Realtime Face-Mask Detection Using Hybrid Deep Learning Techniques

Harteley Sebastian
(hs2397@bath.ac.uk)

MSc Computer Science in Machine Learning and Autonomous System
The University of Bath
⟨2022⟩

Realtime Face-Mask Detection Using Hybrid Deep Learning Techniques

Submitted by: Harteley Sebastian

Copyright

Declaration

This dissertation is submitted to the University of Bath in accordance with the requirements of the degree of Master of Science in the Department of Computer Science. No portion of the work in this dissertation has been submitted in support of an application for any other degree or qualification of this or any other university or institution of learning. Except where specifically acknowledged, it is the work of the author.

Abstract

The COVID-19 disease is causing a problem for societies and economies worldwide, culminating in a global health crisis [30]. Due to this situation, more people are concerned about their health conditions. Also, the government is trying to prevent the spread of COVID-19 disease and make the public aware of how to stay healthy. This project aims to create an application to help the government track and analyse people's behaviour upon using face-mask to prevent the spread of COVID-19 disease. In this application, we have divided the system working into several steps. We have compared the face detector models (Single Shot Multibox Detector and Haar-Cascade Classifier) in real-time in some everyday situations where face detection might become tricky and presented the results above. We can conclude that the Caffe model performs better when detecting a person's face's side view than the Haar-Cascade Classifier. While for the face-mask classifier, upon comparing all the models' performance, Nasnet outperforms the rest of the models with an accuracy of 99.75%. When applying all the CNN models on the application for real-time classification, NasNet, MobileNet_V2, GoogleNet work perfectly. Lastly, applying the age and gender predictor working side by side with the face detector and face-mask classifier seems to work well, with some minor errors on the age prediction (2-5 years error rate).

Contents

List of Figures

List of Tables

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my senior supervisor, Dr Cherry Zhao for her patience, motivation and immense knowledge. Your guidance has helped me in completing my dissertation.

Secondly, I would like to thank all of my family and friends who supported me throughout the completion of my project. Their kind cooperation has allowed me to finish this research.

Also, I would like to thank University of Bath and the lecturers who have taught me throughout my degree program here. They have instilled in me the wisdom and integrity to face any challenges as well as constantly striving for excellence.

Last but not the least, all of my colleagues at University of Bath for providing me the warm and friendly atmosphere.

Chapter 1

Introduction

1.1 Problem Description

Since 2019, the new virus (SARS-CoV-2) that causes infectious Coronavirus 2 disease has been recorded in Wuhan for the first time, and it has since become a public health problem throughout the world. This disease is causing a problem for societies and economies worldwide, culminating in a global health crisis [30]. Due to this situation, more people are concerned about their health conditions. The government is trying to prevent the spread and make the public aware of how to be healthy by avoiding contact from person to person. Coronavirus 2 is a respiratory viral infection that is rising, which is now called COVID-19. COVID-19 has created a significant healthcare problem worldwide, especially in the third wave [59]. Several industries have closed because of this outbreak. Additionally, due of their enormous impact on people's daily lives, certain industries, including maintenance work and infrastructure building, have not been hindered [63, 41].

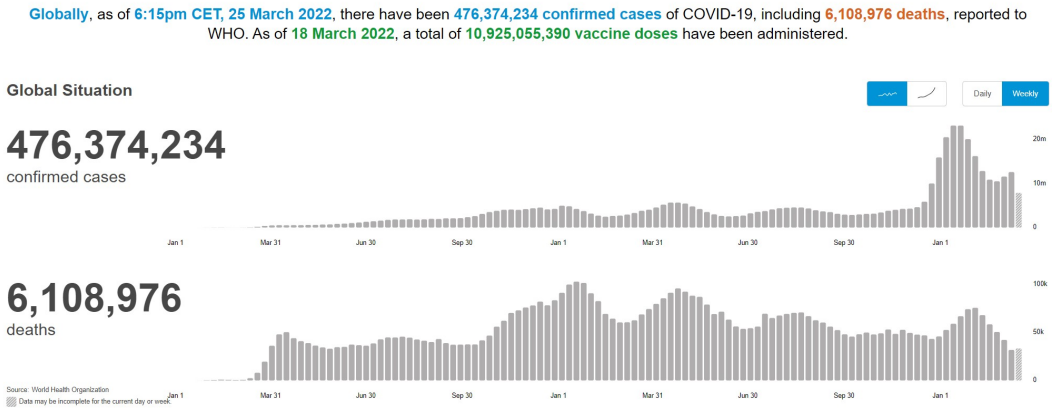


Figure 1.1: WHO Coronavirus (COVID-19) Dashboard. Adapted from World Health Organization [35]

The virus has already spread to most countries throughout the world [39]. According to the most recent WHO figures (see figure 1.1), on 25th March 2022, there were 476,374,234 confirmed cases and 6,108,976 deaths. Coronavirus can be transmitted primarily through respiratory droplets made by people when they talk, cough or sneeze, and breathing [59].

Figure 1.2: Should we all be wearing masks?. Retrieved from CNBC [50]

One way to prevent the sickness from spreading from one person to another is by wearing a face mask. It can restrict the passage of droplets in the air from one person to the next. Leung et al [28] shown that the use of surgical masks helps to prevent the spread of COVID-19 diseases. WHO also strictly recommends that people wear masks while in public places. Many public places, such as markets, cafes, malls, and others, request people to wear masks [11]. If they do not do so, they are not allowed to enter the place. So as we know, wearing a face mask is necessary nowadays.

Artificial Intelligence is transforming the global world. Using Artificial Intelligence techniques such as computer vision to detect whether a person wears a mask or not can help the government track, analyze, and prevent the spread of COVID-19 automatically [60]. As a result, the focus of this research study will be implementing a deep learning solution with several methods to compare, which can automatically detect people wearing a face mask in real-time accurately.

1.2 Objectives and Deliverables

This project will use face detector models, age and gender prediction model and some pre-trained Convolutional Neural Networks (CNN) models. The face detector models will detect and track face(s) from a real-time camera or uploaded video by the user. The face detector outputs will then be resized to 224x224 (VGG and ResNet default input size) before feeding them as inputs to the developed CNN models. The CNN role here is to classify whether the face detected is wearing a face mask or not. Then, the age and gender predictor will play the role to predict the approximate age and the gender of the face(s). The report will explain the methods used for this project and analyse the results of the models.

This project aims to achieve a high level of accuracy (>90%) on face detection and face mask classification for the best face detector and the best CNN model chosen. The extensions of the objective are to develop a simple user interface where users can use a live webcam or upload a video and be able to predict estimated age and gender from the person's face.

Chapter 2

Literature and Technology Survey

2.1 Machine Learning

With the ever-increasing data collection, generation, and storage options in today's digital age, the amount of data is also growing exponentially. However, manual evaluation and analysis are proving problematic in the face of this immense mass and diversity of data. When processing data electronically, the most significant challenge remains the identification of the truly relevant information and the extraction of knowledge.

In machine learning, algorithms are trained to find patterns and correlations in large data sets and make the best decisions and predictions based on this analysis. The machine learning workflow can be broken down into seven steps (see Figure 2.1): Before an algorithm can be applied, data must first be selected, preprocessed, and transformed. Afterwards, the chosen model will be trained on the part of the data set and then evaluated on another subset of the data. In this process, the results are presented visually or textually by the system and evaluated by experts for their suitability. After the evaluation, the model can be tested again with a different set of hyperparameters (if there are any). At the end of the process, the machine learning model can be used to answer the questions that were asked at the beginning, i.e., make predictions [58].

Figure 2.1: Machine Learning Steps

Machine learning techniques are divided into supervised and unsupervised learning. In supervised learning, the categories are known and given by an expert. In contrast, unsupervised learning does not require prior knowledge of the data set. The different supervised learning methods fall into two categories: classification and regression. In classification, a model assigns a target value from a finite discrete set to the input data. An example would be object recognition

from photos. On the other hand, in regression, a real target value is assigned to the input data. This task corresponds to areal-valued forecast [53].

2.2 Deep Learning for Classification

Deep learning models are considered better than traditional machine learning models because they provide an end-to-end solution for classification problems. Deep Learning layers virtual neurons to create massive artificial neural network architectures. The task of each neuron is to add up all of the inputs and determine whether or not to transmit an output signal to the layer of neurons above it. A network layer's neurons are connected to neurons in layers above and below it. Like our own brain, this neural network can solve various issues by learning the best weights for each of these connections. Despite its simple name, a neural network may reflect complex challenges due to the sheer number of connections between neurons [23].

Even though deep learning is a type of machine learning, it has gained appeal due to its adaptability and the fact that it is based on how our brains work. It is often preferred since it provides a complete solution to classification problems. Because it may delete functionality, CNN-based design is a better end-to-end approach.

2.2.1 Artificial Neural Networks (ANN)

The first Artificial Neural Networks (ANN) were developed in the early 1950s, shortly after computers were built that could execute the necessary computational instructions. They are based on the idea of simulating small computational units - "neurons" - arranged in layers and connected by many "synapses". Each unit in the lowest layer takes a value from the outside, such as the brightness of a particular image pixel, and relays that information to some or all of the units in the next higher layer. By applying a simple mathematical rule, those units then integrate the inputs from the first layer and pass the results upward until, finally, the last layer provides an answer - for example, by classifying the original image as a "car" or a "person". The so-called Perceptron, the first and most straightforward version of a neural network, was proposed by Frank Rosenblatt, which consists of a single neuron [3].

The power of such networks lies in their learning ability. Suppose they are fed training data sets and the corresponding correct answers. In that case, they can gradually improve their performance by adjusting the strength of each connection until an input at the lowest level leads to correct results at the highest level. This process, which is somewhat reminiscent of the learning processes in the brain, eventually results in a network that correctly classifies new inputs not included in the training data set.

Shallow vs Deep Learning

As we have considered them, shallow neural networks have only a single feature transformation, i.e. one hidden layer. Like many other machine learning algorithms, they transform features only once into a different representation. One problem with such flat architectures is the classification quality since learners must manage to generalize the high-dimensional initial features to the most important ones with only a single transformation. On the other hand, Deep Neural Networks have more than one hidden layer between the input layer and the output layer and can learn features from (low-processed) raw data, hence the term Deep Learning. The number of layers contributing to a data model is referred to as the depth of the model. The

benefit of multiple neuron layers is that "new" information can be formed between the layers, representing the original information. These representations are a modification or abstraction of the actual input signals. Deep learning often involves dozens or even hundreds of successive representational layers, all of which are automatically learned by providing the training data. While time is saved by omitting the feature engineering step, deep neural networks need time to set up the structure and fine-tune the many possible parameters. Deep neural networks are more complex than traditional machine learning algorithms, and the model parameters are difficult to analyze and interpret [5].

Types of Neural Networks

Despite the very uniform basic idea for constructing neural networks, there are some ways in which they can differ. On the one hand, the networks are differentiated in the processing and the kind of data fed through them. These can be, for example, simple values but also collections of values like images. On the other hand, they can also differ in the type and arrangement of neurons and connections.

The most basic type of neural network is the feed-forward deep network, often known as the multilayer perceptron (MLP). An input layer, one or more hidden layers, and an output layer make up this completely linked network. The hidden layers are so-called "dense layers", i.e. entirely interconnected neurons with weights to all neurons of the previous and following layer. The neurons in the input layer take and forward them with the weights to the neurons in the hidden layers. These neurons use an activation layer and have a variable bias, which both are responsible for their output. The neurons on the output layer also have an activation function and a bias. The activation function of these output neurons may differ from those of the hidden neurons. The computed activations are finally returned as the result or output. This process is called feed-forward, as the calculated outputs of the respective nodes are only propagated forward to the next layer [62].

2.2.2 Convolutional Neural Network (CNN)

CNN models are now considered cutting-edge because they can solve picture identification challenges. CNN templates consist of numerous layers, including input-output and unknown sorting layers. The convolutional, pooling and dense layers are the three essential layers of this model (Fully connected layer).

1. **Convolutional Layer:** Many filters make up a convolutional layer, each with its parameters to consider. The filters' height and weight are less than the input. Each filter in the input is convolved to build a neuron-based activation map [24]. This layer has a series of kernels that glide over the image in width and height directions, providing a two-dimensional activation map for each kernel. With activation functions like RELU, Sigmoid, Tanh, and SoftMax, nonlinearity is applied to activation maps.
2. **Pooling Layer:** A pooling layer is frequently incorporated between two convolutional layers. By downsampling the representation, the pooling layer reduces the number of parameters and calculations [24]. Values are extracted from activation maps using stride and pooling methods. Pooling functions such as Max pooling and Average pooling are available. Max pooling is more widely used than Average pooling because it has a better performance.

3. **Dense Layer:** Thick layers, also known as flattening layers, are entirely interconnected. Features of convolution layers created from previously hidden layers are flattened into a 1-D array in these layers for performance categorisation [6].

The CNN model can be seen in figure 2.2

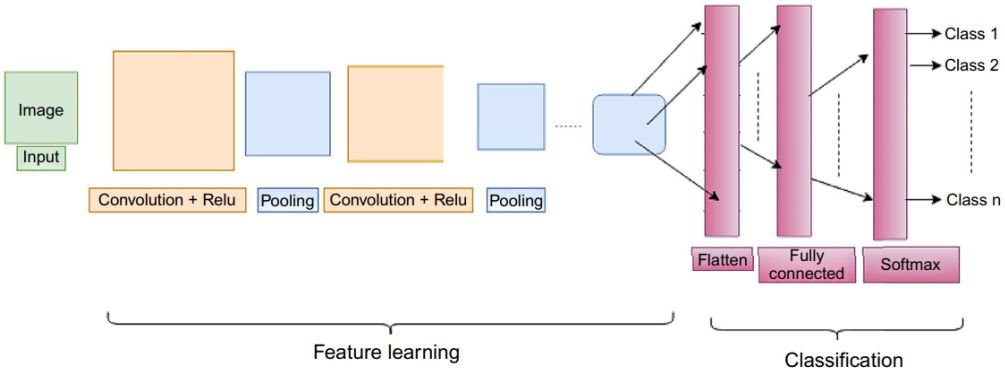


Figure 2.2: Simple Convolutional Neural Network

2.2.3 Transfer Learning

Transfer learning is a practical solution to the problem of data scarcity, and it has quickly become well-known in the field of Computer Vision. Transfer learning is a machine learning approach in which a trained CNN model is used for a different purpose on another problem.Pre-trained CNN models are a frequent approach for computer vision in deep learning due to the vast processing and time resources required to develop neural network models for these tasks and the enormous leaps in their performance on related problems [14]. After training a base network with a base dataset and task, we apply or transfer the newly acquired features to a second target network that will be trained with a target dataset and task. This process is known as transfer learning. This method is more likely to succeed if the characteristics are generic, i.e., rather than being particular to the base job, it applies to both the base and target tasks [61]. Nonetheless, it is popular to do transfer learning in deep learning, even given that massive resources are necessary to train deep learning models or the extensive and complex datasets on which deep learning models are taught.

Unlike the linear activity function, the activity level in sigmoid functions is bounded both upward and downward. So, it has the advantage that the activity in the network cannot "spill over" unintentionally and only produce error values. Due to the existing horizontal asymptote, the use of sigmoid functions often leads to a disappearance of the transitions. It thus could lead to a learning standstill in the respective subarea of the network [3].

The Rectified Linear Unit (ReLU) is also called a linear function with a threshold. This function describes a simple straight line, which projects any negative input to 0. The constant progression of the straight line leads to faster and constant learning. The fact that the ReLU function does not provide negative values ensures stabilization of the weights during the learning phase. It can be implemented without much computational effort and leaves neurons with negative values disabled, significantly reducing the computational cost and making it

popular for large networks [13]. However, when the gradient reaches zero during training, the ReLU stops responding to changes and returns no signal; this is also known as the "Dying ReLU" problem and can cause large parts of the network to fail [3] [15].

2.3 Related Work

Facial Recognition

In 2020, Mayur Surve et al. [51] developed a model that records live images from a camera. It then uses several algorithms to recognise and identify faces. They also created a single-click interface for capturing images, creating datasets, and importing datasets. They used the Haar cascade approach to recognise the face in the image.

In 2021, Anirudha B Shetty et al. [46] proposed using the Haar Cascade classifier and Local Binary Pattern classifier for facial recognition for biometric identification purposes. In this paper, the LBP classifier and Haar Cascade classifier are compared. This paper discovered that the Haar cascade classifier is more accurate than the LBP classifier. The Haar Cascade can detect 19 faces out of 20 people in an image (95% accuracy), while the LBP classifier can detect 17 faces out of 20 people (85% accuracy). However, it seems that both classifiers have the tendency not to detect the faces of the youngsters.

Face Mask Detection

Several works have been done for the prediction of Face Mask Detection images. Face Mask Detection image classification literature is mainly divided into machine learning and deep learning-based techniques. Various Face Mask Detection datasets are available for classification tasks. However, they are partially or not entirely available for everyone. However, This data is publicly available. Several research works focus on face recognition for identity verification and image reconstruction. However, the main aim and object of this study are to categorise people who are wearing and not wearing masks in public places such as Malls, Theatres, and Parks to overcome the deadly disaster and prevent the spreading of the disease.

Bosheng Qin et al. [40] proposed a face mask identification method using the SRCNet model for classification and achieved an accuracy of 98.7% in classifying the data into three groups, namely wearing a face-mask, not wearing face mask, and incorrect wearing of a face-mask.

Md. Sabbir Ejaz et al. [10] used the feature selection method as Principal Component Analysis (PCA) algorithm to detect face masks. It was shown that PCA is effective in detecting faces without a mask with an accuracy of 96.25 percent; however, accuracy is only 68.7 percent in recognizing faces behind a mask.

Rodriguez et al. [34] designed a model to automatically detect a face mask called a surgical mask in operating rooms. This architecture's objective is to generate alarms when anybody in the hospital is not wearing a mask. This architecture achieved an accuracy of 95%.

Javed et al. [25] proposed an interactive MRGAN model that eliminates objects like microphones in the facial Face Mask Detection using a Transfer Learning-based model named InceptionV3.

Theckedath et al. [56] experimented by looking at three distinct networks for recognising normal facial behaviour in people between the ages of 18 and 50. Its goal is to determine which network is the most efficient in detecting impact states. The three pre-trained networks used

were VGG-16, ResNet50, and SE-ResNet50, and their top fully connected layer was adjusted. They successfully compared VGG16, ResNet50, and SE-ResNet50; all three pre-trained networks provide excellent accuracy, precision, and recall numbers.

Inception v3 has been demonstrated to achieve higher than 78.1% accuracy on the ImageNet dataset, according to Szegedy et al.'s original publication, "Rethinking the Inception Architecture for Computer Vision" [52]. The model results from several concepts developed over time by many scholars [16].

Therefore, to contribute to the further improvements of face mask recognition and classification in the battle against COVID-19, a transfer learning-based approach and a CNN base model are proposed that utilise training models such as VGG-16, Resnet-50, Inception_V3 (GoogleNet), MobileNet, and NasNet.

Chapter 3

Proposed Methodology

The objective of this chapter is to justify that the proposed model has overcome all the limitations present in previous surrogate models. The proposed methodology is for the classification of Face mask detection and the proposed method of the whole system working based on real-time.

The proposed methodology will use Haar-Cascade Classifier, Deep Neural Networks (DNN) and Convolutional Neural Network (CNN). The way the system works is to take a real-time frame and detect faces using a Haar-Cascade Classifier or a DNN (res_10 300 x 300 SSD Caffe model trained over 140,000 iterations) to find faces and then draw a bounding box over those faces. The model sends (faces) regions of interest to our CNN model to predict whether a person wears a face mask. While doing so, the other two CNNs (age_net and gender_net Caffe models) will predict the approximate age and the gender of the person. Through this method, the authorized organization may monitor the locations and take action.

3.1 Haar-Cascade for Face Detection

Haar Cascading is a machine learning approach that involves drilling a classifier from many positive and negative pictures. The algorithm is advanced by Paul Viola and Michael Jones [2, 31]. Haar feature-based cascade classifiers are the classifiers used for object detection. This classifier uses a machine learning strategy to locate items by instilling a cascade operation from the pictures to find features in other photos. The identification of faces and facial emotions in images by using Haar Cascading has been thriving. The job is done by showing positive and negative pictures to the classifier. The image's characteristics are then extracted. The pixels in the white rectangle are added together, while the pixels in the black rectangle are subtracted to generate each feature. It recognises the faces of various people in a variety of scenarios. Any size of the Haar-like feature could be calculated in constant time thanks to integral images [45].

3.1.1 Haar-Like Feature

The most significant feature of the Haar cascade classifier for human face detection is the Haar features. Haar features are used to assess the existence of features in a picture [48]. To produce a single value for each feature, subtract the number of pixels under the white rectangle from the number of pixels under the black rectangle.

The programme will automatically search for Haar qualities based on the lines and borders of the dark and bright photographs. The three categories of Haar features are edge features, line features, and centre features [19]. Figure 3.1 shows the Haar features, rectangular characteristics for quick human face identification.

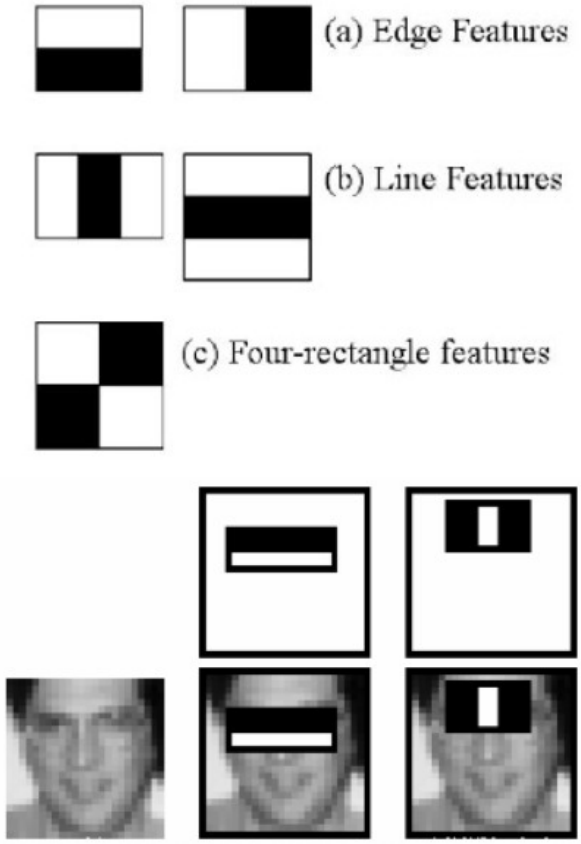


Figure 3.1: Haar-features finding the best threshold to classify the face on an image to positive and negative. Retrieved from: OpenCV [8]

Once the Haar feature has been found in that image area, the image integral will be computed in that pixel region. The image essential calculation procedure may be seen in equation 3.1 for the Haar Cascade classifier [37].

$$s(x, y) = i(x, y) + s(x, y - 1) + s(x - 1, y) - s(x - 1, y - 1) \tag{3.1}$$

The equation's informations:

- s(x, y) = the sum of each pixel's value
- i(x, y) = the input image obtained pixel's intensity value
- s(x, y - 1) = the value of y-axis of the pixel
- s(x - 1, y) = the value of x-axis of the pixel
- s(x - 1, y - 1) = the diagonal pixel value

3.1.2 Cascade Classifier

The Viola-Jones approach begins by examining these properties in the given image with a base window size of 24x24. Considering all possible haar feature properties like type, location, and scale, we must count 160,000 features in this window. However, this is almost impossible. As a result, the AdaBoost algorithm was developed to address this problem. AdaBoost is a machine learning method for selecting the best features from a database of 160,000. It builds robust classifiers by linearly combining weak classifiers [38]. This system will detect an image of a human face if all the stages are finished. If the image does not go through one of the stages, no human face is seen in the image. The Adaboost algorithm can be seen in figure 3.2.

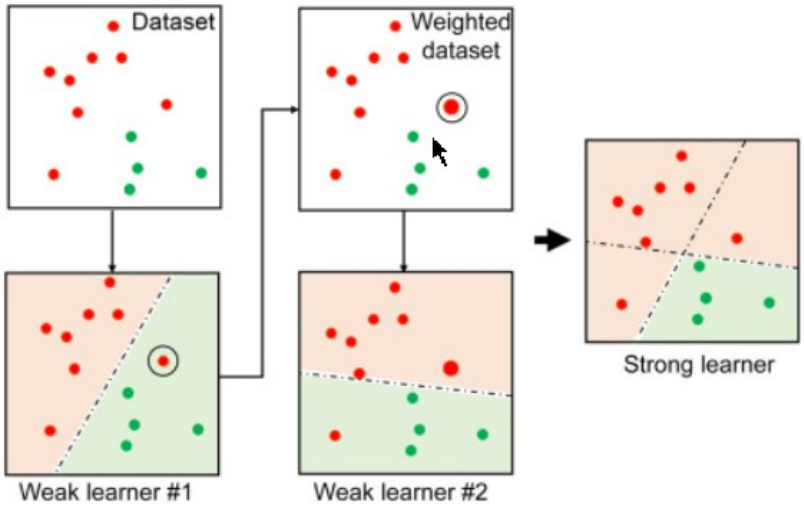


Figure 3.2: Implementation of the AdaBoost classifier on a dataset with two features and two classifications. Weak learner #2 corrects the mistake made by weak learner #1, allowing the decision boundaries of the two weak learners to be combined to become a strong learner [32].

3.2 Face-Mask Classifier Using CNN

As we know that machine learning approaches are highly biased to right choice of descriptor for feature extraction and selection. However the ability of deep leering networks are they are end-to-end network [57]. They performed automatically feature extraction and classification. Also compare to traditional approaches deep learning models produced remarkable results. Here are widely adopted Deep learning architectures are presented. Proposed networks for the classification of face mask detection. The networks are fine-tuned, added customized layers such as global average pooling, dense layer, and dropout layer for classification. The proposed methodology shown in Fig 9. work take input of 224 x 224 and apply pre-processing steps and normalization, and increase the number of images using data augmentation. The proposed images fed into the proposed architectures for classification and then evaluate the performance of the network using evaluation matrix's mention below.

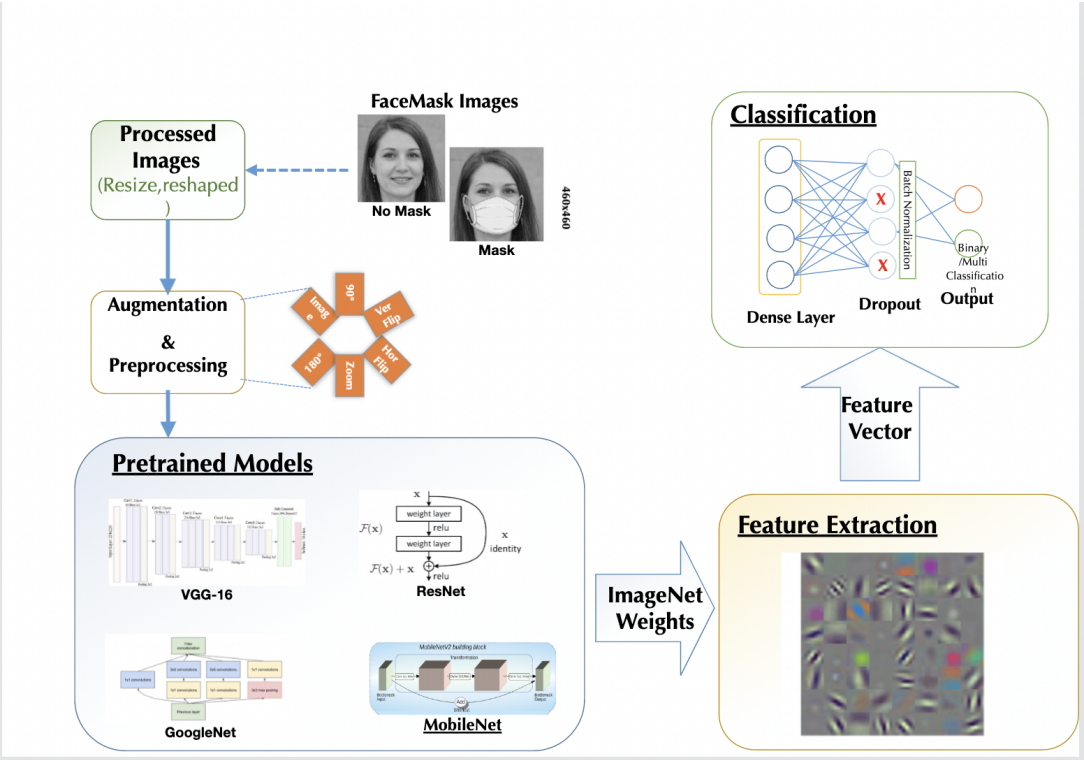


Figure 3.3: CNN Model Diagram

3.2.1 Data Augmentation

The frequent challenge in developing the model is overfitting. In a nutshell, when a trained model performs well on the training set but not so well on the testing set, this is known as overfitting. The practice of data augmentation is one way to avoid overfitting [44]. This method multiplies the original image to increase the size of the training set. The likelihood of overfitting can be decreased by enlarging the dataset. This part will present earlier research that we may use for our study that uses various data augmentation approaches to enhance performance.

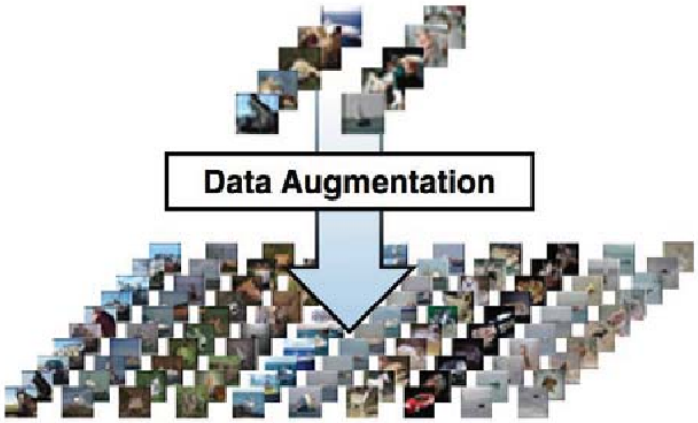


Figure 3.4: Data Augmentation

3.2.2 Visual Geometry Group (VGG-16)

In 2014, VGG-16 was one of the first CNN-based models presented. This architecture won the first ImageNet competition. In the previously suggested Alexnet architecture, the kernel size filter was changed from 11 to 5 of size 3x3. They have 13 convolutional layers and three thick layers in their structure [47]. This Architecture map can be seen in figure 3.5.

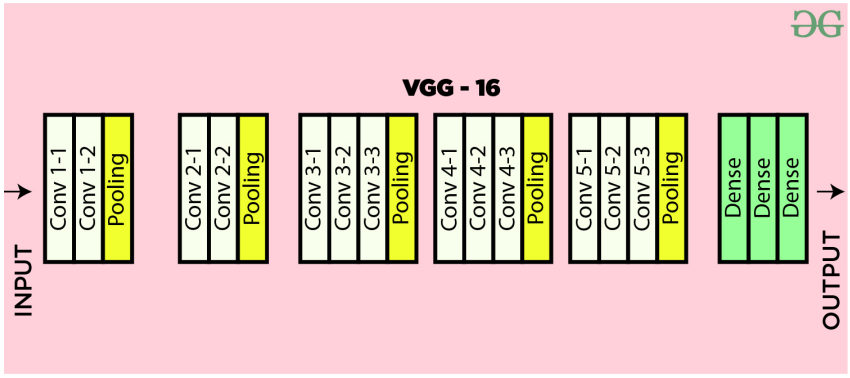


Figure 3.5: VGG-16 architecture map. Retrieved from GeeksforGeeks [36]

3.2.3 Residual Network

The residual relation is used in the ResNet residual network model. This is one of the most often used models for sorting, segmenting, and identifying targets. In 2015, this model won the ImageNet competition [20].

The deep learning model had a vanishing gradient problem before the ResNet model, which produced unstable behaviour while training a deep neural network. To tackle this problem, skip connection is a common module that is used in several convolutional architectures. Using a skip connection, we provide an alternative path for the gradient during backpropagation. Most ResNet models use double to triple layer skips when dealing with nonlinearities like batch normalisation and ReLU [1]. Because models like VGG have been known to decline after a certain depth, ResNet can flow backwards from later layers to early filters through this module (Figure 3.6).

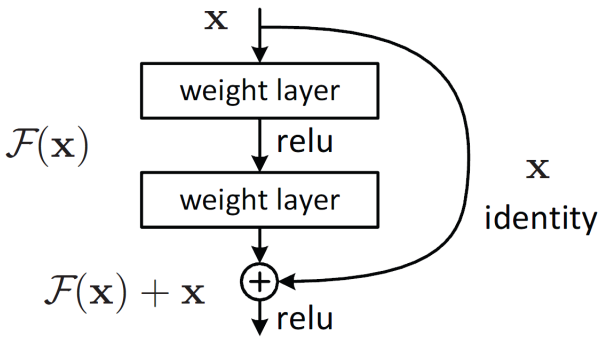


Figure 3.6: Residual Building Block [17]

3.2.4 Inception V3 (GoogleNet)

Inception V3 is the third Deep Learning Convolutional Architecture from Google. Although the original ImageNet dataset has over 1 million training pictures, the Tensorflow version has 1,001 classes due to a background class not included in the original ImageNet. Inception V3 was created for ImageNet's Large Visual Recognition Challenge, where it came in second place [21].

The model's symmetric and asymmetric construction components include convolutions, average pooling, max pooling, concatenations, dropouts, and ultimately connected layers. The activation inputs are batch normalised and used frequently throughout the model. Loss is calculated using Softmax [16].

A high-level depiction of the model is shown in the following figure 3.7.

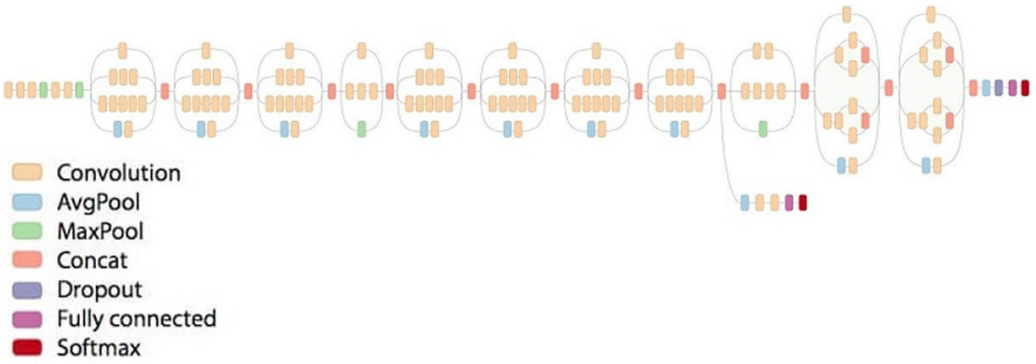


Figure 3.7: Inception v3 architecture. Retrieved from Intel [21]

3.2.5 MobileNet V2

This (fig 3.8) is the best deep learning network architecture already existing, according to Sandler et al. [18]. It was constructed to work in a resource-constrained setting. The network employs extension over the bottleneck layer with in-remaining connections inverted. As opposed to typical residual connections, which join two extended units, reversed in this instance. The connection between thin blocks is referred to as a residual connection. The model expands on the previously proposed width multiplier notion. In MobileNet-V1, which addresses the potential for representation, A pixel's multiple interests are covered by its "d" dimensions (d) = number of channels) in less dimensional space. Hence, breadth until the whole activation space is reduced, multiplier. There are many different areas of interest in space. The presence of ReLu, which uses to erase all negative values without degrading performance, supports this idea. This behaviour supports the idea that dimensionality reduction should not impact the outcomes even if the width multiplier had just excluded the values that would have produced negative input to ReLu. The two types of blocks in the network are shown in figure 3.8, one with a residual connection and a stride of 1 and the other without a residual connection and a

stride of 2. Each block combines a depthwise convolution (filter size 3X3) layer, a pointwise convolution (1X1) layer, and a Relu6 layer for pointwise convolution.

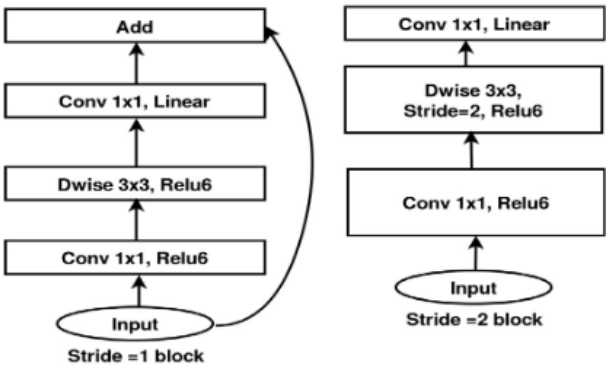


Figure 3.8: MobileNet_V2 architecture. Retrieved from [18]

3.2.6 NasNet

In order to attain the highest performance on a particular job, Neural Architecture Search (NAS) automates the construction of the topology of neural networks. The objective is to create the architecture with few resources and little human involvement [42]. It operates on a set of preset operations (such as convolutional layers, recurrent, pooling, fully connected, etc.) and their connections that make up the search space of potential network topologies. A controller next selects the search space from a list of potential candidate designs. They are trained and rated based on how well the candidate architectures perform on the validation test. The search is readjusted, and fresh candidates are found using the rating. The procedure repeats until a specific condition is reached, producing the ideal architecture. On the test set, the ideal architecture is assessed.

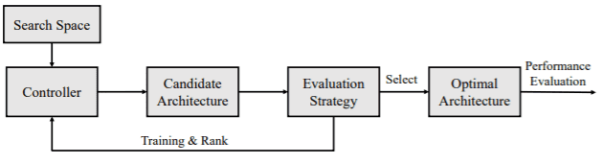


Figure 3.9: NasNet architecture. Retrieved from [42]

3.2.7 Training a Neural Network

Activation Function

The activation function determines when a neuron is activated. The output then becomes the input of the downstream neurons in the following network layer or outputs the result in the last neuron layer. In the case of continuous activation functions, one uses those with an s-shaped course, so-called sigmoid functions. This sigmoid, S-shaped, monotonically increasing, and differentiable logistic function is often used as the activation function. The value range of the function is between 0 and 1. Therefore, the sigmoid activation function is differentiable. Moreover, the first derivative can be determined very quickly analytically because this can be calculated from the original function [3].

Loss Function and Backpropagation

After the output has been calculated based on the weights and the activation function, it needs to be assessed. This is done by calculating the loss, which is the difference between the desired output and the current output. In other words, the loss function tells us how accurately our neural network makes predictions for a given input. Based on the loss, the weights are adjusted so that the model can make a better prediction in the next run. The goal is to minimize it throughout training [3].

Usually, weights and biases are initially set to random values, which often result in a high loss value when starting to train a neural network. The algorithm then adjusts each weight to minimize the difference between the calculated and correct values. The term "backpropagation" comes from the fact that the algorithm goes back and adjusts the weights and biases after the result has been calculated. The smaller the loss for a network, the more accurate it becomes. The learning process can thus be quantified as minimizing the output of the loss function. The type of error minimization is a gradient descent method that minimizes the mean square error. A parameter sets the speed of the procedure called the learning rate [15].

Regularization and Dropout

The dropout technique is a simple way to reduce the network's complexity during training. Here, a previously defined number of neurons in a layer are randomly deactivated, i.e. they "drop out". So, these units fail during a training epoch session. Doing so keeps the units from co-adapting and depending too much on some connections. This can leads the model to learn multiple independent representations and to become less sensitive to the specific weights of each neuron. The probability of keeping each node is determined randomly. The dropout rate at which individual neurons are temporarily deactivated is specified as a hyperparameter [49]. Another possible regularization is based on the error function of the network to be minimized and thus becomes a natural part of the learning procedure. For this purpose, a penalty term or complexity term is introduced.

Epochs and Batches

After all the weights have been adjusted, another pass of all input data is made. The error is measured again, and the backward propagation of this error is to adjust the weights again. A complete run of all input data is called an epoch. The number of training epochs is an important hyperparameter for neural network training. Depending on the size of the data set, the input data can also be divided into groups of equal size (batches), and the training can be performed per batch. It could be helpful, for example, to let an artificial neural network learn faster or to comply with the limitations of the computing capacity of the executing computer. When splitting into batches, it is essential to have a normal distribution of the values within each batch compared to the entire data set. Once all batches have passed through the neural network, an epoch is complete [5].

Batch Normalization

This is a normalization procedure in which the output of a layer is normalized before the activation function. So that the mean value is close to 0 and the standard deviation is close to 1. The normalization is calculated batch-wise during training and later determined mean values during training are used. Especially in the context of sigmoid as an activation function,

batch normalization is particularly well suited since the derivative of the sigmoid function has the highest value with y= 0.25 at the gradient descent at x= 0 and takes on smaller and smaller values as the deviation from 0 increases. Batch normalization solves the problem of the vanishing gradient to a certain degree. However, it uses dropout superfluous because batch normalization achieves a network regulation. Randomly composed batches will show different training examples; thus, the network will not provide a deterministic value for a single training example. In addition, an acceleration of the training is achieved because of batch normalization. The layers during the training have a minor internal covariate shift [3].

Optimization Algorithm

The Batch Gradient Descent method is also called the Vanilla Gradient Descent because of its simplicity. It calculates the gradients of the cost function for the entire training data and then updates the network parameters. This is also called offline learning. However, this method is unsuitable if vast amounts of data have to be processed because the amount of data may not fit into the memory space.

On the other hand, the Stochastic Gradient Descent (SGD) method updates the parameters in each training step after calculating the error or loss function's gradients. As a distinction from the first variant, this method is called Online Learning. New training data can be added relatively easily. Furthermore, the training is usually much faster than with batch gradient descent. However, due to a large number of updates, large fluctuations can occur, making the method's convergence much more difficult [15].

A solution to this is the third alternative, which combines the best of both worlds: The method Mini-Batch Gradient Descent divides the training data into several sets and then performs the Batch Gradient Descent for each of these sets. The disadvantage of this method is that another parameter is added as a so-called hyperparameter, namely the size of the Mini-Batch subsets. Depending on the application, the sizes 32, 64, 128, or 256 are often used, i.e. a multiple of the number 2, following the low-level data processing in processors (CPU or GPU). The idea behind adaptive algorithms is that in accordance with the parameters, one should adjust the learning rate, updating more frequently for frequent values and less frequently for uncommon parameters. For this reason, it is particularly well suited for dealing with sparse data.

The main idea behind Adagrad is to downscale the gradient vector. In other words, the learning rate per dimension gradually decays. With steeper dimensions, it decays faster. A small change in the AdaGrad algorithm resulted in the RMSProp method. Exponential decay of the sk obtains the only difference; thus, the algorithm changes too. Here, only the gradients of the last iterations are accumulated now, and not all gradients since the beginning of training. The decay rate is mostly initialized to 0.9. That is why RMSProp converges faster than AdaGrad [15]. Another optimizer is called Adam. It is the abbreviation for Adaptive Moment Estimation. It is a combination of the previous gradients (Momentum Optimizer) and the average of the squared gradients (RMSProp), both under exponential decay [43].

3.3 Single Shot Detector

In order to extract high-level features, the primary network of the Single Shot Detector (SSD) employs a pre-trained VGG16 network on ImageNet. This network is truncated before the last

classification layer, and Fully-Conected Layer 6 and Fully-Conected Layer 7 are changed to convolutional layers. By including additional convolution feature layers with successively lower resolutions—Conv8 2, Conv9 2, Conv10 2, and Conv11 2—SSD then expands this primary network [4].

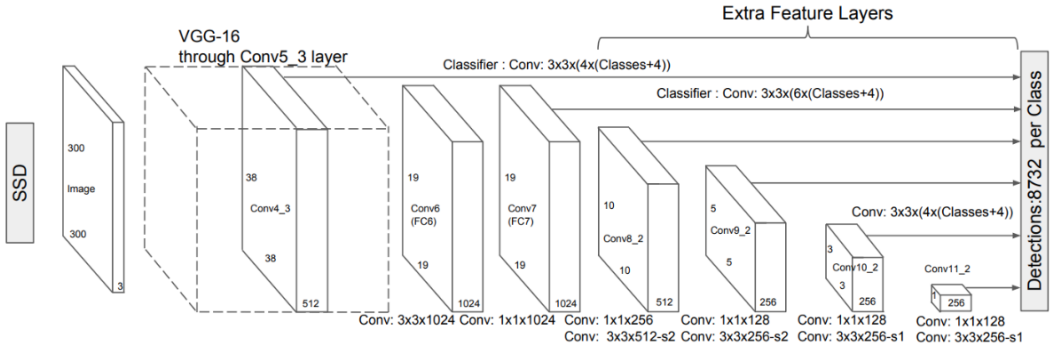


Figure 3.10: SSD Architecture. Retrieved From [29]

The SSD approach creates bounding boxes on a fixed-size collection and the object class instances is scored by the presence in those boxes using a feed-forward convolutional network. In the last phase, a non-maximum suppression step generates the final detections. The initial network layers are built on a widely used method for high-quality photo categorization (truncated before any classification layers) [29].

Multi-Scale Feature Maps

To detect objects of different scales, SSD predicts offsets and confidence scores over a variety of feature maps with varying resolutions. More specifically, lower resolution feature maps identify larger objects in the image, in part because semantic features of small-scale objects are lost in these upper layers [4]. As an illustration, it could be seen in Fig. 3.11, the 8 x 8 feature map (b) recognizes the cat, the smaller thing in picture (a), whereas the lower resolution 4 x 4 feature map (c) recognizes the dog, the larger object.

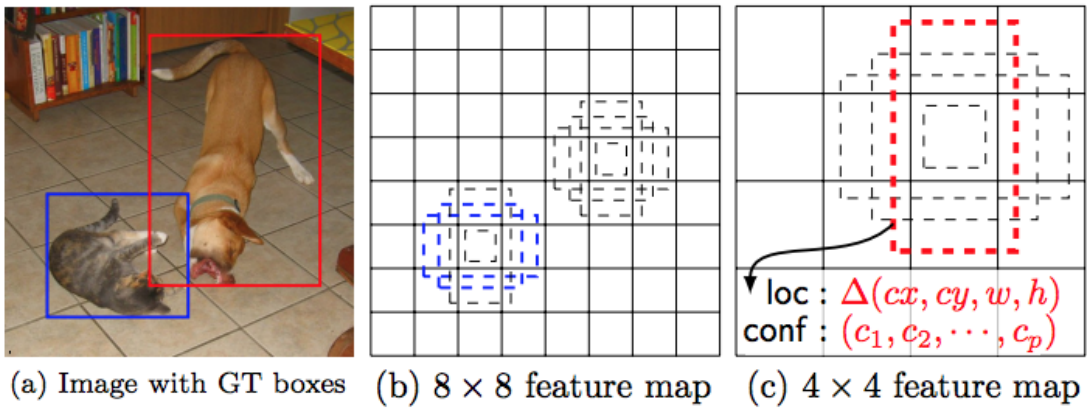


Figure 3.11: Multi-Scale Feature Maps. Retrieved From [29]

Chapter 4

Implementations

4.1 System Process

The application system process is divided into three steps. First, we detect and track face(s) using the face detector models proposed above on a real-time camera, or the user can upload a video file onto the application. The detected face will then be bounded with a rectangular box. The bounding boxes will be resized to the desired input size for feeding them into the CNN model. Next step, the CNN model job is to classify whether the face(s) detected is wearing a face mask or not. Finally, while the CNN model is classifying the face mask, the age and gender predictor will predict the approximate age and gender of the face.

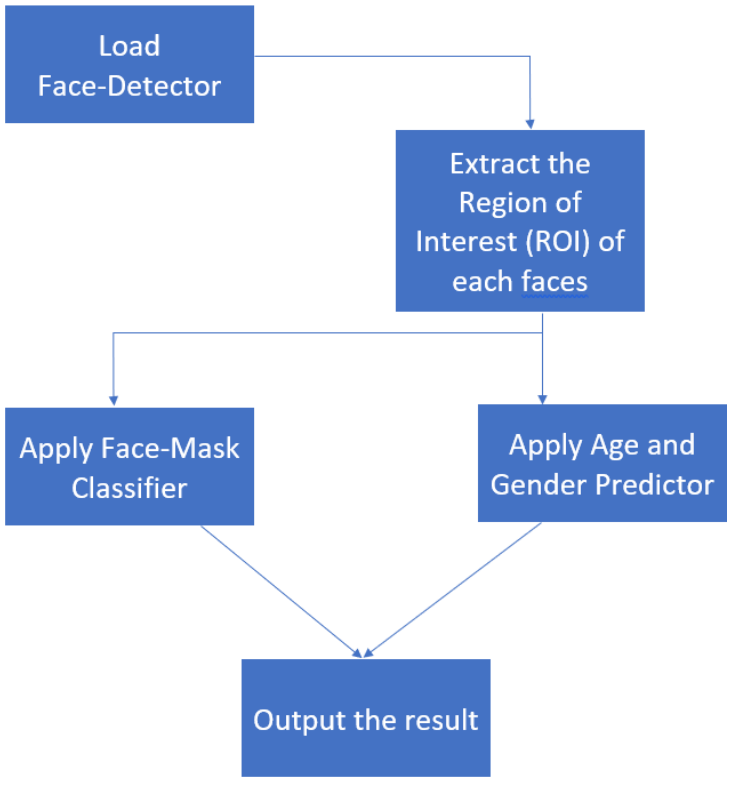


Figure 4.1: The Process of the System

4.2 Face Detector

As stated above, we used the Haar cascade frontal face detector and the DNN Caffe model face detector for our face detection task. The face detector model was loaded, and real-time video from the system was captured using the open-cv library. The picture frame's region of interest (ROI) is extracted. Then, comparisons and performances for each face detector are made in order to find the best face detector to be used on our application system.

4.2.1 Haar-Cascade Classifier

The Haar-Cascade Classifier is obtained from a github repository that is provided by the OpenCV team. Pre-trained Haar cascades are kept in a repository by the OpenCV library. Most of these Haar cascades are employed in identifying faces, eyes, mouths, and whole or partial bodies. For this project, we will be using the pre-trained Haar-Cascade model called haarcascade_frontalface_default.xml. Available at: `https://github.com/opencv/opencv/tree/master/data/haarcascades` [54].

4.2.2 Single Shot Multibox Detector

The Single shot multibox detector is obtained from a github repository that is provided by the OpenCV team and made publicly available. This model is called res 10 300x300 SSD, it is a DNN for face detection. The OpenCV team provided this file at `https://github.com/opencv/opencv_3rdparty/blob/dnn_samples_face_detector_20170830/res10_300x300_ssd_iter_140000.caffemodel` [55]. This Caffe model was trained with the SSD framework [29]).

4.3 Face-Mask Classifier

For the face-mask classifier, we will use pre-trained CNN models proposed above, such as VGG-16, ResNet-50, MobileNet_V2, Inception_V3 (GoogleNet), and NasNet. In order to use our pre-trained models to process the input and classify them to show the output we want, we will need to tune the hyperparameter and train them with our dataset.

4.3.1 The Dataset

The Datasets used in this project are retrieved from Kaggle. The datasets used:

- Kaggle, Face Mask Detection 12K Images Dataset [22]: this Face Mask Detection Classification dataset is made available by Ashish Jangra. The collection contains around 12,000 photos totaling 328.92MB in size. The photographs with the face mask (6,000) were scraped from Google, while the images without the face mask were preprocessed using Jessica Li's CelebFace dataset (https://www.kaggle.com/jessicali9530). This dataset consists of 3 folders: Test set (with mask: 483, without mask: 509), Train set (with mask: 5,000, without mask: 5,000), and Validation set (with mask: 400, without mask: 400). This dataset is available at `https://www.kaggle.com/datasets/ashishjangra27/face-mask-12k-images-dataset`.
- Kaggle, Face Mask Detection [33]: This dataset provides 853 images, which includes 3 classes: With mask, Without mask, and Mask worn incorrectly. In this project, only

With mask and Withouth mask dataset folder is used. The dataset is available at `https://makeml.app/datasets/mask`.

Both datasets are then mixed and picked manually to obtain better dataset. The final dataset consists of 2 directories: WithMask, and WithoutMask. More details could be seen in table 4.1.

	2k_newDataset
With mask	1000
Without mask	1000

Table 4.1: Final datasets' details.

4.3.2 Data Augmentation

Before entering the CNN network, we use a data augmentation technique to create images that are as natural-looking as possible to increase the training set's size. Additionally, it is a part of data pre-processing. Below are some of the setups we used for data augmentation: Initially, we set the rotation range to 20 degrees to randomly rotate the photos. Second, we use a zoom_range, randomly zooming in on 15% of the images. Following that, we set the width shift range and height shift range to be within 0.2 of each other to translate pictures vertically and horizontally randomly. Then, we utilise the shear_range, which applies shearing transformations at a random rate of 15%. Finally, the horizontal flip is set as "True, " meaning we will randomly flip half of the photos horizontally (can be seen in fig 4.2).

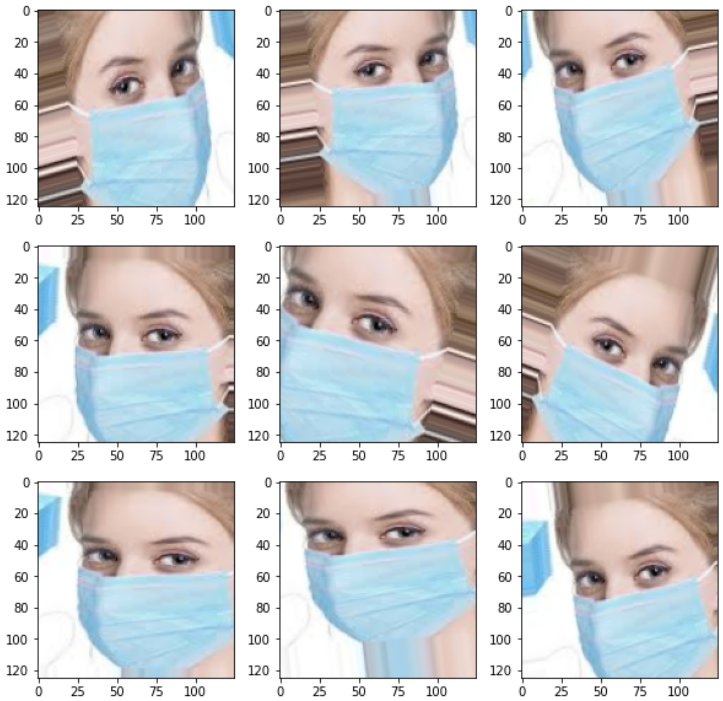


Figure 4.2: Data Augmentation Example

A validation set, a sample of data kept from training our model and used to assess model competence while adjusting the model's hyperparameters, was allotted 20% of the training

set in our project. The validation dataset, which is different from the test dataset and is not utilized for model training, is used to offer an objective assessment of the skill of the final tuned model for comparing or choosing amongst final models.

4.3.3 Training the pre-trained CNN Models

CNN Layers

VGG-16, ResNet-50, MobileNet_V2, Inception_V3 (GoogleNet), and NasNet will be used for the face-mask classification. The architecture of the models used are mostly default layers with some modification made on the input size and the network layers.

The Input size of each models:

	input size
VGG-16	224 x 224 x 3
ResNet-50	224 x 224 x 3
MobileNet_V2	224 x 224 x 3
Inception_V3 (GoogleNet)	224 x 224 x 3
NasNet	331 x 331 x 3

Table 4.2: Input size of the models

Little modifications made applied to all the models. As could be seen in fig 4.3:

```
model = base_model.output
model = tf.keras.layers.GlobalAveragePooling2D()(model)
model = Flatten(name="flatten")(model)
model = Dense(128, activation="relu")(model)
model = Dropout(0.5)(model)
model = Dense(2, activation="softmax")(model)
```

Figure 4.3: The code for modification applied to the architecture of the models.

The flatten layer connects to 128 dense layer with ReLu activation function, and since there are two classes to categorize, our project uses Softmax as the output activation function. According to Géron [12], the Softmax function will ensure that all probabilities are between 0 and 1 and add up to 1. For the categorization job, Softmax is specifically set up to produce N values for each class. The results are then normalized using the Softmax function, which changes the weighted total values into probabilities that add up to 1. The chance that a certain class would include each value in the output of the Softmax function is represented by each value.

Setting Up the Parameter

The training setup is listed below:

- Trained on Google Colab with a specification of GPU 1x Tesla K80, 12 Gigabyte of RAM, and 2496 CUDA cores.

- Python programming language with an open source Tensorflow and Keras network with pre-trained weights from ImageNet.
- 30 epochs, batch size of 32 and a Dropout of 0.5 is applied.
- Adam optimizer with learning rate and weight decay of 0.0001.
- Validation loss calculated using binary cross-entropy.

Adam Optimizer

Adam (Adaptive Moment Estimation) is an optimization methodology that may be used to repeatedly update network weights based on training data as opposed to the traditional stochastic gradient descent method. It is a combination of AdaGrad and RMSprop. RMSprop with momentum derives its parameter updates from momentum on the rescaled gradient as opposed to Adam updates, which are derived directly using a running average of the first and second moments of the gradient [26]. AdaGrad maintains a per-parameter learning rate that improves efficiency when dealing with scenarios with sparse gradients. The correction is accomplished by decreasing the gradient vector along the steepest dimensions [12]. Since 0.001 would cause our algorithm to learn too quickly and prevent convergence, we changed Adam's default learning rate to 0.0001. The optimal learning rate for our dataset and algorithms is thus 0.0001.

Binary Cross-Entropy Loss

A loss function that works well for binary classification tasks, such as dog vs cat, 0 or 1, with mask or without mask, is the Binary cross-entropy loss function. Additionally, it is the inverse of the logarithmic probabilities for each class label. Based on the projected probability for each class, the objective is to determine the difference between the actual and expected value. This loss function's formula may be expressed as follows:

$$L(\hat{y}, y) = \frac{1}{N}\sum_{i=1}^{N} -(ylog(p) + (1-y)log(1-p)) \tag{4.1}$$

P can be seen as the likelihood of class 1, and (1-p) as the likelihood of class 0. When the observation equals 1, the first half of the function will be triggered, and the second part will be omitted. The opposite will also occur. The most typical loss function for binary classification problems is binary cross-entropy.

4.3.4 Evaluation of the CNN Models

In general, the following matrices use to evaluate the performance of Face Mask Detection Images; These matrices are calculated from the confusion matrix, such as precision in equation 4.2, recall in equation 4.3, F1-score in equation 4.4. The confusion matrix consists of four types of parameters where TP stands for True Positive, FP for False positives, TN for True negatives, and FN for false negatives, shown in table 4.3, and the equations are below [7].

	Positive	Negative
Positive	True Positive (TP)	False Positive (FP)
Negative	True Negative (TN)	False Negative (FN)

Table 4.3: Confusion Matrix

$$Precision = \frac{TP}{TP + FP} \tag{4.2}$$

$$Recall = \frac{TP}{TP + FN} \tag{4.3}$$

$$F1 - Score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{4.4}$$

When determining whether or if our model is reliable enough, precision reveals how many correctly predicted situations were optimistic. However, accuracy and precision are not the same things. In contrast to precision, which deals with comparable observations of the same item, accuracy measures how close a statistic is to the actual or acceptable value. The recall is the proportion of real positive cases that our model accurately anticipated. Recall often reduces when accuracy rises, and vice versa. A harmonic mean of recall and accuracy is the F1-score. As a result, it offers a combined image of these two metrics. Furthermore, the F1-score is maximized when accuracy and recall are identical [7].

4.4 Age and Gender Prediction

For the age and gender predictor model, we will use a CNN model proposed by Gil Levi and Tal Hassner, made publicly available in the GitHub repository at `https://github.com/GilLevi/AgeGenderDeepLearning`. They try to bridge the gap between automatic facial recognition and techniques for estimating age and gender. They achieve this by following the model successfully set by current facial recognition systems: Deep convolutional neural networks (CNNs) may be used to make significant advancements in face recognition algorithms, as demonstrated in recent years [27]. They show comparable benefits using a simple network architecture created with the sparse availability of exact age and gender labels in current face data sets in mind.

They tested their network using the recently announced Adience 1 standard for identifying the age and gender of unfiltered facial photos [9]. They demonstrate that, despite the Adience set's challenging pictures and straightforward network design, their approach surpasses the current state of the art by wide margins. The difficulty of reliably assessing age and gender in unrestricted situations, as shown by the Adience photos, remains unresolved even though these results offer an excellent baseline for deep-learning-based systems. This is because they allow potential for improvement through more complex system designs. They open their trained models and classification system to the public to provide future researchers with a foundation to build more powerful techniques. Please visit this page for further details: `www.openu.ac.il/home/hassner/projects/cnn_agegender`.

Chapter 5

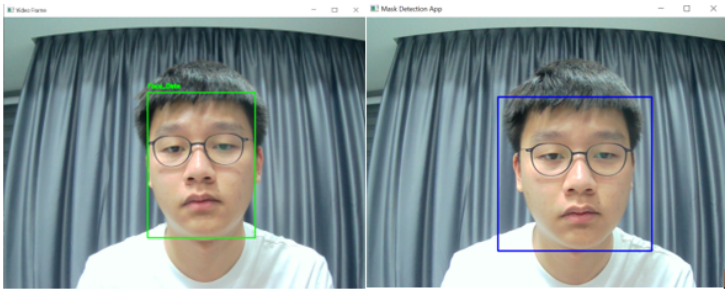
Experiments and Results

We will comparatively introduce our tests and results in this chapter by addressing our hypothesis using the algorithm proposed in chapter 3 and the implementations in chapter 4. We will first talk about the trials in our face detection models, CNN models for face-mask detection, and age and gender predictor performance, before moving on to the next chapter on creating the application.

5.1 Face Detector

5.1.1 Front-View Face

Both models did a great job detecting a person's front face, as seen in fig 5.1. Both models could detect the face quick and easy.



(a) (b)

Figure 5.1: Front view performance on: (a) Haar-Cascade Classifier (b) Caffe Model Face Detector

5.1.2 Side-View Face

Vertical head movements were examined by moving the head up and down within the camera's field of view, and side face view detections were tested by moving the head from left to right.

Haar-Cascade

Figure 5.2: The detection of a face from the side view and vertical movement (up and down) utilizing a Haar-Cascade detector.

Neither the side views nor the vertical head motions were monitored using this detector.

Caffe Model: Res_10 300 x 300 SSD

Figure 5.3: The detection of a face from the side view and vertical movement (up and down) utilizing a Caffe Model.

In side views, the face detector had no trouble detecting faces, as well as downward vertical and upward vertical face detection.

5.1.3 Comparing Results Among the Two Models

Haar-Cascade classifier on front face views were accurately recognized; however, favourable lighting and minimal to no head movements were needed. No detections were made for side face views or vertical head motions. Furthermore, The Caffe model face detector accurately identified most head movements. However, upward vertical motions could sometimes not be detected, depending on the lighting. The detector picked up side motions and provided accurate predictions of bounding boxes. The mask detector functions effectively in most lighting situations and is unaffected by slight occlusions.

5.2 CNN Models on Face-Mask Classification

5.2.1 CNN Models' Training Setup

Following are the settings used to perform experiments: 1. Experiments perform on Google Colab platform having a specification of GPU 1x Tesla K80, 12 Gigabyte of RAM, and 2496 CUDA cores. 2. Batch size of 32 3. Cyclic Learning rate from le-1 to le-6 with weight decay of le-4 4. Categorical Cross Entropy loss function. Binary Cross Entropy (BCE) for binary classification where a is the actual label, and b is the predicted label for calculating the loss between predicted and actual samples of m number of samples. 5. The number of epochs is set to 30

5.2.2 CNN Models' Results

The experimental results of the models' training and validation accuracy (fig 5.4), training and validation loss(fig 5.4), and ROC (fig 5.5) of VGG-16, ResNet-50, MobileNet_V2, GoogleNet, and NasNet can be seen below. VGG-16, ResNet-50, MobileNet_V2, GoogleNet, and NasNet achieved an accuracy of 97.25%, 98.75%, 98.25%, 99%, and 99.75% respectively.

Models' Training and Validation Performances

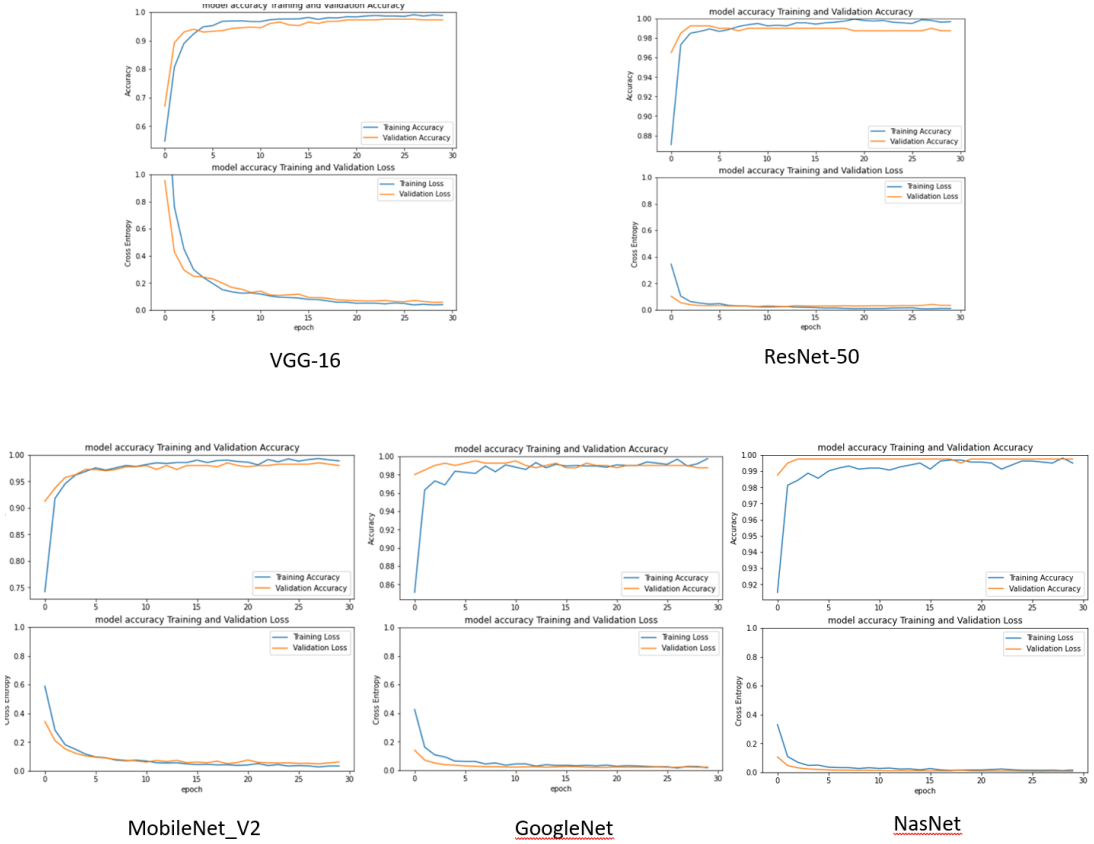


Figure 5.4: Models' training accuracy and validation accuracy: VGG-16, ResNet-50, MobileNet_V2, GoogleNet, and NasNet

It can be seen from above that the VGG-16 model converges rapidly until 15 epochs, and after that, the convergence becomes smooth's. In contrast, Resnet-50 and NasNet start converging from five epochs. While MobileNet and GoogleNet start converging around nine epochs.

ROC Curve

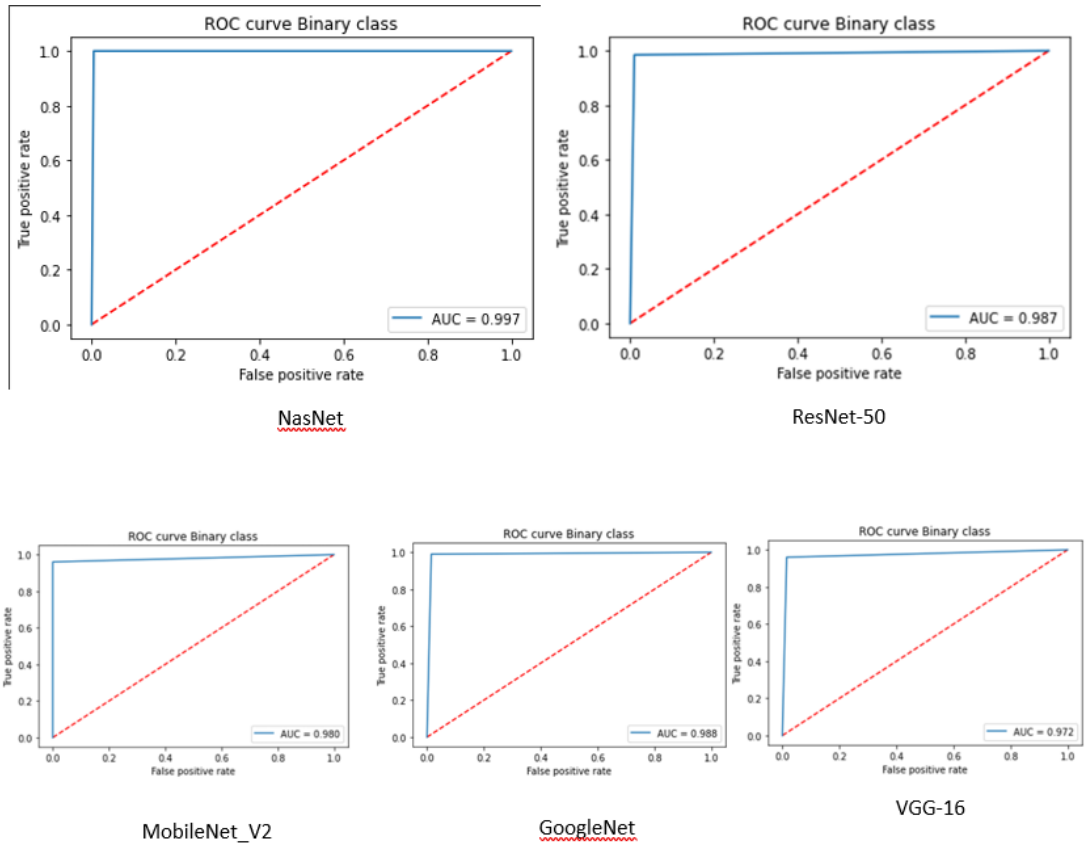


Figure 5.5: Models' ROC curve: VGG-16, ResNet-50, MobileNet_V2, GoogleNet, and NasNet

Results Comparison

Models	Accuracy	ROC
VGG-16	97.25%	97.2%
ResNet-50	98.75%	98.7%
MobileNet_V2	98.25%	98.0%
GoogleNet	99.0%	98.8%
NasNet	99.75%	99.7%

Table 5.1: Results Table Comparison

The results of those five models are shown in table 5.1. The NasNet model shows the highest validation accuracy with 99.75%. There are a few observations gathered from the result. Firstly the starting accuracy obtained from all the pre-trained models is high. This can be addressed by the nature of the pre-trained model, as they are previously trained with massive datasets; the model weights have already been optimized. Therefore, it shows a decent accuracy (>90%) from the starting epochs. Although NasNet is the only one with quite a significant overfitting problem on early epochs, it has the highest training and validation accuracy among the rest. Nonetheless, the other models' performances are outstanding as well (>98%).

Confusion Matrixes

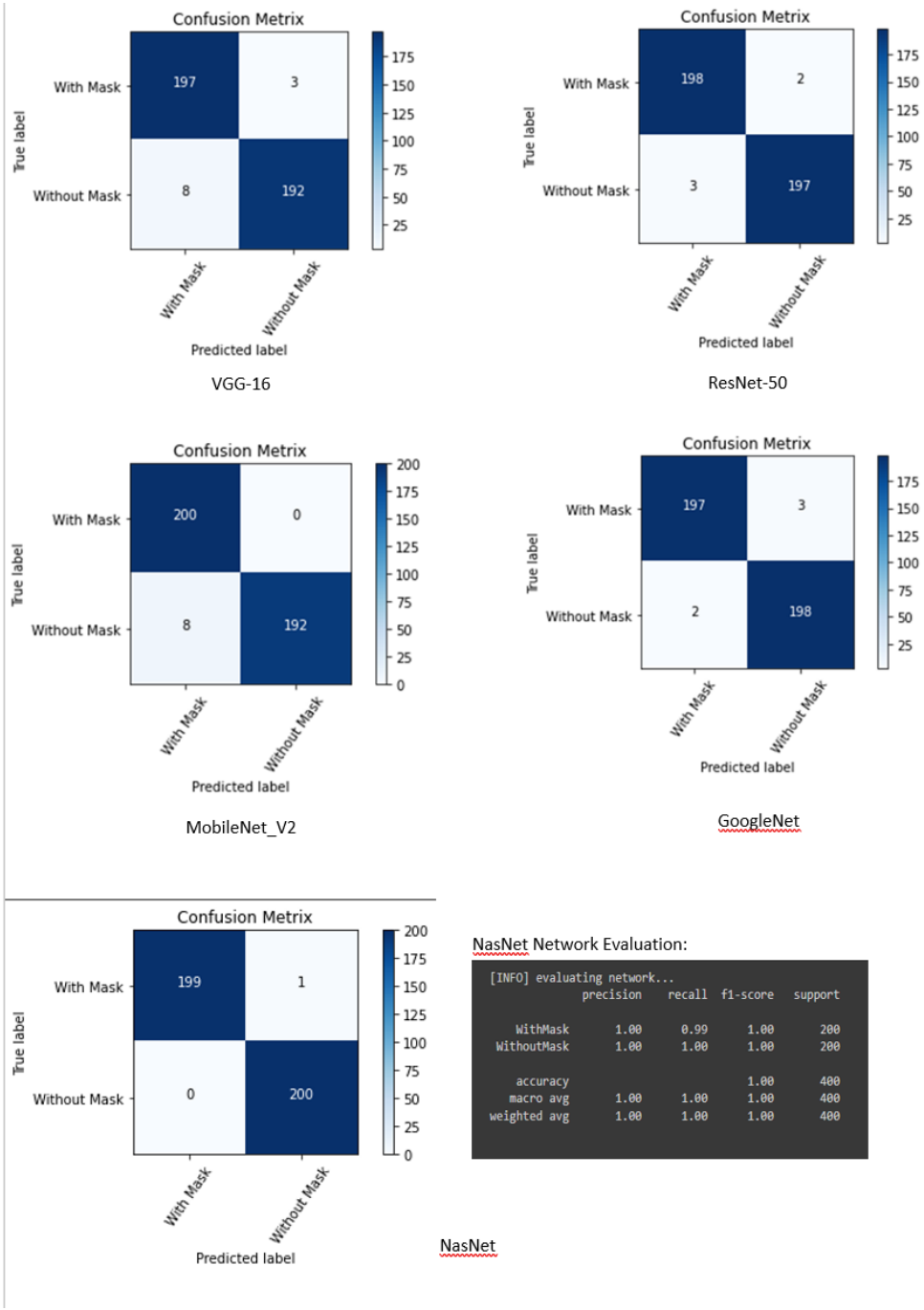


Figure 5.6: Models' Confusion Matrix

From the confusion matrixes figure 5.6 above, it seems that most models wrongly classified the classes without a mask. However, Nasnet only made one mistake in classifying a face without a mask when the true label is with a mask. Looking from all the performences evaluation above, it can be concluded quickly that NasNet outperforms the rest of the models.

5.2.3 Experiments on Real-Time Face-Mask Classification

Since we concluded that the Caffe model's performance is better than the Haar-Cascade, we decided to use the Caffe Model as a face detector for our application. The face detector output will then be resized before feeding it to our CNN models for face mask classification. We set the bounding box color to turn green for a mask detected on the face, and red for a mask not detected on the face. Above the bounding boxes, we also provided the confidence rate of each models having correct classification. Below we will provide some analysis and proof of the experiments.

Figure 5.7: Experiments on MobileNet_V2, GoogleNet, and NasNet shows really good results

As seen above (Fig 5.7), the experiments on MobileNet_V2, GoogleNet, and NasNet show excellent results. Those models could classify correctly and with a perfect confidence rate of 100%. However, upon experimenting on VGG-16 and ResNet-50, the results show that these

two models' performance is unstable. VGG-16 will always classify a person as wearing a mask, even though the person is not wearing one (can be seen on fig 5.8), and ResNet works the other way around (can be seen on fig 5.9). Although, it could be seen on VGG-16 output that when a person is wearing a face mask, it has a more confidence rate of 74.30% compared to the person not wearing a face mask but detected as wearing one with a confidence rate of 57.57%.

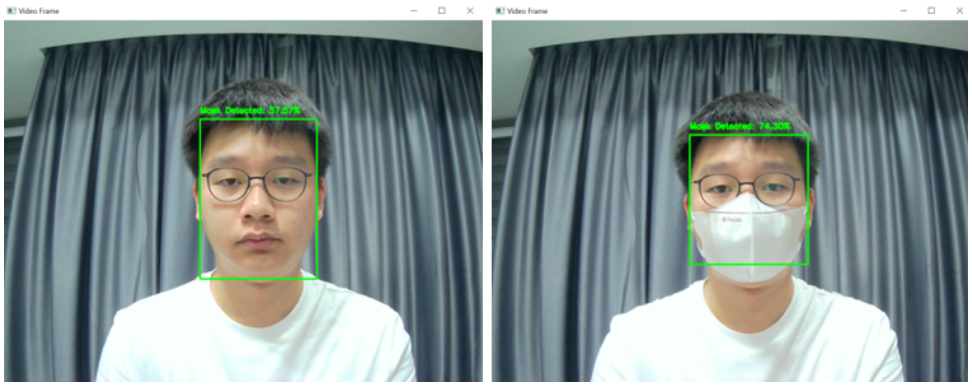


Figure 5.8: Experiments on VGG-16

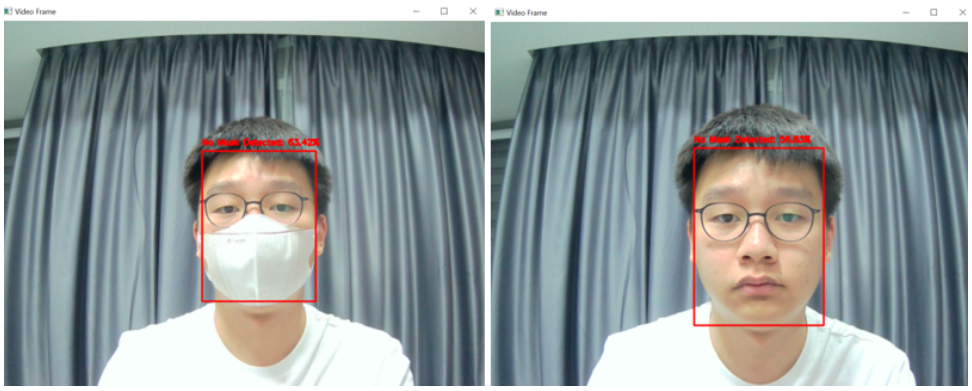


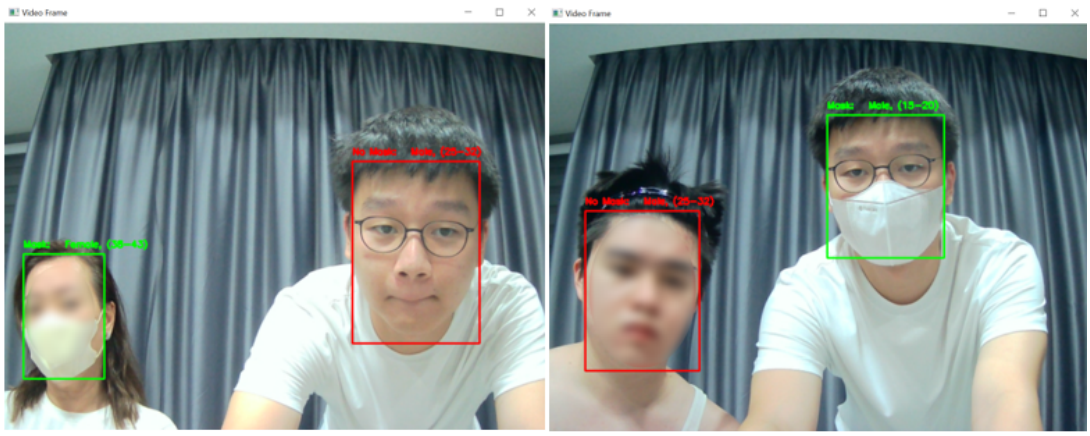
Figure 5.9: Experiments on ResNet-50

This might be happening because MobileNet_V2, GoogleNet, and NasNet models have the convolution network working in parallel and capturing local and global information. The idea is that they convolve in different parallel sizes, from the most accurate detailing (1x1) to the bigger one (5x5). Furthermore, the network performance depthwise and pointwise could extract more fine-grained information, which does not seems to work the same for networks such as VGG-16 and ResNet-50 models. These networks, such as MobileNet_V2, GoogleNet, and NasNet have more complicated layers and training parameters. That is why they are well-suited in Realtime and on video frames.

5.3 Age and Gender Predictor

Since this is an extension of the objective, some small experiments were made on this age and gender predictor. From all the experiments and results above, we have decided to use Caffe model as the face detector, and NasNet as the face-mask classifier. The face-mask classifier

will work side-by-side with this age and gender predictor. The results can be seen below (Fig 5.10).



(a) (b)

Figure 5.10: Age and Gender predictor working side by side with face detector(Caffe model), and face-mask classifier(NasNet)

The age and gender predictor was tried on a female in the age of 48 (fig5.10(a)) and a male in the age 27 (fig5.10(b)). Although, there are some mistakes made on estimating the age of the female and myself, the error rate is only around 2-5 years. The gender predictor is working fine.

5.4 The System Application

We are applying all three models into a system application made by using tkinter. It is a simple user interface made for the purpose of easier usage for the users (Fig 5.11). The user interface have three options: Live Stream for real-time classification using webcam provided by own devices, Video Stream enables user to upload a video file for classification, and Exit for exiting the application. It can be further redesign into a more interesting and exciting graphical interface.

Figure 5.11: UI

Chapter 6

Conclusion & Future Work

6.1 Conclusion

The COVID-19 disease is causing a problem for societies and economies worldwide, culminating in a global health crisis [30]. Due to this situation, more people are concerned about their health conditions. Also, the government is trying to prevent the spread of COVID-19 disease and make the public aware of how to stay healthy. Leung et al. [28] shown that the use of surgical masks helps to prevent the spread of COVID -19 diseases. This project aims to create an application to help the government track and analyse people's behaviour upon using face-mask to prevent the spread of COVID-19 disease. In this application, we have divided the system working into several steps. First, we detect and track face(s) using the face detector models proposed above on a real-time camera, or the user can upload a video file onto the application. The detected face will then be bounded with a rectangular box. The bounding boxes will be resized to the desired input size for feeding them into the CNN model. Next step, the CNN model job is to classify whether the face(s) detected is wearing a face mask or not. Finally, while the CNN model is classifying the face mask, the age and gender predictor will predict the approximate age and gender of the face.

We have compared the face detector models (Single Shot Multibox Detector and Haar-Cascade Classifier) in real-time in some everyday situations where face detection might become tricky and presented the results above. We can conclude that the Caffe model performs better when detecting a person's face's side view than the Haar-Cascade Classifier. While for the face-mask classifier, upon comparing all the models' performance, Nasnet outperforms the rest of the models with an accuracy of 99.75%. When applying all the CNN models on the application for real-time classification, NasNet, MobileNet_V2, GoogleNet work perfectly. Although VGG-16 and ResNet-50 seem to have unstable performance. Lastly, applying the age and gender predictor working side by side with the face detector and face-mask classifier seems to work well, with some minor errors on the age prediction (2-5 years error rate). We can conclude that the face-mask classifier performance we obtain from our proposed methodology works excellently, as it has an accuracy of 99.75% with only one miss-classification made on the validation dataset.

6.2 Future Work

The face-mask classification model can be improved for future work by increasing the classes with another label called "incorrect mask". As we experimented further with our face-mask classification model, we discovered that it classified a white napkin as a face mask (see figure on A.6 appendix). Increasing the classes with incorrect masks could help the government prevent people from trying to deceive the classifier.

Moreover, it seems that age and gender predictors can sometimes show the wrong prediction depending on the lighting of the video frame. After further experimenting with this predictor, we found out that the predictor seems biased when predicting the age of Asians. The predictor tends to wrongly estimate the age of an Asian person to be younger. Training a CNN model with a more diverse dataset could be helpful for this case.

Bibliography

[1] Nikolas Adaloglou. Intuitive explanation of skip connections in deep learning. (Accessed at: 03 May 2022). Available from: https://theaisummer.com/skip-connections/, 2020.

[2] Sirajdin Olagoke Adeshina, Haidi Ibrahim, Soo Siang Teoh, and Seng Chun Hoo. Custom face classification model for classroom using haar-like and lbp features with their performance comparisons. *Electronics*, 10(2):102, 2021.

[3] Charu C Aggarwal et al. Neural networks and deep learning. *Springer*, 10:978–3, 2018.

[4] Manal El Aidouni. Ssd : Understanding single shot object detection. (Accessed at: 17 August 2022). Available from: https://manalelaidouni.github.io/Single%20shot%20object%20detection.html, 2019.

[5] Francois Chollet. The limitations of deep learning. *Deep learning with Python*, 2017.

[6] Cs231n.github.io. Convolutional neural networks for visual recognition. (Accessed at: 28 April 2022). Available from: http://cs231n.github.io/convolutional-networks/, 2018.

[7] Xinyang Deng, Qi Liu, Yong Deng, and Sankaran Mahadevan. An improved method to construct basic probability assignment based on the confusion matrix for classification problem. *Information Sciences*, 340:250–261, 2016.

[8] Doxygen. Opencv: Cascade classifier. (Accessed at: 03 May 2022). Available from: https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html, 2022.

[9] Eran Eidinger, Roee Enbar, and Tal Hassner. Age and gender estimation of unfiltered faces. *IEEE Transactions on information forensics and security*, 9(12):2170–2179, 2014.

[10] Md Sabbir Ejaz, Md Rabiul Islam, Md Sifatullah, and Ananya Sarker. Implementation of principal component analysis on masked and non-masked face recognition. In *2019 1st international conference on advances in science, engineering and robotics technology (ICASERT)*, pages 1–5. IEEE, 2019.

[11] Yaqing Fang, Yiting Nie, and Marshare Penny. Transmission dynamics of the covid-19 outbreak and effectiveness of government interventions: A data-driven analysis. *Journal of medical virology*, 92(6):645–659, 2020.

[12] Aurélien Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. " O'Reilly Media, Inc.", 2019.

[13] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323. JMLR Workshop and Conference Proceedings, 2011.

[14] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Francis Bach. Deep learning (adaptive computation and machine learning series), 2017.

[15] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Y Bengio. Deep learning cambridge. *MS: The MIT Press. https://www. deeplearningbook. org*, 2016.

[16] GoogleCloud. Advanced guide to inception v3. (Accessed at: 01 May 2022). Available from: `https://cloud.google.com/tpu/docs/inception-v3-advanced`, 2022.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[18] Andrew Howard, Andrey Zhmoginov, Liang-Chieh Chen, Mark Sandler, and Menglong Zhu. Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. *CVPR*, 2018.

[19] Shaik Asif Hussain and Ahlam Salim Abdallah Al Balushi. A real time face emotion classification and recognition using deep learning model. In *Journal of Physics: Conference Series*, page 012087. IOP Publishing, 2020.

[20] ImageNet. Large scale visual recognition challenge 2015 (ilsvrc2015). (Accessed at: 30 April 2022). Available from: `https://image-net.org/challenges/LSVRC/2015/results`, 2015.

[21] Intel. Inception v3 deep convolutional architecture for classifying acute myeloid/lymphoblastic leukemia. (Accessed at: 01 May 2022). Available from: `https://www.intel.com/content/www/us/en/developer/articles/technical/inception-v3-deep-convolutional-architecture-for-classifying-acute-myeloidlymphob` `html`, February 17, 2019.

[22] Ashish Jangra. Face mask detection 12k images dataset. `https://www.kaggle.com/datasets/ashishjangra27/face-mask-12k-images-dataset`, 2020.

[23] F Kane. Machine learning vs. deep learning: What's the difference? (Accessed at: 28 April 2022), April, 2020.

[24] Q Ke and F Boussaid. Computer vision for assistive healthcare. (Accessed at: 28 April 2022), 2018.

[25] Muhammad Kamran Javed Khan, Nizam Ud Din, Seho Bae, and Juneho Yi. Interactive removal of microphone object in facial images. *Electronics*, 8(10):1115, 2019.

[26] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[27] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[28] Nancy HL Leung, Daniel KW Chu, Eunice YC Shiu, Kwok-Hung Chan, James J McDevitt, Benien JP Hau, Hui-Ling Yen, Yuguo Li, Dennis KM Ip, JS Peiris, et al. Respiratory virus shedding in exhaled breath and efficacy of face masks. *Nature medicine*, 26(5):676–680, 2020.

[29] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: single shot multibox detector. *CoRR*, abs/1512.02325, 2015.

[30] Mohamed Loey, Gunasekaran Manogaran, Mohamed Hamed N Taha, and Nour Eldeen M Khalifa. Measurement. In *"A hybrid deep transfer learning model with machine learning methods for face mask detection in the era of the COVID-19 pandemic", vol. 167, Article ID 108288*, 2021.

[31] Samiksha Malhotra, Vaibhav Aggarwal, Himanshu Mangal, Preeti Nagrath, and Rachna Jain. Comparison between attendance system implemented through haar cascade classifier and face recognition library. In *IOP Conference Series: Materials Science and Engineering*, page 012045. IOP Publishing, 2021.

[32] Siddharth Misra, Hao Li, and J He. Noninvasive fracture characterization based on the classification of sonic wave travel times. *Machine Learning for Subsurface Characterization*, pages 243–287, 2020.

[33] Make ML. Mask dataset. `https://makeml.app/datasets/mask`, May 18, 2020.

[34] Adrian Nieto-Rodriguez, Manuel Mucientes, and Victor M Brea. System for medical mask detection in the operating room through facial attributes. In *Iberian Conference on Pattern Recognition and Image Analysis*, pages 138–145. Springer, 2015.

[35] World Health Organization. Who coronavirus (covid-19) dashboard. Retrieved From `https://www.cnbc.com/2020/04/02/should-you-wear-a-face-mask-to-prevent-covid-19-doctors-weigh-in.html`, March 25, 2022.

[36] Pawangfg. Vgg-16 | cnn model. (Accessed at: 03 May 2022). Available from: `https://www.geeksforgeeks.org/vgg-16-cnn-model/`, 2020.

[37] Le Tran Huu Phuc, HyeJun Jeon, Nguyen Tam Nguyen Truong, Jung Jae Hak, et al. Applying the haar-cascade algorithm for detecting safety equipment in safety management systems for multiple working environments. *Electronics*, 8(10):1079, 2019.

[38] Adri Priadana and Muhammad Habibi. Face detection using haar cascades to filter selfie face image on instagram. In *2019 International Conference of Artificial Intelligence and Information Technology (ICAIIT)*, pages 6–9. IEEE, 2019.

[39] Bosheng Qin and Dongxiao Li. Identifying facemask-wearing condition using image super-resolution with classification network to prevent covid-19. *Sensors*, 20(18):5236, 2020.

[40] Bosheng Qin and Dongxiao Li. Identifying facemask-wearing condition using image super-resolution with classification network to prevent covid-19. *Sensors*, 20(18):5236, 2020.

[41] Moein Razavi, Hamed Alikhani, Vahid Janfaza, Benyamin Sadeghi, and Ehsan Alikhani. An

automatic system to monitor the physical distance and face mask wearing of construction workers in covid-19 pandemic. *SN computer science*, 3(1):1–8, 2022.

[42] Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Xiaojiang Chen, and Xin Wang. A comprehensive survey of neural architecture search: Challenges and solutions. *ACM Computing Surveys (CSUR)*, 54(4):1–34, 2021.

[43] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[44] Uday Shankar Shanthamallu, Andreas Spanias, Cihan Tepedelenlioglu, and Mike Stanley. A brief survey of machine learning methods and their sensor and iot applications. In *2017 8th International Conference on Information, Intelligence, Systems & Applications (IISA)*, pages 1–8. IEEE, 2017.

[45] Anirudha B Shetty, Bhoomika, Deeksha, Jeevan Rebeiro, and Ramyashree. Facial recognition using haar cascade and lbp classifiers. *Global Transitions Proceedings*, 2(2):330–335, 2021. International Conference on Computing System and its Applications (ICCSA-2021).

[46] Anirudha B Shetty, Jeevan Rebeiro, et al. Facial recognition using haar cascade and lbp classifiers. *Global Transitions Proceedings*, 2(2):330–335, 2021.

[47] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[48] Vandna Singh, Vinod Shokeen, and Bhupendra Singh. Face detection by haar cascade classifier with simple and complex backgrounds images using opencv implementation. *International Journal of Advanced Technology in Engineering and Science*, 1(12):33–38, 2013.

[49] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[50] Cory Stieg. What you need to know if you're considering a face mask, according to experts. `https://www.cnbc.com/2020/04/02/should-you-wear-a-face-mask-to-prevent-covid-19-doctors-weigh-in.html`, 2020.

[51] Mayur Surve, Priya Joshi, Sujata Jamadar, and Minakshi Vharkate. Automatic attendance system using face recognition technique. *International Journal of Recent Technology and Engineering (IJRTE)*, 9(1):2134–2138, 2020.

[52] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[53] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to data mining*. Pearson Education India, 2016.

[54] OpenCV Team. Haar-cascade pre-trained model. *OpenCV*, 2013.

[55] OpenCV Team. Dnn model for face detector sample. *OpenCV*, 2017.

[56] Dhananjay Theckedath and RR Sedamkar. Detecting affect states using vgg16, resnet50 and se-resnet50 networks. *SN Computer Science*, 1(2):1–7, 2020.

[57] Dhananjay Theckedath and RR Sedamkar. Detecting affect states using vgg16, resnet50 and se-resnet50 networks. *SN Computer Science*, 1(2):1–7, 2020.

[58] D. M. van Rijmenam, D. M. an Rijmenam. 7 steps to machine learning: How to prepare for an automated future. (Accessed at: 06 February 2022). Available from: `https://medium.com/dataseries/7-steps-to-machine-learning-how-toprepare-for-an-automated-future-78c7918cb35d`, 2019.

[59] Zekun Wang, Pengwei Wang, Peter C Louis, Lee E Wheless, and Yuankai Huo. Wearmask: Fast in-browser face mask detection with serverless edge computing for covid-19. *arXiv preprint arXiv:2101.00784*, 2021.

[60] Zhongyuan Wang, Guangcheng Wang, Baojin Huang, Zhangyang Xiong, Qi Hong, Hao Wu, Peng Yi, Kui Jiang, Nanxi Wang, Yingjiao Pei, et al. Masked face recognition dataset and application. *arXiv preprint arXiv:2003.09093*, 2020.

[61] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *Advances in neural information processing systems*, 27, 2014.

[62] Aston Zhang, Zachary C Lipton, Mu Li, and Alexander J Smola. Dive into deep learning. *arXiv preprint arXiv:2106.11342*, 2021.

[63] Xiongwei Zhang, Hager Saleh, Eman MG Younis, Radhya Sahal, and Abdelmgeid A Ali. Predicting coronavirus pandemic in real-time using machine learning and big data streaming system. *Complexity*, 2020, 2020.

Appendix A

Appendix

A.1 Project Plan

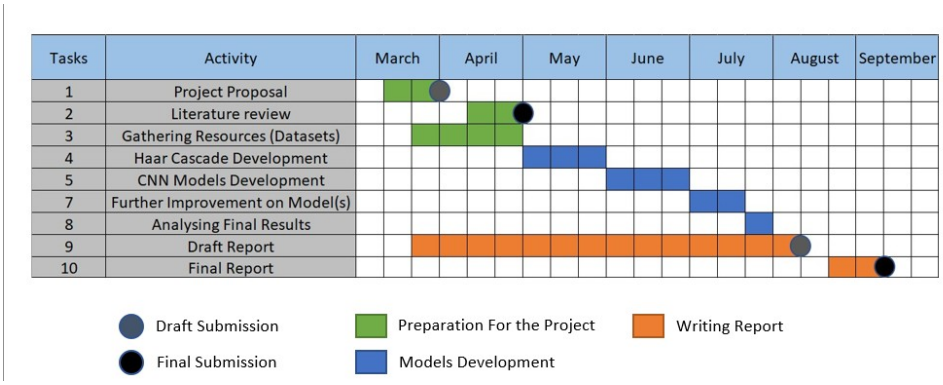


Figure A.1: Project Plan

As can be seen in figure A.1, this project will start with a project proposal, which will be submitted on 1st April 2022 for the supervisor's feedback. After the feedback from the supervisor, we will be able to start with the literature review for the report and gather the dataset we are planning to use for this project. The beginning of May is when we finally start implementing the proposed methods. The processes and results will be recorded for the report. While doing so, we should also start with writing the report. The draft report will then be submitted to the supervisor in August. Finally, after receiving feedback for the draft report, we can then continue finalizing the report for final submission, which is on 9th September 2022.

A.2 Further Experiments

VGG-16

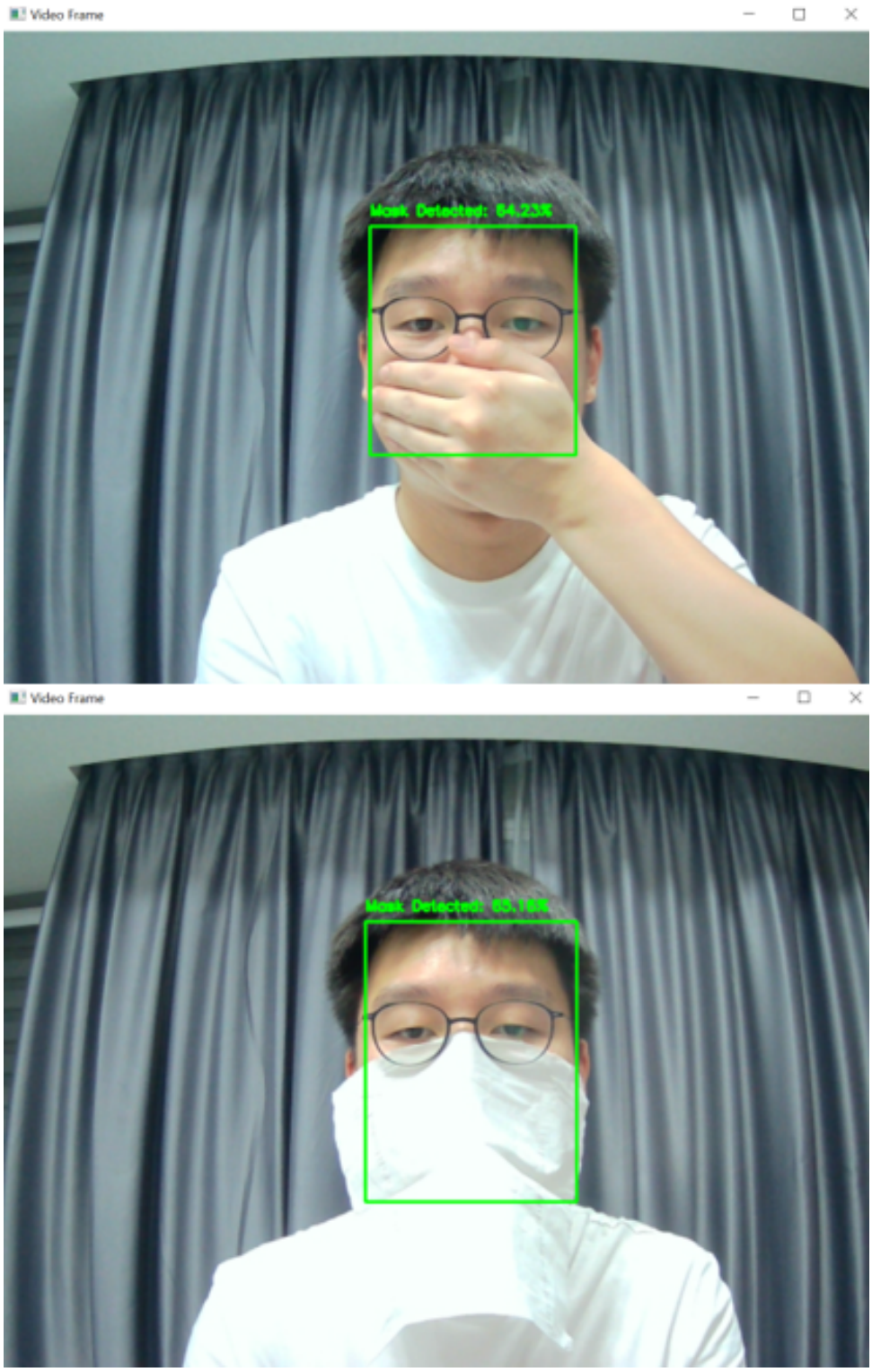


Figure A.2: Further Experiments with white napkin and hand

ResNet-50

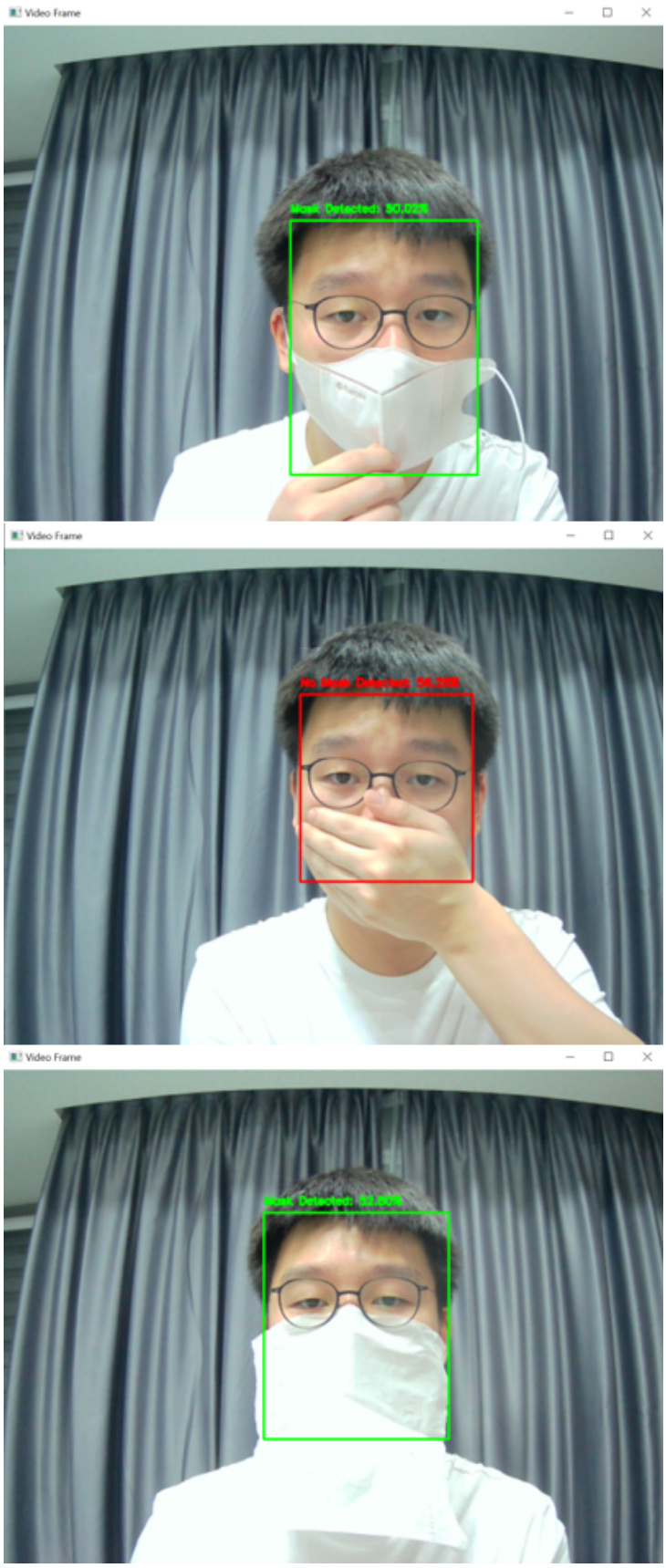


Figure A.3: Further Experiments with white napkin and hand

MobileNet_V2

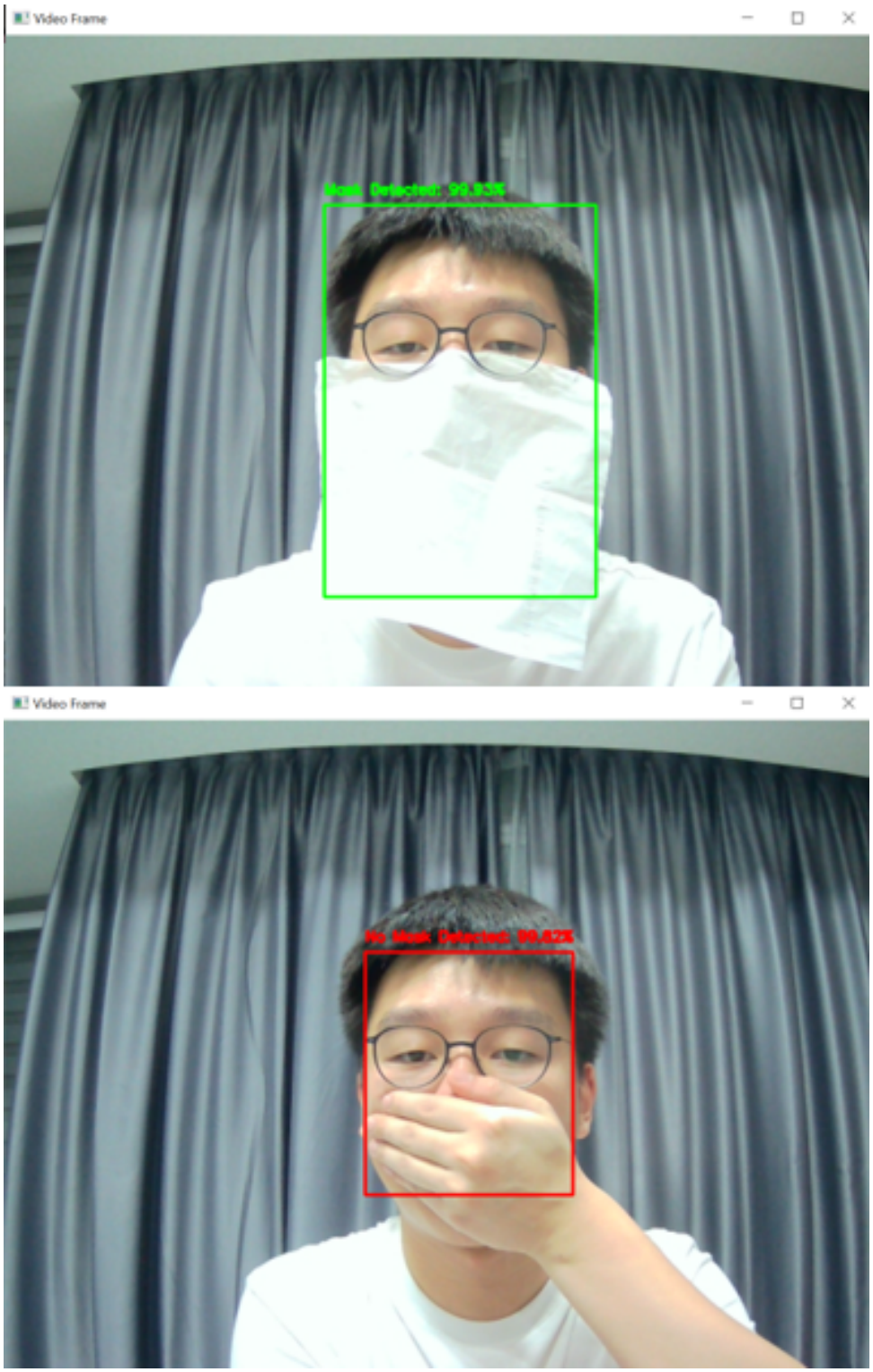


Figure A.4: Further Experiments with white napkin and hand

GoogleNet

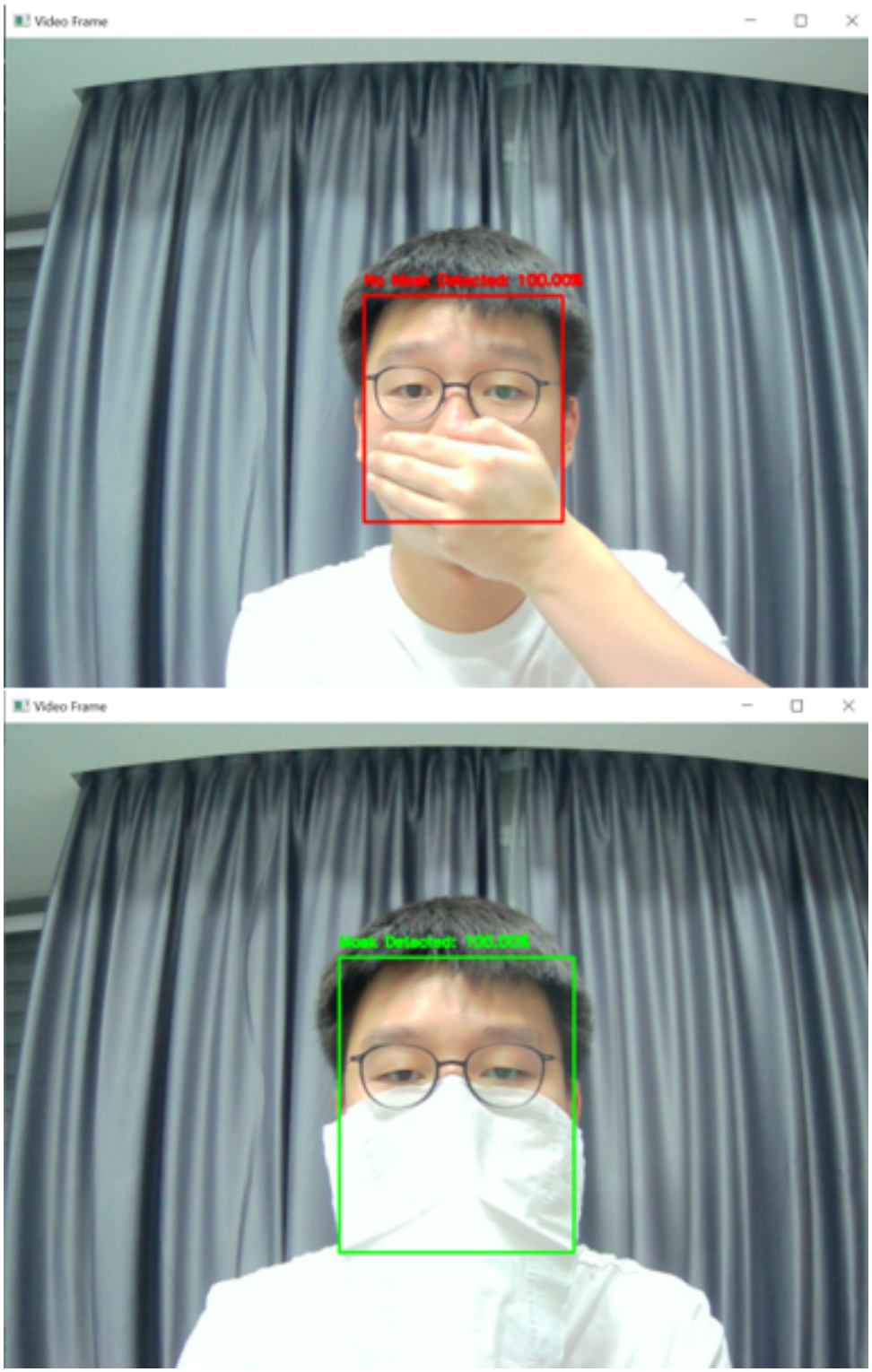


Figure A.5: Further Experiments with white napkin and hand

NasNet

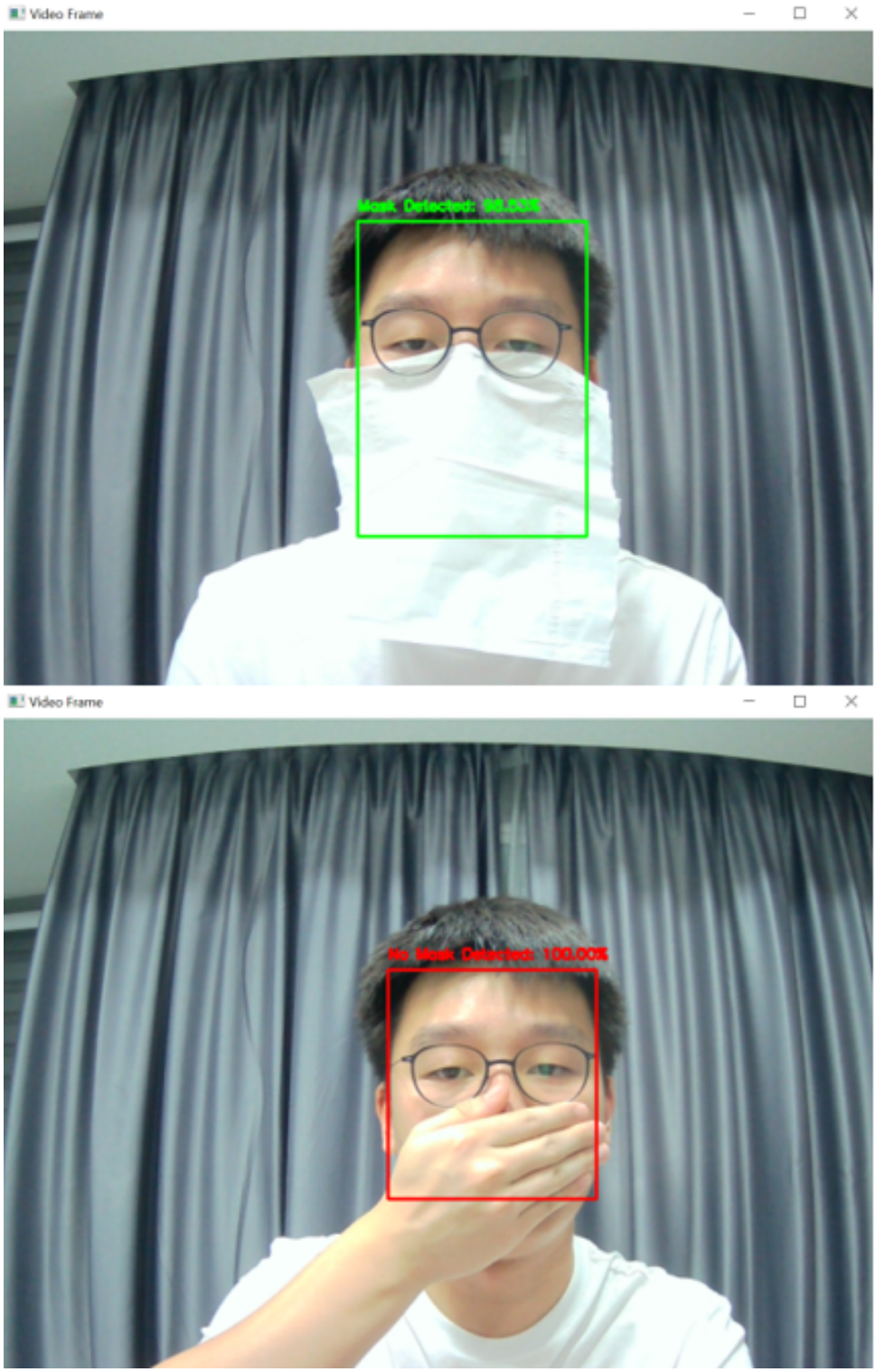


Figure A.6: Further Experiments with white napkin and hand