

**REPUBLIC OF TURKEY
HASAN KALYONCU UNIVERSITY
FACULTY OF ENGINEERING
COMPUTER ENGINEERING DEPARTMENT**



**A MULTIPURPOSE MASK DETECTION SYSTEM
USING DEEP LEARNING
GRADUATION PROJECT REPORT**

**Asmaallah FALLAHA
Dala KRAYEM
Mohammad Taj Eddin ASHOUR**

**Supervisor
Dr. Öğr.Üyesi Saed ALQARALEH**

**GAZİANTEP
2022**

APRIL 2022

Graduation Project in Computer Engineering Department

MAD

**HASAN KALYONCU UNIVERSITY
FACULTY OF ENGINEERING
COMPUTER OF ENGINEERING DEPARTMENT**



**A MULTIPURPOSE MASK DETECTION SYSTEM
USING DEEP LEARNING**

**GRADUATION PROJECT
IN
COMPUTER ENGINEERING**

BY

Asmaallah FALLAHA

Dala KRAYEM

Mohammad Taj Eddin ASHOUR

APRIL 2022

A Multipurpose Mask Detection System

Using Deep Learning

Graduation Project

in

Computer Engineering

Hasan Kalyoncu University

Supervisor

Asst. Prof. Dr. Saed ALQARALEH

By

Asmaallah FALLAHA

Dala KRAYEM

Mohammad Taj Eddin ASHOUR

April 2022

Copyright © 2022 A Multipurpose Mask Detection System Using Deep Learning All rights reserved. No part of this publication may be reproduced, distributed, or transmitted in any form or by any means, including photocopying, recording, or other electronic or mechanical methods, without the prior written permission of the publisher, except in the case of brief quotations embodied in critical reviews and certain other noncommercial uses permitted by copyright law.

REPUBLIC OF TURKEY
HASAN KALYONCU UNIVERSITY
FACULTY OF ENGINEERING
COMPUTER ENGINEERING DEPARTMENT

Project Name: A Multipurpose Mask Detection System Using Deep Learning

Name of the student(s): Asmaallah Fallaha, Dala Krayem, Mohammad Taj Eddin Ashour

Exam date:

We certify that this project satisfies all the requirements as a project for the graduation project

Prof. Dr. Muhammet Fatih Hasoğlu

Head of the Computer Engineering Department

This is to certify that we have read this project and that in our consensus/majority opinion it is fully adequate, in scope and quality, as a project for the graduation project.

Assoc. Dr. Öğr.Üyesi Saed Alqaraleh
Supervisor

Examining Committee Members(**Title and Name-surname**)

Signature

Whereby declare that all information contained in this document was gathered and provided by academic principles and ethical behavior. We further certify that we have properly cited and referenced any non-original material and results, as required by these rules of conduct.

Asmaallah Fallaha

Dala Krayem

Mohammad Taj Eddin Ashour

ABSTRACT

A MULTI-PURPOSE FACE MASK DETECTION SYSTEM USING DEEP LEARNING

Fallaha, Asmaallah

Krayem, Dala

Ashour, Mohammad Taj Eddin

Graduation Project in Computer Engineering

Supervisor: Asst. Prof. Saed ALQARALEH

April 2022, 98 pages

In previous years, both before and after the COVID-19 epidemic, employees in a variety of industries, including the medical, chemical, and nuclear fields, have been obliged to wear face masks while performing their jobs. It is practically hard to physically supervise around the clock in public places. Building an automated mask detection system can easily tackle this problem due to Convolutional Neural Networks (CNN) and deep learning's remarkable performance in all domains.

This paper applies the work of five deep learning models, in particular Convolutional neural networks, those being, MobileNetV2, VGG19, and three sequential models. Additionally, four datasets, each with roughly 6K, 12K, 4K, and 4K images, were utilized and in some parts combined to make one dataset (All-In-One dataset), hence guaranteeing that the comparison's results were reliable. Considering this, 5 different experiments were made to monitor the differences and the performance of the mentioned models. To start with, the 5 models alongside the 5 datasets were tested to assess their performance. The experimental findings demonstrated that all models performed well, however, VGG19 and MobileNetV2 proved to be two of the most efficient systems. Next, tuning techniques or also known as hyperparameter optimizations such as the number of filters, the size of the filters, the Dense rate, and the Dropout value which is important to note that they significantly impact the CNN model's overall performance, used on each model with the 4 datasets excluding the All-In-One dataset, this resulted in big improvements on all the models but once again mostly affected VGG19 and MobileNetV2. Again, using VGG19 and MobileNetV2 with only the All-In-One dataset and the tuning techniques, a noticeable difference was seen across the evaluation matrices such as Accuracy, Precision, Recall, and F1 score. Lastly, choosing VGG19 and MobileNetV2, with late fusion techniques (version one & version two) due to their simplicity to construct a multi-classifier system, were used to finalize the experiments. Late fusion version one was utilized to estimate the average output for both MobileNetV2 and VGG19 models. By taking the average output of both the models the performance intensified and strengthened greatly. As for late fusion version two we took the last layer to be concatenated and added a dense layer to extract its output. When compared, version one and version two of late fusion obtained very similar results theoretically, however in real-time version two outperformed version one of late fusion.

ÖZET

A MULTI-PURPOSE FACE MASK DETECTION SYSTEM USING DEEP LEARNING

Fallaha, Asmaallah

Krayem, Dala

Ashour, Mohammad Taj Eddin

Bitirme Projesi Bilgisayar Mühendisliği Bölümü

Tez Yöneticisi: Asst. Prof. Saed ALQARALEH

Nisan 2022, 98 sayfa

Önceki yıllarda, COVID-19 salgını öncesinde ve sonrasında tıp, kimya ve nükleer alanlar da dahil olmak üzere çeşitli sektörlerdeki çalışanlar işlerini yaparken yüz maskesi takmak zorunda kaldılar. Halka açık yerlerde günün her saati fiziksel olarak maske takılıp takılmadığını denetlemek pratik olarak zordur. Otomatik bir maske algılama sistemi oluşturularak, Konvolüsyonel Sinir Ağları (CNN) ve derin öğrenmenin tüm alanlardaki olağanüstü performansı sayesinde bu sorun kolayca çözebilir.

Bu makale, beş derin öğrenme modelinin, özellikle Konvolüsyonel Sinir Ağları (CNN), MobileNetV2, VGG19 ve üç ardışık modelin çalışmalarını uygular. Ek olarak, her biri kabaca 6K, 12K, 4K ve 4K görüntülere sahip dört veri seti kullanıldı ve bazı kısımlarda bir veri seti (Hepsi Bir Arada veri seti) oluşturmak için birleştirildi, böylece karşılaştırma sonuçlarının güvenilir olduğu garanti edildi. Bunu göz önünde bulundurarak söz konusu modellerin performanslarını ve farklılıklarını izlemek için 5 farklı deney yapılmıştır. Başlangıç olarak, performanslarını değerlendirmek için 5 veri setinin yanı sıra 5 model test edildi. Deneysel bulgular tüm modellerin performanslarının iyi olduğunu gösterdi, ancak VGG19 ve MobileNetV2 en verimli sistemlerden ikisi oldu.

Daha sonra, ayarlama teknikleri veya hiperparametre optimizasyonları olarak da bilinen filtre sayısı, filtrelerin boyutu, yoğun hız ve CNN modelinin genel performansını önemli ölçüde etkileyen bırakma değeri gibi Hepsi Bir Arada veri seti hariç 4 veri seti ile kullanıldı. Bu, tüm modellerde büyük iyileştirmelerle sonuçlandı, ancak bir kez daha en çok VGG19 ve MobileNetV2'yi etkiledi. VGG19 ve MobileNetV2'de sadece Hepsi Bir Arada veri seti ayar teknikleri ile kullanıldığında Accuracy, Precision, Recall ve F1 skoru gibi değerlendirme matrislerinde gözle görülür bir fark görülmüştür.

Son olarak, füzyon teknikleriyle (Erken ve geç füzyon) VGG19 ve MobileNetV2'yi seçmek, çok sınıflandırıcı bir sistem oluşturmadaki basitlikleri nedeniyle, hem MobileNetV2 hem de VGG19 modelleri için ortalama çıktıyı tahmin etmekte. Her iki modelin ortalama çıktısını almak performansı büyük ölçüde yoğunlaştı ve güçlendirdi.

ACKNOWLEDGEMENTS

Our gratitude to our supervisor Dr. Saed cannot be put into words, his great effort and time put to help finalize the graduation project can be noticed tremendously through the knowledge he provided throughout the report. We would like to thank Dr. Saed for all he has done.

PREFACE

The submitted study is the resulting work of Hasan KALYONCU University's Computer Engineering Department's Graduation Project. Face Mask Detection System is a project that aims to help control the wearing of the face mask in public places wherever it is necessary and prevent the spread of viruses and chemicals between people. Being the generation that got to witness a global pandemic, we chose to select this project because of how deeply we were affected by it and felt the need to make change. The brains behind the idea of the project were from the outstanding Asst. Prof. Saed AlQARALEH not only directed us throughout our project but also helped us understand the hardships of the real world and always tried to push us to perfection. Dr. Saed Alqaraleh is a true example of a professor that is deeply interested in the knowledge of their students and always made sure the information given is well-understood and clear. Dr. Saed has helped us and continues to do so in many ways and for that, we, are very thankful. We are very grateful for the sharing of their knowledge, time, and experience we got to share with him. We would also like to thank Prof. Dr. Muhammet Fatih HASOGLU head of department and Res. Asst. Furkan ÖZKAN for their support.

Asmaallah FALLAHA
Dala KRAYEM
Mohammad Taj Eddin ASHOUR

TABLE OF CONTENTS

| | |
|--|-------------|
| ABSTRACT..... | vi |
| ÖZET | vii |
| ACKNOWLEDGEMENTS | x |
| PREFACE | ix |
| TABLE OF CONTENTS..... | x |
| TABLE OF FIGURES | xiii |
| LIST OF TABLES | xiv |
| CHAPTER 1 | 1 |
| <i>Introduction</i> | <i>1</i> |
| 1.1 Project Purpose..... | 1 |
| 1.2 Problem Statement | 1 |
| 1.3 Solution Statement | 2 |
| 1.4 Glossary..... | 2 |
| CHAPTER 2..... | 3 |
| <i>Literature Review</i> | <i>3</i> |
| 2.1 What is the reason the system was made for? | 3 |
| 2.2 Related Work | 3 |
| CHAPTER 3..... | 7 |
| <i>Specific Requirements.....</i> | <i>7</i> |
| 3.1 User Requirements | 7 |
| 3.2 System Requirements | 7 |
| 3.3 Non-Functional Requirement..... | 10 |
| 3.4 Stakeholder..... | 13 |
| CHAPTER 4..... | 15 |
| <i>Feasibility Study</i> | <i>15</i> |
| 4.1 Project Description..... | 15 |
| 4.2 Justification for the proposed system | 15 |

| | | |
|------------------------|--|-----------|
| 4.3 | Considerations..... | 15 |
| 4.4 | Existing System and Technology | 16 |
| 4.5 | Power supply:..... | 20 |
| 4.6 | Solutions..... | 21 |
| 4.7 | Risk & Cost for Mask Detection | 22 |
| 4.8 | Comparison of solutions | 26 |
| 4.9 | Conclusion..... | 29 |
| CHAPTER 5 | | 30 |
| | <i>System Modeling</i> | 30 |
| 5.1 | Context diagram | 30 |
| 5.2 | Use case modeling..... | 31 |
| 5.3 | Class Diagrams..... | 33 |
| 5.4 | Class diagram | 35 |
| 5.5 | Sequence diagram | 36 |
| 5.6 | State diagram..... | 38 |
| 5.7 | Activity diagram..... | 40 |
| CHAPTER 6 | | 41 |
| | <i>Project Scheduling</i> | 41 |
| 6.1 | Task Network | 41 |
| 6.2 | Gant Chart | 43 |
| 6.3 | Pert Chart | 43 |
| CHAPTER 7 | | 46 |
| | <i>Mock-up and Interfaces</i> | 46 |
| 7.1 | Mock-up | 46 |
| 7.2 | Interfaces | 48 |
| CHAPTER 8 | | 52 |
| | <i>Image Classification System</i> | 52 |
| 8.1 | Labeling of image classification: | 52 |
| 8.2 | Types of image classification:..... | 53 |
| 8.3 | Image Classification Steps: | 55 |
| 8.4 | CNN Improvement Strategy: | 68 |

| | | |
|-------------------|---------------------------------------|-----------|
| 8.5 | Developed Mask Detection System | 76 |
| CHAPTER 9 | | 79 |
| | <i>Implementation</i> | 79 |
| 9.1 | Libraries: | 79 |
| 9.2 | Evaluation Matrix..... | 80 |
| 9.3 | Experiments and Results | 82 |
| CHAPTER 10 | | 94 |
| | <i>Conclusion</i> | 94 |
| | <i>Glossary</i> | 96 |
| | <i>References</i> | 97 |

TABLE OF FIGURES

| | |
|---|-----------|
| <i>Figure 5.1: Context Diagram for the System</i> | <i>30</i> |
| <i>Figure 5.2: Use Case Diagram, Face Mask Detection System</i> | <i>31</i> |
| <i>Figure 5.3: Use Case Diagram, Video Preprocessing.....</i> | <i>32</i> |
| <i>Figure 5.4: Controlling Gate, System and Person in charge.....</i> | <i>32</i> |
| <i>Figure 5.5: Controlling Gate, System and Gate Controller.....</i> | <i>33</i> |
| <i>Figure 5.6: Classes and their relations in the proposed system.....</i> | <i>34</i> |
| <i>Figure 5.7: Class Diagram of the developed system.....</i> | <i>35</i> |
| <i>Figure 5.8: The Sequence Diagram for the developed system</i> | <i>36</i> |
| <i>Figure 5.9: The systems Video Preprocessing Sequence Diagram.....</i> | <i>37</i> |
| <i>Figure 5.10: The systems State Diagram</i> | <i>38</i> |
| <i>Figure 5.11: Activity Diagram for the proposed system</i> | <i>40</i> |
| <i>Figure 6.1: Task Network of developed system.....</i> | <i>42</i> |
| <i>Figure 6.2: Gant Chart of developed system.....</i> | <i>43</i> |
| <i>Figure 6.3: Systems Pert Chart of developed system</i> | <i>45</i> |
| <i>Figure 7.1: Mock-up 1 of the system.....</i> | <i>46</i> |
| <i>Figure 7.2: Mock-up 2 of the system.....</i> | <i>47</i> |
| <i>Figure 7.3: Systems Admin Interface</i> | <i>48</i> |
| <i>Figure 7.4: Systems User Interface.....</i> | <i>49</i> |
| <i>Figure 7.5: Systems User Interface, showing Red Label</i> | <i>49</i> |
| <i>Figure 7.6: Systems User Interface, showing Orange Label.....</i> | <i>50</i> |
| <i>Figure 7.7: Systems Admin Interface, showing User Management.....</i> | <i>50</i> |
| <i>Figure 7.8: Systems Admin Interface, showing Camera Management.....</i> | <i>51</i> |
| <i>Figure 8.1: Mask Detection System (Multi Classifier).....</i> | <i>55</i> |
| <i>Figure 8.2: Early Fusion Method</i> | <i>71</i> |
| <i>Figure 8.3: Late Fusion Method</i> | <i>73</i> |
| <i>Figure 8.4: Intermediate Fusion Method</i> | <i>75</i> |
| <i>Figure 8.5: Late Fusion Method</i> | <i>77</i> |
| <i>Figure 8.6: late fusion after modifying on layers.....</i> | <i>78</i> |
| <i>Figure 9.1: Experiment 3, Sample A, Sample B, Sample C.....</i> | <i>90</i> |
| <i>Figure 9.2: Experiment 4, Stage 2 (Sample A, Sample B, Sample C)</i> | <i>91</i> |
| <i>Figure 9.3: Experiment 4, Stage 2 (Sample A, Sample B, Sample C)</i> | <i>92</i> |
| <i>Figure 9.4: Experiment 4, Stage 3 (Sample A, Sample B, Sample C)</i> | <i>93</i> |

LIST OF TABLES

| | |
|--|-----------|
| <i>Table 4.1: Network Video Recoder</i> | <i>17</i> |
| <i>Table 4.2: Camera</i> | <i>17</i> |
| <i>Table 4.3: Computers Comparison</i> | <i>18</i> |
| <i>Table 4.4: Database Comparison</i> | <i>20</i> |
| <i>Table 4.5: UPS Comparison</i> | <i>20</i> |
| <i>Table 4.6: Screens Specifications.....</i> | <i>21</i> |
| <i>Table 4.7: Personal, Development Cost.....</i> | <i>23</i> |
| <i>Table 4.8: Expenses, Development Cost</i> | <i>23</i> |
| <i>Table 4.9: Development Costs</i> | <i>23</i> |
| <i>Table 4.10: Personal, Operational Cost</i> | <i>24</i> |
| <i>Table 4.11: Expenses, Operational Cost.....</i> | <i>24</i> |
| <i>Table 4.12: Cash Flow.....</i> | <i>25</i> |
| <i>Table 4.13: Comparison of Solutions.....</i> | <i>26</i> |
| <i>Table 5.1: Controlling Gate, System and Person in charge.....</i> | <i>32</i> |
| <i>Table 5.2: Controlling Gate2, System and Gate Controller.....</i> | <i>33</i> |
| <i>Table 5.3: State Diagram Descriptions.....</i> | <i>39</i> |
| <i>Table 6.1: Project Tasks of developed system.....</i> | <i>41</i> |
| <i>Table 6.2: Pert Char of developed system t</i> | <i>43</i> |
| <i>Table 9.1: The first experiments results</i> | <i>83</i> |
| <i>Table 9.2: Experiment 2 Results</i> | <i>85</i> |
| <i>Table 9.3: RandomizedSearchCV Parameters</i> | <i>88</i> |
| <i>Table 9.4: Experiment without parameters tuning.....</i> | <i>88</i> |
| <i>Table 9.5: Experiment 3 with parameters tuning.....</i> | <i>88</i> |
| <i>Table 9.6: Best parameters for each model.....</i> | <i>89</i> |
| <i>Table 9.7: Improvement percentage in Experiment 3</i> | <i>89</i> |
| <i>Table 9.8: Experiment 4 - Stage 1 late fusion Results.....</i> | <i>91</i> |
| <i>Table 9.9: Results of the concatenated classifier</i> | <i>92</i> |
| <i>Table 9.10: Concatenated Classifier - Stage 2 Results</i> | <i>93</i> |

CHAPTER 1

Introduction

The project aimed to develop a system that enables accurately detecting of a face mask on the face in public areas (such as airports, factories, markets, public transport, hospitals, etc.) the system contributes to the public health care as it is useful in any workplace.

1.1 Project Purpose

Goals of the project:

- Provide a functioning model that allows for effective and accurate mask detection.
- Detects the existence of the face mask using image processing techniques.
- Develop an effective computer vision-based system for real-time automated monitoring of people to detect face masks in public.
- As a result of developing the face mask detector, it will enforce mask usage in public locations wherever it will be needed.

1.2 Problem Statement

A majority of nations mandate the covering of the nose and mouth in public areas as a fundamental precaution to keep people safe from virus outbreaks and chemical exposures. For instance, even years after the epidemic, the majority of airlines today insist that the face mask be worn inside the airplane. As a result, the need for establishing an alternate method to monitor persons who are not abiding by the requirements of wearing masks where it is required was driven by the near impossibility of physical 24/7 observation. This issue can be solved quickly, and the system may be utilized everywhere in the world with ease, by utilizing deep learning, machine learning, and a few image processing methods.

1.3 Solution Statement

The use of face masks enables effective prevention of many diseases and can put an end to workers inhaling chemical substances in factories. The system will ensure that all rules and regulations are being followed in public places where it is required by the law to wear a mask. This can be implemented by having an automatic technique which will deliver a very precise intelligent program that utilizes image processing in order to detect the mask on the face of the citizen.

An open-sourced face mask detection system that uses AI to detect if people are wearing masks or not will be an effective way to keep communities safe.

The face mask detector system was made to be functional even when faces are indistinguishable, blurry, and far from the camera.

1.4 Glossary

The glossary will be listed at the end of the report.

CHAPTER 2

Literature Review

In this chapter we provide an overview of knowledge and previously published studies on the topic, making it possible to shed an eye on all the relevant issues.

2.1 What is the reason the system was made for?

The benefit of the face mask detection system is that it can automatically identify those who are violating the ground rules of the public area. Many face mask systems and tools have been developed where it is used in factories and many health workplaces with the goal of using them to aid in halting the spread of disease and hazardous materials.

Face mask detectors are officially used in the 21st century around the world specifically after the pandemic that happened due to coronavirus. The development of the ideal systems and technologies will benefit societies around the globe.

2.2 Related Work

There are many existing systems that may have the same functionality “the ability to detect whether the person is wearing a mask or not” but they are different in the way the system was built.

Earlier datasets for such projects mostly consisted of images taken in supervised environments, whereas the most recent datasets are created by training the system with online images. Larger datasets are much more necessary for better and more efficient training.

Up until today, researchers mostly focused only on gray-scale face images which only captured images and with no real time detection. Some were built on pattern identification models, while others were built using AdaBoost.

In [1], this paper proposes a framework based on a new algorithm which focuses on performance in terms of getting high frame rates. The object detector framework is built

with three contributions, the first is an integral image that can be calculated just using a few processes per pixel from an image. The second contribution is the AdaBoost algorithm in which It chooses a small number of key visual features from a bigger set to produce highly efficient classifiers [2]. The third contribution is a "cascade" method for merging progressively complicated classifiers, allowing background parts of an image to be swiftly dismissed while more computation is spent on potential object-like regions. As a result, the model achieved 95% as an accuracy of face detection and face detection at 15 FPS (Frames per seconds) which is the rate at which a display's back-to-back images, known as frames, appear and produce moving imagery.

In [3], In this paper a new model was developed that focuses on improving the performance. The model has three stages, all stages are built on CNN networks. Stage one is Proposal Network (P-Net) which obtains the candidate windows and their bounding box regression vectors. Stage two is Refine Network (R-Net), R-net accomplishes calibration using bounding box regression and NMS candidate merge, as well as rejecting a large number of incorrect candidates. Stage three (O-Net) that describes the face in more detail. Specifically, the network will output the positions of five facial landmarks. Three datasets were also used in this paper, FDDB dataset contains annotations for 5171 faces in a set of 2845 images. WIDER FACE dataset contains 393703 labeled faces in 32203 images, and AFLW contains 24386 face's annotations. The work reached an accuracy of 97.14% for the built model and in terms of performance it takes 16 FPS on 2.6GHZ CPU and 99 FPS on NVidia Titan black GPU.

In [4], in this paper machine learning was used to create a hyper deep and classical model using TensorFlow, Keras, OpenCV and Scikit-Learn libraries. Resnet50 model was used which is a convolutional neural network (CNN) that is 50 layers deep, and SVM which is a supervised learning model used for classification. All these components build a face detector alongside MobileNet architecture which is used as an image classifier. The model was trained on a dataset which contains 3918 images separated into two classes, faces with masks and faces without masks. As a result, the system achieved 98% of accuracy by the trained architecture.

In [5], this paper used DNN which enables real-time detection without consuming a lot of resources and MobileNetV2 classifier to recognize face masks with the use of

TensorFlow, Keras, and OpenCV libraries. Meanwhile the SSDMNv2 model which is based on DNN is an approach that uses Single Shot Multibox Detector as a face detector and MobilenetV2. The datasets were constructed from a combination of Kaggle and “Prajna Bhandary” datasets. The Kaggle dataset contains a total of 678 images of people wearing medical masks and an XML file containing descriptions of these images. The “Prajna Bhandary” dataset consists of 1376 pictures belonging to two classes, 690 images of people wearing masks and 686 images of people without masks. As a result of using the SSDMNv2 model with MobilenetV2 classifier a system was developed with 92.64% accuracy rate and 15.71 FPS (Frames per seconds) performance.

In [6], this paper provides a model built on CNN and MobileNetV2 classifier. The paper used two datasets: Kaggle's dataset and RMFD dataset which were combined into one dataset that contains 1916 image with mask and 1930 without mask. Then the data was split to 75% for training the model and 25% for testing it. As a result, the papers model achieved 96.85% in detecting people's face masks.

As we are aware, the goal of hyperparameter optimization is to identify a tuple of hyperparameters that will produce an optimal model which will minimize a predetermined loss function on provided independent data. Approaches for hyperparameter optimization (HPO), like grid search and random search, are used to adjust hyperparameters. The adaptable Tree Pazen Estimator (ATPE), a new automatic machine learning-based HPO that can estimate workpiece quality in high dimensions, is proposed in this paper [7]. The ATPE offers an adaptive warm-up procedure for TPE and has the ability to automatically tune high dimension hyperparameters. Workpiece quality datasets based on MNIST and XGBoost are used to test the proposed technique. The outcomes demonstrated that it outperforms RS, annealing, and TPE and accelerates convergence of eight hyperparameters without human intervention. The findings demonstrate that ATPE not only simplifies the process of tuning hyperparameters but also achieves cutting-edge performance, demonstrating its promise for the prediction of workpiece quality.

In this paper [8], we take into account using Gaussian processes to construct the relationship between the machine learning models' performance and their hyperparameters. By doing so, the hyperparameter tuning issue can be reduced to an optimization issue, which Bayesian optimization is then employed to resolve. Bayesian

optimization is built on the Bayesian theorem. In order to update the posterior of the optimization function, it sets a prior over the optimization function and obtains data from the previous sample. Several tests were performed using common test datasets such as MNIST dataset and CIFAR-10. The results of the experiments demonstrate that the suggested method may, when time costs are taken into account, identify the optimal hyperparameters for popular machine learning models like the neural networks and the random forest algorithm.

According to this paper [9], "late fusion of high-level features" from DCNNs and "ensemble of multiclass classifications" offer significantly better performance for classifying images than traditional hand-crafted features. The "ensemble of multiclass classifications" is proven to be significantly more efficient than the "late fusion of high-level features." Here, it is shown how using several DCNN results in a more generic feature map. However, combining feature maps works best when done after constructing a Logsoftmax prediction matrix. The best combining method for the CIFAR-10 dataset is "weighted sum ensemble," but the best method for the CIFAR-100 dataset is "simple sum ensemble." Additionally, the experimental results demonstrate that DCNN performance improves with depth.

In the previous papers, many methods for detecting the face mask using image classification and computer vision techniques were developed. While some of these articles focused mainly on developing systems without the use of tuning techniques, other papers did utilize these techniques but their primary goal was only image classification. The main difference of our approach in this study is that the focus was both on image classification and the use of tuning techniques (hyperparameter optimization). We have built an efficient mask detection system based on the extensive experiments of tuning parameters that investigated the performance of multiple classifiers. The system followed a technique of combining the two best models together to then achieve a 99.7% test accuracy with the aid of the hyperparameter optimization. This percentage is regarded as a significant accomplishment when compared to the above studies.

CHAPTER 3

Specific Requirements

It is a process of defining the service that customers require from the system and its constraints. There are two types of requirements; user and system requirements.

3.1 User Requirements

These requirements are written for the customers, it's written in a simple and natural language that does not contain details. The purpose of this requirement is to allow customers to ensure that the system will perform as expected, it's also a form of communication between the developers and the customers.

1. Capture the video stream
2. Video preprocessing
3. Object detection
4. Face detection
5. Mask detection
6. Open the gate
7. Showing results

3.2 System Requirements

System requirements are written for developers, they are clearer and more precisely detailed than user requirements. The purpose of these requirements is to inform developers on what to build in great detail. In more detail, these requirements are written below.

3.2.1 Capture the video stream

Receive the video stream from the CCTV system. The system should check for the CCTV video stream, if the input gives a black screen, an alert should be sent to the person in charge to check the connection between the system and the CCTV devices. In

case the stream was received correctly the system should start the video preprocessing stage.

3.2.2 Video preprocessing

In this stage, the system will start analyzing the received video.

3.2.1.1 Capturing the frame from the stream

In this phase we may face a problem with the buffer's capacity being full, this could prevent the frame from being placed and used. To prevent such problems from happening, a buffer of good performance that can work with the quality of the frame captured should be taken into consideration.

3.2.1.2 Resize the frame size to fit the chosen model

We should be sure about the model frame sizes, after the system should resize the frame to fit the sizes of the used model, if the frame size was incompatible with the model the system will stop with an error. We should be sure about the model frame sizes to avoid such errors.

3.2.1.3 Convert the image into an array

When a picture is converted to a numPy array, an array with a series of values that each represent a single pixel in the image will show, this is the procedure by which the computer can understand pictures. In case if we have a huge number of frames, we should ensure that the system memory can hold such a large number of frames to avoid memory errors.

3.2.1.4 Convert the image into grayscale

The RGB image consists of three dimensions, whereas the grayscale image is only of two dimensions. Some algorithms can only be applied to two-dimension images. Cameras that support night vision use grayscale with image, this increases the performance of the system in lower lighting environments [10]. If the image was originally grayscale coming from the camera an error will occur. Before the image is converted the system should check for the color-type of the image.

3.2.1.5 Object detection

The system may face difficulties in detecting multiple objects because of the lack of accuracy in the captured images, this may occur due to foggy and dusty weather, or a new object appearing in which the model is not trained on it, in this case all elements that could appear within the frame must be included in the study.

- Searching for salient features of the human face such as eyes, nose, eyebrows etc.

Sometimes the features of the face can be unrecognizable due to many reasons such as sunglasses or medical glasses, or the angle of the face. One of the solutions can be by placing stickers at each gate explaining the right way the face should be prepared.

3.2.3 Face Detection

The system will start the face detecting phase, and show up the region of the face

1. Specifying the location of the face.

After detecting the face features, the system will detect and decide the whole face. If the system passes the previous step correctly this step will also pass. If an error occurs, the system should return to “Object Detection” step.

2. The output of the face detection will surround the face with a rectangular box.

The rectangle will appear when the steps are done and successful, if not the system should go back to “Object Detection” step.

3.2.4 Mask Detection

- Use the output of the face detection model as the input of the mask detection model. The error rate is quite low in this phase indicating that the system has finished the previous steps verification operations successfully.
- After selecting the dataset, we should train the model using that dataset. A training model is a dataset that is used to train an algorithm. It consists of sample output data as well as related sets of data input that have an impact on the output. The input data is passed into the algorithm through the training model, and the processed output is compared to the sample output. The model is modified based on the results of this association.
- After picking the face in the previous steps, the model decides whether there is a face mask. Certain individuals may try to trick the system by covering their faces with their hands or anything that covers the face that isn't considered a mask, this

will not be accepted by the system and the individual will not be able to pass by. On the other hand, due to religious reasons some individuals may wear a face covering also known as a niqab or a veil, this is classified as a mask.

Others could be using winter scarves around their neck and covering their face, this can also be classified as a mask. Even in such situations the system may not be able to always detect as accurately as possible, if so, this will give an orange alert that will be sent to the person in charge.

Therefore, it is necessary to classify what is acceptable as a mask and what isn't, this can be achieved by adding images to the dataset that contains all the cases mentioned such as the niqab, or covering the face with one's hand, etc.

3.2.5 Open the Gate

- After detecting if a person is wearing a mask or not, procedures are going to be taken to open the main gate. If the person is not wearing a mask the gate will not open. In case the system couldn't access the gate, an alert should be shown to the person in charge to check the connection between the system and the gate.
- Other technical problems could be encountered with the gate, one of the solutions can be to direct people to enter from another gate or to inform the person in charge to open the gate manually.

3.2.6 Showing results

1. To monitor if the person is wearing a mask or not, a green box with the label "MASK" will show on the screen around the face. If the system couldn't connect to the screen an alert will be sent to the person in charge.
2. If the individual is not wearing a mask, a red box with the label "NO MASK" will be shown.
3. In case the system could not detect if a mask is worn, an orange box with the label "NOT DEFINED" will be shown.
4. In case no result is present on the screen, a bell will be used to give a sound in case there was a face mask.

3.3 Non-Functional Requirement

This is described as the system's quality and functionality such as data processing, consumption, all possible scenarios where the system might be involved, used techniques and language, and other tools that are used to implement and maintain the system.

3.3.1 Product Requirement

3.3.1.1 Usability

The main feature of our system is that it is easy-to-use considering little to no complications in the use of the interface, the user does not need to have any previous knowledge on how the system is used.

It's a simple screen that shows the results as the following:

- In a situation where a security guard is available, the screen will be shown to them only and is not viewed by the people entering. The simplicity of the system here can be noticed due to how much it eases the job of the guard because they are only needed when the system cannot decipher the face of the individual.
- The labels are either green or red depending on the persons' mask, if it is worn correctly a green box will border the face with the label "MASK" and the gate will open, whereas if the person is not covering the mouth and nose or is not wearing a mask completely, a red box will border the face with the label "NO MASK" and will not be granted entry through the gates.
- If the system could not apprehend and distinguish whether a mask is worn or not, may it be because of the face not being clear or centered, a third box will be shown with orange and the label "indistinct", the person in charge needs to then inspect the persons mask and grant entry manually through the gates.
- On the other hand, when an individual is passing through the gate, the system will detect if the person is wearing a mask, in which the gates will open and let the individual through. If not, they will not be authorized entry through the gates.

Additionally, the person in charge will be provided with a user manual that includes images with texts to be guided on how to use the system. Introductory stickers will also be placed on the gates for easier entry.

3.3.1.2 Efficiency and reliability

To ensure the quality of the face detection system and provide the best achievable results, highest accuracy, and fastest time response, this can be obtained by training multiple models, either the model of face detection or the model of mask detection or both, and then choosing the model with the most accurate, fastest, and less costly of resources.

Using night vision security cameras can help with the efficiency of the model which makes it easier to recognize the image in low-light or no-light situations; this will improve the processing of night images by the system.

3.3.1.3 Dependability

To always ensure the availability of the system, UPS power supply is used. In the case of a power outage, the system automatically switches to the ups power unit.

The system was also built with two hosting servers, one being the main server and the other as a backup server in case the main server goes down.

In addition, the system is able to do self-checks unassisted, for instance the system is provided with multiple cameras and if a problem with one of the cameras occurs, the system automatically switches to the other stand-by camera.

3.3.1.4 Security

The network is built on two levels, the first level is provided with internet access in case the IT manager wants to manage the system or access the data on the server. In which case the system's data is encrypted to prevent any type of external attacks and threats. The Second level is the local network, where the server is connected to all parts and devices in the system, if so, there is no need for internet access and the devices will work locally.

3.3.2 Organizational requirements

3.3.2.1 Development requirement

- Python programming language
- Python Libraries:
 - TensorFlow, open-source machine learning platform.
 - OpenCV, computer vision and machine learning library.
 - Keras, API based on TensorFlow.
- The host server is Linux, because after comparing the Linux server to the Windows server, Linux surpassed for being completely free and much safer due to no malwares [11].
- Firebase Real-time Database, is a NoSQL database which is a non-tabular database. Firebase is a real-time database where it stores and syncs data in real-time between the users, in which they can participate in the collaboration of the data from different devices which means users don't necessarily need to use one device to be able to use Firebase [12].
- CCTV system with high resolution and night vision cameras, where the system reads the images through it.
- Small network to connect the system and devices together.

3.3.2.2 External Requirements

- Ethical Requirements

The system was developed to keep communities safe. This is done through the automated real-time mask detection system; the system improves the quality of life by providing peace of mind in work places with an obligation to wear the face mask.

- Regulatory requirements

The dataset that was used in the system is taken from a public domain with no copyright restrictions, this ensures no laws are broken.

3.4 Stakeholder

To understand who the stakeholders are; Stakeholders are individual groups or organizations that may have an impact on the project or be impacted by the project.

List of the stakeholders:

- Citizens
- Developers of the system
- Project managers
- Customers
- Security team
- Analysis team

CHAPTER 4

Feasibility Study

A feasibility analysis, as the name suggests, is used to establish the viability of a concept, such as ensuring that a project is legally, technically, and economically feasible. It tells us whether a project is worthwhile investing our time and money in—in some situations, a project may be impossible to complete. There are a variety of reasons for this, including the need for too many resources, which not only prevents other parts of the project to be performed correctly but also may cost more than the organization would gain back by taking on a project that isn't profitable.

4.1 Project Description

Using computer vision, machine learning, and deep learning, we require real-time mask detection to handle surveillance concerns in high-population areas where it is mandatory to wear face masks in public and in private sectors.

4.2 Justification for the proposed system

The project's goal is to meet the requirements for developing a system that allows to properly detect the mask on a person's face in public places, thereby contributing to public health and security.

4.3 Considerations

Because the system's goal is to ensure the wearing of masks, it must be reliable. Additionally, in the event of an unplanned failure, the system must be able to recover and be ready to be utilized again.

The following are the most important criteria on which we are basing our project:

Maintainability: The system was developed to be easily modified to the changes that may be required later on and have the likelihood of restoring or repairing a failed component or system within a given time.

Dependability: Reflects the degree in which the user believes that the system will perform as expected and will not presumably fail, in other words, that the system will be available during expected hours.

Reliable: In order to provide client satisfaction, the system should operate exactly as the end-user anticipates it to which is to identify the mask. To ensure the reliability of the proposed system, the system was trained with several models and according to that, the one with the highest accuracy was chosen.

Response time: The response time of this system is rapid. The system is able to detect the face mask as soon as it marks a person.

Technology: All software & hardware that were utilized are available and can easily be obtained. This makes developing the system possible with currently available technologies.

Finance: The most efficient yet simultaneously affordable hardware and software was selected to minimize the costs but still delivering a highly efficient system.

Resources: To examine the resources required to complete the project and whether the number of available resources, such as screens and cameras, is sufficient to complete the project effectively.

Legal: the proposed system obeys the law in which all technologies can be obtained by law. The usage of personal information is also secured while capturing the frame to make sure that the captured frame is not used in illegal manners.

Time: To consider the period needed to complete the project, and also to consider the constraints on a schedule like weekends, Eid holidays, National holidays etc.

8 months were given to complete the proposed system. In general, the system was divided into the Analysis phase which took approximately 4 months and the implementation and testing phase which also took 4 months. If the delivery deadline is exceeded a contract with the customer will guarantee their rights. In case the delay is caused by the customer we are not responsible for this delay.

4.4 Existing System and Technology

Describe existing systems that accomplish or partially accomplish the proposed system's goals.

4.4.1 CCTV system

First, we need a network video recorder, which should be compatible with the camera.

Table 0.1: Network Video Recorder

| Network Video Recorder | | |
|------------------------|-------|-----------|
| Name | Dahua | Hikvision |
| Max. channels | 8 | 8 |
| Price | 1,100 | 1,050 |

In the case of which the organization using the project chooses to provide the entity of the cameras, the system will be integrated onto the cameras without the additional ones provided by the systems' part.

Table 0.2: Camera

| Camera | | |
|--------------|-----------------|-----------------------------|
| Name | Dahua 4 MP DHOP | Hikvision 4 MP Fixed Bullet |
| Price | 1,900 | 3,000 |
| Resolution | 4 MP | 4 MP |
| FPS | 25/30 | 30 |
| LED Distance | 30m | 60m |

Because within the system the distance of the LED is disregarded, and all the requirements and specifications are mostly the same, we chose to use DAHUA and the network video recorder of DAHUA for the similar cheap price tags and compatibility of them together. Prices are retrieved from the general market.

4.4.2 Host Server

Table 0.3: Computers Comparison

| Name | Custom PC | Asus Tinker | Raspberry Pi 4 Model-B |
|---------|----------------------------------|--|------------------------|
| CPU | Intel Core i7-11700F 2.50 GHz | Dual-core Arm® Cortex®-A72 @ 2.0 GHz | ARMv8 1.5GHz |
| GPU | NVIDIA GeForce RTX 2060 | ARM® Mali™-T764 GPU | Integrated |
| RAM | 16 GB | 2GB | 4GB |
| Memory | SSD | 16GB eMMC | Micro-SD card |
| Network | Gigabit Ethernet | Gigabit Ethernet | Gigabit Ethernet |
| Price | 19,990€ | 1,750€ | 1,000€ |

As we need a powerful GPU to process the frames with high FPS and less time, the first option was chosen, which has a RTX 2060 GPU.

4.4.3 Operating System

- Open source: Linux is a completely open-sourced project one can view the Linux code unlike windows which is a closed source OS [13].
- Security: Linux is much safer and less vulnerable to attacks, while windows require anti-virus programs to be protected against hacks and malware. The way Linux works and the process of package management and many other features makes it more secure than windows.
- Range of systems: Linux can be used in old or new computers regardless of the hardware requirements by using older versions (such as Puppy Linux). Windows needs a minimum hardware requirement to be run successfully.
- Supports programming languages: major programming languages like Python, C++, Java, etc. are supported by Linux. Many libraries are developed solely for Linux.
- Privacy: Linux does not collect data from its users. In contrast, windows by default collects information and data.
- Free: Linux is free and accessible to the public.

According to these statements above, Linux was preferred to be used and chosen.

4.4.4 Programming language

In the fields of computer vision, machine learning and deep learning there are many programming languages that support these fields like java, python, R and C++.

But python was selected for some of the following reasons:

- Library ecosystem, there are a lot of libraries that support machine learning, deep learning, and data processing in python.
- Seamless Integration, Python integrates easily with enterprise systems, making it possible to construct web services.
- Platform independence, Python can run on all platforms.
- Visualization options, Python has several libraries, some of which are excellent visualization tools. Data scientists can use libraries like Matplotlib to build histograms, graphs, and plots to improve interpretation, show, and visualize data.
- Easy coding, python uses plain English, this advantage is very helpful when faced with complicated scenarios.
- Community Support, Python has a significant user community around the world, and these groups are always helpful when code issues arise [14].

4.4.5 Database:

- **Firestore**, is a NoSQL database which is a non-tabular database. Firestore is a real-time database where it stores and syncs data.

Advantages:

1. Mostly free to use. It has multiple plans to choose from according to the needs of the user.
 2. Scalable horizontally
 3. Handles large amounts of data effectively
- **MySQL**, is an open-source free database that uses SQL which vertically scales databases in a table-based form which is made out of rows and columns. Users can manipulate and control data as needed.

Advantages:

1. Completely free.

2. Vertically scalable through tables
3. Used mostly for complex data.

Table 0.4: Database Comparison

| | |
|--------------------------|--|
| MySQL | Firebase |
| Free to the public | Mostly free but is paid for more features |
| Open-Source SQL Database | Cloud based or (platform by google) |
| Handles complex data | Handles large amounts of data |
| Vertical scalability | Horizontal scalability |
| Based on SQL | NoSQL database that stores data in real-time |

After analyzing the above table and comparing between the two database services, it was decided upon Firebase for it being a real-time database which is needed in a system like face mask detection.

4.5 Power supply:

Table 0.5: UPS Comparison

| | |
|-------------------|------------------|
| Powerful SLE-2000 | Makelsan Lion |
| 2000 VA | 2200 VA |
| 1200 W | 1320 W |
| Line Interactive | Line Interactive |
| 1,800₺ | 1,770₺ |

Using Makelsan Lion UPS, which comes with 2200 VA and 1320 W and line interactive features.

4.5.1 Gate Controller:

5V 2-Channel Relay Board Module

4.5.2 Human-computer interaction:

Table 0.6: Screens Specifications

| | HP V27i | Samsung M5 | Philips 273V7QDSB |
|------------|-----------|-------------|-------------------|
| Resolution | 1920x1080 | 1920 x 1080 | 1920x1080 |
| Size | 27 inches | 27 inches | 27 inches |
| Price | 3,700₺ | 2,940₺ | 2,690₺ |

The PHILIPS screen was chosen to be used after comparing the specifications of the screens and figuring all are the same except for the price, in which the PHILIPS screen came first in cheapest price tag.

4.6 Solutions

1. In [15], which we refer to as “Method1”, the project aims to detect the existence of a facemask in real-time. A green square around the person's face indicates the presence of a mask, along with the percentage of accuracy of the mask's location. The absence of the mask, on the other hand, is shown by a red square around the person's face, along with its percentage, which may be used to correctly differentiate whether a person is wearing the mask properly or not. Because of its high detection rate and processing speed, it can process a 384 by 288-pixel image in about 0.067 seconds. The Viola-Jones algorithm is a popular method for facial detection. The system requires a working webcam, ram of 4GB and/or more, 64-bit processor, python, and its library such Tensorflow and Keras, MobileNet as classifier and the input is given by CCTV camera. The model archives an accuracy with 97%.
2. In this paper [16], which we refer to as “Method2”, they used the YOLO V4 algorithm to develop a face mask detection for COVID-19 prevention. The face mask detection will be used in a real-time application to detect all types of commercial face masks. The average FPS generated by the face detection in all these models is around 11.1 FPS. A digital webcam camera, a PC, and a speaker make up the entire hardware system. The face detection computation is done entirely in a computer that is connected to the GPU to improve the image's

graphical calculation. When the system detects a user who is not wearing or wearing a face mask in front of the camera, it will instruct the speaker to remind him or her to do so, and this will continue until the user puts their face mask on properly. The webcam camera was mounted on the top of the monitor, allowing the user to be detected from about 5 meters. For the object detection method calculation, the model is equipped with a MiniPC with a GTX 1060 Nvidia GPU. In every situation, such as lighting, mess up area, clean area, and when the users are standing close to each other, the YOLO v4 algorithm can detect and distinguish a non-wearing and a wearing-mask user correctly.

3. In this paper [17], which will be referred to “Method 3”, the system will be used to monitor people’s face masks and social distance during the ongoing COVID-19 pandemic, if the person is not following the rules, alerts will be sent to the police for action to be taken. It implements the wearing of the mask and helps the guidelines of social distancing to be followed. The model was created to be run on raspberry pi4 model-B with the ARMv8 1.5 GHz processor and 4 GB of RAM as the preferred device. With the use of OpenCV and MobileNet V2, the solution will analyze Real-Time Streaming Protocol (RTSP) video streams with neural networking models. The accuracy of the proposed system was between 85% and 95%, the precision score of social distancing and masks was 91.7%. The generated frames from the model were around 28.07 FPS.

To extract useful image features, Single Shot Detector MultiBox (SSD) was used along with the VGG19 model pre-trained on ImageNet as its basic model for real-time object detection.

4.7 Risk & Cost for Mask Detection

Because all technologies are available and could be obtained, there is no shortage in expertise. The system is also able to be integrated with other systems; and since it's a mature system ("a technology that has been in use long enough that almost all of its initial flaws and inherent problems are removed or reduced"), our system is risk-free.

4.7.1 Development Cost:

The cost for building the model

Personal:

Table 0.7: Personal, Development Cost

| | | |
|---|--|---------|
| 1 | Analysts (832hour 50₺/hour) | 41,600₺ |
| 2 | Artificial Intelligence Developer (520hour 48₺/hour) | 24,960₺ |
| 3 | CCTV Camera Technician (18hour 150₺/hour) | 2,800₺ |
| 4 | Telecommunications Specialist (24hour 37₺/hour) | 888₺ |

Expenses:

Table 0.8: Expenses, Development Cost

| | | |
|---|---------------------------------------|--------|
| 1 | User tutorial posters (4₺/per poster) | 4,000₺ |
|---|---------------------------------------|--------|

New Hardware & Software:

Table 0.9: Development Costs

| | | |
|---|------------------------|---------|
| 1 | Network Video Recorder | 1,100₺ |
| 2 | Camera | 1,900₺ |
| 3 | Custom PC | 19,990₺ |
| 4 | Screen | 2,690₺ |
| 5 | Gate controller | 25₺ |
| 6 | Power Supplier | 1,770₺ |

Total Development Costs: 101,723₺

4.7.2 Operational Cost:

For maintaining the system

Personal:

Table 0.10: Personal, Operational Cost

| | | |
|---|--|--------|
| 1 | Artificial Intelligence Developer (100hour 48£/hour) | 4,800£ |
|---|--|--------|

Dedicated to maintaining the system on a regular basis as well as updating the systems that are currently in use.

Expenses:

Table 0.11: Expenses, Operational Cost

| | | |
|---|--|--------|
| 1 | CCTV system maintenance (10hour 150£/hour) | 1,500£ |
| 2 | Network maintenance (10hour 37£/hour) | 370£ |

Maintaining the network, checking the CCTV system, and ensuring that it is operating as required.

Total Projected Annual Costs: 6,670£

Table 0.12: Cash Flow

| Cash flow | Year0 | Year 1 | Year 2 | Year 3 | Year 4 | Year 5 | Year 6 |
|----------------------|----------|----------|-----------|-----------|-----------|-----------|-----------|
| Dev. Costs | -101,723 | | | | | | |
| Oper. Costs | | -6,670 | -10,200 | -14,500 | -16,000 | -18,000 | -21,612 |
| Present Value | 1 | 0.714 | 0.510 | 0.364 | 0.260 | 0.186 | 0.133 |
| time-adj Costs | -101,723 | -4,764 | -5,204 | -5,284 | -4,165 | -3,347 | -2,870 |
| Cumulative Costs | -101,723 | -106,487 | -111,691 | -116,976 | -121,141 | -124,487 | -127,358 |
| | | | | | | | |
| Benefits | 0 | 15,000 | 22,600 | 34,000 | 42,867 | 52,367 | 61,867 |
| T-adj Benefits | 0 | 10,714 | 11,530.61 | 12,390.67 | 11,158.55 | 9,737 | 8,217 |
| Cumulative Benefits | 0 | 10,714 | 22,244.90 | 34,635.57 | 45,794.11 | 55,530.88 | 63,747.41 |
| Net Costs + Benefits | -101,723 | -86,244 | -58,795 | -18,876 | 31,083 | 89,961 | 156,579 |

- Discount rate 40%

The discount rate is Loss rate of equipment used.

- Break-even Point

Which is the part of the year when the payback occurs.

Break-even point of the project: 4.38, the second quarter of fourth year

- Return on Investment

Which is the ratio of an investment's value to its cost

ROI = 122.94%

4.7.3 Intangible Benefit:

- Ensure the rules and regulations are being followed.
- Force people to wear masks.
- Keep communities safe.
- Contribute to public healthcare.
- Decrease load work.

4.7.4 Tangible Benefit:

- Avoid government penalties and sanctions for non-compliance with the prescribed norms.
- Save money by not hiring employees to monitor and control the use of face masks.

4.8 Comparison of solutions

Table 0.13: Comparison of Solutions

| Feasibility Criteria | Wt. | Method1 | Method2 | Method3 | Method4 |
|--|-----|---|---|--|--|
| Operational Feasibility: 1. Could there be a reduction in cost and or an increase in benefits? 2. Will the proposed system really benefit the organization and how well the system would work? 3. Can or will end-users and management adapt to the change? | 20 | Without the need for continuous human surveillance, security personnel would be able to easily detect those who are not wearing a mask. This system can be further extended | In every situation, such as lighting, mess up area, clean area, and when the users are standing close to each other, the YOLO v4 algorithm can detect and distinguish a non-wearing and a wearing-mask user correctly | The system will do society a favor by saving time through minimizing physical surveillance work and helping to lower the spread of the virus. It can be implemented effectively to inspect citizens in public settings. Thus, this proposed system will operate in an efficient and automated manner. Changes within the system can easily be improved depending on the needs of the public. | The proposed system will meet the organizational needs due to its accurate results and fast response time. The systems cost will be suitable according to the budget and after analyzing the costs it can be seen that the benefits will return very quickly due to the system's ability to be used in any location needed. Due to using the newest technology through the implementation process and the ease in the use of the system, any |

| | | | | | |
|---|----|---|---|--|--|
| | | | | | change that occurs will be easily adaptable by the user. |
| | | Score: 95 | Score: 80 | Score: 85 | Score: 95 |
| Technical Feasibility: 1. Is the project feasible within the limits of current technology? 2. Does the technology have the capacity to handle the solution? 3. Manpower programmers, testers & debuggers | 10 | With the technology (python and its libraries) we have now, it is possible to complete this project at current time | Because the model gave such high accuracy results the system will be able to handle the solution. | We can implement the system with the current programming tools and hardware. | The technologies acquired are local which will erase most constraints on the projects dead-line and budget. The technology is considered of good qualities which will give it the ability to handle most situations. |
| | | Score: 95 | Score: 90 | Score: 90 | Score: 95 |
| Economic feasibility: Cost to develop: | 10 | 85,600₺ | 90,450₺ | 92,600₺ | 101,723₺ |
| Payback period (discounted): | | 3.2 | 3.5 | 3.7 | 4.38 |
| Net present value: | | 22,300₺ | 28,000₺ | 29,300₺ | 156,579₺ |
| | | Score: 90 | Score: 90 | Score: 85 | Score: 90 |
| Schedule Feasibility | 10 | 7-8 Months | 8-9 Months | 9 Months | 8 Months |
| | | Score: 90 | Score: 85 | Score: 80 | Score: 95 |
| Maintainability : Probability of successfully completing a | 5 | The system able to be modified or upgrade later | The system able to be modified or upgrade later | The system able to be modified or upgrade later | The system able to be modified or upgrade later according to the |

| | | | | | |
|--|-----|---|---|---|--|
| repair action in each amount of time | | | | | requirements needed |
| | | Score: 90 | Score: 90 | Score: 90 | Score: 95 |
| Dependability: Availability of the system | 10 | The system is available during working hours. | The system is available during working hours. | The system is available during working hours. | The system is available during the working hours and upon motion detection. |
| | | Score: 95 | Score: 100 | Score: 100 | Score: 100 |
| Reliable: How much we can trust the result shown by the system | 20 | The system achieved an accuracy with 97% | The system achieved accuracy near to 90.8% | The precision score reached up to 91.7% | After training the system with the dataset that contains all cases that are possible to happen, the system will give reliable and accurate results each and every time. Because the dataset will be trained to detect even the trickiest situations the fault rate is significantly reduced. |
| | | Score: 95 | Score: 85 | Score: 90 | Score: 95 |
| Response Time: The period between when a user performs an action and when the computer begins to display the result. | 15 | 0.067s | <1s | Not mentioned. | The system has yet to be tested which means the results are still unknown, but it is known from comparing other methods and systems that the response time will be suitable and very fast. |
| | | Score: 95 | Score: 80 | Score: 0 | Score: 90 |
| Rank: | 100 | 93.1 | 87.5 | 77.5 | 94.4 |

4.9 Conclusion

A feasibility study's primary goal is to determine whether a constructed system is financially viable and, if so, whether it will succeed or fail. Method 4 is a brief view of the proposed system and its viability. After comparing methods and solutions it was decided to use method 4 for building the system, it gave accurate results, suitable costs and fast pay back in benefits.

CHAPTER 5

System Modeling

The process of constructing abstract models of a system, each of which presents a different view or perspective on that system

5.1 Context diagram

The Context Diagram depicts the system under study as a single high-level process and the system's connection with other external entities.

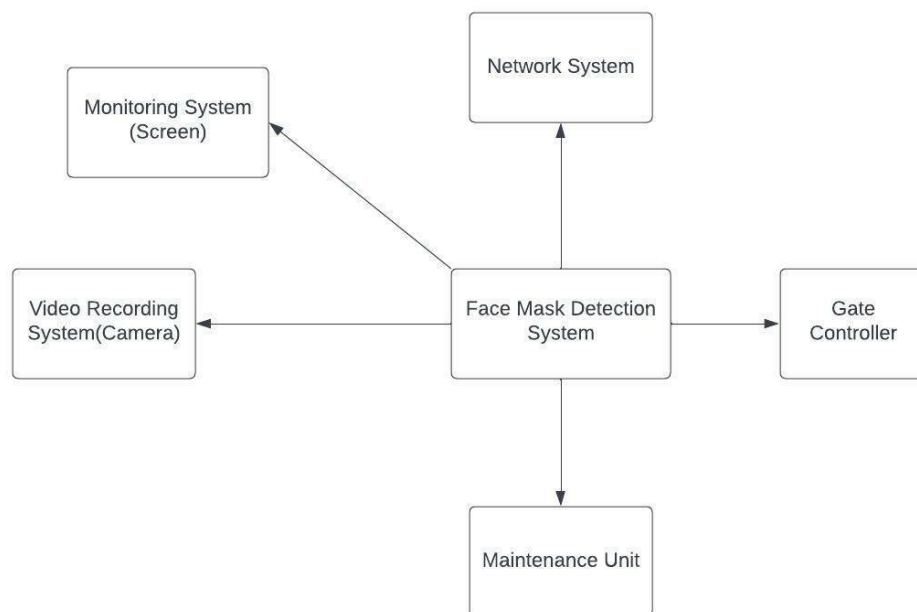


Figure 0.1: Context Diagram for the System

It is worth mentioning that:

- Network system: Support integrating the proposed system to Networks.
- Gate Controller: The gate in which will be controlled to be opened or closed according to the result of the system proposed.

- Maintenance unit: The system for self-checking errors and fixing and updating the proposed system when it is needed.
- Video recording system & Monitoring system: Include cameras needed for the proposed system in order to capture the frame, taking into consideration these resources are available in the organization for which the project will be built. The cameras will automatically save and store the camera footage.

5.2 Use case modeling

A use-case model represents how various types of users interact with a system in order to solve an issue. As a result, the users' interactions with the system.

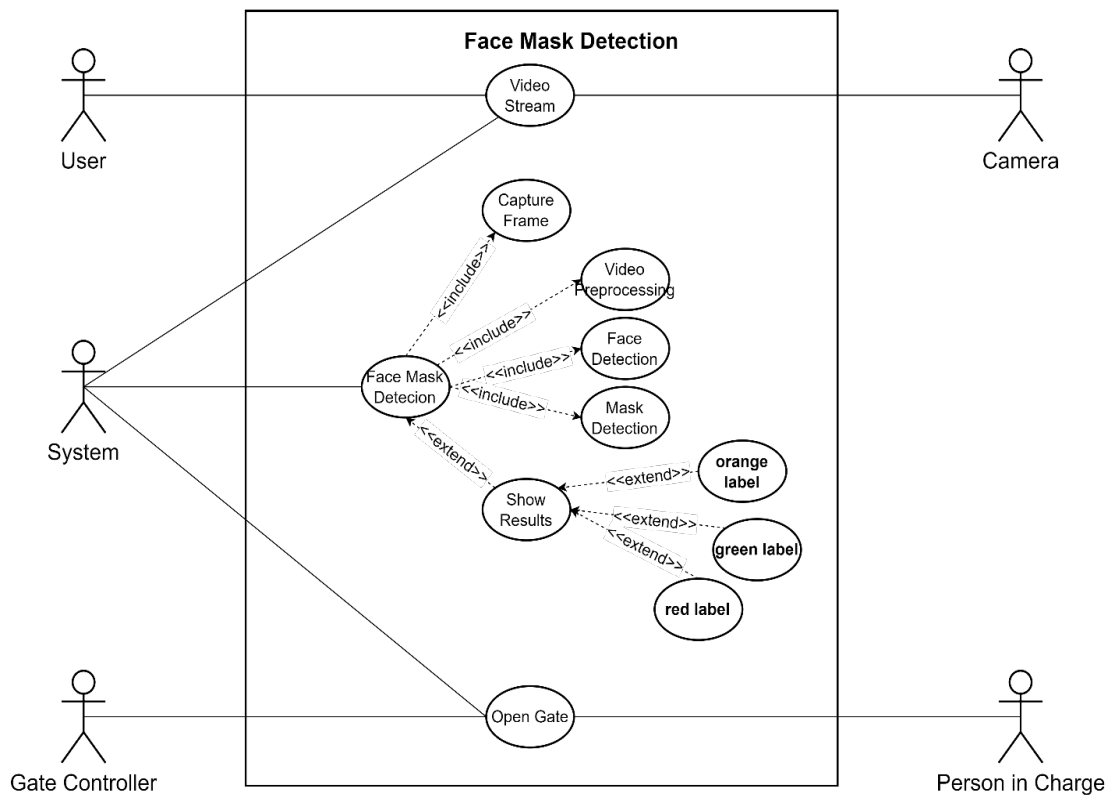


Figure 0.2: Use Case Diagram, Face Mask Detection System

It is noteworthy that green labels indicate that person is wearing mask, red label for person who is not wearing mask while orange shows an undetected case.

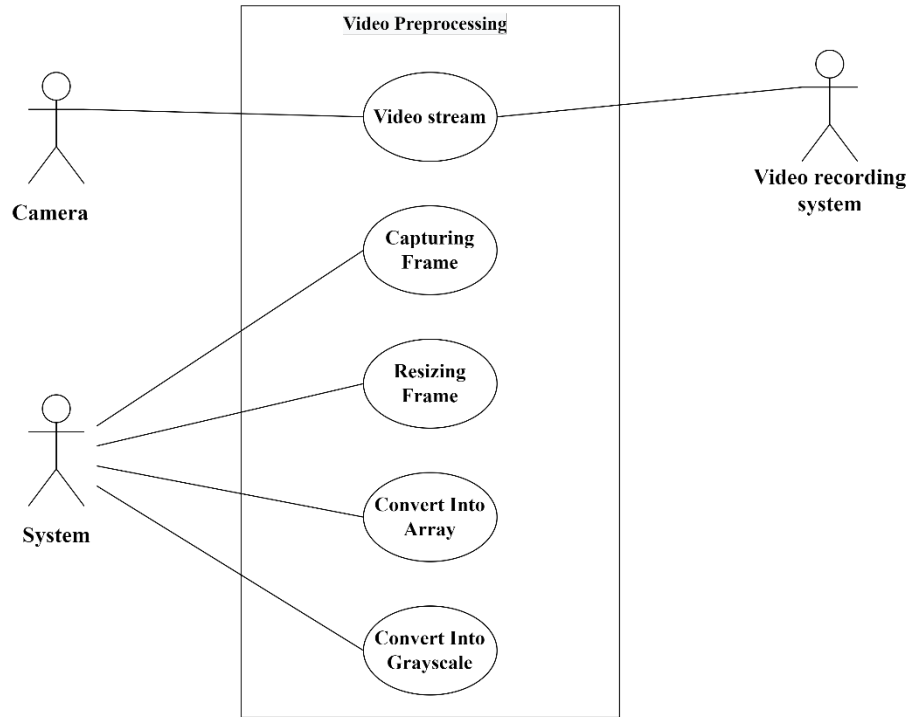


Figure 0.3: Use Case Diagram, Video Preprocessing

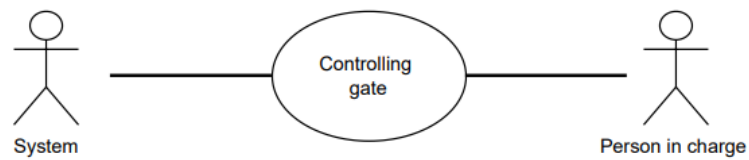


Figure 0.4: Controlling Gate, System and Person in charge

Table 0.1: Controlling Gate, System and Person in charge

| Controlling gate | |
|------------------|---|
| Actors | System, Gate controller |
| Description | The system will open the gate if the person was wearing a mask, otherwise the gate will stay closed |
| Data | Video stream |
| Stimulus | Detecting the mask |
| Response | Showing on screen the green label and sending an order to open the gate |

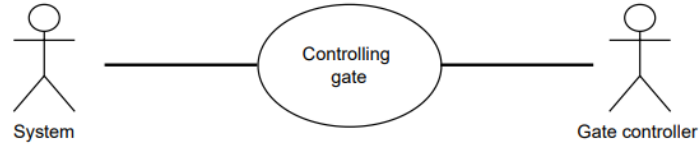


Figure 0.5: Controlling Gate, System and Gate Controller

Table 0.2: Controlling Gate2, System and Gate Controller

| Controlling Gate | |
|------------------|---|
| Actors | System, Person in charge |
| Description | The system will automatically open the gate if the person was wearing a mask. Given the case that the proposed system is not able to detect the case or control the gate, the system will notify the person in charge to take the responsibility of controlling the gate manually |
| Data | Video stream |
| Stimulus | System could not control the gate |
| Response | Showing on screen the label where a red label is for no mask available, green for mask and orange for an undetected case |

5.3 Class Diagrams

A class diagram is a diagram that specifies and offers the overview and structure of a system in terms of classes, attributes, and methods, as well as the relationships between different classes.

5.3.1 Classes and associations

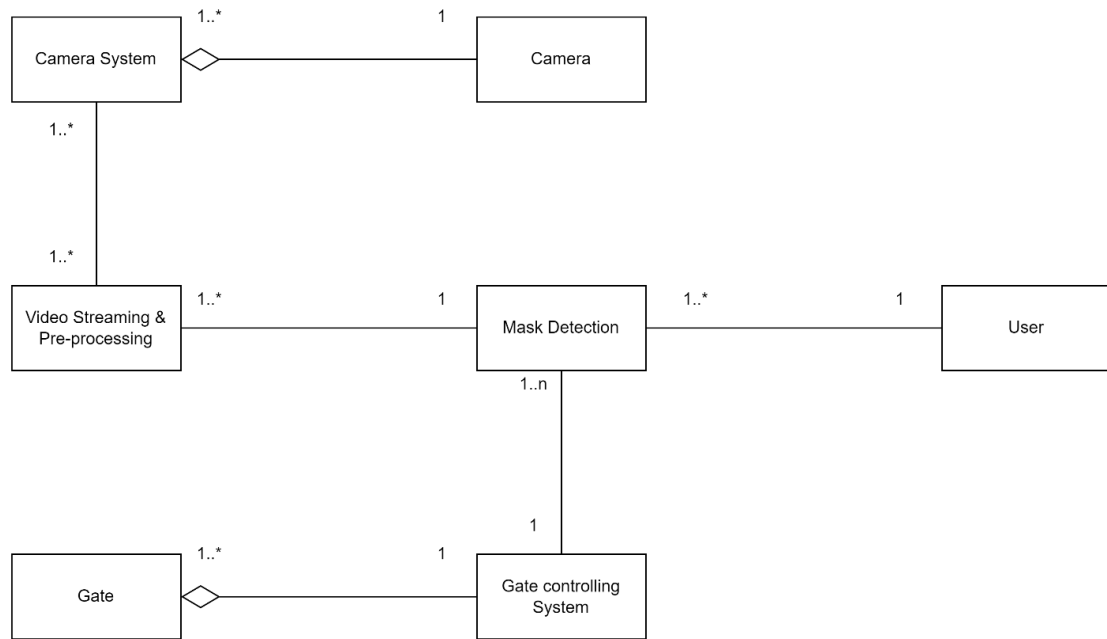


Figure 0.6: Classes and their relations in the proposed system

5.4 Class diagram

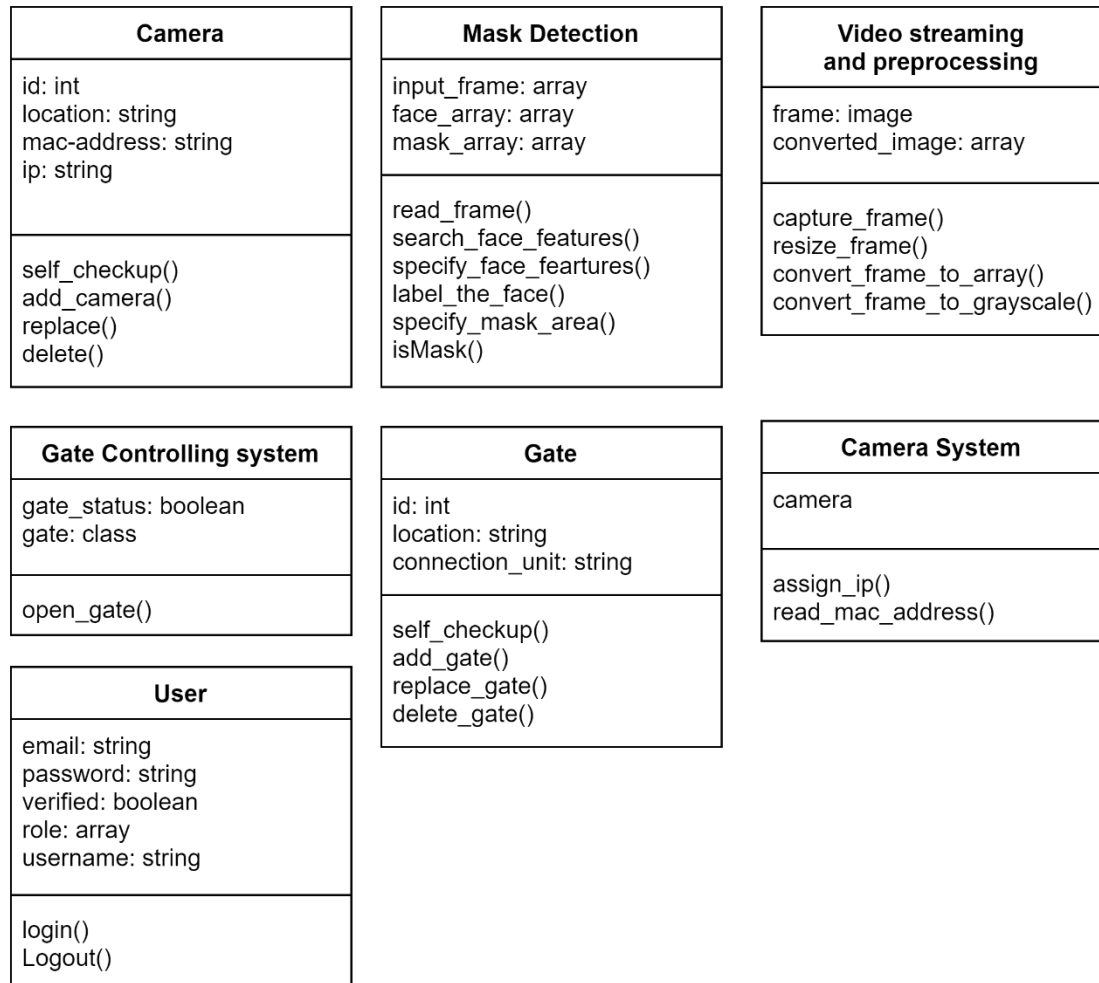
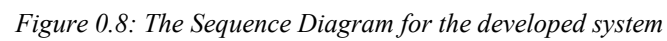


Figure 0.7: Class Diagram of the developed system

Sequence diagram is used to construct event sequences between objects for a certain outcome and demonstrates object collaboration. First diagram explains the whole face mask detection system, the second one explains the video preprocessing step and the output of the step.



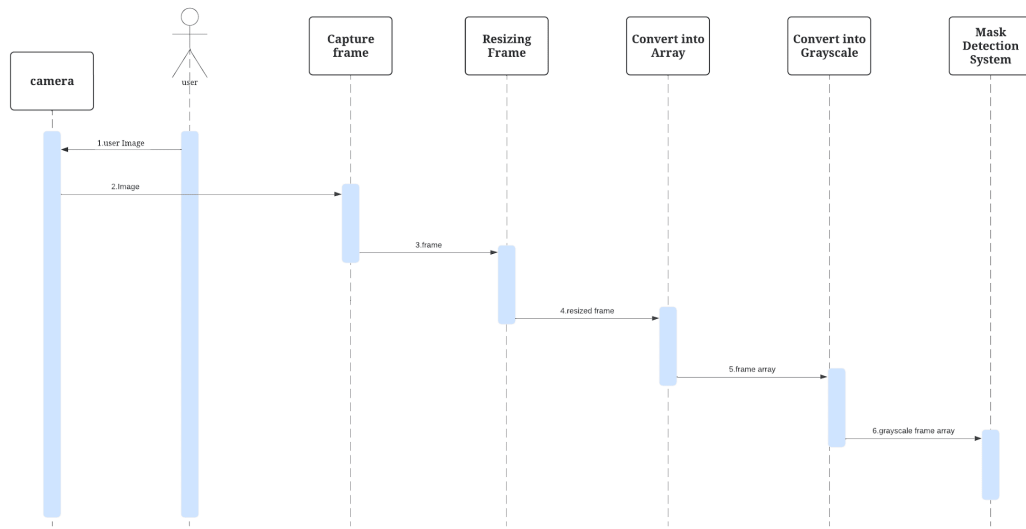


Figure 0.9: The systems Video Preprocessing Sequence Diagram

As shown in the **figure 5.9**, the sequence diagram describes the video preprocessing operation. After the steps of the process are done the output which is grayscale frame array will be transferred to the rest of the system (object detection, face detection, mask detection and the gate controller).

5.6 State diagram

A state diagram represents the behavior of a system by considering all the various states of an object once an event occurs. This behavior is represented and studied as a sequence of events that takes place in one or more states.

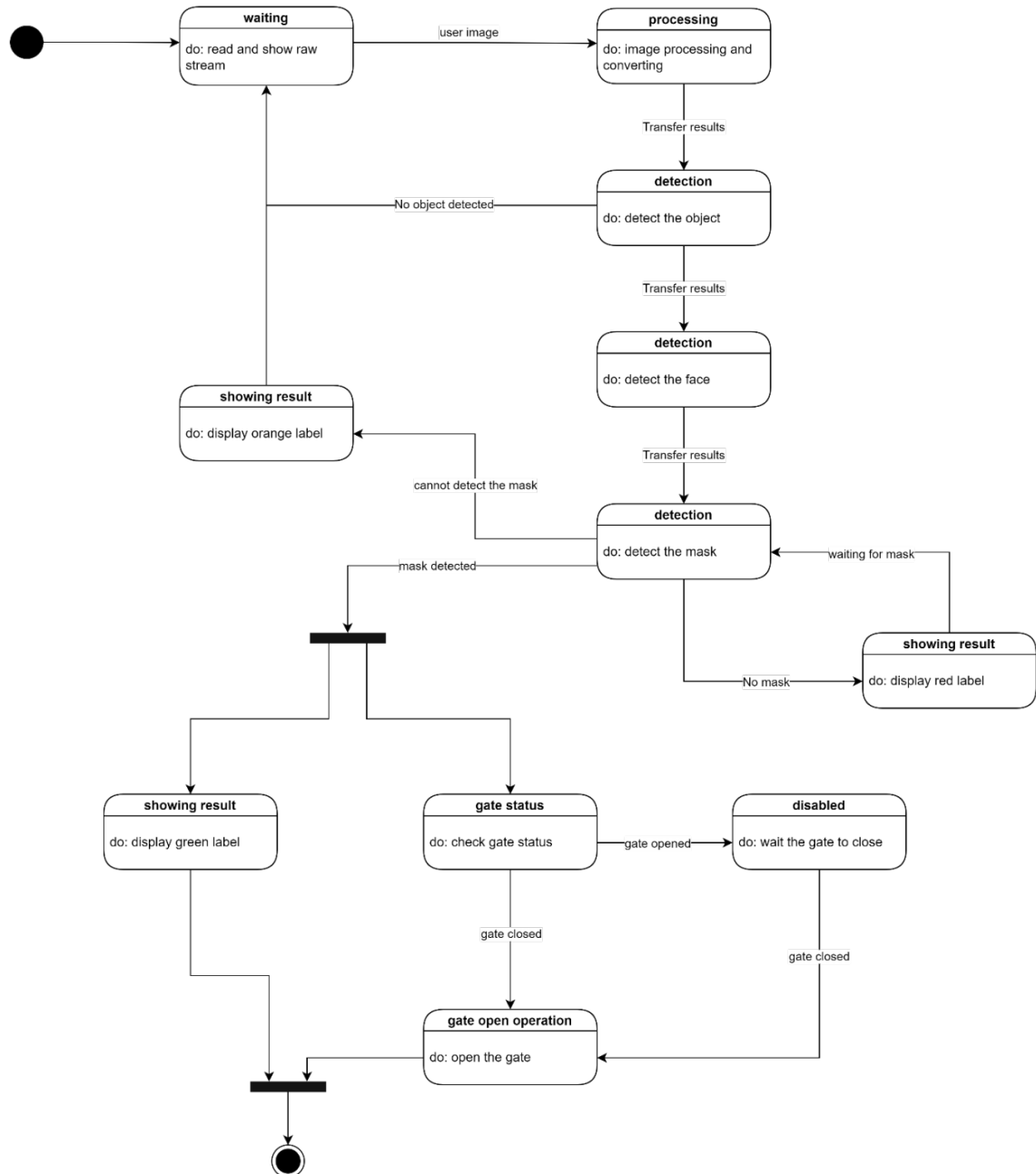


Figure 0.10: The systems State Diagram

Table 0.3: State Diagram Descriptions

| State | Description |
|----------------|--|
| Waiting | The system will read the stream from the camera system and will wait until the person appears in front of the camera |
| Processing | The system will resize the frame to fit into the model, then convert the frame into array and grayscale array |
| Detection | The system will detect the object, then the face, and finally the mask |
| Showing result | As mentioned before, the system will show three labels which are green, red, or orange |
| Disabled | In this state the system will wait for the gate to close, to open it again for the next oncoming person |
| Gate status | The system will check the gate's status either it is open or closed |
| Gate open | The system will open the gate |

5.7 Activity diagram

An activity diagram is essentially a flowchart that shows the flow from one activity to another. The process can be described as a system operation.

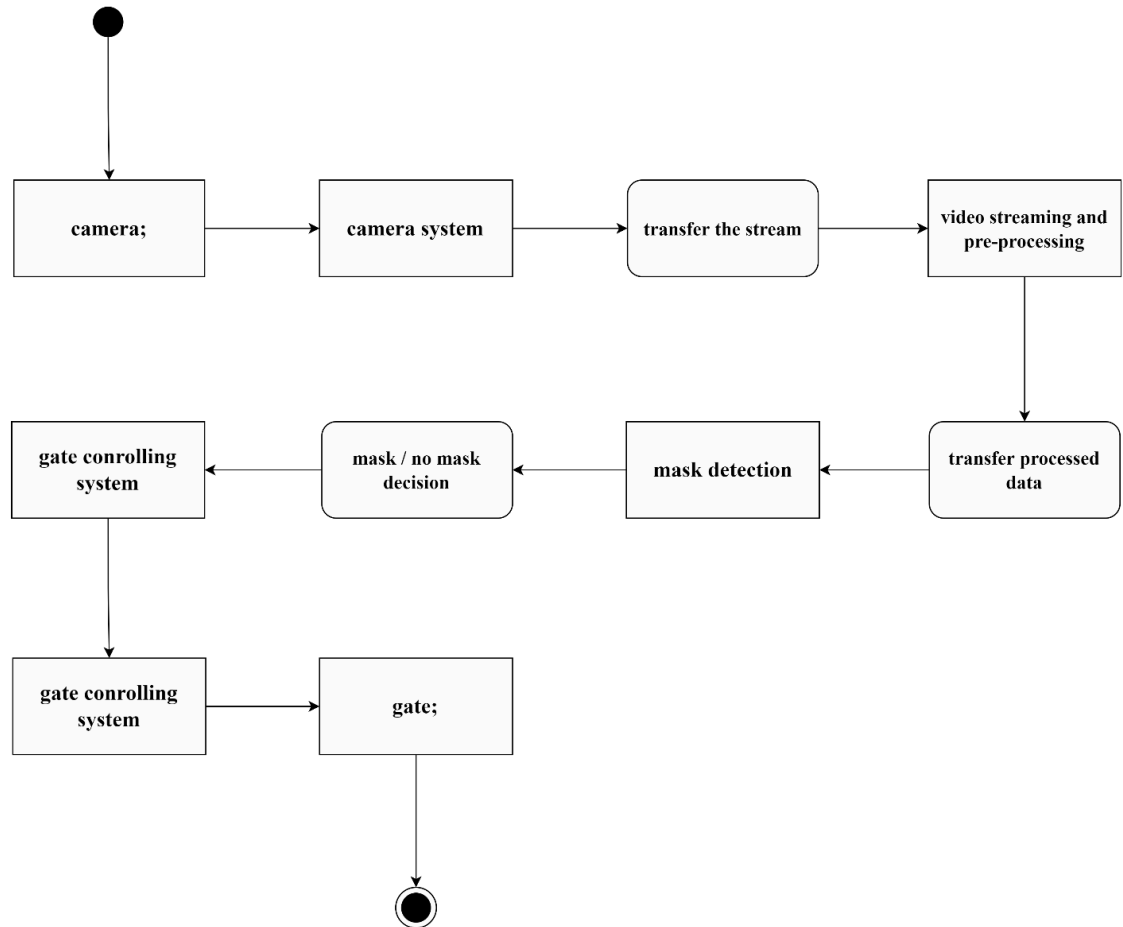


Figure 0.11: Activity Diagram for the proposed system

CHAPTER 6

Project Scheduling

6.1 Task Network

The task network, a visual depiction of the task flow for a project, is shown in the table below. Sometimes it serves as the method for entering job dependencies and sequence into an automated project scheduling application.

Table 0.1: Project Tasks of developed system

| Task ID | Task Name | Duration /week | Start | Finish | Predecessors |
|---------|---|----------------|--------------|--------------|--------------|
| 1 | Market Research | 3 | Fri 11/02/22 | Thu 03/03/22 | |
| 2 | Determine User and System Requirements and Stakeholder | 3 | Fri 04/03/22 | Thu 24/03/22 | 1 |
| 3 | Determine Functional and Non-Functional Requirements | 2 | Fri 25/03/22 | Thu 07/04/22 | 2 |
| 4 | Requirements Checking (validation) | 1 | Fri 08/04/22 | Thu 14/04/22 | 3 |
| 5 | Feasibility Study | 2 | Fri 15/04/22 | Thu 28/04/22 | 4 |
| 6 | Project Planning | 2 | Fri 29/04/22 | Thu 12/05/22 | 5 |
| 7 | Interaction Models (User Case and Sequence) and Context Diagram | 2 | Fri 29/04/22 | Thu 12/05/22 | 5 |
| 8 | Structural Model (Class) | 1 | Fri 13/05/22 | Thu 19/05/22 | 7 |
| 9 | Behavioral Model (Activity and State) | 1 | Fri 20/05/22 | Thu 26/05/22 | 8 |
| 10 | Mock-up and Interfaces | 1 | Fri 27/05/22 | Thu 02/06/22 | 9 |
| 11 | Planning Report | 0 | Fri 13/05/22 | Fri 13/05/22 | 6,10 |
| 12 | Database and Dataset Preparation | 2 | Fri 13/05/22 | Thu 26/05/22 | 11 |
| 13 | Building object Detection Model | 2 | Fri 27/05/22 | Thu 09/06/22 | 12 |
| 14 | Building Face Detection Model | 2 | Fri 27/05/22 | Thu 09/06/22 | 12 |
| 15 | Building Mask Detection Model | 1 | Fri 10/06/22 | Thu 16/06/22 | 13,14 |
| 16 | Training and compiling models | 1 | Fri 17/06/22 | Thu 23/06/22 | 15 |
| 17 | Component Testing | 2 | Fri 24/06/22 | Thu 07/07/22 | 16 |

| | | | | | |
|----|--|---|--------------|--------------|----|
| 18 | Integration the Systems | 3 | Fri 08/07/22 | Thu 28/07/22 | 17 |
| 19 | Evaluation and Modification the Errors | 4 | Fri 29/07/22 | Thu 25/08/22 | 18 |
| 20 | User Documentation | 1 | Fri 26/08/22 | Thu 01/09/22 | 19 |
| 21 | Project Complete | 1 | Fri 02/09/22 | Thu 08/09/22 | 20 |

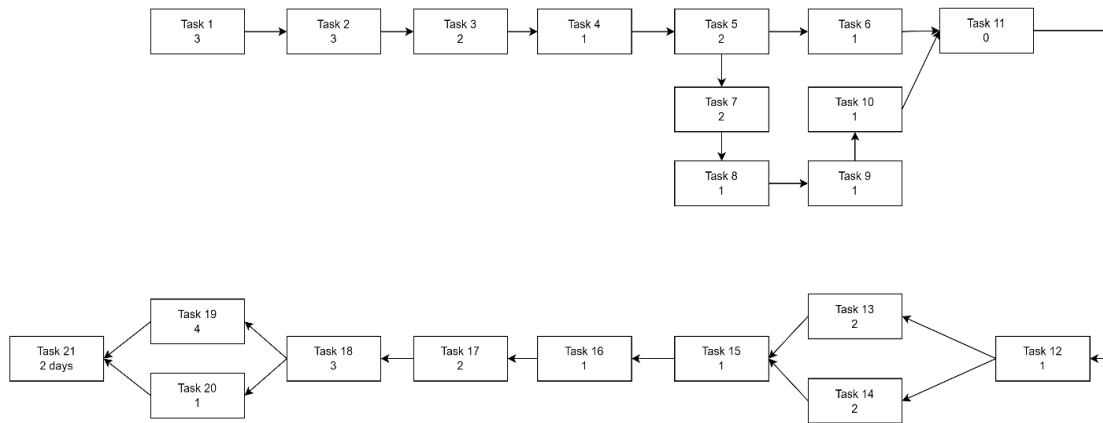


Figure 0.1: Task Network of developed system

Path:

sequence of tasks in a project plan

#1: 1,2,3,4,5,6,11,12,13,15,16,17,18,19,20,21

Cost (duration): $3+3+2+1+2+2+0+1+2+1+1+2+3+4+1+1=29$ week

#2: 1,2,3,4,5,6,11,12,14,15,16,17,18,19,20,21

Cost (duration): $3+3+2+1+2+2+0+1+2+1+1+2+3+4+1+1=29$ week

#3: 1,2,3,4,5,7,8,9,10,11,12,13,15,16,17,18,19,20,21

Cost (duration): $3+3+2+1+2+2+1+1+1+0+1+2+1+1+2+3+4+1+1=32$ week

#4: 1,2,3,4,5,7,8,9,10,11,12,14,15,16,17,18,19,20,21

Cost (duration): $3+3+2+1+2+2+1+1+1+0+1+2+1+1+2+3+4+1+1=32$ week

Critical Paths:

Longest cycle of activities in a planned project, which has to be done on schedule to be complete upon the due date of the project.

Our critical paths: #3 and #4

6.2 Gant Chart

An illustration of a project timetable is a Gantt chart. The timetable for the project is shown in the chart below.

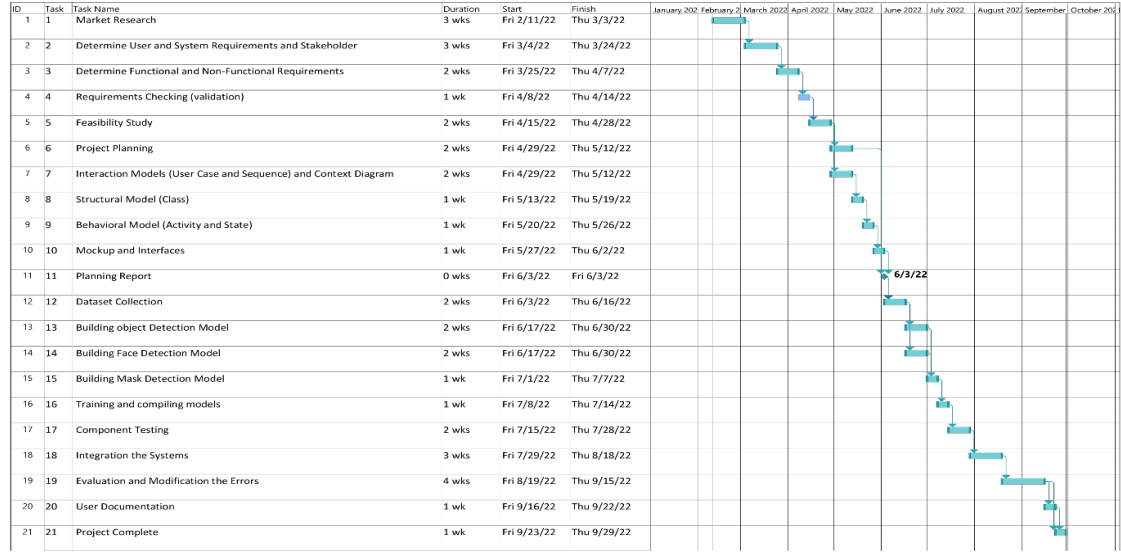


Figure 0.2: Gant Chart of developed system

6.3 Pert Chart

This section will demonstrate the Pert chart, a statistical tool used in project management that was created to evaluate and display the steps necessary to complete a certain project.

Table 0.2: Pert Char of developed system t

| Task# | Task | Preceed ID | Succ. ID | Estimated time |
|-------|--|------------|----------|----------------|
| 1 | Market Research | 1 | 2 | 3 |
| 2 | Determine User and System Requirements and Stakeholder | 2 | 3 | 3 |
| 3 | Determine Functional and Non-Functional Requirements | 3 | 4 | 2 |
| 4 | Requirements Checking (validation) | 4 | 5 | 1 |
| 5 | Feasibility Study | 5 | 6 | 2 |
| 6 | Project Planning | 6 | 7 | 2 |

| | | | | |
|----|---|----|----|---|
| 7 | Interaction Models (User Case and Sequence) and Context Diagram | 6 | 8 | 2 |
| 8 | Structural Model (Class) | 8 | 9 | 1 |
| 9 | Behavioral Model (Activity and State) | 9 | 10 | 1 |
| 10 | Mockup and Interfaces | 10 | 11 | 1 |
| | Dummy Task | 7 | 12 | |
| | Dummy Task | 11 | 12 | |
| 11 | Planning Report | 12 | 13 | 0 |
| 12 | Dataset Collection | 13 | 14 | 2 |
| 13 | Building object Detection Model | 14 | 15 | 2 |
| 14 | Building Face Detection Model | 14 | 16 | 2 |
| | Dummy Task | 15 | 17 | |
| | Dummy Task | 16 | 17 | |
| 15 | Building Mask Detection Model | 17 | 18 | 1 |
| 16 | Training and compiling models | 18 | 19 | 1 |
| 17 | Component Testing | 19 | 20 | 2 |
| 18 | Integration the Systems | 20 | 21 | 3 |
| 19 | Evaluation and Modification the Errors | 21 | 22 | 4 |
| 20 | User Documentation | 21 | 23 | 1 |
| | Dummy Task | 22 | 24 | |
| | Dummy Task | 23 | 24 | |
| 21 | Project Complete | 24 | 25 | 1 |

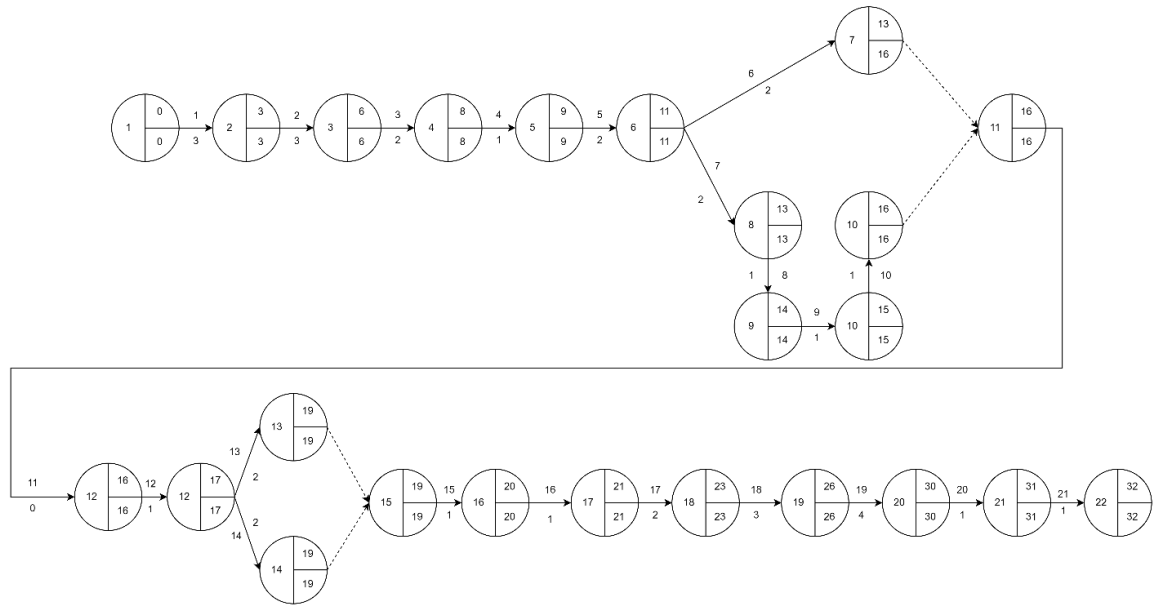


Figure 0.3: Systems Pert Chart of developed system

CHAPTER 7

Mock-up and Interfaces

7.1 Mock-up

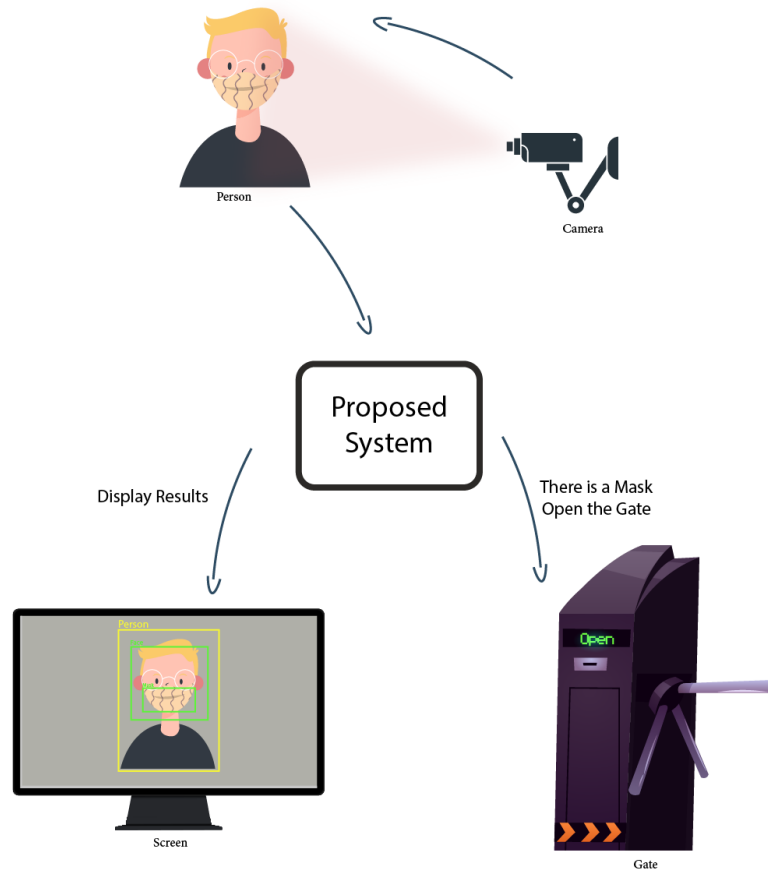


Figure 0.1: Mock-up 1 of the system

In **figure 7.1** The layout of the system here shows how the system will work in real-life. The camera will be directed towards the face of the user, when detected, the proposed system will analyze through the processing of the system in which the system will either give an order to open or close the gate according to the results displayed on the screen.

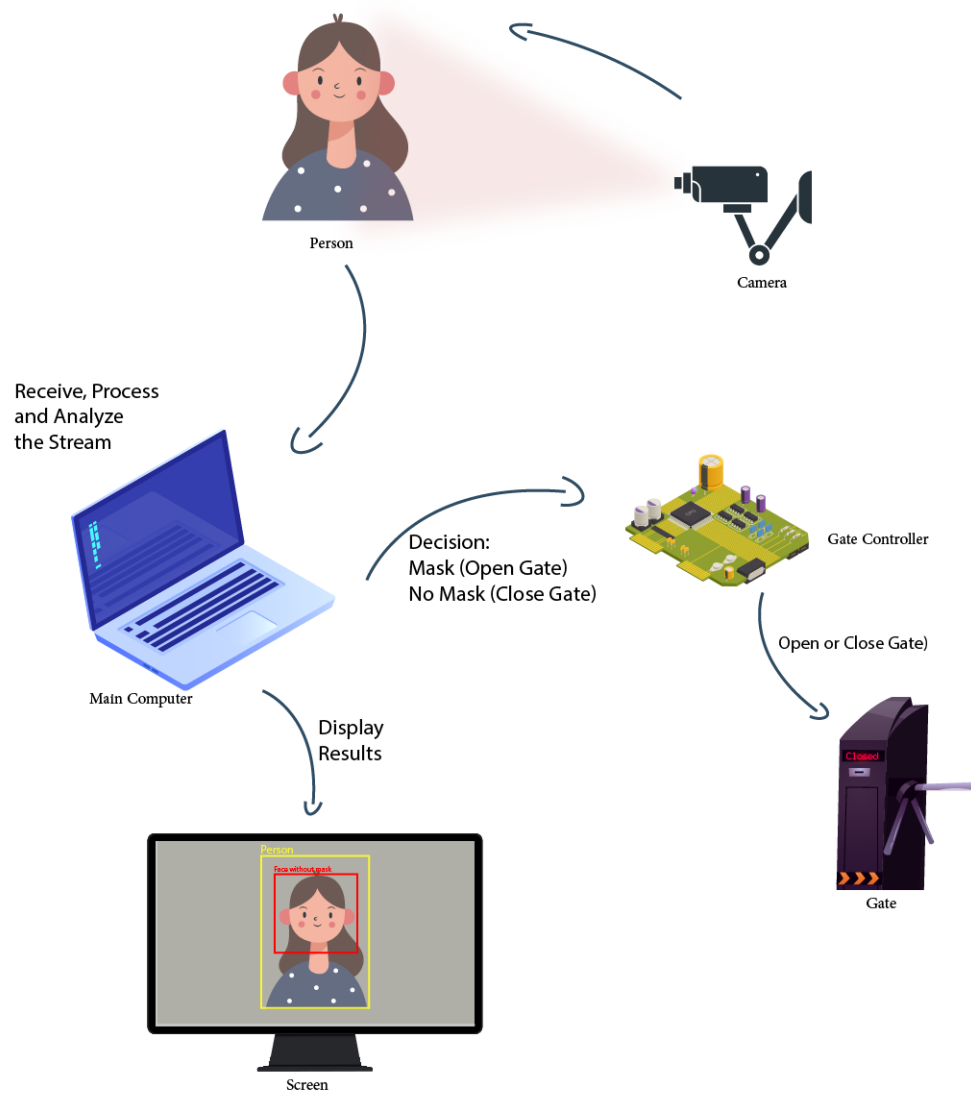


Figure 0.2: Mock-up 2 of the system

In **figure 7.2** the system is described in more detail on how it is working and the process that goes inside the system. As explained in the previous figure, the proposed system will open the gate only if the person is wearing a mask correctly.

7.2 Interfaces

- Admin interface

The admin is eligible to view the recorded video and to save and print pictures if needed. The admin's interface will also show a Camera management button which will give the ability to handle the different cameras used and a user management button that manages the users. There will also be a help button which contains the manual of the system.

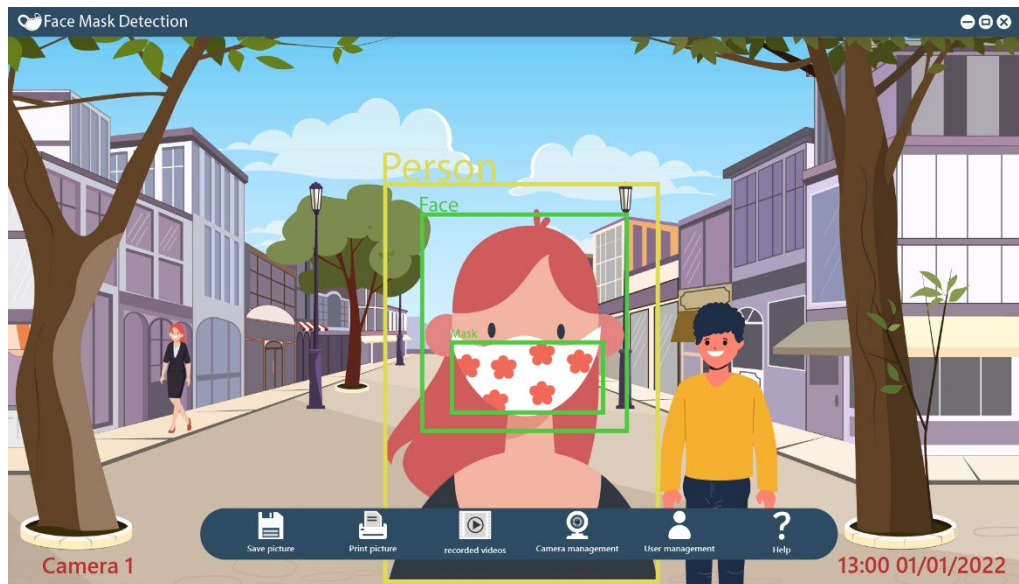


Figure 0.3: Systems Admin Interface

Figure 7.3 is showing what the screen will display while on Admin mode, in this mode the features that are described above are only visible to the admin screen.

- User interfaces

The user is eligible to save and print pictures if needed. There will also be a help button which has the guides of the system.

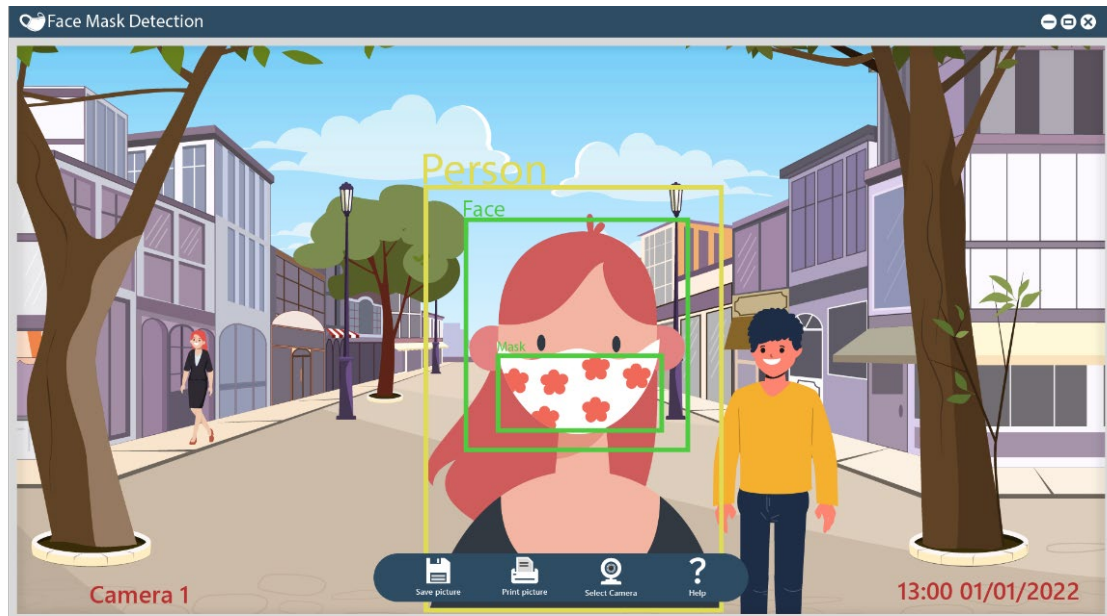


Figure 0.4: Systems User Interface

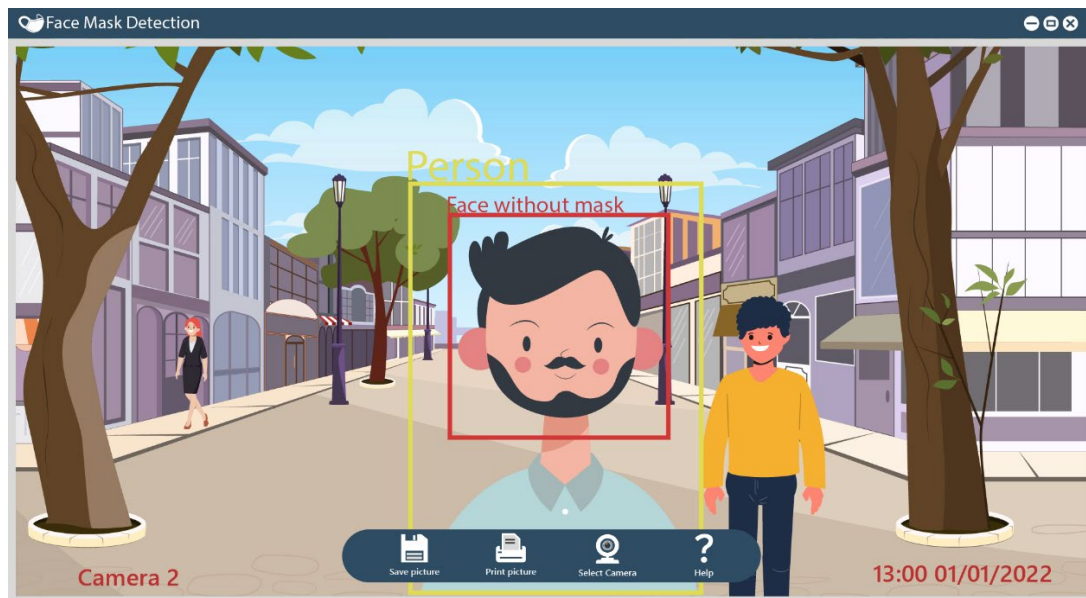


Figure 0.5: Systems User Interface, showing Red Label



Figure 0.6: Systems User Interface, showing Orange Label

Figures 7.4, 7.5 and 7.6 are what will be shown when in user mode in which, the green label is when the mask is worn correctly and detected by the system, the red label is when the mask is not worn, while the orange is when the system cannot detect the case.



Figure 0.7: Systems Admin Interface, showing User Management

Figure 7.7, shows the user management in the admin interface where the main admin can remove/add an admin and view number of users and remove when needed.

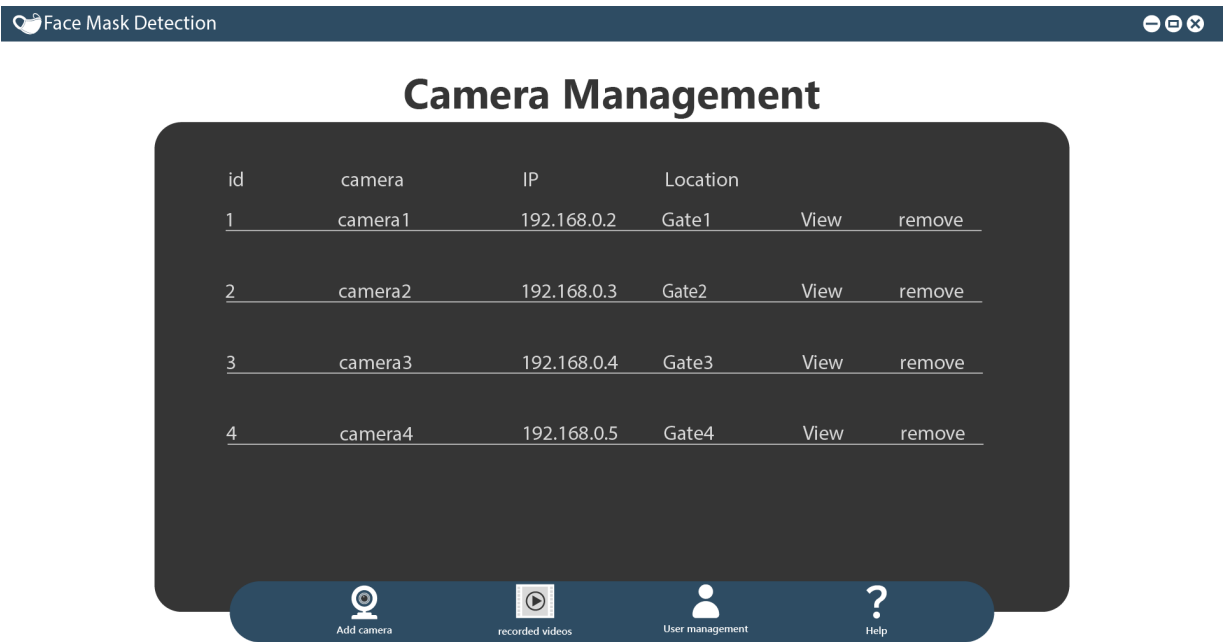


Figure 0.8: Systems Admin Interface, showing Camera Management

Figure 7.8, which is included only in the admin interface, will show the number of cameras available and the ability to remove one or all of them.

CHAPTER 8

Image Classification System

Because image classification is a crucial part of this system, we first need to explain what it is and what it does.

Image classification or (Image Recognition) can be described as the assignment of classes or labels to an image, which essentially helps in assigning each image a keyword and categorizing these images as is. While it is easy for a human brain to automatically detect what the object being seen is, it is near impossible for a computer to do the same without feeding it the algorithm and data to imitate how the brain works, this suggests that manually classifying the images is a hectic task especially when they are massive in numbers, hence the need to use a classifier in certain models.

Image classification is used in many fields, particularly in the system of face mask detection where it is used to process the image of a person's face, facial features, mask, etc.

The training of classifiers feed on massive amounts of data as well as analyzing this data and extracting the needed features. Since image classification is a complicated and thorough task, it is important to note that some standard criteria regarding accuracy as well as time efficiency should be taken into consideration and are evaluated accordingly. Data type, data size, and expected results are most important to determine which approach is the most efficient.

8.1 Labeling of image classification:

As mentioned above, image classification is extracting context from an image by assigning a label to what the image is.

There are two classifications for labeling: single label and multiple labels. To begin with, single label classification has one label to each picture that the model sees, meaning only one criterion is analyzed and considered when classifying.

While a multi label classification as stated by its name is where each image contains more than one label and can be placed into several categories.

8.2 Types of image classification:

There are two types of image classification;

- **Supervised:** Supervised classification is the classification of images through visually picking out samples of data in an image and designating each to pre-chosen categories. It is called supervised because of the need for a human oversight to inspect, since the greater number of the data given is raw data, its input and output need to be labeled during the training phase by said human oversight. As the model learns what ties the input and the output and what the relationship between them is, only then can it be used for new unseen datasets and from then on have the ability to determine the context of the image or say predict the outcomes. A predictive supervised model is trained to recognize wanted features and categorize them according to its initial training during the first phases. By learning the patterns between the input and the output the model can be used to classify different types such as images or words, and predicting outcomes through the learning patterns. This works by giving the classifier a prediction through the input data and then correcting it when the prediction is wrong, the training process progresses until it reaches a low error rate. Assuming the images next to their labels are identical to their class labels, the classifier knows what each category is supposed to look like. Thus, if the prediction is incorrect certain techniques can be applied to correct it. Decision trees, Random forests, K-nearest neighbors and Neural networks are some examples of supervised classification.
- **Unsupervised:** As the name states is an unsupervised way of classification which means it works in an automatic manner that does not require the examination of a human being, in other words it is a self-taught learning mechanism. Unsupervised classification works with unlabeled raw data and is only given the input, unlike the supervised classification where both the input and output are expected to be labeled. Considering the vast amount of unlabeled data on the web, this classification method comes in handy. The computer uses methods to determine what each pixel is and to which other pixel it is related to, to then group them into

their classes. Although unsupervised classifying can be unpredictable, it allows for more complex processing tasks. Unsupervised classification uses an algorithm called clustering, even though the model does not contain any labels it clusters the information already given into groups and looks for similarity in them, then it predicts what these clusters of groupings are. Yet this is the same reason why the unsupervised method cannot be reliable accuracy wise, as there is no way to be 100% sure on whether the prediction is correct or not. Sometimes these clustering groups can be very similar to the point the model can no longer differentiate between several groups of clusters, in which case an estimated expected number of clusters should be provided. Examples of unsupervised classification methods are; K-means, Self-organizing maps, Isodata, etc.

- There is also what is called a Semi-supervised classification where only a small part of the raw data is labeled and the rest is left for the model to train itself from the remains of these labeled images.

8.3 Image Classification Steps:

Mask detection (classification) systems have some common steps (shown in the figure below) that can be summerized as follows:

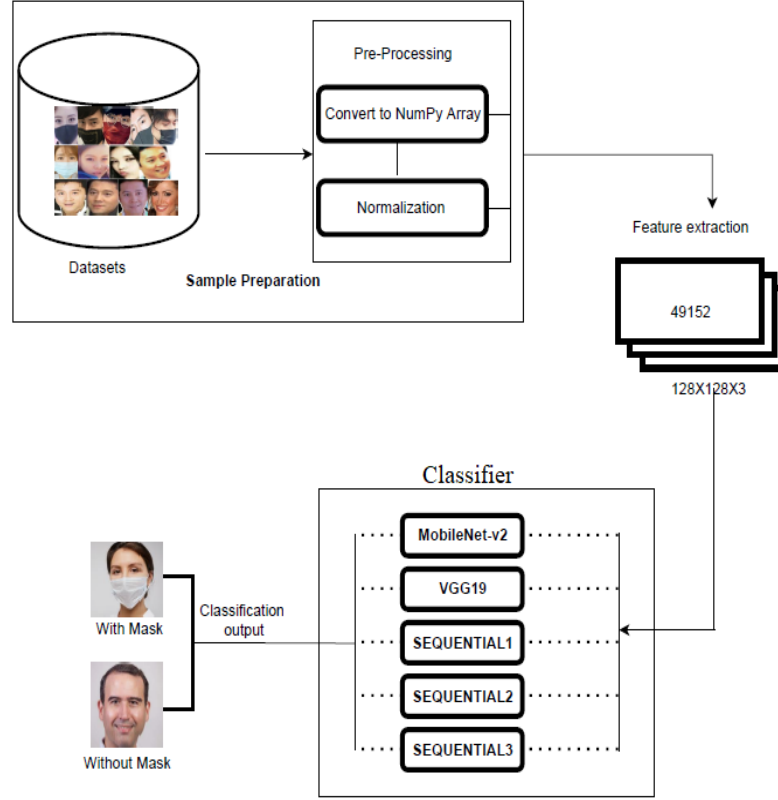


Figure 0.1: Mask Detection System (Multi Classifier)

1- Preparation of datasets:

Data is a crucial component of training any model, and experimenting with a variety of datasets is the key to success in the field of machine learning. However, it can be challenging to find a suitable dataset for each type of machine learning project. So, using an existing dataset that has been cleaned to produce the best results, we combined several datasets to create a new dataset.

The following datasets were used:

- Dataset 1:

This dataset cleared and equally distributed across each class created by Vijay Kumar [18] the dataset has three class, with mask 2994, 2994 of images without masks and another one with masks worn incorrectly. Only the first two categories were utilized because the

model works with just two classes, with and without mask. Thus, the total size of the dataset is 10549 divided into; 5599 images with mask and 4950 without mask. The dataset contains images with identical dimensions each measuring 128 x 128 pixels.

- Dataset 2:

This dataset was created with the goal of developing an algorithm that can quickly determine whether or not someone is wearing a face mask. Therefore, this dataset has scraped images from Google as well as Jessica Li's CelebFace dataset by ASHISH JANGRA [19]. The dataset consists of almost 12K images with two classes including ones with masks and others without mask. The dimension of the images is not fixed so the images are of different sizes, where the smallest size is 25x25 and the largest one is 563x563.

- Dataset 3:

The RMFD dataset, Kaggle datasets, and photos gathered via the Bing search API were all combined to create the dataset. Additionally, it has 4079 photos split into "with mask" and "without mask" classes. 1930 images are in the "without mask" class, whereas 2149 images belong to the "with mask" class. "Chandrika Deb" was the one who gathered the data [20]. The photos' sizes vary, with the lowest being 26x37 and the largest is 4734x5412.

- Dataset 4:

Face Mask Detection (FMD) [21] dataset has a total of 3833 images which are divided into two categories, 1915 images with masks and 1918 without a mask. The dimensions of the images in this dataset are not similar, it contains different dimensions, where the smallest size is 26x37 and the largest is 4734x5412. This dataset has images of people with original masks so it is very representative in the real world.

- Dataset 5:

Lastly, all of the mentioned data sets were combined to generate the "All-in-one" dataset. It has two groups of 25638 photographs each, including 12751 images without a mask and 12887 images with a mask. The photos all have varying sizes, with the smallest being 25x25 and the largest being 4734x5412. Preprocessing is the second phase in the preparation of datasets. It primarily focuses on removing the unused and distracting portions of the input (the images, in our case). In addition to the previous preprocessing

procedures, manipulation functions were employed to improve efficiency, in this case using the "OpenCV" package. The output array is also supplied into the TensorFlow preprocess function to ensure the preprocessing quality.

2- Feature extraction:

This procedure takes raw data and converts it into what is considered a format the computer and machine learning software can comprehend. This process manipulates data by enhancing the performance. Because similar data distribution is needed for each pixel, the image array is passed into the "preprocess input" function of TensorFlow in "tf" mode and the output value of these arrays range between 1 and -1. This step speeds up the convergence while training the network. Following these steps, the final array includes 49152 features since the dimensions are (128, 128, 3).

What is left to do is to read every image from the dataset in a loop, then saving them all in one single numPy array and dividing the array into two parts, one part will be for training the models and the other part will be for testing.

3- System classifier:

To have a successful classification system, one must choose an appropriate classification model. In machine learning, a classification model, often known as a classifier, predicts categorical labels (classes). A classification model aims to derive something meaningful from the values that are observed. It will attempt to predict the value of one or more outputs when given one or more inputs this can be done with a training dataset with many samples of inputs and outputs in which it is necessary for a classifier to be able to learn and predict outcomes. The output labels are then used on the dataset.

Currently Convolutional neural networks are considered as the State-of-the-art computer-based systems in general and classification models in particular. In more detail, Convolutional neural networks, often known as (CNN), defied predictions and recently overtook other types of categorization algorithms to claim the title of the most sophisticated computer vision. These convolutional neural network models exhibit exceptional performance when used in computer vision tasks like image classification.

Convolution layers, pooling layers, Dropout layers, activation function layers, flatten layers, fully linked layers, and other building blocks are some of the components that

make up the CNN architecture. A common CNN model design consists of a stack of several convolutional layers, a pooling layer, one or more fully connected layers, and finally one or more additional layers.

Finding the most effective kernels is the aim of the CNN model in the convolution layer. The performance of a model is greatly influenced by the size of the kernels, the quantity of kernels, the padding, and the stride, which are all hyperparameters. The performance of the CNN model as a whole is significantly impacted by the Dropout value and Dense rate in the Dense and Dropout layers. To determine the best hyperparameters, approaches hyperparameter optimization approaches can be utilized.

- **CNN Layers:**

In this section we will discuss mainly the most common CNN layers:

1- Input Layer:

The input layer comes first and the output layer comes last in any neural network. Images are inputs in this instance and are contained in the input layer. The first convolutional layer receives these images as an input and the output of the first layer will be used as the input for the second layer, and so on. Up until the final layer, this process will continue.

2- Convolutional Layer:

We refer to the patterns as "kernel," "filter," or "feature detector" in the terminology of convolutional neural networks. Computers read images as pixels and it is expressed as a matrix ($N \times N \times 3$) - (height by width by depth). Images make use of three channels (RGB), so that is why we have a depth of 3.

A collection of teachable filters is used in the 2D convolution layer. Different filters that detect different features are convolved on the input file to produce a set of activation maps that are passed to the next layer in the CNN. By applying a filter to an input to create a feature map that summarizes the presence of detected features in the input. One of the building blocks of image processing, it. The 2D convolution layer is used for all of the processes involved in blurring or sharpening, detecting edges, and reducing noise in images.

Following the convolution operation on our images, pooling is then applied to the convolution's results, which is the convolved image.

3- Pooling Layers:

Pooling is used to reduce the size of the image. There are two types of pooling:

- **Max Pooling:** It merely involves choosing the highest value from the matrix of the specified size (default size is 2 X 2). This technique is useful for removing highly significant or image-highlighted features.
- **Average Pooling:** In contrast to max-pooling, average pooling averages out all of the pixel values in the pooling layer's matrix (the default size is 2 X 2).

At first glance, it might seem like this results in the loss of information, but in reality, more useful information is gained. We can reduce overfitting and accelerate the computation by removing some noise from the data and extracting only the significant ones. Maxpooling is typically used because it performs significantly better than average pooling.

Convolutional layer output is in multi-dimensional shape, whereas dense layer input is in single-dimensional shape 1-D array, so we cannot pass convolutional layer output directly to dense layer. Therefore, between the convolutional and dense layers, we will use the flatten method.

4- Flatten Layer:

Data is flattened when it is made into a 1-dimensional array for input into the following layer. We flatten the output of the convolutional layers to create a single long feature vector. It also has a connection to the dense layer, the final classification model. To put it another way, we connect the final layer to the single line containing all the pixel data. Please take note that flattening input has no impact on batch size.

5- Dense Layer:

A fully connected layer of a neural network. The outputs from the layer above are all received by each neuron in a dense layer, and each neuron then sends one output to the layer above. A vector with an 'm' dimensionality is produced by the dense layer. Changing the vector's dimensions is essentially what a dense layer is used for. Additionally, operations on the vector, such as rotation, scaling, and translation, are applied by dense layers. This is CNN network's final stage.

6- Dropout Layer:

Dropout is a technique used to prevent overfitting or to enhance model generalization. Dropout basically means that a random subset of the dense layer nodes is turned off during

each training cycle. The percentage of nodes to silence at a given time is determined by the dropout rate, which ranges from 0 to 1. A new network architecture is made out of the parent network by temporarily removing all forward and backward connections with dropped nodes. Dropout is used to reduce co-adaptation when multiple neurons in a layer extract the same or very similar hidden features from the input data. In dropout, we randomly disable a portion of a layer's neurons at each training step by zeroing out the neuron values. The percentage of neurons that will be disabled is referred to as the dropout rate.

For Conv2D and Dense layers as well, they require an activation function an activation function is a function helps the neural network to understand intricate patterns from the input data. The activation process ultimately decides what signals should be sent to the following neuron. It receives the output signal from the preceding cell and transforms it into a format that can be used as the input signal for the subsequent cell. The hidden layer activation function that is selected will determine how well the network model learns the training dataset. The kind of predictions the model is capable of making will depend on the activation function that is selected for the output layer. The ReLU function is currently the most popular activation function in neural networks, but there are three other activation functions one might want to take into consideration for use in hidden layers. The fact that ReLU does not activate all neurons simultaneously is one of its biggest advantages over other activation mechanisms. All negative inputs are converted to zero, and the neuron is not activated as a result. Fewer neurons are activated at a time as a result, which greatly increases computational efficiency. As compared to tanh and sigmoid activation functions, ReLU converges six times more quickly. To Perform Output Activation, we use the Softmax activation function when there are two or more classes. The mask detection model is appropriate. Since we have two classes Sigmoid is used for binary classification methods, it is a good option for our model.

- **CNN's Main Parameters:**

The compile method is used to configure a model's learning process. It needs the following parameters:

- **Loss Function:**

A loss function measures how well the neural network models the training data by comparing the target and predicted output values. We strive to reduce this loss between the predicted and target outputs during training. The prediction of the model built is directly correlated with the loss function. The model will yield good results if the loss function value is low. Loss functions can be broadly divided into two categories: classification and regression. Our task in classification problems is to predict the respective probabilities of each class the problem involves. When it comes to regression, on the other hand, our task is to predict the continuous value in relation to a given set of independent features to the learning algorithm. In our project we care about classification loss function we used during trials CategoricalCrossentropy and Binary Crossentropy, because they fit with our model since we are dealing with two classes.

- **Optimizers:**

Can be understood as an algorithm that is used to change the learning rate of a neural network in order to minimize and reduce losses. This results in maximizing the efficiency of the model to its highest potential. Optimizers rely on the model's parameters such as its Weights and learn to change it to reduce the loss and maximize the efficiency. To put it simply, optimizers play around with weights to mold the model into a better form, it links the loss function and the model's parameters together by updating the model according to the output of the loss function. The loss function then acts as a guide on whether it's moving to its right direction or not.

There are many types of optimizers such as:

1- Adagrad:

Adagrad adjusts the learning rate individually to each specific feature which means it is specific to each weight.

- Pro: It is best used with sparse data where there are missing inputs in the dataset and there is no need to update the learning rate manually because it already changes according to the iterations happening
- Con: with Adagrad the learning rate becomes smaller and smaller significantly with time which leads to slow convergence.

2- Adadelat:

Adadelata is an extension of Adagrad. It removes the decaying learning rate problem that is expected in it, so instead of piling all previous squared gradients that are accumulated it limits the window of these past gradients to a fixed size.

- Pro: There is no decaying and the training does not pause.
- Con: Expensive computational wise

3- Adam:

Adam optimizer stands for Adaptive Moment Estimation. It is a technique that reduces the time taken to train a model. The Adam optimizer adds the expected value of past gradients, this means it's slow at the moment but catches up speed as it goes, it works with momentums and this leads to faster convergence. The idea behind it is to avoid jumping over the minimum this can be obtained by decreasing the velocity just a bit for a more careful roll. Adam works best with problems that are large in data or parameters and where time is a concern, and we used this optimizer for its good performance.

- Pro: Very fast
- Con: Computationally costly.

Here, the focus is on choosing the classification method, a crucial component of a successful classification system. This section displays the five state-of-the-art CNN-based mask detectors. It's worth mentioning that every model was built with the K-Fold, in which it is an essential part in avoiding overfitting. In the K-Fold validation technique, which divides the data into k-subsets at random, each of the k subsets is used as a test set, while the remaining k- 1 subsets are used for training. Moreover, the number 3 was chosen as the number of folds in each and every model discussed. In addition, evaluation metrics such as Accuracy, Precision, Recall, and F1 score were used and found to ensure the robustness of the results.

In more detail, these models are:

a- Sequential 1:

The model is built sequentially in Keras lib by adding layers to it one by one. By adding layers, we can make a stack of layers which will be interconnected and can be used.

For building the model:

- The model has two Conv2D, the first with 200 kernel/filter of (3x3) followed by Relu layer and MaxPooling layer of (2x2). The second Conv2D with 100 kernel/filter of (3x3) followed by Relu layer and MaxPooling layer of (2x2).
- Then the flatten layer converts the data into one dimension array and the Dropout layer is used to eliminate overfitting. Furthermore, two dense layers were applied, the first with 50 units and Relu activation and the other with two outputs for two categories and softmax activation.

Once the model is created, we can configure the model with categorical_crossentropy loss function since we have two categories and Adam optimizer and accuracy metrics with the “model.compile()” [22].

```
# MODEL 1
kfold01 = KFold(n_splits=3)
for train, test in kfold01.split(data, labels):
    # Define the model architecture
    model=Sequential()

    model.add(Conv2D(200, (3,3), input_shape=data.shape[1:]))
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size=(2,2)))

    model.add(Conv2D(100, (3,3)))
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size=(2,2)))

    model.add(Flatten())
    model.add(Dropout(0.5))

    model.add(Dense(50, activation='relu'))

    model.add(Dense(2, activation='softmax'))

    # Compile the model
    model.compile(
        optimizer='adam',
        loss='binary_crossentropy',
        metrics=['accuracy', tf.keras.metrics.AUC(name = 'auc'), f1_m, precision_m, recall_m]
    )
```

b- Sequential 2:

The model was created using a sequential approach in keras by gradually adding layers to it. By adding layers, we can create a stack of layers that can be used and connected.

These models consist of four Conv2D layers. The first layer is with 64 kernel/filter of (3x3), followed by Relu activation layer and MaxPooling2D layer of (2x2). The second layer with 256 kernel/filter of (3x3) followed by Relu activation layer and MaxPooling2D layer of (2x2). The third layer with 128 kernel/filter of (3x3) followed by Relu activation

layer and Dropout layer for avoiding overfitting. The last layer is Conv2D with 32 kernel/filter of (3x3) followed by Relu activation layer and MaxPooling2D layer of (2x2). Again, Dropout and flatten layer is added to the model, then the model gets three extra Dense layers, one with 100 units and Relu activation and the other with 16 units and Relu activation. The last one has 2 units for two categories and softmax activation which is appropriate when we have binary classification. Once the model is created, we can configure the model with binary_crossentropy loss function and Adam optimizer and accuracy metrics with the “model.compile()” [23].

```
kfold01 = KFold(n_splits=3)
for train, test in kfold01.split(data, labels):
    model = Sequential()
    model.add(Conv2D(64, (3, 3), input_shape=(224, 224, 3)))
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size = (2, 2)))
    model.add(Conv2D(256, (3, 3)))
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size = (2, 2)))
    model.add(Conv2D(128, (3, 3)))
    model.add(Activation('relu'))
    model.add(Dropout(0.25))
    model.add(Conv2D(32, (3, 3)))
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size = (2, 2)))
    model.add(Dropout(0.25))
    model.add(Flatten())
    model.add(Dense(100, activation='relu'))
    model.add(Dense(16, activation='relu'))
    model.add(Dense(len(CATEGORIES)))
    model.add(Activation('softmax'))

    # Compile the model
    model.compile(
        optimizer='adam',
        loss='binary_crossentropy',
        metrics=['accuracy', tf.keras.metrics.AUC(name = 'auc'), f1_m, precision_m, recall_m]
    )
```

c- Sequential 3:

The model was created sequentially in the Keras library by gradually adding layers to it. We can create a stack of layers by adding layers that will be connected and used.

The structure of the code is as follows:

- The model has three Conv2D, all of the three layers with 32 kernel/filter of (3x3) followed by Relu layer and MaxPooling layer of (2x2) as a default. Only the first layer includes input_shape= (224, 224, 3).

- Then flatten layer is used to convert data into a one-dimension array. Furthermore, dense layer is applied with 100 units and Relu activation and again the Dropout layer is added to avoid overfitting.
- Finally, other dense layer with 2 units was put for two categories and softmax activation

Once the model is created, we configure the model with binary_crossentropy loss function since we have two categories and Adam optimizer and accuracy metrics with “model.compile()” [24].

```
kfold = KFold(n_splits=3)
for train, test in kfold.split(data, labels):
    model=Sequential()
    model.add(Conv2D(32, (3,3), activation='relu', input_shape=(224, 224, 3)))
    model.add(MaxPooling2D() )
    model.add(Conv2D(32, (3,3), activation='relu'))
    model.add(MaxPooling2D() )
    model.add(Conv2D(32, (3,3), activation='relu'))
    model.add(MaxPooling2D() )
    model.add(Flatten())
    model.add(Dense(100, activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(2, activation='softmax'))

    # Compile the model
    model.compile(
        optimizer='adam',
        loss='binary_crossentropy',
        metrics=['accuracy', tf.keras.metrics.AUC(name = 'auc'), f1_m, precision_m, recall_m]
    )
```

d- MobileNetV2:

MobileNet-v2 introduced by google in early 2018 is also a Convolutional Neural Network that is 53 layers deep and again trained on the ImageNet database built on the same objective of its former prototype MobileNetV1, it is mostly used for mobile devices and other low-powered computing devices. This can be seized in MobileNetV1 by using Depthwise Separable Convolution the employment of inverted residual structure in MobileNetV2 results in the introduction of a better module. This residual structure, which serves as the foundation for V2's design, has tiny bottleneck layers at its input and output. There are two categories of residual blocks. One is for residual with a stride of 1, and the second is for downsizing with a stride of 2, and with the help of a skip connection both the beginning and the end of the convolutional block are connected. V2 follows an approach that goes from narrow to wide to narrow again, the network is widened using a (1×1) convolution and then squeezes the network to match the number of channels initially. In the narrowing layers the non-linearities are removed. MobileNetV2 achieved

high speed and came to be an effective extracting tool for object detection, V2 came up to be 35% faster while still holding the same accuracy of V1. MobileNetV2 offers a highly efficient mobile-oriented model used for image recognition.

The model based on a pre-trained MobileNetV2 model [25], it loads the MobileNetV2 network with weights="imagenet" in which its pre-trained model is partially for image and set the include_top parameter, which determines whether or not to include the fully linked layer at the network's top, a False value. And input_tensor for shape of the image After base model is done the next is constructing fully connected layer, and creating head model object passing the base model output as first parameter, then AveragePooling2D with 7x7 after that flatten these layers and adding Dense layer with 128 neurons with "relu" activation so relu basically go to activation function for nonlinear cases, the Dropout layer were utilized to avoid overfitting finally Dense layer of 2 units one for mask case and other for not mask and the activation is "softmax".Once we done, we call model function accepting two parameter one for input and the other for output, the input is the basemodel and the output headModel. the base model's layers in a loop and freeze them to prevent their updating during the first training process. Since there are two categories, the model was compiled using the Adam optimizer and the binary Crossentropy_loss function.

```
kfold = KFold(n_splits=3)
for train, test in kfold.split(data, labels):
    baseModel = MobileNetV2(
        weights="imagenet",
        include_top=False,
        input_shape=(224, 224, 3)
    )

    headModel = baseModel.output
    headModel = AveragePooling2D(pool_size=(7, 7))(headModel)
    headModel = Flatten(name="flatten")(headModel)
    headModel = Dense(128, activation="relu")(headModel)
    headModel = Dropout(0.5)(headModel)
    headModel = Dense(2, activation="softmax")(headModel)

    model = Model(inputs=baseModel.input, outputs=headModel)
    for layer in baseModel.layers:
        layer.trainable = False

    # Compile the model
    opt = Adam(learning_rate=INIT_LR, decay=INIT_LR / EPOCHS)
    model.compile(
        loss="binary_crossentropy",
        optimizer=opt,
        metrics=["accuracy", tf.keras.metrics.AUC(name='auc'), f1_m, precision_m, recall_m]
    )
```

e- VGG19:

VGG which stands for Visual Geometry group proposed by Simonyan and Zisserman in 2014 at the university of Oxford is a Convolutional Neural network which contains many variants like VGG 11, 16 and several others.

VGG19 is a Convolutional Neural network variation that is 19 layers deep. These layers consist of 16 convolution layers for feature extraction, 3 fully connected layers for classification, each layer is then followed by 5 MaxPool layers, and contains 1 SoftMax layer. It is pre-trained using the ImageNet database which is made up of 14 million images belonging to approximately 1000 classes. The model has performed fairly well for image classification.

To understand VGG19 we can look at how its architected. It is very popular due to using several 3x3 filters or kernels in each convolutional layer. Additionally, the network receives input in the predetermined size of 224*224 RGB images, this means in RGB that the shape is (224,224,3). Spatial padding is also used to maintain the resolution of the image. As previously noted, max pooling is carried out across each layer in a (2×2) pixel window.

By means of this we can realize that deep CNNs achieves better accuracy by implementing more layers and reducing the computational effort through implementing layers using the 3*3 filter, this reduces the number of parameters and thus the time of computation [26].

The loading of the VGG19 is the initial stage of model building. For the weights="imagenet" which is a pretrained model partially for image and assign False value to the include_top parameter whether to not include the fully-connected layer at the top of the network. And input_shape for shape of the image. In addition, run a loop over all of the base model's layers and freeze them to prevent updates during the initial training. After loading VGG19 the next is constructing fully connected layer first layer is VGG19, flatten these layers and adding Dense layer with 2 neurons with "sigmoid" activation. Finally, the model was compiled using the categorical Crossentropy_loss function and the Adam optimizer.

```

kfold = KFold(n_splits=3)
for train, test in kfold.split(data, labels):
    vgg19 = VGG19(
        weights='imagenet',
        include_top=False,
        input_shape=(224, 224, 3)
    )

    for layer in vgg19.layers:
        layer.trainable = False

    model = Sequential()
    model.add(vgg19)
    model.add(Flatten())
    model.add(Dense(2, activation='softmax'))

    # Compile the model
    model.compile(
        optimizer="adam",
        loss="binary_crossentropy",
        metrics=["accuracy", tf.keras.metrics.AUC(name = 'auc'), f1_m, precision_m, recall_m]
    )

```

4- Class assignment (Classification output):

Each facial input image in this work belongs to one of two classes; the first class is a person wearing a face mask, while the second class is a person not wearing one. As a result, the models in use will anticipate one class for each input image. Humans in real life make a wide range of decisions, but multiple opinions on one decision can help for a better more precise outcome, this improves the quality of the decision and decreases the percentage of error. This same theory is implemented in a classification approach through a technique called as Fusion/Multimodal. As a result, information fuses together inside a multimodal machine learning.

The purpose of employing multimodality is to gather crucial data from various inputs, combining it, and using the resulting feature to address a specific issue. This provides an output with a richer performance when compared to individual modalities. This multimodal analysis can be extremely helpful in fields like medicine where it is crucial to get a 100% accurate output.

8.4 CNN Improvement Strategy:

In this section in an attempt to improve our work, a hyperparameters optimization technique and fusion were utilized, the details will be explained below:

8.4.1 Hyperparameters Optimization:

Selecting the best collection of hyperparameters for a learning algorithm is known as hyperparameter optimization or tuning. Hyperparameter is a parameter in which its value affects how learning takes place.

Finding the most effective kernels is the aim of the CNN model in the convolution layer. The performance of a model is greatly influenced by the size of the kernels, the quantity of kernels, the padding, and the stride, which are all hyperparameters. The performance of the CNN model as a whole is significantly impacted by the Dropout value and Dense rate in the Dense and Dropout layers. To determine the best hyperparameters, we utilized approaches for hyperparameter optimization. There are a few standard methods for hyperparameter optimization, like Grid search. It works by trying every possible combination of the model's parameters. This suggests that the entire search will be time-consuming and costly in terms of computation. Random search is a different tactic that was employed in this study. As its name implies, random search operates by testing a variety of randomly chosen combinations. This method requires less processing time to produce the results because it tests fewer models. An advanced method called a genetic algorithm makes use of natural processes like mutation, evolution, reproduction, etc. These algorithms use complex arithmetic calculations, but they follow rules of nature. Simple mathematical equations can be used to mimic how an organism grows in its surroundings and interacts with other living things.

8.4.2 Multimodal Strategy:

There are more requests for deployable and pertinent decision-support tools as a result of machine learning's rising popularity and the implementations that have followed. As a development on the single model, Multimodality was introduced. Current conducted studies use multimodal input data. Multimodality can be defined as the ability to monitor a system using various input. Based on this, there are three widely used techniques for multimodal machine learning:

- Early Fusion
- Intermediate/Joint Fusion
- Late/Decision Fusion

Fusion: The process of joining information from two or more modalities to perform a wider prediction.

1- Early Fusion:

A common method of combining various data before completing the analysis is called "data level fusion." This process is known as input level fusion. For early fusion

techniques, there are two viable methods. Early fusion procedures can be applied in two different ways. The first strategy involves merging data while eliminating correlation between two sensors. The second strategy involves merging data in its lower dimensional common space.

Early fusion is applied on raw or pre-processed data, also data features are extracted before the process of fusion. These extracted features are then combined into a single representation. This is also considered to be one of the challenges of early data fusion. Because all of the data is merged together from the beginning, early fusion gives true representation of multimedia.

Before entering the data into a unified model, early fusion involves combining the data at the input level by concatenating extracted or original features. There are several ways to combine data, however concatenation or pooling are typically used in early fusion.

- Exploits dependencies between features
- Can end up with very high dimension
- Challenging to use if features have different granularities

Some disadvantages for early fusion are that big amounts of data are removed from the modalities and that early fusion requires the synchronization of timestamps.

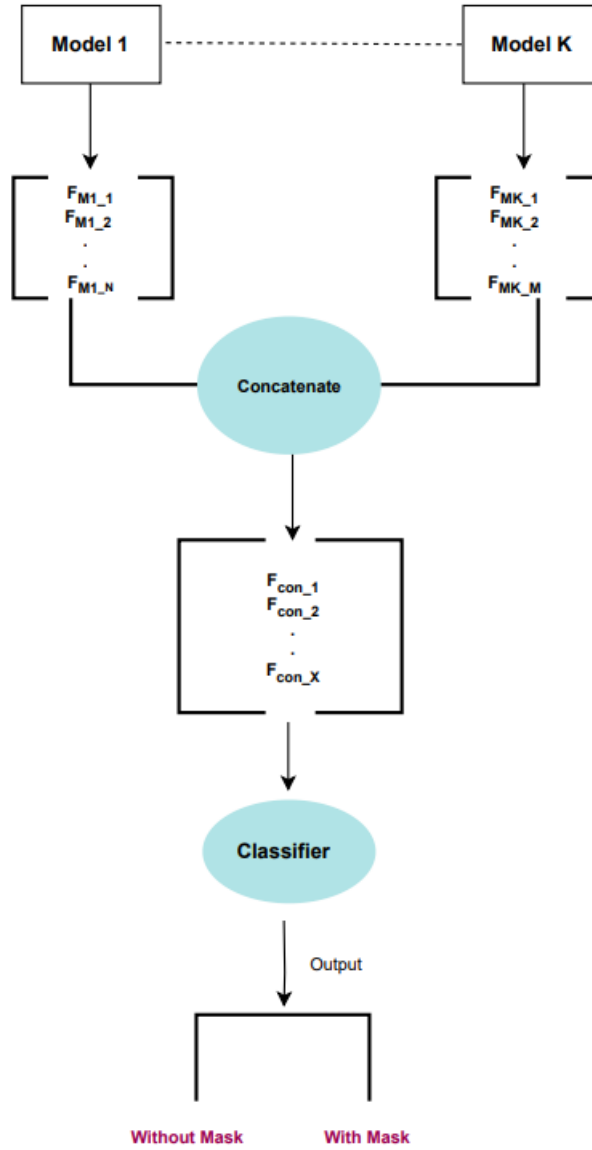


Figure 0.2: Early Fusion Method

Where:

K is the number of used models

N is the dimensionality which represent the output of Selected layer of Model 1

M is the dimensionality which represent the output of Selected layer of Model K

X is the number of features of the concatenating the two selected layers

2- Late Fusion:

Late fusion also starts with feature extraction but the features are combined into a multimodal representation and the Model architecture in late fusion aggregates predictions at the decision level, and for that reason is sometimes called decision fusion. In late fusion, usually, a number of algorithmic models are trained (typically one per data type). Late fusion thus gives less errors because they are dealt with independently.

Large numbers of models must be trained for late fusion (depending on the number of data modalities you are integrating). The inability to update the cost function of the model for text data based on the imaging data, as the information from the modalities never exists in the same model, as well as the expense of training not only multiple models but also choosing an appropriate weighting scheme, are drawbacks to late fusion.

- Train a unimodal predictor and a multimodal fusion one
- Require multiple training stage
- Does not model low level interactions between modalities
- Fusion mechanism can be voting, weighted sum or ML approach

A disadvantage of late fusion is its high computational cost due to the separation of every modality in the learning stage.

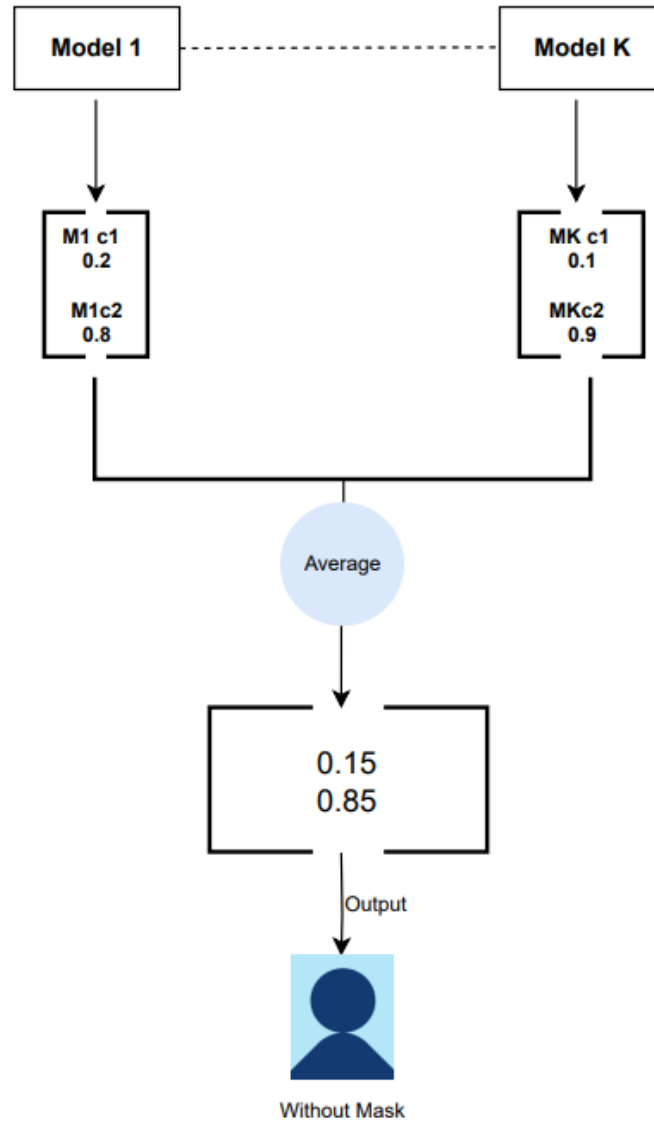


Figure 0.3: Late Fusion Method

Where:

K: Number of used Models

M1c1: represent the membership of the sample to the Mask class

M1c2: represent the membership of the sample to the Without Mask class

MKc1: represent the membership of the sample to the Mask class

MKc2: represent the membership of the sample to the Without Mask class

3- Intermediate Fusion:

The widely used deep neural network serves as the foundation for the design of intermediate fusion. This approach is the most adaptable since it permits data fusion at various model-training stages. Performance is significantly improved using multimodal data fusion using neural networks.

Multiple layers are used in intermediate fusion to transform input data into a higher level of representation (features). Each layer operates on linear and nonlinear functions that change the scale, skew, and swing of the input data to produce a new representation of the original input data. In deep learning multimodal settings, intermediate fusion is the fusion of different modalities into one single hidden layer.

Different types of layers, such as 2D convolution, 3D convolution, and completely linked, can be used to learn features. A fusion layer or shared representation layer is the layer where the fusion of various modality features occurs. A disadvantage of intermediate fusion is the fact that it requires high expertise and it also requires a lot of time.

Intermediate fusion uses encoding methods such as Bag of words, Vlad, and fisher vector. Encoding means compressing information to get an output depending on the type of what is being encoded and here each one is explained individually:

- Bag of words:

The term bag-of-words, or simply "BoW," refers to a technique for extracting features from text for use in modeling. A bag-of-words is a textual illustration that shows in what occurrence words appear in a manuscript and is referred to as a "bag" of words because any details regarding the arrangement or structure of the words in the document are ignored.

This method is really straightforward and flexible, and it may be applied in a variety of ways to extract features from documents.

- Vlad:

VLAD stands for Vector of Linearly Aggregated Descriptors. VLAD is a possible solution to the very large-scale issue of image search. The goal of VLAD is to represent a single image by a predetermined number of feature vectors composed of all feature descriptors collected from the image. This representation is suggested as a solution to the quantization error issue with the Bag-of-Words representation

- Fisher vector:

FV serves as a global image descriptor in image classification and is essentially an image representation created by pooling local image features and it allows the representation of images to be more flexible than others. FV also has the benefit of being computed with considerably smaller vocabularies, which results in lower computing costs.

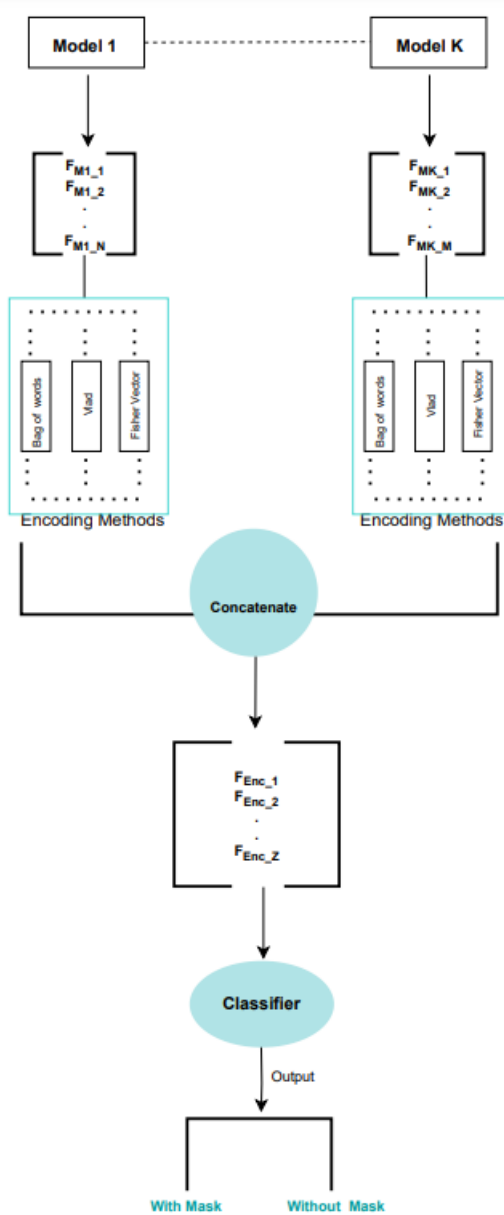


Figure 0.4: Intermediate Fusion Method

Where:

N is the dimensionality which represents the output of Selected layer of Model1

M is the dimensionality which represents the output of Selected layer of Model k

Z is the number of features of the concatenating encoding feature

K is number of used models

Also, dashed lines refer to the possibility of using one of the mentioned methods.

8.5 Developed Mask Detection System

Based on experimental investigation the developed model consists of VGG19 and MobileNetV2 and late fusion, where the parameters of mentioned models were tuned using random search. For the majority of concepts, late fusion schemes tend to perform better than early fusion.

Despite the fact that intermediate fusion can be considered better than early and late fusion due to its flexibility, late fusion has better potential and achieves a better performance overall because the errors from multiple models are dealt with independently thus giving a low error rate.

In this study a multimodal system was investigated after finding the optimal neural network architecture, with an inclusion of VGG19 and MobileNetV2. Plus, late fusion was used experimentally. The experiment done with the late fusion version one, took use of the two pre-trained models VGG19 and MobileNetV2 from the previous experiment. Then the average of the output between the two model's is taken. The main architecture of the developed systems using late fusion is represented in the following diagram;

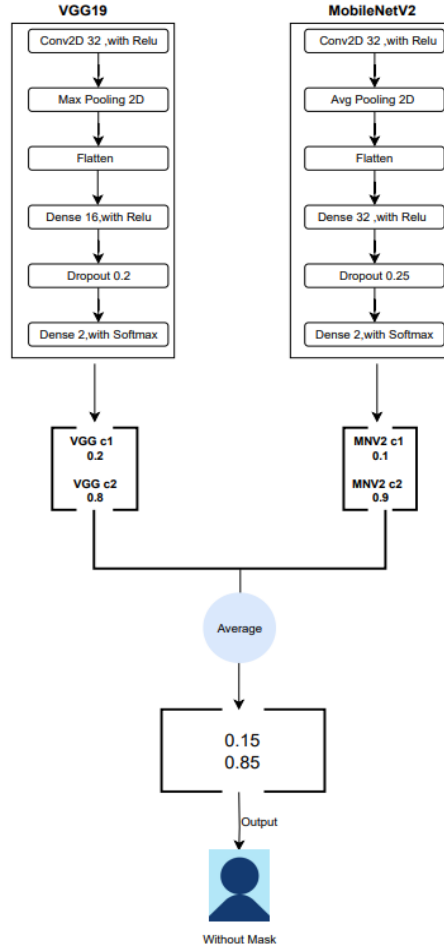


Figure 0.5: Late Fusion Method

Where:

VGG c1: Represents the membership of the sample to the Mask Class.

VGG c2: Represents the membership of the sample to the Without Mask Class.

MNV2 c1: Represents the membership of the sample to the Mask Class.

MNV2 c2: Represents the membership of the sample to the Without Mask Class.

As for the experiment using version two of late fusion, VGG19 included these factors; a Conv2D layer with a (3,3) kernel size and a 32-size filter, as well as MaxPooling2D, Flatten, and Dense at a rate of 16. Again, Conv2D layer was added for MobileNetV2 with (1,1) kernel size and a filter size of 32, as well as AveragePooling2D, Flatten, and Dense with rate 32. Considering these are the original layers, we eliminated the last

dense layer from both the VGG19 and the MobileNetV2, then added a concatenated layer following a dense layer with rate 2 as an output layer.

The main architecture of the developed systems using version two of late fusion is represented in the following diagram:

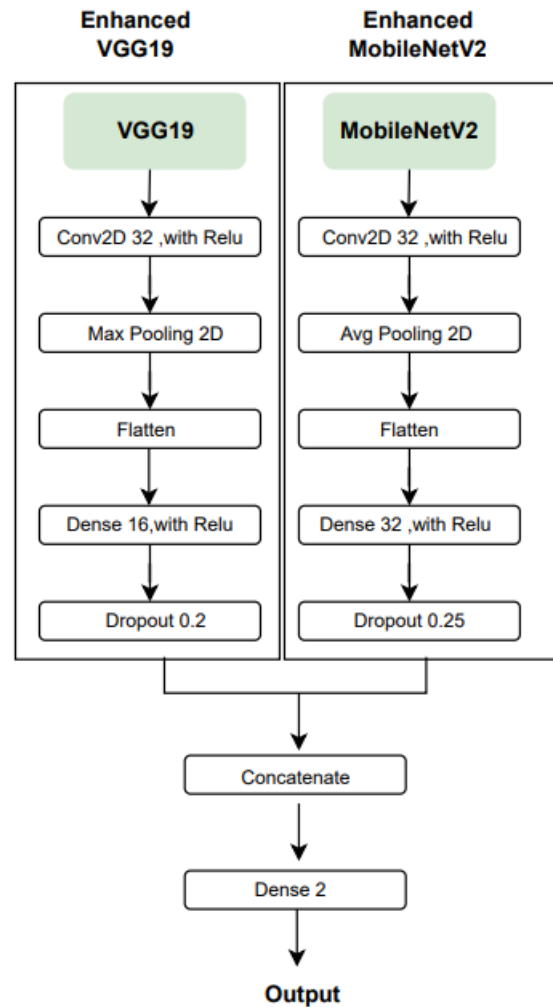


Figure 0.6: late fusion after modifying on layers

CHAPTER 9

Implementation

In this project the developed system was implemented using python, the detail of implementation and the used libraries are summarized below:

9.1 Libraries:

A library is a collection of modules that gives a Python application more functionality. A Python script can import libraries to use their functions, classes, and other objects. The following list of python Programming libraries used in this project:

1- Sklearn library:

Also known as Scikit-learn, is a free and open-source Python machine learning library. It offers a variety of tools for projects including model selection, regression, clustering, dimensionality reduction, and classification. It provides a consistent user interface for interacting with different machine learning models. Additionally, there are various utility methods for preprocessing, cross-validation, and scoring in the library.

Some of the main features of scikit-learn include:

- a variety of preprocessing and feature engineering tools
- Tools for evaluating and choosing models
- Scaling features and missing data processing functions

2- Numpy:

NumPy is a Python library for working with big, multi-dimensional numerical arrays and matrices. It offers methods for interacting with arrays and performing mathematical operations on them. It is a crucial foundation library for scientific computing with Python.

Some of the main features of NumPy include:

- an efficient data structure for storing and processing huge arrays of homogenous data.

- array reading and writing operations to and from files.

NumPy is a vital Python library for scientific computing that is widely used in machine learning, data analysis, and scientific computing in general.

3- Tensorflow:

TensorFlow is a free and open-source machine learning and artificial intelligence software library. It was built by Google and is extensively utilized in many different applications, including as computational biology, image and video recognition, and natural language processing.

- There are several pre-defined models and layers offered for machine learning models.
- Tools for preparing data, building models, and evaluating them
- Distributed training support across numerous devices and servers

4- Keras:

A free and open-source Python library for developing and training machine learning models. It is built on top of other well-known machine learning libraries like TensorFlow and Theano and is intended to be user-friendly, extendable, and modular. The ability of Keras to define and train neural network models is one of its key features.

5- Opencv:

A collection of programming functions with a focus on real-time computer vision is called OpenCV. It was created by Intel and is used for a variety of tasks, including as object and facial identification.

It is important to note that a crucial component of any project is evaluating the performance of the developed model(s). The performance of models is frequently assessed using accuracy, but this is insufficient to give the model a fair evaluation. The used evaluation Matrixes are summarized below.

9.2 Evaluation Matrix

This section will discuss various evaluation metrics that can be used to track and gauge a model's performance.

1- Accuracy:

It typically serves as the primary metric for model evaluation counts the proportion of accurate predictions over all predictions:

$$Accuracy = \frac{TN + TP}{TN + FP + TP + FN}$$

Accuracy Formula

2- Precision:

How many accurate positive predictions are made is a measure of precision (true positives).

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

Precision Formula

3- Recall:

Recall is a metric for how many out of all the positive cases in the data that the classifier correctly predicted.

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

Recall Formula

4- F1-Score:

F1-Score is a measure combining both precision and recall.

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

F1-score Formula

Where:

TN: is the True Negative.

TP: is the True Positive.

FP: is the False Positive.

FN: is the False Negative.

5- The Area Under the Curve (AUC):

AUC is a metric for classifiers' propensity to distinguish between different classes. The greater the AUC, the better the model does at separating the positive and negative classes. If the classifier can successfully discriminate between all Positive and Negative class points, its AUC value is 1. However, if the AUC had been 0, the classifier would have predicted all Positives as Negatives and all Negatives as Positives

$$AUC = \frac{\sum Rank(+) - |+| \times (|+| + 1)/2}{|+| + |-|}$$

Area Under Curve Formula

Where:

$\sum Rank(+)$ is the sum of all examples that have been classified positively.

$|+|$ is the number of positive examples in the dataset

$|-|$ is the number of negative examples in the dataset

9.3 Experiments and Results

In this section we will discuss in detail the conducted experiments and its results:

- Experiment 1:

To begin with, the basic artificial neural network models that were tested are MobileNetV2, VGG19, and three Sequential Models, along with different architectures; these were used in the experiments within several datasets (the four mentioned datasets) to detect the performance of each model. This will help to best select the most suitable one. After these training sessions, the datasets are combined to create the all-in-one dataset and again train all the models with it. The model's precision was assessed using 3-fold cross-validation technique. The dataset was split into three different subsets for this technique. The remaining three groups were used for training, while one subset was used for testing. The test dataset was altered throughout each training phase, resulting in the execution of 3 distinct training procedures on all datasets.

Table 19 displays the result of training all the models throughout all of the mentioned datasets.

Table 0.1: The first experiments results

| | dataset | Accuracy | AUC | F1 | Loss | Precision | Recall |
|--------------|------------|----------|-------|-------|-------|-----------|--------|
| Sequential 1 | dataset1 | 0,958 | 0,977 | 0,958 | 0,148 | 0,958 | 0,958 |
| | dataset2 | 0,971 | 0,986 | 0,971 | 0,104 | 0,971 | 0,971 |
| | dataset3 | 0,855 | 0,902 | 0,856 | 0,655 | 0,856 | 0,856 |
| | dataset4 | 0,929 | 0,958 | 0,929 | 0,288 | 0,929 | 0,929 |
| | ALL-IN-ONE | 0,859 | 0,915 | 0,859 | 1,24 | 0,859 | 0,859 |
| Sequential 2 | dataset1 | 0,85 | 0,898 | 0,851 | 0,476 | 0,851 | 0,851 |
| | dataset2 | 0,987 | 0,994 | 0,987 | 0,047 | 0,987 | 0,987 |
| | dataset3 | 0,858 | 0,745 | 0,859 | 0,53 | 0,859 | 0,859 |
| | dataset4 | 0,926 | 0,957 | 0,926 | 0,275 | 0,926 | 0,926 |
| | ALL-IN-ONE | 0,93 | 0,967 | 0,93 | 0,233 | 0,93 | 0,93 |
| Sequential 3 | dataset1 | 0,933 | 0,964 | 0,933 | 0,277 | 0,933 | 0,933 |
| | dataset2 | 0,979 | 0,99 | 0,979 | 0,072 | 0,979 | 0,979 |
| | dataset3 | 0,854 | 0,906 | 0,855 | 0,449 | 0,855 | 0,855 |
| | dataset4 | 0,943 | 0,969 | 0,943 | 0,2 | 0,943 | 0,943 |
| | ALL-IN-ONE | 0,956 | 0,936 | 0,935 | 0,36 | 0,936 | 0,936 |
| MobileNetV2 | dataset1 | 0,939 | 0,979 | 0,939 | 0,147 | 0,939 | 0,939 |
| | dataset2 | 0,992 | 0,998 | 0,992 | 0,02 | 0,992 | 0,992 |
| | dataset3 | 0,933 | 0,974 | 0,934 | 0,167 | 0,934 | 0,934 |
| | dataset4 | 0,97 | 0,993 | 0,97 | 0,078 | 0,97 | 0,97 |
| | ALL-IN-ONE | 0,917 | 0,935 | 0,917 | 0,505 | 0,917 | 0,917 |
| VGG19 | dataset1 | 0,91 | 0,944 | 0,911 | 0,273 | 0,911 | 0,911 |
| | dataset2 | 0,997 | 0,997 | 0,989 | 0,026 | 0,989 | 0,989 |
| | dataset3 | 0,909 | 0,953 | 0,91 | 0,248 | 0,91 | 0,91 |
| | dataset4 | 0,968 | 0,99 | 0,968 | 0,088 | 0,968 | 0,968 |
| | ALL-IN-ONE | 0,975 | 0,963 | 0,976 | 1,744 | 0,976 | 0,976 |

The results can be summarized as follows, the accuracy of training all-in-one dataset on all the model was pretty high with an exception in Sequential 1 model due to the lack of

layers in this model. This indicates that the dataset is good enough for mask detection and is giving good results in almost all of the models, but the Sequential model did not give accurate results even when the data was balanced and the AUC value of all Sequential models was near to 1, this shows that the models are more effective at distinguishing between the positive and negative classes, this also suggests that the models are overfitting to test set. Because the dataset used includes pictures with no background i.e., sky, trees, other objects, etc. in return it is giving extremely accurate results because it is not having a hard time assessing the face, yet this is not the case during real time streaming where the video includes a background that has several items other than the face that is being evaluated. Consequentially the sequential model is giving low to weak performance. So, the comparison is now between MobileNetV2 and VGG19 models because they have high accuracy with 0.975% for VGG19 and 0.917% for MobileNetV2. In addition, they both performed well in real time trials.

- Experiment 2:

As for Experiment2, several datasets were utilized to assess the performance of various basic artificial neural network models, including MobileNetV2, VGG19, and three sequential models with distinct architectures. The goal was to determine which model performed the best. Selecting the ideal number of hyperparameters is one of the key elements that influence how well a CNN model performs. Overall, to do so, hyperparameter optimization or tuning techniques in machine learning were used. The used tuning techniques were the Kernel size, filter size, Dropout value, and Dense rate. In this experiment, the values of the filters used are 32, 64, 96, and 128. As for the kernel size, the following values were applied (1,1), (3,3), (5,5), (7,7). For the Dropout value (0,2), (0,3), (0,5), and (0,25), and lastly the Dense rate values used were 16,32,64,128 and 256.

The results can be summarized as follows, the improvement of the results was noticeable with MobileNetV2 and VGG19, unlike the sequential models where there was no

significant increase in performance. As an outcome of this observation, MobileNetV2 and VGG19 can be considered as the best two models.

Small adjustments such as the convolutional layer and Relu activation were made on the structure of MobileNetV2 and VGG19. These changes were made to give the ability to edit the kernel size and the filter size of the previously mentioned models.

Accuracy, Precision, Recall, and F1 score were employed in this experiment to guarantee the reliability of the findings. The accompanying table shows that all evaluation metrics for each model were quite high

MobileNetN2:

```
headModel = Conv2D(kernel_size=kernel_size, filters=filters)(headModel)
headModel = Activation('relu')(headModel)
```

Vgg19:

```
model.add(Conv2D(kernel_size=kernel_size, filters=filters))
model.add(Activation('relu'))
```

Accuracy, Precision, Recall, and F1 score were employed in this experiment to guarantee the reliability of the findings. The accompanying table shows that all evaluation metrics for each model were quite high.

Table 0.2: Experiment 2 Results

| | dataset | accuracy | f1 | precision | recall | Kernel size | Filters | Dropout | Dense rate |
|--------|----------|----------|-------|-----------|--------|-------------|---------|---------|------------|
| S e | dataset1 | 0.958 | 0.958 | 0.958 | 0.958 | 3,3 | 32 | 0,3 | 128 |
| | dataset2 | 0.977 | 0.977 | 0.977 | 0.976 | 3,3 | 32 | 0,2 | 128 |

| | | | | | | | | | |
|---|----------|-------|-------|-------|-------|-----|-----|------|-----|
| q u e n t i a l 1 | dataset3 | 0.924 | 0.923 | 0.923 | 0.925 | 3,3 | 32 | 0,5 | 256 |
| | dataset4 | 0.911 | 0.910 | 0.919 | 0.911 | 1,1 | 96 | 0,2 | 32 |
| S e q u e n t i a l 2 | dataset1 | 0.958 | 0.958 | 0.958 | 0.958 | 3,3 | 128 | 0,2 | 64 |
| | dataset2 | 0.982 | 0.982 | 0.983 | 0.983 | 3,3 | 64 | 0,5 | 128 |
| | dataset3 | 0.926 | 0.926 | 0.926 | 0.926 | 1,1 | 128 | 0,25 | 128 |
| | dataset4 | 0.932 | 0.932 | 0.932 | 0.932 | 3,3 | 64 | 0,25 | 32 |
| S e q u e n t i a l 3 | dataset1 | 0.980 | 0.980 | 0.981 | 0.980 | 3,3 | 32 | 0,25 | 256 |
| | dataset2 | 0.977 | 0.977 | 0.977 | 0.977 | 5,5 | 32 | 0,2 | 64 |
| | dataset3 | 0.944 | 0.944 | 0.944 | 0.944 | 3,3 | 64 | 0,25 | 128 |
| | dataset4 | 0.926 | 0.926 | 0.927 | 0.926 | 5,5 | 64 | 0,2 | 64 |

| | | | | | | | | | |
|-------------|----------|-------|-------|-------|-------|-----|-----|------|-----|
| MobileNetV2 | dataset1 | 0.983 | 0.983 | 0.983 | 0.983 | 3,3 | 64 | 0,25 | 256 |
| | dataset2 | 0.994 | 0.994 | 0.994 | 0.994 | 1,1 | 128 | 0,3 | 64 |
| | dataset3 | 0.976 | 0.976 | 0.976 | 0.977 | 3,3 | 96 | 0,2 | 256 |
| | dataset4 | 0.964 | 0.964 | 0.966 | 0.964 | 3,3 | 32 | 0,2 | 128 |
| VGG19 | dataset1 | 0.984 | 0.984 | 0.985 | 0.984 | 3,3 | 128 | 0,25 | 16 |
| | dataset2 | 0.996 | 0.996 | 0.996 | 0.996 | 3,3 | 128 | 0,25 | 64 |
| | dataset3 | 0.993 | 0.993 | 0.994 | 0.993 | 3,3 | 64 | 0,25 | 32 |
| | dataset4 | 0.989 | 0.989 | 0.989 | 0.989 | 1,1 | 96 | 0,5 | 64 |

- Experiment 3:

Experiment 3 was conducted based on the results of previous experiments results where the results MobileNetV2 and VGG19 were significantly enhanced after the application of hyperparameters optimization. As discussed, the VGG19 and MobileNetV2 artificial neural networks were selected and each model was trained on the all-in-one dataset. The RandomizedSearchCV hyperparameter optimization method from the Scikit-Learn library was used as the training approach. Cross-validated search over parameter settings is utilized in this approach to optimize the estimator's parameter values [27].

The estimator, param distributions, cv, and n_iter parameters that were passed into RandomizedSearchCV are described below:

- param_distributions:

which will get the inputs as a dictionary, are shown below:

Table 0.3: RandomizedSearchCV Parameters

| | | | | |
|-------------|-------|-------|-------|-------|
| Kernel size | (1,1) | (3,3) | (5,5) | (7,7) |
| Filters | 32 | 64 | 96 | 128 |
| Dense rate | 128 | 64 | 32 | 16 |
| Dropout | 0.2 | 0.25 | 0.3 | 0.5 |

- estimator:

The estimator that is being used is called KerasClassifier. A custom wrapper class from Keras that integrates the Scikit-learn classifier API with Keras parametric models. All previously set parameters will be passed by the estimator. The KerasClassifier will receive the build function of the model and batch size which is 16.

- cv:

Indicates the cross-validation folds number, in this experiment 3 folds were used.

- n_itr:

Refers to a number of parameter settings that are sampled.

As a clarification for the previous experiments, a different technique was used to apply the cross-validation without parameters tuning, and the results for VGG19 and MobileNetV2 models is as shown:

Table 0.4: Experiment without parameters tuning

| model | accuracy | f1 | precision | recall |
|-------------|----------|-------|-----------|--------|
| MobileNetV2 | 0.917 | 0.917 | 0.917 | 0.917 |
| VGG19 | 0.975 | 0.976 | 0.976 | 0.976 |

On the other hand, in this experiment, we obtained the results displayed in the table below after training the two models using identical parameters and datasets.

Table 0.5: Experiment 3 with parameters tuning

| Model | accuracy | f1 | precision | recall |
|-------------|----------|--------|-----------|--------|
| VGG19 | 0.9947 | 0.9947 | 0.9948 | 0.9947 |
| MobileNetV2 | 0.9925 | 0.9925 | 0.9926 | 0.9925 |

The table shown below shows the best parameters for each model

Table 0.6: Best parameters for each model

| model | Kernel size | Filters | Dropout | Dense rate |
|-------------|-------------|---------|---------|------------|
| VGG19 | 3,3 | 32 | 0,2 | 16 |
| MobileNetV2 | 1,1 | 32 | 0,25 | 32 |

The test results demonstrate that VGG19 obtained the highest scores across all metrics. We can notice the significant difference between the two experiments by comparing the new results in table 3 with the outcomes of the prior experiment in table 2. The following table shows the improvement percentage for all metrics after using the parameters tuning technique:

Table 0.7: Improvement percentage in Experiment 3

| Model | accuracy | f1 | precision | recall |
|-------------|----------|--------|-----------|--------|
| VGG19 | +1.97% | +1.87% | +1.88% | +1.87% |
| MobileNetV2 | +7.55% | +7.55% | +7.56% | +7.55% |

The real-time test was carried out as well, the OpenCV library was used to capture the frame and perform the necessary input processing. The DNN caffe model was used to identify the position of the faces and return that position to the model that we had previously built.

Real-time Test:



Figure 0.1: Experiment 3, Sample A, Sample B, Sample C

Based on the real-time test we can say that the accuracy of the two models is extremely close after real-time testing. But in situations where there are several faces, particularly in images with three faces, VGG19 outperforms MobileNetV2.

- Experiment 4:

Last but not least, experiment 4 was done to build a multi-classifier system, and it was divided into three stages using the fusion techniques (late fusion)

For the **first edition**, based on previous experiments, a modified version of MobileNetV2 and VGG19 were used for this stage. This edition was carried out with the late fusion by taking the output of each of the mentioned models and taking the average of them as the eventual output.

The trained models were used to build a multi-classifier system in this edition. Using the same input for both models, the system will then determine the average output for both

models. Following are the results of the multi-modality system, which was achieved by taking the average output for both models as discussed before to be done in edition one.

Table 0.8: Experiment 4 - Stage 1 late fusion Results

| | |
|---------------|--------------------|
| Test accuracy | 0.9962862094577866 |
| F1-score | 0.996274545432744 |
| Precision | 0.996403682157097 |
| Recall | 0.9961553953565232 |

The following validation samples were used during the real-time test of the combined model.

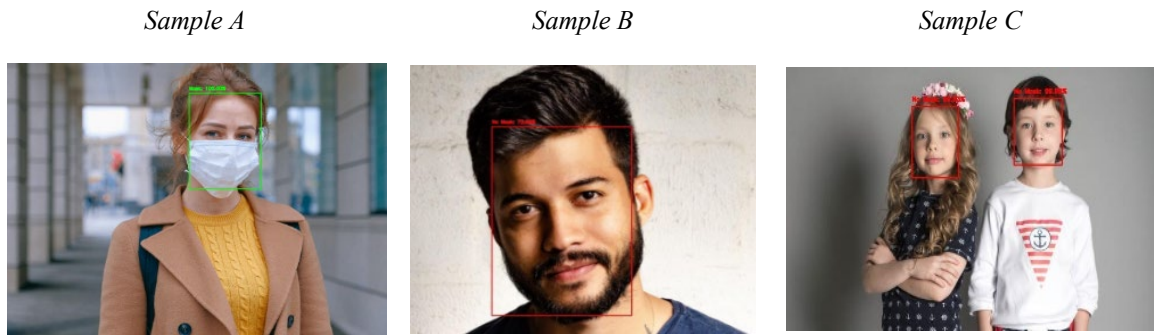


Figure 0.2: Experiment 4, Stage 2 (Sample A, Sample B, Sample C)

Sample A is classified by the system as 100% a mask. Sample B was also classified as a mask but with an accuracy of 72.9 %. In sample C, the first person was detected without a mask with an accuracy of 99.8%, while the second person was specified as without a mask but with 99.9% accuracy.

In the second edition of this experiment, VGG19 and MobileNetV2 were used once more, but the two models were built and trained on a subset of the all-in-one dataset, using only 4660 images alongside the late fusion technique. The subset was used to train the model in a shorter time. The subset was gathered from the first, second and third datasets. The table below shows the test result for the built system:

Table 0.9: Results of the concatenated classifier

| | |
|---------------|--------------------|
| Test accuracy | 0.9957081545064378 |
| F1-score | 0.9957081545064378 |
| Precision | 0.9957081545064378 |
| Recall | 0.9957081545064378 |

Real-time Test:

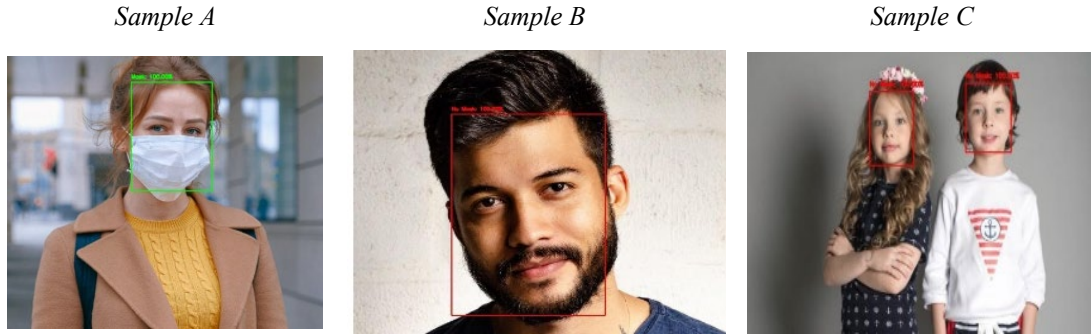


Figure 0.3: Experiment 4, Stage 2 (Sample A, Sample B, Sample C)

Related to sample A, both systems detected the mask with an accuracy of 100%. However, in sample B, the previously stated system detected the face without a mask but with an accuracy rate of 72.9% while the new system detected the face without a mask but with 100% accuracy. In the last sample, the previous system's accuracy for the left face was 99.8%, and 99.9% accuracy for the right face. The new system's accuracy detection was 100% for both faces.

As a result, from the real-time experiment, the new multi-classifier system improved in detecting the NoMask case.

As for the **last edition** of this experiment, after training the system on the same subset, the entirety of the all-in-one dataset was utilized in this experiment to check the stability of the system on a huge dataset with the use of late fusion. 20% of the dataset was utilized for testing and the remaining 80% was used for training.

The previously built architecture was used again which achieved the following test results:

Table 0.10: Concatenated Classifier - Stage 2 Results

| | |
|---------------|--------------------|
| Test accuracy | 0.9978540772532188 |
| F1-score | 0.9978539883120651 |
| Precision | 0.997861190681496 |
| Recall | 0.9978486849827205 |

Real-time test for the third stage is shown as following:

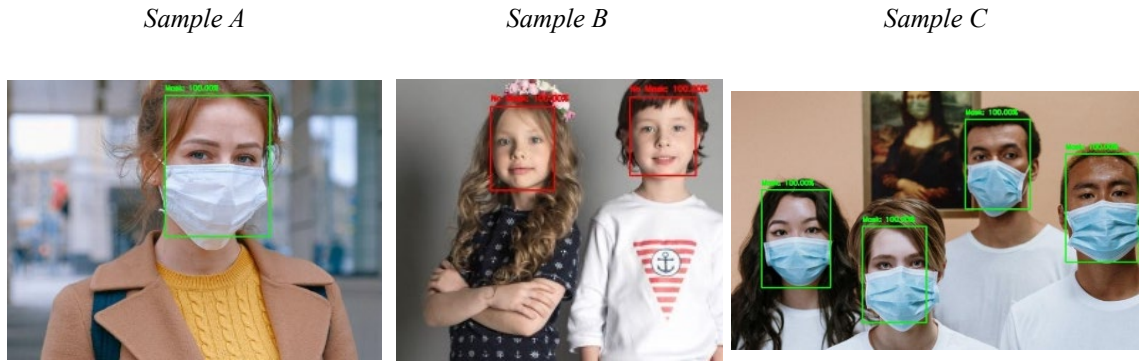


Figure 0.4: Experiment 4, Stage 3 (Sample A, Sample B, Sample C)

The improvements in metrics numbers, as shown in the table, are not significant, and when comparing the system in the last experiment to the prior systems, the differences are hardly perceptible. However, the new system's differences are obviously visible in real-time. The late fusion system's accuracy in some scenarios was 72.1%, which is higher than the last approach in stage 3 of experiment 4, and it functioned particularly well in the NoMask scenario. The results of the latest real-time experiment were greater and more accurate than those of prior real-time trials.

CHAPTER 10

Conclusion

This document proposed a solution upon the spread of the COVID-19 virus that can determine whether a face mask is on or not. Aspects of deep learning are used in this multipurpose mask detection system, which will be used primarily in busy and densely crowded areas. This technology automatically examines the individual's face and combines facial recognition with object detection to detect a person not wearing a face mask through automated real-time detection.

An essential part of any system is initially the database, 4 open-source datasets were chosen to use and then combined into 1 dataset named All-in-one. This is followed by the experiment part which implemented the models on these datasets. The first experiment utilized the cross-validation strategy, and to check its accuracy of it, the model used the following evaluation matrix; F1 score, precision, AUC, and recall. This experiment compared the performances of the models and concluded that VGG19 and MobileNetV2 were the best.

The method used for the second experiment is tuning which was implemented on all the datasets excluding the all-in-one dataset. This experiment observed the improvements made from the tuning of the results.

As for the third experiment, only the all-in-one dataset was utilized alongside MobileNetV2 and VGG19. Then, images previously unseen by the model were used to test the model's performance.

Last but not least, the goal of the fourth experiment was to create a multi-classifier system by modifying MobileNetV2 and VGG19. For both of these models, the system calculates the average output by taking the average of both the models in version one of late fusion. As for late fusion version two we took the last layer to be concatenated and added a dense layer to extract its output and the percentages of the results were enhanced in real-time testing.

In conclusion, it can be shown from these experiments that VGG19 and MobileNetV2 were what produced the best results and improved them over the course of these several studies to construct the most optimal system. In the last stage, when comparing version one and two of late fusion, the differences between them are rarely noticeable, and the improvements in terms of the evaluation matrices are not significant either. However, the new system's differences are seen in real-time and it performed especially well in the NoMask scenario. Results from late fusion version two during real time were superior and more accurate than those from late fusion version one. The final steps from this on will be to publicize the project where it will be put to use in several different places such as hospitals, airports, operating rooms (OR), and many different public places as needed. We would also like to state that researchers can use this study for anything related to object detection for their projects.

Glossary

| Term | Definition |
|-------|--|
| WHO | World Health Organization |
| RMFD | Real-World Masked Face Dataset |
| CNN | Convolutional Neural Network |
| P-Net | Proposal Network |
| R-Net | Refine Network |
| FDDB | Face Detection Data Set and Benchmark |
| AFLW | Annotated Facial Landmarks in the Wild |
| FPS | Frames Per Second |
| SVM | support vector Machine |
| CCTV | Closed-circuit television |
| DNN | Deep Neural Network |
| RGB | Red Green and Blue |

References

- [1] P. Viola and M. Jones, "Rapid Object Detection using a Boosted Cascade of Simple," 2001.
- [2] Y. Freund and R. Schapire, "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting," 1996.
- [3] K. Zhang, Z. Zhang, Z. Li and Y. Qiao, "Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks".2016.
- [4] S. Kalaimani, M. Palangappa and S. Bhuvan, "Face Mask Detection by using Optimistic Convolutional Neural Network," IEEE, 2021.
- [5] P. Nagrath, R. Jain, A. Madan, R. Arora, P. Kataria and J. Hemanth, "Corrigendum to "SSDMNV2: A real time DNN-based face mask detection system using single shot multibox detector and MobileNetV2," 2021.
- [6] S. A. Sanjaya and S. A. Rakhmawan, "Face Mask Detection Using MobileNetV2 in The Era of COVID-19 Pandemic," IEEE, 2021.
- [7] Y. X. G. L. Wen L, "A new automatic machine learning based hyperparameter, Measurement and Control," pp. 53(7-8):1088-1098, 2020.
- [8] X.-Y. C. H. Z. L.-D. X. H. L. S.-H. D. Jia Wu, "Hyperparameter Optimization for Machine Learning Models Based on Bayesian, Journal of Electronic Science and Technology," 2019. [Online]. Available: <https://doi.org/10.11989/JEST.1674-862X.80904120>.
- [9] R. Murugan, K. Choudhury, M. A. Laskar and R. H. Laskar, "A comparative analysis between late fusion of features approach," 2021. [Online]. Available: <https://doi.org/10.1002/cpe.6371>.
- [10] T. Welsh, M. Ashikhmin and K. Mueller, "Transferring Color to Greyscale Images," 2002.
- [11] S. Sikchi, "COMPARATIVE STUDY OF DIFFERENT DESKTOP OPERATING SYSTEMS," 2021. [Online]. Available: <https://sikchisagar9330.medium.com/comparative-study-of-different-desktop-operating-systems-1de58212c32d>.
- [12] A. Dearmer, "Firebase vs. MySQL: Battle of the Databases," 2020. [Online]. Available: <https://www.integrate.io/blog/firebase-vs-mysql/>. [Accessed 2022].
- [13] A. Das, "11 Reasons Why Linux Is Better Than Windows," 2021. [Online]. Available: <https://itsfoss.com/linux-better-than-windows/>. [Accessed 2022].
- [14] S. Kumar, "Why Python is Best for AI, ML, and Deep Learning," 2021. [Online]. Available: <https://www.rtinsights.com/why-python-is-best-for-ai-ml-and-deep-learning/>. [Accessed 2022].
- [15] M. Wasserman, "Face/Off: High speed facial tracking using the Viola Jones Method," Jul 2019. [Online]. Available: <https://towardsdatascience.com/face-off-high-speed-facial-tracking-using-the-viola-jones-method-f9e0ba491c9f>. [Accessed 2022].

- [16] S. Susanto, F. A. Putra, R. Analia and I. K. L. N. Suciningtyas, "The Face Mask Detection For Preventing the Spread of COVID-19 at Politeknik Negeri Batam," 2021.
- [17] S. Yadav, "Deep Learning based Safe Social Distancing and Face Mask Detection in Public Areas for COVID-19 Safety Guidelines Adherence," 2020.
- [18] V. KUMAR, "Face mask Detection," Kaggle, 2021. [Online]. Available: <https://www.kaggle.com/datasets/vijaykumar1799/face-mask-detection>. [Accessed 10 09 2022].
- [19] A. JANGRA, "Face Mask Detection ~12K Images Dataset," 2020. [Online]. Available: <https://www.kaggle.com/datasets/ashishjangra27/face-mask-12k-images-dataset>. [Accessed 2022].
- [20] C. Deb, 2020. [Online]. Available: <https://github.com/chandrikadeb7/Face-Mask-Detection>. [Accessed 2022].
- [21] B. Isrinivas, "Face-Mask-Detection," 2020. [Online]. Available: <https://github.com/balajisrinivas/Face-Mask-Detection/tree/master/dataset>. [Accessed 2020].
- [22] "face-mask-detection-keras," Perceptron, 2019. [Online]. Available: <https://github.com/aieml/face-mask-detection-keras>. [Accessed 2022].
- [23] "Real-time Face Mask Detection with OpenCV," [Online]. Available: <https://projectgurukul.org/face-mask-detection/>.
- [24] K. Malik, "FaceMaskDetector," 2020. [Online]. Available: <https://github.com/Karan-Malik/FaceMaskDetector>.
- [25] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," IEEE, 2020.
- [26] S. Teboulbi, M. Seifeddine, M. A. Hajjaji and M. Abdellatif, "Face Mask Classification Based on Deep Learning Framework," 2022.
- [27] S.-l. "RandomizedSearchCV," [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html.