Detecting Deepfake Images using Deep Convolutional Neural Network

by

Arpita Dhar
17101069
Prima Acharjee
18301293
Likhan Biswas
17101405
Shemonti Ahmed
21341062
Abida Sultana
16201085

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science and Engineering

Department of Computer Science and Engineering
Brac University
September 2021

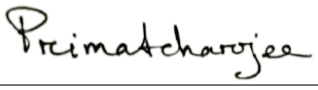
Declaration

It is hereby declared that

1. The thesis I/we submitted was prepared while pursuing a degree at Brac University and it is our original research work.
2. Except as properly acknowledged through thorough and precise citation, the paper does not contain any content that is authored or published by others.
3. Our thesis does not contain any kind of work done by anyone else of any other university.
4. All major sources of assistance have been cited properly.

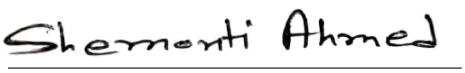
Student's Full Name & Signature:

Arpita Dhar
17101069

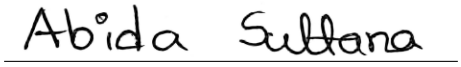


Prima Acharjee
18301293

Likhan Biswas
17101405



Shemonti Ahmed
21341062



Abida Sultana
16201085

Approval

The thesis/project titled "A Deep CNN Model For Prediction of Deepfake Images" submitted by

1. Arpita Dhar (17101069)
2. Prima Acharjee (18301293)
3. Likhan Biswas (17101405)
4. Shemonti Ahmed (21341062)
5. Abida Sultana (16201085)

Of Summer, 2021 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science and Engineering.

Examining Committee:

Supervisor:
(Member)

Zavid Parvez Digitally signed by Zavid Parvez
DN: cn=Zavid Parvez, o=Brac University, ou=CSE, email=zavid.parvez@bracu.ac.bd, c=US
Date: 2021.10.05 11:10:52 +11'00'

Dr. Mohammad Zavid Parvez
Assistant Professor
Department of Computer Science and Engineering
BRAC University

Co-Supervisor:
(Member)

Dewan Ziaul Karim
Lecturer
Department of Computer Science and Engineering
Brac University

Program Coordinator:
(Member)

Dr.Md.Golam Rabiul Alam
Associate Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi
Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Ethics Statement

The thesis paper is written in strict accordance with research ethical guidelines as well as Brac University's regulations and procedures. We have employed data from original sources in our thesis. We made sure we're using citations and references correctly. All five co-authors of this paper, accept liability for the breaches of the thesis code. We used many Questionnaire Free tools, websites, and YouTube videos to solve difficulties. Finally, we announce that we acknowledge and thank everyone who supported us. We did not use any unethical methods to complete our thesis. Our work complies with the Brac University's ethical standards.

Abstract

In recent years, advancement in the realm of machine learning has introduced a feature known as Deepfake pictures, which allows users to substitute a genuine face with a fake one that seems real. As a result, distinguishing between authentic and fraudulent pictures has become difficult. There have been several cases in recent years where Deepfake pictures have been used to defame famous leaders and even regular people. Furthermore, cases have been documented in which Deepfake yet realistic pictures were used to promote political discontent, blackmail, spread fake news, and even carry out false terrorism attacks. The objective of our model is to differentiate between real and Deepfake images so that the above mentioned situations can be avoided. This project represents a deep CNN model with 13000 images divided in two segments: Training and Testing. The dataset was prepared using necessary image augmentation techniques. A total of 2 categories are considered (real image category and fake image category). For testing purpose we have used a total number of 3000 images divided into two parts for real and fake class, each consisting 1500 images. 75% of the whole data was used as Testing data and remaining 25% as Training data. The dataset was tested against a custom CNN model referred to in the paper as the 18-layered CNN model and five of the transfer learning models. Our suggested model was successful in achieving 98.77% accuracy whereas the best result out of the transfer learning model was achieved by InceptionV3 with 97.10% testing accuracy. The custom CNN model shows promising results in the case of detecting real and DeepFake images than all the other models used before.

Keywords: CNN; DeepFake; Deep Learning; Image Processing; Transfer Learning

Dedication

This paper is dedicated to our families, who have been a great support to us. We would also like to dedicate this paper to our dear friends, who have been patient and helpful while completing this research. Our respected supervisor and co-supervisor have been a constant guide and without their suggestions and instructions, we would not be able to finish our thesis. We dedicate our paper to them as well.

Acknowledgement

First and foremost, all thanks and praises to the Almighty for the blessings throughout our research without which the thesis would not be successful.
Secondly, we would thank our parents and well wishers who helped us throughout this project.
After that, we sincerely thank our supervisor Dr. Mohammad Zavid Parvez sir and our co-supervisor Dewan Ziaul Karim sir for their constant help and guidance.
Finally, we want to express our gratitude to all of the faculty members and students for creating such a humble learning atmosphere in which we could both grow personally and do this research successfully.

Table of Contents

List of Figures

List of Tables

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

$*$ Multiplication

$3D$ Three-dimensional

API Application Programming Interface

AUC Area Under Curve

$AUROC$ Area Under the Receiver Operating Characteristics

Avg Average

CNN Convolutional Neural Network

$Colab$ Colaboratory

$CONV$ Convolutional

CT Computed Tomography

$DenseNet$ Densely Connected Convolutional Networks

FC Fully Connected

GAN Generative Adversarial Network

$GIMP$ GNU Image Manipulation Program

GPU Graphics processing unit

$MTCNN$ Multi-Task Cascaded Convolutional Neural Networks

No Number

$Pram$ Parameter

$R-CNN$ Region Based Convolutional Neural Network

$ReLU$ Rectified Linear Activation Function

$ResNet$ Residual Neural Network

RGB Red Green Blue

RNN Recurrent Neural Networks

SGD Stochastic gradient descent

VGG Visual Geometry Group

Chapter 1

Introduction

1.1 Background Information

The expansion of technology has led human life to be easier in many ways, but on the other hand, there has been some misuse of technology as well, which led to some severe problems. Nowadays, the usage of different tools and software have made it easy to manipulate any digital image. Anyone with a little knowledge of Photoshop or GIMP [12] can easily forge an image of anything and anyone. Nevertheless, the usage of this kind of forgeries has been widely looked into in modern times. The advancement in the fields of Artificial Intelligence and, most importantly, Machine Learning has enabled people to edit a raw image and exploit it in both good and bad ways. Moreover, these tools are able to procure astonishing realistic results, which led us to the world of Deepfake images or, to simply put it, Deepfakes [17] . DeepFake is a technology that uses Deep Learning to create fake images in which one person's face is replaced onto the face of someone else. This may also constitute a risk to celebrities, whose faces may be simply morphed for spreading rumors and chaos. The earliest documented effort at an exchange of faces was captured on a picture of Abraham Lincoln [30]. His face was placed on top of John Calhoun's body in the lithography. Following this event, false recordings began to circulate widely, endangering national security, the social standing of prominent celebrities, and the safety of important political figures. In recent years, influential people have been the victim of Deepfake images, where their photos or videos were being manipulated and used to spread the wrong propaganda. No doubt, this can create controversy and chaos in people's lives, which can cause heavy damage too. So to tackle the situation, Deepfake detector systems are being created using various techniques of convolution neural networking, augmentation process, image processing and filtering [61]. Developed systems are even being able to detect the most finely forged images with greater accuracy than before. However, in this paper, we have used a deep CNN model that distinguishes between Real and Deepfake images using binary classification and provides accuracy. Thus, it is able to detect manipulated images created via artificial intelligence with great precision.

1.2 Problem Statement

The CNN approaches are computer-intensive and difficult to describe or manage. Tests were performed using a VGG16 Deep Neural Network to overcome the task of binary classification. In order to produce the best performance, the more complicated the architecture, the more computing power is required [45]. Only a complicated deep neural network structure can produce improved outcomes. The Deepfake algorithm increases the precision of Deepfake picture detection through multiple angles, nevertheless, many of the dataset utilized are poor in consistency, with a rough quality, even with the naked eyes, almost all of the interference marks could be seen. To locate the false image by combining the contrasting loss and competing error to detect fake images in the fake video later, some researchers also proposed a deep learning-based method in 2019. It is possible to loosely break the model of deep learning into discriminating and generative models. The Generative type model has a first-hand influence on efficiency of real world modelling, which needs much previous experience [61]. Generation efficiency has steadily upgraded, and it is impossible for the eyes to trace. It acquires abstract characteristics automatically [17]. Next, The aim of Digital Media Forensics technologies is to determine credibility of a picture or video automatically. This is a technology that doesn't have to work with informations from begining. The introduction in ongoing advancement of Deepfake technologies enable users to forge the images with a single key, unlike conventional editing software. The fake picture quality is becoming greater and greater with continuous growth. The naked eye can see much of the tampering marks, which results in its poor identification and unstable generalization capabilities. Image efficiency is declining rapidly [46]. Later on, A series of attacks were made at different Deepfake images of faces generated by various algorithms. The attacks were carried out to check the robustness of the algorithms. Attacks included adding Gaussian Blur [62], scaling images and extracting great detail from images. The results showed that some operations could lead to errors in the analysis of the Deepfake architecture, but none of the attacks had a major impact on the discriminative power of the CT. The algorithm was designed to be robust to a range of different types of data compression and decompression operations.

1.3 Research Motivation

The Deepfake is a new concept that has become exceedingly popular in the recent years for the creation of super realistic images using different tools like FaceApp [52], Photoshop etc. Fake news or fake information always spreads fast over social sites which sometimes becomes a threat to an individual, society, political system and in fact a nation. However, deepfake imposes a great threat. Even more than regular false news because they are tough to identify and people are likely to trust the fake as real. In order to prevent such falsification, we have designed a system that detects deepfake images using machine learning, convolutional neural networks networks which will allow us to differentiate among real image and fake image which otherwise would be difficult.

1.4 Research Objectives

Our project uses Convolutional Neural Network (CNN) along with Transfer learning method to detect deep fake images. CNN is among the main parts of the Neural Network, and it uses image detection and classification to detect objects and recognize the faces of a real person. Moreover, we will also be able to determine whether the images have been subjected to tamper or not. There are many people who worked on this project before where the fake image processing technique using CNN is incorporated into work. So it is easier to detect if the images from the data set are fake or real. In recent times, many types of crime are occurring by making fake images. So if we know how the fake images are generated and also can detect and classify fake images, crimes can be apprehended, and the number of spreading fake news, blackmails, terrorism can be decreased. So as a whole, by using this simple architecture, we can basically accomplish the correct result of detecting fake images from a data set. The auto-generated artificial pictures could be efficiently evaluated using Convolutional neural networks (CNN), as the forged images contain visible visual features. In a variety of computer vision and machine learning challenges, CNN has demonstrated its superior performance. CNN has a number of uses, one of which is image categorization. Images are classified as objects that take up the majority of the picture, and categorization is done by determining the class this object belongs to. One part of the Deep Learning technique is the Convolutional Neural Network (CNN) that is generally used for image recognition and analysis and designed specifically to deal with data that are pixelated. Image is taken as the input and allocates priority (learnable biases and weights) to distinct attributes of the picture, and distinguishes one from another. In comparison to other classification algorithms, CNN requires significantly less pre-processing. While filters are crafted manually in other deep learning approaches, CNN can learn these characteristics or filters with adequate training. CNN's layers have neurons that are organized in three dimensions, which are : width, height, and depth unlike other conventional Neural Networks, where layers build up the structure.CONV/RELU/POOL/FC etc are most common among the widely used layers. Each of the layers has a straightforward API which uses a discrete function which might or might not contain parameters to convert a 3D input into an output that is also 3D. As mentioned before, a basic CNN is a succession of layers where, by a differentiable function, every layer of a CNN translates a single set of operations to the other. The Pooling Layer, Con-

volutional Layer, and Fully-Connected Layer are the three kinds of layers that can be utilized to create a CNN structure. Stacking of these layers is used to build a complete CNN structure. The raw pixel values the image's raw pixel values will be stored by INPUT (e.g. INPUT [224x224x3] stands for an image with width of 224, height of 224, consisting of three color channels respectively: R,G,B). The outcomes from the neurons connected to particular areas of input will be calculated by the CONV layer, which will calculate a dot product among a limited part of the input size that they are connected with and their weights and. If we choose to utilize 12 filters, this might lead towards a size of [224x224x12]. Moreover, an element based activation function is assigned by the RELU layer. Downsampling along the spatial dimensions operation is performed by the pooling layer. Finally the class score is computed by the FC or fully connected layer. Thus by layering the primary pixel value to the score of the final class, the transformation of an original image is done by a CNN model. As base-learners, InceptionV3, ResNet50, DenseNet121, VGG16, and VGG19 are implemented and a newly constructed CNN model as we mentioned just now, also known as the 18-layer custom CNN model, is employed. Utilizing these models, the above mentioned custom model gained an accuracy of 98.77%.

Chapter 2

Related Work

2.1 Literature Review

Our study of the literature examines how far we've come and what more can be done to identify deepfake utilizing deep learning techniques, particularly Convolutional Neural Networks (CNN). We just look at Deepfakes, but we've also explored at detecting a further actual image accuracy rate. Deepfakes have been detected using aberrations or inefficiencies inherent in the formulation algorithms, such as the lack of genuine eye gaze, uneven color profile information, and visible lips with talks. In order to distinguish actual images from Deep Fakes, a neural network-based transfer learning classification technique is chosen.

2.1.1 Deepfake Generation

This segment attempts to analyze fundamentally previous important studies in the field of identification of Deepfake material in digital technology. We examine the various methods used with the desired outcome, such as pair learning, face warping artefacts etc., to demonstrate how multiple technologies face their unique problems as a result of the heterogeneity of the method. Digital media forensic technology is intended to automatically test the picture or video credibility and judge if they have Digital tampering, synthesizing [56]. In this technology, we do not have to deal with information from the media in advance. Its universality and practicality make it a major area of study. Traditional digital media passive forensics technology primarily involves Passive Forensics based on traces left after imitation. It is based on the consistency of digital media imaging devices, and based on digital media statistical characteristics [26]. Unlike conventional image editing software such as PS, Deepfake technology's advent and continual growth, no technology users may utilize web-apps to modify the image with one key. It is simple to use and also difficult to determine whether the false image is genuine. Deepfakes are typically generated by methods based on the first introduction of Generative Adversarial Networks (GANs) [58] Via Goodfellow et al.[17] a new method for counting Generative models through an Adversarial model was proposed by the authors. A mode in which two models train at the same time: a Generative model, which captures the distribution of data, and A discriminatory model, capable of estimating the possibility of a sample originating from training data rather than from training data of generative model [17]. This one's problem is, the new method needs to examine more for its betterment. In 2018,

David Güera et al. suggest a temporal aware pipeline to sponrteniously identify deepfaked images; a Convolutional Neural Network (CNN) is utilized to retrieve frame level characteristics that are then utilized to train a Recurrent Neural Network (RNN) [22]. However, RNNs have the issue of disappearing gradients that hamper lengthy data acquisitions [30]. Peng Zhou et al. proposes a two stream faster R-CNN and trains it end to end to identify a distorted picture of the tampered regions [26]. Chih-Chung Hsu et al. suggested a deep learning-based method in 2019 to identify the fake image by integrating the contrastive loss [46]. Only limited resolution images can be created based on the current DeepFake algorithm, and distortion is required to fit the real face into the actual video. It is proven that the objects left in the distortion matching phase can be efficiently captured by the convolutional neural network (CNN). The following algorithm increases the accuracy rate of fake image detection. Through the continuous DeepFake technology growth, the standard of picture forgery is getting higher, which results in greater efficiency of the altered detection circulated in the network of methods. The trouble with Fast R-CNN is that it's slow because it has to do selective searches that are computationally very slow. The debate above indicates that most study designs use a common form of technology to detect deep video or images. The detection mechanisms, however, must gradually be studied to better our lives because most detection systems are bound to have such heterogeneous characteristics [44].

2.1.2 CNN

Convolutional Neural Network is a widely used deep learning based model to distinguish between Deepfakes and real ones. Md. Shohel Rana et al. [54] in his paper proposed DeepfakeStack, a deep ensemble learning approach that has been developed by comparing various Deep Learning-based state-of-the-art models. This model incorporates the predictions of these level-0 models. By using out-of-sample data predictions provided by base models, the level-1 model is trained. As base-learners, InceptionResNetV2, InceptionV3, ResNet101, DenseNet121, MobileNet, and DenseNet169 are implemented and a newly constructed CNN model, also known as Deepfake Classifier, is employed. Utilizing these models, the DeepfakeStack gained an accuracy of 99.65% and an AUROC of 1.0 in the experiment. Another system proposed by Xu Chang et al. [62] is based on image augmentation and image noise to detect DeepFake images with the help of the VGG network. To highlight the image noise [11] the RGB images that need to be detected are used through SRM filter technique to get the noise characteristics of the image. Because of the depth of forgery upon a large dataset over the network and poor image quality, their experiment uses Celeb-DF [50] as their dataset which shows significant results in detecting DeepFake face images. Where on one hand, this experiment achieved an AUC performance of 73.2% while using SRM filter [14] and VGG-16 , on the other hand their NA-VGG approach on the same dataset was able to achieve an AUC performance of 85.7% [44]. Anuj Badale et al. [50] suggested a method utilizing Neural Networks, as described in this study based on a Deepfake video detection approach. Original and deepfake frames were extracted using videos. Depending on the video from which it was taken, each frame has been assigned a class, such as "real" or "fake." The inputs are flattened to provide them the dimension (256x256x3), which is used as an input to the neural network's 1st layer. As a result, the data is prepared

for entry. For categorical cross entropy, it was determined that Adam achieved 91% accuracy while sgd (stochastic gradient descent) achieved 88%. In binary cross entropy, Adam achieved 90% accuracy and sgd achieved 86% accuracy, whereas mean square achieved 80% accuracy in sgd and 86% accuracy in Adam [50]. Furthermore, Shraddha Suratkar et al. [55] uses a CNN architecture and the Transfer Learning methodology, which provides a way for exposing fraudulent videos. The suggested methodology extracts features from each frame of a video and employs CNN. The models used in the paper and their testing accuracy percentage are: InceptionV3: 91.56, ResNet50: 84.04, ResNet18: 84.68, NasNetMobile:89.17, MobileNet: 88.67, Xception: 90.08. The model based on Inceptionv3 achieved the highest testing accuracy among all the other models of the aforementioned paper.

2.1.3 Transfer Learning Approach

Due to the accessibility of free popular open sensor technology for deep fake creation, there has been a significant increase in the dissemination of fake news due to Deepfake appearances. There are some previous approaches that showed early different methods yet having few Deepfake physical flaws. For instance, Yang et al. [41] proposed a novel approach for exposing AI-generated false face pictures and experiments are conducted to show this occurrence, and a categorization technique based on this cue is further developed. Here, a Deepfake is created by cropping a picture to (64 x 64) pixels and passing it into a deep generative neural network. A face sensor is utilized to generate a classification model for this face, and then facial landmarks are identified. Post-processing techniques such as boundary smoothing are used to create an image/video frame. One of the steps Suratkar et al. [55] proposed the use of a CNN architecture and the Transfer Learning methodology, this article presents a method for exposing fraudulent videos. The developed technique employs a CNN to retrieve features from each frame of a video in order to train a binary classifier that can effectively distinguish between authentic and altered clips. Yuezun et al. [32] investigated the algorithm created using DNN-based software [33] and tested using eye-blinking detection datasets as a reference level. LSTMs [6] are memory units that regulate when and how prior obscured states are erased. They adjusted the settings of the CNN layers that they had gotten and trained on LSTM cells and the fully connected layer by setting the size of the batch by 4. The learning rate started at 0.01 and decomposed by 0.9 each 2 epochs, also used the ADAM optimizer [45] and finished training until 100th epoch. Rosli et al. [62] used different Convolutional Neural Network models, VGG19 and ResNet50, for image counterfeit detection in deep learning with transfer learning. They employed these two to determine false pictures from image splicing as a result of this research, where 1500 images including 450 fabricated pictures were used as dataset and the result achieved 94.65% and 95.08% of accuracy individually. Zhang et al. [42] employed Mobilenet V1 and Inception V3 models were retrained using a variety of data collection techniques that included an extracted face feature from DeepFake made video and actual footage. 1 frame per second was used to extract each frame from chosen clips into a PNG file. After that, the retrieved frame was sent into the facial analysis configuration, which recovered the face. DLIB facial encoder [57] often collects non-facial areas in certain frames. As a result, datasets were reduced. To show consistency, the dataset was separated into an 80 percent training set and

a 10% validation and testing set, with the training steps of 5000 the learning rate of 0.005. Kim et al. [60] applied the Representation Learning [5] and Knowledge Distillation frameworks [29] to develop the Feature Representation Transfer Adaptation Learning (FReTAL) technique, which is based on supervised learning. They implemented FReTAL to execute classification challenges on fresh deepfake datasets with the least amount of calamitous loss possible. By extracting information from a transfer learning based model without requiring target domain data during area adaption, their model was able to detect new forms of deepfake. They show that FReTAL beats all benchmarks on the adaptation of the domain test, with up to 86.97 percent accuracy on minimal Deepfake, using FaceForensics++ datasets [39]. Patel et al. [53] modeled the variants that were put to check on the same data, and were used for training and testing. The training set accommodates about 7000 pictures, whereas the test set contained approximately 2000 pictures for classification. From the experiment, mobileNet has the greatest accuracy of all the models, at 90.2%, followed by DenseNet121, which has an accuracy of 89.7%. InceptionV3 has the lowest recall rate of 88.1%. MobileNet showed the most impressive overall performance. DenseNet121, on the other hand, outperforms MobileNet in recall, which is far more essential, so DenseNet121 was selected. But, till now, we did not get a proper model that has given a higher accuracy with proper recall performance.

Chapter 3

Workflow

For the detection of whether an image is real or fake, we have used a custom 18-layer Deep Convolutional Neural Network Model and Transfer Learning approach to compare and get the best result. The models used for the aforementioned experiments are respectively: InceptionV3, VGG-19,VGG-16, DenseNet121, Resnet50. Our project consists of a total number of two classes defined respectively as Real and Fake. We have tested our proposed model depending on the Training Accuracy [2], Testing Accuracy [1], Recall [4], Precision [4], and AUC Performance [34]. The outline of our project is divided into seven steps. However, the Workflow Graph is given below in Figure 3.1.

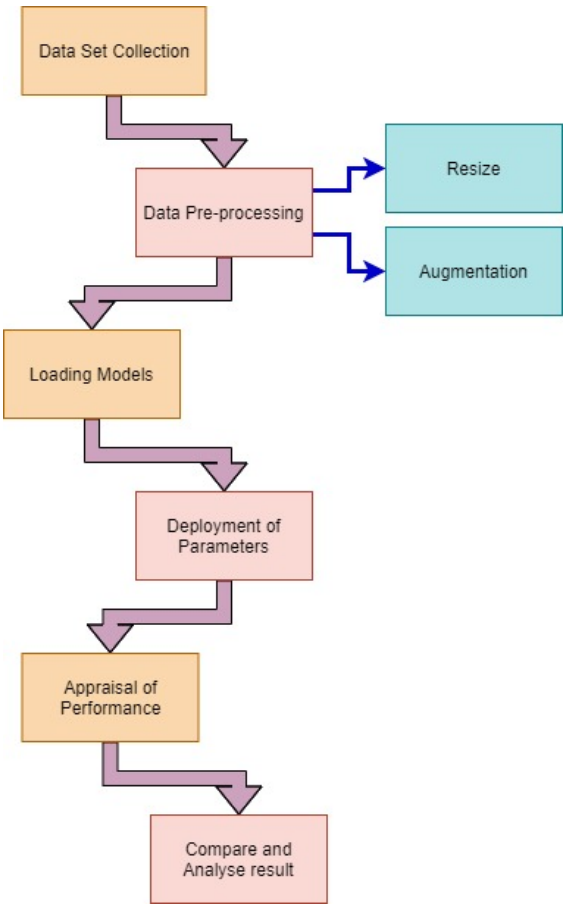


Figure 3.1: Block Diagram of the Proposed Method

Chapter 4

Proposed Methodology

4.1 Dataset Collection

We have created a dataset with Real and DeepFake images. Here, collection of the real images has been done from the dataset from kaggle "140k Real and Fake Faces" [43] and the fake images from "DeepFakes" [48] which is also retrieved from kaggle, where the Deep fake images were generated through MTCNN [37]. A total number of 13000 images have been used to create our dataset. These images are separated into two classes, named respectively Real and Fake where 3000 images are gathered for testing purposes and 10000 pictures for training. Every image was in RGB format [9] with the resolution of 92 pixels for fake images and 256 pixels for real images that were processed afterwards. Few of the images from our dataset are given in Figure 4.1 and Figure 4.2 displays the dataset division.

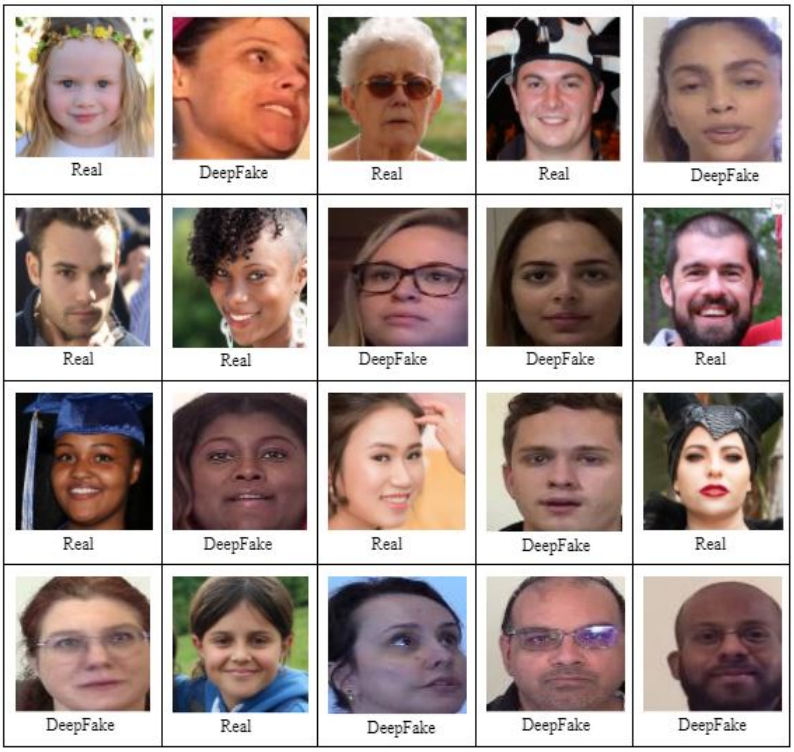


Figure 4.1: Deepfake and Real Images From the Used Dataset

Figure 4.2 shows the percentage of the training and testing data from the dataset.

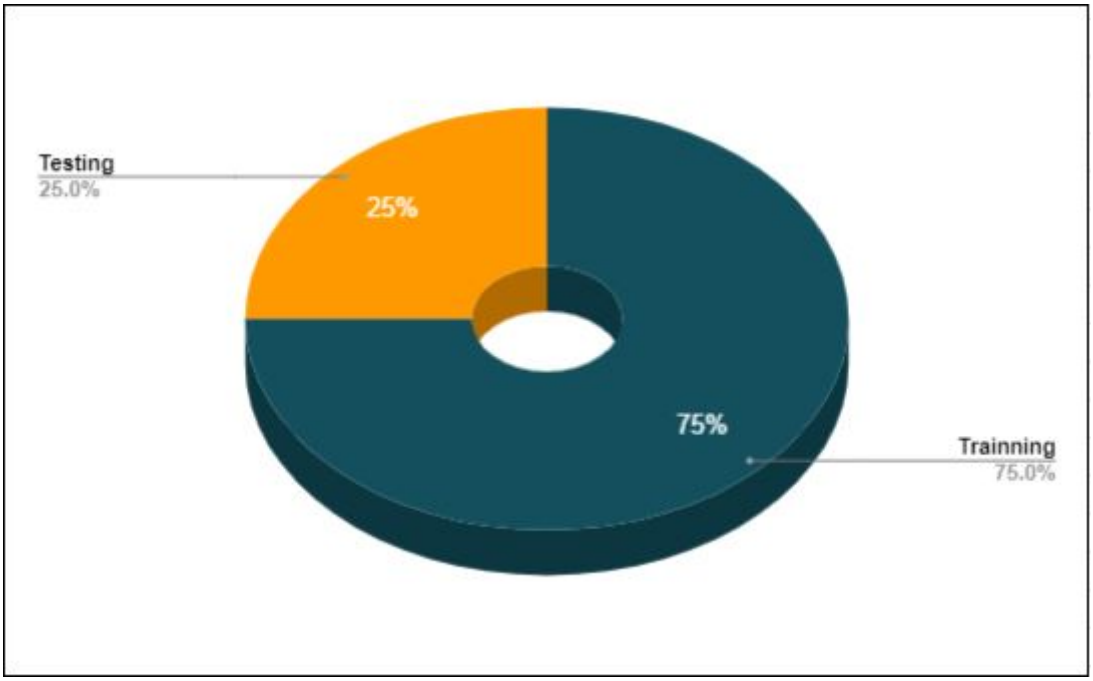


Figure 4.2: Training and Testing Data Percentage

Apart from the training and testing data, we have used 300 unseen images separated into two classes called Real and Fake for generating classification reports and confusion matrices.

4.2 Data Pre-processing

Since information retrieval during the training phase is more challenging, data pre-processing aims to minimise inconsistent data as well as distorted and inadequate input. The final training set is the outcome of data pre-processing. Filtering, normalization, modification, extraction of features [25], identification etc. are some of the techniques that are used generally [28]. Here for our work purpose we have chosen: i. Resizing and ii. Augmentation of the raw data inputs. An unprocessed dataset usage might often result in disappointing outcomes. In this model we have processed our raw data in mainly two ways, which are : Resizing the and Augmentation of the raw images. Detailed descriptions are given below :

❄ **Resizing the Data:** As our system necessitates a consistent dimension of the data, the pictures were scaled down to a standard resolution of 224×224 pixels for the transfer learning approach. Eventually this lessens the shrinking amount. Because CNN models can take a fixed dimension as input, we have resized each RGB image into a stable resolution of 256×256 pixels for the 18-layer CNN model.

❄ **Augmenting Images:** By using augmentation to our data, we have helped in maximizing the amount of training data. To enhance our pictures, we have generated a larger dataset that will help the model generalize to the circumstances it will see in

production. In various situations, different augmentation strategies are more or less beneficial. These mainly affect the training set and should not be included during the evaluation process. We have used different image augmenting methods for our data. The techniques that we have used in the said model are: Rescale, shear, zoom, rotation, width shift, height shift, brightness, vertical flip and horizontal flip for both the approaches.

4.3 Architecture of Proposed Models

4.3.1 18-Layered CNN Model

In the suggested method, we have created a multi layered deep CNN model in order to categorize between real and deep fake images. A basic CNN model has three fundamental layers, which are: Fully connected layer (FC) [16], Convolutional layer [23] and Pooling layer [28]. Apart from these three layers we have used two additional layers which are : the Activation layer and the Dropout layer. The details of each layer are discussed below :

1. **Convolutional Layer:** The very first layer uses convolution to extract information from the input image. Through learning visual attributes with tiny portions of input data, the convolution keeps the connection among the pixels[47] . It is a mathematical system having 2 inputs: a filter or kernel and an image matrix. Convolution of an image with several filters may be used to accomplish tasks such as edge detection, blurring, and sharpening. For this CNN model we have used the Conv2D layer in the convolutional layer. In total we have used six Conv 2D layers to create the model.

2. **Pooling Layer:** Some of the images from the used datasets might be too extensive, the portion of pooling layers will degrade the count of parameters of such images. Subsampling or downsampling also known as, Spatial pooling, will decrease the dimensionality of each map while preserving crucial data. For each of the convolutional layers we have used one MaxPooling2D layer. This means it adds six layers in total with the other layers.

3. **Fully Connected Layer (FC):** The Fully Connected layer or FC layer, one can flatten the matrix into a vector and send it to a Fully Connected layer, similar to a Neural Network. This layer accumulates the neurons, weights and biases. The aforementioned model contains the following layers :

- Flatten layer : One flatten layer is used after the sixth MaxPooling layer. This overall helps to flatter the entire network.
- Dense layer : A total number of three dense layers have been used in this model after the flatten layer. This layer basically feeds all the neurons with the outputs from previous layers.

- Batch Normalization layer : After every Conv2D layer a batch normalization layer is added. Apart from that two of the layers are added after the first two dense layers.
- Dropout layer : For every iteration during the training phase, this layer converts the inputs to zero with a frequency of rate, which helps to avoid overfitting.
- Activation layer : In this model, Sigmoid function is used as the activation layer. The following equation shows the activation Sigmoid function that has been used in the Dense layer

$$f(x) = 1/1 + e^{-x} \tag{4.1}$$

Here, x is a constant that has a real number as its value. The range of this function is 0 to 1. Hence, it is a great fit for the model as it is a binary categorization. The architecture given in Figure 4.3 shows the 18-layer proposed CNN model.

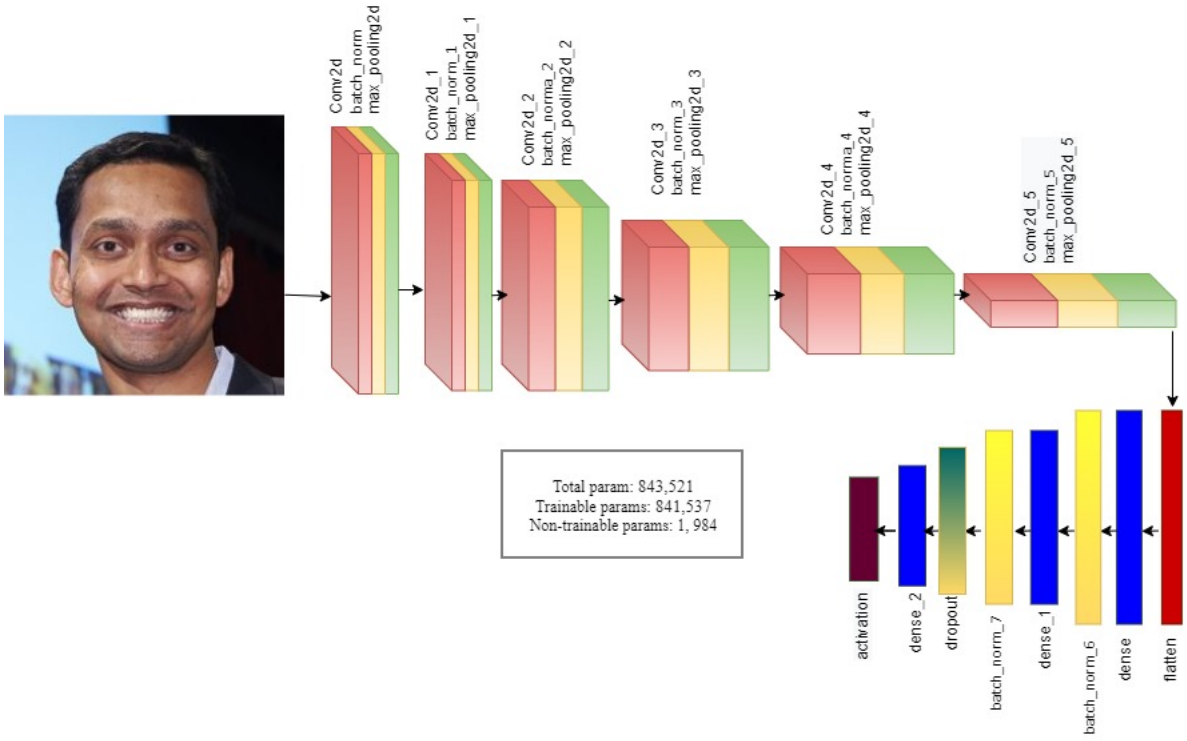


Figure 4.3: Architecture of the Proposed CNN Model

The summary table of the layer used for our suggested CNN model is given below in Figure 4.4 :

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 254, 254, 32)      896
batch_normalization (BatchNo (None, 254, 254, 32)      128
max_pooling2d (MaxPooling2D) (None, 127, 127, 32)      0
conv2d_1 (Conv2D)            (None, 125, 125, 64)      18496
batch_normalization_1 (Batch (None, 125, 125, 64)      256
max_pooling2d_1 (MaxPooling2 (None, 62, 62, 64)        0
conv2d_2 (Conv2D)            (None, 60, 60, 128)       73856
batch_normalization_2 (Batch (None, 60, 60, 128)       512
max_pooling2d_2 (MaxPooling2 (None, 30, 30, 128)       0
conv2d_3 (Conv2D)            (None, 28, 28, 256)       295168
batch_normalization_3 (Batch (None, 28, 28, 256)       1024
max_pooling2d_3 (MaxPooling2 (None, 14, 14, 256)       0
conv2d_4 (Conv2D)            (None, 12, 12, 128)       295040
batch_normalization_4 (Batch (None, 12, 12, 128)       512
max_pooling2d_4 (MaxPooling2 (None, 6, 6, 128)         0
conv2d_5 (Conv2D)            (None, 4, 4, 64)          73792
batch_normalization_5 (Batch (None, 4, 4, 64)          256
max_pooling2d_5 (MaxPooling2 (None, 2, 2, 64)          0
flatten (Flatten)            (None, 256)               0
dense (Dense)                (None, 256)               65792
batch_normalization_6 (Batch (None, 256)               1024
dense_1 (Dense)              (None, 64)                16448
batch_normalization_7 (Batch (None, 64)                256
dropout (Dropout)            (None, 64)                0
dense_2 (Dense)              (None, 1)                 65
activation (Activation)      (None, 1)                 0
=================================================================
Total params: 843,521
Trainable params: 841,537
Non-trainable params: 1,984
```

Figure 4.4: Summary Table of the Proposed CNN Model

4.3.2 Architecture of Pre-trained models

For pre-trained models, it's the neural network that has been employed in prior situations and has gained information which could be brought into new targeted samples. Five pre-trained models have been used for the experiment, which are : Inception V3, VGG-16, VGG19, DenseNet121, ResNet50. The details of each model is given below:

- **Inception V3:** An InceptionV3 network's architecture [27] is created one step at a time, as described below:

  1. Factorized Convolutions: This decreases the number of parameters in a network, which improves computational efficiency. It also monitors the efficiency of the network.

  2. Smaller Convolutions: Substituting smaller convolution operations for larger convolution operations results in much quicker training. A 5*5 filter, for example, has 25 properties; two 3*3 filters, in place of a 5*5

convolution, has just 18 (3×3 + 3×3) properties.

3. Asymmetric Convolutions: : Instead of a 3*3 convolution, a 1*3 convolution preceded by a 3*1 convolution may be used. The number of parameters would be somewhat greater than the asymmetric convolution described if a 3*3 convolution was substituted with a 2*2 convolution.

4. Auxiliary Classifier: while training, an auxiliary classifier is used as a small CNN embedded among layers and the loss is incorporated in the actual network loss. The auxiliary classifier functions as a regularizer in InceptionV3.

5. Grid Size Reduction: Pooling procedures are commonly used to reduce grid size. However, a more effective approach is given to overcome computational cost bottlenecks in the Figure 4.5 below:

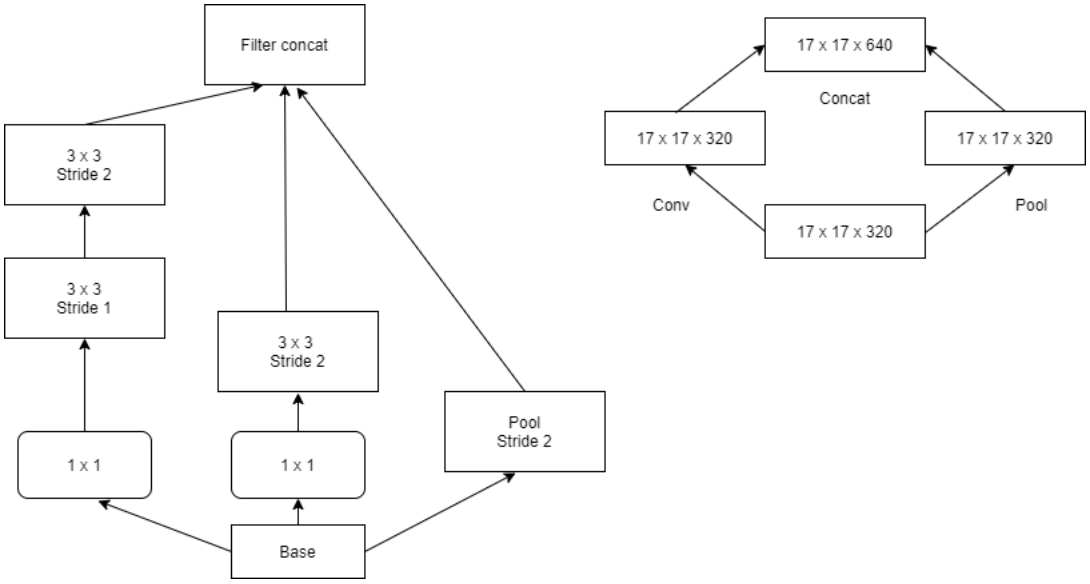


Figure 4.5: Grid Size Reduction of InceptionV3

The generic architecture of InceptionV3 is given below in figure 4.6 :

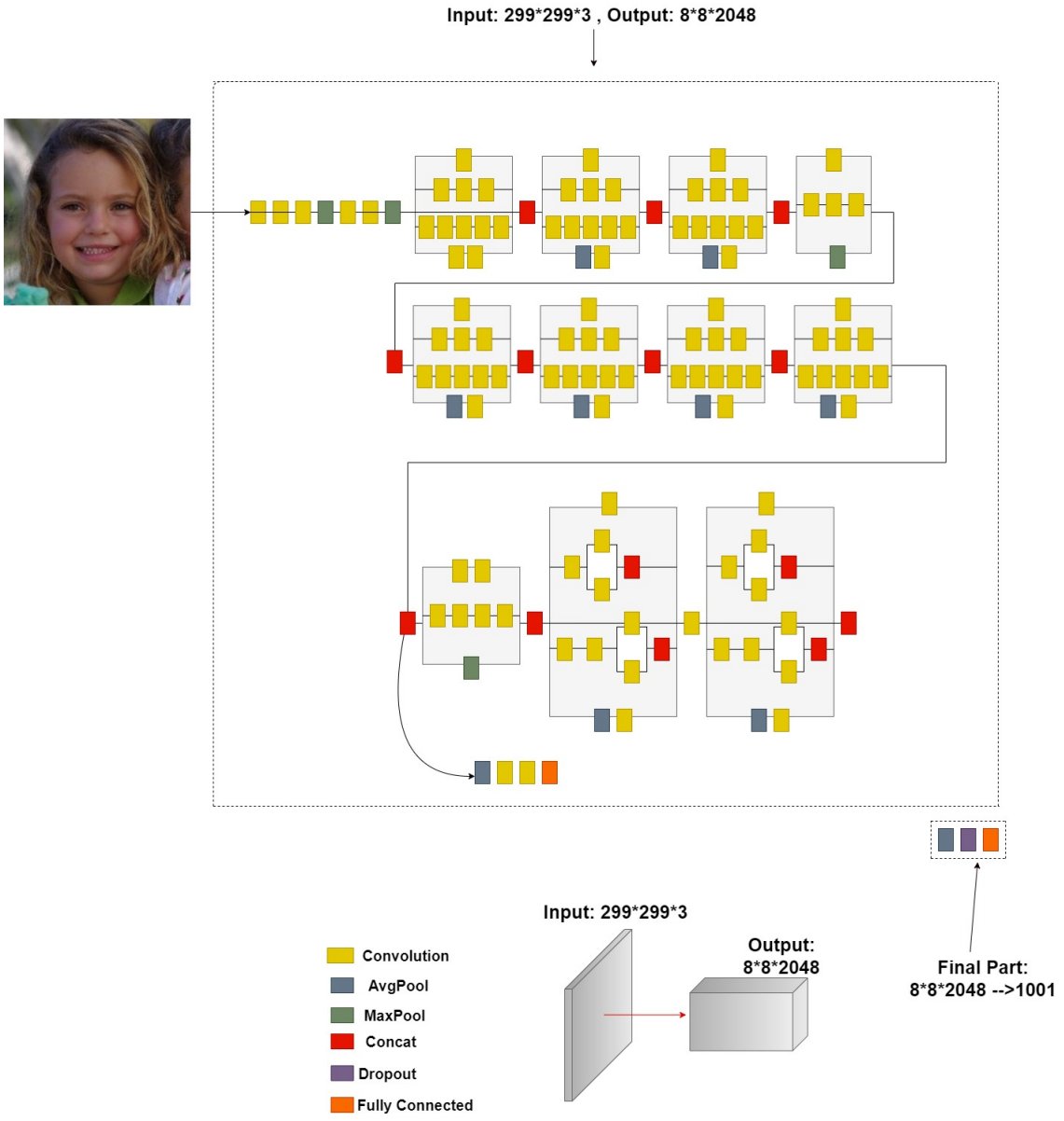


Figure 4.6: Generic Architecture of InceptionV3

VGG-16: An input to the convolutional-1 layer is a 224 x 224 RGB picture with a predefined size. The image is processed by a set of convolutional layers with an extremely tiny field: 3*3 the small size to retain the notion of right, down, left, up and center. In one of the setups It also uses 1*1 convolution filters, which might be considered as a linear change of the input channels. The spatial padding of convolutional layer input is set to one pixel for 3 x 3 convolutional layers and the stride of convolution is set to one pixel, so that the spatial resolution is retained after convolution. Five max-pooling layers, which precede part of the convolutional layers, do spatial pooling. With stride 2 across a 2*2 pixel frame Max-pooling is done .Following a sequence of convolutional layers of varying depth in various designs, 3 fully-connected layers are added where the first two have 4096 number of channels individually, while the third performs categorization and therefore has 1,000 channels [31]. The activation layer is the last layer. In all networks, the completely linked levels are configured in the same way. Figure 4.7 shows the summary of VGG-16 model.

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

Total params: 14,714,688
Trainable params: 0
Non-trainable params: 14,714,688

Figure 4.7: Summary Table of VGG-16 model

The architecture of VGG-16 is given below in figure 4.8:

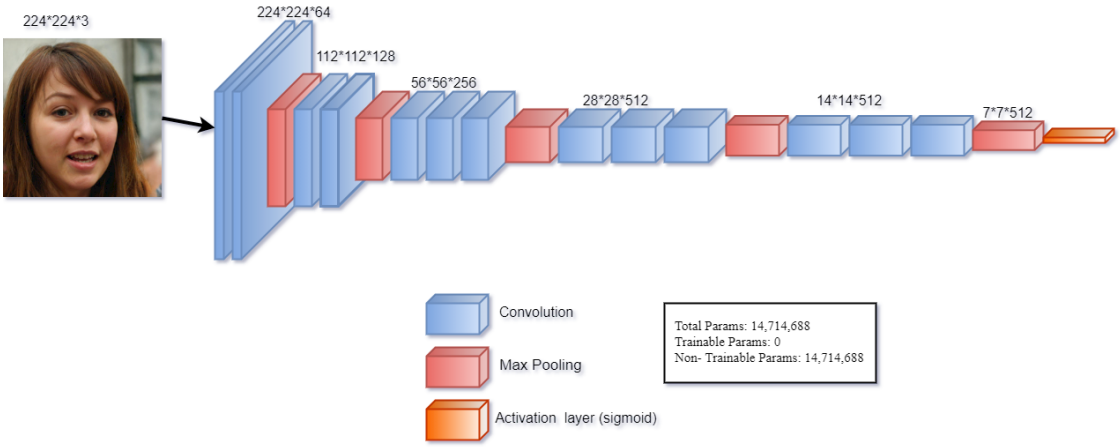


Figure 4.8: Architecture of VGG-16

DenseNet121: For this one, we have given (224 × 224) RGB images as input and the required elements of this network are- DenseBlocks [21], Growth Rate, Connectivity, Bottleneck layers [36]. DenseNet-121 has 4 AvgPool [19] along with 120 Convolutions. [59] Here, The bottleneck layer is a 1x1 kernel [10], while the convolution process is executed by a 3x3 kernel. A 1*1 convolutional with 2 × 2 average pooling layer with a stride [13] of 2 are further included in each transition layer. As a result, a fundamental convolution layer with 64 (7X7) filters and a stride of 2 is created and, maximum pooling of 3x3 and a stride of 2 is there also. Dense Block 1 has 2 convolutions performed 6 times, Dense Block 2 has 2 convolutions replicated 12 times, Dense Block 3 has 2 convolutions tested 24 times, Dense Block 4 has 2 convolutions extended 16 times. The Global Average Pooling layer includes all of the network's feature mappings for classifications and distribution. All layers, including those together in the same dense block and transition layers [3], distribute their parameters over various inputs, allowing hidden layers to make use of characteristics collected earlier in the process [52]. Because transition layers produce a lot of duplicated information, the layers in the 2nd and 3rd dense blocks give the transition layers' output the least weight. The architecture of DenseNet121 is given below in Figure 4.9:

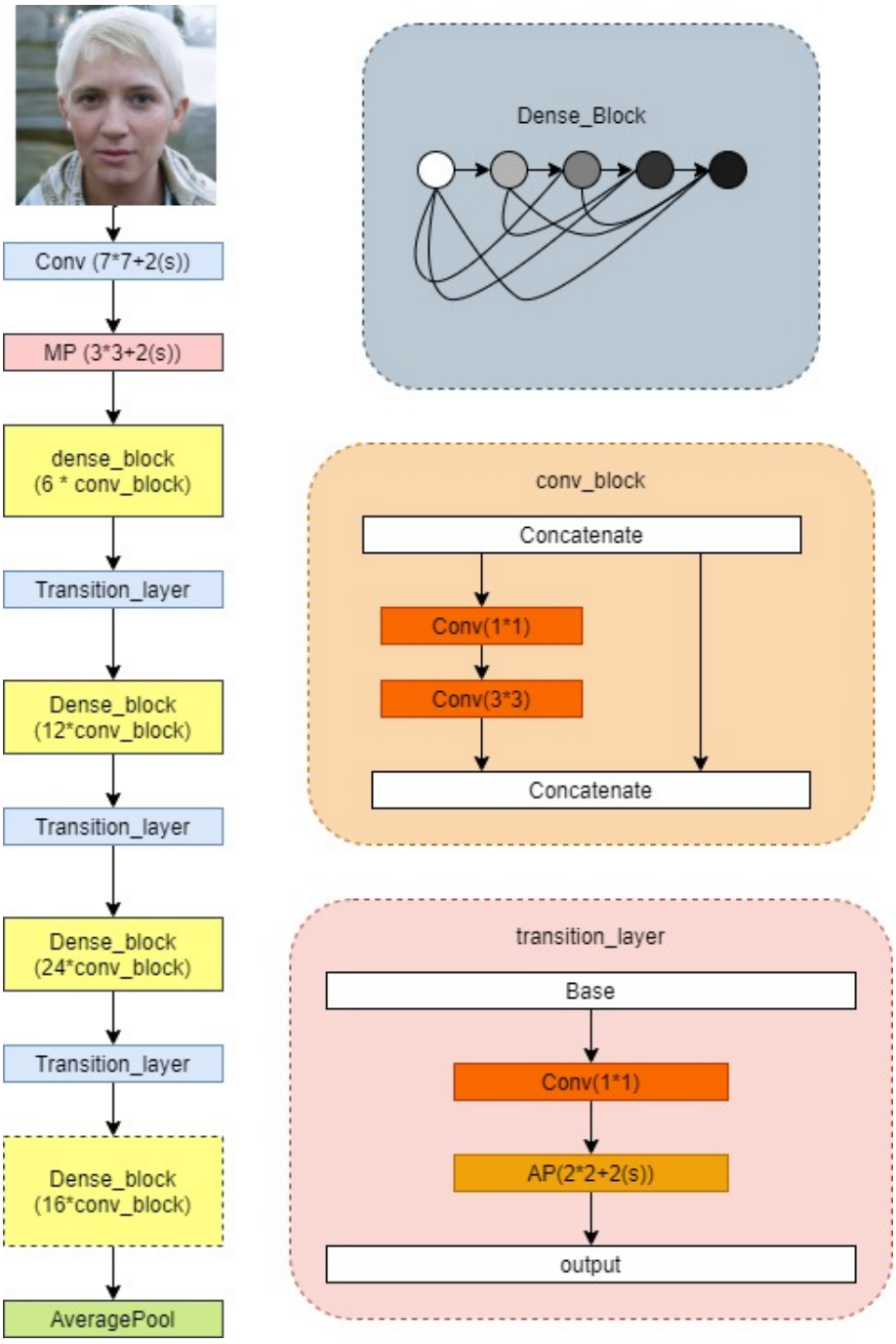


Figure 4.9: Architecture of DenseNet121

VGG-19: This network was RGB picture as input, given a specific size (224×224) [38], indicating that the matrix format was 224, 224, 3. Only one pre-processing has been employed to remove the mean RGB value from every pixel, which was calculated over the entire training dataset. The kernels with a size of (3×3) and a size of one pixel for strides to cover the entire picture concept. To keep the image's spatial resolution, spatial padding has been utilized. With stride 2, max pool was achieved over a 2×2 pixel frame. The summary table for the said model is given below in Figure 4.10:

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_conv4 (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_conv4 (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_conv4 (Conv2D)	(None, 14, 14, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0

Total params: 20,024,384
Trainable params: 0
Non-trainable params: 20,024,384

Figure 4.10: Summary Table of VGG-19 model

The architecture of VGG-19 is given below in Figure 4.11 :

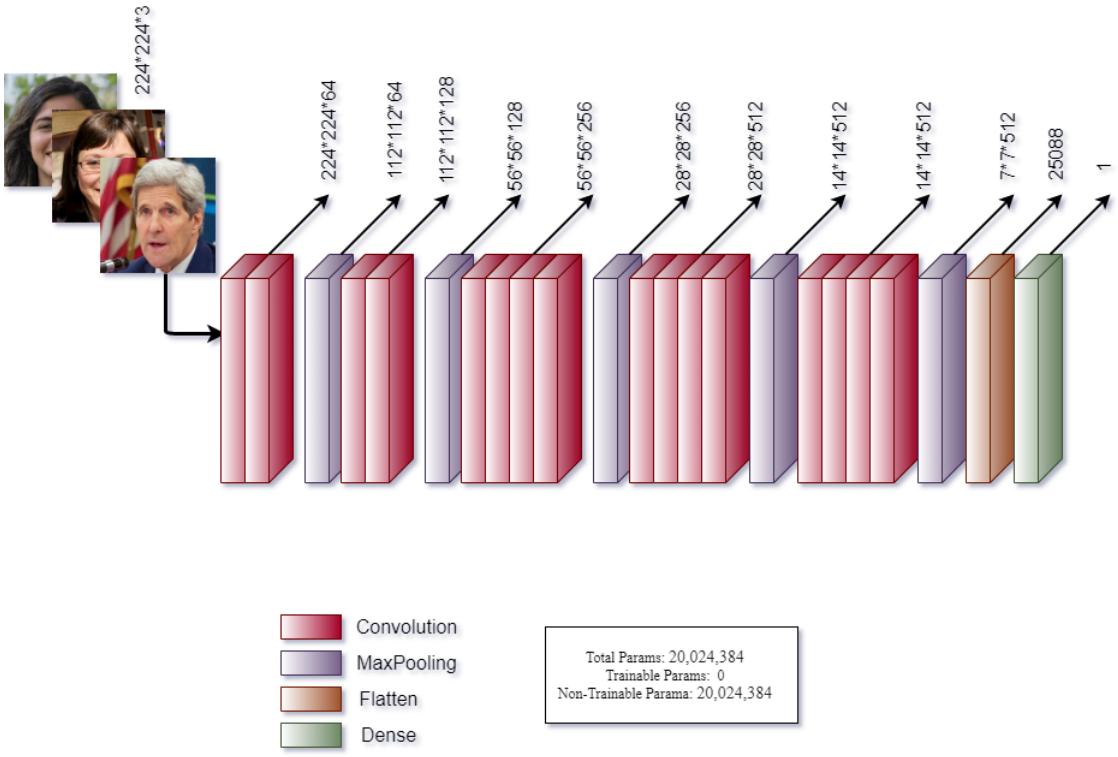


Figure 4.11: Architecture of VGG-19

ResNet50: The ResNet 50 architecture, has the following elements: There is 1 layer from a convolutional layer having kernel size of 7*7 with 64 different kernels, each having stride size 2. Then we have a max pool layer having a stride length of 2. There is a 1×164 sized kernel in the later convolution, following by a 3×364 sized kernel, and lastly 1×1256 sized kernel. These 3 layers are executed in triplicate altogether, providing us 9 layers in this phase. Followed by it is a kernel of 1×1128, following by a kernel of 3×3128, and finally a kernel with 1×1512. This procedure is conducted four times, providing us a total of twelve layers. Then there's a 1×1256 kernel, followed by 3×3256 and 1×11024 kernels, which are replicated 6 times for a total of 18 layers.Then a 1 × 1512 kernel was added, followed by two additional 3 × 3512 and 1 × 12048 kernels, for a sum of 9 layers. Then, we run an average pool and finish with a fully connected layer with 1000 vertices, followed by an activation function, giving us one layer. The activation functions and the max or average pooling layers are not counted. As a result, we get a Deep Convolutional network with $1 + 9 + 12 + 18 + 9 + 1 = 50$ layers [24] . The summary table for the aforementioned model is given below in Figure 4.12:

Layer (type)	Output Shape	Param #
resnet50 (Functional)	(None, 2048)	23587712
dense (Dense)	(None, 1)	2049

Total params: 23,589,761
Trainable params: 2,049
Non-trainable params: 23,587,712

Figure 4.12: Summary Table of ResNet50 model

The architecture of ResNet50 is given below in Figure 4.13:

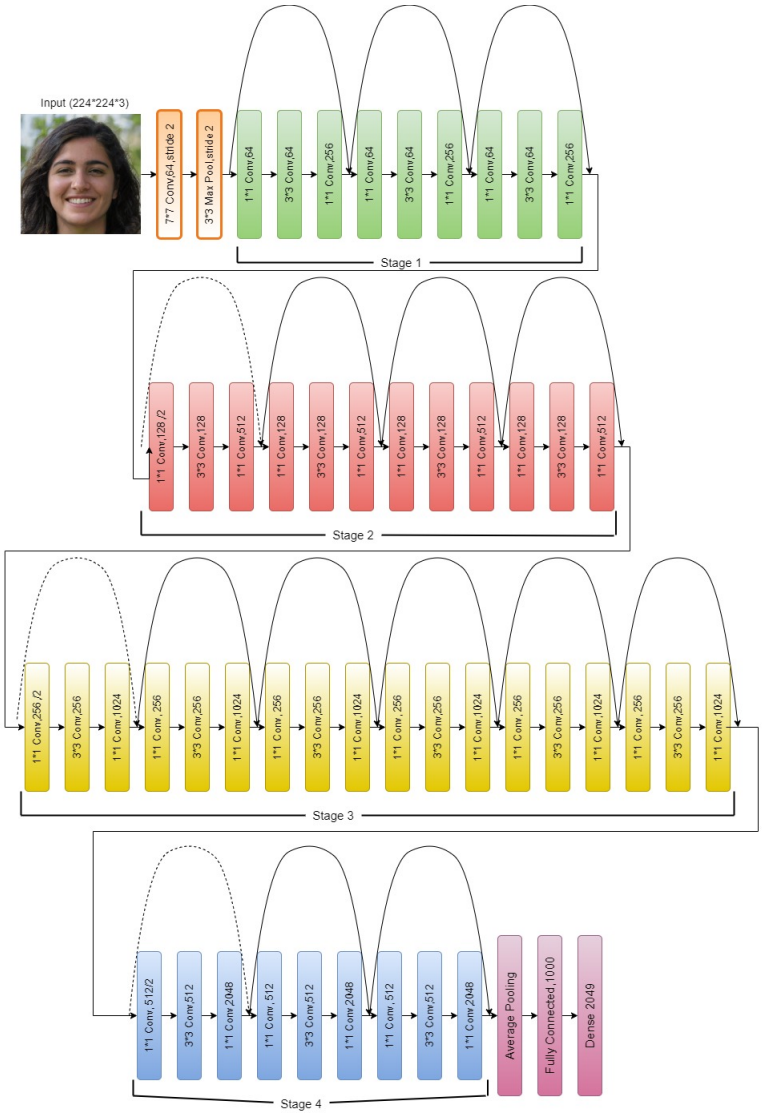


Figure 4.13: Architecture of ResNet50

Researchers are presently using pre-trained models, as shown in Table 4.1. The table shows how parameters and depth there are in the network and how much input picture size is required for training [51]. As a result of these models, DeepFake pictures are classified.

Table 4.1: Attributes of Pre-trained Models

Pre-Trained Model	**Input Size**	**Depth**	**Network Size(MB)**	**Parameters**
InceptionV3	299×299	159	89	23,851,784
VGG16	224×224	23	528	138,357,544
DenseNet121	224×224	121	33	8,062,504
VGG19	224×224	26	549	143,667,240
ResNet50	224×224	-	98	25,636,712

Here, VGG16 and VGG19 are series-connected networks, while all other networks are directed acyclic graphs [49]. To access and use these models, call the built-in Keras function and Transfer learning application in MATLAB, Google Colab, Jupyter Notebook and other platforms that support these technologies. For example, in paper[40], researchers used MATLAB to classify electronic devices. There are predefined weights/features in each of the models that make processing and recognizing layers easier. In this work, five pre-trained models are used to analyze the data. With DeepFake pictures to categorize, we want to see which model performs best in terms of accuracy, loss, recall, precision, execution time, network size, and validity.

Chapter 5

Deployment of Parameter

The dataset training for this model against the suggested network is dependent on some parameters, which are batch size, epoch, learning rate, optimizer, callbacks, and regularisation. Training is enabled after pre-processing the training images. Properties in the Transfer Learning approach can be changed before the training begins. The custom CNN model and the pre-trained models are selected before the machine learning process is initiated and the directory from the two generated categories is imported according to the required first layer size of the input. All data has been divided into two parts: 75% for training and 25% testing. Out of the 13000 images, ten thousand are for training, while the other three thousand are for testing. The optimizer that has been utilized is "Adam" [35] which applies a gradient-based approach focused on innovative predictions of moments that are of lower-order to improve Stochastic objective functions. Optimizer "Adam" is used because the approach is so simple to develop, computationally efficient, requires minimal memory, and is invariant to diagonal rescaling of the gradients, it is ideally suited for situations with huge amounts of data and/or parameter values [18]. The initial learning rate for the optimizer choices is set at 0.001. Low values result in sluggish training while excessive values prevent optimal convergence from taking place. Standard values for L1_L2 Regularization and Momentum are 0.001 and 0.9, respectively. To prevent overfitting, the first value is used, while the second is used to calculate the percentage of contribution from the prior simulation to the following. We've utilized 32 batches and 20 epochs (each epoch, there are 313 iterations). To add to the difficulty of learning, the training and validation data are mixed together before going on to the next epoch. And the color mode for each image is set to RGB. We have used an Activation Function called "sigmoid". Sigmoid function has been used due to its range from zero to one. Consequently, it is helpful for models that need prediction of the probability as an outcome. It's a no-brainer to use the sigmoid because probability exists only between zero and one. Under the loss section, we have used "binary cross entropy". Comparing expected probabilities to actual class output, which might be 0 or 1, is called binary cross entropy. If there is a deviation from what is predicted, it penalizes the probability. How near or how distant is this value from what it should be. We've set the class mode to "binary" in order to forecast the result. In this case, the output might be either zero or one because we are dealing with a binary situation. Lastly, The "rgb" color mode was selected in the color mode section, which means that the pictures would be converted to 3 channels. In addition to training alternatives, the execution environment has

a bearing on training completion times as well. In all experiments, the execution environment is a graphics processing unit (gpu). The random GPUs utilized by Google Colab were Nvidia Tesla K80 [63], Nvidia Tesla T4, and Nvidia Tesla P4 for benchmarking and informational purposes. Type "!nvidia-smi" on Google Colab command line to examine the GPU's parameters.
Reiteration of the optimizer is discussed in the context of the training process as a deep-learning method. The equation(5.1) describes the "Adam" optimizer function in generic terms.

$$\omega_{t+1} = \omega_t - \alpha m_t \tag{5.1}$$

$m_t =$ aggregate of gradients at time t
$\alpha =$ learning rate at time t
$\omega_t =$ weights at time t
$\omega_{t+1} =$ weights at time $t+1$

Equation(4.1) belongs to the Activation Function "Sigmoid" [7] which we have discussed before. Equation(5.2) and (5.3) describes L1_L2 regularization in generic terms. The L1_L2 regularizer applies both L1 and L2 penalties to a given set of data.

$$L1: \text{ Cost } = \sum_{i=0}^{N} \left(y_i - \sum_{j=0}^{M} x_{ij} W_j \right)^2 + \lambda \sum_{j=0}^{M} |W_j| \tag{5.2}$$

$$L2: \text{ Cost } = \sum_{i=0}^{N} \left(y_i - \sum_{j=0}^{M} x_{ij} W_j \right)^2 + \lambda \sum_{j=0}^{M} W_j^2 \tag{5.3}$$

Equation(5.4) describes the loss function for 2 class binary classification in generic terms.

$$H_p(q) = -\frac{1}{N} \sum_{i=1}^{N} y_i \cdot \log\left(p\left(y_i\right)\right) + \left(1 - y_i\right) \cdot \log\left(1 - p\left(y_i\right)\right) \tag{5.4}$$

Here,

$p\left(y_i\right)$ is the probability of class 1, and $\left(1 - p\left(y_i\right)\right)$ is the probability of class 0.

By clicking the "model.fit" cell in the code, training begins. Below the cell is a progress bar that updates the epoch and steps of the process. As soon as the specified epoch reaches its finish, it signifies that the main validation parameter has been attained, based on the chosen pre-trained model with all parameters and matrices in it.
Table 5.1 lists the parameters and their values below for the pre- trained models :

Table 5.1: Parameters Used for the Pre-trained Models and the 18-Layer CNN Model

Parameter	**Pre- trained models (Value)**	**18-layered CNN Models (Value)**
Training Data	75%	75%
Testing Data	25%	25%
Target Size	(224, 224)	(256,256)
Batch Size	32	32
Epoch	20	20
Step	313	313
Verbose	1	1
Initial Learning Rate	0.001	0.1
Minimum Learning Rate	-	0.000001
L1_L2 Regularization	0.001	0.001
Momentum	0.9	0.9
Execution Environment	GPU	GPU
Optimizer	Adam	Adam
Loss Function	Binary CrossEntropy	Binary CrossEntropy
Activation Function	Sigmoid	Sigmoid
Class Mode	Binary	Binary
Color Mode	RGB	RGB
Metrics	Accuracy, Loss, Precision, Recall	Accuracy, Loss, Precision, Recall
Callback	ReduceLROnPlateau	ReduceLROnPlateau

We have calculated the accuracy, precision, recall and AUC performance [8], [15], [20]for each of the models. The equations are described below:

- Equation for Accuracy:

$$\text{Accuracy} = \text{True Positive} + \frac{\text{True Negative}}{\text{Condition Positive}} + \text{Condition Positive} + \text{Condition Negative} \tag{5.5}$$

- Equation for Precision:

$$\text{Precision} = \text{TruePositive} \times \frac{\text{True Positive Rate}}{\text{TruePositive}} + \text{FalsePositive} \tag{5.6}$$

- Equation for recall:

$$\text{Recall} = \text{True Positive} \times \frac{\text{True Positive Rate}}{\text{True Positive}} + \text{False Negative} \tag{5.7}$$

- AUC: The Area Under the Curve (AUC) is a metric for a classifier's ability to differentiate among classes [8]. It simply denotes the scale on which it is measured or the degree of separability. It offers an accumulated performance measure throughout all classification thresholds because AUC is scale and classification threshold invariant. The AUC rate indicates how well the model distinguishes between positive and negative classes. Higher AUC rate indicates better performance of any given model.

Chapter 6

Appraisal of Performance

6.1 Performance of CNN Model

We selected a total of 3000 images for testing, separated into two portions for real and fake classes, each comprising 1500 images. Testing data occupies 25% of the entire data, whereas Training data accounted for 75%. Finally, our proposed model was shown to be accurate to the tune of 98.77 percent. The results for testing and training accuracy and loss are shown in Table 6.1.

Table 6.1: Training and Testing Accuracy and Loss of Proposed CNN Model

Training Accuracy	**Testing accuracy**	**Training Loss**	**Testing Loss**
99.82%	98.77%	0.63%	5.87%

The table shows the suggested model provides 98.77% testing accuracy. In addition to that, the visual of the training and validation or testing results through graphs are given below in Figure 6.1 and 6.2.

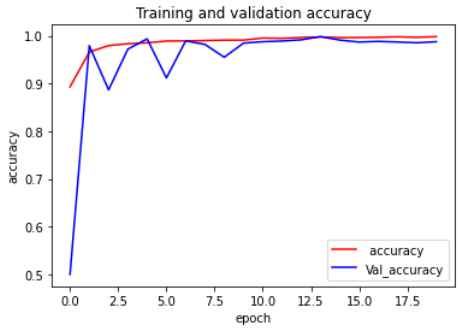


Figure 6.1: Training and testing accuracy

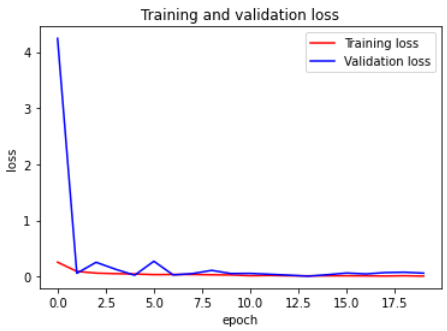


Figure 6.2: Training and Validation loss

Moreover, the metrics results of accuracy, loss, auc, recall, summary for training and validation class are shown below in Figure 6.3 and 6.4:

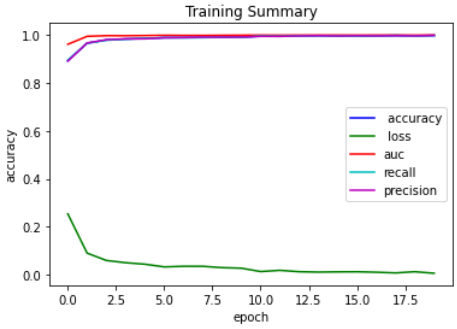


Figure 6.3: Training Summary Graph

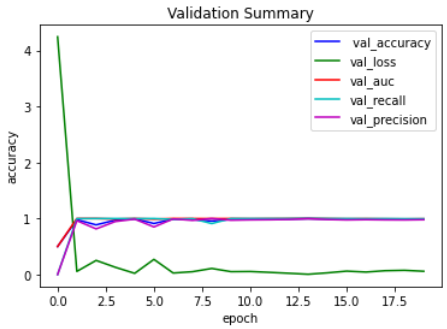


Figure 6.4: Testing Summary Graph

Further classification report is shown in Table 6.2.

Table 6.2: Classification Report for the Proposed CNN Model

Class	Precision	Recall	F1-Score	Support
Real	0.99	1.00	1.00	150
Fake	1.00	0.99	1.00	150
Accuracy	-	-	1.00	300
Macro avg	1.00	0.51	1.00	300
Weighted avg	1.00	1.00	1.00	300

The confusion metrics for the CNN model is given below in Table 6.3:

Table 6.3: Confusion Matrix

150	0
1	149

6.2 Performance of Pre-trained Models

A total number of 3000 images have been divided into two sections where each section consists of 1500 images. The two classes are named as Fake and Real. However, these images have been tested against the pre-trained models. From the results, it can be concluded that InceptionV3 acquired the best training and testing accuracy of 97.76% and 97.10%. According to the results, VGG-16, DenseNet121, VGG-19 and ResNet50 respectively achieved their accuracy in descending order. Figure 6.5 and 6.6 shows the training accuracy of the models and the testing accuracy and Figure 6.7 shows the AUC rate comparison :

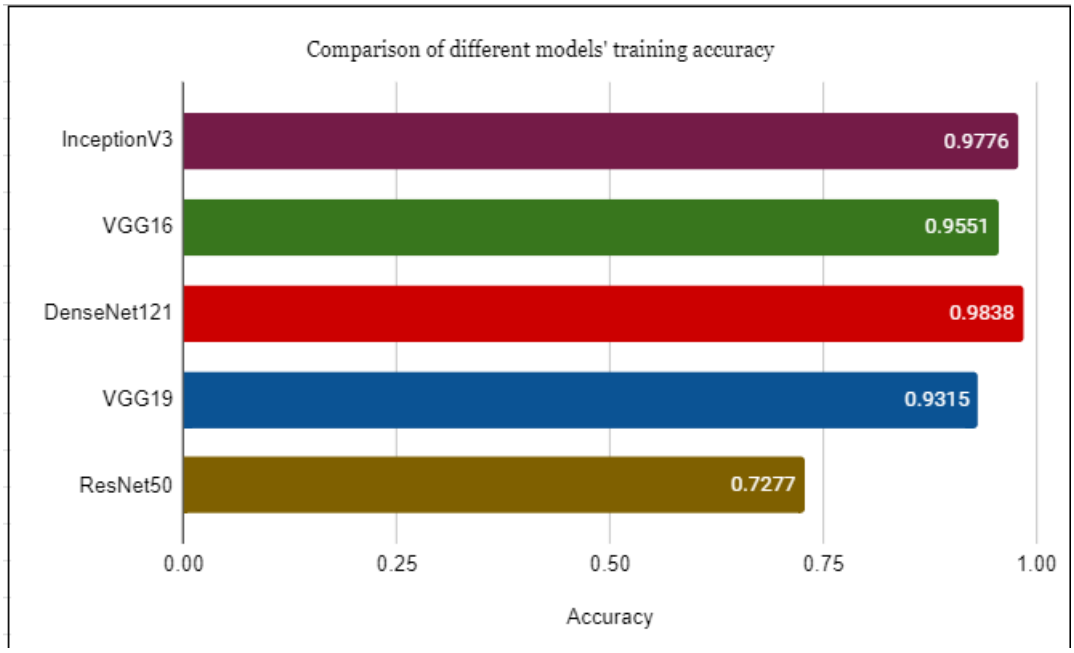


Figure 6.5: Training accuracy of the tested pre-trained models

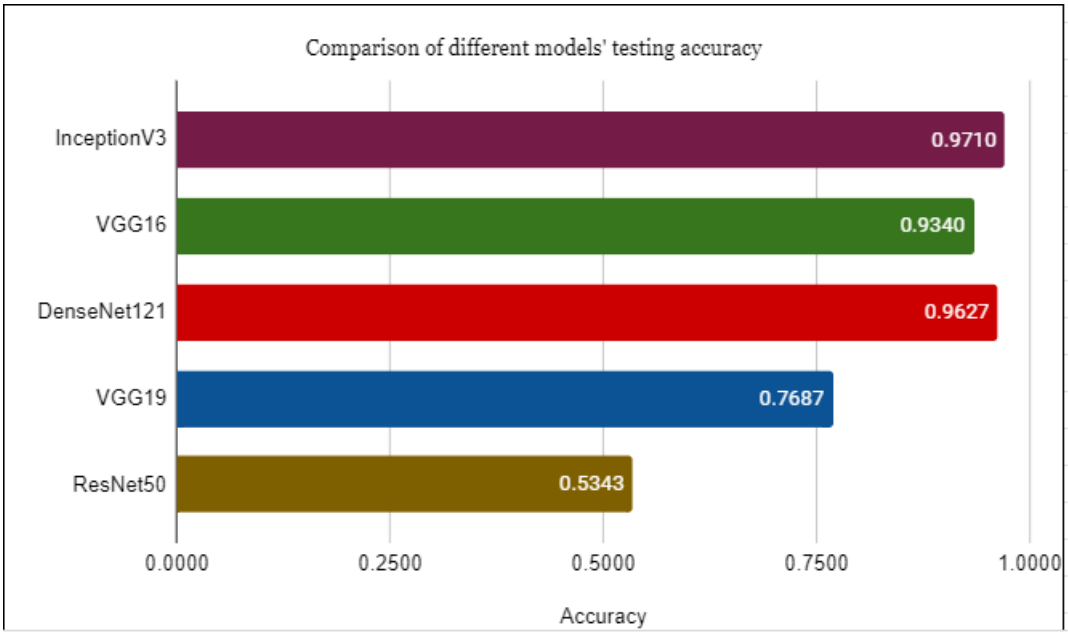


Figure 6.6: Testing accuracy of the tested pre-trained models

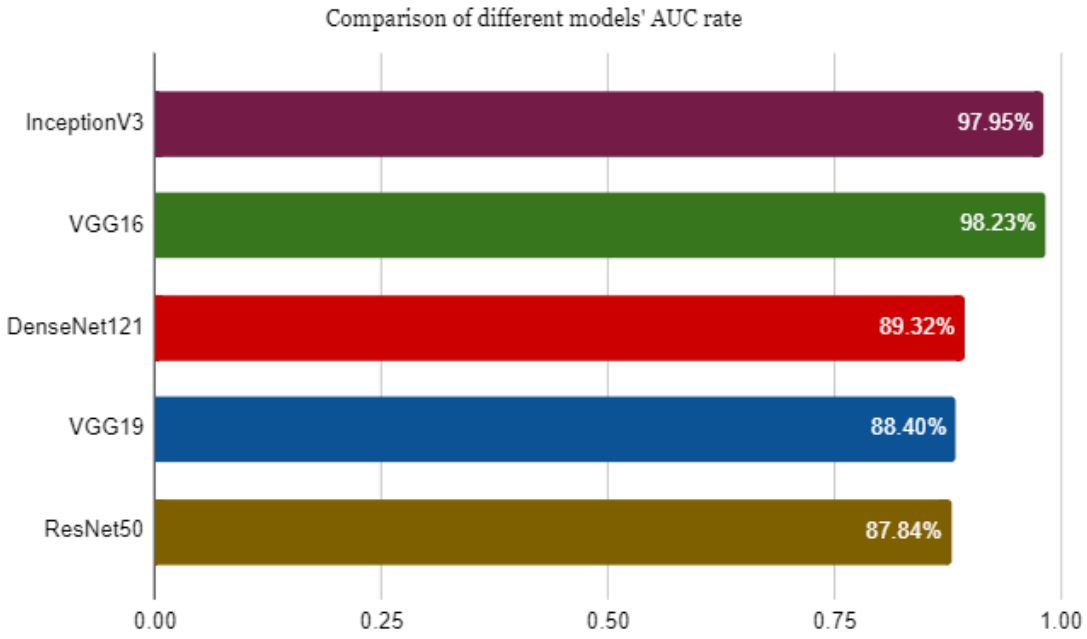
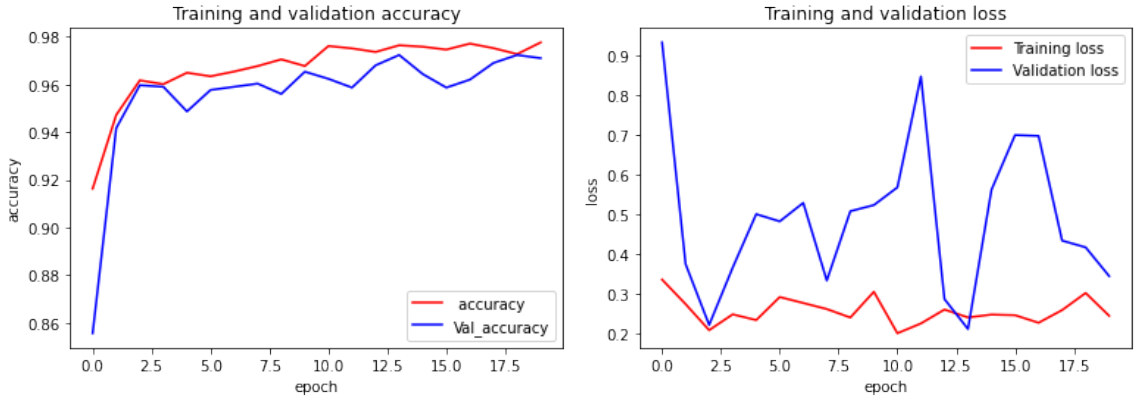


Figure 6.7: AUC rates of the used pre-trained models

For further analysis the training and testing accuracy and loss graphs and the summary of metrics graphs are given below:

- **InceptionV3:** Figure 6.8 shows Training and testing accuracy, loss and summary of InceptionV3



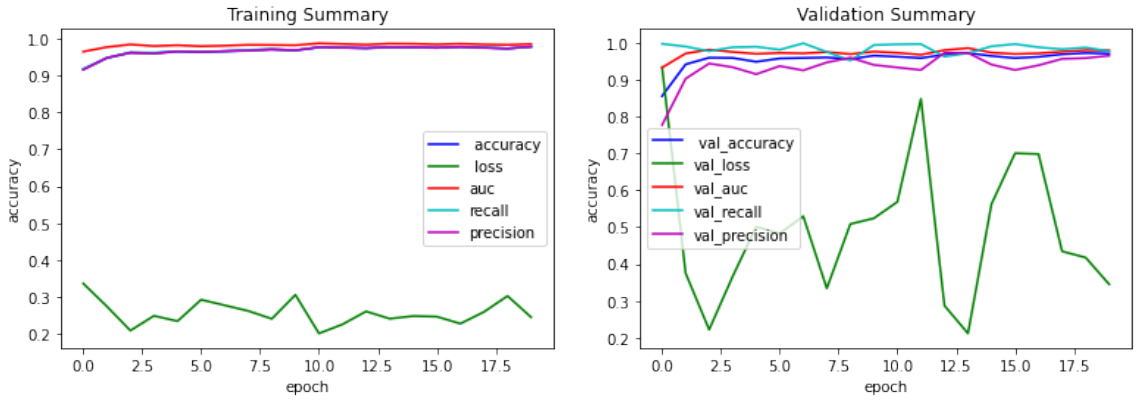


Figure 6.8: InceptionV3 Training and Testing graphs

- **VGG-16:** Figure 6.9 shows Training and testing accuracy, loss and summary of VGG-16 :

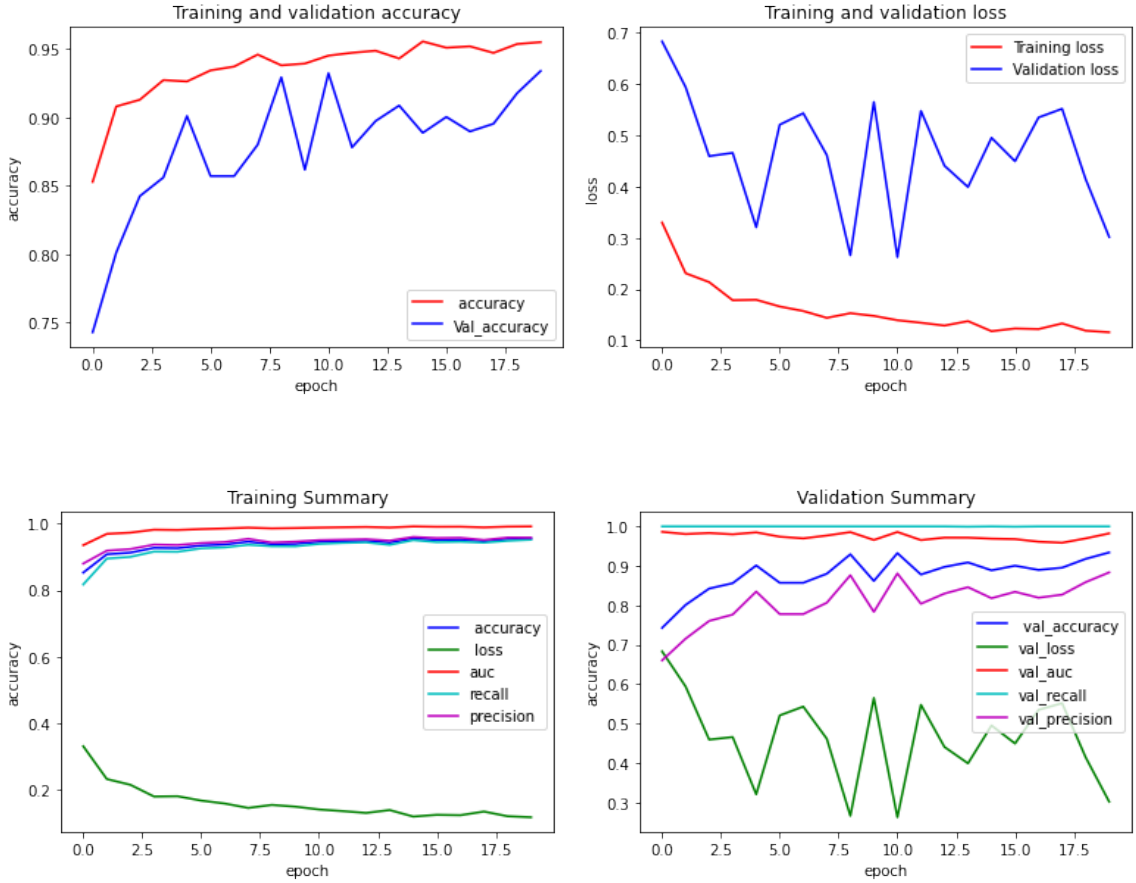


Figure 6.9: VGG-16 Training and Testing graphs

- **VGG19:** Figure 6.10 shows Training and testing accuracy, loss and summary of VGG-19 :

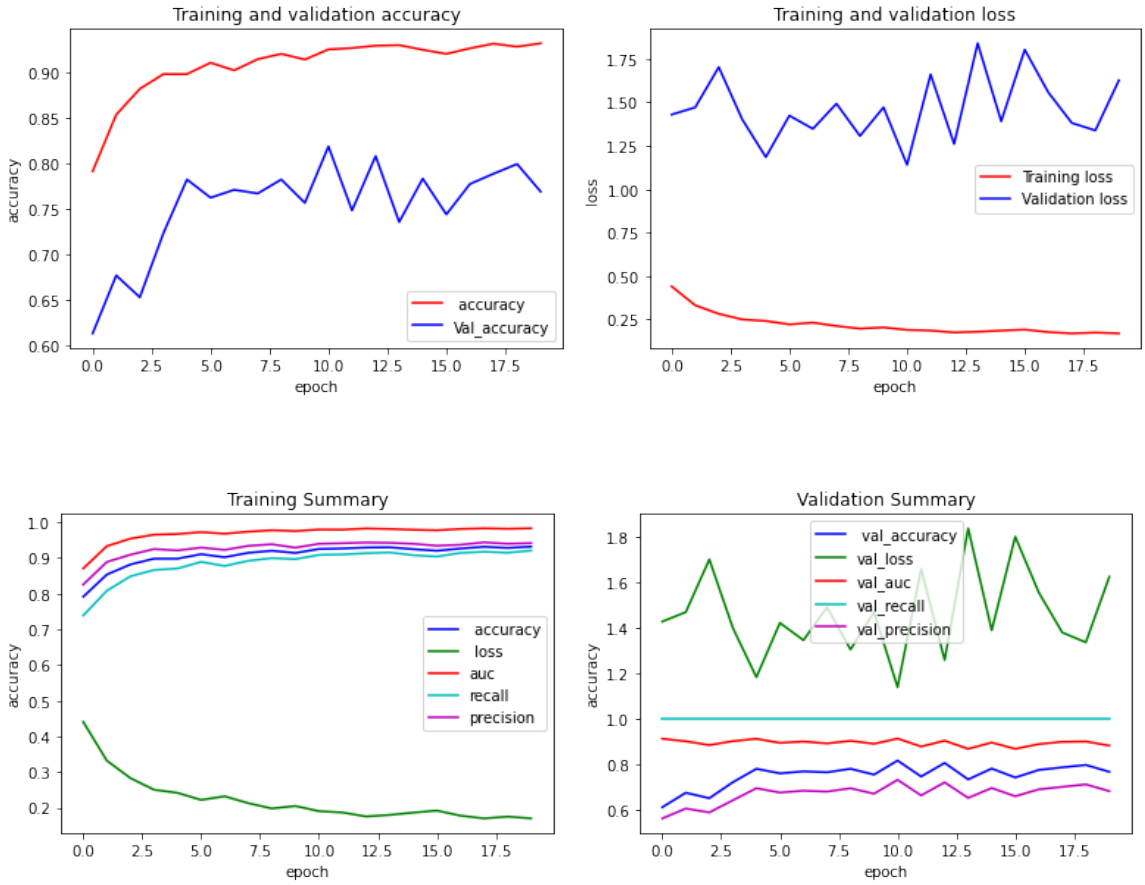
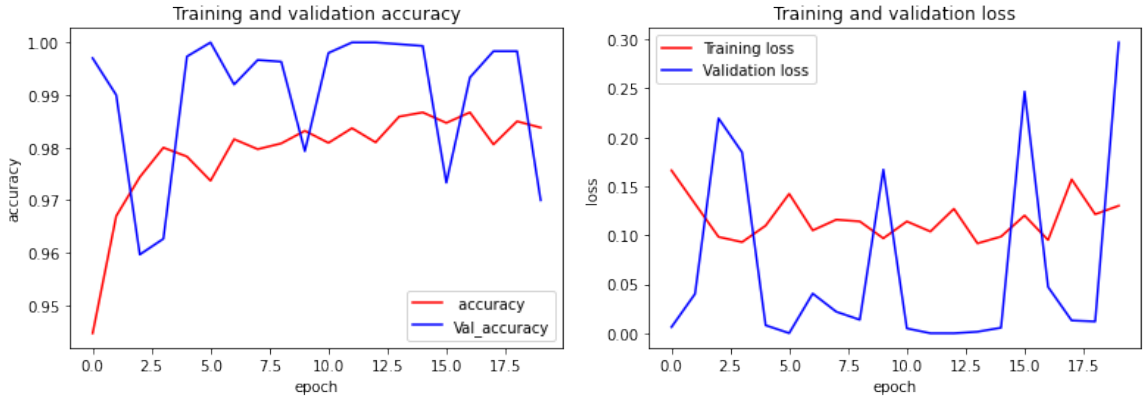


Figure 6.10: VGG-19 Training and Testing graphs

- **DenseNet121:** Figure 6.11 shows Training and testing accuracy, loss and summary of DenseNet121 :



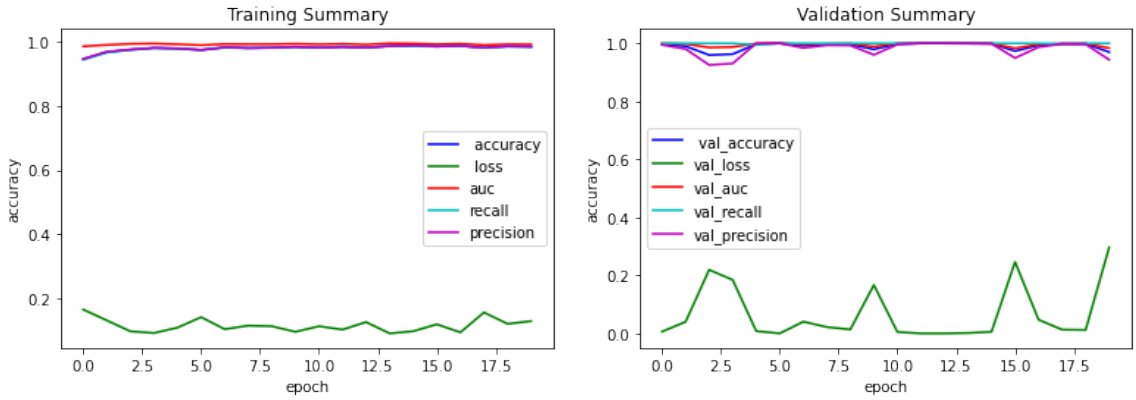


Figure 6.11: DenseNet121 Training and Testing graphs

- **ResNet50:** Figure 6.12 shows Training and testing accuracy, loss and summary of ResNet50 :

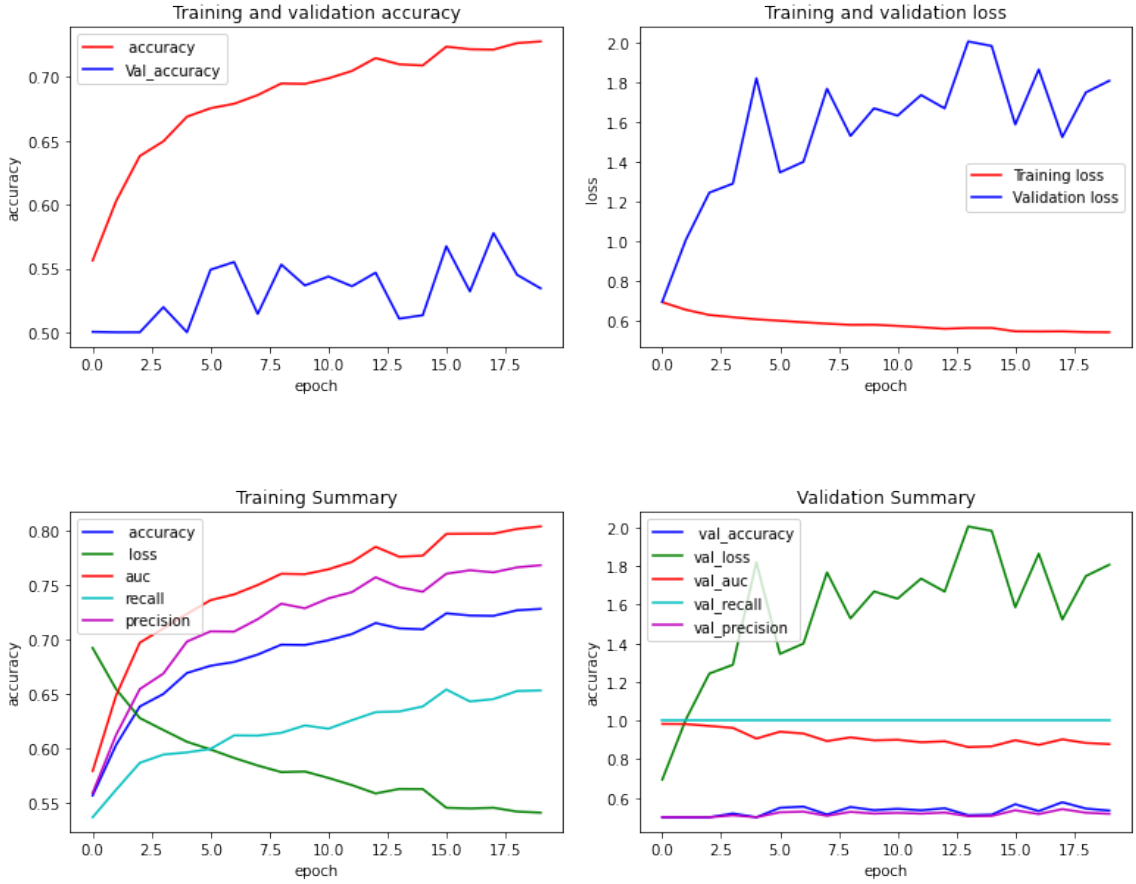


Figure 6.12: ResNet50 Training and Testing graphs

A separate directory called Unseen set is created, which contains 300 images for generating the necessary metrics and classification report. This section shows how the pre-trained models act on random images. The necessary metrics such as, Recall,Confusion matrix,Precision, F1 score and accuracy have been generated using the said section. Table 6.4 shows the metrics values.

Table 6.4: Comparison of Several Evaluation Metrics for Various Models

Model	Precision		Recall		F1-Score		Accuracy
	Real	Fake	Real	Fake	Real	Fake	
Inception V3	0.95	0.98	0.97	0.95	0.97	0.96	0.96
VGG16	0.82	1.00	11.00	0.77	0.90	0.87	0.89
DenseNet121	0.75	1.00	1.00	0.67	0.86	0.80	0.83
VGG19	0.69	1.00	1.00	0.55	0.82	0.71	0.77
ResNet50	0.54	1.00	1.00	0.15	0.70	0.27	0.58

6.3 Compare and Analysis

From the above mentioned tables and figures it can be concluded that, InceptionV3 shows the results among all the transfer learning based pre-trained models. If we compare the results of the suggested 18-layered CNN model and the pre-trained models, we can reach to a conclusion that the suggested custom CNN model approach gives us higher accuracy in case of detecting deep fake images. Our proposed model was shown to be accurate to the tune of 98.77 percent where InceptionV3 gives an accuracy of 97.10 percent. Apart from that, Table 6.5 shows the comparison table between the accuracy rate of transfer learning based models or pretrained models and our suggested CNN model against our dataset. However, in both cases, we came to the conclusion that our approach towards detection of Real versus DeepFake images has shown promising results with higher accuracy rates rather than previous attempts.

Table 6.5: Training and Testing Accuracy Comparison between the pre-trained and CNN model

Model	Train	Test
InceptionV3	97.76%	97.10%
DenseNet121	98.38%	96.27%
VGG16	95.51%	93.40%
VGG19	93.15%	76.87%
RESNET50	72.77%	53.43%
Proposed CNN model	99.82%	98.77%

Chapter 7

Conclusion and Future Works

7.1 Conclusion

This paper provides a deep CNN model that can efficiently identify Real and DeepFake images with the highest of 98.77% accuracy. With 13000 images divided into two classes, our model provides a structured, efficient and effective way of distinguishing between Real and Deep fake images. Moreover, comparing the model with previous related works and transfer learning methods, we came to the conclusion that the proposed 18-layered CNN model is a better approach when it comes to deepfake image detection. The suggested model delivers higher accuracy than the other methods. Although we aim to make the model more efficient and better in future.

7.2 Future Work

As a scope for future work, our model can be dilated for extracted frames of videos. It can then detect DeepFake image frames from videos as well. Moreover, the model may also be able to handle multiclass problems. Other than that, it may also become viable for android or ios application systems. A mobile application may be created based on the model so that DeepFake detection becomes easier in order to enlarge the utilization of the technique.

Bibliography

[1] F. Freese *et al.*, "Testing accuracy.," *Forest Science*, vol. 6, no. 2, pp. 139–45, 1960.

[2] E. D. Pulakos, "A comparison of rater training programs: Error training and accuracy training.," *Journal of Applied Psychology*, vol. 69, no. 4, p. 581, 1984.

[3] J. K. Hale and K. Sakamoto, "Existence and stability of transition layers," *Japan Journal of Applied Mathematics*, vol. 5, no. 3, pp. 367–405, 1988.

[4] M. Buckland and F. Gey, "The relationship between recall and precision," *Journal of the American society for information science*, vol. 45, no. 1, pp. 12–19, 1994.

[5] J. G. Greeno and R. P. Hall, "Practicing representation: Learning with and about representational forms," *Phi Delta Kappan*, vol. 78, pp. 361–367, 1997.

[6] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[7] K. Shibata and K. Ito, "Gauss-sigmoid neural network," in *IJCNN'99. International Joint Conference on Neural Networks. Proceedings (Cat. No. 99CH36339)*, IEEE, vol. 2, 1999, pp. 1203–1208.

[8] J. Myerson, L. Green, and M. Warusawitharana, "Area under the curve as a measure of discounting," *Journal of the experimental analysis of behavior*, vol. 76, no. 2, pp. 235–243, 2001.

[9] W.-S. Kim, D. Cho, and H.-M. Kim, "Color format extension," *ISO/IEC JTC1/SC29/WG11 and ITU*, 2003.

[10] C. S. Ong, A. Smola, R. Williamson, *et al.*, "Learning the kernel with hyperkernels," 2005.

[11] C. Boncelet, "Image noise models," in *The essential guide to image processing*, Elsevier, 2009, pp. 143–167.

[12] R. W. Solomon, "Free and open source software for the manipulation of digital images," *American Journal of Roentgenology*, vol. 192, no. 6, W330–W334, 2009.

[13] J. H. Hollman, K. B. Childs, M. L. McNeil, A. C. Mueller, C. M. Quilter, and J. W. Youdas, "Number of strides required for reliable measurements of pace, rhythm and variability parameters of gait during normal and dual task walking in older individuals," *Gait & posture*, vol. 32, no. 1, pp. 23–28, 2010.

[14] B. MacLean, D. M. Tomazela, N. Shulman, M. Chambers, G. L. Finney, B. Frewen, R. Kern, D. L. Tabb, D. C. Liebler, and M. J. MacCoss, "Skyline: An open source document editor for creating and analyzing targeted proteomics experiments," *Bioinformatics*, vol. 26, no. 7, pp. 966–968, 2010.

[15] B. G. Marcot, "Metrics for evaluating performance and uncertainty of bayesian network models," *Ecological modelling*, vol. 230, pp. 50–62, 2012.

[16] D. Sun, J. Wulff, E. B. Sudderth, H. Pfister, and M. J. Black, "A fully-connected layered model of foreground and background flow," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 2451–2458.

[17] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[18] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[19] D. Yu, H. Wang, P. Chen, and Z. Wei, "Mixed pooling for convolutional neural networks," in *International conference on rough sets and knowledge technology*, Springer, 2014, pp. 364–375.

[20] M. Hossin and M. N. Sulaiman, "A review on evaluation metrics for data classification evaluations," *International journal of data mining & knowledge management process*, vol. 5, no. 2, p. 1, 2015.

[21] M. Jiang, A. Beutel, P. Cui, B. Hooi, S. Yang, and C. Faloutsos, "A general suspiciousness metric for dense blocks in multimodal data," in *2015 IEEE International Conference on Data Mining*, IEEE, 2015, pp. 781–786.

[22] R. Lin, S. Liu, M. Yang, M. Li, M. Zhou, and S. Li, "Hierarchical recurrent neural network for document modeling," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015, pp. 899–907.

[23] B. Bayar and M. C. Stamm, "A deep learning approach to universal image manipulation detection using a new convolutional layer," in *Proceedings of the 4th ACM workshop on information hiding and multimedia security*, 2016, pp. 5–10.

[24] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[25] M. H. Jafari, S. Samavi, N. Karimi, S. M. R. Soroushmehr, K. Ward, and K. Najarian, "Automatic detection of melanoma using broad extraction of features from digital images," in *2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, IEEE, 2016, pp. 1357–1360.

[26] A. Radford, L. Metz, and S. Chintala, "Deep convolutional generative adversarial network," in *Under review as a conference paper at ICLR*, 2016.

[27] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[28] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in *2017 International Conference on Engineering and Technology (ICET)*, Ieee, 2017, pp. 1–6.

[29] G. Chen, W. Choi, X. Yu, T. Han, and M. Chandraker, "Learning efficient object detection models with knowledge distillation," *Advances in neural information processing systems*, vol. 30, 2017.

[30] D. Güera and E. J. Delp, "Deepfake video detection using recurrent neural networks," in *2018 15th IEEE international conference on advanced video and signal based surveillance (AVSS)*, IEEE, 2018, pp. 1–6.

[31] M. ul Hassan, "Vgg16-convolutional network for classification and detection," *en lınea].[consulta: 10 abril 2019]. Disponible en: https://neurohive. io/en/popular-networks/vgg16*, 2018.

[32] Y. Li, M.-C. Chang, and S. Lyu, "In ictu oculi: Exposing ai created fake videos by detecting eye blinking," in *2018 IEEE International Workshop on Information Forensics and Security (WIFS)*, IEEE, 2018, pp. 1–7.

[33] H. Lu, L. Wang, M. Ye, K. Yan, and Q. Jin, "Dnn-based image classification for software gui testing," in *2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, IEEE, 2018, pp. 1818–1823.

[34] S. Narkhede, "Understanding auc-roc curve," *Towards Data Science*, vol. 26, pp. 220–227, 2018.

[35] Z. Zhang, "Improved adam optimizer for deep neural networks," in *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*, IEEE, 2018, pp. 1–2.

[36] F. Zhou, X. Li, and Z. Li, "High-frequency details enhancing densenet for super-resolution," *Neurocomputing*, vol. 290, pp. 34–42, 2018.

[37] E. Kaziakhmedov, K. Kireev, G. Melnikov, M. Pautov, and A. Petiushko, "Real-world attack on mtcnn face detection system," in *2019 International Multi-Conference on Engineering, Computer and Information Sciences (SIBIRCON)*, IEEE, 2019, pp. 0422–0427.

[38] M. Mateen, J. Wen, S. Song, Z. Huang, *et al.*, "Fundus image classification using vgg-19 architecture with pca and svd," *Symmetry*, vol. 11, no. 1, p. 1, 2019.

[39] A. Rossler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, and M. Nießner, "Faceforensics++: Learning to detect manipulated facial images," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1–11.

[40] R. C. Salvador, A. A. Bandala, I. M. Javel, R. A. R. Bedruz, E. P. Dadios, and R. R. P. Vicerra, "Deeptronic: An electronic device classification model using deep convolutional neural networks," in *2018 IEEE 10th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM)*, IEEE, 2019, pp. 1–5.

[41] X. Yang, Y. Li, and S. Lyu, "Exposing deep fakes using inconsistent head poses," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2019, pp. 8261–8265.

[42] Z. Zhang and Q. Liu, "Detect video forgery by performing transfer learning on deep neural network," in *The International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery*, Springer, 2019, pp. 415–422.

[43] N. Akoury, S. Wang, J. Whiting, S. Hood, N. Peng, and M. Iyyer, "Storium: A dataset and evaluation platform for machine-in-the-loop story generation," *arXiv preprint arXiv:2010.01717*, 2020.

[44] X. Chang, J. Wu, T. Yang, and G. Feng, "Deepfake face image detection based on improved vgg convolutional neural network," in *2020 39th Chinese Control Conference (CCC)*, IEEE, 2020, pp. 7252–7256.

[45] L. Guarnera, O. Giudice, and S. Battiato, "Fighting deepfake by exposing the convolutional traces on images," *IEEE Access*, vol. 8, pp. 165 085–165 098, 2020.

[46] C.-C. Hsu, Y.-X. Zhuang, and C.-Y. Lee, "Deep fake image detection based on pairwise learning," *Applied Sciences*, vol. 10, no. 1, p. 370, 2020.

[47] W. Jiang, K. Zhang, N. Wang, and M. Yu, "Meshcut data augmentation for deep learning in computer vision," *PLoS One*, vol. 15, no. 12, e0243613, 2020.

[48] E. Johansson, "Detecting deepfakes and forged videos using deep learning," *Master's Theses in Mathematical Sciences*, 2020.

[49] P. Kumari and K. Seeja, "Periocular biometrics for non-ideal images: With off-the-shelf deep cnn & transfer learning approach," *Procedia Computer Science*, vol. 167, pp. 344–352, 2020.

[50] Y. Li, X. Yang, P. Sun, H. Qi, and S. Lyu, "Celeb-df: A large-scale challenging dataset for deepfake forensics," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 3207–3216.

[51] M. E. Mital, R. R. Tobias, H. Villaruel, J. M. Maningo, R. K. Billones, R. R. Vicerra, A. Bandala, and E. Dadios, "Transfer learning approach for the classification of conidial fungi (genus aspergillus) thru pre-trained deep learning models," in *2020 IEEE REGION 10 CONFERENCE (TENCON)*, IEEE, 2020, pp. 1069–1074.

[52] A. Neyaz, A. Kumar, S. Krishnan, J. Placker, and Q. Liu, "Security, privacy and steganographic analysis of faceapp and tiktok," *International Journal of Computer Science and Security*, vol. 14, no. 2, pp. 38–59, 2020.

[53] M. Patel, A. Gupta, S. Tanwar, and M. Obaidat, "Trans-df: A transfer learning-based end-to-end deepfake detector," in *2020 IEEE 5th International Conference on Computing Communication and Automation (ICCCA)*, IEEE, 2020, pp. 796–801.

[54] M. S. Rana and A. H. Sung, "Deepfakestack: A deep ensemble-based learning technique for deepfake detection," in *2020 7th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2020 6th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom)*, IEEE, 2020, pp. 70–75.

[55] S. Suratkar, E. Johnson, K. Variyambat, M. Panchal, and F. Kazi, "Employing transfer-learning based cnn architectures to enhance the generalizability of deepfake detection," in *2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, IEEE, 2020, pp. 1–9.

[56] J. Taylor, "Facebook: A contemporary analysis of the influential global power of facebook and how it affects our society (and recommendations on how to fix it)," 2020.

[57] J. Yang, J. Adu, H. Chen, J. Zhang, and J. Tang, "A facial expression recongnition method based on dlib, ri-lbp and resnet," in *Journal of Physics: Conference Series*, IOP Publishing, vol. 1634, 2020, p. 012 080.

[58] H. Alqahtani, M. Kavakli-Thorne, and G. Kumar, "Applications of generative adversarial networks (gans): An updated review," *Archives of Computational Methods in Engineering*, vol. 28, no. 2, pp. 525–552, 2021.

[59] N. Hasan, Y. Bao, A. Shawon, and Y. Huang, "Densenet convolutional neural networks application for predicting covid-19 using ct image," *SN Computer Science*, vol. 2, no. 5, pp. 1–11, 2021.

[60] M. Kim, S. Tariq, and S. S. Woo, "Fretal: Generalizing deepfake detection using knowledge distillation and representation learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 1001–1012.

[61] B. U. Mahmud and A. Sharmin, "Deep insights of deepfake technology: A review," *arXiv preprint arXiv:2105.00192*, 2021.

[62] M. Safaei-Farouji and A. Kadkhodaie, "Application of ensemble machine learning methods for kerogen type estimation from petrophysical well logs," *Journal of Petroleum Science and Engineering*, p. 109 455, 2021.

[63] U. Yaqub and M. Al-Mouhamed, "Simulation acceleration of index modulation using a cluster of nvidia tesla k-80,"