INSTITUTE FOR ADVANCED
COMPUTING AND
SOFTWARE
DEVELOPMENT
AKURDI, PUNE

Documentation On

"Fake Image Detection"

PG-DBDA SEP 2022

Submitted By:

Group No: 6

Khadke Tushar Sheshrao 229316

Niket Pravinkumar Chaudhari 229324

Mr. Rohit Puranik
Centre Coordinator

Dr. Shantanu Pathak
Project Guide

ACKNOWLEDGEMENT

This project **'Fake Image Detection'** was a great learning experience for us and we are submitting this work to Institute for Advanced Computing and Software Development.

We all are very glad to mention the name of Dr. Shantanu Pathak for his valuable guidance to work on this project. The guidance and support from him helped us to overcome various obstacles and intricacies during the course of project work.

Our most heartfelt gratitude goes to Mr. Rohit Puranik (Centre Coordinator) who gave us all the required support and coordinated with us to provide all the necessities we needed to complete the project and throughout the course up to the last day here in IACSD, Akurdi.

From

Khadke Tushar Sheshrao

Niket Pravinkumar Chaudhari

Index

Figures

1. Introduction

1.1 PROBLEM STATEMENT

The problem statement for fake image detection is to develop a machine learning model that can accurately identify whether an image is real or fake.

1.2 Abstract

Fake image detection is a rapidly developing field of research that seeks to distinguish between genuine and manipulated images. With the rise of digital photography and image editing software, the spread of fake images has become a growing problem in various fields, including journalism, social media, and law enforcement. To address this challenge, researchers have explored various approaches to detect fake images, including forensic techniques, machine learning, and deep learning. We review the different types of fake images, the challenges involved in detecting them, and the evaluation metrics commonly used to assess detection performance. Ultimately, this paper aims to provide a comprehensive understanding of the current state of fake image detection research, as well as directions for future work in this important area.

1.3 Product Scope

The product scope for fake image detection should be designed to meet the needs of the intended users and provide effective and efficient detection of fake images in their specific context.

1.4 Aims & Objectives

To identify and prevent the spread of fake images that can be harmful to individuals, communities, or society at large. To promote trust and transparency in digital media by detecting and exposing manipulated images. To develop and validate a detection algorithm or model that can accurately distinguish between genuine and fake images. To test and evaluate the performance of the detection algorithm or model using a large and diverse dataset of fake images. Overall, the aims and objectives for fake image detection should be aligned with the needs and goals of the intended users and stakeholders and aim to promote trust, accuracy, and transparency in digital media.

2. Overall Description

2.1 Workflow of Project:

The diagram below shows the workflow of this project.

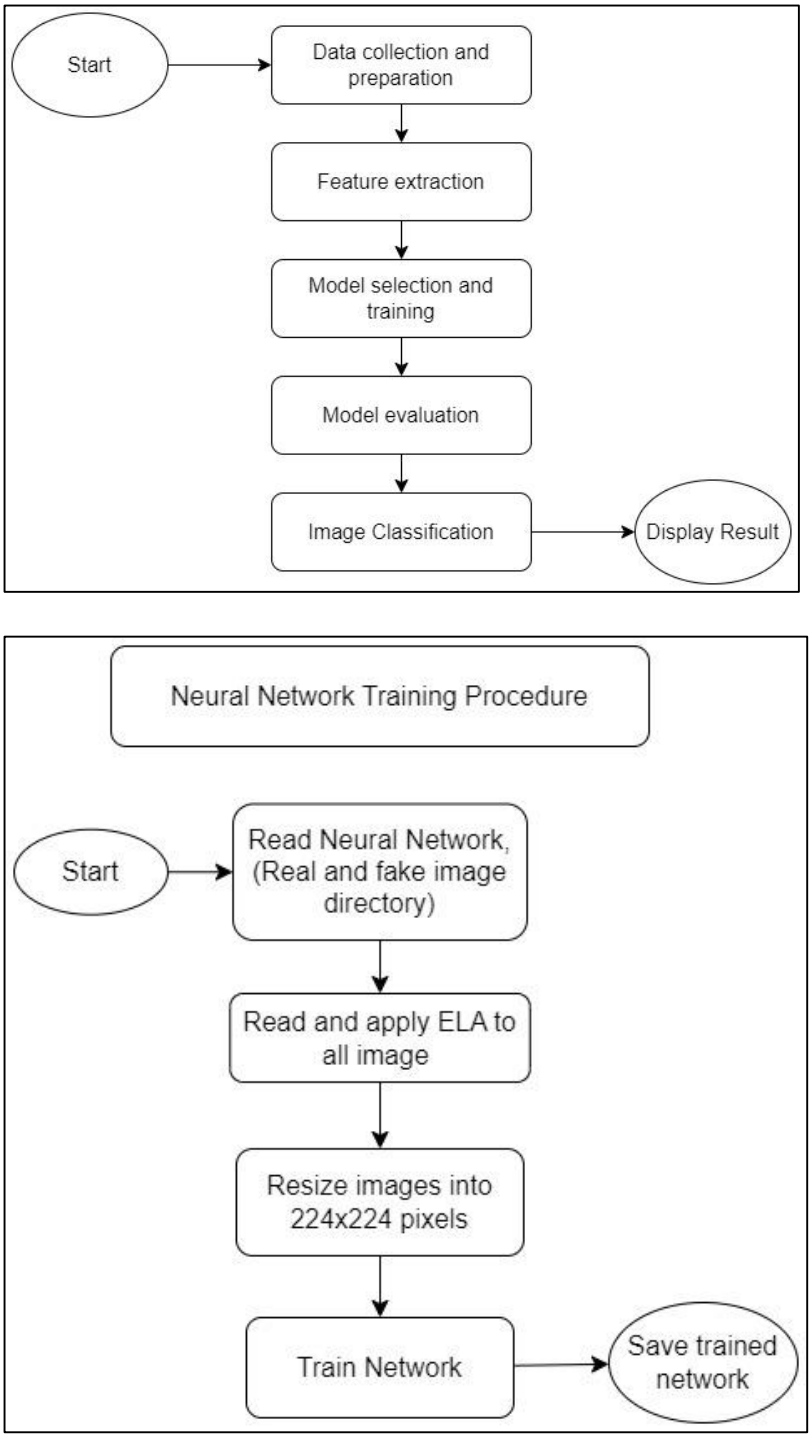

Figure 1Workflow Diagram

2.2 Data Preprocessing :

Preprocessing: The acquired images are preprocessed to remove any irrelevant information, reduce noise, and adjust brightness and contrast levels.
Feature Extraction: The system extracts relevant features from the preprocessed images, such as color histograms, texture features, or frequency domain features.

2.3 Model Training :

The system trains a fake image detection model using machine learning or deep learning algorithms. The model is trained using a large and diverse dataset of genuine and fake images to learn to distinguish between them.

2.4 Model Evaluation:

The system evaluates the performance of the trained model using a test dataset. The evaluation metrics can include accuracy, precision, recall, and F1 score.

2.5 Image Classification:

The system classifies new images into genuine or fake using the trained model. The classification result is based on the features extracted from the image and the learned parameters of the model.

2.6 Exploratory Data Analysis:

Exploratory Data Analysis (EDA) is an essential step in any data analysis project, including fake image detection. The main goal of EDA is to gain a better understanding of the data, identify patterns, trends, and potential problems, and formulate hypotheses that can be tested later on. Here are some steps that you can follow to perform EDA for fake image detection. Data Collection: Collect a dataset of images that includes both fake and real images. You can use existing datasets like DeepFake Detection Challenge (DFDC) or create your own dataset. Dimensionality Reduction: Dimensionality reduction is the process of reducing the number of features in the dataset. This can help to reduce noise and improve the accuracy of the model. Overall, EDA is a critical step in the fake image detection process that helps to identify potential challenges, formulate hypotheses, and build accurate models.

2.7 Model Building:

1. Train/Test split:

One important aspect of all machine learning models is to determine their accuracy. Now, in order to determine their accuracy, one can train the model using the given dataset and then predict the response values for the same dataset using that model and hence, find the accuracy of the model. A better option is to split our data into two parts: first one for training our machine learning model, and second one for testing our model.

- Split the dataset into two pieces: a training set and a testing set.
- Train the model on the training set.
- Test the model on the testing set, and evaluate how well our model did.

Advantages of train/test split:

- Model can be trained and tested on different data than the one used for training.
- Response values are known for the test dataset, hence predictions can be evaluated
- Testing accuracy is a better estimate than training accuracy of out-of-sample performance.

2.8. Convolutional Neural Network :

CNN or Convolutional Neural Network, is a type of artificial neural network commonly used in machine learning and computer vision tasks. It is particularly effective in processing and analyzing images and other forms of structured data.

The main feature of a CNN is its ability to automatically learn and extract features from raw data. This is achieved through the use of convolutional layers, which apply a set of filters to the input data to extract features at different spatial scales.

In addition to convolutional layers, a typical CNN also includes other types of layers, such as pooling layers, which down sample the feature maps to reduce their dimensionality, and fully connected layers, which perform the final classification or regression task.

CNNs have been used successfully in a wide range of applications, including image classification, object detection, image segmentation, and speech recognition. They have achieved state-of-the-art performance in many of these tasks, and continue to be an active area of research in machine learning.

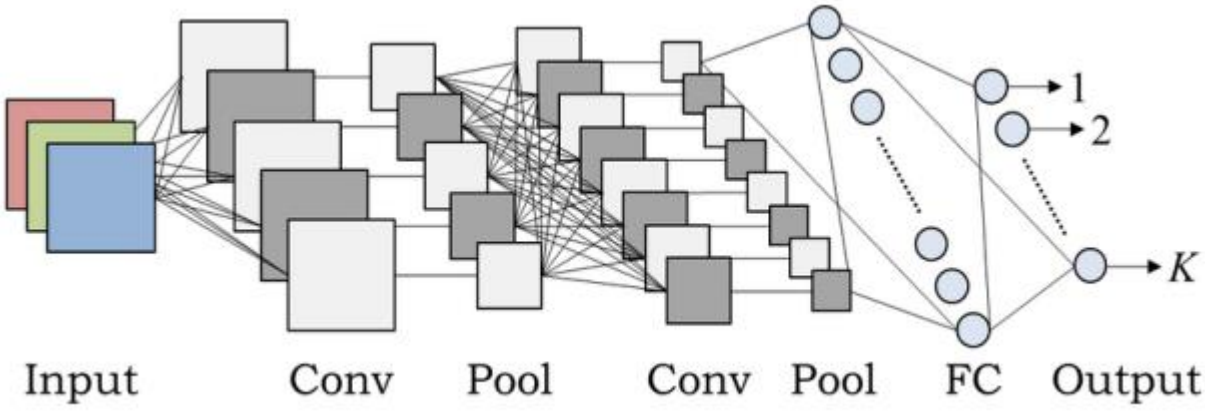

Figure 2 Convolution Neural Network Architecture.

This is Model Summary of CNN :

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 224, 224, 64)      1792

 conv2d_1 (Conv2D)           (None, 222, 222, 48)      27696

 max_pooling2d (MaxPooling2D  (None, 111, 111, 48)     0
 )

 conv2d_2 (Conv2D)           (None, 109, 109, 24)      10392

 conv2d_3 (Conv2D)           (None, 107, 107, 48)      10416

 max_pooling2d_1 (MaxPooling  (None, 53, 53, 48)       0
 2D)

 dropout (Dropout)           (None, 53, 53, 48)        0

 flatten (Flatten)           (None, 134832)            0

 dense (Dense)               (None, 128)               17258624

 dense_1 (Dense)             (None, 1)                 129

=================================================================
Total params: 17,309,049
Trainable params: 17,309,049
Non-trainable params: 0
_________________________________________________________________
```

This is loss diagram of CNN :

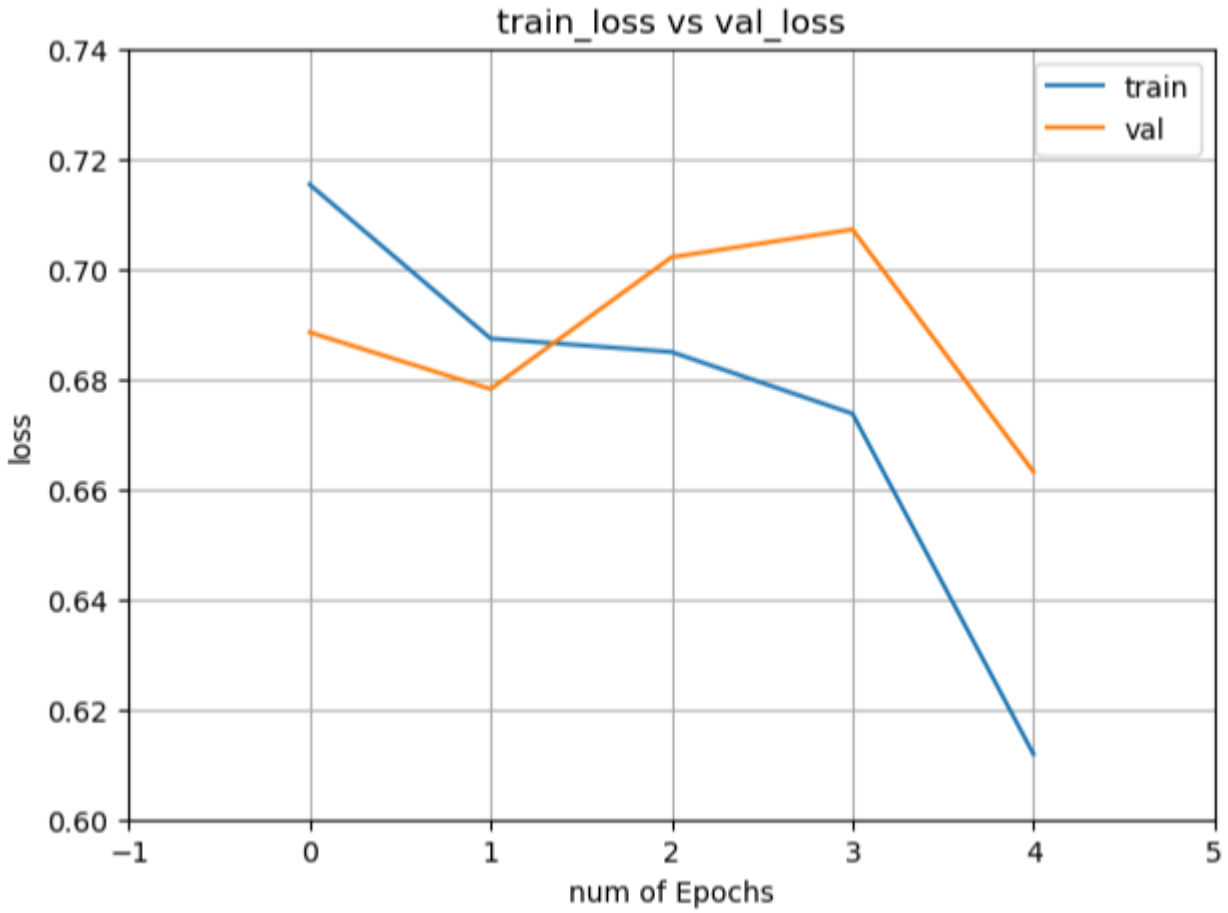

Figure 3 train_loss vs val_loss

In first epoch training loss is 71 % and validation loss is 69 % , after last epoch training loss is 61 % and validation loss is 66 %

This is accuracy diagram of CNN :

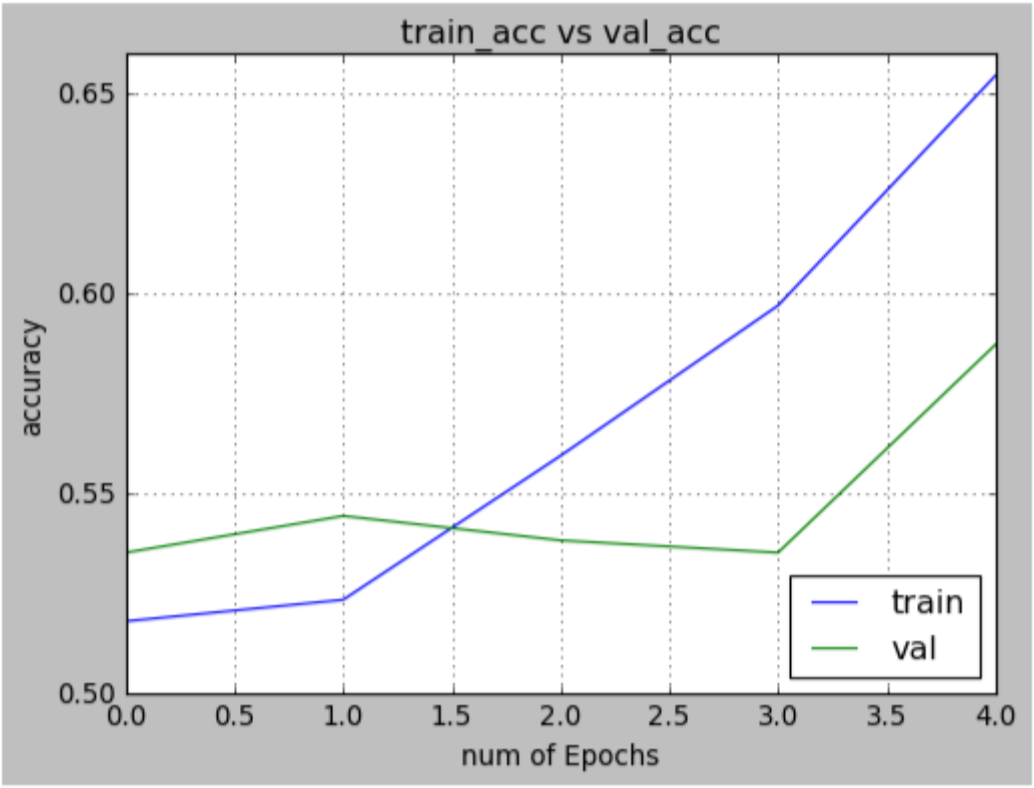

Figure 4 train_acc vs val_acc

In first epoch training accuracy is 51 % and validation accuracy is 53 % , after last epoch training accuracy is 65 % and validation accuracy is 60 %

2.9. VGG16 :

VGG16 is a convolutional neural network architecture that was developed by the Visual Geometry Group (VGG) at the University of Oxford in 2014. It is a deep neural network consisting of 16 layers, including 13 convolutional layers and 3 fully connected layers.

The VGG16 architecture is characterized by its simplicity and uniformity, with all convolutional layers having a 3x3 filter size and a stride of 1 pixel, and all pooling layers having a 2x2 filter size and a stride of 2 pixels. This uniformity makes it easy to understand and reproduce the network architecture.

The VGG16 architecture consists of 16 layers, which can be divided into two main parts: the convolutional base and the fully connected layers. Here is a summary of each layer in the VGG16 architecture:

- Input layer: This layer accepts the input image of size 224x224x3.
- Convolutional layers: There are 13 convolutional layers in the VGG16 architecture, each with a 3x3 filter size and a stride of 1 pixel. The number of filters increases from 64 to 512 in the deeper layers.
- Max pooling layers: There are 5 max pooling layers in the VGG16 architecture, each with a 2x2 filter size and a stride of 2 pixels.
- Fully connected layers: There are 3 fully connected layers in the VGG16 architecture, each with 4096 neurons. The final layer has 1000 neurons, one for each class in the ImageNet dataset.
- Softmax layer: This layer applies the softmax activation function to the output of the final fully connected layer, producing a probability distribution over the classes.

Overall, the VGG16 architecture has over 138 million parameters, making it one of the largest and most computationally expensive deep neural networks.

```
Model: "vgg16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3)]     0

 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792

 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928

 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0

 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856

 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584

 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0

 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168

 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080

 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080

 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0

 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160

 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0

 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0

 flatten (Flatten)           (None, 25088)             0

 fc1 (Dense)                 (None, 4096)              102764544

 fc2 (Dense)                 (None, 4096)              16781312

 predictions (Dense)         (None, 1000)              4097000

=================================================================
Total params: 138,357,544
Trainable params: 138,357,544
Non-trainable params: 0
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792

 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928

 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0

 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856

 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584

 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0

 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168

 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080

 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080

 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0
 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160

 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0

 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0

 flatten (Flatten)           (None, 25088)             0

 fc1 (Dense)                 (None, 4096)              102764544

 fc2 (Dense)                 (None, 4096)              16781312

 dense_2 (Dense)             (None, 1)                 4097

=================================================================
Total params: 134,264,641
Trainable params: 4,097
Non-trainable params: 134,260,544
```

This is loss diagram of VGG16 :

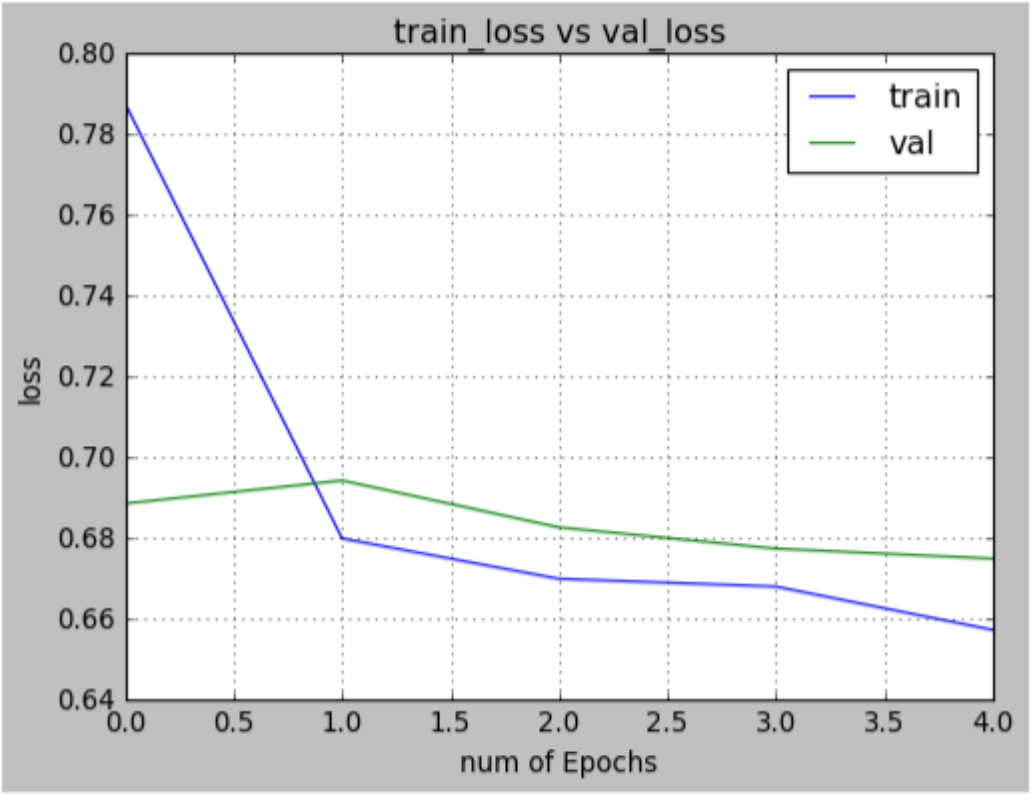

Figure 5 VGG16 train_loss vs val_loss

In first epoch training loss is 79 % and validation loss is 69 % , after last epoch training loss is 65 % and validation loss is 65 %

This is accuracy diagram of VGG16 :

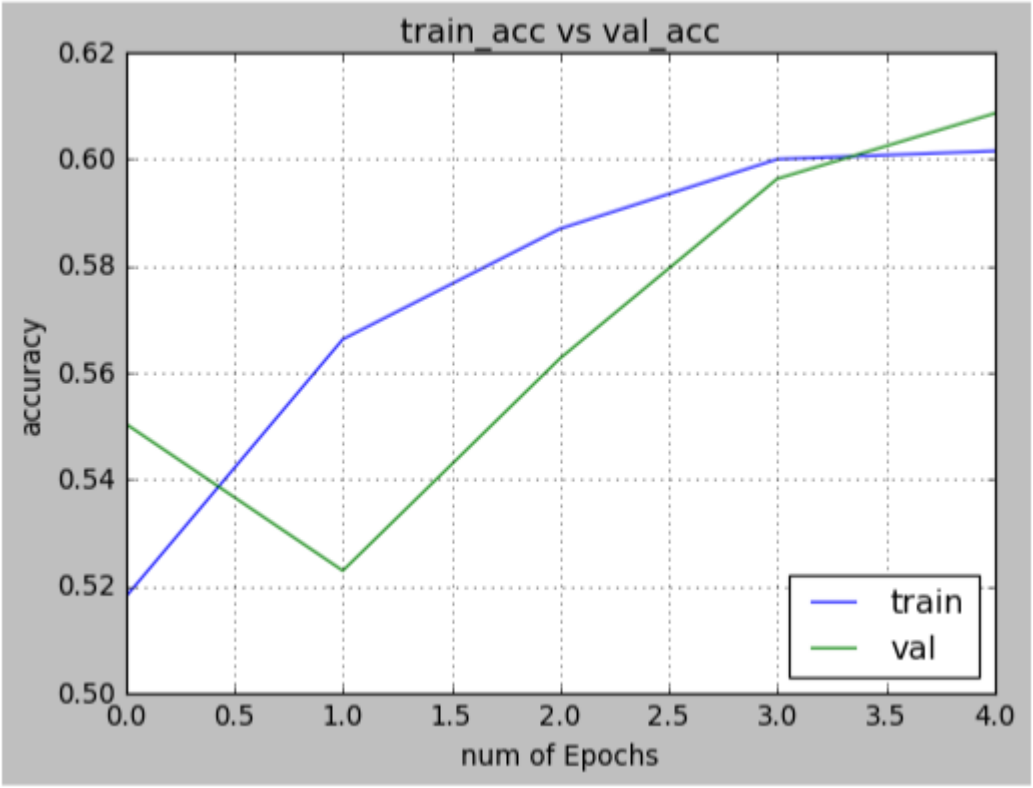

Figure 6 VGG16 train_acc vs val_acc

In first epoch training accuracy is 51 % and validation accuracy is 53 % , after last epoch training accuracy is 65 % and validation accuracy is 60 %

This is our prediction for real image :

```
## Change the value of n for other images. I have chosen these images randomly.

n = 15

prediction = model.predict(prepare(X_test[n]))


x = ["Real-Face" if y_test[n]== 0 else "Fake-Face"]
print("Actual: ",x[0])
rounded_pred = model.predict(x = prepare(X_test[n]), batch_size=10, verbose=0)
y = ["Real-Face" if rounded_prediction[0]== 0 else "Fake-Face"]
print("Prediction: ", y[0])
plt.imshow(load_img(X_test[n]), cmap='gray')
plt.show()
```

```
1/1 [==============================] - 2s 2s/step
Actual:  Real-Face
Prediction:  Real-Face
```

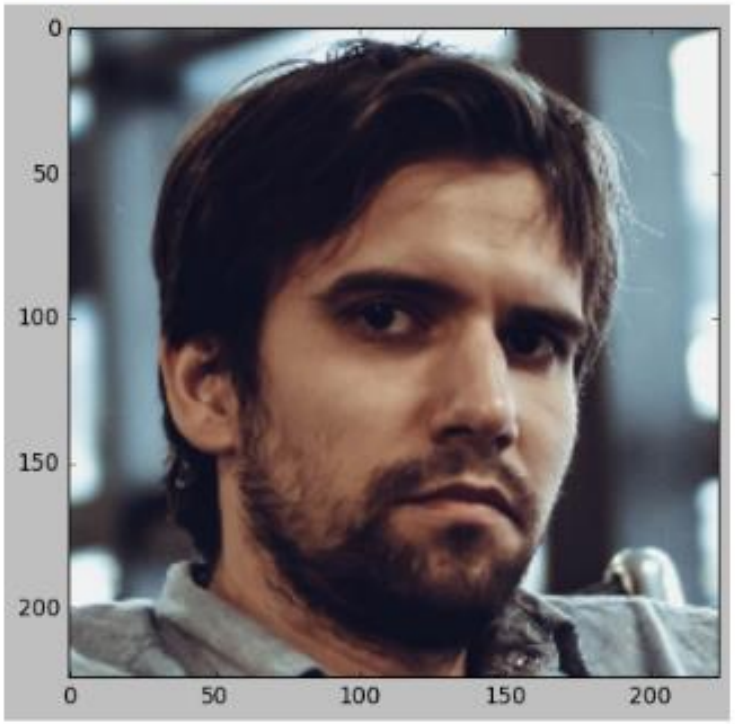

Figure 7 Prediction 1

This is our prediction for fake image :

```
n = 250

prediction = model.predict(prepare(X_test[n]))

x = ["Real-Face" if y_test[n]== 0 else "Fake-Face"]
print("Actual: ",x[0])
rounded_pred = model.predict(x = prepare(X_test[n]), batch_size=10, verbose=0)
y = ["Real-Face" if rounded_pred[0]== 0 else "Fake-Face"]
print("Prediction: ", y[0])
plt.imshow(load_img(X_test[n]), cmap='gray')
plt.show()
```

```
1/1 [==============================] - 2s 2s/step
Actual:  Fake-Face
Prediction:  Fake-Face
```

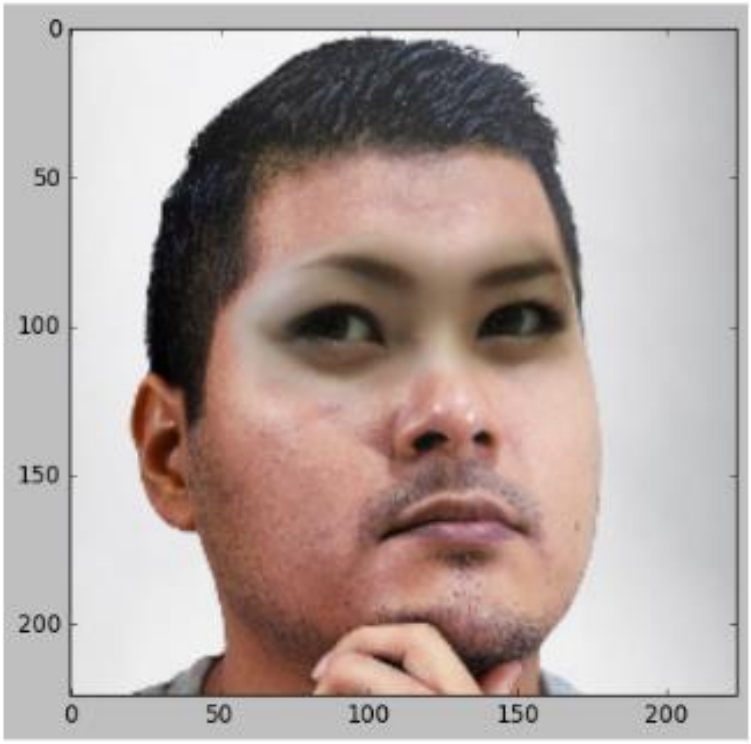

Figure 8 Prediction 2

3. Future Scope

The future scope of fake image detection is likely to be significant, as the problem of image manipulation is becoming increasingly widespread and sophisticated. Some potential areas of development in this field include:

1. Improving the accuracy of existing methods: As machine learning algorithms and other techniques for fake image detection continue to evolve, there is likely to be ongoing research into ways to improve the accuracy of these methods.
2. Developing new detection methods: Researchers are likely to explore new ways of detecting fake images, such as using blockchain technology to verify the authenticity of images, or developing new algorithms that can detect manipulated videos.
3. Incorporating context: To make fake image detection more effective, researchers may start incorporating contextual information, such as social media profiles, text captions, and other metadata, into their algorithms.
4. Automating the detection process: As the volume of digital images continues to grow, there may be a need for automated fake image detection systems that can quickly and accurately identify manipulated images in large datasets.
5. Collaboration across industries: There may be a need for collaboration between technology companies, government agencies, and other stakeholders to develop shared standards and best practices for detecting fake images

4. Conclusion

In conclusion, the detection of fake images is becoming increasingly important as the manipulation of digital images becomes more widespread and sophisticated. With the rapid advancement of machine learning and computer vision technologies, there is great potential for developing effective and accurate algorithms for detecting fake images.

The problem of fake image detection is complex and challenging, as there are many different types of image manipulation techniques, and new methods are constantly being developed. However, by developing accurate and reliable methods for detecting fake images, we can help ensure that people have access to accurate information and can make informed decisions based on trustworthy sources.

Effective fake image detection can also help to prevent the spread of misinformation and disinformation, which can have far-reaching consequences for individuals, organizations, and societies as a whole. As such, the continued development of methods and technologies for detecting fake images is likely to be a critical area of research in the years to come.