Dissertation on
Detection of Face Masks Using Convolutional Neural Network

Thesis submitted towards partial fulfilment
of the requirements for the degree of

Master in Multimedia Development

Submitted by
Meghna Paul

EXAMINATION ROLL NO.: M4MMD24007
UNIVERSITY REGISTRATION NO.: 160392 *of*
2021-2022

Under the guidance of
MR. JOYDEEP MUKHERJEE

School of Education Technology
Jadavpur University

Course affiliated to
Faculty of Engineering and Technology
Jadavpur University
Kolkata-700032
India

2024

Master in Multimedia Development
Course affiliated to
Faculty of Engineering and Technology
Jadavpur University
Kolkata, India

CERTIFICATE OF RECOMMENDATION

This is to certify that the thesis entitled **"Detection of Face Masks Using Convolutional Neural Network"** is a bonafide work carried out by **Meghna Paul** under our supervision and guidance for partial fulfillment of the requirements for the degree of Master in Multimedia Development in School of Education Technology, during the academic session 2023-2024.

SUPERVISOR
School of Education
Technology Jadavpur
University,
Kolkata-700 032

DIRECTOR
School of Education
Technology Jadavpur
University,
Kolkata-700 032

DEAN – FISLM
Kolkata, India
Jadavpur
University,
Kolkata-700 032

Master in Multimedia Development
Course affiliated to
Faculty of Engineering and Technology
Jadavpur University
Kolkata, India

CERTIFICATE OF APPROVAL **

This foregoing thesis is hereby approved as a credible study of an engineering subject carried out and presented in a manner satisfactory to warranty its acceptance as a prerequisite to the degree for which it has been submitted. It is understood that by this approval the undersigned do not endorse or approve any statement made or opinion expressed, or conclusion drawn therein but approve the thesis only for purpose for which it has been submitted.

Committee of final examination ----------
for evaluation of Thesis

** Only in case the thesis is approved.

DECLARATION OF ORIGINALITY AND COMPLIANCE OF ACADEMIC ETHICS

I hereby declare that this thesis contains literature survey and original research work by the undersigned candidate, as part of her **Master in Multimedia Development** studies.

All information in this document has been obtained and presented in accordance with academic rules and ethical conduct.

I also declare that, as required by this rule and conduct, I have fully cited and referenced all materials and results that are not original to this work.

NAME: **MEGHNA PAUL**

EXAMINATION ROLL NUMBER: **M4MMD24007**

THESIS TITLE: **DETECTION OF FACE MASKS USING CONVOLUTIONAL NEURAL NETWORKS**

SIGNATURE: DATE:

Acknowledgement

I feel extremely glad in presenting this Thesis at School of Education Technology, Jadavpur University, Kolkata, in the partial fulfilment of the requirements for the degree of Master in Multimedia Development.

I would like to express my sincere gratitude to my guide**, Mr. Joydeep Mukherjee**, for the continuous support of my M.E study and research, for his patience motivation, enthusiasm and immense knowledge. My guide helped me in all the time in research and writing of this thesis.

Besides my guide, I would also like to thank **Prof. (Dr.) Matangini Chattopadhyay,** Director of School of Education Technology and **Dr. Saswati Mukherjee,** Assistant Professor for their support, encouragement and timely advice.
I would also like to take this opportunity to thank all of my classmates of MMD and M.Tech IT (Courseware Engineering) Programme who motivated me to complete my research work successfully. I do wish to thank all of our department as support staff and all of those who were associated with this research contributed in some form or the others.

Lastly, I would like to acknowledge and thank my parents for their strong, unconditional support, inspiration and encouragement that they have provided me, without which I would not be able to complete this research.

With Regards,

Date:

Meghna Paul
Examination Roll No:M4MMD24007
Registration No:160392of 2021-2022
Master in Multimedia Development
School of Education Technology
Jadavpur University, Kolkata-700032

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

CNN	Convolutional Neural Network
MTCNN	Multi-Task Cascade Convolutional Neural Network
VGG	Visual Geometry Group
DL	Deep Learning
ML	Machine Learning
AI	Artificial Intelligence
ResNet	Residual Networks
TSVM	Transudative Support Vector Machine
IoT	Internet of Things
ReLu	Rectified Linear Unit
YOLO	You Only Look Once
NasNet Mobile	Neural Search Architecture Netwok
LFW	Labelled Faces in the wild
DT	Decision Tree
KNN	K-nearest neighbour
RF	Random Forest
SVM	Support Vector Machine

Executive Summary

The significance of wearing masks lies in their ability to prevent the transmission of contagious diseases, protect both individuals and communities, ensure compliance with health guidelines, boost public confidence, reduce risk in high-risk settings, facilitate international travel, address virus mutations, and prepare for future outbreaks. Masks are a fundamental tool in the collective effort to safeguard public health during pandemics and other health emergencies. The primary objective of this research is to develop a system that is efficient and accurate in detecting facemasks. By utilizing CNN, a deep learning architecture suite for face detection, image processing techniques are applied to reduce overfitting. The CNN architecture consists of multiple convolutional layers, pooling layers, and fully connected layers. Transfer learning is also employed, using a pre-trained model, to improve efficiency. The model is trained using an annotated dataset, fine-tuned with various hyperparameters, and optimized for accuracy and speed. Evaluation of matrices such as accuracy, precision, recall, and F1 score also have done to access the performance of the model.

Convolutional Neural Networks (CNN), is well -suited deep learning architecture for image recognition tasks and for this reason it is used in method. First, the images are collected from online sources. Then the images are pre-processed by resizing and converting them to grayscale. CNN requires input images to be of a specific size and in grayscale. Before feeding images into the CNN model, the images should be processed. In the CNN module, the present research worker uses two convolution layers with 32 filters and 64 filters, respectively. Additionally, include two max pooling layers and a ReLU activation function with a dropout ratio of 0.5.

In this work, a total of 7553 images are used to successfully achieve a 94.80% accuracy rate, which is higher than the accuracy achieved in the previous study [1].

Chapter 1

1. Introduction

1.1. Importance of Face Mask During COVID

Face mask detection using Convolutional Neural Network (CNN) is a deep learning-based approach for detecting whether or not a person is wearing a face mask. CNN is an artificial neural network specifically designed for image detection tasks. To train a CNN for face mask detection, a large dataset of images depicting individuals both wearing and not wearing masks is required. CNN is trained to learn the features that distinguish between a masked and unmasked face. Once trained, CNN can be used to detect faces and classify them as either with or without masks in a new image.

i. The first step is to detect all the faces in the image.
ii. Once faces have been detected, CNN is used to extract features from each face. These features are learned by CNN during training.
iii. After performing feature extraction, the images are classified into two categories: with masks and without masks. This is accomplished by passing the features through a fully connected neural network, which is trained to predict the facial class. Overall, face detection using CNN is a powerful tool that can be utilized to enhance public health, security.

1.2. Problem Statement

Detection of Face Masks Using Convolutional Neural Network

1.3. Objectives of Research

The objectives of the research are as follows:

i. To create a robust classification model aimed at accurately detecting the presence or absence of face masks, thereby contributing to the prevention of COVID-19 transmission.
ii. To propose and develop an advanced face mask detection system utilizing a convolutional neural network (CNN) model, with the objective of improving efficiency and efficacy in identifying individuals adhering to mask-wearing protocols.

1.4. Assumptions and Scopes

1.4.1. Assumptions

i. Availability of Dataset: Assuming that there is a dataset available that contains labelled images of people wearing and not wearing face masks. The dataset should be diverse, including different types of masks, various facial orientations, and diverse backgrounds.
ii. Image Quality: Assuming that the images in the dataset are of high quality, with clear and distinguishable features for both masked and unmasked faces.

1.4.2. Scopes

i. Create a system that can detect if a person is wearing a face mask in images or video frames in real-time or close to real-time.
ii. Develop and optimize a Convolutional Neural Network (CNN) model to detect face masks. Explore the possibility of utilizing pre-trained

Models such as ResNet and MobileNet are creating custom architectures.

iii. Add the functionality to send alerts or notifications to security personnel when there is a failure detection of mask.

iv. Propose a strategy to ensure the system can handle increased loads by considering the number of concurrent users.

v. Determine any specific hardware requirements, such as GPU acceleration, to achieve real-time performance.

vi. If applicable, integrate the face mask detection system with other security or access control systems.

1.5. Organization of Thesis

i. Chapter1: Covers introduction, problem statement, objectives, assumption and scope.
ii. Chapter2: Literature survey presents a review of related work on face mask detection, using CNN and relevant image processing techniques.
iii. Chapter3: Presents background study of the research. It is about Deep Learning, Machine Learning, Artificial Intelligence, Convolution Neural Network.
iv. Chapter4: Describes procedural frame work on face mask detection, data preprocessing and development of model.
v. Chapter5: Describes the experimental result analysis, comparative analysis, feature plot, confusion matrix.
vi. Chapter6: Conclude with a conclusion along with some suggestions for future research work.
vii. References: List all the sources and references used in this thesis following a consistent citation.
viii. Appendix: All code snippets are given.

Chapter 2

2. Literature Survey

Pandey et al. [1] used two convolutional layers with 200 Conv2D 3x3 kernels. A flatten layer and 100 Conv2D 3x3 kernels were included in the first and second convolutional layers. Finally, a dense layer consisting of 50 neurons is linked. The output layer, which is the last layer, had two neurons: one for a masked face and the other for an unmasked face. After evaluation, the model's testing accuracy is found to be 96%.

This research work is done by Sidik and Djamal [2], which is focused on developing a face mask detection system using a CNN, which is trained on a dataset from Kaggle. The system is intended for use in public areas where monitoring mask-wearing compliance is important. The impact of the learning rate on the model's accuracy is highlighted, suggesting that a smaller learning rate led to better accuracy. The research paper likely provided more details about the CNN architecture, training process, and the specific results and findings related to face mask detection.

Kavitha et al. [3] utilized MTCNN for face mask detection due to its higher accuracy. The paper concluded that VGG16, when combined with MTCNN, offered better accuracy for face mask detection compared to InceptionV3 when used with MTCNN. In other words, it is found that the VGG16 model, in combination with the MTCNN face detection approach, provided higher accuracy in the task of detecting whether individuals are wearing face masks or not.

The performance evaluation of five different CNN-based models is discussed in this research work, which is done by Ali Mazumder et al. [4]. The models included (i) ResNet50, (ii) MobileNetV2, (iii) VGG19, (iv) VGG16, and (v) InceptionV3. Among the five models, MobileNetV2 had the highest accuracy (96.04%) and took 90 seconds for each epoch.

Khamlae et al. [5] collected a dataset of images of people wearing masks correctly, wearing masks incorrectly, and not wearing masks from Kaggle. A deep learning model is trained to classify the images into these three classes. The model achieved an accuracy of 93.3% for correctly worn masks, 82.7% for incorrectly worn masks, and 67.0% for not wearing masks.

Durga et al. [6] explained that face mask detection is accomplished using machine learning with MobileNetV2 for face mask detection. The dataset is pre-processed, the model is trained using convolutional neural networks, and the output layer is designed for binary classification to determine whether a person in an input image is wearing a mask or not. More details about the model's architecture, training process, and evaluation metrics were likely included in the paper for a comprehensive understanding of this methodology and results.

In this research work dataset is collected by Machiraju et al. [7] from Kaggle, which were used for supervised learning. A model is built using Keras and TensorFlow, with a specific focus on MobileNetV2, a lightweight neural network architecture. Additionally, Scikit-Learn is mentioned, which could be for various tasks related to traditional machine learning or data pre-processing [7].

Three deep learning methods were used by Mehedi Shamrat et al. [8] for face mask detection, including max pooling, average pooling, and the MobileNetV2 architecture. Max pooling achieved a training accuracy of 96.49% and a validation accuracy of 98.67%. Average pooling achieved a training accuracy of 95.19% and a validation accuracy of 96.23%. MobileNetV2 architecture achieved the highest accuracy, with a training accuracy of 99.72% and a validation accuracy of 99.82%.

Giri et al. [9] utilized the deep CNN learning methodology to address the classification problem. Various model architectures were utilized, such as (i) VGG 16, (ii) ResNet 50, (iii) ResNet 101, and (iv) VGG 19. On a large dataset, the model is compared based on accuracy. VGG 16 achieved an accuracy of 99.37%, which outperformed other models.

Wang and Lursinsap [10] explained transfer learning and the TSVM model. Data were collected for training, testing, and validation. The picture is then inputted into the VGG model. The VGG model is used to extract image features. These features were trained using TSVM, incorporating fuzzy concepts. This approach achieved 95.5% accuracy.

Reza et al. [11] proposed four different deep learning models, namely (i) MobileNetV2, (ii) InceptionV3, (iii) VGG16, and (iv) ResNet 50, for face mask detection on the NVIDIA Jetson TX2 and Jetson Nano. The results indicated that MobileNet outperformed the other three models in terms of speed. Moreover, the prediction ran faster on the Jetson TX2 than on the Jetson Nano, as expected. In the future, techniques such as quantization, pruning, and knowledge transfer were planned to be implemented to reduce the complexity of the model and improve its performance on IoT devices.

Handoko et al. [12] compared two algorithms, YOLO-X and MobileNetV2, for detecting face masks. The results showed that the YOLO-X output performed better than MobileNetV2, achieving 95.0%, 98.7%, and 96.1% for accuracy, average precision, recall, and F1-score, respectively. Both accuracy and speed are important when detecting face masks. Therefore, YOLO-X outperformed MobileNetV2 due to its larger training image dataset. As a result, the performance of YOLO-X is expected to improve.

Kayali et al. [13] developed a CNN-based model for face mask detection and evaluated its performance on images of different sizes. The model demonstrated relatively high accuracy in making these classifications, with the highest accuracy achieved for 128x128 pixel images.

Ibitoye and Oluwole [14] introduced a system called "Face Mask Detection" that used a Deep Convolutional model with a VGG16 architecture. Python-based integrated development environments, like Anaconda Navigator, were used to implement the algorithm and system. The system is assessed, and the findings demonstrated improved performance.

In this approach Sharma and Tomar [15] TensorFlow and Keras is used in CNN. With the prevailing pandemic of COVID-19, these systems were expected to benefit many kinds of organizations worldwide. These types of systems were especially important.

In this research work Kumar et al [16] MobileNetV2 is used for feature extraction after collecting data from Kaggle. CNN models were used for training and testing. This approach achieved an accuracy of 100% during epoch 20 and 99.99% during epoch 10.

Kayali et al. [17] proposed obtaining the collected dataset by adding face masks to the LFW datasets to detect three mask-wearing conditions: correct wearing, incorrect wearing, and no mask. Then the (i) NASNet Mobile and (ii) ResNet 50 networks were trained using the dataset under consideration. The ResNet50 outperformed the NASNet Mobile by achieving high detection accuracy.

Wang et al. [18] presented an approach to enhance the accuracy of face recognition algorithms by introducing an improved CNN architecture and optimizing various factors such as model depth, activation functions, dropout, and optimization algorithms. Data is collected online, and a substantial

improvement in face recognition accuracy is demonstrated after implementing the improvements.

Naufal et al. [28] used 3,725 images for masks and 3,828 images for no masks to compare among (i) KNN, (ii) SVM, and (iii) CNN algorithms. The performance of KNN and SVM is good, each of which had an accuracy of 81.15% and 87.21%. CNN had better accuracy performance but a longer execution time of 2,507.802 seconds.

Vijitkunsawat et al. [29] used 1376 images as data. After augmentation, it increased to 3216 images where 2572 images were used for training, 322 images for testing, and 322 images for validation. This paper presented three famous algorithms for machine learning: KNN, SVM, and MobileNet. From the experiment, it is seen that the percentage accuracy rate of the MobileNet algorithm is the highest.

Chapter 3

3. Background Study

In 2020, the people suffering from COVID-19 pandemic the covid -19 has been the most life-altering event in most of everyone's lifetimes. Wearing a facemask is one of the most important ways to combat infection. This project ensures that people are wearing masks while in public places. Machine Learning is widely used for Face Detection. In order to further improve the detection accuracy of the current face detection and recognition algorithms, this approach proposes a face detection algorithm based on improved convolutional neural networks.

3.1. Deep Learning

Deep learning is a method used in artificial intelligence (AI) that trains computers to process data similarly to how the human brain does [20]. Deep learning models are able to provide accurate analyses and predictions by recognizing intricate patterns in text, pictures, audio, and other kinds of data. Automating tasks that often need human intellect, such visual description or text-to-sound transcription, is possible with deep learning algorithms. Deep learning is a subset of machine learning, and machine learning is a subset of artificial intelligence. Figure. 1 represents relations among DL, ML, and AI.

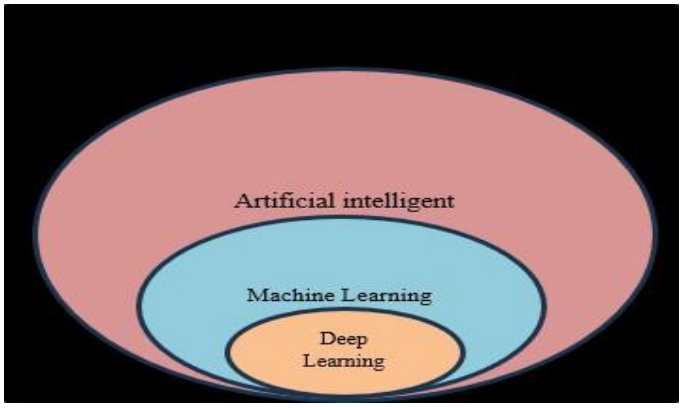

Figure 1: Relation among DL, ML, AI
Source - Self-prepared using MS word 2016

3.2. Machine Learning

Artificial intelligence (AI) systems can automatically learn from their experiences and get better over time thanks to a technique called machine learning. The development of computer programs that can access data and use it to learn for themselves is the focus of machine learning.

3.3. Artificial Intelligence

Artificial intelligence is the simulation of human intelligence processes by machines, especially computer systems. Applications of AI include expert systems, natural language processing, speech recognition, and machine vision.

3.4. Convolutional Neural Network

One of the DL-based algorithms for computer vision is CNN, which can identify and categorize features in images. CNN is capable of segmenting and classifying images. For higher-level categorization, CNN automatically extracts features by using raw pixel data from the image, trains the model, and then specifies the image's properties. A convoluted layer can apply multiple filters and each time a filter is applied across the input. A convoluted level usually follows a data pooling level to reduce dimensionality and noise from output. It uses to prediction from levels to generate a final output representing probability score to characterize the probability so that particular feature belongs to a particular class [30].

a. **Input Layer**: It contains visual data that is shown as a 3-D matrix.

b. **Convolutional Layer:** The first layer used to extract the different features from the input images. Convolution is a mathematical operation that is carried out in this layer between the input image and a filter of a specific size, 3x3. The dot product between the filter and the portions of the input

image with respect to the filter size (3x3) is taken by sliding the filter over the input image.

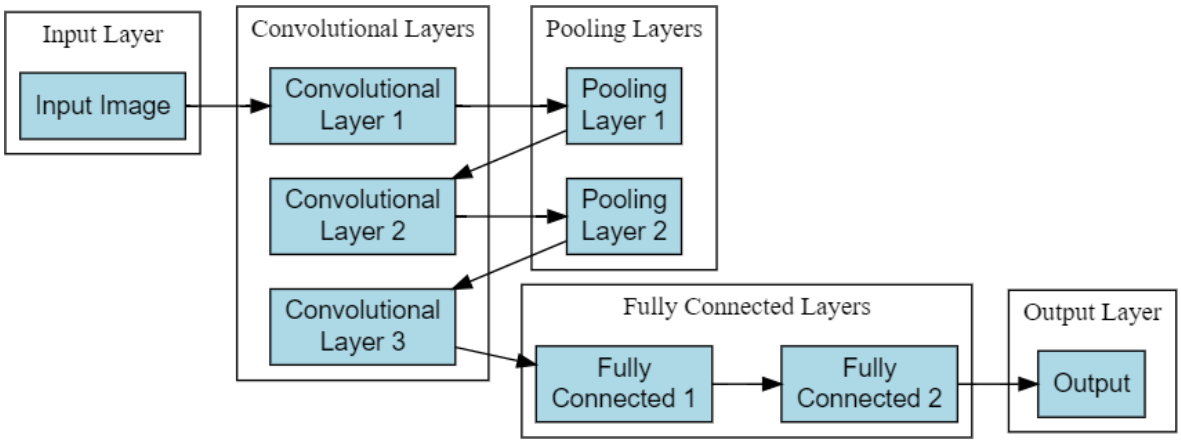

Figure 2: Schematic Diagram of Convolution Neural Network
Source - Self-prepared using MS word 2016

The output is known as the feature map, and it provides details about the image, including its corners and edges. This feature map is later fed to additional layers to teach them additional features from the input image.

Convolution operation: It is observed from the picture that the filter is applied to a portion of the image. Between the input pixel and filter, a dot product is calculated. This process repeats until the kernel sweeps over the entire image. The result is a dot product, which gives the final result in matrix form. Figure. 3 represents a convolutional operation.

Using the dot product operation between the image and kernel, convolution is performed. Each section of the image is transformed by the kernel using a shift operation to create a map feature with stride, padding, and depth parameters.

- **Kernel or Filter or Feature Detectors:** The kernel in a convolutional neural network is simply a filter used to extract features from the images.

Equation = [i-k] + 1, where ***i*** is Size of input, ***k*** is Size of kernel

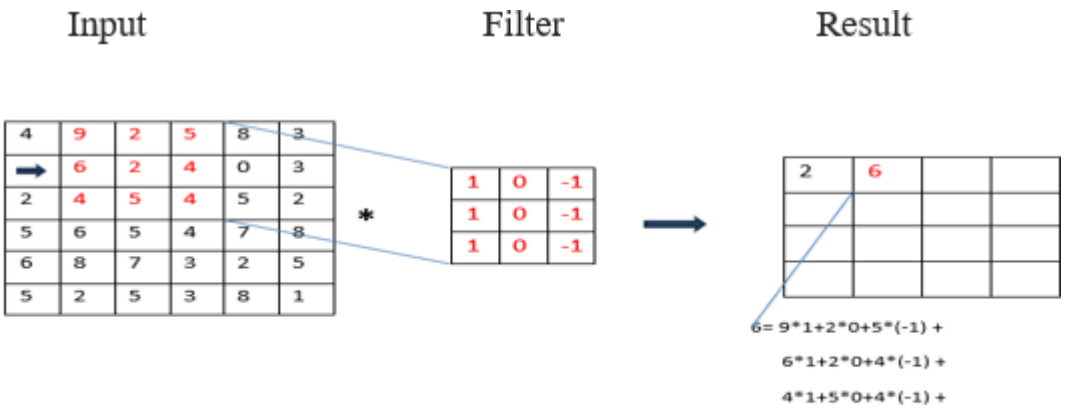

Figure 3: Convolutional operation

Where stride s=1, size f=3, padding p=0.

- **Padding:** The padding plays an important role in creating CNN [26]. When the kernel moves over the original image, it passes through the middle layer more times than the edge layers, due to which there is an overlap. To overcome this problem, a new concept is introduced called padding. It is an additional layer that can add to the borders of an image while preserving the size of the original picture. Figure 4 shows Padding operation.

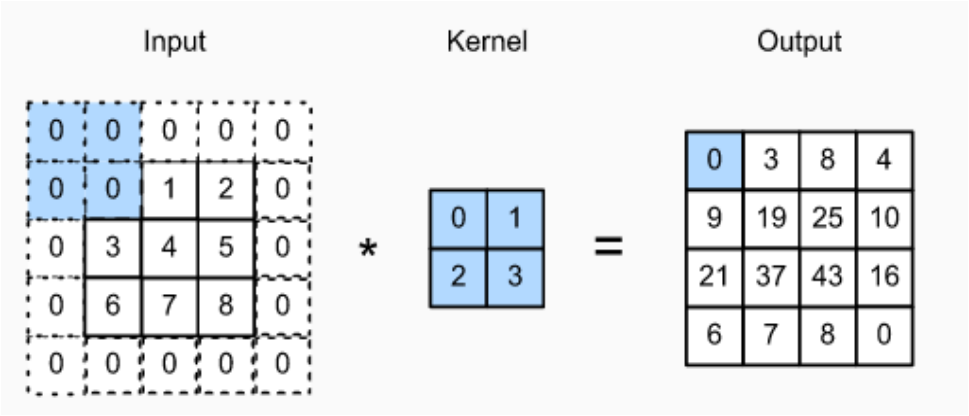

Figure 4: Padding operation

Source: TechTarget [25]

So, if an n * n matrix is convolved with an ff matrix with a padding p, then the size of the output image will be:

$$(\boldsymbol{n + 2p - f + 1}) * (\boldsymbol{n + 2p - f + 1})$$

- **Stride:** Stride is important parameter, where the array is initialized, the pixels are transferred to the input matrix [25]. The count of pixels that are activated in the input matrix is referred to as the strides. If the number of strides is 1, the filters are moved one pixel at a time. Similarly, if the number of strides is 2, the filters are shifted by 2 pixels, and so on. Strides are important because they determine how the filter interacts with the input during convolution. In other words, strides control the features that may be overlooked when flattening the image. They indicate the number of steps taken in each convolution. The Figure 5 below illustrates the operation of strides.

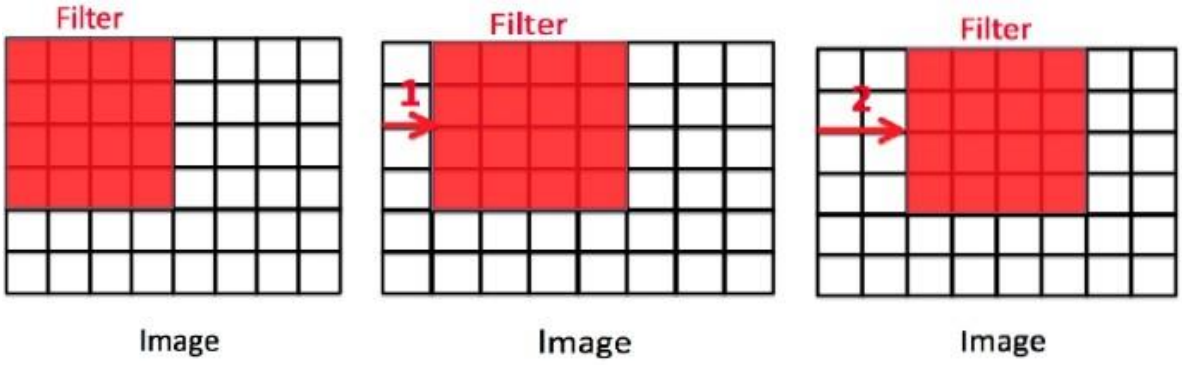

Figure 5: Stride operation

Source: TechTarget [25]

In the first image, stride = 0, in the second image: stride = 1, and in the third image: stride = 2. The size of the output image is calculated by:

$$[\{\frac{(\boldsymbol{n+2p-f+1})}{\boldsymbol{s}}\}+\mathbf{1}]*\{\frac{\boldsymbol{n+2p-f+1}}{\boldsymbol{s}}\}$$

if an n * n matrix is convolved with an f*f matrix with a padding p.

c. Pooling Layer: The pooling layer is a key component of a CNN and has a significant role in pre-processing an image [25]. It helps to reduce the

size of the image by decreasing the number of parameters, especially when the image is large. This reduction in size also leads to a decrease in pixel density, resulting in a downscaled image from the previous layers. Essentially, the pooling layer progressively reduces the spatial dimensions of the image, thereby reducing network complexity and computational requirements. Pooling is applied after the nonlinearity is applied to the feature maps. There are two types of spatial pooling.

Max Pooling

Max pooling is a rule to take the maximum of a region and help proceed with the most crucial features from the image. It is a sample-based process that transfers continuous functions into discrete counterparts. Its primary objective is to downscale an input by reducing its dimensionality and making assumptions about features contained in the sub-region that are rejected. Figure 6 represents the Max-pooling Operation.

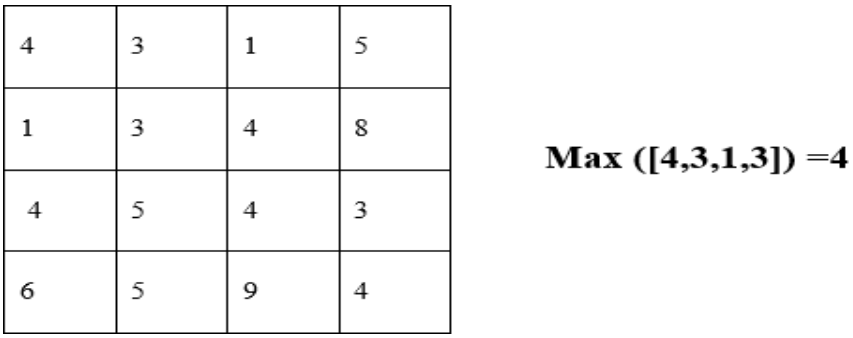

Figure 6: Max-pooling operation
Source - Self-prepared using MS word 2016

Average Pooling

Average pooling is different from Max Pooling; it retains information about less essential features. It simply downscales by dividing the input

matrix into rectangular regions and calculating the average values of each area. The following Figure 7 shows average pooling operations.

Figure 7: Average Pooling operation
Source – Self-prepared using MS word 2016

d. Flatten Layer

Once the model has learned the feature, it is ready to flatten. To flatten the final feature map and feed it into neural network for classification purposes, flatten layers are used to convert data into a 1-dimensional array for inputting the next layer. Figure 8 represents Flatten Operation.

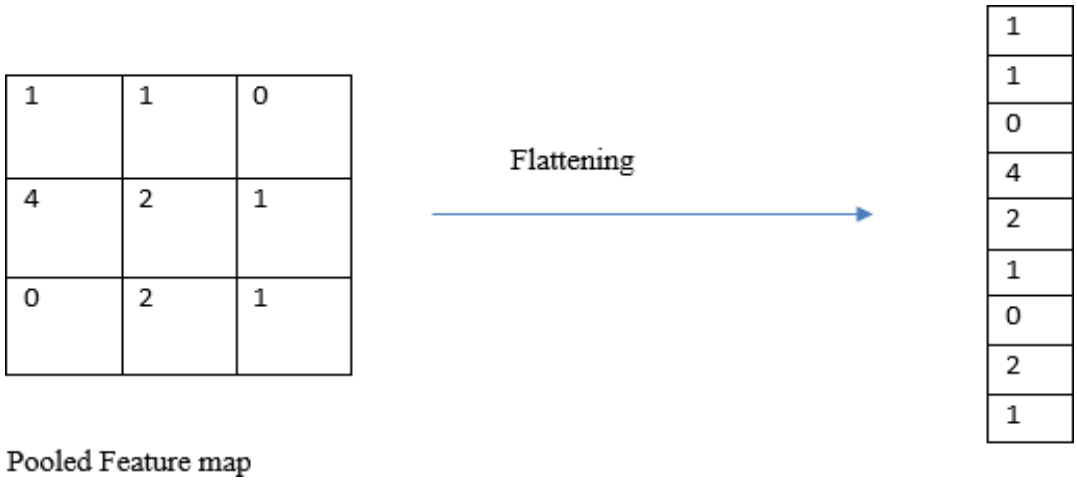

Figure 8: Flatten operation
Source - Self-prepared using MS word 2016

e. Fully Connected Layer

Fully Connected tiers make up the network's final few tiers. The output from the last pooling or convolutional layer is passed into the fully connected layer, where it is flattened before being applied.

f. Dropout layers

Dropout layer play important role of CNNs. The dropout layer acts as a mask, eliminating some neurons' contributions to the after layer while maintaining the functionality of all other neurons. Figure 9 shows the dropout operation.

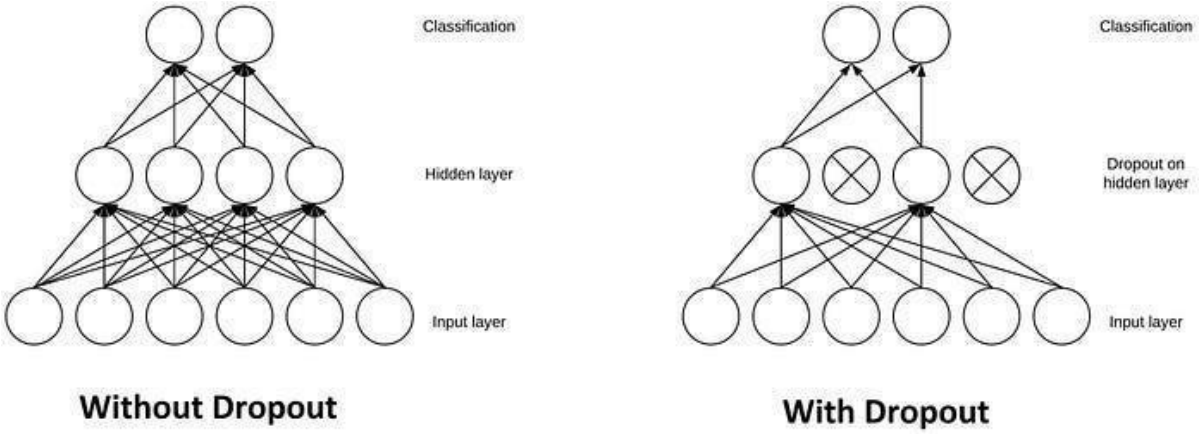

Figure 9: Dropout operation
Source: Coding ninjas [26]

g. Activation Function

Activation Function (Figure 10) decides whether a neuron should be activated or not. This means that it will decide whether the neuron’s input to the network is important or not in the process of prediction. There are several commonly used activation functions, such as the ReLU, SoftMax, tanH, and Sigmoid functions. Each of these functions has a specific use.

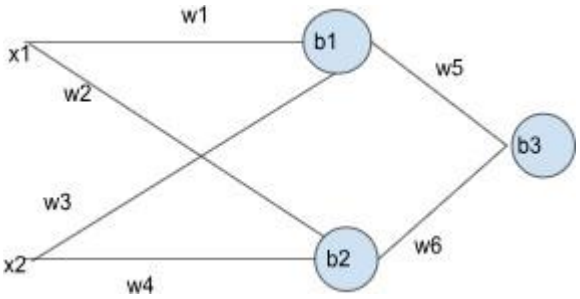

Figure 10: Activation Function
Source: Self-prepared using MS word 2016

where w is weight, x is input, and b is neurons.

- **Sigmoid -** For a binary classification in the CNN model
- **tanH -** The tanH function is very similar to the sigmoid function. The only difference is that it is symmetric around the origin. The range of values, in this case, is from -1 to 1.
- **SoftMax**- SoftMax is used in multinomial logistic regression and is often used as the last activation function of a neural network to normalize the output of a network to a probability distribution over predicted output class.
- **ReLU**- The main advantage of using the ReLU function over other activation functions is that it does not activate all the neurons at the same time.

h. Loss Function

Loss function is used for measuring difference between the predicted result and actual target.

i. Output Layer: This has one hot encoding for the label.

3.5. Python for CNN

CNN is a type of deep neural network used for image recognition and classification tasks in machine learning. Python libraries like TensorFlow, Keras, PyTorch, and Caffe provide pre-built CNN architectures and tools for building and training them on specific datasets.

- **Python**

Python is a programming language that is interpreted, object-oriented, and high-level. It is known for its dynamic semantics and is often used for rapid application development or as a scripting language to connect different components. Python has built-in data structures, dynamic typing, and dynamic binding, which make it easy to use. Its clear syntax promotes readability and makes it simple to learn, reducing the cost of program maintenance. Python also supports modules and packages, which encourage code modularity and reuse. The Python interpreter and standard library are freely available for all popular platforms [27].

- **Matplotlib and Seaborn**

Matplotlib and Seaborn are both utilized for creating visual plots. Matplotlib is more suitable for simple plots, whereas Seaborn is more suitable for complex statistical plots.

- **TensorFlow**

TensorFlow utilizes machine learning and deep learning models and algorithms for its operations.

- **Pandas**

Pandas is also a Python library; it uses Matplotlib and NumPy with less code.

- **NumPy**

NumPy can be used to perform a wide variety of mathematical operations on arrays. NumPy supports some specific scientific functions, such as linear algebra.

Chapter 4

4. Proposed Approach

This is the procedural frame work of face mask detection which is shown in Figure 11. Steps are elaborate later.

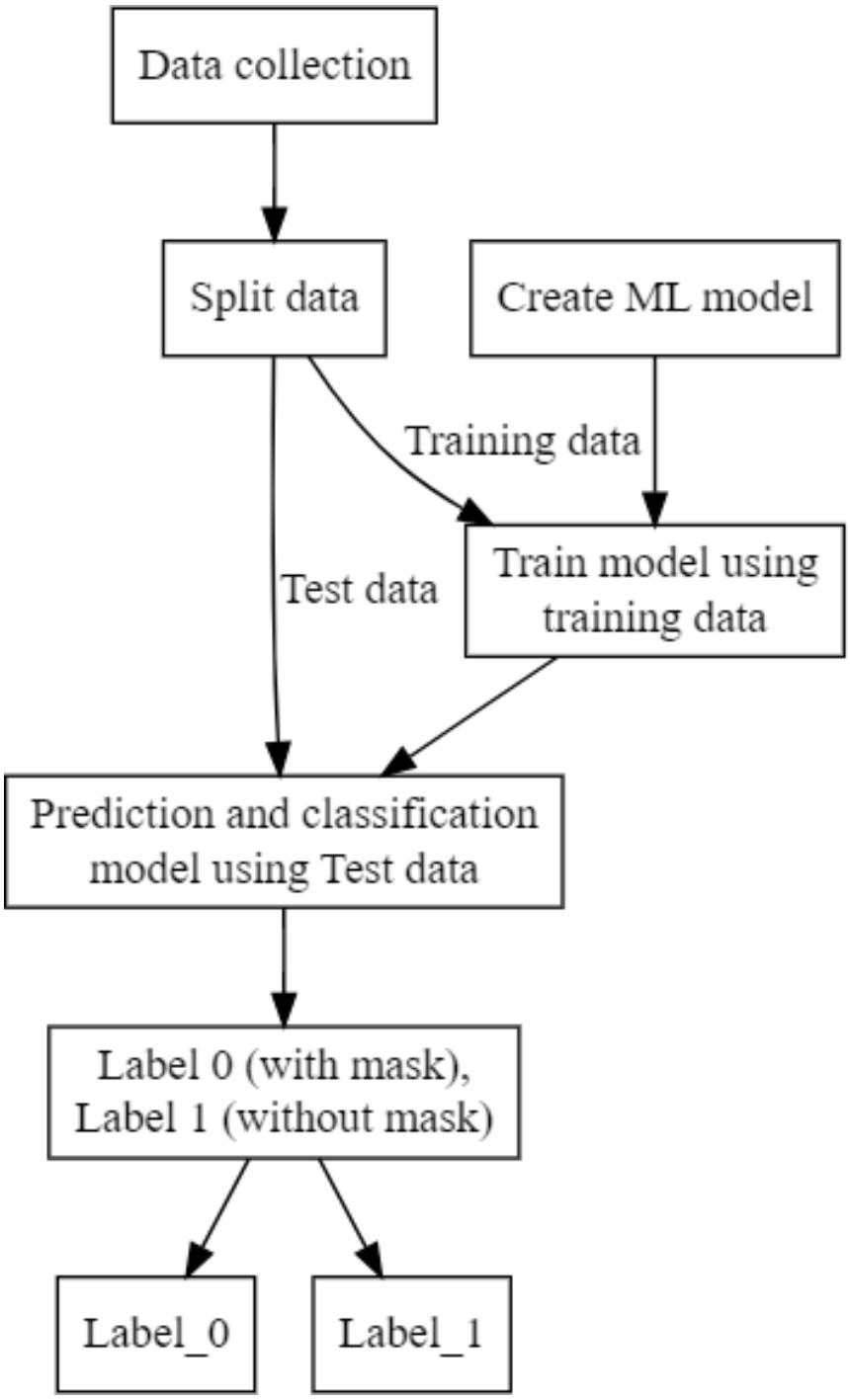

Figure 11: Procedural framework for Face mask Detection
Source: Self-prepared using MS Word 2016

4.1. Data Collection

A large set of data are collected from online [19], which is publicly available data, known as Dataset 1. The dataset consists of 7553 images. Images are divided into 5:1 ratio, for training and testing respectively. Also, present researchers use real life photos for testing which is shown in Figure12 which is known as Dataset 2. After data collection, all images are converted RGB to grayscale for KNN, DT and SVM algorithm. But for CNN there is no need to convert to grayscale. Images are labelled as label 0 for with mask and label 1 without mask.

Table 1: Dataset description

	Total images	Training	Testing
Dataset 1	7553	6294	1259
Dataset 2	1550	1291	259

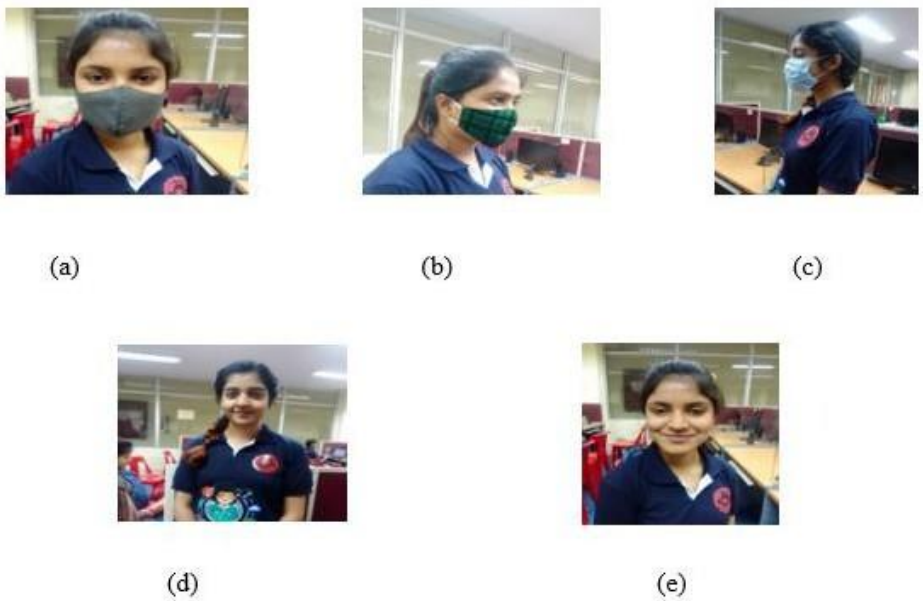

Figure 12: Sample photos for testing and training purposes (with mask [a, b, c] and without mask [d, e])

4.2. Data Pre-processing

In the initial phase of this research work, several crucial steps are undertaken to ensure that the dataset is suitable for machine learning. The original dataset consists of 7553 images, which are categorized into two groups: images with masks (Label 0) and images without masks (Label 1). The following steps are done for data preprocessing.

4.2.1. Image resizing

All images in the dataset are resized to 128x128 pixels. This standardization not only ensures a consistent input size for the machine learning model but also reduces computational complexity. A core part of CNN architecture is its convolutional filters. These filters operate on a fixed size gride of pixels in the input image, if images have different sizes, these filters wouldn't be able to slide across the image consistently.

4.2.2. Data Splitting

In order to assess the performance of the model accurately, the dataset is split into two subsets: training and testing. The training set (X train and Y train) consists of 6413 images, while the testing set (X test and Y test) contains 1140 images.

4.3. Model development

- **Input Layer:** This layer is the input layer, and it requires grayscale images with dimensions of 128x128 pixels.

- **Convolution 2D Layers:** This method utilizes three convolutional layers. The first layer applies 32 filters, the second layer applies 64 filters, and the third layer applies 128 filters.

- **Kernel Size:** Convolutional layers utilize a kernel size of 3x3 for convolution.

- **Maxpooling 2D Layers:** Two maxpooling operations are performed in this case. The maxpooling layer decreases the spatial dimensions of the feature by a factor of 3x3.

- **Padding:** Zero padding is added to the input in this method.

- **ReLu:** The ReLu activation function is employed in this study to introduce non-linearity into the network.

- **Fully connected layer:** The model consists of three fully connected layers. The initial fully connected layer has 128 neurons, the second fully connected layer has 64 neurons, and the final fully connected layer has 2 neurons which is also known as output layers.

- **Dropout Layer:** In this research, a dropout rate of 0.5 is utilized to reduce overfitting.

- **SoftMax Layer:** The SoftMax activation function is utilized in the final section. It is used to create a SoftMax layer that generates a probability distribution for the two classes.

- **Optimizer:** In this research, the "Adam" optimizer is used, which is a widely used algorithm for optimizing convergence by adjusting the learning process during training.

- **Loss:** The sparse categorical cross entropy function is used during training. It is specifically used when the SoftMax activation function is employed.

Chapter 5

5. Experimental Results and Analysis

5.1. Experimental Setup

This research uses Google Colab with a T4 GPU, 12.7GB of RAM, and Python 3.10.12

TensorFlow and Keras are built-in-libraries of python, which are used to define the CNN model. Here, the CNN model is trained with 21 epochs to achieve improved accuracy. Figure 13 shows the accuracy of the model.

```
31/31 [==============================] - 0s 7ms/step - loss: 0.0730 - accuracy: 0.9782
Test Loss = 0.07299493253231049
Test Accuracy = 0.9781704545021057
```

Figure. 13 Accuracy of the model

The model achieves an accuracy of 0.9782. It has a test dataset loss of approximately 0.073, which indicates good performance. Generally, a lower loss and higher accuracy suggest better model performance.

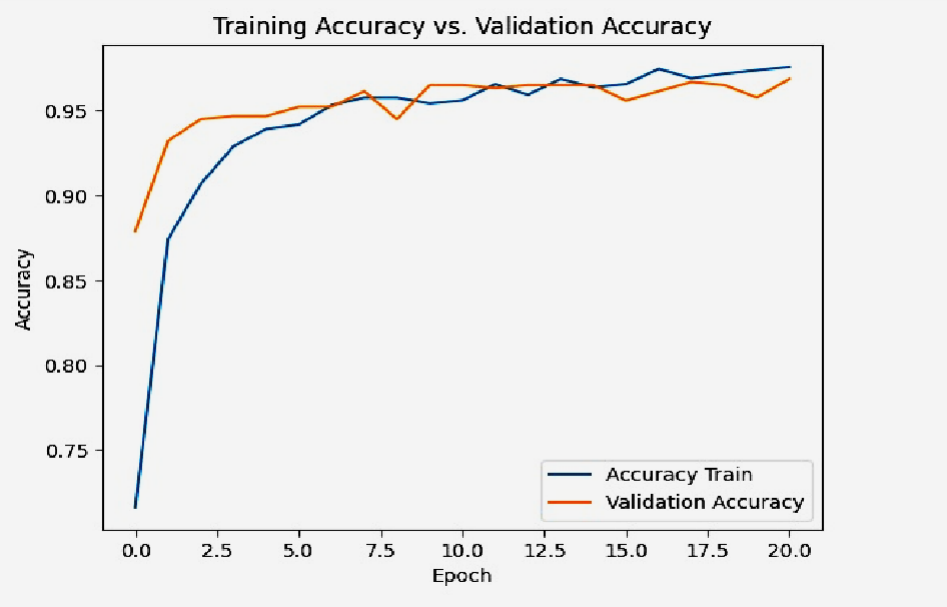

Figure.14 Accuracy of epochs graph

The graph in Figure 14 illustrates the accuracy of a face mask detection model trained with a convolutional neural network (CNN). The horizontal axis represents the epoch, indicating the number of times the model is trained on the complete training dataset. The vertical axis represents accuracy, which indicates the percentage of correctly classified images.

The validation accuracy shows how well the model is able to generalize to new data. In this case, the validation accuracy is improving over time, indicating that the model is becoming better at generalizing. However, the validation accuracy is still lower than the training accuracy, suggesting that the model is still somewhat overfitting to the training data. In summary, the graph demonstrates that the model is progressively improving its accuracy in detecting face masks. However, it is important to have continuous training of the model and evaluate it on a validation set to ensure that it is able to generalize well to new data.

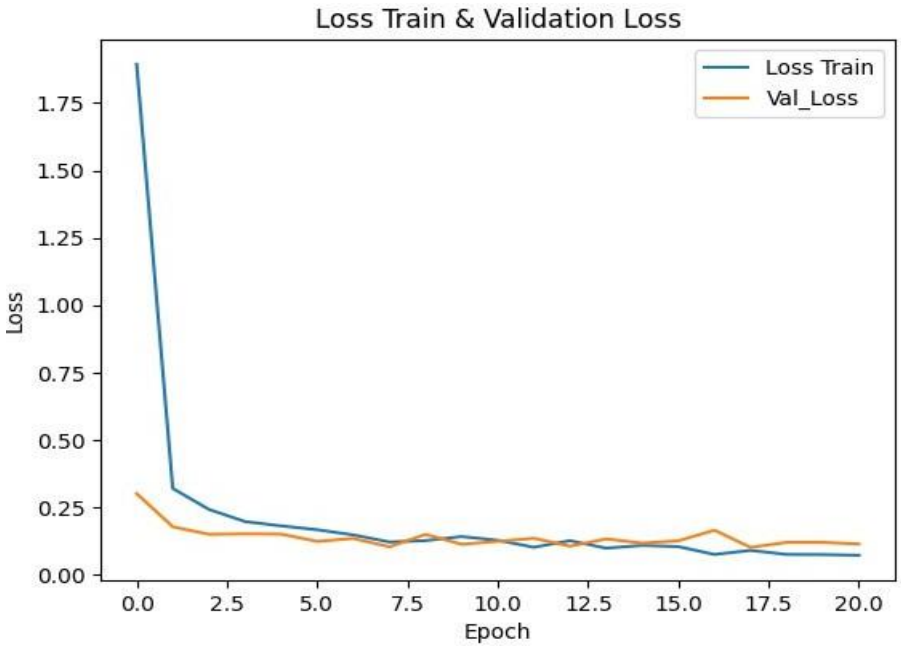

Fig.15: Loss vs epochs graph

In this graph Figure. 15, the horizontal axis represents the epoch, indicating the number of times the model has been trained on the complete

training dataset. The vertical axis represents the loss, which serves as an indicator of the model's performance on either the training or validation data. The validation loss is a measure of how well the model is able to generalize to new data. The validation loss is decreasing over time, which means that the model is learning to generalize better.

5.2. Feature Plot

The feature plot is a useful tool for understanding the distribution of pixel values in an image and for identifying potential areas of interest to analyse and understand data and visualize the features (variables) in a dataset. It frequently generates different sorts of plots or visualizations in machine learning and data analysis. A feature plot is given below where filter index is 30 and layer is the second layer of convolution 2D.

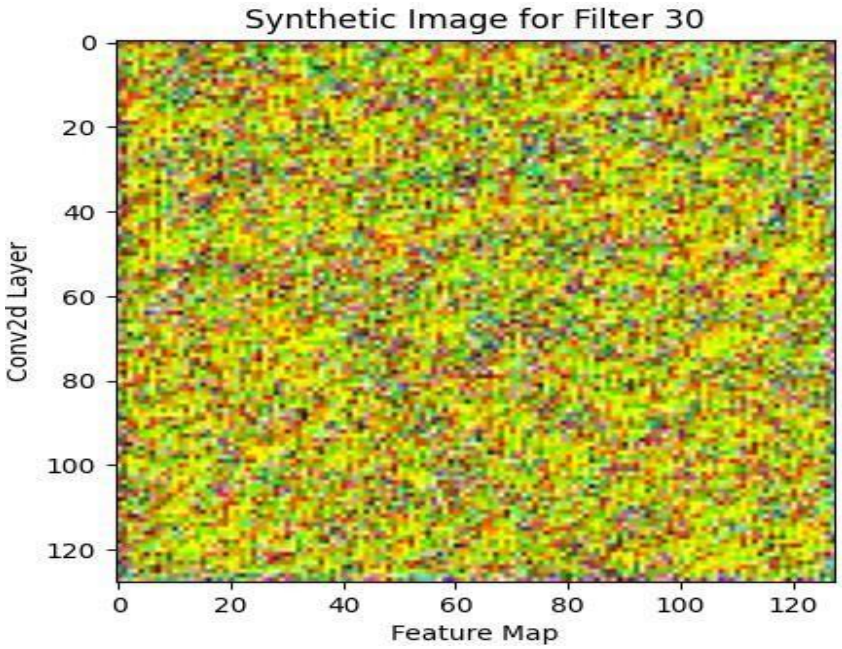

Figure 16: Feature map vs convolutional 2d layer

5.3. Confusion Matrix

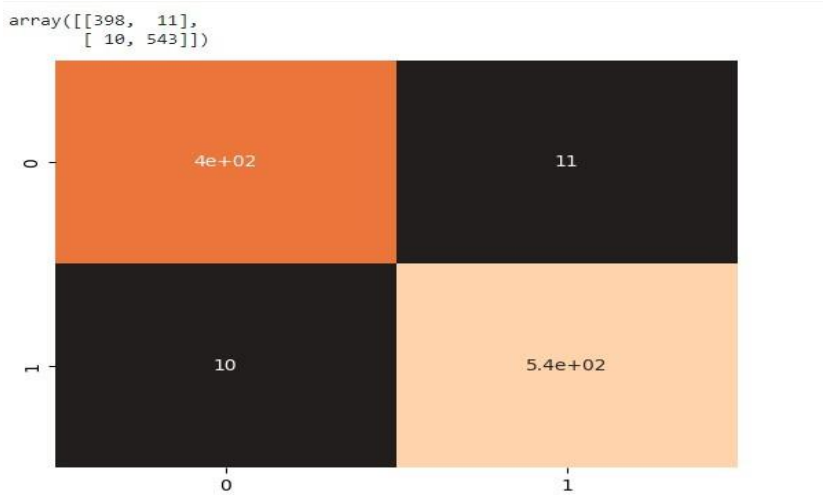

Figure 17: Confusion Matrix

In this Figure 17, True Positive (TP) is 398, False Negative (FN) is 11, False Positive (FP) is 10, and True Negative (TN) is 543. Now, present research worker can calculate the precision, recall and F1-score using the following formulas.

$$Precision = \frac{TP}{Tp+FP} \qquad Recall = \frac{TP}{TP+FN}$$

$$F1\ score = 2 \times \frac{recall \times precision}{recall+precision}$$

Accuracy precision, recall, and F1- Score of CNN are presented in Table 2 for dataset 1, and Table 3 for dataset 2. The precision value is 0.98. This means 98% of the positively predicted instances are classified correctly. The recall value is 0.97, indicating that 97% of the actual positive instances are correctly identified. The F1 -score is 0.97, the harmonic means of precision and recall.

Table 2: Table of Precision Recall F1 Score for dataset 1

ACCURACY	PRECISION	RECALL	F1SCORE
0.97	**0.98**	**0.97**	**0.97**

Table 3: Table of Precision Recall F1 Score for dataset 2

ACCURACY	PRECISION	RECALL	F1SCORE
0.92	**0.97**	**1.0**	**0.95**

5.4. Comparative Analysis

5.4.1. For dataset 1

Table 4 shows the accuracy, Precision, recall, F1 score by various machine learning and deep learning model for dataset1. Highest accuracy achieved by CNN and lowest accuracy obtained by Decision tree 77%. CNN give 96% precision, 97% recall and 96% F1 score.

Table 4: Compare accuracy and recall of model for dataset 1

	Accuracy	Precision	recall	F1 score
CNN	0.97	0.98	0.97	0.97
KNN	0.79	0.64	0.87	0.73
Decision Tree	0.77	0.73	0.74	0.73
SVM	0.86	0.79	0.40	0.53

5.3.2. For Dataset2

Similarly, Table 5 shown Decision Tree and SVM achieved highest accuracy 100% for dataset 2. 99% accuracy is being achieved by KNN. Whereas recall of CNN is 100% but recall of KNN is 79%.

Table 5: Compare accuracy and recall of model for dataset 2

	Accuracy	Precision	recall	F1 score
CNN	0.92	0.97	1.00	0.95
KNN	0.99	0.99	0.79	0.88
Decision Tree	1.00	1.00	0.79	0.88
SVM	1.00	1.00	0.79	0.88

Decision Tree and SVM achieved 100% accuracy and precision also 100%, but difference among recall, accuracy and Precision is too much. Where in case of CNN difference among recall, accuracy and precision is adjustable. Therefore, CNN outperform than other models with respect to of accuracy, precision, recall and f1 score. According to result of comparison consider that for Face mask detection used CNN. So, this research worked on Existing model [1] with same dataset.

However, when the base model is being applied to the dataset used in this research work, the accuracy drops to 87.94%. On the same dataset, when CNN is being applied, the accuracy increases to 97.00%.

Existing model: In the previous study [1], a base model is being employed. This model consists of two convolutional layers with a kernel size of 3x3. The first convolutional layer utilized 200 filters, while the second layer has employed 100 filters. Additionally, two max pooling layers are implemented, with a pooling size of 3x3. The first dense layer consists of 50 neurons, and the second dense layer (output layer) consists of 2 neurons.

CNN: In this research work, three convolution layers are being utilized with 32, 64, and 128 filters respectively. The kernel size for each layer is 3x3. Padding is being applied to the first and third convolution layers. A dropout ratio of 0.5 is used. Additionally, three dense layers are being implemented, with 128 neurons in the first layer, 64 neurons in the second layer, and 2 neurons in the third layer respectively.

Chapter 6

6. Conclusion and Future Scopes

Upon using the earlier model with the previous dataset, it is being observed that 96% accuracy is achieved. However, when the existing model is applied to the dataset used in this research work, the accuracy drops to 87.94%. On the same dataset, when CNN is applied, the accuracy increases to 97.00%, which is very good accuracy. All other models used in this thesis have achieved lower accuracy than the accuracy of proposed model using Convolutional Neural Networks. This suggests that the Proposed Model using Convolutional Neural Networks is doing better as compared to other models.

Here are some specific research directions that could be explored to enhance the future potential of facemask detection using CNN:

i. The robustness of CNN models to handle a wider range of variations in facial appearance, including different skin tones, hairstyles, and facial expressions can be enhanced.
ii. CNN models that can detect masks in real time, even in challenging conditions like low-light or crowded environments could be further developed.
iii. New CNN models can be created and improvised, that can identify different types of masks, such as surgical masks, cloth masks.
iv. Improving the design of CNN models that can detect improperly worn masks, such as masks that are not covering the nose or chin properly, can help in further improvement in future scope of work.

References

1. V. K. Pandey, V. K. Gupta and S. Kumar, "Face Mask Detection Using Convolutional Neural Network," 2021 3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N), Greater Noida, India, 2021, pp. 951-954. doi: 10.1109/ICAC3N53548.2021.9725689

2. R. P. Sidik and E. Contessa Djamal, "Face Mask Detection using Convolutional Neural Network," 2021 4th International Conference of Computer and Informatics Engineering (IC2IE), Depok, Indonesia, 2021, pp. 85-89, doi: 10.1109/IC2IE53219.2021.9649065

3. M.N.Kavitha, N.Kanimozhi, S.S.Saranya, S.Janani Sri, V.Kalpana, K.Jay avarthiniy, " Face Mask Detection Using Deep Learning", 2022 Second International Conference on Artificial and Smart Energy(ICAIS), Coimbatore,India 2022, pp. 319-324 ,doi: 10.1109/ICAIS53314.2022.9742825

4. M. S. Ali Mazumder, T. Hossain, F. M. J. Mehedi Shamrat, N. Jahan, Z. Tasnim and A. Khater, "Deep Learning Approaches for Diabetic Retinopathy Detection by Image Classification," 2022 3rd International Conference on Smart Electronics and Communication (ICOSEC), Trichy, India, 2022, pp. 1504-1510, doi: 10.1109/ICOSEC54921.2022.9952159

5 P. Khamlae, K.Sookhana Phibarn, W.Choenswat ," An Application of Deep Learning Techniques to Face Mask Detection During the COVID -19 Pandamic", 2021.IEEE 3rd Global Conference on Life Science and Technologies(2021) , pp.298-299, doi:10.1109/Life Tech 52111.2021.9391922

6. G. Lakshmi Durga, H. Potluri, A. Vinnakota, N. P. Prativada and K. Yelavarti, "Face Mask Detection using MobileNetV2," 2022 Second International Conference on Artificial Intelligence and Smart Energy (ICAIS), Coimbatore, India, 2022, pp. 933-940, doi:10.1109/ICAIS53314.2022.9742933

7. S. Machiraju, S. Urolagin, R. K. Mishra and V. Sharma, "Face Mask Detection using Keras, Opencv and Tensorflow by Implementing Mobilenetv2," 2021 3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N), Greater Noida, India, 2021, pp. 1485-1489, doi: 10.1109/ICAC3N53548.2021.9725546

8. F. M. J. Mehedi Shamrat, S. Chakraborty, M. M. Billah, M. A. Jubair, M. S. Islam and R. Ranjan, "Face Mask Detection using Convolutional Neural Network (CNN) to reduce the spread of Covid-19," 2021 5th International Conference on Trends in Electronics and Informatics (ICOEI), Tirunelveli, India, 2021, pp. 1231-1237, doi: 10.1109/ICOEI51242.2021.9452836

9. A. Giri, D. S. Bisht, A. Chauhan and I. Kumar, "Computerized Face Mask Detection System Using Deep CNN and Transfer Learning," 2023 International Conference on Device Intelligence, Computing and Communication Technologies, (DICCT), Dehradun, India, 2023, pp. 457-462, doi: 10.1109/DICCT56244.2023.10110187

10. H. Wang and C. Lursinsap, "Detecting Facial Images In Public With And Without Masks Using VGG And FR-TSVM Models," 2021 18th International Joint Conference on Computer Science and Software Engineering (JCSSE), Lampang, Thailand, 2021, pp. 1-5, doi: 10.1109/JCSSE53117.2021.9493848

11. S. R. Reza, X. Dong and L. Qian, "Robust Face Mask Detection using Deep Learning on IoT Devices," 2021 IEEE International Conference on Communications Workshops (ICC Workshops), Montreal, QC, Canada, 2021, pp. 1-6, doi: 10.1109/ICCWorkshops50388.2021.9473701

12. A. B. Handoko, V. C. Putra, I. Setyawan, D. Utomo, J. Lee and I. K. Timotius, "Evaluation of YOLO-X and MobileNetV2 as Face Mask Detection Algorithms," 2022 IEEE Industrial Electronics and Applications Conference (IEACon), Kuala Lumpur, Malaysia, 2022, pp. 105-110, doi: 10.1109/IEACon55029.2022.9951831

13. D. Kayali, P. Olawale, Y. Kirsal-Ever and K. Dimililer, "The effect of Compressor-Decompressor Networks with different image sizes on Mask Detection using Convolutional Neural Networks - VGG-16," 2022 Innovations in Intelligent Systems and Applications Conference (ASYU), Antalya, Turkey, 2022, pp. 1-5, doi: 10.1109/ASYU56188.2022.9925317

14. O. Tolulope Ibitoye and O. Oluwole Osaloni, "Masked Faces Classification using Deep Convolutional Neural Network with VGG-16 Architecture," 2022 3rd International Conference on Electrical Engineering and Informatics (ICon EEI), Pekanbaru, Indonesia, 2022, pp. 168-171, doi: 10.1109/IConEEI55709.2022.9972288

15. D. Sharma and G. S. Tomar, "Face Mask Detection Analysis for Covid19 Using CNN and Deep Learning," 2022 IEEE 11th International Conference on Communication Systems and Network Technologies (CSNT), Indore, India, 2022, pp. 239-252, doi: 10.1109/CSNT54456.2022.9787579

16. P. Kumar, V. K. Gupta and D. P. Singh, "Face Mask Detection Using Convolution Neural Network," 2022 3rd International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT), Ghaziabad, India, 2022, pp. 1-6, doi: 10.1109/ICICT55121.2022.10064500

17. D. Kayali, K. Dimililer and B. Sekeroglu, "Face Mask Detection and Classification for COVID-19 using Deep Learning," 2021 International Conference on INnovations in Intelligent SysTems and Applications (INISTA), Kocaeli, Turkey, 2021, pp. 1-6, doi: 10.1109/INISTA52262.2021.9548642

18. D. Wang, H. Yu, D. Wang and G. Li, "Face Recognition System Based on CNN," 2020 International Conference on Computer Information and Big Data Applications (CIBDA), Guiyang, China, 2020, pp. 470-473, doi: 10.1109/CIBDA50819.2020.00111

19. "Face Mask Dataset." [Online]. Available: https://www.kaggle.com/datasets/omkargurav/face-mask-dataset

20. Amazon Web Services, "What is Deep Learning?" Amazon Web Services. [Online]. Available: https://aws.amazon.com/what-is/deep-learning/#:~:text=Deep%20learning%20is%20a%20method,produce%20accurate%20insights%20and%20predictions

21. “Deep learning," Wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/Deep_learning/

22. “Classify — XLearn 0.45.0 documentation," Read the Docs. [Online]. Available: https://xlearn.readthedocs.io/en/latest/source/introduction/classify.html

23. Yann MJL, "What is Machine Learning in Simple English," Medium. [Online]. Available: https://yannmjl.medium.com/what-is-machine-learning-in-simple-english-b0aaa251cb60#:~:text=Machine%20learning%20is%20an%20application,it%20to%20learn.

24. "AI (Artificial Intelligence) - Definition," TechTarget. [Online]. Available: https://www.techtarget.com/searchenterpriseai/definition/AI-Artificial-Intelligence#:~:text=What%20is%20artificial%20intelligence%20(AI,speech%20recognition%20and%

25. Coding Ninjas, "Convolution Layer, Padding, Stride, and Pooling in CNN," Coding Ninjas. [Online]. Available: https://www.codingninjas.com/studio/library/convolution-layer-padding-stride-and-pooling-in-cnn

26. Raj, D., "Convolutional Neural Networks (CNN) Architectures Explained," Medium. [Online]. Available: https://medium.com/@draj0718/convolutional-neural-networks-cnn-architectures-explained-716fb197b243

27. Python Software Foundation, "Why is Python a dynamic language and also is it not so anymore?" Python.org. [Online]. Available: https://www.python.org/doc/essays/blurb/

28. Naufal, Mohammad Farid, Selvia Ferdiana Kusuma, Zefanya Ardya Prayuska, Ang Alexander Yoshua, Yohanes Albert Lauwoto, Nicky Setyawan Dinata, and David Sugiarto. "Comparative analysis of image classification algorithms for face mask detection." *Journal of Information Systems Engineering and Business Intelligence* 7, no. 1 (2021): 56-66. https://e-journal.unair.ac.id/JISEBI/article/view/25167

29. Vijitkunsawat, Wuttichai, and Peerasak Chantngarm. "Study of the performance of machine learning algorithms for face mask detection." In *2020-5th international conference on information technology (InCIT)*, pp. 39-43. IEEE, 2020. https://doi.org/10.1109/InCIT50588.2020.9310963

30. Eman, Mohammed, Tarek M. Mahmoud, Mostafa M. Ibrahim, and Tarek Abd El-Hafeez. "Innovative hybrid approach for masked face recognition using pretrained mask detection and segmentation, robust PCA, and KNN classifier." Sensors 23, no. 15 (2023): 6727. https://doi.org/10.3390/s23156727

Appendix

```
import cv2
import os
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
import numpy as np
import keras
import tensorflow as tf
from sklearn.metrics import confusion_matrix

import seaborn as sns
from sklearn.metrics import classification_report
```

```
[ ] with_mask='/content/drive/MyDrive/Dataset/train/with_mask'
    without_mask='/content/drive/MyDrive/Dataset/train/without_mask'
```

```
i=1
plt.figure(figsize=(20,20))
for img in os.listdir(with_mask):
    if i==51:
        break
    plt.subplot(5,10,i)
    img_arr=cv2.imread(os.path.join(with_mask,img))
    plt.imshow(img_arr)
    i+=1
    plt.axis('off')
```

```
i=1
plt.figure(figsize=(20,20))
for img in os.listdir(without_mask):
    if i==51:
        break
    plt.subplot(5,10,i)
    img_arr=cv2.imread(os.path.join(without_mask,img))
    plt.imshow(img_arr)
    i+=1
    plt.axis('off')
```

```
label=[]
data=[]
size=[]
#with mask label 0
for img in os.listdir(with_mask):
    img_arr=cv2.imread(os.path.join(with_mask,img))
    data.append(img_arr)
    label.append(0)
    size.append(img_arr.shape)
#without mask label 1
for img in os.listdir(without_mask):
    img_arr=cv2.imread(os.path.join(without_mask,img))
    data.append(img_arr)
    label.append(1)
    size.append(img_arr.shape)
```

```
[ ] #Resize Image
    Size=128
    for x in range(len(data)):
        data[x]=cv2.resize(data[x],(Size,Size))
```

split data

```
[ ] data=np.array(data)
    label=np.array(label)
    X_train,X_test,y_train,y_test=train_test_split(data,label,test_size=.15,shuffle=True,random_state=44)
    print('X Train Shape is :',X_train.shape)
    print('X Test Shape is :',X_test.shape)
    print('Y Train Shape is :',y_train.shape)
    print('Y Test Shape is :',y_test.shape)
```

```
model=keras.models.Sequential()
model.add(keras.layers.Conv2D(filters=32,kernel_size=(3,3),padding='same',activation=tf.nn.relu,input_shape=(128,128,3)))
model.add(keras.layers.MaxPooling2D((3,3)))
model.add(keras.layers.Conv2D(filters=64,kernel_size=(3,3),activation=tf.nn.relu))
model.add(keras.layers.MaxPooling2D((3,3)))
model.add(keras.layers.Conv2D(filters=128,kernel_size=(3,3),padding='same',activation=tf.nn.relu))
model.add(keras.layers.MaxPooling2D((3,3)))
model.add(keras.layers.Flatten())
model.add(keras.layers.Dense(128, activation='relu'))
model.add(keras.layers.Dropout(0.5))
model.add(keras.layers.Dense(64, activation='relu'))
model.add(keras.layers.Dropout(0.5))
model.add(keras.layers.Dense(2, activation='softmax'))
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 128, 128, 32)      896

 max_pooling2d (MaxPooling2  (None, 42, 42, 32)        0
 D)

 conv2d_1 (Conv2D)           (None, 40, 40, 64)        18496

 max_pooling2d_1 (MaxPoolin  (None, 13, 13, 64)        0
 g2D)

 conv2d_2 (Conv2D)           (None, 13, 13, 128)       73856

 max_pooling2d_2 (MaxPoolin  (None, 4, 4, 128)         0
 g2D)

 flatten (Flatten)           (None, 2048)              0

 dense (Dense)               (None, 128)               262272

 dropout (Dropout)           (None, 128)               0

 dense_1 (Dense)             (None, 64)                8256

 dropout_1 (Dropout)         (None, 64)                0

 dense_2 (Dense)             (None, 2)                 130

=================================================================
Total params: 363906 (1.39 MB)
Trainable params: 363906 (1.39 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
[ ] tf_arr = tf.convert_to_tensor(X_train, dtype=tf.float32)
```

```
[ ] model.compile(optimizer ='adam',loss='sparse_categorical_crossentropy',metrics=['accuracy'])
    hist=model.fit(X_train,y_train,epochs=21,validation_split = 0.1)
```

```
plt.plot(hist['accuracy'],label='Accuracy Train')
plt.plot(hist['val_accuracy'],label='Accuracy_Loss')
plt.title('Accuracy Train & Validation Accuracy')
plt.legend()
```

```
plt.plot(hist['loss'],label='Loss Train')
plt.plot(hist['val_loss'],label='Val_Loss')
plt.title('Loss Train & Validation Loss')
plt.legend()
```

```
score, acc = model.evaluate(X_test, y_test)
print('Test Loss =', score)
print('Test Accuracy =', acc)
```

```
31/31 [==============================] - 0s 5ms/step - loss: 0.2434 - accuracy: 0.9449
Test Loss = 0.243412747979164 12
Test Accuracy = 0.94490647315979
```

```
prediction = model.predict(X_test)
prediction
```

```
pred_label=[]
for row in prediction:
    N=np.argmax(row)
    if N==1:
        pred_label.append('with mask')
    else:
     pred_label.append('without mask')
pd.DataFrame(pred_label,columns=['Prediction'])
#prediction
```

```
plt.figure(figsize=(20,30))
i=1
for img in X_test:
   if i==51:
       break
   plt.subplot(10,5,i)
   plt.imshow(img)
   plt.title(pred_label[i-1])
   plt.legend()
   i+=1
   plt.axis('off')
```

Confution Matrix

```
[ ]  y_pred=[1 if m=='with mask' else 0 for m in pred_label]
     CM = confusion_matrix(y_test, y_pred)
     sns.heatmap(CM, center = True,annot=True,cbar=False)
     CM
```

```
import numpy as np
import tensorflow as tf
from tensorflow.keras.applications import VGG16
from tensorflow.keras.preprocessing import image
import matplotlib.pyplot as plt


# Choose a layer to visualize
layer_name = 'conv2d_2'
selected_layer = model.get_layer(layer_name)
model = tf.keras.Model(inputs=model.input, outputs=selected_layer.output)

# Function for activation maximization
def maximize_activation(filter_index, model, iterations=20, step=1.0):
    input_shape = model.input_shape[1:]  # Get input shape without batch size
    input_img = tf.random.uniform((1,) + input_shape, 0, 1)  # Initial random input
    input_var = tf.Variable(input_img)

    for _ in range(iterations):
        with tf.GradientTape() as tape:
            outputs = model(input_var)
            loss = -tf.reduce_mean(outputs[..., filter_index])
        grads = tape.gradient(loss, input_var)
        grads /= (tf.sqrt(tf.reduce_mean(tf.square(grads))) + 1e-5)
        input_var.assign_add(grads * step)
```

```
[ ]         grads /= (tf.sqrt(tf.reduce_mean(tf.square(grads))) + 1e-5)
            input_var.assign_add(grads * step)

        return input_var.numpy()[0]

    # Choose a filter/neuron index to visualize
    filter_index = 29

    # Generate synthetic image that maximally activates the chosen neuron
    synthetic_img = maximize_activation(filter_index, model)

    # Display the synthetic image
    plt.imshow(synthetic_img)
    plt.title(f'Synthetic Image for Filter {filter_index}')
    plt.show()
```

```
import tensorflow as tf
from tensorflow.keras.applications import VGG16

# Load pre-trained VGG16 model
#base_model = VGG16(weights='imagenet', include_top=False)

# Create a model that includes only layers up to a certain layer
selected_layer_name = 'conv2d_2_input'
selected_layer = model.get_layer(selected_layer_name)
feature_selection_model = tf.keras.Model(inputs=model.input, outputs=selected_layer.output)

# Load and preprocess an image
image_path = '/content/drive/MyDrive/Dataset/train/with_mask/with_mask_1.jpg'
image = tf.keras.preprocessing.image.load_img(image_path, target_size=(128, 128))
input_image = tf.keras.preprocessing.image.img_to_array(image)
input_image = tf.keras.applications.vgg16.preprocess_input(input_image)
input_image = tf.expand_dims(input_image, axis=0)

# Get the activations from the selected layer
activations = feature_selection_model.predict(input_image)

# Visualize the feature maps
import matplotlib.pyplot as plt
for i in range(activations.shape[-1]):
    plt.subplot(4, 8, i+1)
    plt.imshow(activations[0, :, :, i], cmap='viridis')
    plt.axis('off')
```