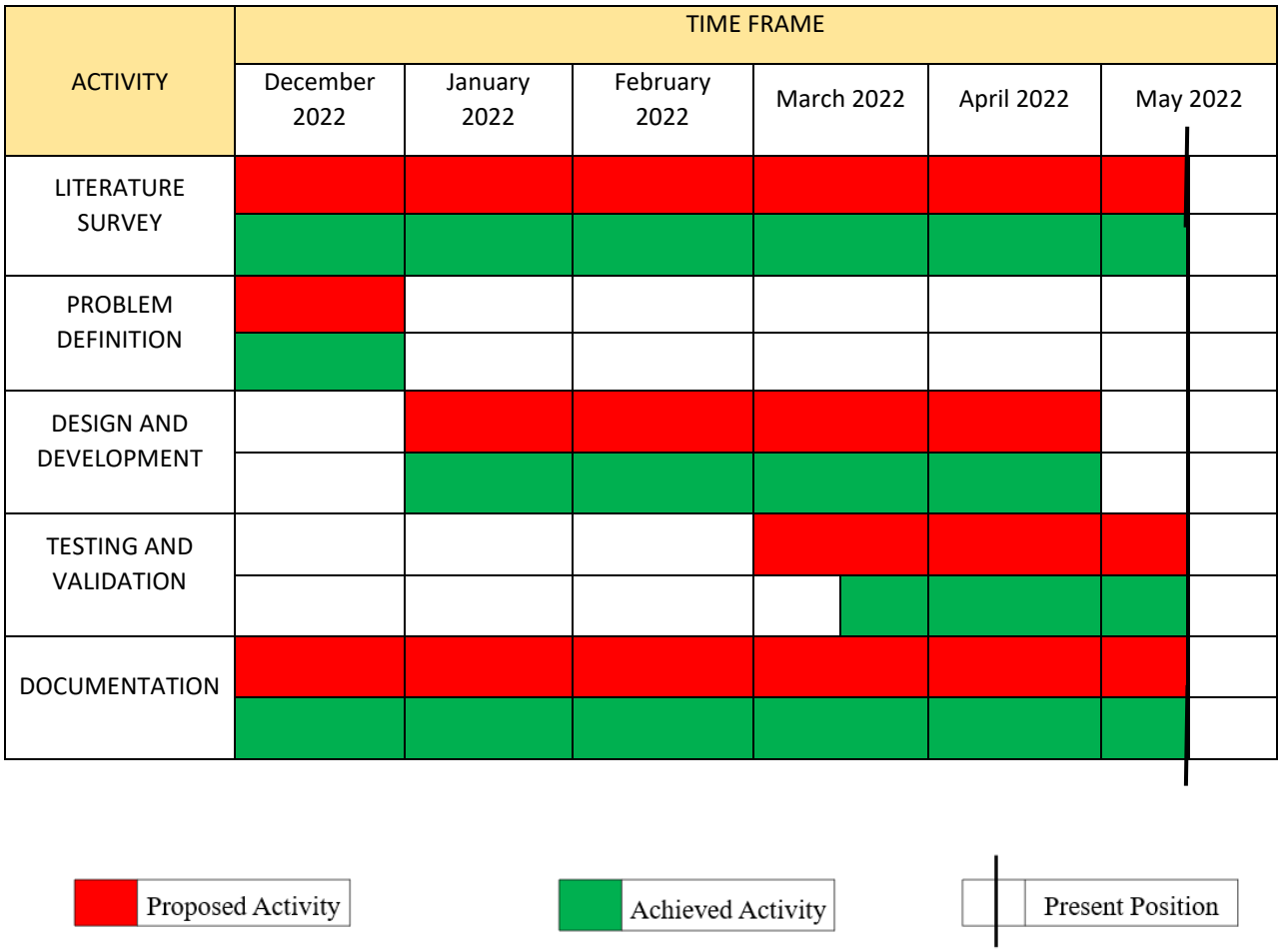

Figure 15. Gantt chart

REFERENCE

[1] S. Baluja, "Population-Based Incremental Learning: A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning", 1994.

[2] G. Burel and C. Carel, "Detection and Localization of Faces on Digital Images", Pattern Recognition Letters, vol. 15, pp. 963-967, Oct. 1994.

[3] A.J. Colmenarez and T.S. Huang, "Face Detection With Information-Based Maximum Discrimination", Computer Vision and Pattern Recognition, pp. 782-787, 1997.

[4] C. Frankel, M.J. Swain and V. Athitsos, "WebSeer: An Image Search Engine for the World Wide Web", Aug. 1996.

[5] V. Govindaraju, "Locating Human Faces in Photographs", Int'l J. Computer Vision, vol. 19, no. 2, pp. 129-146, 1996.

[6] H.M. Hunke, Locating and Tracking of Human Faces With Neural Networks, 1994.

[7] Y. Le Cun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, et al., "Backpropagation Applied to Handwritten Zip Code Recognition", Neural Computation, vol. 1, pp. 541-551, 1989.

[8] T.K. Leung, M.C. Burl and P. Perona, "Finding Faces in Cluttered Scenes Using Random Labeled Graph Matching", Proc. Fifth Int'l Conf. Computer Vision, pp. 637-644, 1995-June.