

# A Deep Face-Mask Detection Model using DenseNet169 and Image Processing techniques

by

Durjoy Bhowmik

17301153

Mohd.Rahat Bin Abdullah

17301215

Mohammed Tanvirul Islam

17301056

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

Department of Computer Science and Engineering  
Brac University  
January 2022

© 2022. Brac University  
All rights reserved.

# Declaration

It is hereby declared that

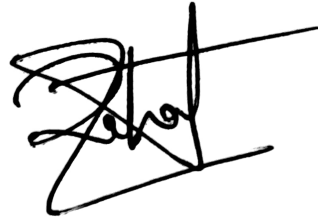
1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

**Student's Full Name & Signature:**

Durjoy Bhowmik

---

Durjoy Bhowmik  
17301153



---

Mohd. Rahat Bin Abdullah  
17301215

মহম্মদ তানবিরুল ইসলাম

---

Mohammed Tanvirul Islam  
17301056

# Approval

The thesis/project titled “FaceMask Detection using Deep learning” submitted by

1. Durjoy Bhowmik(17301153)
2. Mohd.Rahat Bin Abdullah(17301215)
3. Mohammed Tanvirul Islam(17301056)

Of Fall, 2021 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on January 16, 2021.

## Examining Committee:

Supervisor:  
(Member)



---

Dr. Jia Uddin  
Assistant Professor (Research Track)  
Department of Technology Studies, Endicott College  
Woosong University, Daejeon, South Korea Associate Professor (On leave)  
Department of Computer Science and Engineering  
Brac University

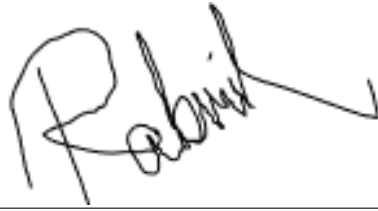
Co-Supervisor:  
(Member)



---

Md. Tanzim Reza  
Lecturer  
Department of Computer Science and Engineering  
Brac University

Thesis Coordinator:



---

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

Head of Department:  
(Chair)



---

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

# Abstract

The world stood still during the massive breakout of the Covid-19 worldwide. This massive outbreak of this contagious disease was occurred by being airborne. Not only COVID but also there are many other contagious disease which spread through air. So at present time, mask has become an essential part of our life which protects us from being affected from getting affected by COVID along with small diseases like cold, flu etc. We can get rid of these diseases and stop them from spreading just by wearing a face mask properly. In our research we would propose a way to identify or detect whether a person is using a face mask properly or not. For this we have used image data. The dataset that we have use are being made by us. Which consists of 1,45,537 images. We have divided this dataset into three segments. Which are with mask, without mask and misplaced mask. Among them 1,45,537 number are of images are of Asian region and rest is of the other countries. The main idea was to detect masked face properly using Deep learning architecture. We have implemented DenseNet169 and VGG19 to train the model and test it on images and videos. The accuracy that we got by using DenseNet169 is 91.47% in color images and 88.83% in grayscale. On the other hand in VGG19 we have got accuracy of 88.52% in color images and 92.4% in grayscale. Which makes this model more reliable than the rest. When we implemented this on video we got accuracy of 75.36% in DenseNet169. On the other hand, in VGG19 we have got 92.30% from gray scale. We have tried to provide a brief understanding of this architecture along with statistical results that we got from our dataset with a view to identify a person wearing mask properly or not. In addition it can identify the persons without wearing mask or persons wearing mask improperly.

**Keywords:** COVID-19; Machine Learning; Transfer learning; CNN; Densenet169; VGG19; Face Mask; Video detection; Softmax.

# Dedication

It is our genuine gratefulness and warmest regard that we dedicate this work to all our loved ones for their love and inspiration.

## Acknowledgement

Firstly, all praise to the Almighty for whom our thesis have been completed without any major interruption.

Secondly, to our supervisor Dr. Jia Uddin sir for giving his precious time to us and special thanks to our co-supervisor Md. Tanzim Reza sir for his kind support and advice in our work. He helped us whenever we needed help. He was always there for us no matter what time it was. His helping nature and dedication inspired us even more to work for the research.

Thirdly, we are very happy to express our appreciation and gratitude to the Department of Computer Science and Engineering, BRAC University and our educators for assisting us with all the fundamental help.

And finally to our parents without their throughout support it may not be possible. With their kind support and prayer we are now on the verge of our graduation.

# Table of Contents

<b>Declaration</b>	<b>i</b>
<b>Approval</b>	<b>ii</b>
<b>Abstract</b>	<b>iv</b>
<b>Dedication</b>	<b>v</b>
<b>Acknowledgment</b>	<b>vi</b>
<b>Table of Contents</b>	<b>vii</b>
<b>List of Figures</b>	<b>ix</b>
<b>List of Tables</b>	<b>x</b>
<b>Nomenclature</b>	<b>xi</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Problem Statement . . . . .	2
1.2 Motivation . . . . .	3
1.3 Research Objectives . . . . .	3
1.4 Thesis Outline . . . . .	3
<b>2 Related Work</b>	<b>5</b>
<b>3 Dataset and Workflow Analysis</b>	<b>9</b>
3.1 Dataset and Workflow Analysis . . . . .	9
3.2 Data and Analysis . . . . .	10
3.2.1 Data Collection Statistics . . . . .	10
3.3 Preprocessing . . . . .	10
3.3.1 Data Augmentation . . . . .	10
3.3.2 Grayscale Conversion . . . . .	11
3.3.3 Activation Function: Rectified Linear Units(ReLu) . . . . .	11
3.4 Classifier . . . . .	12
3.4.1 Softmax . . . . .	12
3.5 Optimizer . . . . .	13
3.5.1 Adam optimizer . . . . .	13
3.6 Sample Dataset . . . . .	14
3.6.1 Dataset Analysis and Statistics . . . . .	14



<b>4</b>	<b>Methodology</b>	<b>19</b>
4.1	Convolutional Neural Network . . . . .	19
4.2	Transfer Learning . . . . .	20
4.3	Machine Learning . . . . .	22
4.4	Save and Load Function . . . . .	23
4.5	Densenet-169 . . . . .	24
4.6	VGG-19 . . . . .	26
4.7	Confusion Matrix . . . . .	27
<b>5</b>	<b>Implementation and Result Analysis</b>	<b>28</b>
5.1	Densenet169 . . . . .	28
5.1.1	Implementing DenseNet169 on Images . . . . .	28
5.1.2	Implementing DenseNet169 on video . . . . .	29
5.1.3	Confusion matrix for Densenet169(Color Image) . . . . .	29
5.1.4	Confusion matrix for Densenet169(Grayscale Image) . . . . .	29
5.1.5	Iteration of checking loss and accuracy(Densenet169) . . . . .	29
5.2	VGG19 . . . . .	33
5.2.1	Implementing VGG19 on Images . . . . .	33
5.2.2	Implementing VGG19 on video . . . . .	33
5.2.3	Confusion matrix for VGG19(Color Image) . . . . .	33
5.2.4	Confusion matrix for VGG19(Grayscale Image) . . . . .	33
5.2.5	Iteration of checking loss and accuracy(VGG19) . . . . .	36
5.3	Comparison of result between DenseNet169 and VGG19 . . . . .	36
<b>6</b>	<b>Conclusion and Future Work</b>	<b>38</b>
	<b>Bibliography</b>	<b>41</b>

# List of Figures

3.1	Workflow Diagram . . . . .	9
3.2	Grayscale Image . . . . .	11
3.3	Residual learning: a building block . . . . .	12
3.4	With,Without,Misplaced Mask . . . . .	15
3.5	With,Without,Misplaced Mask . . . . .	15
3.6	Statistical analysis of the colored image data sets . . . . .	16
3.7	Statistical analysis of the colored train image data sets . . . . .	16
3.8	Statistical analysis of the colored test image data sets . . . . .	17
3.9	Statistical analysis of the Gray Scale image data sets . . . . .	17
3.10	Statistical analysis of the Gray Scale Train image data sets . . . . .	18
3.11	Statistical analysis of the Gray Scale Test image data sets . . . . .	18
4.1	Convulational Neural Network Architecture . . . . .	20
4.2	Transfer Learning . . . . .	21
4.3	Transfer Learning with Pre-trained Deep Learning Models as Feature Extractors . . . . .	22
4.4	Fine Tuning Off-the-shelf Pre-trained Models . . . . .	22
4.5	Classification of save and load function . . . . .	23
4.6	DenseNet Architecture . . . . .	24
4.7	DenseNet Architecture . . . . .	25
4.8	A DenseNet Architecture with 3 dense blocks . . . . .	25
4.9	The Structure of VGG-19 Network . . . . .	26
4.10	VGG19 network model . . . . .	26
4.11	Training frame of mask detection model . . . . .	27
5.1	Accuracy of DenseNet169 on color images . . . . .	28
5.2	Accuracy of DenseNet169 on grayscale images . . . . .	29
5.3	Confusion Matrix of DenseNet169 on Color image . . . . .	30
5.4	DenseNet169 Color image F1-score . . . . .	30
5.5	Confusion Matrix of DenseNet169 on Grayscale image . . . . .	31
5.6	DenseNet169 Grayscale image F1-score . . . . .	31
5.7	Accuracy of VGG19 on color images . . . . .	33
5.8	Accuracy of VGG19 on grayscale images . . . . .	34
5.9	Confusion Matrix of VGG19 on Color image . . . . .	34
5.10	VGG19 Color image F1-score . . . . .	35
5.11	Confusion Matrix of VGG19 on Grayscale image . . . . .	35
5.12	VGG19 Grayscale image F1-score . . . . .	35
5.13	Accuracy comparison between Densenet169 and VGG19 . . . . .	37

# List of Tables

4.1	TP, TN, FP and FN parameter definitions . . . . .	27
5.1	Iteration table of DenseNet169 on color images training & testing . .	32
5.2	Iteration table of DenseNet169 on gray scale images training & testing	32
5.3	Iteration table of VGG19 on color images training & testing . . . . .	36
5.4	Iteration table of VGG19 on gray scale images training & testing . .	36

# Nomenclature

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

$\alpha$  Alpha

$\beta$  Beta

$\delta$  Delta

$\epsilon$  Epsilon

$\sigma$  Sigma

*AI* Artificial Intelligence

*CCTV* Closed Circuit Television

*CNN* Convolutional Neural Networks

*MI* Machine Learning

*ReLU* Rectified Linear Units

*SARS* Severe Acute Respiratory Syndrom

*VGG* Visual Geometry Group

# Chapter 1

## Introduction

CoronaVirus, which is also known as the COVID-19 or COVID, is actually a pandemic on a global scale that is ongoing because of Coronavirus disease. SARS-CoV-2 is another common name for this disease, which is also known as COVID-19. When this happens, you might have SARS (severe acute respiratory syndrome). The illness COVID-19 is extremely infectious. The virus was discovered for the first time in Wuhan, China. That December, the incident happened [19]. CoronaVirus was therefore designated a public health danger of worldwide concern by the WHO on January 30th, 2020. On March 11, 2020, it was declared a pandemic. In reality, between May 31, 2020, and May 31, 2021, COVID-19 was responsible for over 281 million confirmed illnesses and 5.41 million confirmed deaths which made this one of the worst pandemics in the history of mankind [30]. One of the major safety measures for being safe from getting affected by coronavirus is wearing a face mask which has been instructed by WHO. Because this virus spreads through the channel of air. So at the time when someone who is affected, sneezes or makes communication with another person, the droplets of water disseminate through the air coming out from their nose and mouth which affect other people. As a result, the gravitate of wearing a facemask in public places is on the rise because of this Coronavirus epidemic caused by COVID-19 is happening throughout the world. Before this pandemic people used to use a face mask to avoid air pollution mostly from their self-consciousness. But now using a face mask has become a must for everyone for their own safety. Scientists have discovered that wearing a face mask works as a way of stopping COVID-19 transmission from one person to another[14]. World wide scientific co-operations got an extraordinary degree of rising by the coronavirus epidemic. AI that is based on Machine Learning (ML) and even Deep Learning can be very useful in the fight against COVID-19 in many ways. ML allows evaluating the huge quantity of data for evaluating the situation and gives directions before it spreads more from the data of the collective. It can also detect infections. Such as, AI can detect infection from the chest X-ray of a patient. The situation is getting challenging day by day because of the spreading of COVID-19 and its transmission perpetuation got so bad that some countries had to impose the law of wearing face masks in public. This law was made because of the huge death rate of affected patients. It's a tough task to monitor such a huge population of the countries. The process of monitoring includes the detection work of anyone who's not wearing a mask on their face. Face-mask detection applications have grown popular as a result of the COVID-19 epidemic. When other sanitizing and social distancing issues

have previously been solved, the face mask detection issue has yet to be adequately handled. At this point wearing a mask on the face in this pandemic situation is an important preventive measure. It's a must to do for the people who are at great risk of illness at a severe level because of COVID-19. According to researchers, COVID-19 is transferred by persons who come into touch with one another within 6 feet. It can also be transferred by persons who are sick but have no symptoms or are unaware that they are infected. Because of this CDC (Centers Disease Control And Prevention) has recommended wearing a mask on the face in public areas for all people over 2 years of age. So the detection of face-mask has become a problematic issue in the domain of computer vision and processing of the image. We can have various use cases from face recognition to facial motion capturing and revealed with high precision. Because machine learning algorithms have improved so quickly, the risks of not utilizing face-mask detection technologies have yet to be adequately addressed. Furthermore, this is a more important technique because it is utilized for more than only face identification in static pictures and movies. In addition, real-time supervision and inspection. At present the advancement of convolution neural networking and deep learning, we can get high accuracy in the classification of images and detection of objects. As a security measure, France has installed AI software in the Paris Metro system's cameras which are used for surveillance, to detect people not wearing a mask. In addition, it gathers statistical data anonymously which can be used in the work of prediction of the authorities and help the potential outbreak of the COVID-19. The software system was developed by DatakaLab. As a matter of fact, in this paper, we are working on a face-mask detection model which will be based on classical machine learning classifiers and deep transfer learning. This can be used to detect people not wearing masks on their faces to make the COVID-19 transmission less. This project is going to be an integrated form of deep transfer learning and classical machine learning algorithm. We will try to find out the best algorithm which gives the best accuracy and consumes less time to detect and train procedures.

## 1.1 Problem Statement

Facemask detection is a classification and detection issue as it involves detecting the faces of individuals in digital photographs and then deciding whether or not they are wearing a mask. The first component of this issue has garnered a lot of interest in computer vision literature because of the extensive deployment of detect people face-detection methods. Secondly, the used component, on the contrary(determining if a face is covered or not), has only lately piqued the interest of academics in the aftermath of the coronavirus epidemic, despite substantial work on this in the previous year. In most situations, it just checks to see if the image has a mask. There is no extra effort taken to ensure that the masks are correctly applied to the face and therefore used in line with the recommendations of health specialists. This lowers the effectiveness of current face-mask detection methods, necessitating additional computer vision research into not just identifying the existence of face masks in photographs but also assessing whether they are worn correctly. MAsked FAcEs(MAFA) is among the most often used datasets for face-mask detection training, and it consists of only primary labels specifying whether or not face-masks are visible in the pictures, which is similar to most other publicly accessible datasets

for this purpose. Facial pictures in such datasets generally fall into one of two categories: face with properly maintained masks (marked green), and face with poorly maintained masks (marked red). Because the precision of mask installation is unknown, mask detectors that exist often get learned from excessive noisy data (given the detector's intended function) and don't flag faces when doesn't cover their mouth but a mask is present, nose or chin. We've noticed that most existing face-mask detection experiments have this problem. In most cases, we can see any information for confusing images without masks datasets. we will be able to avoid the above-mentioned problems on a bigger note.

## 1.2 Motivation

Since the emergence of COVID-19, the World Health Organization has released various preventative measures in order to combat the spread of coronaviruses. One of the most obvious rules is to keep one's distance from others and keep one's surroundings clean. Face masks inhibit the corona from spreading across a whole city. There are now mandatory face mask rules in the majority of countries. Keeping an eye on a face mask manually may be difficult, especially in congested areas like hospitals, airports, train stations, and shopping malls. This prompted researchers to develop an automated approach for detecting face masks.

## 1.3 Research Objectives

This research aims to develop a Facemask detection system for detecting facemasks in also confusing images/videos. We will use computer vision and deep transfer learning to determine whether or not the individual in the image or video stream has worn a face mask. Usually, transfer learning uses feature detection capabilities of the pre-training methods and applies them to our simple model. **The research objectives of our thesis project are given below:**

- To deeply understand Transfer learning, and how it works.
- To develop a model for Densenet169 and VGG19 which will give us better results and apply it to deep transfer learning.
- To develop the datasets for the proposed model and evaluate the model.
- To deeply understand facemask detection techniques and offer recommendations on improving the model.

## 1.4 Thesis Outline

**Chapter 1** Introduction. In this chapter, we've provided an overview of our project's issue description, motivation, research aims, and methodology.

**Chapter 2** Literature Review. We discussed publications from the field of computer science that addressed a similar subject. Additionally, the background study's objective was to ascertain the shortcomings of earlier studies. Furthermore, we have mentioned our contribution and the rationale for collecting primary data.

**Chapter 3** Workflow and Data analysis. Details of our research methodology are covered in-depth, as are the many classification schemes that were employed in the various algorithms.

**Chapter 4** Methodology. Densenet169 and VGG19 are explained in this section, as well as their designs and mathematical formulae.

**Chapter 5** Results and Accuracy. In this section, we show the graphs of the accuracy and loss of all the methodologies and classifiers we used in our thesis work. There were also comparisons between the algorithms, as well as confusion matrices and compared between the algorithms.

**Chapter 6** Conclusion and Future work. We conclude our study with a quick review of our findings and some suggestions for how we may enhance our work in the future.



# Chapter 2

## Related Work

Due to the worldwide COVID-19 coronavirus outbreak, the use of face masks in public is becoming more popular. People used to use masks to protect their health from air pollution before Covid-19. Other people hide their feelings from the world by masking their faces, while others feel self-conscious about their appearance. COVID-19 can be slowed down by wearing a face mask, according to research. COVID-19 transmission is slowed by wearing face masks, according to scientists [14]. Object detection and recognition in numerous application areas have seen considerable advances in deep learning throughout time. The majority of the research focuses on image reconstruction and facial recognition to confirm identity. The primary goal of this study, however, to prevent the spread of COVID-19, is necessary to identify persons who do not use masks in public places. The manuscript separates the classification into two categories: an individual wearing a mask and no mask identified. This whole system is going to make contributions in crowded places like shopping malls, airports, railway stations or supermarkets. This part aims to critically review previous relevant work in the field of Facemask Detection Systems in the context of transfer learning and specifically in the context of Deep learning. We will analyse the different techniques used for the main results achieved and we will show how to stop COVID-19 from spreading further, identify those who don't use masks in public places, the limited computational capabilities and the massive number of people that makes the facemask detection systems harder.

According to [20], The author proposes RetinaFaceMask, an efficient and high-accuracy facemask detector. The proposed RetinaFaceMask detector is a one-step detector that combines high-level semantic information with different feature maps through a Pyramid Feature Network (FPN) and a novel context module to identify face masks. They provide a context attention detection and cross-class object removal head method to enhance detection. Furthermore, since the face mask dataset is tiny and features may be difficult to extract, Transfer learning is used to transfer learnt kernels from networks trained on a large dataset for a comparable face recognition job. RetinaFaceMask yields state-of-the-art face and masks detection precision, respectively, of 2.3% and 1.5% higher than the baseline result, and 11.0% and 5.9% better than the baseline result in a recall.

According to [11], The author offered many methods for Multi-scale face mask categorization and real-time detection for the medical industry in this COVID-19 Pandemic Situation. Crowd monitoring on the roadways and in shopping malls is more

useful. They employed two separate face mask datasets of 680 and 1400 pictures, respectively, and constructed two different detection models, FMY3 employs the Yolov3 Algorithm, whereas FMNMobile employs the NASNetMobile and Resnet SSD300 Algorithms. Both models achieved the 34 percent Mean Average Precision (mAP) and 91.7 per cent Recall rate on the FMY3 Model and 98 percent and 99 percent accuracy and recall rate on the FMNMobile Model by calculating various probabilistic accuracy measures.

According to [25], the authors experimented on networks train detection precision of three. The first dataset was of 90,000 unmasked faces and 5000 masked faces. 10,000 images were being used to dataset balance of RMFD dataset were being used for validation, testing and training. The second dataset utilized was the SMFD (Simulated Masked Face Dataset), which included 1570 images. Among them, for the validation and testing stages, 785 was for uncovered faces and 785 was for simulated masked faces. The third dataset was LFW(Labeled Faces in the Wild). which contained masked faces of 13,000. Which consisted of celebrities from all around the world and was solely utilized as a benchmark testing dataset throughout the testing phase. However, the suggested model was never trained on it. Here two models were being proposed. The first proposed model was used by ResNet50. Which was featured as an extractor. The second proposed model was classical machine learning. Such as SVM(Support Vector Machine), ensemble and decision tree. Here, ResNet50 was used for the traditional machine learning and the extraction phase. It was used for validation, testing and training. Logistic Regression, K-Nearest Linear Regression, and Neighbors Algorithm are the Ensemble techniques utilized. MATLAB was used for implementation. 70% of the dataset was used for training, 10% for validation and 20% for the testing phase. Following installation and testing, the DS1 testing accuracy ranged from 93.44 percent using a decision tree classifier to 99.64 percent using an ensemble classifier. When utilizing the decision tree classifier for DS4 testing, accuracy varied from 99.76 percent to 100 percent. When using the SVM classifier, accuracy ranged from 99.76 percent to 100 percent. The decision tree classifier achieved 99.54 percent accuracy on the face mask dataset, while the SVM classifier achieved 99.49 percent accuracy. The author achieved 99.50 percent accuracy with the decision tree classifier and 99.35 percent using the SVM classifier on the combined mass dataset. The author used deep transfer learning models to approach the mask faces from the neurotrophic environment. Here they got the highest accuracy using the SVM classifier. Such as 99.49% in SMFD, 100% in LFW and 99.64% in RMFD.

In the paper [23], the author utilized a deep learning model built by "Yolov5" to develop an excellent approach for detecting face masks. The YoloV5 trains the model across several epochs ranging from five to ten. Epochs are: 20, 50,100, 300 and 500. There were 853 pictures in the public face mask collection, with three labels: "Without Mask," "Incorrect Mask" and "With Mask." The face mask dataset's 853 pictures were separated into three groups: model testing required 86 images, model training required 682 photos, and result validation required 85 images. The author separated this model into two parts: the face mask detection model and the training model. The training dataset of 685 pictures with annotations in the VOC format was used to construct the face mask detection model using YoloV5. An epoch, or several runs over the whole training dataset, has been demonstrated in the training model process to impact the model's performance. The generated deep learning

models are also evaluated on the face mask dataset of 86 images. The number of processing steps rises as the number of epochs increases, which improves mask detection performance. However, the experimental findings revealed that the model trained with 300 epochs performed the best, even outperforming the model trained with 500 epochs. Face mask detection using 300 epochs provides the best accuracy, with a precision and recall of 96.5 percent.

According to [15], the authors used a deep learning method known as Facemasknet and got an accuracy of 98.6%. They've designed their project into two phases which are implementing and training the detector. A total of 15 photos of inappropriately worn people were included in the dataset. 10 photos with masks, 10 photos without masks. They created their detector model using MATLAB programming. FacemaskNet architecture was used to train the model. With a total of 20 training epochs, the initial learning rate is  $1e-4$ . The pictures in the input image are scaled to  $227 \times 227 \times 3$  pixel intensity. Following that, the model was built to be trained, and it was tested on the test set. The input image is loaded and preprocessed. Face detection, as well as the region of interest, are used to determine where all of the image's faces are located. Their constructed model is of 8 layers. Images are taken by the input layer and The 2D convolution layer is employed since it accepts three-dimensional input (red, blue, and green) pixels. In convolutional neural networks, the Rectified Linear Unit (ReLU) layer is utilized to activate output functions. The norm1 layer allows for faster training and greater accuracy. Images are scaled using the max-pooling layer. The needed classification, which identifies a face and a mask, is generated after employing the softmax layer.

According to [27], MobileNetV2, a convolutional neural network that requires short processing resources and can be easily integrated into mobile apps and computer vision, is used in the model design. As a consequence, it will be able to create an inexpensive mask detection system that can help determine if a person is wearing a mask or not as well as function as a surveillance system because it can handle both real-time videos and still pictures. With 99.56% accuracy on validation data, 99.75% accuracy on testing data, and 99.98% accuracy on training data, the face detector model was a success. There are 11,792 photos in the dataset, with 992 in the testing set, 10,000 in the training set, and 800 in the validation set. There are two types of images in each set: faces wearing masks and faces don't wear masks.

In [28], The author created the MAsked FAcEs (MAFA) dataset, which comprises 30,811 pictures, and constructed the LLE-CNNs model, getting an Average Precision of 76.4 percent, according to the author. On the RMFD, ResNet-50 and SVM achieved 99.49 percent accuracy. In a subsequent article, For object detection, they utilized YOLO-V2, which is based on ResNet-50 and obtained an average precision of 81 percent on the Face Mask Dataset (FMD) and Medical Masks Dataset (MMD). They used an InceptionV3-based model to achieve 99.9 percent accuracy on the Simulated Masked Face Dataset. The final training dataset included 4065 MAFA pictures, 3894 WIDER FACE images, and 1138 extra internet images. In all, 9,097 pictures with 17,532 labelled boxes were separated into two groups: 80 percent training and validation, and 20 percent testing. By using consumers' current devices, our serverless edge-computing approach reduces hardware expenses. Convert the PyTorch model to an NCNN model with a model size of 582 KB using the NCNN library. Then, using the NCNN library for inference, we developed a new C++ application to automate the detection process from picture preprocessing

to eventually outputting the box location, category, and confidence. [24] The identified bounding boxes were rendered with original face pictures using HTML and CSS to show the detection findings. The trained YOLO model achieved an average precision of 0.52 after 120 epochs with a batch size of 16. On typical edge devices, the detection FPS is shown. Using YOLOV2 and ResNet-50, Mohamed Loey et al. achieved an AP of 81 percent. The accuracy of their model was 89 percent.

From the above-mentioned discussion, we can see that the majority of the researchers use the YOLOV3 algorithm for its efficiency and accuracy. Though there are so many other efficient algorithms available, Such as CNN. On the other hand, in terms of using architecture. InceptionV3, Xception, MobileNet, MobileNetV2, VGG16, ResNet50 all are efficient. But ResNet50, MobileNetV2 and VGG16 are mostly used by the researchers for getting the most accuracy. So, for not only detecting facemask, the approach should be detecting and if not found then give notification or buzz alarm so that the purpose of the matter is greatly served and be a success.

# Chapter 3

## Dataset and Workflow Analysis

### 3.1 Dataset and Workflow Analysis

This Chapter describes the process through which we will carry out our task. Flowcharts have been used to depict our workflow in Section 3.1 of this document. Our research began with the collection of as many photos as possible of individuals wearing masks, people who were not wearing masks, and people who had misplaced their masks, as well as the collection of a large data set of persons who were wearing masks. It is necessary to resize all of the photographs to a single size of 100X100 pixels after they have been collected. It is the first step in the Pre-processing process. Pre-processing also covers the act of tagging and categorizing the photos that have been captured. As a classifier layer, we employed two algorithms, Softmax and Relu, which were both developed by us. We are pre-training using the Densenet-169 model and then applying transfer learning to improve performance. When it came to the hardware parameters we employed to run our models, we went with a Ryzen 7 1700 Eight-Core Processor, an Nvidia gtx 1050ti GPU, 16GB of RAM, and a 1TB hard drive. Last but not least, when we have run all of the models and gotten the results, we compare them with another model and with the two distinct classifiers, and we examine the data even further by comparing the findings with those from comparable research.

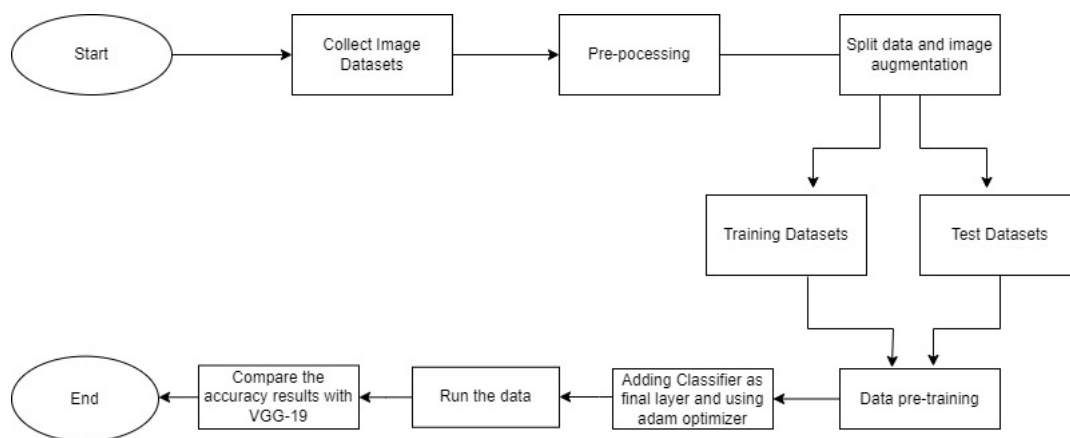


Figure 3.1: Workflow Diagram

## 3.2 Data and Analysis

### 3.2.1 Data Collection Statistics

One of the most challenging aspects of any learning-based research project is preparing the data for analysis. As a result, each dataset is unique and particular to the project. We've compiled a global sample of people to use in our research for this new model. Face masks are being worn by nearly everyone to protect themselves against COVID-19, and we have gathered over 146,000 images of people using various types, forms, and materials of face masks. Currently, our data collection has precisely 145537 normal photos, but we hope to expand it in the future to include data from other sources, allowing us to catalog nearly every sort of mask in existence. A mix of men and women are also seen wearing face masks appropriately and badly. We gathered data from a wide range of ages to ensure that our findings were as precise as possible. It's worth noting that we used a total of 124472 images in the process of training the model. Only 67440 of them are appropriately masked, 3828 are unmasked, and 53204 of them are incorrectly masked. Additionally, there are 21065 photos available for use in testing. Photos with masks make up 5848 of the images, while images without masks make up 1687 of them.

## 3.3 Preprocessing

### 3.3.1 Data Augmentation

All of the images in this research have been scaled for the purpose of image training and testing. Image resizing refers to the process of altering the image's scale. It allows for the selection of fewer pixels, which in turn reduces the training time required for the intended experiment. The greater the number of pixels that we must employ, the more difficult it will be for the model to perform because the pixels are our inputs for the model. This also makes it easier for us to scale down the images later on for the sake of zooming in or out on the picture. We used ImageDataGenerator to generate the data sets for our test, train and validation data sets. During the training-

- `rescale=1./255`; Where we are converting the RGB value from the range of 0-255 to a 0-1 number.
- `rotation range=20`; The degree to which the image will be rotated at random.
- `shear range=0.15`; Changing the angle in the opposite direction of the clock. The value is expressed in degrees.
- `fill_mode='nearest'`; It establishes rules for the pixels in the input region that have been moved.
- `width_shift_range=0.1`; It moves the picture to the left or right (horizontal shifts). If the value is float and less than 1, it will use the percentage of the total width as a range for the value.
- `height_shift_range=0.1`; It functions in the same way as width shift range.

- zoom range=0.15; When photos are randomly zoomed in and out, this function range will be used.
- horizontal flip=True; Horizontal image flipping at random.
- vertical flip=False; Image flipping at random, which is not something we will be doing. Just to make things easier for testing and validating the data set, we rescale the photos down to the range of 0 to 1.

When it comes to input, all of the images are 100X100 pixels in size and are shot in batches of 32. After that, the data is fed into the models we've built. First, we use Densenet169 and VGG19 models to build our models before putting them through the two separate methods.

### 3.3.2 Grayscale Conversion

Grayscale conversion from a color image necessitates a deeper understanding of the color image. Red, Green, and Blue are the three hues that make up an image's pixel color (RGB). The properties of brightness, chroma, and hue are used to depict the RGB color values in three dimensions (XYZ). The quality of a color image is determined by the number of bits that the digital device can represent in each color. There are four different types of color images: 8 bit, 16-bit high color, 24-bit true color (the closest thing to the real thing), and 32-bit deep color picture. The maximum number of colors permitted by the digital device is determined by the number of bits. The combination of Red, Green, and Blue occupies 24 bits and allows 16.7 million possible colors if each color fills 8 bits. A pixel's color is represented by 24 bits in a color picture. An 8-bit number is used to indicate brightness in a grayscale picture. 0 to 255 is the brightness scale for a grayscale picture. This means that the RGB values (24-bit) of a color image may be used as the basis for a grayscale image (8 bit). In this research, we are using RGB color images to train and test our model as well as we also using grayscale conversion images to train and test our model.



Figure 3.2: Grayscale Image

### 3.3.3 Activation Function: Rectified Linear Units(ReLU)

Convention dictates that immediately following each convolutional layer, a nonlinear layer or activation layer is applied. This layer's function is to add nonlinearity to a

system that has previously only computed linear processes during the convolution layer. While nonlinear functions such as tanh and sigmoid have been utilized in the past, researchers have shown that ReLU layers perform significantly better since the network can train much quicker (due to the computational efficiency) despite making a substantial impact to the accuracy. Because the gradients drop exponentially across each layer, the bottom levels of a network train at a much slower rate. This layer uses  $f(x) = \max(0, x)$  as the function applied to all of its input volume. Everything in this layer is set to 0, so it's really simple. Since it doesn't alter convolutional receptive fields, this layer improves model nonlinearity and overall network robustness.

$$f(x) = \operatorname{argmax}(0, x), \text{ Where } f(x) = \begin{cases} 0 & \text{if, } x < 0 \\ 1 & \text{if, } x \geq 0 \end{cases}$$

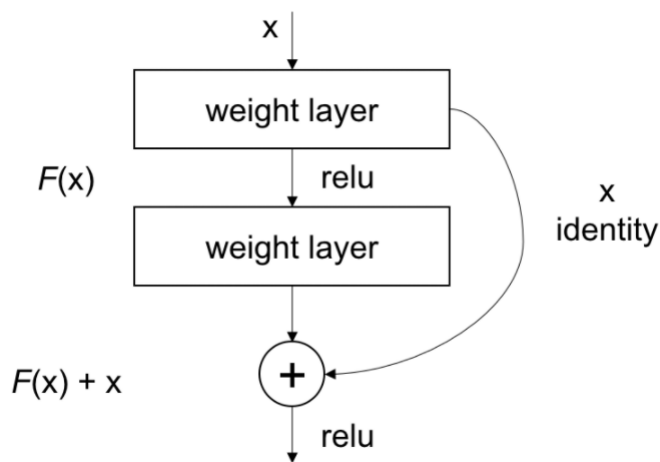


Figure 3.3: Residual learning: a building block [5]

## 3.4 Classifier

### 3.4.1 Softmax

One way to represent a set of numbers as probabilities is to use the Softmax mathematical function, which multiplies all the values in a set by the scale at which they appear in the vector. This is a classifier that we are use. Additionally, Softmax is an activation function. It is a type of classifier that is frequently used in deep learning architectures. It multiplies all values received, regardless of their nature, and converts them to a total of a number that will always be between 0 and 1. This is a probability function that converts a vector distribution of a number's probability distribution to a real distribution[12]. This is one of the classifiers that is used to validate the models' accuracy. Softmax is defined by the equation

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}}$$



Where,  $\sigma = \text{softmax}$

$\vec{z}$  = input vector

$e^{z_i}$  = standard exponential function for input vector

$K$  = number of classes in the multi-class classifier

$e^{z_j}$  = standard exponential function for output vector

$e^{z_j}$  = standard exponential function for output vector

## 3.5 Optimizer

### 3.5.1 Adam optimizer

Adam is an optimization technique that, in place of the standard stochastic gradient descent procedure, may be used to iteratively update network weights using training data. The approach is extremely efficient when dealing with complex problems with a large number of variables or data. It is memory-efficient and consumes less memory. On the surface, it appears to be a mix of the 'gradient descent with momentum' and the 'RMSP' algorithms[16]. Two gradient descent methods are combined in the Adam optimizer.

Momentum: This approach uses the 'exponentially weighted average' of the gradients to speed up the gradient descent algorithm. A faster convergence to the minima is achieved by using averages.

$$w_{t+1} = w_t - \alpha m_t$$

where,

$$m_t = \beta m_{t-1} + (1 - \beta) \left[ \frac{\delta L}{\delta w_t} \right]$$

$m_t$  = aggregate of gradients at time  $t$  [current] (initially,  $m_t = 0$  )

$m_{t-1}$  = aggregate of gradients at time  $t - 1$  [previous]

$W_t$  = weights at time  $t$

$W_{t+1}$  = weights at time  $t+1$

$\alpha_t$  = learning rate at time  $t$

$\partial L$  = derivative of Loss Function

$\partial W_t$  = derivative of weights at time  $t$

$\beta$  = Moving average parameter (const, 0.9 )

Root Mean Square Propagation (RMSP): Adaptive learning method that aims to enhance AdaGrad is RMSprop. In place of AdaGrad's cumulative sum of squared gradients, the 'exponential moving average' is used instead.

$$w_{t+1} = w_t - \frac{\alpha_t}{(v_t + \epsilon)^{1/2}} * \left[ \frac{\delta L}{\delta w_t} \right]$$

where,

$$v_t = \beta v_{t-1} + (1 - \beta) * \left[ \frac{\delta L}{\delta w_t} \right]^2$$

$W_t$  = weights at time  $t$

$W_{t+1}$  = weights at time  $t + 1$

$\alpha_t$  = learning rate at time  $t$

$\partial L$  = derivative of Loss Function

$\partial W_t$  = derivative of weights at time  $t$

$V_t$  = sum of square of past gradients. [i.e sum(/-1)] (initially,

$\beta$  = Moving average parameter (const, 0.9)

$\epsilon$  = A small positive constant ( $10^{-8}$ )

As a result, Adam Optimizer incorporates the strengths of the previous two approaches into a more efficient gradient descent. Using the equations from the previous two ways, we may arrive at

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) \left[ \frac{\delta L}{\delta w_t} \right] \quad v_t = \beta_2 v_{t-1} + (1 - \beta_2) \left[ \frac{\delta L}{\delta w_t} \right]^2$$

After each iteration, we instinctively adjust to the gradient descent such that it remains constant and impartial throughout the process, therefore the name Adam. Now, instead of our normal weight parameters  $m_t$  and  $v_t$ , we take the bias-corrected weight parameters. Putting them into our general equation, we get

$$w_{t+1} = w_t - \widehat{m}_t \left( \frac{\alpha}{\sqrt{\widehat{v}_t + \epsilon}} \right)$$

In all of our methods, this optimizer is used because of its high efficiency and minimal memory use requirements.

## 3.6 Sample Dataset

### 3.6.1 Dataset Analysis and Statistics

One of the most important aspects of learning-based research is the creation of the dataset. Each dataset in this case is unique and particularly relevant to the topic at hand. This collection includes images of Bangladeshi individuals, as well as images of people from across the globe. In our dataset, we've mostly included images of people from the Asian area. At this time, the most common protective measure people take to avoid contracting COVID-19 is to don a face mask. Images of people with masks of various materials, shapes, and designs, as well as those without masks, have been gathered. Those were also cited in the article. We now have a dataset of 1,45,537 photographs, but we intend to add more in the future so that our data set includes every facet of mask use, whether it is intentional or not. We have data on men, women, and children who are wearing masks correctly, masks that are misplaced, and masks that are not being worn at all. To ensure that our results are as accurate

as possible, we have gathered images of people of all ages in our collection. Moreover, we also convert RGB images to Grayscale.

Here we have collected 1,45,537 images for our dataset in which 73288 images are

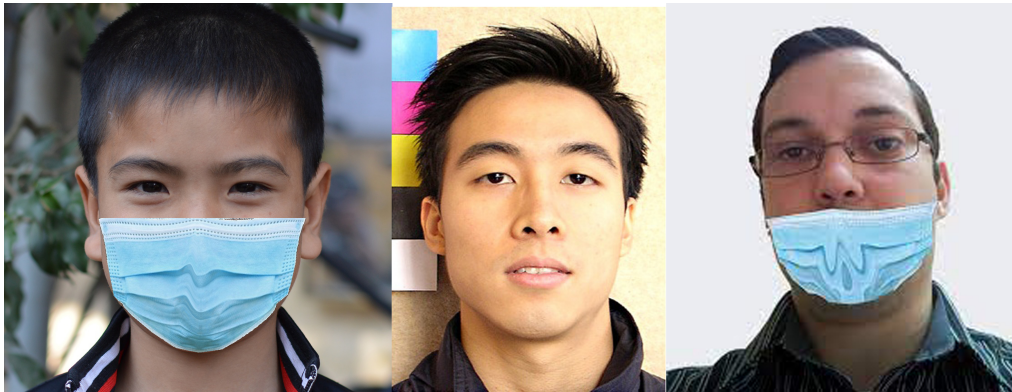


Figure 3.4: With, Without, Misplaced Mask



Figure 3.5: With, Without, Misplaced Mask

of with mask, 66734 images are of misplaced masks and 5515 images are of without mask. Then we had divided the dataset of color images into two parts for test and train purpose. For train purpose we have taken 124472 images of which 67440 were with mask, 53204 were misplaced mask and 3828 without mask. Then we took 21065 images for test purpose. In which 5848 are with mask, 13530 are misplaced mask and 1687 are without mask. Then we covered 145512 images of 1, 45,537 dataset into Grayscale. In which 73264 were with mask, 66734 were misplaced mask, 5514 were without mask. For train purpose we took 124472 images. Of which 67440 were with mask, 53204 were misplaced mask and 3828 were without mask. For test purpose we took 21040 images. Where 5824 were with mask, 13530 were misplaced mask and 1686 were without mask.

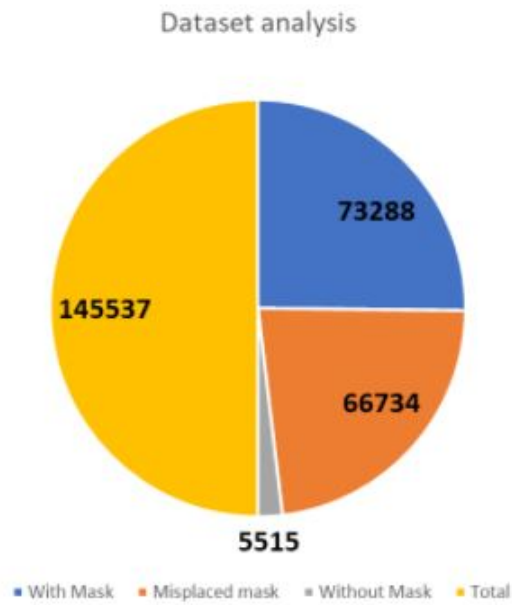


Figure 3.6: Statistical analysis of the colored image data sets

Dataset analysis of color image for train purpose

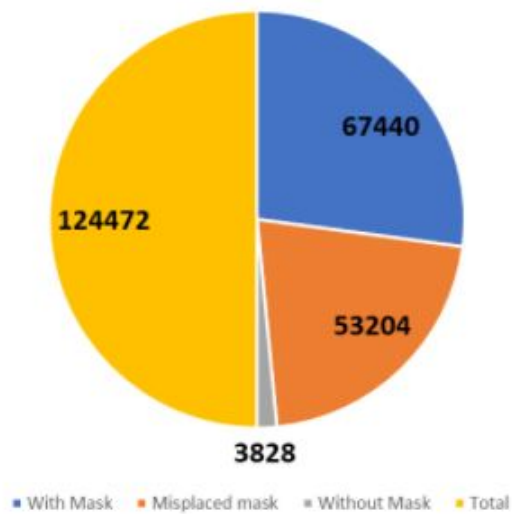


Figure 3.7: Statistical analysis of the colored train image data sets

Dataset analysis of color image for test purpose

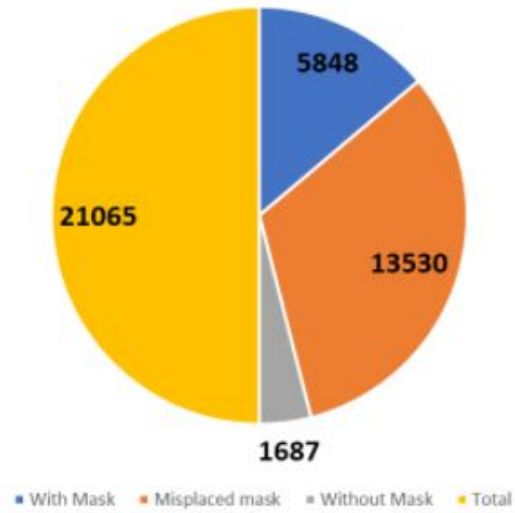


Figure 3.8: Statistical analysis of the colored test image data sets

Dataset analysis of grayscale image

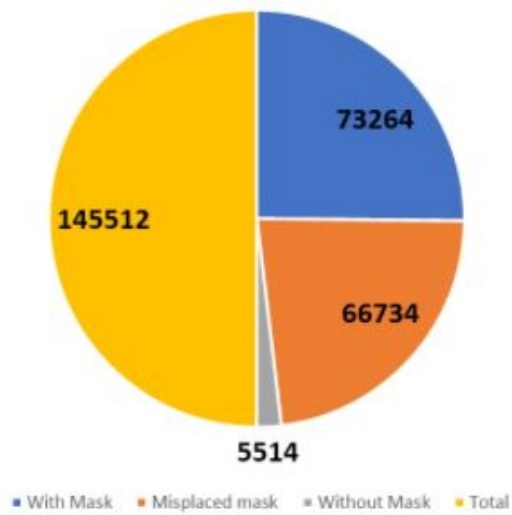


Figure 3.9: Statistical analysis of the Gray Scale image data sets

Dataset analysis of grayscale image for train purpose

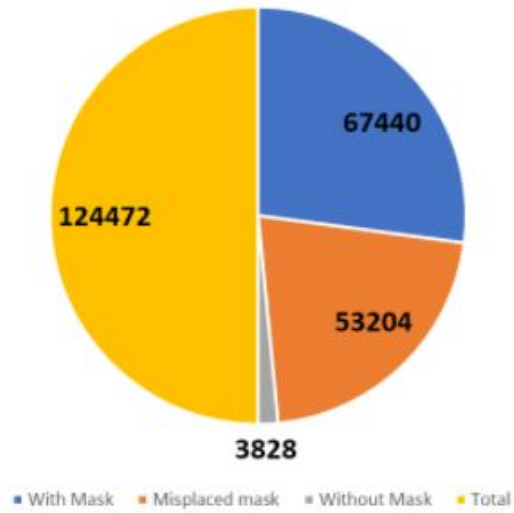


Figure 3.10: Statistical analysis of the Gray Scale Train image data sets

Dataset analysis of grayscale image for test purpose

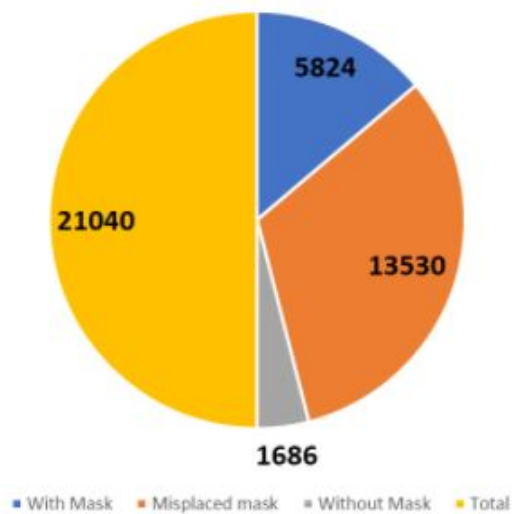


Figure 3.11: Statistical analysis of the Gray Scale Test image data sets

# Chapter 4

## Methodology

In order to recognize face masks, we have trained different CNN architectures, such as Densenet169 and VGG19, among others. For the purpose of making the program more functional, we separated our datasets into three classes. In our dataset of 145,000 photos, 50 percent of the images have masks, 45.85 percent of the images have masks that are misplaced, and the remaining images have no masks. We have converted the same dataset into gray scale images too in order to obtain the efficiency. All of the photographs were formed in order to adjust the transfer learning model, which had an input size of 100x100 pixels on it. Besides, we have augmented our dataset in order to increase the performance and results of our CNN model. After implementing the models, we saved the model and loaded it from another program. With respect to that saved model, we have tested video of face masks.

### 4.1 Convolutional Neural Network

The full form of CNN is Convolutional Neural Network. It is designed for processing pixel data. CNN is being used for image processing and recognition. It is a type of artificial neural network. CNN is a powerful image processing AI that uses deep learning. CNN is being used to do descriptive and generative tasks. For image and video recognition CNN often uses machine vision. In CNN a system is being used which reduces the requirement of processing. There are many layers in a CNN. Which are an input layer, an output layer and a hidden layer. The hidden layer is consist of convolutional layers normalization layer, a fully connected layer and a pooling layer. This system is efficient as it removes the limitations which increase the efficiency of image processing and it's simple to train limited image processing.[8]According to Keiron O'Shea and Ryan Nash, CNNs are made up of neurons that learn to be the best they can be.[4] Each neuron keeps getting feedback and performing a task. There are three dimensions to the CNN: the input's spatial dimension (height and width), as well as its depth. The neurons in the CNN are organized in these three dimensions. A CNN is made up of three layers: a convolutional layer, a pooling layer, and a fully connected layer. The convolutional layer determines the output of neurons linked to particular sections of the input volume by calculating the scalar product of their weights and the region associated to the input volume. The input, a vectorized picture, is processed through a kernel or filter, a two-dimensional array of weights. With the input data and kernel, a dot product is done. After applying the kernel to the whole picture in a methodical manner, a two-dimensional array called

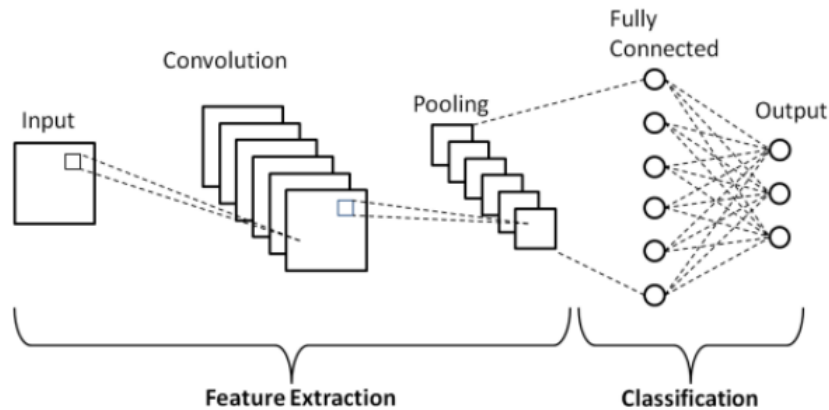


Figure 4.1: Convolutional Neural Network Architecture

the feature map is formed. Activation functions such as Relu are used to "excite" the kernel when a certain characteristic occurs at a particular spatial location in the input. The activation map is sent via the convolutional layer and then through the pooling layer. Pooling is used to reduce the dimensionality and complexity of a design. Fully linked layers contain neurons that are directly connected to the neurons in the two adjacent layers without even being connected to any other layers.

## 4.2 Transfer Learning

Transfer learning occurs when previously developed models are applied to a new challenge or situation. Transfer learning is not a subset of machine learning algorithms; rather, it is a strategy or approach for training models.[26] The knowledge gained during earlier training is retrieved to assist with the performance of a new activity. The new job will be tied to the previously learned task in some way, such as categorizing objects inside a certain file type. Typically, the trained model requires a high degree of generalisation in order effectively adapt to new unseen data. Transfer learning eliminates the need to start from beginning every time a new task arises. Transfer learning saves both time and resources since train the new machine learning algorithm may be time-consuming and resource-intensive. Large datasets must also be meticulously labeled, which takes a long time. Organizations face a lot of unlabeled data, especially when training a ml algorithm on a large amount of data. It is possible to train an algorithm on a labeled dataset, then apply it to a job that requires unlabeled data. Generalization is an essential component of transfer learning. This implies that only information that can be applied to a different model under various circumstances is passed forward. Transfer learning models will be more generalized rather than bound to a specific training dataset. It is possible to use models generated in this manner in a variety of contexts and datasets.

For the purpose of classifying photos, we're use transfer learning in this work. Using labelled data, a machine learning model may be taught to recognize and classify the object in a picture. Using transfer learning, the model may be used to recognize another specific subject in a batch of photos. The model's basic components will be left untouched, saving time and money. Models that identify an object's edge might include these components of its model. Retraining a new model is unneces-



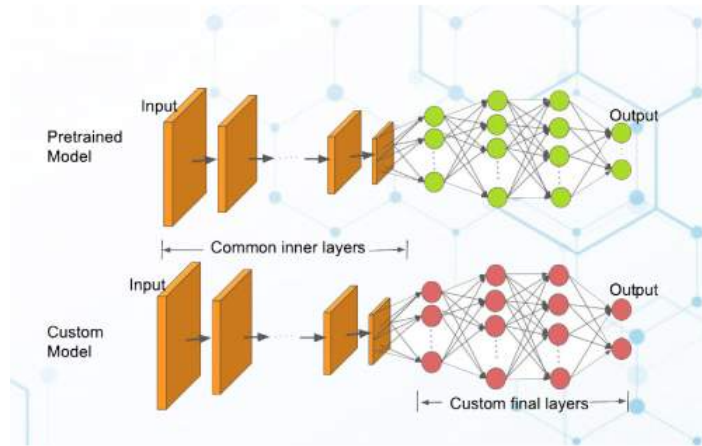


Figure 4.2: Transfer Learning

sary thanks to the transfer of this information. Inductive learning is a term that refers to models that use deep learning.[26] Algorithms based on inductive learning are designed such that they can learn from a large number of training samples. In classification, for example, the model learns a mapping between the input characteristics and the labels for the various classes. It is necessary for such a learner to make assumptions about the distribution of the training data in order for it to perform effectively on unseen data. Inductive bias is a term used to describe these kinds of presuppositions. The hypothesis space it confines to and the search procedure within the hypothesis space may all be used to describe the inductive bias or assumptions. In other words, these biases have an effect on how and what the model learns for the specific task and domain in question. The layered design of deep learning models and systems allows them to learn a variety of characteristics at different levels. For supervised learning, these layers are joined to one final layer (typically a fully connected layer) to produce the final output. Using a pre-trained network (like VGG) as a fixed feature extractor without its final layer is possible with this tiered design. In the last several years, deep learning have made significant progress. Due to this, we have been able to take on challenging situations and produce remarkable outcomes. Deep learning systems, on the other hand, take substantially more time and data to train than typical ML systems. Some of the most advanced deep learning networks have been built and tested in a wide range of industries.

It is possible to fine-tune the architecture of deep neural networks by manipulating a wide range of hyperparameters. As previously noted, the earliest layers of the neural network have been shown to capture general information, whilst the subsequent ones focus more on the job at hand. Using a face-recognition issue as an example, the following graphic illustrates how the network's lower layers learn highly general characteristics and its upper levels learn features that are task-specific in nature. It is possible to freeze (fix weights) specific layers while retraining or to fine-tune the remainder of them based on this new information. In this situation, we begin our retraining process with an understanding of the network's general architecture and

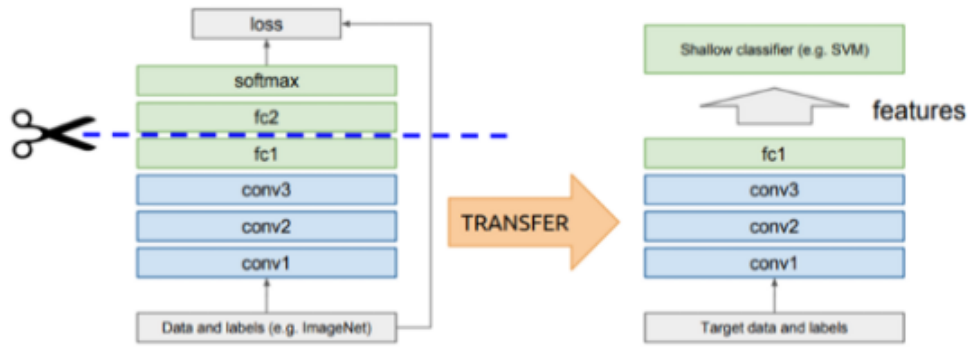


Figure 4.3: Transfer Learning with Pre-trained Deep Learning Models as Feature Extractors

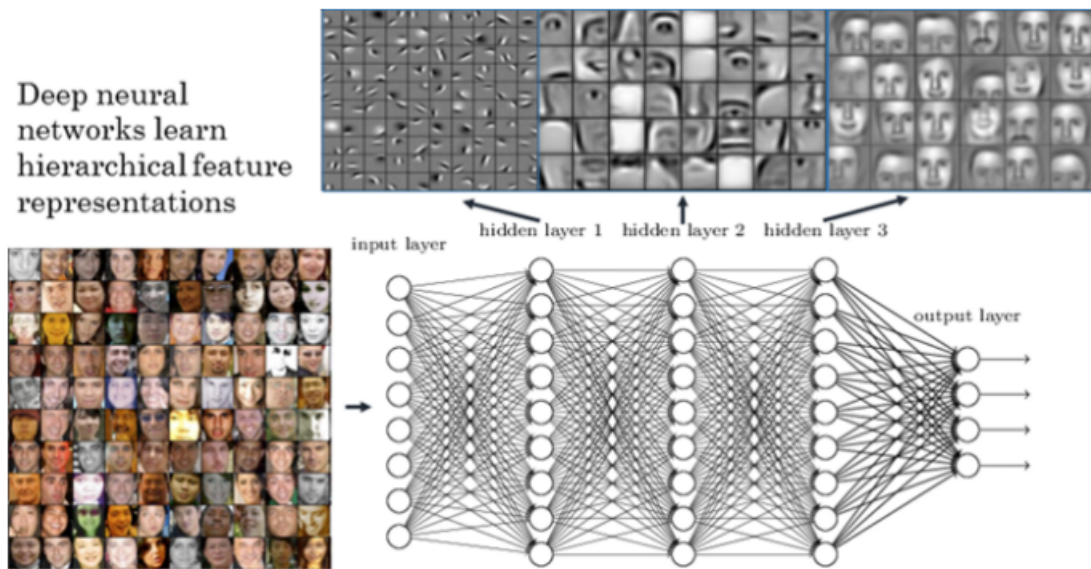


Figure 4.4: Fine Tuning Off-the-shelf Pre-trained Models

the states that exist inside it. As a result, we are able to workout less and get greater results.[9]

## 4.3 Machine Learning

In 1952, Arthur Samuel, an IBM computer scientist and a founder in artificial intelligence and computer gaming, created the term "machine learning.". Machine learning is indeed the analysis of computer algorithms that can learn and develop on their own via experience and data.[29] It is considered to be an element of artificial intelligence. The construction of computer applications that can access data and utilize it to learn for themselves is the goal of machine learning research and development. Our modern life is surrounded by the blessing of machine learning. From medical diagnosis to industrial factories, there are numerous instances of machine learning in our daily life. In the field of machine learning, the detection of faces is an extremely essential component. It is being utilized in a variety of fields, including digital secu-

rity, smart advertising, the medical field, and many more.[18] The objective of the study "Face Mask Detection Using AI" is to propose a method that can recognize a picture of a human and determine the likelihood that the individual is wearing a mask or not. In our case, the deep learning approach was utilized to detect a mask on a person's face. In a pandemic crisis like COVID-19, detection of face masks has become highly crucial all over the globe. Numerous health systems around the world are facing unusual problems as a result of the outbreak of a new coronavirus (COVID-19). The identification of face masks in the workplace is an unparalleled advantage of machine learning when it comes to maintaining a healthy working environment. The method has become more accurate as a result of the machine being trained with hundreds of thousands of photographs. The classification of photos into three groups: images with mask, images without mask, and images with misplaced mask has made the process more practical for the current situation. The extent of security may be increased even further if human beings' access to certain areas is restricted based on the form in which they are wearing a mask.

## 4.4 Save and Load Function

The progress of the model may be stored both during and after training. This implies that a model may pick up where it left off and avoid having to go through a lengthy training process. Saving also allows you to share your model with others, allowing them to duplicate your work. In order to publish research models and approaches, the majority of machine learning practitioners release both the code used to generate the model and the training weights, or parameters, that were used to develop the model. Sharing this data allows others to better understand how the model works and to experiment with it using fresh data. In this research, we are using entire model technique for load and save function. [17]

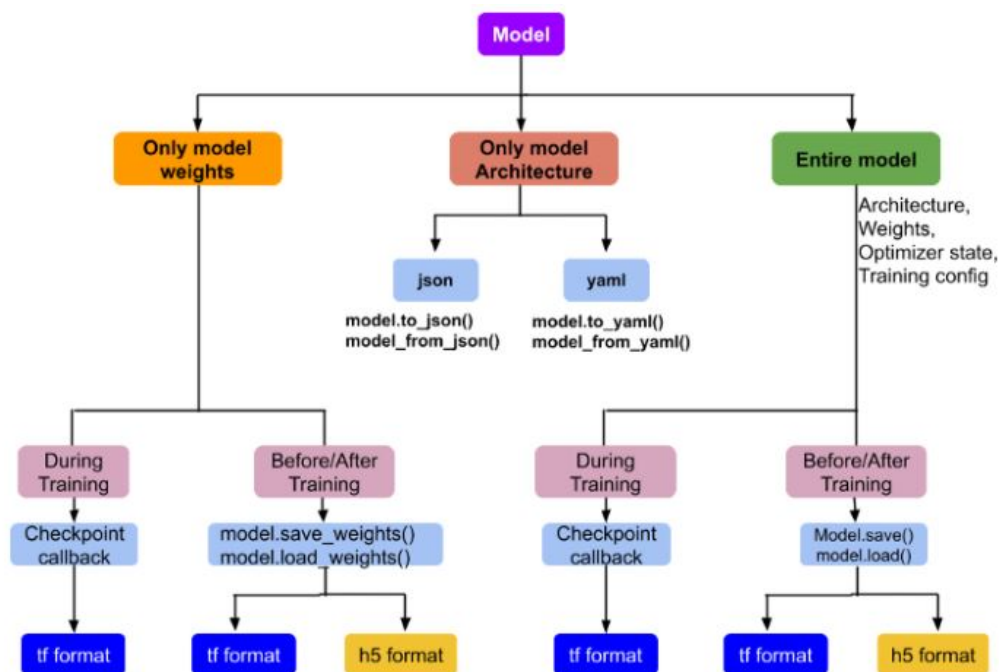


Figure 4.5: Classification of save and load function

## 4.5 Densenet-169

DenseNet is one of the new addition in the neural networks used for the recognition of visual objects. DenseNet169 is a model of the DenseNet group. DenseNet group is being designed to perform image classification. DenseNet169 is larger than the rest others of the DenseNet group. Mostly in DenseNet, all the images are being trained on The ImageNet image database but here we have trained the model and saved it and tested it by loading our saved model instead of ImageNet. Here the output of the previous layer gets concatenated with the future layer DenseNet. DenseNet was designed for declining the accuracy in a high-level neural network which is caused by the vanishing gradient. As there is a long path that exists between the input and output layer and the information gets vanished even before reaching its destination. DenseNet belongs to the category of the classic networks.[21] According to recent stats, convolutional layers can be more efficient and accurate. If they have a shorter connection between layers close to the input and close to outputs as well. And can be substantially deeper also. Here DenseNet has been used to connect each layer to every layer which is in a feed-forward fashion. Usually, traditional convolutional networks have L layers. And L connection exists between L layers. Which means one connection between each layer and its subsequent layers. Here we

Layers	Output Size	DenseNet 169
Convolution	112×112	7×7 conv, stride 2
Pooling	56×56	3×3 max pool, stride 2
Dense Block (1)	56×56	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$
Transition Layer (1)	56×56	1×1 conv
	28×28	2×2 average pool, stride 2
Dense Block (2)	28×28	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$
Transition Layer (2)	28×28	1×1 conv
	14×14	2×2 average pool, stride 2
Dense Block (3)	14×14	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$
Transition Layer (3)	14×14	1×1 conv
	7×7	2×2 average pool, stride 2
Dense Block (4)	7×7	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$
Classification Layer	1×1	7×7 global average pool
	1000	1000D fully-connected, softmax

Figure 4.6: DenseNet Architecture

have  $L(L+1)/2$  direct connections in the network. And here for every layer as inputs, all the presiding layers have been used. For the input of all subsequent layers, its feature maps are being used. We get many advantages in DenseNet. They reduce the vanishing gradient problem. Feature propagation gets strengthened, feature reuse gets encouraged and it reduces the number of parameters. Our proposed architecture can be evaluated on highly competitive image recognition benchmark

ImageNet and in addition, we have used the save and load function here. Grouping of the layers are possible as explained above if there is a total similarity in feature map dimension at the time of concatenation or addition. DenseNet is being divided into DenseBlocks with a different number of filters. But within the blocks, the dimensions are the same. Batch normalization is applied by using downsampling by transition layer. Which is considered an important step in CNN. According to the increasing of the dimension of the channel the number between the DenseBlocks of the filters changes. The growth rate is being denoted by  $K$ . [7] It plays a vital role to generalize the  $l$ -th layer. The amount of information that needs to be added in every layer is being controlled by this:

$$k_l = k_0 + k * (l - 1)$$

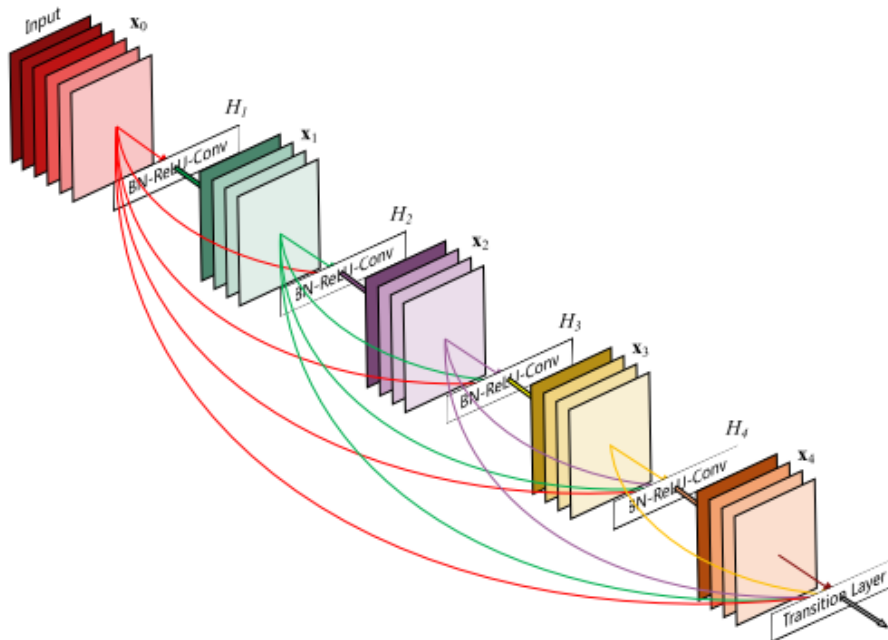


Figure 4.7: DenseNet Architecture

[13]

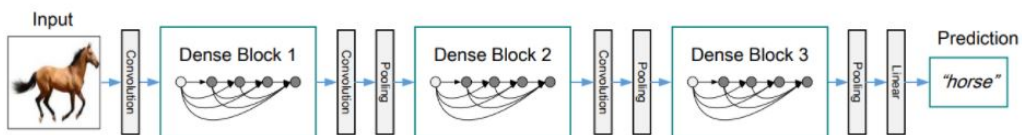


Figure 4.8: A DenseNet Architecture with 3 dense blocks

## 4.6 VGG-19

VGG19 is a 19-layer variation of the VGG model. It consists of sixteen convolution layers, three fully linked layers, five MaxPool layers, and one SoftMax layer. There are further VGG versions, such as VGG11, VGG16, and so on. VGG19 has a compute power of 19.6 billion floating point operations per second (FLOPs). In a CNN, there are three main layers: 1) the convolutional layer; 2) the pooling layer; and 3) the fully-connected layer (FC). When the FC layers are ready for the final classification, they are trained with a lot of convolutional and pooling layers in between. CNN has had a lot of success with image recognition tasks, and many people use it. Deep CNN models that have been trained can be used instead of a feature extractor. By using the network that was already trained as the feature extractor, the deep CNN can do well with small data in another field. This is because the feature extractor is already trained.[1] Gatys et al.[2] used a VGG-19 network that had been trained to recognize objects to make textures. When Huang et al.[3] started DenseBox, they used a pre-trained VGG19 network from ImageNet. DenseBox is a fully CNN framework for object detection. Li et al. [6] used the VGG-19 that was trained on ImageNet to find hierarchical cnn features for visual object tracking. When applying the VGG-19 on the ImageNet dataset as a feature extractor, a novel TranVGG19 is suggested for facemask identification in this research. In figure 4.5 shows the structure of VGG-19. Filter dimensions are 3x3 and depth 64, as indicated by "Conv3-64" denotation. Pool/2 indicates that it is a maximum pool with a 2\*2 filter.

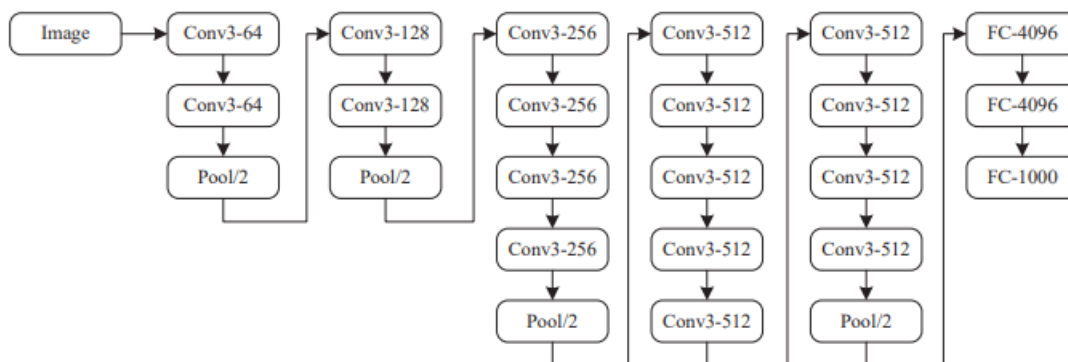


Figure 4.9: The Structure of VGG-19 Network

During the face mask detection training, the VGG-19 is frozen and transferred via transfer learning. In order to extract features, all of the convolutional and maxpool layers and first FC layer are used. After the previous feature based layers, a new FC layer with a hidden neuron count of 128 is introduced. Finally, the new softmax classifier and the new softmax classifier are trained using face mask detection data. [10]

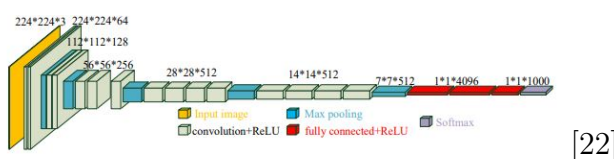


Figure 4.10: VGG19 network model [22]

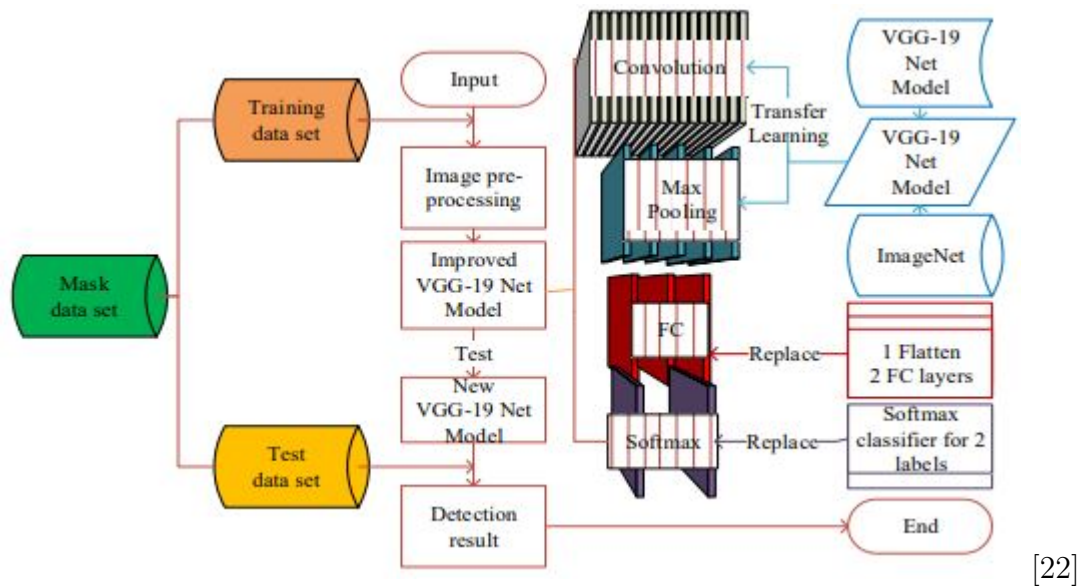


Figure 4.11: Training frame of mask detection model

## 4.7 Confusion Matrix

The Confusion Matrix is a benchmark for machine learning classification that measures how effective any programme is. For example, if you have an imbalanced amount of observations in each class, or if you have more than two classes in your dataset, just looking at classification accuracy alone might be confusing. A confusion matrix may thus provide you with a more accurate picture of what your classification model is getting right and also what sorts of mistakes it is generating. There are few elements of the confusion matrix and they are as follows:

True positive: The values that were both genuinely positive and expected to be positive.

True negative: The values which were negative and were also expected to be negative.

False positive: The values that were actually negative but were mistakenly assumed to be positive. Also called a Type I Error.

False negative: The values that were actually positive but were mistakenly assumed to be negative. Also identified as a Type II Error.

There are 3 classes in our datasets which are: with mask, without mask and misplaced mask. At the beginning, we used color images to create the confusion matrix for VGG19 and Densenet169, which we then tested. Then we converted the whole dataset into grayscale images and ran another round of tests. As a result, we achieved four confusion matrices as output.

	Positive	Negative
True	TP	TN
False	FP	FN

Table 4.1: TP, TN, FP and FN parameter definitions

# Chapter 5

## Implementation and Result Analysis

### 5.1 Densenet169

#### 5.1.1 Implementing DenseNet169 on Images

The first model that we choose to implement on our dataset was DenseNet169. Our expectation was this model would perform better for our dataset. We have implemented this both on colored images and grayscale. When we implemented this model for training and testing on color images we got accuracy of 91.47%.

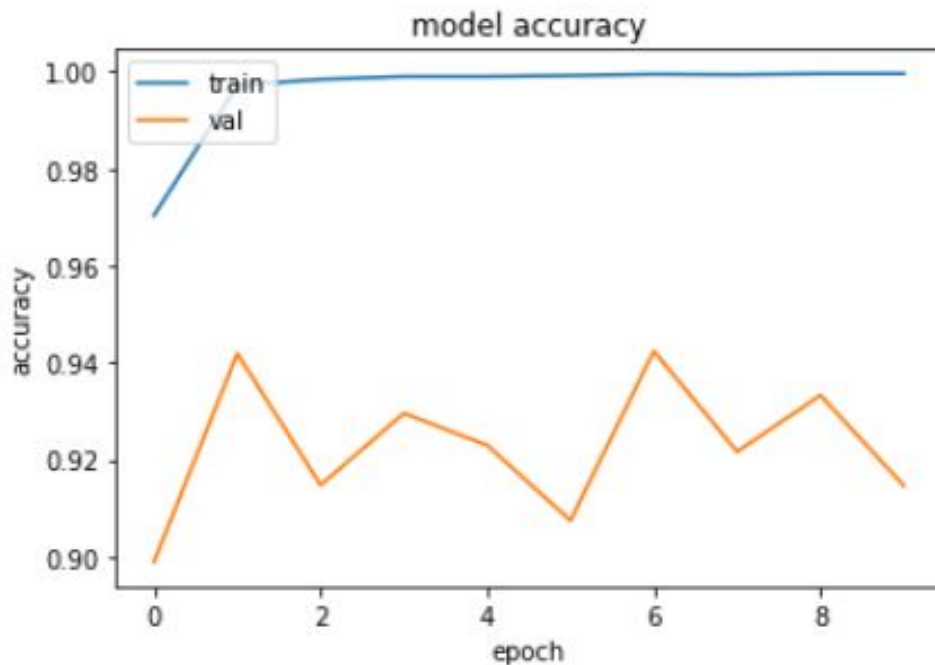


Figure 5.1: Accuracy of DenseNet169 on color images

Then we implemented on grayscale images we got accuracy of 88.83%.



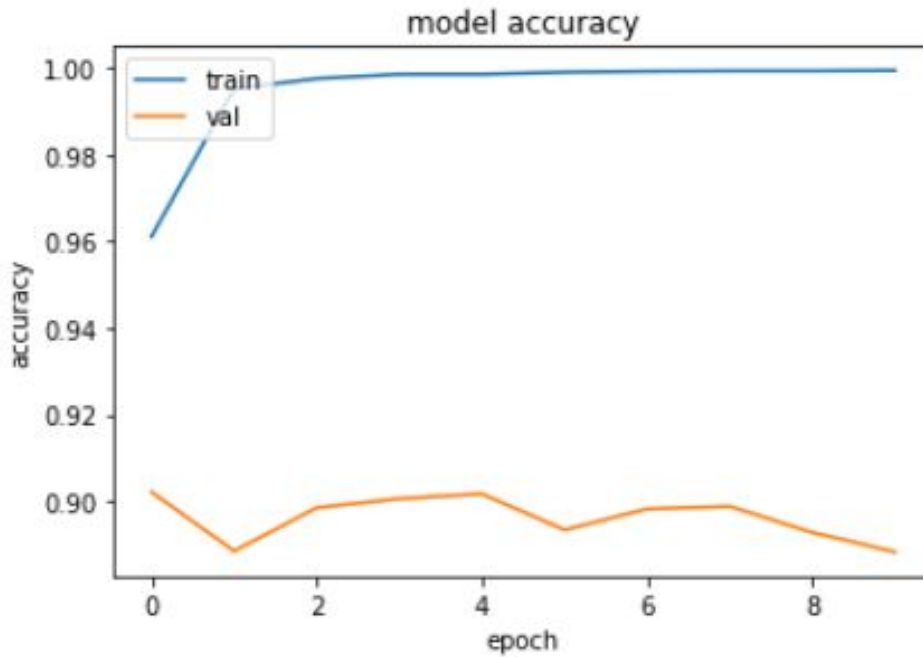


Figure 5.2: Accuracy of DenseNet169 on grayscale images

### 5.1.2 Implementing DenseNet169 on video

We covered videos into images frame by frame and implemented this model on that. Where we got accuracy of 75.36% on images of the video.

DenseNet169
75.36%

### 5.1.3 Confusion matrix for Densenet169(Color Image)

This is the confusion matrix that we obtained after implementing Densenet169 using the color image dataset following the result.

### 5.1.4 Confusion matrix for Densenet169(Grayscale Image)

This is the confusion matrix that we obtained after implementing Densenet169 using the Grayscale image dataset following the result.

### 5.1.5 Iteration of checking loss and accuracy(Densenet169)

Here we can see after implementing the model, we are about to get some results. The table shows result of 5 iteration each into 2 intervals. Which shows the loss and accuracy.

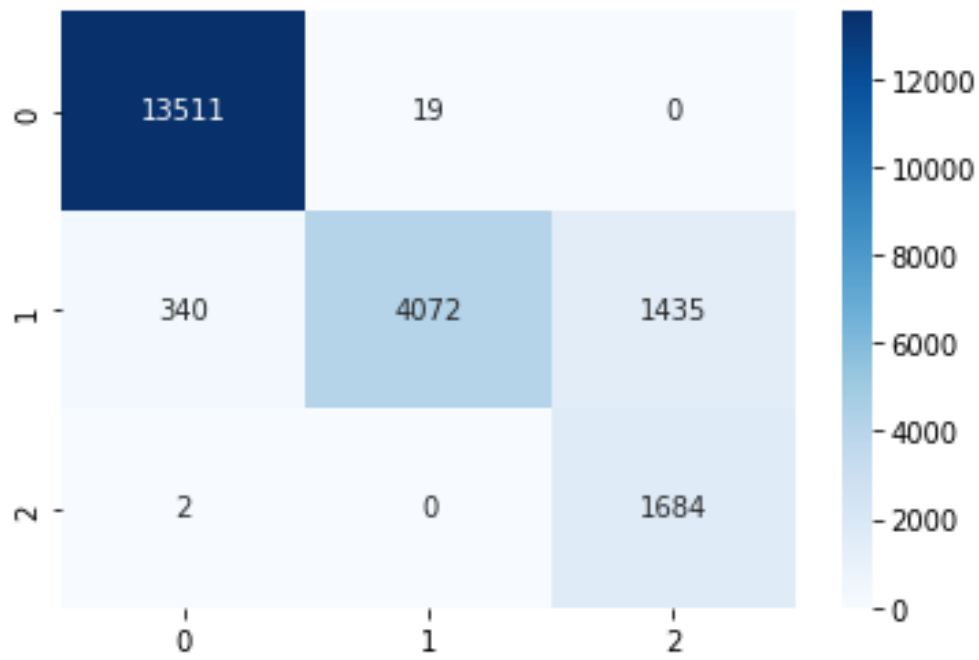


Figure 5.3: Confusion Matrix of DenseNet169 on Color image

	precision	recall	f1-score	support
0	0.98	1.00	0.99	13530
1	1.00	0.70	0.82	5847
2	0.54	1.00	0.70	1686
accuracy			0.91	21063
macro avg	0.84	0.90	0.84	21063
weighted avg	0.95	0.91	0.92	21063

Figure 5.4: DenseNet169 Color image F1-score

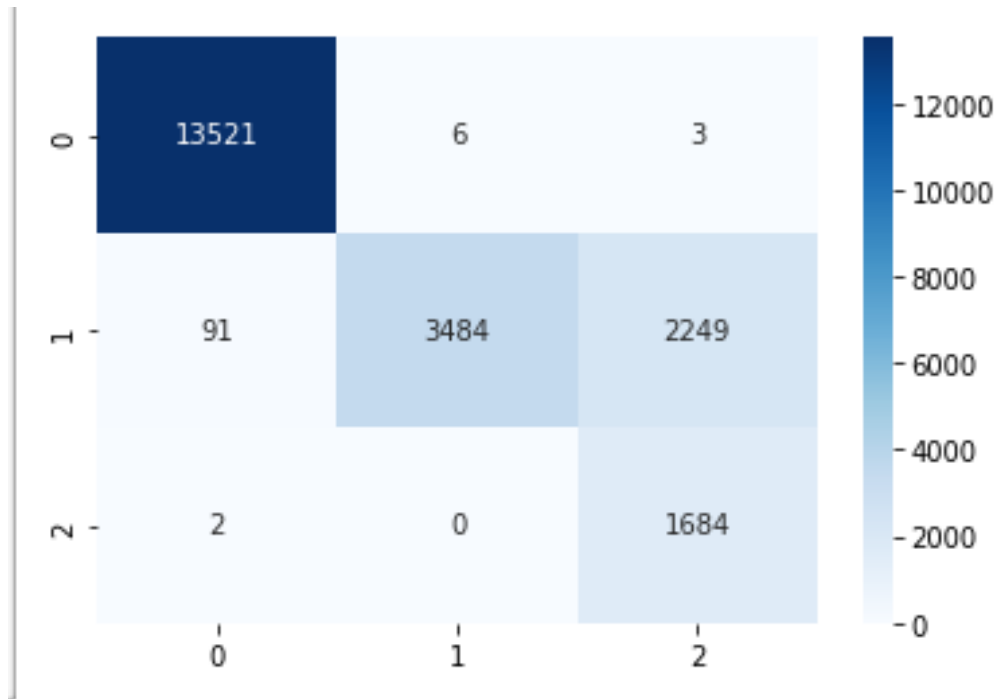


Figure 5.5: Confusion Matrix of DenseNet169 on Grayscale image

	precision	recall	f1-score	support
0	0.99	1.00	1.00	13530
1	1.00	0.60	0.75	5824
2	0.43	1.00	0.60	1686
accuracy			0.89	21040
macro avg	0.81	0.87	0.78	21040
weighted avg	0.95	0.89	0.90	21040

Figure 5.6: DenseNet169 Grayscale image F1-score

	Training loss	Categorical accuracy	Validation loss	Validation accuracy
1	0.0783	0.9704	0.4046	0.8990
2	0.0088	0.9972	0.2439	0.9419
3	0.0052	0.9984	0.4736	0.9148
4	0.0036	0.9990	0.3601	0.9296
5	0.0028	0.9990	0.5611	0.9230
6	0.0023	0.9992	0.9526	0.9075
7	0.0019	0.9995	0.3150	0.9424
8	0.0019	0.9994	0.6688	0.9217
9	0.0015	0.9996	0.5195	0.9333
10	0.0017	0.9996	0.7371	0.9147

Table 5.1: Iteration table of DenseNet169 on color images training & testing

	Training loss	Categorical accuracy	Validation loss	Validation accuracy
1	0.1059	0.9611	0.3315	0.9021
2	0.0148	0.9952	0.5632	0.8885
3	0.0075	0.9975	0.5225	0.8985
4	0.0049	0.9985	0.6633	0.9006
5	0.0047	0.9985	0.5613	0.9017
6	0.0032	0.9990	0.9252	0.8934
7	0.0025	0.9992	0.7991	0.8982
8	0.0023	0.9993	0.7676	0.8988
9	0.0027	0.9993	1.0633	0.8928
10	0.0017	0.9994	1.4306	0.8883

Table 5.2: Iteration table of DenseNet169 on gray scale images training & testing

## 5.2 VGG19

### 5.2.1 Implementing VGG19 on Images

The second model that we choose to implement on our dataset was VGG19. Our expectation was this model would perform better for our dataset. We have implemented this both on colored images and grayscale. When we implemented this model for training and testing on color images we got accuracy of 88.52% Then we

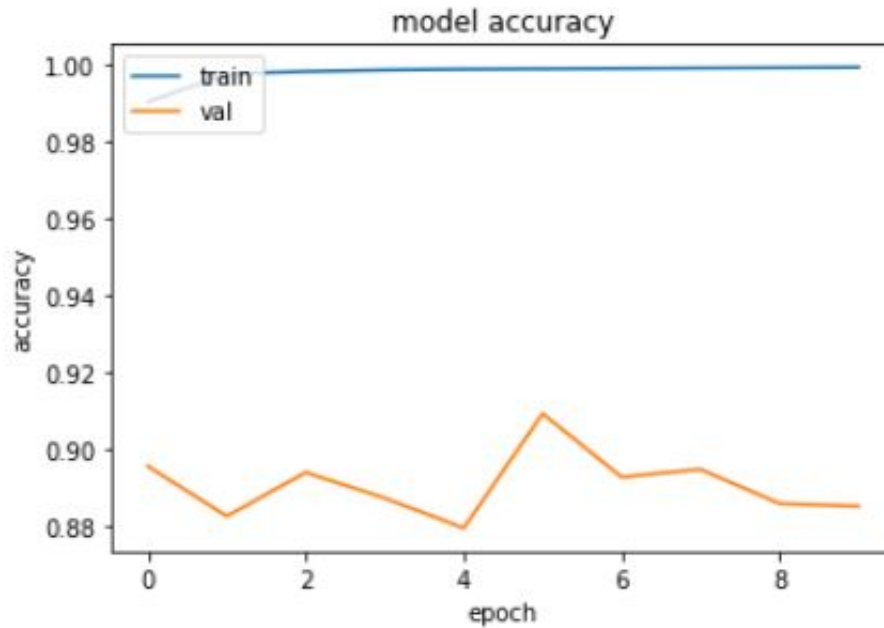


Figure 5.7: Accuracy of VGG19 on color images

implemented on grayscale images we got accuracy of 92.44%.

### 5.2.2 Implementing VGG19 on video

We got accuracy 92.30% of on grayscale converted images of the video.

VGG19
92.30%

### 5.2.3 Confusion matrix for VGG19(Color Image)

This is the confusion matrix that we obtained after implementing VGG19 to a color image dataset following the result.

### 5.2.4 Confusion matrix for VGG19(Grayscale Image)

This is the confusion matrix that we obtained after implementing VGG19 using the Grayscale image dataset following the result.

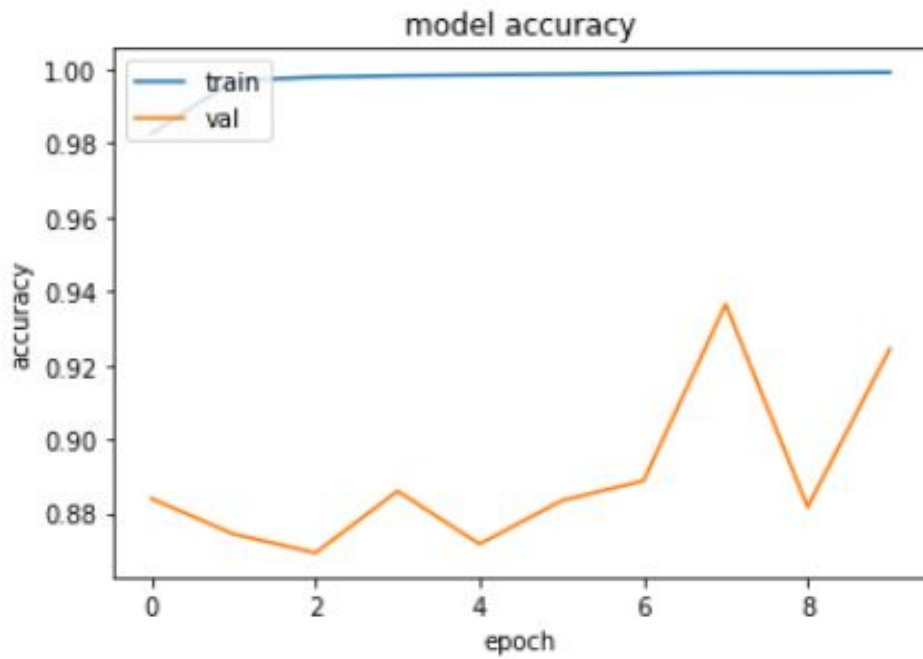


Figure 5.8: Accuracy of VGG19 on grayscale images

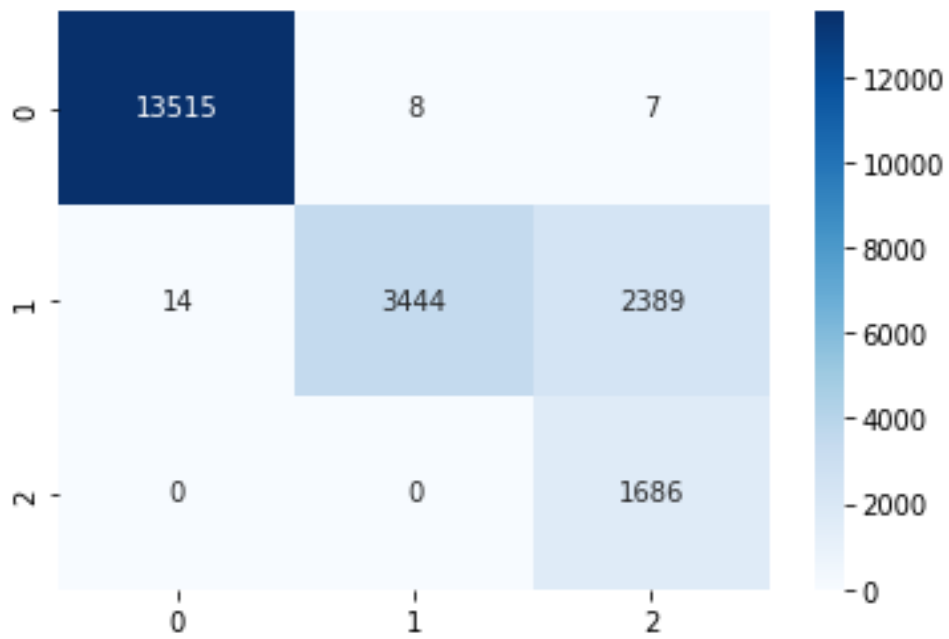


Figure 5.9: Confusion Matrix of VGG19 on Color image

	precision	recall	f1-score	support
0	1.00	1.00	1.00	13530
1	1.00	0.59	0.74	5847
2	0.41	1.00	0.58	1686
accuracy			0.89	21063
macro avg	0.80	0.86	0.77	21063
weighted avg	0.95	0.89	0.89	21063

Figure 5.10: VGG19 Color image F1-score

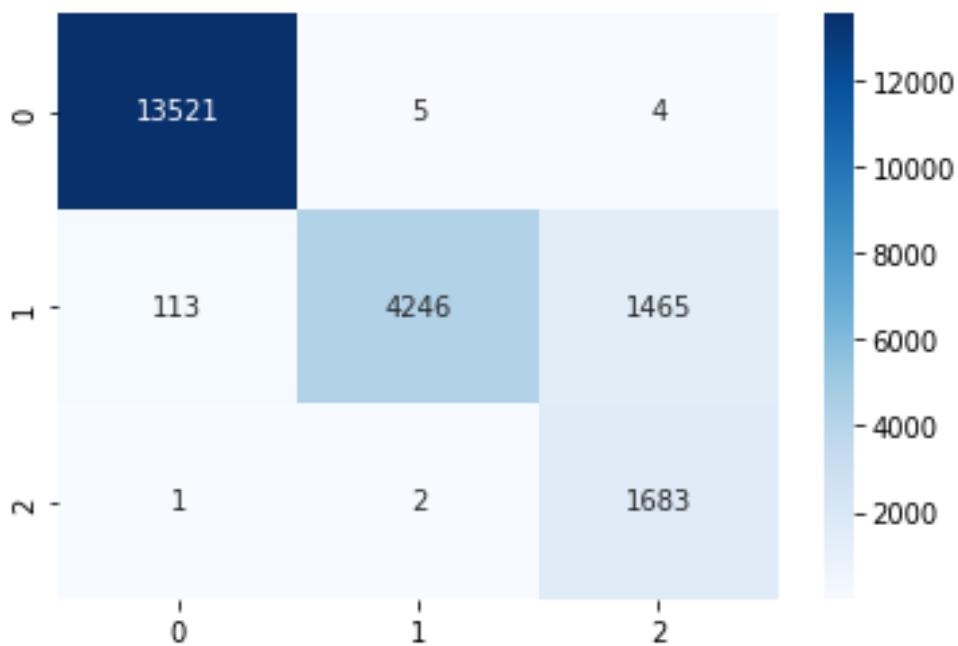


Figure 5.11: Confusion Matrix of VGG19 on Grayscale image

	precision	recall	f1-score	support
0	0.99	1.00	1.00	13530
1	1.00	0.73	0.84	5824
2	0.53	1.00	0.70	1686
accuracy			0.92	21040
macro avg	0.84	0.91	0.84	21040
weighted avg	0.96	0.92	0.93	21040

Figure 5.12: VGG19 Grayscale image F1-score

### 5.2.5 Iteration of checking loss and accuracy(VGG19)

Here we can see after implementing the model, we are about to get some results. The table shows result of 5 iteration each into 2 intervals. Which shows the loss and accuracy.

	Training loss	Categorical accuracy	Validation loss	Validation accuracy
1	0.0274	0.9905	0.3293	0.8956
2	0.0072	0.9978	0.6589	0.8826
3	0.0052	0.9984	0.5591	0.8940
4	0.0039	0.9988	0.5834	0.8873
5	0.0036	0.9990	1.0089	0.8795
6	0.0031	0.9991	0.5532	0.9093
7	0.0026	0.9992	1.0294	0.8928
8	0.0025	0.9993	0.6826	0.8948
9	0.0021	0.9994	1.0048	0.8859
10	0.0021	0.9995	1.1712	0.8852

Table 5.3: Iteration table of VGG19 on color images training & testing

	Training loss	Categorical accuracy	Validation loss	Validation accuracy
1	0.0462	0.9826	0.4579	0.8840
2	0.0094	0.9970	0.7129	0.8744
3	0.0070	0.9979	0.6888	0.8693
4	0.0054	0.9983	0.5959	0.8860
5	0.0045	0.9985	0.7505	0.8717
6	0.0041	0.9987	0.6604	0.8834
7	0.0034	0.9989	0.8409	0.8888
8	0.0030	0.9991	0.2939	0.9365
9	0.0031	0.9991	0.7637	0.8817
10	0.0025	0.9992	0.3794	0.9244

Table 5.4: Iteration table of VGG19 on gray scale images training & testing

## 5.3 Comparison of result between DenseNet169 and VGG19

From here we can see that we have trained our data set and tested them using DenseNet169 and VGG19. Where is got accuracy of 91.47% on color image dataset and 88.83% from the grayscale image dataset by DenseNet169. Where we got 88.52% accuracy of color images and 92.44% on gray scale by using VGG19. Here we can



see that in DenseNet169 the accuracy of color images is higher than the accuracy of gray scale. On the other hand, the accuracy of grayscale images are higher than the accuracy of color images in VGG19.

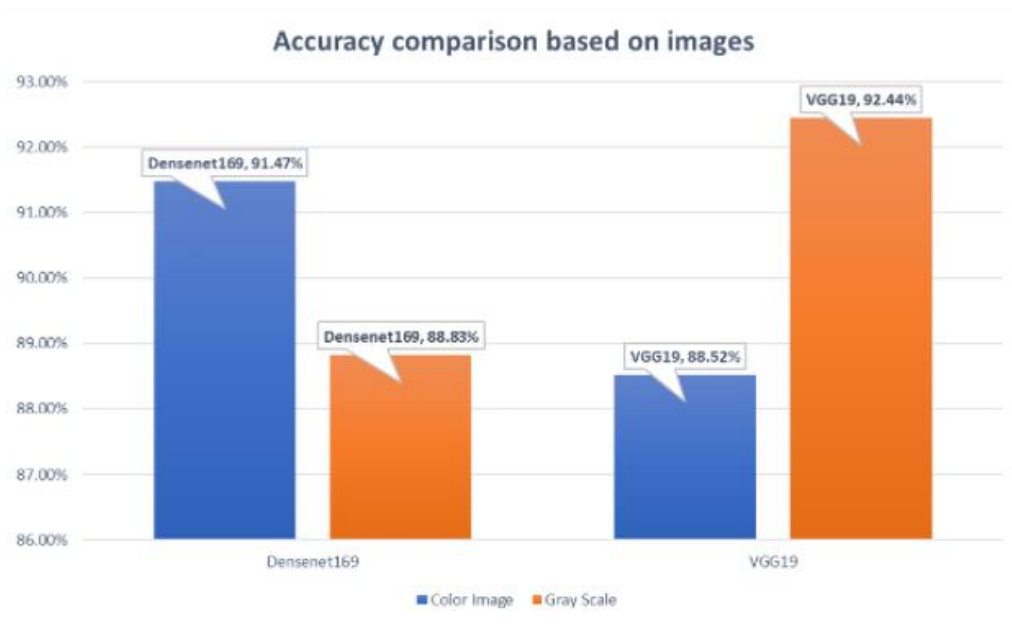


Figure 5.13: Accuracy comparison between Densenet169 and VGG19

# Chapter 6

## Conclusion and Future Work

The transmission of airborne infections is particularly quick and easy in nations with large population densities. Bangladesh has also reported a significant increase in COVID-19 cases and virus-related fatalities over the previous week, raising concerns about the possibility of the omicron variant circulating over the whole country. Consequently, wearing face masks has evolved into an inescapable circumstance rather than an elective duty in modern times. The Densenet169 method, a deep learning model, was used to identify the presence of masks on people's faces. A detailed comparison of the whole program with other detection algorithms such as VGG19 has also been performed. As a result of training the model with both color and grayscale images, this study has gained a whole wide range of new. As a result, we can compare the time complexity of training, accuracy, precision, effectiveness and other factors in detail. Because of the COVID-19 epidemic, everyone is well aware of the need to wear face masks. Putting on a face mask correctly, even in the midst of a deadly epidemic like the COVID19, is not a practice that everybody has developed. Humankind is in desperate need of a computerized system that can identify face masks and deny human beings entry to certain areas in response to this. Specifically, in this research, we have concentrated on determining if a person is correctly wearing a mask or not, which is a binary classification performed utilizing various CNN architectural configurations. Working on picture segmentation, which is able to calculate the precise location of a mask on a person's face, would be something we would want to pursue in the future. We also shall put this whole program into CCTV cameras, which may be utilized in a range of fields, including hospitals, educational institutions, factories, shopping malls, and other public locations. A hybrid of current models may also be created and tested to see how effective they are.

# Bibliography

- [1] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, “How transferable are features in deep neural networks?” *arXiv preprint arXiv:1411.1792*, 2014.
- [2] L. Gatys, A. S. Ecker, and M. Bethge, “Texture synthesis using convolutional neural networks,” *Advances in neural information processing systems*, vol. 28, pp. 262–270, 2015.
- [3] L. Huang, Y. Yang, Y. Deng, and Y. Yu, “Densebox: Unifying landmark localization with end to end object detection,” *arXiv preprint arXiv:1509.04874*, 2015.
- [4] K. O’Shea and R. Nash, “An introduction to convolutional neural networks,” *arXiv preprint arXiv:1511.08458*, 2015.
- [5] 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.
- [6] Y. Li, Y. Zhang, Y. Xu, J. Wang, and Z. Miao, “Robust scale adaptive kernel correlation filter tracker with hierarchical convolutional features,” *IEEE Signal Processing Letters*, vol. 23, no. 8, pp. 1136–1140, 2016.
- [7] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.
- [8] T. Contributor, *What is convolutional neural network? - definition from whatis.com*, Apr. 2018. [Online]. Available: <https://www.techtarget.com/searchenterpriseai/definition/convolutional-neural-network>.
- [9] D. ( Sarkar, *A comprehensive hands-on guide to transfer learning with real-world applications in deep learning*, Nov. 2018. [Online]. Available: <https://towardsdatascience.com/a-comprehensive-hands-on-guide-to-transfer-learning-with-real-world-applications-in-deep-learning-212bf3b2f27a>.
- [10] L. Wen, X. Li, X. Li, and L. Gao, “A new transfer learning based on vgg-19 network for fault diagnosis,” in *2019 IEEE 23rd international conference on computer supported cooperative work in design (CSCWD)*, IEEE, 2019, pp. 205–209.
- [11] S. K. Addagarla, G. K. Chakravarthi, and P. Anitha, “Real time multi-scale facial mask detection and classification using deep transfer learning techniques,” *International Journal*, vol. 9, no. 4, pp. 4402–4408, 2020.
- [12] J. Brownlee, *Softmax activation function with python*, Jun. 2020. [Online]. Available: <https://machinelearningmastery.com/softmax-activation-function-with-python/>.

- [13] *Densenet architecture explained with pytorch implementation from torchvision*, Aug. 2020. [Online]. Available: [https://amaarora.github.io/2020/08/02/densenets.html?fbclid=IwAR1POGNaiXS\\_-3TzT4G9oHn6t15ZAeVX71zFOs4mDgqgYkYYtU](https://amaarora.github.io/2020/08/02/densenets.html?fbclid=IwAR1POGNaiXS_-3TzT4G9oHn6t15ZAeVX71zFOs4mDgqgYkYYtU)
- [14] S. Feng, C. Shen, N. Xia, W. Song, M. Fan, and B. J. Cowling, “Rational use of face masks in the covid-19 pandemic,” *The Lancet Respiratory Medicine*, vol. 8, no. 5, pp. 434–436, 2020.
- [15] M. Inamdar and N. Mehendale, “Real-time face mask identification using face-masknet deep learning network,” *Available at SSRN 3663305*, 2020.
- [16] *Intuition of adam optimizer*, Oct. 2020. [Online]. Available: <https://www.geeksforgeeks.org/intuition-of-adam-optimizer/#:~:text=Adam%20optimizer%20involves%20a%20combination,minima%20in%20a%20faster%20pace..>
- [17] V. Janapati, *Saving and loading of keras sequential and functional models*, Oct. 2020. [Online]. Available: <https://medium.com/swlh/saving-and-loading-of-keras-sequential-and-functional-models-73ce704561f4>.
- [18] A. M. Kaur, A. J. Nicols, A. A. Lakshminarayanan, A. T. Kalafati, A. K. Rogers-Nelson, and A. M. Shealy, *Top 10 real-life examples of machine learning*, Jan. 2020. [Online]. Available: <https://bigdata-madesimple.com/top-10-real-life-examples-of-machine-learning/>.
- [19] *Novel coronavirus – china*, Jan. 2020. [Online]. Available: <https://www.who.int/csr/don/12-january-2020-novel-coronavirus-china/en/>.
- [20] B. Qin and D. Li, “Identifying facemask-wearing condition using image super-resolution with classification network to prevent covid-19,” *Sensors*, vol. 20, no. 18, p. 5236, 2020.
- [21] G. Singhal, *Gaurav singhal*, May 2020. [Online]. Available: <https://www.pluralsight.com/guides/introduction-to-densenet-with-tensorflow?fbclid=IwAR2ZX2Wlxmi3KYvi0wm7O9iog8V1BVIr1X13MMuMe7qREgp5qWxCOruiPkA>.
- [22] J. Xiao, J. Wang, S. Cao, and B. Li, “Application of a novel and improved vgg-19 network in the detection of workers wearing masks,” in *Journal of Physics: Conference Series*, IOP Publishing, vol. 1518, 2020, p. 012 041.
- [23] J. Ieamsaard, S. N. Charoensook, and S. Yammen, “Deep learning-based face mask detection using yolov5,” in *2021 9th International Electrical Engineering Congress (iEECON)*, IEEE, 2021, pp. 428–431.
- [24] A. Kumar, A. Kalia, A. Sharma, and M. Kaushal, “A hybrid tiny yolo v4-spp module based improved face mask detection vision system,” *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–14, 2021.
- [25] M. Loey, G. Manogaran, M. H. N. Taha, and N. E. M. Khalifa, “A hybrid deep transfer learning model with machine learning methods for face mask detection in the era of the covid-19 pandemic,” *Measurement*, vol. 167, p. 108 288, 2021.
- [26] P. Sharma, *Transfer learning: Understanding transfer learning for deep learning*, Oct. 2021. [Online]. Available: <https://www.analyticsvidhya.com/blog/2021/10/understanding-transfer-learning-for-deep-learning/>.

- [27] S. Taneja, A. Nayyar, P. Nagrath, *et al.*, “Face mask detection using deep learning during covid-19,” in *Proceedings of Second International Conference on Computing, Communications, and Cyber-Security*, Springer, 2021, pp. 39–51.
- [28] Z. Wang, P. Wang, P. C. Louis, L. E. Wheless, and Y. Huo, “Wearmask: Fast in-browser face mask detection with serverless edge computing for covid-19,” *arXiv preprint arXiv:2101.00784*, 2021.
- [29] *When was machine learning invented?* Feb. 2021. [Online]. Available: <https://pandio.com/blog/when-was-machine-learning-invented/>.
- [30] *Who coronavirus (covid-19) dashboard.* [Online]. Available: <https://covid19.who.int/>.