Department of Electrical and Computer Engineering
North South University

CSE498R
Deep Fake Detection: Real vs Fake Image Classification through Deep Neural

Md. Kamrul Hasan	**ID# 1812957642**
Salma Abdul Hai	**ID# 1731786642**
Tanjum Tanha	**ID# 1731157042**
Jaynab Sultana	**ID# 1721182042**

Faculty Advisor:

Dr Mohammad Monirujjaman Khan

Associate Professor

Electrical and Computer Engineering Department

Summer, 2021

LETTER OF TRANSMITTAL

November, 2021

To

Dr. Mohammad Rezaul Bari

Chairman,

Department of Electrical and Computer Engineering

North South University, Dhaka

Subject: **Submission of CSE498R Project Report on "Deep Fake Detection: Real VS Fake Image Classification through Deep Neural Networks"**

Dear Sir,

With due respect, we would like to submit our **CSE498R Project Report** on **"Deep Fake Detection: Real VS Fake Image Classification through Deep Neural Networks"** as a part of our BSc program. The report deals with deep fake image detection. This project was very much valuable to us as it helped us gain experience from practical fields and apply in real life. We tried to the maximum competence to meet all the dimensions required from this report.

We will be highly obliged if you kindly receive this report and provide your valuable judgment. It would be our immense pleasure if you find this report useful and informative to have an apparent perspective on the issue.

Sincerely Yours,

Md. Kamrul Hasan
....
ECE Department
North South University, Bangladesh

Salma Abdul Hai
....
ECE Department
North South University, Bangladesh

Tanjum Tanha
....
ECE Department
North South University, Bangladesh

Jaynab Sultana
....
ECE Department
North South University, Bangladesh

APPROVAL

Md. Kamrul Hasan (ID # 1812957642), Salma Abdul Hai (ID# 1731786642), Tanjum Tanha (ID # 1731157042) and Jaynab Sultana (ID # 1721182042) from Electrical and Computer Engineering Department of North South University, have worked on the Project titled "**Deep Fake Detection: Real VS Fake Image Classification through Deep Neural Networks**" under the supervision of Dr. Mohammad Monirujjaman Khan partial fulfillment of the requirement for the degree of Bachelors of Science in Engineering and has been accepted as satisfactory.

Supervisor's Signature

…………………………………….

Dr Mohammad Monirujjaman Khan

Associate Professor

Department of Electrical and Computer Engineering

North South University

Chairman's Signature

…………………………………….

Dr. Mohammad Rezaul Bari

Associate Professor

Department of Electrical and Computer Engineering

North South University

DECLARATION

This is to certify that this Project is our original work. No part of this work has been submitted elsewhere partially or fully for the award of any other degree or diploma. Any material reproduced in this project has been properly acknowledged.

<u>Students' name & Signature</u>

1. **Md. Kamrul Hasan**

   Kamrul Hasan

2. **Salma Abdul Hai**

   [illegible]

3. **Jaynab Sultana**

   Jaynab Sultana

4. **Tanjum Tanha**

   Tanjum Tanha

ACKNOWLEDGEMENTS

First and foremost, we want to thank the Almighty for providing us with the strength to carry out our duties and finish the report.

We express our deep gratitude to our honorable supervisor Dr. Mohammad Monirujjaman Khan. We acknowledge with the kind of support, inspiration and timely guideline which we received from our supervisor Dr. Mohammad Monirujjaman Khan sir. Encouragement, suggestion and very constructive feedback from our supervisor Dr. Mohammad Monirujjaman Khan sir have contributed highly to explore the idea on our project.

We are grateful to all of the lecturers who have assisted us in completing the project with our full knowledge by offering technical knowledge and ethical assistance.

We are grateful to the North South University, ECE Department for providing this course CSE498R.

We appreciate our family's support.

ABSTRACT

DeepFakes, also known as AI-generated fake images, are intended to distribute harmful content and disinformation to a large number of individuals, a task made more challenging by their inherently contentious nature and the reach of the media today. The immediate problem we have as a community is determining the authenticity of internet content, and with the emergence of the Generative Adversarial Network (GAN) and other Deepfake methods based on deep learning, we face an unprecedented threat of serious violations of fundamental people's rights, especially because deepfakes could be used to propagate false information, manipulate people, harass them, and persuade them. It also has a basic and inescapable impact on how people behave socially. The goal of this article is to utilize convolutional neural networks to classify real and false pictures from a publicly available large image dataset. We intended to compare the performance of four distinct convolutional neural networks (DenseNet121, Vgg16, Resnet50, and NasNetLarge). We have successfully achieved the state-of-the-art for deep fake image detection using densenet121. The model has a test accuracy of 99.997%, a sensitivity of 99.99%, and a specificity of 100% for predicting fake and real images. Furthermore, the model significantly reduces training loss while also improving accuracy.

Keyword: convolutional neural network; deep learning; vgg16; densenet121; resnet50; nasNetLarge.

TABLE OF CONTENTS

Chapter 1 11

Overview 11

1.1 Introduction 12

1.2 Project Description 13

1.3 Purpose of this Project 14

1.4 Project Goal 14

1.5 Motivation towards our project 15

1.6 Problem with the current system 15

1.7 Report Outline 16

Chapter 2 17

Existing Systems 17

2.1 Related Work 18

2.2 Proposed solution 23

2.3 Summary 23

Chapter 3 24

Theoretical Background 24

3.1 Methodology 25

3.2 Datasets 25

3.3 Model Overview 27

3.4 Evaluation Metrics 30

3.5 Summary 31

Chapter 4 32

Result Analysis 32

4.1 Result Analysis of three different models: 33

4.2 Comparative Analysis 48

4.3 Summary 49

Chapter 5 50

Conclusion 50

5.1 Conclusion 51

Reference 52

LIST OF FIGURES

Figure 1.0.1: Model overview 12

Figure 3.0.1: Workflow of the Best Model Selection 25

Figure 3.0.2: Block Diagram of (a) DenseNet121 (b) ResNet50 (c) VGG16 (d) NasNetLarge 27

Figure 4.0.1: Accuracy graph of training and validation dataset for DenseNet121 model 37

Figure 4.0.2: Training and Validation loss graph of DenseNet121 model 38

Figure 4.0.3: Training and validation accuracy graph of Vgg16 38

Figure 4.0.4: Training and Validation loss graph of Vgg16 model 39

Figure 4.0.5: Training and validation accuracy graph of Resnet50 Model 40

Figure 4.0.6: Training and Validation loss graph of Resnet50 Model 41

Figure 4.0.7: Training and validation accuracy graph of NASNetLarge Model 42

Figure 4.0.8: Training and Validation loss graph of NASNetLarge Model 43

Figure 4.0.9: Confusion matrix of DenseNet121 Model 45

Figure 4.0.10: Confusion matrix of Vgg16 Model 46

Figure 4.0.11: Confusion matrix of Resnet50 Model 47

Figure 4.0.12: Confusion matrix of NASNetLarge Model 48

Table List of Content

Table 1: Parameters used in preprocessing images for DenseNet121 and Resnet50 model 26

Table 2: Parameters used in preprocessing images for Vgg16 and NasNetLarge 26

Table 3: Split the dataset into training, validation and test set 33

Table 4:Parameters used for compiling various models 34

Table 5: Training loss, training accuracy, validation loss, validation accuracy values of DenseNet121, Vgg16, Resnet50 and NASNetLarge models. 35

Table 6:Performance score of DensNet121, Vgg16, Resnet50 and NASNetLarge 44

Chapter 1
Overview

1.1 Introduction

DeepFake generates images, videos, and sounds by integrating supervised machine learning techniques (CNN, GANs) with unsupervised machine learning techniques such as autoencoders. Generative Adversarial Networks (GANs) has recently been added to the basic DeepFake framework. DeepFake creation has gotten significantly simpler, needing only a few basic steps. To create realistic altered pictures, you'll need a source image and a set of intended distortions. These DeepFakes created by GAN, however, leave visible visual artifacts that may be examined using Convolutional Neural Networks (CNNs) are a particular kind of neural network that uses (CNNs).

With a new state-of-the-art accuracy along four existing CNN frameworks (DenseNet121, Vgg16, Resnet50, NasNetLarge) used in this experiment, this issue is constrained to categorical image classification in this work, with an image serving as the model input and a prediction of whether the image is real or false serving as the output, which is shown in Figure 1.0.1.

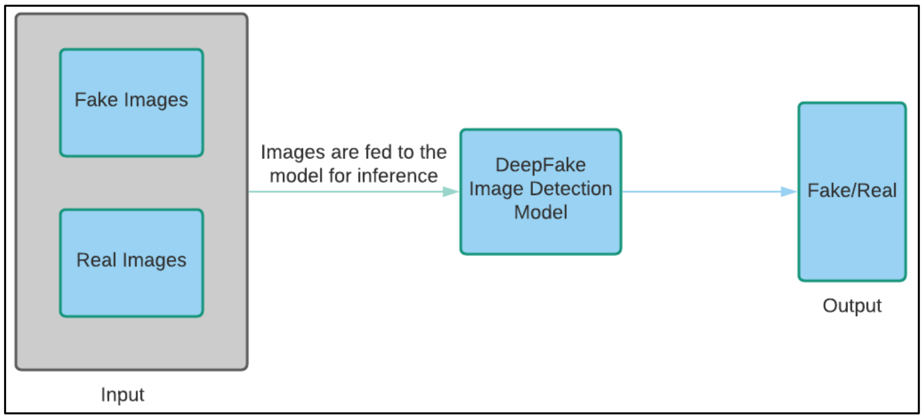

Fake Images
Real Images
Input
Images are fed to the model for inference
DeepFake Image Detection Model
Fake/Real
Output

Figure 1.0.1: Model overview

1.2 Project Description

In today's culture, deepfake is a huge problem. Artificial intelligence methods are used to construct deepfakes (encoders, decoders, generative adversarial networks, and so on). The weakness can be used to identify only poor deepfakes. However, Artificial intelligence has improved and is a powerful technology. As a result, detecting deepfakes has become increasingly difficult. By superimposing images of one person's face onto those of another, fake photos and videos are generated. The encoders and decoders are utilized to learn about and generate a deep replica of the targeted individual.

To add more, Deepfake is a well-known term in politics and business. They were created to promote division between political parties, provoke political violence in the country, and blackmail individuals. Images of celebrities have also been substituted with images of pornstars.

Additionally, Deepfake is a severe societal hazard that affects people's feelings and perceptions. People that are afflicted go through a lot of stress and emotional pain. Many of them are socially ridiculed. Celebrities, politicians, and other prominent personalities are suffering as a result of the abuse of AI-based technologies. Deepfake technology has an influence on individuals and society, and if it is not handled quickly, it can have terrible consequences. In reality, using artificial intelligence and machine learning to swap looks is a fatal exploitation. Such technology has repercussions and presents a threat to our society and its citizens. By modifying pornographic movies or confronting unknown people, such as politicians campaigning for office, who may be Deepfake victims, any well-known or well-known figure in our culture could be attacked. Fake videos or photos that circulate on social media and other online channels have the ability to sway voters in any country's state or national elections.

However, employing the suitable deepfake algorithm, such videos or photographs may be identified, reducing the influence on individuals, which is the purpose of our study.

1.3 Purpose of this Project

The goal of this research is to create a system that can determine if a photograph is real or not. The system is initially supplied an image, and image processing begins with that image. Finally, CNN will be utilized to determine whether or not the image is authentic.

1.4 Project Goal

The Goal of our project:

1. Identifying deepfake with high accuracy.
2. Detecting deepfake effectively.
3. Distinguish between real and fake images.

1.5 Motivation towards our project

The concept of deepfakes refers to images, audio, videos that are fakes, that is, they depict events that never occurred but unlike manipulating media in the past like photoshop, these deepfakes are created by deep neural networks to be really indistinguishable from their real counterparts. The advances in the field of deepfakes are both impressive and alarming because in the wrong hands, this technology could be used to spread misinformation and undermine public trust. Many researchers, corporations, and sectors are attempting to develop a method for detecting fake photographs.

This means that, as we get better at generating deepfakes, we must also get better at identifying them. As a result, we make an effort to distinguish between actual and fraudulent images through this research.

1.6 Problem with the current system

Deepfake may be created in a variety of ways. The solution is not excessive in comparison to this deepfake. There aren't many options for efficiently detecting fake images. Many powerful artificial models are utilized to create visuals that seem like genuine images. However, the technique for detecting deep fakes is not always able to distinguish between real and fake images.   To create a deepfake that is dangerous to society's safety, Generative Adversarial Networks are employed. Deepfakes have the potential to deceive an audience into believing something that never happened. For example, a deepfake has been used to make a company's CEO appear to be a cruel, malevolent character. This video was created much over a year ago, and you'd have a hard time telling if it was faked now. Deepfakes, according to several analysts, will be used to affect the election right before it takes place. For example, a deepfake might make it look as though a candidate said or did something negative when it was actually a ploy to discredit that candidate.

1.7 Report Outline

The rest of the paper is arranged as follows-

1. The related work in this project is reviewed and unified into section 2.
2. Theoretical background for this project such as architecture and dataset is described in section 3. This section describes the study method and includes details on the dataset, architecture, data augmentation stages, and the use of a CNN model for deep fake detection.
3. Results and analysis are summarized in section 4.
4. Finally, in section 5, the conclusion is given.

Chapter 2

Existing Systems

2.1 Related Work

Recent research papers and journals were examined to provide a more modern and effective solution to the classification challenge at hand while maximizing the deep learning model's accuracy. The following are a handful of them, together with the datasets they used, their methods, and their results:

Different state-of-the-art generative adversarial networks (GANs) are used by Hsu et al. [1] to produce fake images and real images. Next, the DenseNeT was developed as a two-streamed network designed to allow pairwise input for Common Fake Feature (CFF) learning, which will distinguish the features of real and fake images. The dataset used in this experiment was collected by CelebA. CelebA contains large pose variations in images. Xinyi et al. [2] used celebrity images, 200 manually selected images (100 fake images + 100 real images), and the Chicago Face Dataset. In the case of Nirkin's method, 98% accuracy, 96% true positive, and 0% false-negative were obtained using the ResNet-18 model on manually selected 200 images. Using the Auto Encoder-GAN (AE-GAN) method, 99.88% accuracy, 99.86% true positive, and 0.08% false-positive were obtained. The ResNet-18 model is used on the entire dataset. Khodabakhsh et al. [3] evaluate different methods of fake face detection based on both deep learning methods and conventional texture descriptor-based methods. A set of videos were collected from YouTube. They have collected a new database containing more than 53,000 images from 150 videos that have fake images produced through Computer Graphics Image (CGI) and Generative Adversarial networks (GANs). CGI is used in this task because of the immense availability and the ease of creating high-quality fake face images that include variable sizes in images. Ekraam et al. [4] used different kinds of approaches to detect illegal faces in video streams generated by Deepfakes, Face2Face, and FaceSwap. They used the FaceForensics++ dataset containing 1000 video sequences which involve face tampering methods. In the experiment, they pursued ResNet and DenseNet. For manipulation detection, the performance of DenseNeT with the bidirectional recurrent network is best. The DenseNet accuracy with alignment and bidirectional recurrent network for Deepfakes is 96.9%, Face2Face is 94.35%, and FaceSwap is 96.3%. The ResNet50 accuracy for Deepfakes was 94.9%, Face2Face 94.05% and FaceSwap 95.4%. Ruben et al. [5] surveyed the techniques for making illegal face images, consisting of intense fraud and manipulation detection processes. In this work, some special facial manipulations, such as entire face adjustment, identity exchange, attribute change, and manifestation swap are reviewed. They provided detailed information regarding manipulation techniques for each manipulation group, open-source databases, and original benchmarks to judge the forgery detection techniques, and showed the brief results of these evaluations. They also highlighted the growth and the challenges of the latest generation of Deep Fake. Xin et al. [6] present a new method for disclosing Deep Fakes. Basically, their method is focused on the observation that deep fakes are created by dividing the synthesized facial area in the original image. Then, errors can be brought to light by calculating the 3D head pose from face images. In this work, they used a set of real face images and Deep Fakes to evaluate an SVM classifier. Ricard et al. [7] have achieved 100% accuracy on high-resolution images of the Faces-HQ and medium-resolution images of the CelebA datasets. In the case of low-resolution videos of the FaceForensics++ dataset, they have obtained 91% accuracy. They use a support vector machine (SVM) classifier and logical regression. Li et al. [8] propose to detect facial falsification by scattering the inconsistent parts of the images across the mixing border with the help of noise synthesis. The latest study gives a hint that images created by convolutional neural networks (CNN) are not difficult to mark [9]. The outcomes show that such images share the usual systemic errors, which can be perceived by Dedicated Machine Learning models with the help of data enhancement. Pixel-level facial falsification identification can also be possible. Nhu et al. [10] proposed a deep CNN model for identifying real and counterfeit images generated by GANs in their study. The authors used the CelebA face dataset with 292,599 images. They used the VGG16 architecture with pre-train weights from VGG-Face. By using this, they got 80% accuracy. Besides, they also did some experiments with the ResNet architecture and pre-train weights from ImageNet. By using ImageNet VGG16, the authors got 76% accuracy and by using VGG-Face ResNet50 they got 73% accuracy. Afchar et al. [11] proposed the MesoNet, which examines the mesoscopic features of face images using three shallow (few layers) networks. They used to provide an approach for detecting face tampering in videos that is both automatic and efficient, with an emphasis on two contemporary techniques for creating hyper-realistic constructed videos: Face2Face and Deepfake. They used picture forensics techniques that have been around for a long time. They were able to detect Deepfake and Face2Face with a success rate of over 98% for Deepfake and 95% for Face2Face.. Zi et al. [12] proposed a dataset called WildDeepfake, which contains 7,314 sequences derived from 707 deepfake films obtained from the internet. There are more scenarios, people in each circumstance, and facial expressions in this collection than in the previous one. As a result, WildDeepfake has a superior visual effect, making the scene more realistic. They used to propose Attention-based Deep-fake Detection Networks. By applying this ADDNet-2d on WildDeepfake, the detection accuracy was only 76.25% and ADDNet-3d on WildDeepfake, the detection accuracy was only 65.50%. To detect Deepfake in video frames, Koopman et al. [13] used a photo response non-uniformity analysis. In this PRNU study, a sequence of frames is constructed from the input movies and stored in sequentially selected folders. It crops each video frame with exactly identical pixel values to keep the PRNU design piece and make it consistent. These settings are then divided into 8 groups of the same size, and a typical PRNU pattern is constructed for each using the 2nd order FSTV approach as well as the normalized cross-correlation scores are calculated to compare them to one another. It calculates the average correlation score and estimates variances in correlation scores for each video. Ultimately, it does a t-test on the outcomes of Deepfakes and genuine videos to check whether there are any statistically significant discrepancies between the two sets of results. Cozzolino et al. [14] presented ForensicTransfer (FT), an autoencoder-based architecture that distinguishes genuine photos from fakes. The FT conducts a number of studies and reaches an accuracy rate of up to 80% to 85%. Image processing techniques such as Gaussian blur and Gaussian noise were employed by Xuan et al. [15] This improves the statistical similarity between actual and false photos at the pixel level and forces the forensic classifier to learn more intrinsic and relevant characteristics, allowing it to generalize better than earlier image forensics approaches or image steganalysis networks. Guera et al. [16] suggested a technique that consists of 2 key components: (i) LSTM and (ii) CNN. By concatenating the features of several successive frames and transferring them to the LSTM for analysis, the CNN creates a collection of features for each frame in a given image sequence. The suggested network was trained on 600 films and attained an accuracy rate of 97.1%. Rossler et al. [17] showed the largest non-Deepfake swapped face database to date (5000000 photos from over 1000 videos). The authors also compared and contrasted current forgery classification and segmentation techniques. Rana et al. [18] recommend DeepfakeStack, a deep ensemble learning system, for detecting falsified videos. To create a better composite classifier, their suggested method integrates a number of state-of-the-art DL-based classification models. Using this strategy, they were able to detect Deepfake with an accuracy of 99.65% and an AUROC of 1.0. A method for obtaining deepfake fingerprints from photographs was proposed by Guarnera et al. [19]. The expectation-maximization algorithm is also used to extract fingerprints, emphasizing the ability to retrieve convolution traces concealed in the generative process. The fakeness detection algorithm in this paper was developed using a random forest classifier. Several CNN-based deepfake detection approaches were disclosed by the authors, all of which need a lot of processing resources. Using solely CPU resources, the proposed technique produced good categorization results. He et al. [20] presented the AttGAN framework, which applies an attribute classification constraint on the generated picture in the latent representation to ensure that only the proper features are altered.

This paper presents four different deep learning models, i.e., DenseNet121, Vgg16, Resnet50, and NASNetLarge implementations, and their performances on the training, validation, and test datasets for deep fake detection. Various approaches and hyperparameters have been employed in the implementation of those models in this research so that those models can reliably distinguish real and false images. The best trained model was then selected after a comparative study of those models, with the goal of using this model to categorize real and fake images in numerous places.

2.2 Proposed solution

We are concentrating on deep learning-based deep fake detection because we've previously discussed a number of articles and have gotten ideas from them. Our major goal is to detect deep fakes. We have collected the datasets, divided them into three parts (60 percent for training, 20% for testing, and 20% for evaluation), and then trained, tested, and evaluated the dataset. By performance metrics like accuracy, Confusion Matrix, Precision and Recall, F1-score, AU ROC, we comprehend the efficiency of the model we employed in our project. The system will function as follows: a user will submit an image, which will be detected by our detection system, which will then display the categorical outcome of whether the image is real or fake. Python is the programming language that we have employed.

2.3 Summary

We endeavored to present an overview of the entire project in this chapter, including how things are today, how we will discover problems, and what solutions we aim to apply from this perspective.

Chapter 3

Theoretical Background

3.1 Methodology

This section presents particular procedures that are followed and continued appropriately in order to perform experiments on the project for deep fake detection. This project's system diagram, Figure 3.0.1. depicts how the workflow will unfold. This section will briefly describe each of the blocks below, as well as their relevance.

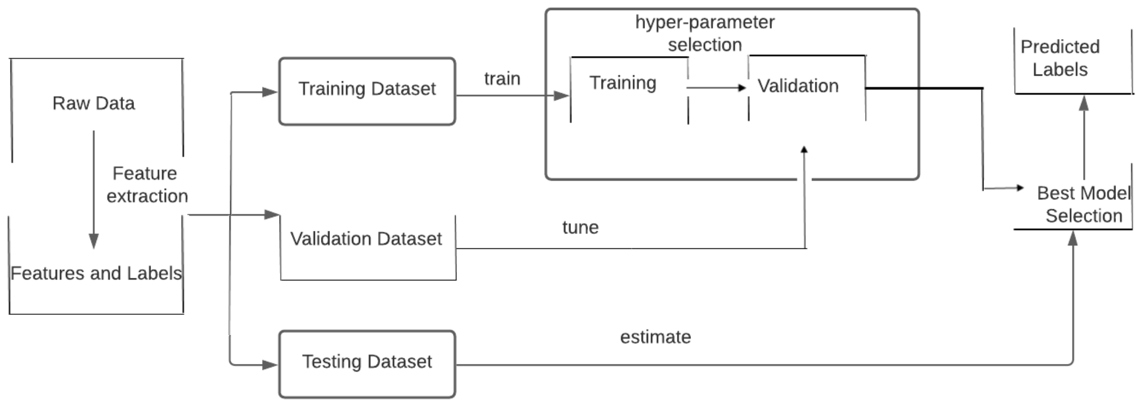


Figure 3.0.1*: Workflow of the Best Model Selection*

3.2 Datasets

Two different datasets have been used from Kaggle's two different links. For 70000 real images, dataset [21] and for 80000 fake images, dataset [22] have been used. After collecting real images and fake images, they are combined into a folder which contains 1,50,000 images (real + fake). Then, that folder is split into 60% for training, 20 % for validation and 20 % for testing using Python's split folder package. In the dataset, images are not the same in size and shape. Images contain some dark regions or noises. These types of variations can cause errors and the system prediction will be false. We have preprocessed our dataset to avoid these noises. ImageDataGenerator class with augmentations such as rescale, rotation_range, fill_mode, horizontal_flip have been used for image preprocessing shown in Tab. 1 and Tab. 2.

Table 1: *Parameters used in preprocessing images for DenseNet121 and Resnet50 model*

Parameter	**Value**
rotation range	15
target size	(224,224)
batch size	64
class mode	‘Categorical’
horizontal flip	True
fill mode	‘nearest’
rescale	1./255

Table 2: *Parameters used in preprocessing images for Vgg16 and NasNetLarge*

Parameter	**Value**
rotation range	15
target size	(224,224)
batch size	64
class mode	‘Categorical’
horizontal flip	True
fill mode	‘nearest’
rescale	1./255

3.3 Model Overview

The descriptions of the models used in this research works, as well as their block diagrams are described below:

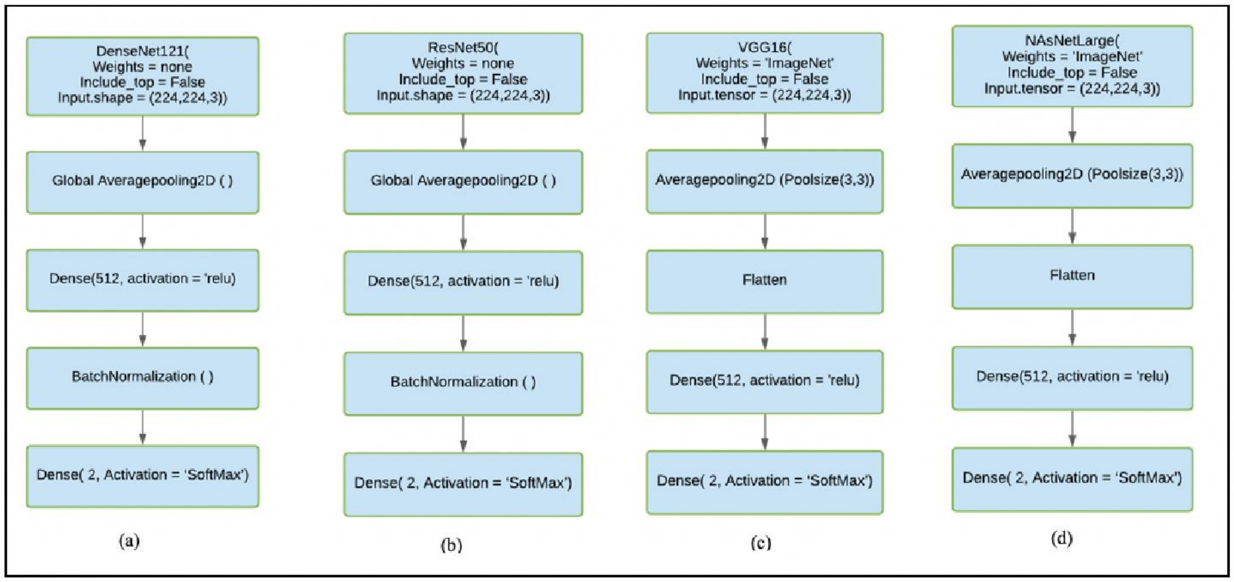

DenseNet121(Weights = none Include_top = False Input.shape = (224,224,3))
Global Averagepooling2D ( )
Dense(512, activation = 'relu)
BatchNormalization ( )
Dense(2, Activation = 'SoftMax')
(a)
ResNet50(Weights = none Include_top = False Input.shape = (224,224,3))
Global Averagepooling2D ( )
Dense(512, activation = 'relu)
BatchNormalization ( )
Dense(2, Activation = 'SoftMax')
(b)
VGG16(Weights = 'ImageNet' Include_top = False Input.tensor = (224,224,3))
Averagepooling2D (Poolsize(3,3))
Flatten
Dense(512, activation = 'relu)
Dense(2, Activation = 'SoftMax')
(c)
NAsNetLarge(Weights = 'ImageNet' Include_top = False Input.tensor = (224,224,3))
Averagepooling2D (Poolsize(3,3))
Flatten
Dense(512, activation = 'relu)
Dense(2, Activation = 'SoftMax')
(d)

Figure 3.0.2: Block Diagram of (a) DenseNet121 (b) ResNet50 (c) VGG16 (d) NasNetLarge

Densenet121: The more efficient convolutional neural network shown in Figure 3.0.2(a). which makes deep analysis and gives output in a very straightforward way is densenet121. In densenet121, without the first layer, each layer is connected with the previous layers. One layer produces a result which is input for the next layer. There are direct connections to each layer. Figure 1 represents that, DenseNet starts with a weights=”None”, including top =False, Input_shape=(224,224,3), these initial values and GlobalAveragePooling2D layers. After that, there's a dense block, which is followed by a BatchNormalization () layer. Dense block has 512 hidden layers and a rectified linear activation function or “relu” which is a linear function that gives output directly if it gets positive input. Otherwise, it will give zero. It is used as default and achieves best performance. To overcome the vanishing gradient problem, to allow the model to learn faster, and obtain high accuracy, “relu” function is used. The last layer uses the logistic function sigmoid activation function. It takes any real value as input and gives output in the range 0 – 1.

ResNet50: Figure 3.0.2(b) shown above represents that there are too many convolutional layers attached to one another and there are too many skipped connections in ResNeT50. There are 5 stages in the ResNet-50 model. Each stage consists of a convolution and an identity block. Both blocks contain convolution seams. The Resnet-50 model has more than 23 million trainable parameters. Figure 4 demonstrates the block diagram and the layers and parameters used in RESNeT50. In Resnet50, the input shape is 224 (height), 224 (width), 3 (color channel). Then, GlobalAveragePooling2D layers are used. Next, the dense block and BatchNormalization () layers are used one after another. Finally, the output dense layer is used. It is also known as the Residual Network. Compared to VGG, RESNeT50 is very deep. It is a 50-layer deep model which can learn rich feature representation for a wide range of images. Here, the connection between layers is called residual because the output of one layer is a convolution of its input + input.

VGG16: Vgg16 is a convolution Neural Network (CNN). It is regarded as one of the best vision model architecture ever created. VGG16 focuses on having convolution layers of 3x3 filters with a stride of 1 and always using the same padding, as well as a MaxPooling layer of 2x2 filters with a stride of 2 instead of a large number of hyperparameters. The convolution and max pool layers are placed in the same way throughout the architecture. For output, it has two fully connected layers (FC) followed by a SoftMax. The number 16 in VGG16 refers to the fact that it has 16 layers of varying weights. With roughly 138 million parameters, this is a massive network [23]. However, in this project, some modifications to the usual vgg16 model were made. Two fully connected layers have been removed and replaced with new layers. In the given vgg16 architecture, after the $7 \times 7 \times 512$ convolution layer, there is a $2 \times 2 \times 512$ convolution layer, a 1

$\times 1 \times 2048$ flatten layer and one fully connected layer with a $1 \times 1 \times 512$ dense layer and finally the output layer of $1 \times 1 \times 2$ shown in Figure 3.0.2(c).

NASNetLarge: The NASNet-Large network obtains results with smaller model size and lower complexity. There are two convolutional cells in NASNet-Large. Convolutional cells give a feature map of the same dimension as a normal cell. If the feature map where height and width are reduced by a factor of two, then it is called a reduction cell. Figure 3.0.2(d) demonstrates that, NASNetLarge starts with a weight="None", include_top= False, Input_shape= (224,224,3), these initial values and the GlobalAveragePooling2D layers. Then, there is a dense block followed by a Flatten () layer. Dense block has 512 hidden layers and a rectified linear activation function or "relu", which is a linear function that gives output directly if it gets positive input. Otherwise, it will give zero. It is used as default and achieves the best performance. To overcome the vanishing gradient problem, to allow models to learn faster, and obtain high accuracy, the "relu" function is used. The last layer used the logistic function sigmoid activation function. It takes any real value as an input and gives an output in the range of 0 – 1.

3.4 Evaluation Metrics

For evaluating the performance of various models, the following performance matrices have been used in this research work:

Precision: To address what proportion of positive identifications are actually correct, we have used precision. It indicates how many of the instances that were accurately predicted turned out to be positive. This is determined by:

$$Precision = \frac{True\ Positive}{(True\ Positive\ + False\ Positive)} \tag{1}$$

Recall: To address what proportion of actual positives are identified correctly, we have kept the recall metric. It indicates how many of the actual positive cases our model was able to properly anticipate. This is calculated by:

$$Recall = \frac{True\ Positive}{(True\ Positive + False\ Negative)} \tag{2}$$

F1 Score: The F1 score is a typical metric for classification problems and is appropriate when we care equally about precision and recall. The F-score, often known as the F1-score, is a metric for determining how accurate a model is on a given dataset [24]. It's used to evaluate binary classification methods that sort instances into positive or negative categories. True predictions are always excellent, whether they are positive or negative. These are the outcomes we want our model to provide. False predictions, on the other hand, are not desirable. We aim to keep these instances to a minimum. The F-score is a strategy for measuring the precision and recall of a model. This is calculated by:

$$F1\ Score = \frac{(\ 2 \times Precision \times Recall)}{(Precision + Recall)} \tag{3}$$

Confusion matrix: The confusion matrix is the most useful figure for understanding the specifics of the model's performance. Confusion matrices are used to provide important predictive metrics such as recall, specificity, accuracy, and precision. Confusion matrices are useful because they make it simple to compare values such as True Positives, False Positives, True Negatives, and False Negatives. This was critical for our research since we wanted to ensure both precision and recall. Our aim was to detect fraudulent images with little or no misclassification, which we were finally able to do using the DenseNet121 model. We were able to determine the overall model performance in terms of specificity (the ratio of correctly negative identified subjects by test to all negative subjects in reality) and sensitivity/recall as a result of this.

3.5 Summary

In this chapter, we have covered how the key sections of the system function, as well as the evaluation metrics that would be utilized for this project.

Chapter 4
Result Analysis

4.1 Result Analysis of three different models:

This section discusses the results achieved for deep fake detection using various pretrained models, such as DenseNet121, Vgg16, Resnet50, and NASNetLarge. Before that, the real images and fake images were collected and then modified from kaggle’s two different dataset links and merged into one dataset. Then, Python’s split-folders package was used to split the whole dataset of 150000 images into train (60%), validation (20%) and test (20%) datasets, which have been shown in Tab. 3. Then, the train dataset is preprocessed by the Keras ImageDataGenerator where the rotation range is "15", fill_mode is "nearest" and horizontal_flip keeps "true" and the validation and test dataset are just rescaled.

Table 3: *Split the dataset into training, validation and test set*

Dataset	**Number of images**
Training	90000
Validation	30000
Testing	30000

After that, four different models (DenseNet121, Vgg16, Resnet50, and NASNetLarge) have been compiled to compare the models’ performance. While compiling Densenet121, the initial learning rate is kept at 1e-3 [25], batch size is 64, max epoch is 5, optimizer is "Adam", decay is 1e-3/Max_epoch and categorical cross entropy is used as the loss function, which has been shown in Tab. 4. But for compiling other models, all the parameters have been kept the same except the epoch, and the epoch size here is 128. All the models have been run on Colab where the execution environment keeps the GPU.

Table 4:*Parameters used for compiling various models*

Parameters	**Value**
Initial Learning Rate	1e-3
Decay	(1e-3)/Max_epoch
Batch Size	64/128
Shuffling	Each epoch
Optimizer	Adam
Loss	Categorical cross entropy
Max_epochs	5
Execution Environment	Gpu

Tab. 5. shown below, demonstrates that, while the DenseNet121 model has been fitted to the training and validation datasets after each epoch, the model improves learning through parameters. Initially, this model had a training accuracy of 99.74%, a training loss of 8.04%, a validation accuracy of 72.15% and a validation loss of 106.49%. But after completing 5 epochs, the model improved a lot by achieving a training accuracy of 99.896%, training loss of 2.64%, validation accuracy of 99.54% and validation loss of 1.28%. While the Vgg16 model has been fit on the training and validation datasets, at epoch 1, this model gains training loss of 20.88%, training accuracy of 92.41%, validation loss of 25.87% and validation accuracy of 89.41%. As the number of epochs increases, the model improves its overall accuracy and loss. At epoch 4, the model achieves its best result by occupying a training loss of 19.05%, training accuracy of 92.18%, validation loss of 24.71% and validation accuracy of 90.02%. At epoch 5, this model overfits by getting a high validation loss. Later, the Resnet50 model has been trained on the training and validation datasets and after 5 epochs, this model achieves a training accuracy of

99.43%, a loss of 1.58% and at epoch 3, this model secures the highest validation accuracy of 96.79%. But at epoch 4 and 5, although the Resnet50 model achieves better training accuracy, the validation accuracy has fallen. Therefore, after 5 epochs, the model stops training due to the early stopping method to overcome the model's overfitting. Finally, the NASNetLarge model has been implemented and fits on the training and validation dataset. This model improves its training and validation accuracy very slowly over the whole 13 epochs. At epoch 1, it has 74.89% of training accuracy, 52.94% of training loss, 79.91% of validation accuracy and 43.74% of validation loss. But it achieves the highest validation accuracy at epoch 10 and the highest training accuracy at epoch 13. Due to the early stopping method, the model has stopped its training to overcome overfitting.

Table 5: *Training loss, training accuracy, validation loss, validation accuracy values of DenseNet121, Vgg16, Resnet50 and NASNetLarge models.*

		DenseNet121		
Epoch	**Training Loss**	**Training Accuracy**	**Validation Loss**	**Validation Accuracy**
1	0.008040	0.997422	1.064918	0.721533
2	0.006386	0.997822	2.161880	0.588267
3	0.004343	0.998633	0.000557	0.999867
4	0.003744	0.998744	0.036283	0.985700
5	0.002640	0.998956	0.012771	0.995367
		Vgg16		
1	0.208764	0.924056	0.258749	0.894133
2	0.199943	0.918433	0.324680	0.871567
3	0.196135	0.920478	0.285699	0.885867
4	0.190509	0.921833	0.247144	0.900200

5	0.183452	0.926544	0.290437	0.887100
		Resnet50		
1	0.342181	0.849633	0.466699	0.824267
2	0.211301	0.914667	0.320690	0.860033
3	0.083806	0.968244	0.085208	0.967867
4	0.030430	0.988633	0.313809	0.874333
5	0.015807	0.994333	0.322948	0.854267
		NASNetLarge		
1	0.52937	0.74887	0.43746	0.79910
2	0.42381	0.80451	0.40333	0.81723
3	0.38882	0.82306	0.39089	0.82650
4	0.36207	0.83684	0.36319	0.84020
5	0.33953	0.84891	0.34019	0.85143
6	0.32456	0.85843	0.35010	0.84870
7	0.31172	0.86551	0.36260	0.84370
8	0.31751	0.86209	0.33823	0.85477
9	0.30122	0.86993	0.35563	0.85040
10	0.28664	0.87631	0.33516	0.85577
11	0.27613	0.88268	0.34582	0.85373
12	0.26425	0.88741	0.35359	0.85560
13	0.25381	0.88941	0.35452	0.85432

To visualize the DenseNet121 model’s training and validation accuracy and loss, Figure 4.0.1. and Figure 4.0.2. have been used. In Figure 4.0.1 The x-axis represents the number of epochs and the y-axis represents the training and validation accuracy. As the number of epochs is increasing, the training accuracy is also increasing gradually, but validation accuracy at first decreases and then starts increasing. Although there is a fluctuation in the graph of validation accuracy, ultimately the validation accuracy reaches up to 99.54% along with the training accuracy of 99.90% after 5 epochs.

```
Text(0.5, 1.0, 'Training and Validation Accuracy')
```

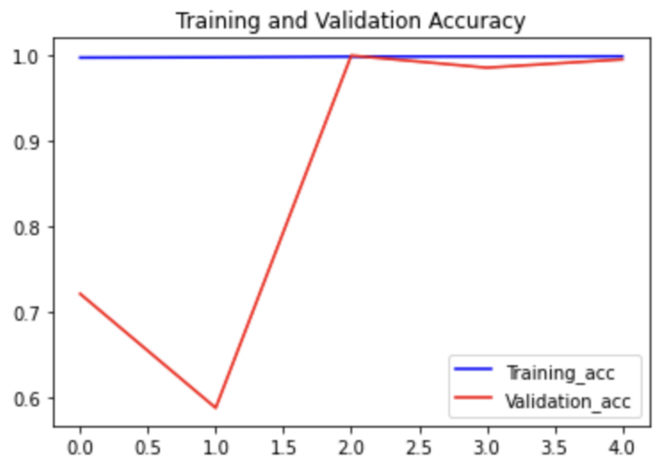


Figure 4.0.1: Accuracy graph of training and validation dataset for DenseNet121 model

Similarly, in Figure 4.0.2. The x-axis represents the number of epochs and the y-axis represents training and validation loss. As the number of epochs increases, training and validation are decreasing. But in the meantime, there is a fluctuation in the validation loss just like the training and validation accuracy. Initially, the loss increases and after epoch 1, it starts decreasing and reaches 1.28% after completing 5 epochs.

```
Text(0.5, 1.0, 'Training and Validation loss')
```

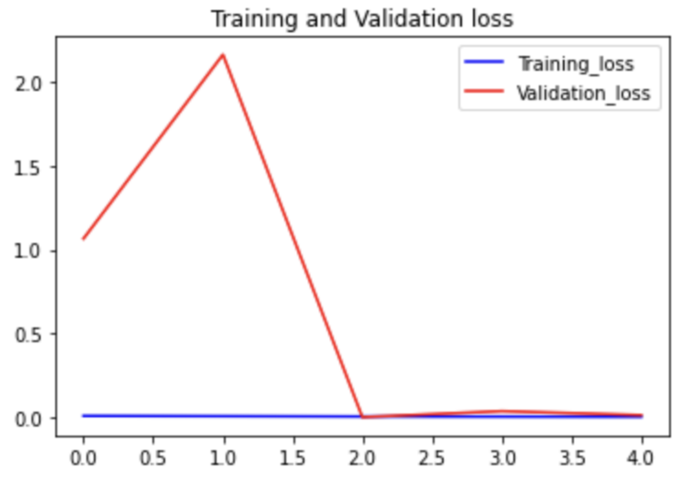
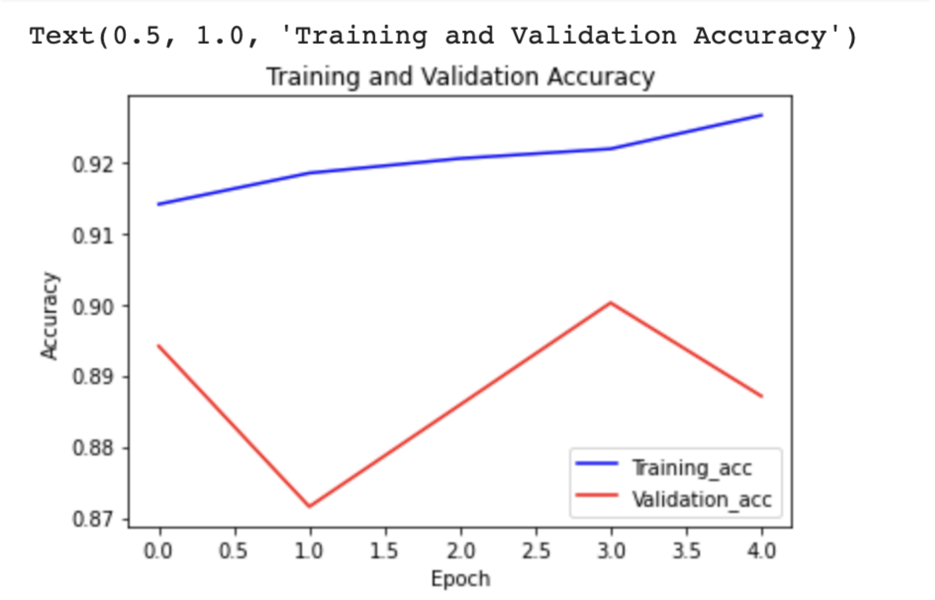


Figure 4.0.2: Training and Validation loss graph of DenseNet121 model

Figure 4.0.3 and Figure 4.0.4 represent the training and validation accuracy and loss graph of the Vgg16 model. The graph in Figure 4.0.3. Shows that as the number of epochs increases, the training accuracy also increases, but there is a fluctuation in validation accuracy.



```
Text(0.5, 1.0, 'Training and Validation Accuracy')
```

Figure 4.0.3: Training and validation accuracy graph of Vgg16

The same thing has been represented in Figure 4.0.4. As training accuracy in Figure 4.0.3. increases, the training loss at the same time decreases in Figure 4.0.4. But due to a fluctuation in validation accuracy, there has also been a fluctuation in validation loss.

```
Text(0.5, 1.0, 'Training and Validation loss')
```

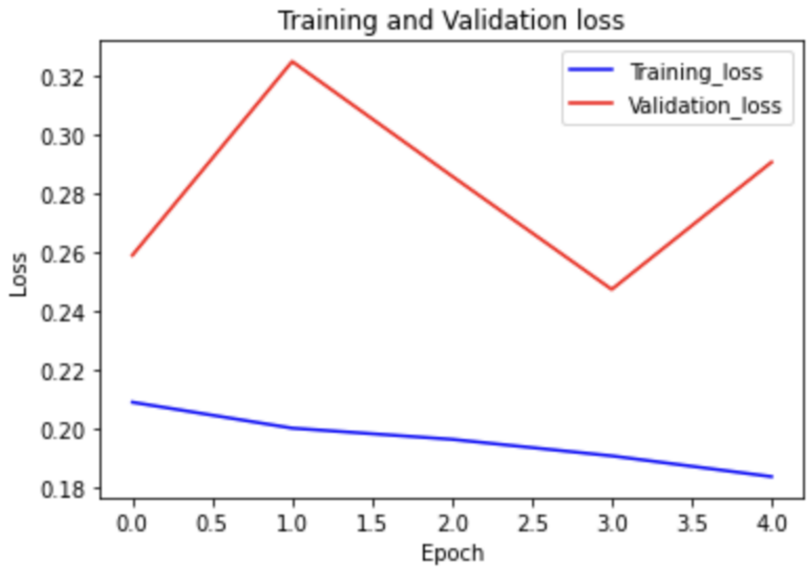


Figure 4.0.4: Training and Validation loss graph of Vgg16 model

Figure 4.0.5. shown below represents the training and validation accuracy graph of the Resnet50 model. From epoch 1 to 3, the Resnet50 model achieves better training and validation accuracy. But after epoch 3, although the training accuracy graph goes up, the validation accuracy graph falls down, and for that reason, the early stopping method stops the model from overcoming the overfitting.

```
Text(0.5, 1.0, 'Training and Validation Accuracy')
```

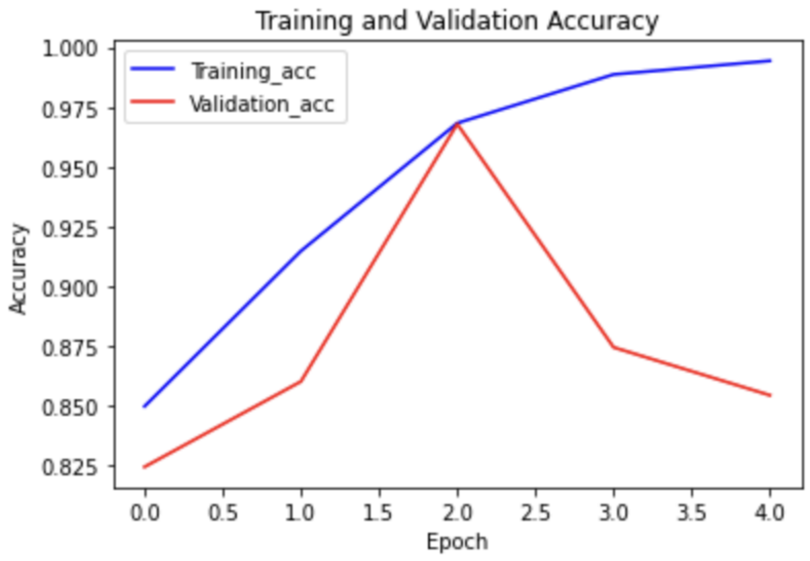


Figure 4.0.5: Training and validation accuracy graph of Resnet50 Model

Figure 4.0.6. shown below, illustrates the training and validation loss graph of the Resnet50 model. As in Figure 4.0.5. the training accuracy and validation accuracy graphs rise, the training and validation loss graphs fall from epoch 1 to 3. When the validation accuracy graph starts falling, the validation loss graph stats rise. Due to that, there has been a chance of model overfitting and, after epoch 5, the model has been stopped to overcome that situation.

```
Text(0.5, 1.0, 'Training and Validation loss')
```

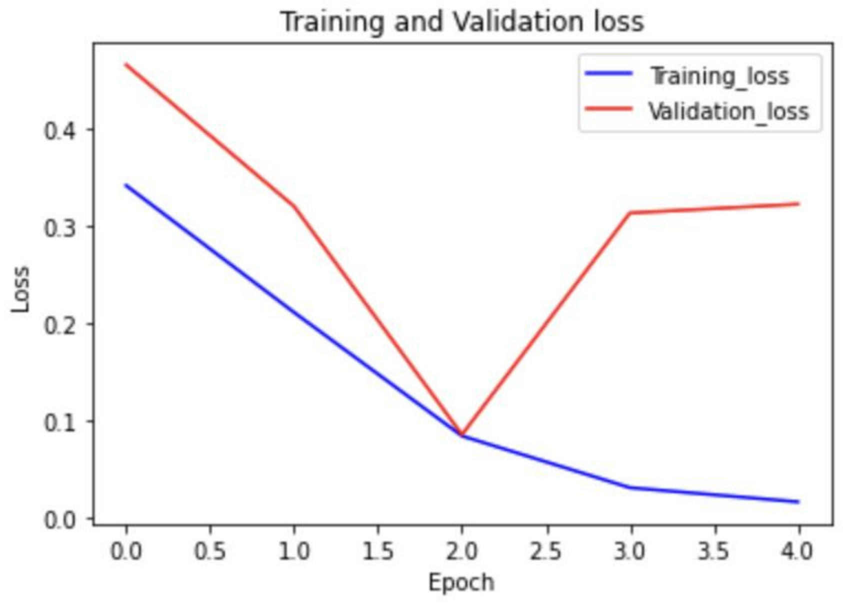


Figure 4.0.6: Training and Validation loss graph of Resnet50 Model

Figure 4.0.7. shown below, illustrates the training and validation accuracy graph of the NASNetLarge model. As the number of epochs increases from 1 to 13, this model slowly and gradually improves training and validation accuracy and reaches the highest 88.94% of training accuracy and 85.57% of validation accuracy.

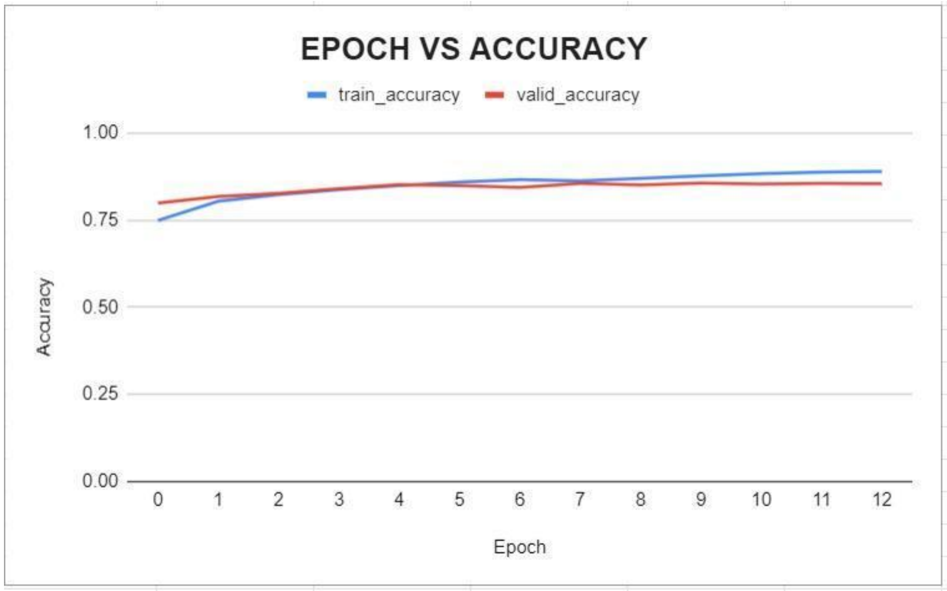


Figure 4.0.7: Training and validation accuracy graph of NASNetLarge Model

And Figure 4.0.8. shown below elaborates the training and validation loss graph of NASNetLarge model. As the number of epochs increases from 1 to 13, due to the model's improvement in the training and validation accuracy, the loss in training and validation graph falls down and reaches up to 25.38% and 33.51% in the training and validation set.

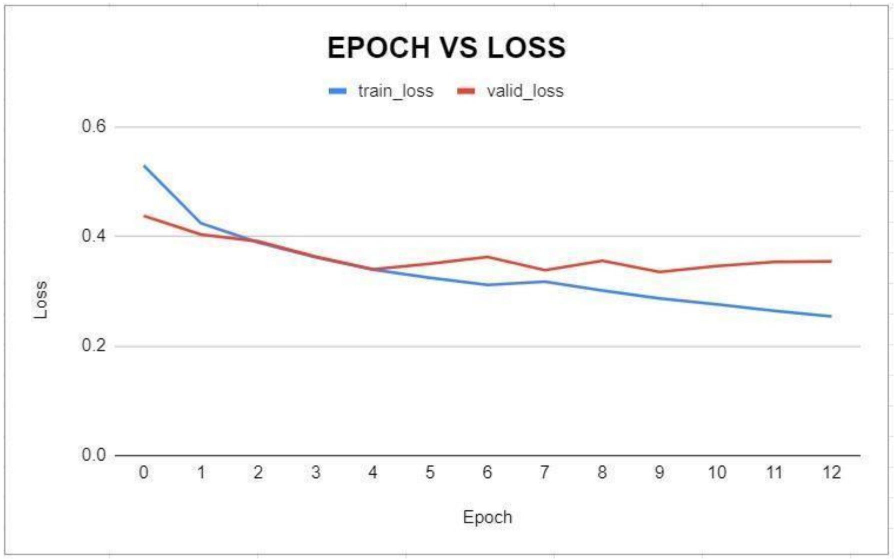


Figure 4.0.8: Training and Validation loss graph of NASNetLarge Model

As there has been used ModelCheckPoint to save the best trained model. So, after completing 5 epochs, the best trained DenseNet121 model has been evaluated on the test dataset and Tab. 6 shows that the model achieves 99.997% of testing accuracy and 0.46% of testing loss. It can also be seen that the DenseNet121 model has 99.99% sensitivity and 100% specificity. This model further achieves 100%, 100% and 100% precision, recall and f1 score respectively in detecting both fake and real images. Tab. 3 also shows that when the best trained Vgg16 model has been evaluated on the test dataset after 5 epochs, it secures 90.03% of testing accuracy and 24.87% of testing loss. After 5 epochs, the vgg16 model does not improve its accuracy. Therefore, there has been an early stopping method to stop further training. It can further be seen that the Vgg16 model has 93.71% sensitivity and 85.81% specificity. But this model has 88%, 94% and 91% precision, recall and f1 score respectively on detecting fake images and 92%, 86% and 89% precision, recall and f1 score respectively on detecting real images. The resnet50 model performs pretty well in detecting deep fakes and its accuracy and loss on the test dataset is 97.03% and 7.94%. It also achieves a sensitivity of 99.44% and a specificity of 94.27% after 5 epochs. Due to higher chances of overfitting, the Resnet50 model has stopped training after 5 epochs and its precision, recall, and f1 score on fake images are 95%, 99%, and 97% and its precision, recall, and f1 score on real images are 99%, 94%, and 97% respectively. If the performance scores of the NASNetLarge model are observed, its test accuracy is 85.42%, loss is 34.00%, sensitivity is 91.58% and specificity is 78.37%. This model also secures 83%, 92% and 87% of precision, recall and f1 score in the detection of fake images and 89%, 78% and 83% of precision, recall and f1 score in detecting real images respectively. If these four models have been compared with each other, it shows that the DenseNet121model secures the highest accuracy compared to other models.

Table 6:*Performance score of DensNet121, Vgg16, Resnet50 and NASNetLarge*

Model	**Dataset**	**Accuracy**	**Loss**	**Sensitivity**	**Specificity**	**Images**	**Precision**	**Recall**	**F1 Score**
DenseNet121	Test	0.99997	0.00046	0.9999	1.0000	Fake	1.00	1.00	1.00
						Real	1.00	1.00	1.00
Vgg16	Test	0.90026	0.24875	0.9371	0.8581	Fake	0.88	0.94	0.91
						Real	0.92	0.86	0.89
Resnet50	Test	0.97027	0.07938	0.9944	0.9427	Fake	0.95	0.99	0.97
						Real	0.99	0.94	0.97
NASNetLarge	Test	0.85416	0.34000	0.9158	0.7837	Fake	0.83	0.92	0.87
						Real	0.89	0.78	0.83

Figure 4.0.9 shown below, highlights true positive, true negative, false positive and false negative cases in detecting real and fake images. From the figures, it can be said that the DenseNet121 model predicts 15999 fake images truly as' fake ’out of 16000 fake images and predicts 1 fake image falsely as' real’. The figure also represents that this model predicts 14000 real images truly as real’ out of 14000 real images.

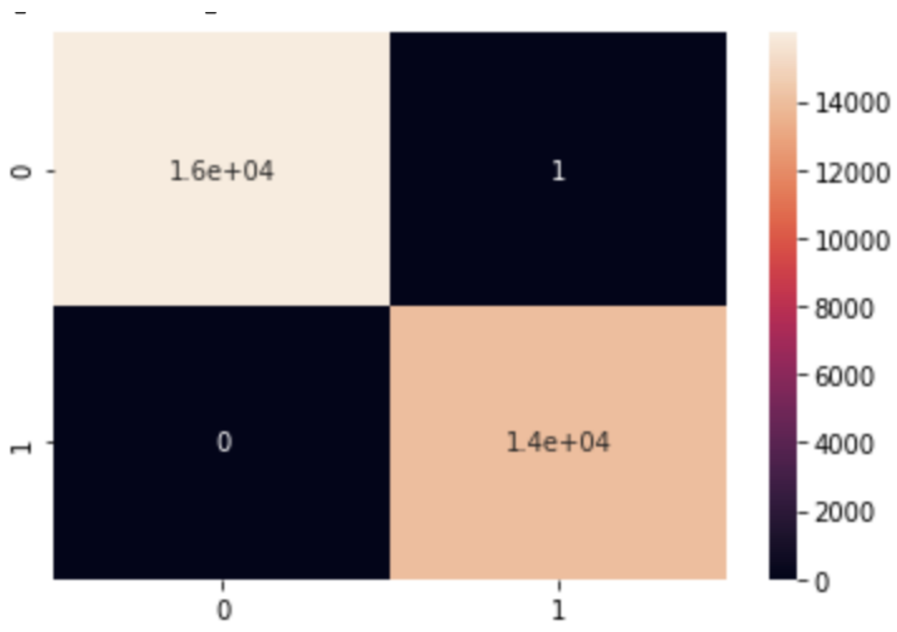


Figure 4.0.9: Confusion matrix of DenseNet121 Model

Figure 4.0.10 shown below, highlights the Vgg16 model predicts 14994 fake images truly as "fake" out of 16000 fake images and also predicts 1006 fake images falsely as "real". It also shows that this model predicts 12014 real images truly as "real" out of 14000 real images and predicts 1986 real images falsely as "fake".

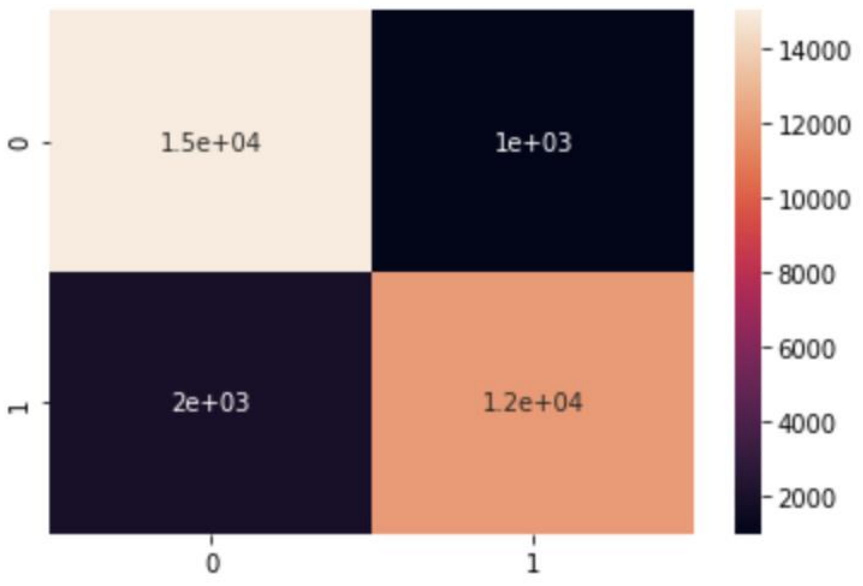


Figure 4.0.10: Confusion matrix of Vgg16 Model

Figure 4.0.11 shown below, describes that the Resnet50 model predicts 15910 fake images truly as "fake" out of 16000 fake images and also predicts 90 fake images falsely as "real". It also shows that this model predicts 13198 real images truly as "real" out of 14000 real images and predicts 802 real images falsely as "fake".

```
Text(0.5, 15.0, 'Predicted label')
```

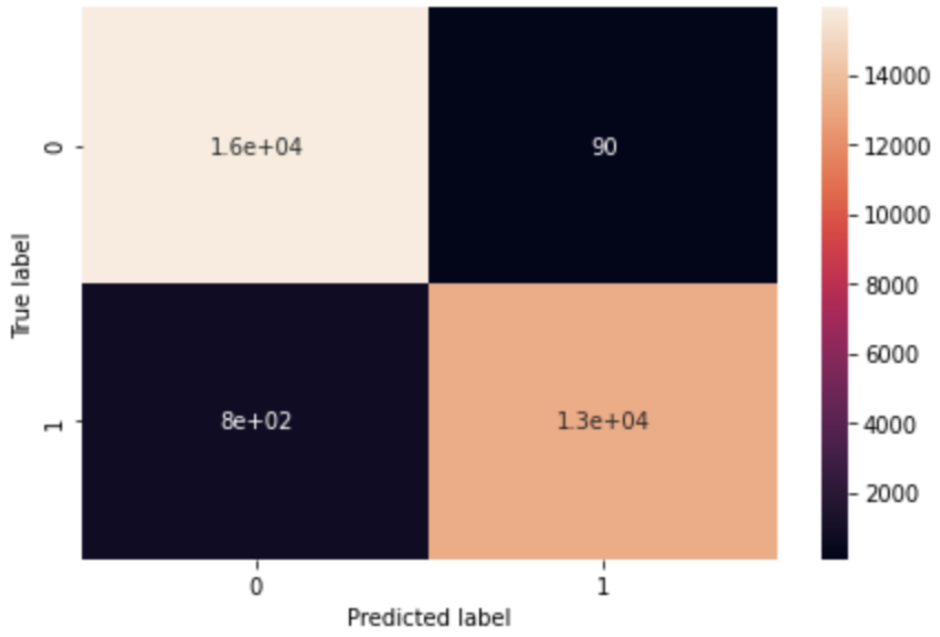


Figure 4.0.11: Confusion matrix of Resnet50 Model

Figure 4.0.12 shown below, represents that the NASNetLarge model predicts 14653 fake images truly as ‘fake’ out of 16000 fake images and also predicts 1347 fake images falsely as' real’. It also shows that this model predicts 10972 real images truly as "real" out of 14000 real images and predicts 3028 real images falsely as "fake".

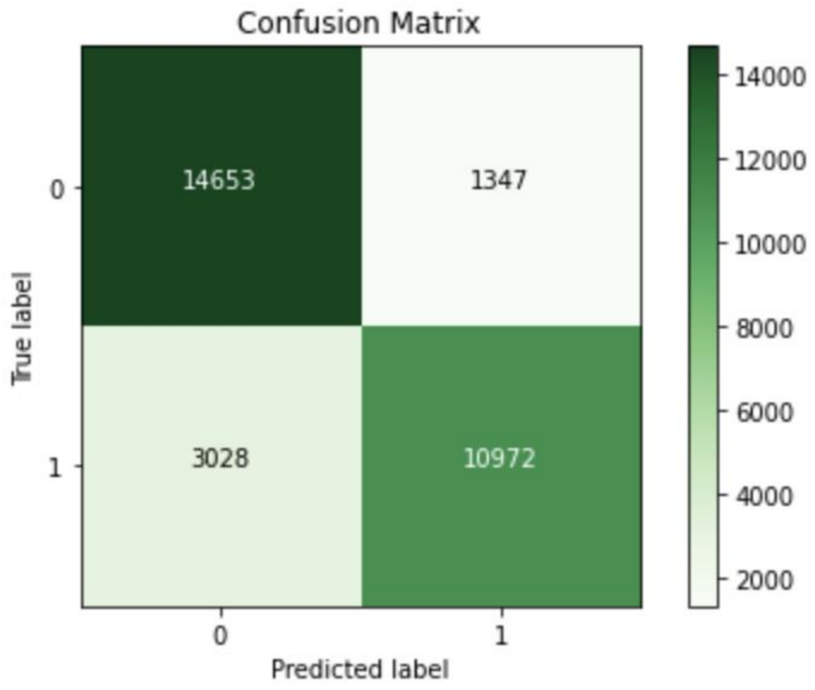


Figure 4.0.12: Confusion matrix of NASNetLarge Model

4.2 Comparative Analysis

Model accuracy, single image classification, model size, training time, average loss, and other criteria were used in this comparison. Following the implementation of four alternative models, the NasNetLarge model is shown to have the lowest accuracy in this dataset. The DenseNet121 model, on the other hand, has the best accuracy. For predicting fake and real images, the model has a test accuracy of 99.997 percent, a sensitivity of 99.99 percent, and a specificity of 100 percent. The reason for applying these four models is that numerous researchers have constructed various types of models with about 99 percent accuracy to far. However, by tweaking the dense layer, our denseNet121 model achieves a test accuracy of 99.997%, surpassing all prior records achieved by researchers and training on this dataset.

4.3 Summary

The outcomes and analysis of our system were discussed in this chapter. In this research, four models were employed. DenseNet121, Vgg16, Resnet50, and NASNetLarge are the models. We analyzed how each of the four models performed and how they differed in terms of size and results.

Chapter 5
Conclusion

5.1 Conclusion

DeepFakes are staying put, and that they have unalterably changed our perspective on the real world. Even models with a 97 percent accuracy rate are insufficient to detect real and fake photos and videos. The next generation of the DeepFake generation method is an infinite issue. To tackle this, we have conducted experiments on four of the existing CNN models (DenseNet121, VGG16, Resnet50, and NASNetLarge). After conducting several experiments on them, we have successfully achieved the new state-of-the-art with the DenseNet121 model, which can now distinguish between fake and real data with almost no errors and has a test accuracy of 99.997 percent, a sensitivity of 99.99 percent, and a specificity of 100 percent. We also tested a unique model called NasNetLarge, which has an accuracy of 85.416 percent in the test set and also tweaked the Vgg16 model. Therefore, the vgg16 model gives 90.03% test accuracy. There's also the possibility of enhancing the rest of the models with more hyperparameter tuning in the future.

Reference

[1] C. Chung, Hsu, Y. X. Zhuang, C. Y. Lee, "Deep Fake Image Detection Based on Pairwise Learning," *Applied Sciences,* vol. 10, no. 1, p. 1, 2020, doi:10.3390/app10010370.

[2] X. Ding, Z. Raziei, E. C. Larson, E. V. Olinick, P. Krueger , M. Hahsler, "Swapped face detection using deep learning and subjective assessment." *EURASIP Journal on Information Security*, vol. 6, 2020, doi:10.1186/s13635-020-00109-8.

[3] A. Khodabakhsh, R. Raghavendra, K. Raja, P. Wasnik, C. Busch, "Fake Face Detection Methods: Can They Be Generalized?" *2018 International Conference of the Biometrics Special Interest Group (BIOSIG)*, pp. 1-6, 2018, doi:10.23919/BIOSIG.2018.8553251.

[4] E. Sabir, J. Cheng, A. Jaiswal, W. A. A. mageed, I. Masi, P. Natarajan, "Recurrent Convolutional Strategies for Face Manipulation Detection in Videos." *In IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2019*, pp. 88-87, 2019, doi: arXiv:1905.00582

[5] R. Tolosana, R. V. Rodriguez, J. Fierrez, A. Morales, J. O. Garcia, "DeepFakes and Beyond: A Survey of Face Manipulation and Fake Detection." *ScienceDirect*, vol.64, pp. 131-148, 2020, doi: arXiv:2001.00179.

[6] X. Yang, Y. Li , S. Lyu, "Exposing Deep Fakes Using Inconsistent Head Poses." *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8261-8265, 2019, doi: 10.1109/ICASSP.2019.8683164.

[7] R. Durall, M. Keuper, F. Pfreundt, and J. Keuper, "Unmasking deepfakes with simple features." *arXiv preprint arXiv, 2019,* doi:1911.00686.

[8] L. Li, J. Bao, T. Zhang, H. Yang, D. Chen, F. Wen, B. Guo, "Face X-ray for More General Face Forgery Detection", 2020, doi: arXiv preprint arXiv:1912.13458v22.

[9] S. Y. Wang, O. Wang, R. Zhang, A. Owens, A. A. Efros, "CNN-Generated Images are Surprisongly easy to spot... for now", 2019, doi: arXiv preprint arXiv:1912.11035.

[10] N. T. Do, I. S. Na, S. H. Kim, "Forensics Face Detection From GANs Using Convolutional Neural Network." *International Symposium on Information Technology Convergenc.* South Korea, 2018.

[11] D. Afchar, V. Nozick, J. Yamagishi, I. Echizen, "Mesonet: a compact facial video forgery detection network." *2018 IEEE International Workshop on Information Forensics and Security (WIFS)*, pp. 1-7, 2018, doi:10.1109/WIFS.2018.8630761.

[12] B. Zi, M. Chang, J. Chen, X. Ma, Y. G. Jiang, "WildDeepfake: A Challenging Real-World Dataset for Deepfake Detection." *Proceedings of the 28th ACM International Conference on Multimedia.* New York, 2020, doi: arxiv-2101.01456.

[13] M. Koopman, A. M. Rodriguez, Z. Geradts, "Detection of Deepfake Video Manipulation ." *20th Irish Machine Vision and Image Processing conference (IMVIP 2018).* Northern Ireland, United Kingdom, 2018.

[14] D. Cozzolino, J. Thies, A. Rössler, C. Riess, M. Nießner, L. Verdoliva, "Forensictransfer: Weakly-supervised domain adaptation for forgery detection." *arXiv preprint, 2018,* doi: arXiv:1812.02510.

[15] X. Xuan, B. Peng, W. Wang, J. Dong, "On the generalization of gan image forensics. In Chinese Conference on Biometric Recognition." *Springer,* pp. 134-141, 2019, doi: arXib. abs/1902.11153

[16] D. Güera, E. J. Delp, "Deepfake Video Detection Using Recurrent Neural Networks." *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS),* 2018, doi:10.1109/AVSS.2018.8639163.

[17] A. Rossler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, "Faceforensics: A large-scale video dataset for forgery detection in human faces." *arXiv preprint,* 2018, doi:arXiv:1803.09179.

[18] M. S. Rana and A. H. Sung, "DeepfakeStack: A Deep Ensemble-based Learning Technique for Deepfake Detection," *2020 7th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2020 6th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom),* 2020, pp. 70-75, doi: 10.1109/CSCloud-EdgeCom49738.2020.00021.
[19] L. Guarnera, O. Giudice and S. Battiato, "Fighting Deepfake by Exposing the Convolutional Traces on Images," in *IEEE Access*, vol. 8, pp. 165085-165098, 2020, doi: 10.1109/ACCESS.2020.3023037.
[20] Z. He, W. Zuo, M. Kan, S. Shan and X. Chen, "AttGAN: Facial Attribute Editing by Only Changing What You Want," in *IEEE Transactions on Image Processing*, vol. 28, no. 11, pp. 5464-5478, Nov. 2019, doi: 10.1109/TIP.2019.2916751.

[21] [online].Available:https://www.kaggle.com/xhlulu/flickrfaceshq-dataset-nvidia-resized-256px?fbclid=IwAR2t-6gPHiN19M7PBxp05WEoeZoMp3bqnfzdJdotaeiY2TCL3-y6mmUsg5I.

[22] [online]. Available: https://www.kaggle.com/c/deepfake-detection-challenge/discussion/121173.

[23] [online]. Available: R. Thakur, "Step by step VGG16 implementation in Keras for beginners," towards data science, 2019.

[24] S. S. Khalil, S. M. Youssef, and S. N. Saleh, “A multi-layer capsule-based forensics model for fake detection of digital visual media,” in 2020 International Conference on Communications, Signal Processing, and their Applications (ICCSPA), 2021, pp. 1–6, doi: 10.1109/ICCSPA49915.2021.9385719.

[25] M. D. K. Hasan, S. Ahmed, Z. M. E. Abdullah, M. M. Khan, D. Anand, A. Singh, M. AlZain, M. Masud, "Deep Learning Approaches for Detecting Pneumonia in COVID-19 Patients by Analyzing Chest X-Ray Images", MathematicalProblems in Engineering, vol. 2021, Article
ID 9929274, 8 pages, 2021. https://doi.org/10.1155/2021/9929274