VIDEO IMAGE PROCESSING OF COVID RESTRICTIONS AT CARE HOMES

Swansea University
Prifysgol Abertawe

VINITH KARUPPAGOUNDER SUBRAMANI (2136530)

Faculty of Science and Engineering

Swansea University

This Dissertation is submitted for the degree of Master of Science

Swansea University **15th December 2022**

ACKNOWLEDEMENT

Firstly, I would like to convey my sincere thanks to **SWANSEA UNIVERSITY** for admitting me into their internationally recognized, reputable university and helping me to grow by learning. Secondly, I would like to convey thanks to my supervisor for this project **Dr. Tom Owen** for guiding me throughout this research and helping me in every aspect of the work. The other person that I would like to thank is my second supervisor **Dr. Zhan Li** for his lectures on Machine learning & Deep Learning during the course modules. Also, I would like to thank the members who supported me during this project, **Bethan Thomas** the project coordinator from Singleton Campus, **Simon** from Tech Up & Go and **Sean Jenkins** from the Atic Lab of The University of Wales Trinity Saint David. Last but not least, the support that **my parents** gave me throughout this whole study was immense and I'm sincerely thankful to all of them.

DECLARATIONS/STATEMENTS

This work has not previously been accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

STATEMENT 1

This dissertation is the result of my own independent work/investigation, except where otherwise stated. Other sources are acknowledged by giving explicit references. A bibliography is appended.

STATEMENT 2

I hereby give consent for my dissertation, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

Signature: Vinith Karuppagounder Subramani **Date: 15/12/2022**

ABSTRACT

Covid-19 being the global pandemic created a significant effect on the care homes. People in care homes are old and not in their prime fitness which makes them easy victims of the viral infection. To overcome this scenario, government introduced several preventive measures but monitoring them regularly is not that practical. This being the motivation, a team from Tech Up & Go created a prototype that could monitor the preventive measures that are followed by people in care homes such as face mask, social distancing & facial temperature variation. In this research, we explored a few pre-trained models including the ResNet101 for the image classification where the ResNet101 was used in the prototype and a few object detection techniques including YOLOv5 which was used in the prototype to determine their performance based on a given Custom Face mask dataset for image classification and the WIDER face dataset & COCO dataset for the object detection. Based on the observed results and evaluation, concluded this research with two proposed models which could enhance the detection and classification capabilities of the prototype. They are YOLOv7 for the object detection (Face detection & Social Distancing part) and InceptionV3 for the image classification (Mask Classification) part based on the results evaluated with the evaluation metrics such as Confusion Matrix, Accuracy, Precision, Recall, F1 Score and Parameters training speed for Image classification and Intersection over Union, Mean Average Precision, and inference time for the object detection.

TABLE OF CONTENTS

INTRODUCTION

Coronavirus (Covid-19) being a transmissible disease was caused by a virus called SARS-CoV-2. Most of the people who are infected with the disease will start to show some mild symptoms of breathing difficulty and some may become better in terms of health without having any additional treatment. But when it comes to older people, its completely different. When people are suffering with any mild to moderate ailments, Covid-19 would make it much worse and cause severe difficulty (Sanyaolu et al., 2020). Being one of the fastest transmissible, could infect any one to make them sick and make them die.

The covid virus usually spread from the affected person when he is speaking, coughing, sneezing or breathing. While doing that, there are some debris that come out of their mouth as droplets or small aerosols which is the key reason for their spread. To restrict that transmission, people need to follow the guidelines like closing their mouth while coughing. They also suggested that if people are not feeling well they could self-isolate till they feel better and healthy (Schellack et al., 2020).

The COVID-19 pandemic, being a global disorder which literally had the effect on each and every person in the world. This world pandemic, though affected all the countries, it was first diagnosed from a scourge in Wuhan, China in December 2019. From the moment, government officials were working day and night to restrict and curb its transmission. Attempts to restrict failed, permitting the virus to unfold to different regions of Asia and later worldwide. The World Health Organization (WHO) declared the outbreak a public health hazard of worldwide concern on 30 January 2020 and a deadly disease on 11 March 2020 (Coronavirus Disease (COVID-19) Pandemic, 2022). As of 1st December 2022, the pandemic had brought about greater than 643 million cases and 6.63 million showed deaths, making it certainly considered one of the deadliest in history.

The effects of the pandemic on health, wellness and care structures were devastating, as validated by the mortality, morbidity and healing rates. Evidence shows that older humans and the workforce residing and operating in the care domestic area had been considerably impacted by the COVID-19 outbreak (Kostyál et al.,2021). Despite warnings of the doubtlessly devastating effect of Covid-19 on care houses, the primary wave of the pandemic noticed a tremendous quantity of extra deaths among citizens. The scale of mortality in care houses laid naked long-standing issues with care domestic provision, in addition to shortcomings withinside the reaction to the pandemic (Daly, M. , 2020). In February and March 2020, coronavirus unfold fast through a few care homes. Many citizens had been affected and huge numbers – which include many human beings with dementia died from intense COVID-19. Some of the ways that care houses had been so badly hit are intrinsic to the citizens, the virus and the environment. Measures may be installed in the location to assist control those and we're higher organized in opposition to coronavirus now, however the underlying ways remain.

Reasons, why care homes have been so significantly affected, include prone citizens – human beings in care houses, specifically nursing houses, are mostly in their 80s or 90s. They also are frequently residing with dementia, frailty or underlying fitness conditions, and occasionally all three. These elements make all care domestic citizens at excessive risk of catching coronavirus and turning significantly ill (Comas-Herrera, A et al.,2020). The domestic environment care houses especially in which human beings live, now no longer like a hospital. Residents and teams of workers are used to socialising and doing activities together. Providing private care that the team of workers need to be in near touch with numerous citizens over a day. All of these near interaction helps the virus unfold speedily in the domestic. Capabilities of the virus coronavirus spread in small droplets (while a person with the virus coughs, sneezes, talks or breathes out) and through touch with surfaces, which includes shared residing areas. The virus causes signs and symptoms in older human beings which might be distinctive and so have been less complicated for the care home workforce to miss. In a few citizens and workforce, coronavirus contamination reasons no signs and symptoms at all.

The First and foremost preventive measure introduced by the government is the wearing of a Face Mask. Face masks have to be worn by all care employees and advocated for site visitors in care settings and whilst supplying care in people's very own homes, regardless of whether or not the individual being cared for is thought or suspected to have COVID-19 or not (Tripathi, R., et al.,2020). This is from time to time stated as 'widespread masking' or 'source control' and is a method of stopping any unfold of contamination from the wearer of the mask. All face mask has to be properly suited to cover the nose, mouth and chin, be worn in line with the manufacturer's recommendations now no longer be allowed to cling across the neck at any time, not be touched as soon as positioned on, be worn in line with the risk-assessed interest and be eliminated and disposed of appropriately, with the wearer cleansing their arms earlier than elimination and after disposal (Machida et al.,2020).

The other key preventive measure introduced by the government is Social Distancing. Social distancing additionally referred to as bodily distancing, is the exercise of maintaining distance between yourself and those you do not stay with. It additionally includes keeping off groups, gatherings and crowds in each indoor and outside space (De Vos, 2020). There's plenty that's nevertheless unknown regarding the coronavirus. Still, public health specialists and researchers have determined that restricting close contact among human beings is one of the superior methods to sluggish the unfolding of COVID-19. And, even though human beings can contact against COVID-19 via way of means of touching their noses and mouths with their fingers which have uncovered to the virus through surfaces and objects, this isn't always ideal to be the number one manner the virus spreads. Instead, getting exposed to droplets which might be dispersed into the air while an infected individual coughs, sneezes or talks are ways to be the primary drivers of the virus' unfold (Hsiao et al.,2020). The droplets can settle withinside the noses and mouths of human beings nearby and might also be inhaled into the lungs. This manner of diffusion is what obliges social distancing necessary.

PROJECT IDEA

This project mainly deals with the restrictions imposed by the government on care homes to prevent the Covid-19 virus to spread from one. It was put up by Professor Tom Owen and become recommend through Tech Up & Go a Private Limited Company. The business enterprise already created a prototype device that is automatically capable of discovering the covid mitigation approaches such as face-protecting, social distancing and facial temperature variations in care houses. They created it to make the reaction time a lot quicker when there's any lapse withinside the processes or while the dangers of contamination are being identified. The prototype become constructed on the Linux platform and OpenCV (a famous photo-processing library) using Computer Vision Techniques and Machine Learning Algorithms.

This preliminary prototype created through them can locate humans with and without facial masks through the use of artificial intelligence, Measure social distancing with the use of depth records and detection of facial temperature variation that indicates the hazard of contamination. They created this mission due to the Covid-19 pandemic. This pandemic now no longer simply examined our healthcare structures to their fullest capability but additionally highlighted the truth of higher contamination control, particularly in care houses or any shared lodging with aged humans(Daly ,2020). The foremost want of the care houses during this pandemic is tracking visits which include the outcome of introducing contamination and the demanding situations from dementia of keeping social distancing (Brown et al., 2020). This ends in a large call for a group of workers appointments. Also, the visitors' restrictions, had a terrible effect on the fitness of the individuals (O'Caoimh et al., 2020). To conquer those situations this challenge become initiated.

AIM & MOTIVATION

The prototype already created consists of three main preventive measures to track and detect. They are Face Mask Detection, Social Distancing and Facial Temperature Variation. Our main aim is to analyse Face Mask Detection and Social Distancing part of the prototype, segregate them into parts, and try some other methods to see how it performs with the already implemented techniques that are used in the prototype. Here, in the prototype, we can split the Face Mask Detection into two parts. They are face detection and face mask detection. We are using some techniques to detect faces and then we are applying some Machine learning algorithms to detect masks on those detected faces. Likewise, they are using some Deep Neural network techniques to implement Social Distancing detection. To increase the overall detection capabilities of the Prototype, we can try the enhanced techniques of each part which in turn will increase the overall detection capabilities.

After conducting some literature reviews of scientific papers, we can determine that they are mostly focused on either one of the preventive measures and among them, they are trying to explore one specific model or algorithms performance over some dataset. These papers are mostly oriented toward the methodologies of the implementation of the Mask Detection model rather than giving us an overall idea about the mask detection performance of various models that people are widely used for image classification. This marks the primary motivation to conduct this research, try to compare different Image Classification models for Mask Detection, explore the various Face Detection techniques that come under the head of Object Detection that is widely used and the reasons why they are widely used with some proof can provide us what would be the ideal Object Detection(Face detection & Social Distancing) and Image Classification (Mask Detection) models that we can use to enhance the prototype. Before we start diving deep into the research let's see some of the works that are already done in these areas regarding Object Detection(Face detection & Social Distancing) and Image Classification (Mask Detection) models.

SUBJECT AREA

ARTIFICIAL INTELLIGENCE (AI)

Artificial Intelligence is the science of making machines think like humans and make decisions (Rich, 1983). Even though this looks fairly easy task to do but in reality it isn't. Computers are great at performing under the rules using programming them, which doesn't match the intelligence of human brain decisions in a difficult scenario. In some cases, a task which looks so easy for humans will be a huge and time taking one for machines. For an instance, a person went to a crowded bar in the evening and orders some drinks. Servers usually serve them but the scenario in the bar will make it a bit difficult in reality. People will be hanging out and dancing all around the place and he has to find a way to cross them and serve those drinks to the customer. While doing that he has to process a lot of information in his brain and make decisions in real-time. Artificial Intelligence wasn't there yet but they are on that track with techniques called Machine Learning & Deep Learning (Fig 1). The sole aim of this artificial intelligence is to reduce human involvement and make decisions like humans by processing large quantities of information.

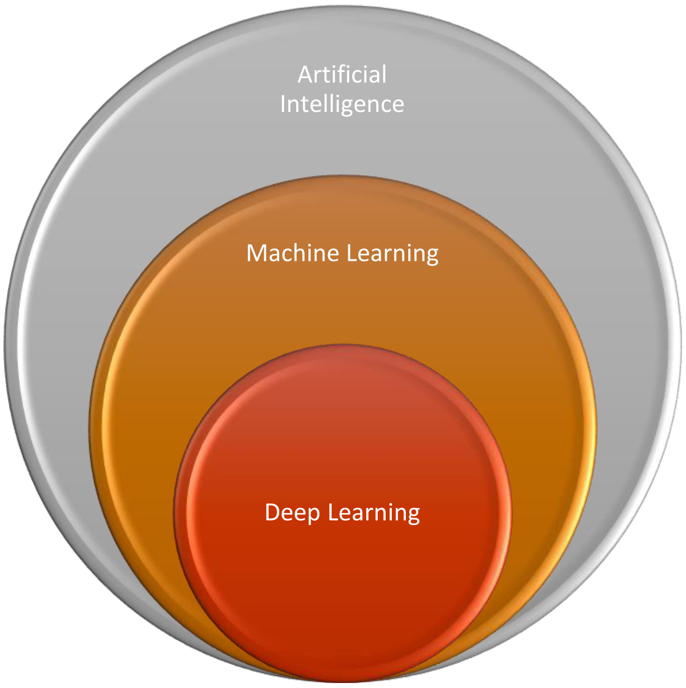


Figure 1 – Artificial Intelligence Classification

MACHINE LEARNING

Machine learning is a subset of Artificial Intelligence where the machines will start learning without being programmed. In simple terms, this can be explained by comparing Programming and Machine Learning (Rich, 1983). In Programming, machines will have rules and it follows the rules, get inputs and produce output. But in Machine learning, the input and out of those inputs are fed into the computer and the computer will start learning and framing rules based on the input and given output and frame the rules accordingly by optimising its learning. This process of machine getting learned and framing rules from inputs and outputs is called Machine Learning. Once the machine learned, we can feed the new inputs and with that learned rules, the machine can give us the output. There are a few types of Machine learning techniques. They are:

Supervised Learning

Supervised Learning is a way of Machine learning where the machines will be fed respective outputs for the given inputs. They are called Labels in technical terms. When we feed those labels the machines can start performing their learning based on the input and labels and start framing their rules. This labelled data provided by the user is the Supervision. The main example of this Supervised Learning is Regression & Classification.

Semi-Supervised Learning

In Semi-Supervised learning, only a partial amount of labelled data will be fed into the system. The Computer will try to find the pattern and label the rest of the data on its own by the identified pattern. Once the input data are fully labelled, then the machine will start learning and frame the rules. Here, being only the partial data are labelled, this way of learning is termed as Semi-Supervised Learning.

Unsupervised Learning

Unsupervised Learning, as the name suggests, here no labels will be provided to the machines. The Computer has to find the pattern on its own without any labels and frame the rules based on those observed patterns. The most widely used Unsupervised Learning technique is Clustering. For instance, the Supermarket store will have a large customer base and they are from different places, ages, sex, race, and more. It’s practically impossible for to find the patterns or groups in those data but for machines, it’s quite easy. By providing those data via Unsupervised Learning techniques such as clustering, the machine will provide us with the pattern or groups based on the product, category etc., which will be very much helpful in making Business Decisions.

Reinforcement Learning

On Supervised and Unsupervised learning, the computers can frame their rules based on labels or patterns and start making predictions or decisions based on the inputs that are fed into the system. The computer can either produce a positive or a negative result. Even if that’s a positive there won’t be any rewards and for negatives, there won’t be any criticism. Reinforcement Learning is a technique where it learns based on the feedback provided to the machine. This method works more over like a trial-and-error method. The basic idea of this technique is that when we are teaching a dog certain behaviours, we will reward it with something to eat and when it did something wrong, we will show them that it made a mistake by some harsh actions. The same principle applies here to reinforcement learning. This method was very useful in cases of high quantity, variation and unforeseeable data. The famous application of this reinforcement learning is Self-Driving Cars.

DEEP LEARNING

Machine learning is about where the machines will start learning without being programmed (Rich, 1983). This doesn’t change their way of behaving and acting with human intelligence. The machines will still think like machines and behave in a way which was not up to the level of human intelligence. For instance, their capability of handling image, video and audio files is not up to the mark of what the human brain is capable of doing. This is where Deep Learning comes on. Deep Learning models are specifically created to mimic human brain function. They are made up of high-level multi-layered neural networks (like neurons in the brain)which are used to transfer the data between the interconnected layers of neurons. Feeding the data in with labels, adding weights and then after one successful forward pass, optimising them with some optimiser to change weights in backpropagation to make it ready for the next forward pass, results in better accuracy than most of the machine learning models. Whenever the dataset is complex and highly varied, deep learning models tend to produce better results than normal machine learning models. There are various types of Deep learning models that are used to minimise human intervention and make predictions or decision-making better. The one which we are going to see and mostly related to this project is Convolutional Neural Network.

Convolutional Neural Network

Convolutional Neural Network (CNN) is a method of Deep Learning mainly used to work on images. The Neural Networks work based on weights and adjust them by using an optimiser. The same technique was used here on CNN but with a slight modification. In CNN, they follow a process called Convolutional where they add weights on each item of an image on a filtration basis. This weight-based filtering was very much useful while scanning a high amount of images and feature detection. Thus making it one of the best techniques to use while handling image data in tasks such as Image Classification and Object Detection. The wide use of analysis on image and video images in recent times has created a specific field which is majorly about the manipulation and handling of images and video images. This field is having one of the highest growth rates in the Industry for the past few years and is called Computer Vision.

COMPUTER VISION

Computer vision is an area of artificial intelligence (AI) that permits computer systems to derive significant data from virtual images, motion pictures and other visible inputs and give responses or make suggestions based on that data (Shapiro & Stockman, 2001). If AI permits computer systems to think, computer vision permits them to see, look at and understand. Computer vision works a lot similar to human imagination and prescient, besides human beings have a head start. The human sight has the benefit of lifetimes of context to educate how to inform items apart, how some distance away there, whether or not they're shifting and whether or not there may be something incorrect in an image. Computer vision trains machines to carry out those functions, however, it has to do it in a lot much less time with cameras, data and algorithms instead of retinas, optic nerves and the visual cortex. Because a machine educated to look into merchandise or watch a manufacturing asset can examine heaps of merchandise or methods a minute, noticing imperceptible defects or issues, it can quickly surpass human capabilities.

LITERATURE REVIEW

While doing the literature review for the project, some good and interesting things have been done in this area. Let’s see some of the most interesting works that are done.

In a paper, the authors try to detect masks based on a transfer learning technique where the Pre-trained models are available on the Hub. Pre-trained models are the state of the art deep learning models which are trained over some large datasets such as the ImageNet Dataset of Google, and COCO Dataset etc., Here, the authors used a pre-trained model called InceptionV3 over the Simulated Masked Face dataset (Jignesh Chowdary et al., 2020). While using that, they followed a technique called Image Augmentation by which we can alter the Images properties to create images with different properties and through that, the limited availability of Data can be tackled, which will be very helpful in producing better results. By this proposed approach, the authors achieved an accuracy of about 99% while training the data and achieved an accuracy of 100% during the testing, which is quite remarkable.

The next work is much more interesting than this because it involves two stages. They are Face Detection and Face Mask Detection. In this paper, the author proposed a model called SSDMNV2 based on Tensorflow, Keras, OpenCV and MobileNetV2 (Nagrath et al., 2021). Since we are aware of some models and how they work, we can explore what is the real reason behind this name. Here, the author proposed a Face detector based on OpenCV DNN(Deep Neural Network) which includes SSD (Single Shot multi-box Detection). The MNV2 is nothing but the pre-trained model used by them for Mask detection called MobileNetV2. The reason why they choose that model is that it is lightweight in nature which requires less computational power and it is also very good in accuracy. By this proposed model the author got an accuracy of 93% which is better when compared to LeNet-5, AlexNet, VGG-16, and ResNet50. He compared them based on the model's accuracy and F-1 Score. He also mentioned that the SSDMNV2 model is not that heavy compared with the others and it could do real-time detection which makes it preferable for deployment purposes.

The other interesting and noticeable work is a real-time surveillance system based on YOLOv5 & ResNet50 to monitor face mask detection and social distancing (Walia et al., 2021). In this paper, the authors proposed a real-time surveillance system where the input video data from the CCTV footage has been transferred to the system where the YOLOv5 model will examine them frame by frame to detect the human and faces of the persons. Human detection is to determine the distance between two individuals using the pixel to real-time distance and face detection of those human detections. Once the face was detected in their images, the features of that extracted face were then transferred into a Mask classifier built on the ResNet50 pre-trained model which will predict whether the detected face was having a face mask or not. In the same way, the social distancing between the individuals would also be calculated and if it is not more than the provided guidelines it will indicate them in red. This proposed model looks more similar to our current prototype but our prototype seems a bit enhanced than this as it was having YOLOv5s and ResNet101.

The other noticeable pre-trained model that was involved in face mask detection (Image Classification) is VGG-19 (Xiao et al., 2020). VGG-19 is an upgraded version of VGG-16. In this paper, the author used the VGG-19 pre-trained model on the ImageNet dataset to construct a face mask detection deep learning model. Here, the author flattened the last fully connected layers and altered the existing SoftMax classification of 1000 categories into 2 categories (with and without a facemask). This SoftMax classification layer is called as Dense layer which is usually the end of the fully connected layer to predict the output based on the activation function. By this proposed method, over 3000 custom build image datasets, the author was supposed to achieve an accuracy of around 96%. Whereas the precision and recall of the model were 97.62% and 96.31% and for the category with the mask 96.82% and 94.07%.

While exploring some of the works related to face detection techniques, met with this interesting paper, where they try to compare two of the most widely used face detection techniques namely Viola-Jones Haar Cascade Classifier (V-J) and Histogram of Oriented Gradients (HOG) (Rahmad et al., 2020). In this paper, the author tries to compare the detection capabilities of these models in 6 scenarios namely Scale, Occlusion, make-up, Pose, Expression and illumination. To create a resilient cascade classifier, the V-J technique calculates an Integral Picture using a Haar-like feature and the AdaBoost process. HOG computes the classifier for each image in the scale of the image, applies sliding windows, extracts the HOG descriptor at each window, and applies the classifier. If the classifier detects an item with a high enough probability that it resembles a face, it records the bounding box of the window and applies non-max. After conducting these experiments, the author received average accuracy scores of 71.1% for the V-J Cascade Classifier and 79% for HoG. This shows that the HOG technique will work better than the V-J Cascade Classifier in terms of face detection. But in present case scenarios, other face detection techniques can work much better than Cascade Classifier & HOG in terms of accuracy such as DNN-based ones and Face detection with dlib and max-margin object detector.

One of the widely used single-step object detectors in the present case scenario is the YOLO algorithm. Since the day it was published, the algorithm has garnered interest with its performance and speed. Due to YOLOv5's effectiveness in detecting items, it was employed in this study of Cats & dogs by the authors (Cengil et al.,2021). Three steps make up this image classification YOLOv5 pipeline. The initial phase is the extraction of features from the images. The CSP network is utilized in the section known as the backbone of the model. Spent reduces the model's parameters by resolving recurrent gradient information issues on large-scale backbones and incorporating gradient changes into the feature map. This guarantees accurate and quick extraction of features of the images. To improve the information flow in YOLOv5, the path aggregation network (PANet) is built in the second section of the model. The dispersion of low-level features is increased by PANet's adoption of a new feature pyramid network topology with an enhanced bottom-up route. At the same time, the adaptive feature pool, which links the feature grid and all attribute levels, is employed to make sure that each attribute level's important information is transmitted straight to the subnet below. The utilization of precise localization signals in the lower layers is improved by PANet, which can increase the object's location accuracy.

The third section involves making predictions. This uses the same head that was used in YOLOv3. For multi-scale prediction, the YOLO layer produces feature maps in three distinct scales. Here, the authors used the YOLOv5 algorithm with various settings. Experiments show that YOLOv5 models produce favourable outcomes for the relevant task. The effectiveness of YOLOv5-based cat/dog detection is shown by the mAP of YOLOv5l, which is 94.1%.

YOLOv7 is the latest state of object detection algorithm proposed by authors Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao under the subject of

Computer Vision and pattern recognition (Wang et al., 2022). YOLOv7 was built on the PyTorch framework with accuracy better than YOLOv5 on the COCO Dataset. The Mean Average Precision score of YOLOv5 on the COCO dataset is 55% whereas the score of YOLOv7 on the COCO dataset is 56.8%. Also, the YOLOv7 benchmark showed that it is about 120% faster than YOLOv5 but there is a limitation on that. YOLOv7 could achieve that speed only on the latest GPUs (like Nvidia RTX 30 series, A100) and not on the older ones (like 1050,1650). Even though YOLOv7 was still in the development and optimization phase, if accuracy was the only thing to achieve, then YOLOv7 is better in terms of accuracy than every other object detection but there are some limitations like inefficient memory usage, some issues while loading the custom dataset.

DESIGN & IMPLEMENTATION

MASK DETECTION MODEL

Creating a Dataset

The Dataset used here was the real images of the person wearing masks. They are collected from a project called **Face-Mask-Detection by Chandrikadeb7 at her GitHub repository** which is an award-winning project at Project ICE – Innovative & Creative Environment event at MIT. They are collected from the sources like Bing Search API, Kaggle datasets and RMFD Datasets. The reason for choosing this dataset was its wide variety of Face Masks used such as Surgical Mask, Cloth Mask, N-95 mask and mask with different colors and patterns. These images with the mask are real-life images. There are a lot of datasets that are available online with artificially masked images. The reason for picking this one over them was the prototype that we are trying to enhance is specially built for care home surveillance. To use it in real-life scenarios for detection, the best way is to use real-life images. The Dataset contains 4092 images 2166 images with a mask and 1926 images without a mask. Once the appropriate dataset was picked, we can proceed with the project. The following figure shows the flowchart of the Mask detection model implementation process.

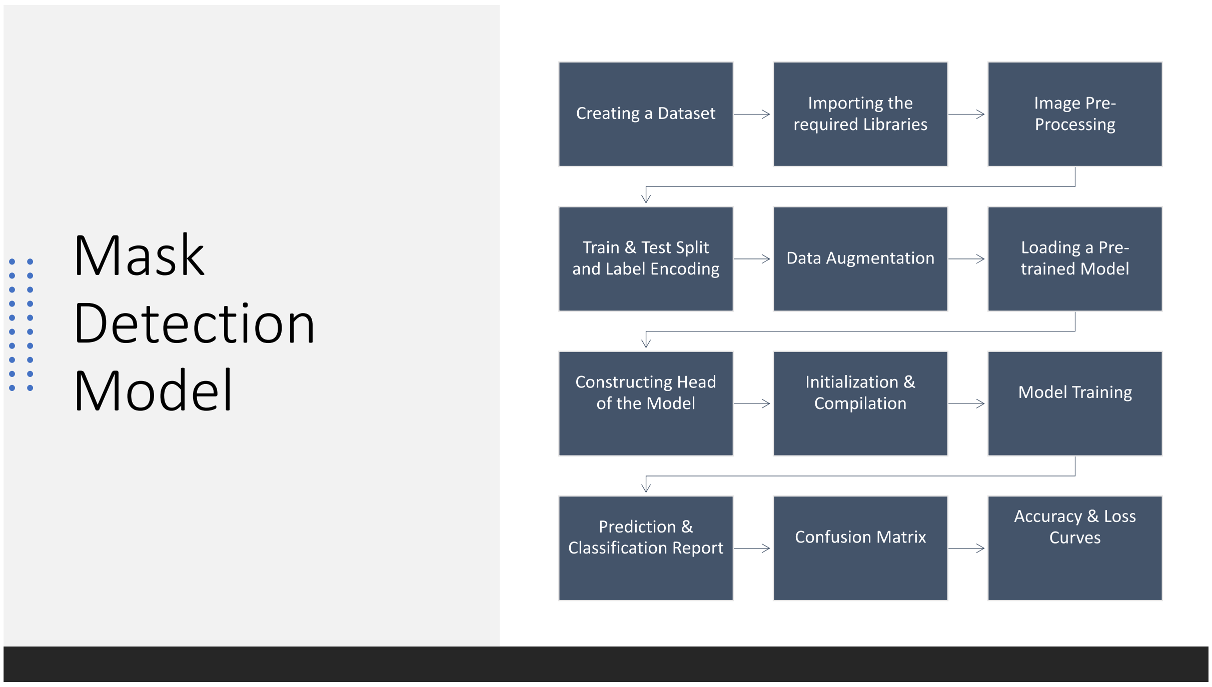
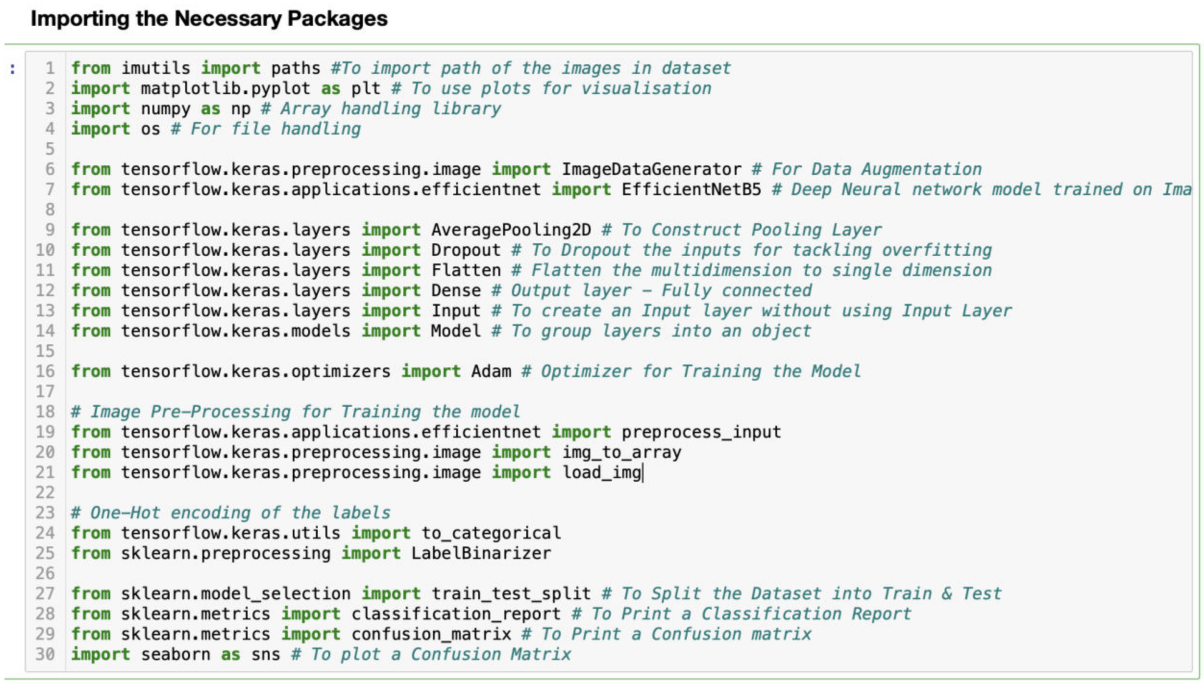

Mask Detection Model
Creating a Dataset
Importing the required Libraries
Image Pre-Processing
Train & Test Split and Label Encoding
Data Augmentation
Loading a Pre-trained Model
Constructing Head of the Model
Initialization & Compilation
Model Training
Prediction & Classification Report
Confusion Matrix
Accuracy & Loss Curves

Figure 2 – Model Implementation Flowchart

Importing Required Libraries

The first and foremost work after picking the dataset was importing the required libraries to implement the Mask detection models. The following figure shows the libraries that are imported to carry out the modelling of face mask detection. In this project, we are going to implement the face mask detection model with 7 pre-trained models and compare and evaluate their performance based on the classification report, Confusion matrix and Accuracy & Loss curves. Figure 3 shows the snippet of libraries imported.


Importing the Necessary Packages

```
from imutils import paths #To import path of the images in dataset
import matplotlib.pyplot as plt # To use plots for visualisation
import numpy as np # Array handling library
import os # For file handling

from tensorflow.keras.preprocessing.image import ImageDataGenerator # For Data Augmentation
from tensorflow.keras.applications.efficientnet import EfficientNetB5 # Deep Neural network model trained on Ima

from tensorflow.keras.layers import AveragePooling2D # To Construct Pooling Layer
from tensorflow.keras.layers import Dropout # To Dropout the inputs for tackling overfitting
from tensorflow.keras.layers import Flatten # Flatten the multidimension to single dimension
from tensorflow.keras.layers import Dense # Output layer - Fully connected
from tensorflow.keras.layers import Input # To create an Input layer without using Input Layer
from tensorflow.keras.models import Model # To group layers into an object

from tensorflow.keras.optimizers import Adam # Optimizer for Training the Model

# Image Pre-Processing for Training the model
from tensorflow.keras.applications.efficientnet import preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.preprocessing.image import load_img

# One-Hot encoding of the labels
from tensorflow.keras.utils import to_categorical
from sklearn.preprocessing import LabelBinarizer

from sklearn.model_selection import train_test_split # To Split the Dataset into Train & Test
from sklearn.metrics import classification_report # To Print a Classification Report
from sklearn.metrics import confusion_matrix # To Print a Confusion matrix
import seaborn as sns # To plot a Confusion Matrix
```

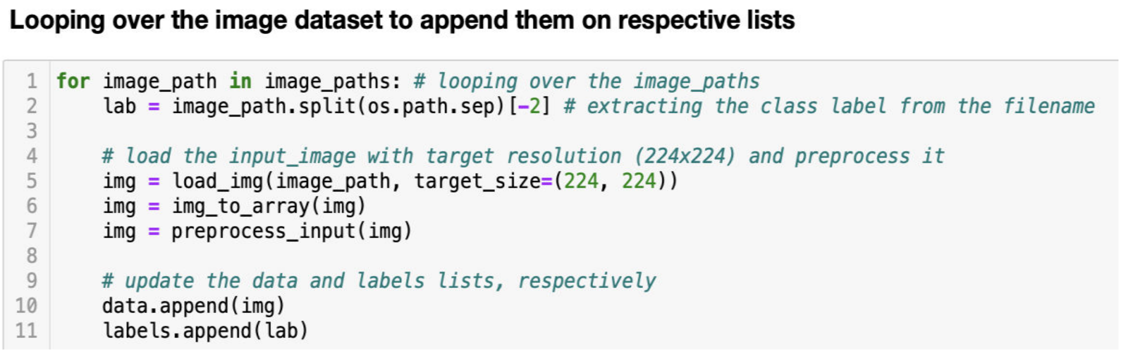
Figure 3 – Importing libraries code snippet

As we have seen from the figure, imutils is a basic image processing function which was used here to import the paths of the images. Matplotlib is a visualization package which was used to plot the Accuracy and Loss Graphs and we can also use it to display some visualizations too. NumPy is a python library for array handling such as array manipulation, transformation, indexing, and slicing etc., Here we used it to convert the images into tensors which are nothing but arrays (like matrix) where training was conducted. OS is a file-handling library used here to handle the dataset folder, and filter class names from the folders. The Machine Learning framework that we used here was TensorFlow, an ML framework by google. Keras is the Deep Learning framework built on top of TensorFlow to carry out deep learning operations. Preprocess_input, img_to_array, and load_img are the image preprocessing functions imported from the Keras preprocessing module to load the image dataset and make it ready for the splitting & training phase. Also used sklearn, a machine learning library in python to split, encode the labels, and create a classification report and confusion matrix with the help of Seaborn (a visualization library in python). We imported a function called ImageDataGenerator for the Data Augmentation technique to tackle the small dataset with image data manipulated to match the real-life scenario. The other important thing imported was the pre-trained model EfficientnetB5 from the Keras application module. Since we are going to conduct a test on 7 pre-trained models, the functions that we must change in the importing area is the pre-Trained model which we use and the pre_process input function based on the pre-trained model that we are using. The optimizer that we are going to use for all the models was Adam which was imported from Keras.Optimizer module. Though we are importing the pre-trained model we aren't going to use them straight, because they are used to classify large categories of images. We scrap those final layers and create a new final head layer with 2 labels classification. To create them, we imported pooling layer, dense, dropout, flatten, model and input function from

Keras.layers module. Once all the functions are imported, we can proceed to the image pre-processing phase.

Image Pre-Processing

The image preprocessing consists of 3 important tasks. The first one is loading the paths image dataset folder into the working environment into a python list variable and creating two empty lists to store images and labels. The second step is to create a loop over the image paths and save those into a python list variable. The same will be applied to the labels where the label names are appended to the list. Figure 4 shows the image preprocessing snippet



Looping over the image dataset to append them on respective lists

```python
for image_path in image_paths: # looping over the image_paths
    lab = image_path.split(os.path.sep)[-2] # extracting the class label from the filename

    # load the input_image with target resolution (224x224) and preprocess it
    img = load_img(image_path, target_size=(224, 224))
    img = img_to_array(img)
    img = preprocess_input(img)

    # update the data and labels lists, respectively
    data.append(img)
    labels.append(lab)
```

Figure 4 – Image loading code snippet

Even though the images are of different resolutions, while loading them up, we specified the target size as (224x224) to make them all have even resolutions. Once the image data and labels are stored into a list by looping, the lists are converted into the NumPy arrays which was like the tensor and make the training easier and more efficient.

Train Test Split & One hot Encoding

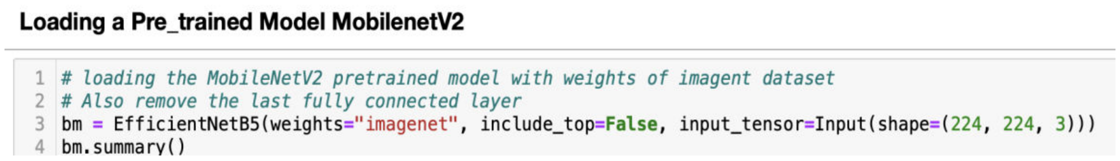
Once all the 4092 images were loaded into the variables, before training them we need to split the data into 2 categories. Train & test, at the ratio of 80% to 20% where 80% of the training data was used to train the model and the remaining 20% of the testing data was used to evaluate the model based on the true labels and the predicted labels. While splitting the dataset into 2, the other key thing to do is to do one hot encoding of the labels. As we all know computer is aware of only 1 and 0, we are one hot encoding the labels based on 0 and 1. For doing that we used LabelBinarizer () function to encode them. While splitting, we mentioned stratify function as yes to split them evenly with the labels to avoid imbalanced splitting. We also mentioned the random state function to 45 which is nothing, but the shuffling done on the dataset before carryout the splitting. After all this, the dataset was successfully split into train and test with the size of (3273,224,224,3) & (819,224,224,3) where 3273 and 819 are the number of images and (224,224,3) represent the resolution of the image and the RGB channels.

Data Augmentation

After the data was successfully split into training and testing, the next important phase is to do data augmentation. Data augmentation is a technique used to alter the existing image data to create newly transformed image data which would help in the lack of data, and detection from the different orientations, different colour levels and angles (Taqi et al., 2018). Here, the augmentation was performed to match the real-life scenario where we usually don’t look directly into the camera. We could be either angled or sideways or sometimes looking at some other place instead of the camera. To match those circumstances, we try to augment the images with rotation range, zoom range, width shift range, height shift range, shear range and horizontal flip. To do image augmentation, we used a function called ImageDataGenerator from Keras.Preprocessing.Image module.

Loading a Pre-Trained model

Once the Data Augmentation was done, the things that we need to do with the data were done. The next step is to do modelling. For modelling, first, we need to load the pre-trained model. In this case, since we are going to try 7 pre-trained models, we need to import them. For instance, if the modelling is based on the MobileNetV2, then we need to import the model from the hub using the syntax. While loading the model, we specified the weights of the ImageNet dataset, which is an image classification dataset that contains 1,281,167 training images, 50,000 validation images and 100,000 test images having a classification category of 1000. The main purpose of using that pre-trained model is the robust architecture of that model and its weights, since it is a model that was trained on 1000 objects, and while loading that pre-trained model, we could save a lot of computational power to get great accuracy. Figure 5 shows the snippet to load the pretrained model.

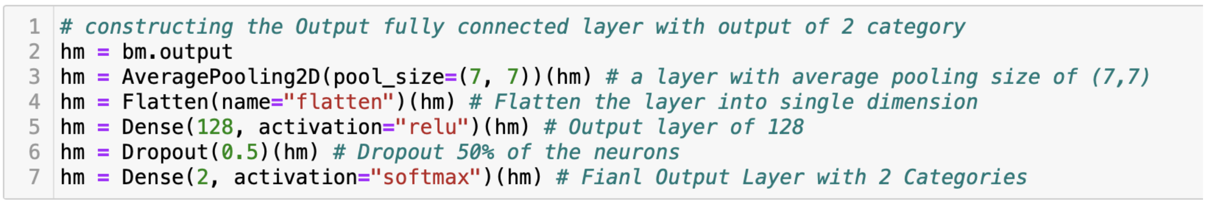

Loading a Pre_trained Model MobilenetV2

```
# loading the MobileNetV2 pretrained model with weights of imagent dataset
# Also remove the last fully connected layer
bm = EfficientNetB5(weights="imagenet", include_top=False, input_tensor=Input(shape=(224, 224, 3)))
bm.summary()
```

Figure 5 – Loading pre-trained model code snippet

While loading the pre-trained model, we specify the include_top to False, so that we can alter that top to make that model classify for our dataset. Being a deep learning model, trained on an ImageNet dataset of 1000 objects, the final output dense layer is 1000, whereas we need only 2 label classifications. So to make that model learn and classify our dataset, we strip that final head layer and replace it with one that could match our dataset and label classification.

Constructing the Head of the Model



```
# constructing the Output fully connected layer with output of 2 category
hm = bm.output
hm = AveragePooling2D(pool_size=(7, 7))(hm) # a layer with average pooling size of (7,7)
hm = Flatten(name="flatten")(hm) # Flatten the layer into single dimension
hm = Dense(128, activation="relu")(hm) # Output layer of 128
hm = Dropout(0.5)(hm) # Dropout 50% of the neurons
hm = Dense(2, activation="softmax")(hm) # Fianl Output Layer with 2 Categories
```

Figure 6 – Head of the model construction code snippet

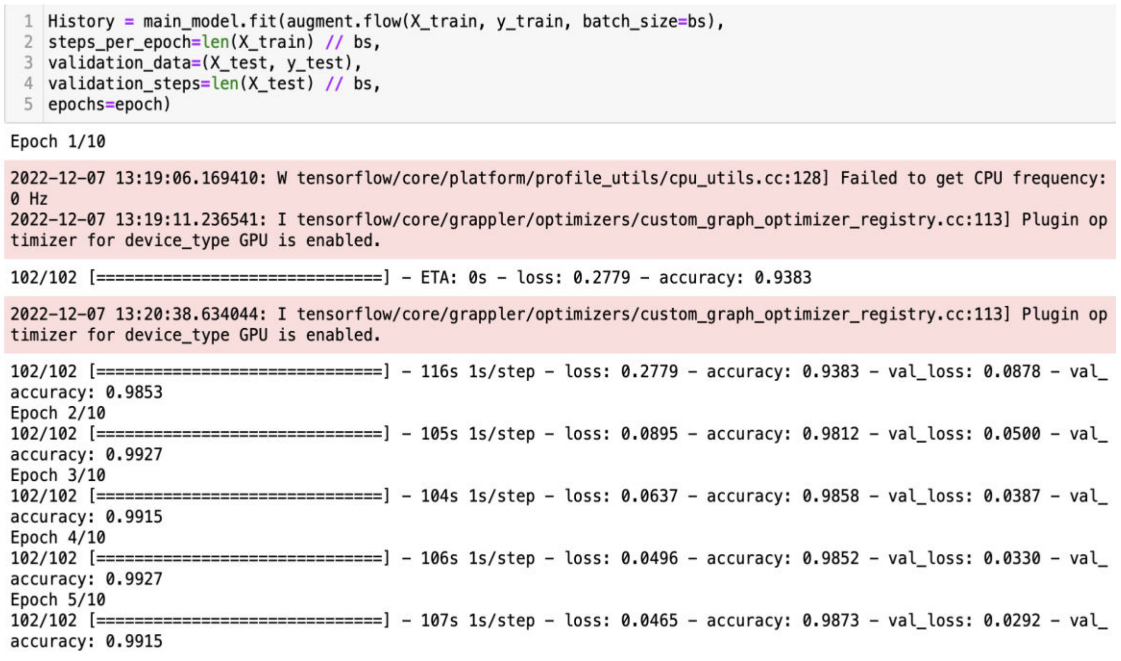
While constructing the Head of the model, we always freeze the layers in the base model i.e.., the pre-trained MobileNetV2 model without a head, so that their weights of them aren't modified while learning. Since it is a model that is trained on a very huge dataset, its weights will produce far superior results on the given small dataset. Once the layers are frozen, we then construct a final pooling layer with a filter size of (7,7) and then flatten them into a single dimension to produce the result. Pooling layers are the layers in Convolutional Neural Network (CNN) used to minimize the feature map dimensions. It is more like a summary writer, which summarizes the information from more dimensions and layers into fewer dimensions and layers. After that, a dense layer of 128 labels was created which is followed by a dropout function of 0.5 with ReLu activation, which means that it will drop 50% of the neurons. The dropout was used to tackle the overfitting of the model. After that, a final dense layer of 2 labels was added with the SoftMax activation function which gives the output with a mask or without a mask (Figure 6). The Activation function is used to decide whether the neurons should be activated or not based on the weights of the model. SoftMax & ReLu are some of the activation functions that behave in a certain way. Based on our needs, we can use them in our model to perform accordingly.

Initialization & Compilation

Initialization and Compilation is the phase before the model training. Here, we mostly initialize the parameters that are required for the training. The parameters are such as learning rate, epochs, batch size, optimizer, decay rate, loss, and metrics. The learning rate is the rate at which the optimizer should move towards the minimum loss function. When the learning rate is getting smaller, there would be a high chance of attaining a global minimum. Epoch is nothing but one full cycle of training. Batch size is the group of data trained at once while training. The optimizer is the parameter, that adjusts the weights of the model while training. The decay rate in simple terms is the depreciation of the Learning rate. While the loss is getting minimized, the learning rate is reduced at the decay rate level to obtain the global minimum. The loss function is the function, that represents the model's performance. It usually compares the prediction & the test data and produces results in the name of the loss function and shows how much optimization is still needed to make the model perfect. Metrics represent the accuracy of the model's performance. In this research, we mentioned the same initialization and compilation parameters for all the pre-trained models since we are going to compare all the models. The learning is 1e-4, which is 0.0001, epochs to 10, batch size to 32, Adam optimizer, binary_crossentrophy loss function since we did one hot encoding and accuracy metrics.

Model training

Model training is the key and important part of modelling. This is where the model gets trained from the dataset using the given parameters. The model can be trained either on CPU or GPU. Figure 7 shows the model training with its code snippet.



```
History = main_model.fit(augment.flow(X_train, y_train, batch_size=bs),
steps_per_epoch=len(X_train) // bs,
validation_data=(X_test, y_test),
validation_steps=len(X_test) // bs,
epochs=epoch)
```

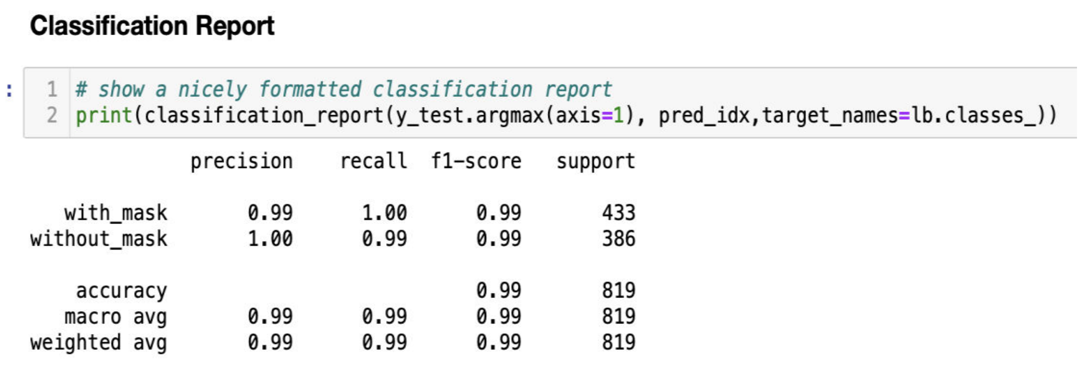
```
Epoch 1/10
2022-12-07 13:19:06.169410: W tensorflow/core/platform/profile_utils/cpu_utils.cc:128] Failed to get CPU frequency:
0 Hz
2022-12-07 13:19:11.236541: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:113] Plugin op
timizer for device_type GPU is enabled.
102/102 [==============================] - ETA: 0s - loss: 0.2779 - accuracy: 0.9383
2022-12-07 13:20:38.634044: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:113] Plugin op
timizer for device_type GPU is enabled.
102/102 [==============================] - 116s 1s/step - loss: 0.2779 - accuracy: 0.9383 - val_loss: 0.0878 - val_
accuracy: 0.9853
Epoch 2/10
102/102 [==============================] - 105s 1s/step - loss: 0.0895 - accuracy: 0.9812 - val_loss: 0.0500 - val_
accuracy: 0.9927
Epoch 3/10
102/102 [==============================] - 104s 1s/step - loss: 0.0637 - accuracy: 0.9858 - val_loss: 0.0387 - val_
accuracy: 0.9915
Epoch 4/10
102/102 [==============================] - 106s 1s/step - loss: 0.0496 - accuracy: 0.9852 - val_loss: 0.0330 - val_
accuracy: 0.9927
Epoch 5/10
102/102 [==============================] - 107s 1s/step - loss: 0.0465 - accuracy: 0.9873 - val_loss: 0.0292 - val_
accuracy: 0.9915
```

Figure – 7 Model Training code snippet

Once the model was created with the newly constructed head layer, the model was fit onto the dataset using the model.fit command. The model while training was validated with testing data (X_test, y_test). As we can see from the fig, for each epoch, the model was trained one full cycle and then on backpropagation, the model readjusts its weights based on the optimizer & loss function, with re-adjusted weights the next forward pass providing the minimized loss with improved accuracy and validation accuracy. In this, the loss reduced from 0.2779 from the 1st epoch to 0.0895 in the 2nd epoch with training accuracy increased from 93.83% to 98.12%. In the same way, the validation accuracy increased from 98.53% to 99.27%.

Prediction & Classification Report

When the model attained its lowest loss value and the highest accuracy scores, we then stop the training and use that model on the test data to predict the test data. These predictions can be done using the predicted model.predict function over the test dataset. Since we hot encoded the label, the deep learning model activation function produces the probability of predicted labels. To convert that probability of predicted labels, we use the np.argmax function to determine the predicted label with the highest probability. After these labels were predicted for the test dataset, we then create a classification report with the predicted labels and true labels of the test data. Figure 8 shows the classification report for the model with its code snippet.



Classification Report

```python
# show a nicely formatted classification report
print(classification_report(y_test.argmax(axis=1), pred_idx,target_names=lb.classes_))
```

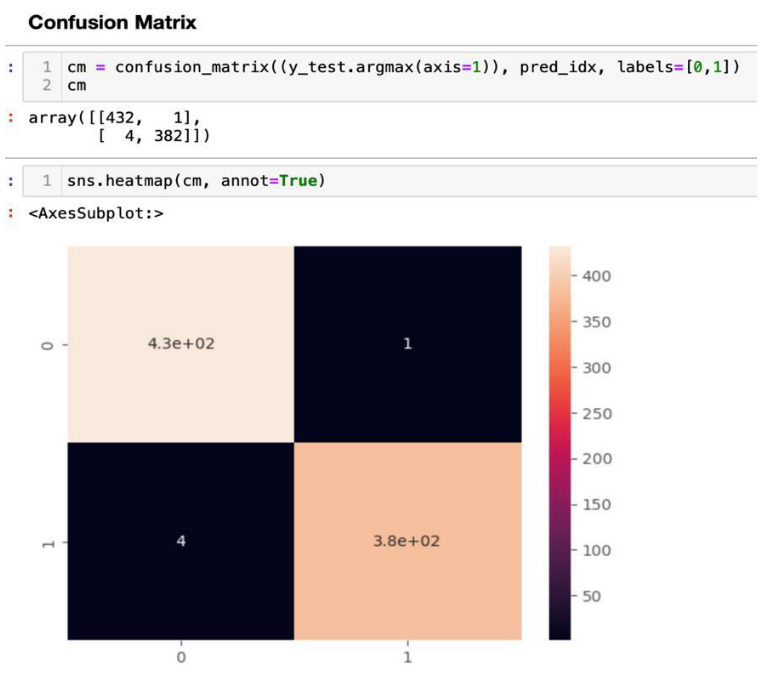
```
              precision    recall  f1-score   support

   with_mask       0.99      1.00      0.99       433
without_mask       1.00      0.99      0.99       386

    accuracy                           0.99       819
   macro avg       0.99      0.99      0.99       819
weighted avg       0.99      0.99      0.99       819
```

Figure 8 – Classification Report code snippet

This classification report includes precision, recall, f1 and support score. The details about them will be discussed in the evaluation section.

Confusion matrix

A confusion matrix is a simple and effective way of visualizing the image classification capability of the model. Since it is a 2-label classification, the confusion matrix has 4 matrix values, which are True Positive, True Negative, False Positive & False Negative. Here, the True positive is the with mask label correctly classified as with mask (432), the True Negative is with mask classified wrongly classified as without mask (4), False Positive is the without mask rightly classified as without (382) and False Negative is the without mask wrongly classified as with mask (1). The following figure shows the confusion matrix and its code snippet.



Confusion Matrix

```python
cm = confusion_matrix((y_test.argmax(axis=1)), pred_idx, labels=[0,1])
cm
```

```
array([[432,   1],
       [  4, 382]])
```

```python
sns.heatmap(cm, annot=True)
```

```
<AxesSubplot:>
```

Figure – 9 Confusion Matrix code snippet

Accuracy & Loss Curves

Plotting the accuracy and loss curves is a way of visualization, where we could see the increase or decrease in Training & Validation (Figure 10).

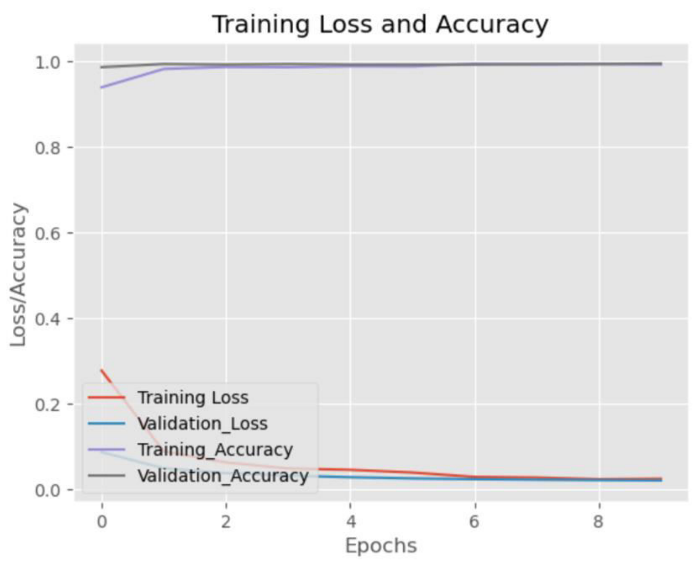


Figure 10 – Accuracy and loss curves code snippet

Accuracy and Training & Validation loss. By observing the graphs, we can determine whether the model is overfitting or underfitting. We used the matplotlib plotting function and ggplot style to plot these learning curves. Loss, accuracy, val_loss & val_accuracy values are used to plot these learning curves. Also created legends, xlabel, ylabel & title with plt.legend, plt.xlabel, plt.ylabel and plt.title functions of the matplotlib packages. The evaluation based on the graphs will be discussed in the evaluation section.

Face Detection Methods

Face Detection is a part of object detection where the system is specifically designed to identify only human faces. Here we are going to explore some of the popular face detection methods such as Haar Cascade Classifier, dlib frontal face classifier, Multitask cascaded convolutional neural network, dlib MMOD CNN, YOLO and OpenCV dnn module-based detector.

Haar Cascade Classifier

Haar Cascade Classifier was a face detection technique proposed by Paul Viola and Micheal Jones on the year of 2001 (Viola & Jones, 2001). They proposed this method in the article. The key advantage of this method is its speed. It works like a simple CNN model where it extracts a lot of features from the images. Once the features are extracted, it was then filtered with a technique called Adaboost by which we can reduce the features. Even though the features are reduced, while applying it through a sliding window, it will take more time to process and produce the output. To process that, a grouping system called cascading was introduced, where if the window at the first stage is failed, then it won’t test the remaining features of that group. Like this, the sliding window process was repeated and if there is a face region there, it would allow all the features of the window and detect the face region.

Dlib frontal face detector

Dlib is a C++ toolbox with machine-learning techniques that may be used to address practical issues. Despite being built in C++, it can be executed in Python thanks to its python bindings. The Histogram of Oriented Gradients (HOG)-extracted features used in the frontal face detector offered by dlib is then subjected to an SVM (Boyko et al., 2018). The distribution of gradient directional information is employed as a feature in the HOG feature descriptor. The dlib also provides a CNN-based detector which was good in terms of accuracy but the main limitation of that was its computational power. Those models can be run only in GPU if we would like to do some real-time predictions.

Multitask Cascaded Convolutional Neural Network (MTCNN)

As the name suggests, it is three stages of CNN cascade arrangement. Initially, the first fully connected network layer collects the candidate window and the regression vectors of the bounding boxes. Once they were collected, then the maximum suppression was applied to them to reduce the high overlap candidates (NMS). After that these candidates are passed into the second layer of CNN, through which the bounding box calibration was done and removes a certain number of false positives. The final third layer is where the facial landmark detection was done (Yang & Zhang ,2022).

Dlib MMOD CNN face detector

The Dlib MMOD CNN-based detector is widely accepted because of its accuracy over the other methods (Jadhav et al., 2021). Even though they are accurate in terms of detection, there is a limitation which is the computational power. Generally, this method would require more computational power than other ones which makes it hard to run over the CPU. If the real-time use case is the main objective, then GPU acceleration would be the best way to run this model to get the proper FPS.

DNN Face Detector in OpenCV

DNN face detector is a face detection function introduced in OpenCV 3.3 in its Deep Neural network module. This is one of the widely used models for face detection in the real world because of its lightweight and detection capabilities. Being lightweight makes it able to run even on a CPU without any GPU acceleration. It is based on a single-shot multibox detector (SSD) and named as Caffe model (Nagrath et al., 2021). The main architecture of this Caffe model is ResNet-10

YOLO

YOLO algorithms (You Look Only Once) are considered state-of-the-art Deep learning algorithms for object detection (Yang & Jiachun, 2018). They are specifically built for that purpose. Usually, YOLO was built on Deep Learning models mostly with CNNs since CNN works very well on the images. YOLO was built to detect a variety of objects with a high level of accuracy. The main limitation of YOLO was its highly dense architecture which makes it hard to run on the CPU and only GPU acceleration could help to run it faster in real-time. To overcome this YOLO had a mini version called YOLO-Tiny which requires very less computational power at the cost of its accuracy. This makes it hard to suggest face detection in real-time scenarios since accuracy is most important. Even though YOLO is very accurate and faster than MTCNN, because of its dense network topology, it takes more VRAM on the GPU to run it in real-time with good FPS.

EVALUATION

FACE MASK DETECTION

Evaluation metrics are the methods used to measure the performance of the model. These evaluation metrics used to measure the performance is ideal because only then we can find out the performance of the model and its areas of improvement that can be done over the model to increase its classification capabilities. The most used evaluation metrics for the image classification models are the Confusion Matrix, Accuracy score, Sensitivity, Recall & Precision score (Krstinić et al., 2020). The reason for using metrics other than Accuracy metrics for the classification problems was to eliminate the limitations of that if they had any. The main limitation of the accuracy metrics was its nature of struggling with imbalanced data. When the data is balanced, the accuracy metrics will reflect the real performance of the data. But when the data is imbalanced there may be a risk that the model may perform well but not in the prediction of that smaller number of labels present while training that model. Let’s start discussing the results based on the evaluation metrics.

Confusion Matrix

A confusion matrix is an evaluation metric in a form of a table used to measure the performance of the machine learning model. This confusion matrix holds the value of predicted labels and actual true labels. Since it is based on the comparison of predicted labels with the actual labels, we can’t evaluate the unsupervised models with a confusion matrix. If the classification is about the prediction of two labels, it would create 4 values, which are called True Positive, True Negative, False Positive, and False Negative. Here, in this research, the two labels that we used for the model are with mask and without a mask. In fig, as we can see, there are four compartments (0,0), (1,1), (0,1), and (1,0) which represents the value of True Positive, True Negative, False Positive and False Negative.

True Positive (TP):

True Positive is the value of correctly classified with mask label based on the actual true labels.

True Negative (TN):

True Negative is the value of correctly classified without mask label based on the actual true labels.

False Positive (FP):

False Positive is the value of incorrectly classified with mask label based on the actual true labels.

False Negative (FN):

False Negative is the value of incorrectly classified without mask label based on the actual true labels.

No.	Models	True Positive	False Positive	True negative	False Negative	Total Prediction Items
1	MobileNetV2	430	16	370	3	819
2	ResNet50V2	432	6	380	1	819
3	ResNet101	433	4	382	0	819
4	InceptionV3	432	1	385	1	819
5	VGG16	431	12	374	2	819
6	VGG19	431	12	374	2	819
7	EfficientNetB5	432	4	382	1	819

Table 1 – Confusion Matrix scores for models

Based on the data from a table, which was collected from the Confusion Matrix of each model, we could see that ResNet101, InceptionV3 and EfficientNetB5 are performing well with very few number of False Positives & False Negatives. Among them, EfficientNet was good with 5 total false predictions, ResNet101 was better with 4 false predictions and the best was InceptionV3 with just 2 false predictions among 819 predictions. Apart from them, one other good model based on the Confusion Matrix was ResNet50V2 with 7 false predictions among 819 predictions. The other models even though performing well overall with less than 20 false predictions among the 819 predictions are not up to the level of the best-performing ones with less than 5 false predictions.

Accuracy Score

The Accuracy Score is the measure of total correctly classified labels from the total prediction items. To be precise, correctly classified with mask and without mask labels from the overall predictions.

Accuracy Score = [TP + TN] / [TP + TN + FP + FN]

No.	Models	Accuracy
1	MobileNetV2	0.98
2	ResNet50V2	0.99
3	ResNet101	1
4	InceptionV3	1
5	VGG16	0.98
6	VGG19	0.98
7	EfficientNetB5	0.99

Table 2 – Accuracy Scores for the models

According to the table, we could conclude that only two models achieved the highest accuracy scores of 100% which are ResNet101 & InceptionV3, and the other best performing models are EfficientNetB5 and ResNet50V2 which attained an accuracy of 99%. The other three models followed them with the next-best accuracy of 98%. The accuracy scores also followed that pattern of the Confusion Matrix where InceptionV3 tops the list with ResNet101 coming close second and Third by EfficientNetB5.

Precision

Since the accuracy score has a limitation while handling, we used to do precision metrics to analyze the prediction capabilities based on the labels. The precision score is the measure of TP to the total predicted positives of that label.

Precision Score = TP / [TP + FP]

No.	Models	Precision	
		with mask	without mask
1	MobileNetV2	0.96	0.99
2	ResNet50V2	0.99	1
3	ResNet101	0.99	1
4	InceptionV3	1	1
5	VGG16	0.97	0.99
6	VGG19	0.97	0.99
7	EfficientNetB5	0.99	1

Table 3 – Precision Scores for Models

According to the scores based on the models, we could conclude that InceptionV3 tops the list with the best accuracy scores of 100% on both the labels which in turn matches the pattern of the confusion matrix & accuracy score since it is a balanced dataset. The ResNet101, ResNet50V2 & EfficientNetB5 come a close second with 100% on one label and 99% on the other one. Even though ResNet101 scored a 100% percent accuracy score, here in precision we could see that it attains a score of 99%, especially in the mask label prediction which makes it a close second. The other three models are performing well but not up to the level of Inception, ResNet & EfficientNet.

Recall

Recall score is the measure of TP to the actual positives of that label.

Recall Score = TP / [TP + FN]

No.	Models	Recall	
		with mask	without mask
1	MobileNetV2	0.99	0.96
2	ResNet50V2	1	0.98
3	ResNet101	1	0.99
4	InceptionV3	1	1
5	VGG16	1	0.97
6	VGG19	1	0.97
7	EfficientNetB5	1	0.99

Table 4 – Recall Scores for the models

The recall score is also called as sensitivity score. As per the table scores, the pattern of Confusion matrix, accuracy & precision follows here with InceptionV3 topping the list with 100% accuracy on both labels. ResNet101 & EfficientNetB5 follow close second with 100% on one and 99% on the other label. The other model's performance is not matching the level of these three models while comparing their scores. Surprisingly, VGG models perform better in terms of mask prediction with an accuracy of 100% but the without mask label it them down by 97% accuracy.

F1 Score

The F1 score is nothing but the aggregate of Precision & Recall. To be precise, the F1 score is the average of precision & recall.

F1 Score = 2 x [Precision x Recall] / [Precision + Recall]

No.	Models	F1 Score	
		with mask	without mask
1	MobileNetV2	0.98	0.97
2	ResNet50V2	0.99	0.99
3	ResNet101	1	0.99
4	InceptionV3	1	1
5	VGG16	0.98	0.98
6	VGG19	0.98	0.98
7	EfficientNetB5	0.99	0.99

Table 5 – F1 scores for the models

According to the scores based on the models, we could conclude that InceptionV3 tops the list with the best accuracy scores of 100% on both the labels which in turn matches the pattern of the confusion matrix, accuracy score, Precision & Recall since it is a balanced dataset. The ResNet101 comes a close second with 100% on one label and 99% on the other one. Even though ResNet101 scored a 100% percent accuracy score, here in the F1 score we could see that it attains a score of 99%, especially in the without mask label prediction which makes it a close second. ResNet50V2 & EfficientNetB5 come next with an accuracy of 99% on both labels. The other three models are performing well but not up to the level of Inception, ResNet & EfficientNet.

Parameters & Epoch time with Accuracy

Though this isn't an evaluation metric, it could depict the models' parameters that are at work during training and how well they performed. From this data, we can predict the model that is not just accurate in predictions but also efficient while training, which means the quickest while training so that we could save some computational power and time.

No.	Models	Trainable Parameters	Non Trainable Parameters	Total Parameters	Avg Time/epoch training	Accuracy
1	MobileNetV2	2388098	34112	2422210	27.3	0.98
2	ResNet50V2	23781890	45440	23827330	44.2	0.99
3	ResNet101	42815362	105344	42920706	80.9	1
4	InceptionV3	22030882	34432	22065314	31.8	1
5	VGG16	14780610	0	14780610	86.5	0.98
6	VGG19	20090306	0	20090306	108.9	0.98
7	EfficientNetB5	28603314	172743	28776507	108.9	0.99

Table 6 – Parameter and Epoch time for the models

As per the table data, ResNet101 & InceptionV3 achieved the topmost accuracy score of 100% with the total parameters of 42920706 & 22065314. Among them, InceptionV3 has less than half of the parameters when compared to the ResNet101 which makes it take less time to train. Thus, giving it an advantage over ResNet and other models. The other model which takes less time than InceptionV3 is MobileNetV2 at 27.3s per epoch but it lacks in accuracy part. So, while combining the epoch training time and accuracy score, InceptionV3 performs best with ResNet101 being second. The other interesting insight here is that ResNet101 is approximately double the parameters of EfficientNetB5 but trained 30s less than the EfficientNetB5 with good accuracy. The other good performing one was ResNet50V2 with an accuracy of 99% and training time of 44.2s which makes it close third to ResNet101 on second. Here, EfficientNet takes the fourth spot because of its high training time.

OBJECT DETECTION (FACE DETECTION & SOCIAL DISTANCING) METRICS

Intersection Over Union (IOU)

Intersection over union is the ground measure that finds the difference between the bounding boxes of the Predicted object and the ground truth. Most cutting-edge object identification algorithms employ this measure. The model predicts numerous bounding boxes for each item during object detection and based on the confidence ratings of each bounding box, it eliminates any extraneous boxes that are over a certain level. Based on our needs, we must specify the threshold value.

IOU = Area of union / Area of Intersection

Mean Average Precision

An extension of average precision is mean average precision. While we only compute individual items in Average precision, mAP provides accuracy for the entire model. We use mAP to determine the model's proportion of accurate predictions.

Inference Time

Inference time is nothing, but the time taken to calculate the total objects in the image. Inference time can be based on either CPU or GPU based on which the model is running.

Based on the GitHub repository of **nodefluxio/face-detector-benchmark** that contains scripts to deploy publicly available face detection models and benchmark them against the famous WIDER face dataset, we extracted the benchmark test results because of the dependency issues while installing the dlib library in the newly launched M1 apple silicon based MacBook. Those results were evaluated with the metrics to determine the best-performing method.

Models	Average IOU	mAP	Inferencing time (on CPU)	Inferencing time (On GPU)
OpenCV Haar Cascade Face Detector	0.219	0.307	0.159	-
DLib HOG Face Detector	0.253	0.365	0.239	-
DLib CNN MMOD Face Detector	0.286	0.416	4.534	0.111
Tensorflow MTCNN Face Detector	0.417	0.517	1.979	0.699

Table 7 – IOU, mAP & Inference time for the models

According to the tables showing the test results of the Face Detection models, Multitask Cascaded Convolutional Neural Network achieved the best IOU and the mAP score of 0.417 & 0.517 which marks the highest score among the other models. The other model that performs the close second was the Dlib CNN MMOD Face detector with the IOU & mAP score of 0.286 & 0.416. The other two models, though very fast in process, they lack accuracy. CNN & MTCNN models are dense ones, which makes them hard to train on CPU and requires GPU acceleration. Through GPU acceleration, they were much faster compared to their CPU performance. Particularly CNN MMOD was a lot faster on GPU than CPU. From these results, we can conclude that MTCNN performs better with high accuracy and good speed in GPU.

The other 3 methods for face detection were YOLOv5, YOLOv7 & OpenCV-based DNN module. Among these, the YOLOv7 was having an mAP of 56.8% compared to 55% mAP for YOLOv5 on the COCO dataset. From this, we could determine that YOLOv7, the state-of-the-art deep learning model would perform best for face detection & social distancing.

DISCUSSION

Based on the evaluation metrics and the performance of the model, we could identify the best-performing model which could probably increase the already implemented prototype by enhancing the detection capabilities. Since we are researching a prototype that could detect and monitor the preventive measures of the care homes, such as Face Mask Detection & Social Distancing, which is based on the YOLOv5 algorithm for its face detection & social distancing part and ResNet101 for the mask classification part, our tests were primarily based on the different models including ResNet101 on the image classification segment and with YOLOv5 for the object detection segment.

In the image classification segment (Mask Classification), we compared around 7 models with the custom face dataset to measure their performance over the others. While doing so we also included the image classification model that was used in the prototype, ResNet101 to make the comparison. Once the models were trained and their results were evaluated with the metrics, we could identify that the InceptionV3 pre-trained model performs slightly better than the ResNet101 based on the Precision, Recall & the F1 score (100% for InceptionV3 vs 99% on ResNet101). It also performed better in terms of training speed with fewer parameters but with slightly better accuracy. By this, we can conclude that the use of InceptionV3 over the ResNet101 model on the prototype could slightly enhance the face mask detection capabilities of the prototype.

Being a low dense architecture with a smaller number of parameters (half the parameter size of ResNet101), the InceptionV3 model performs better in terms of overall accuracy, precision, recall & f1 score. The key reason for this is that when the neural network topology is getting deeper, there arises a problem of accuracy saturation which could make the accuracy of the model degrade thereafter. To tackle this problem, ResNet employs a technique called Residual Mapping. Even though this helps the deep ResNet models to optimize and achieve better accuracy, being just 2 label classification, this deeper network topology could be a lot. That’s why InceptionV3 works better here compared to the ResNet101 being a less dense and faster training one. Also, InceptionV3 includes several optimization-based convolutional techniques such as Factorized Convolutional, Asymmetric Convolutional, and auxiliary classifiers based on Convolutional which helps it perform better.

The other part where we want to compare is the object detection part (face detection & social distancing). This object detection part was quite easy to conclude because the latest state-of-the-art object detection technique called YOLOv7 was recently proposed and was proven to be a great performer compared to YOLOv5 which was used in the prototype in terms of both speed and accuracy. The results show that YOLOv7 has an mAP of 56.8% on the COCO dataset whereas the YOLOv5 is 55%. Even though there are some limitations of the YOLOv7 being a newly proposed and still a long way to optimize it, if accuracy is the priority, we can blindly use them if the computational power was not a constraint. If the proper GPU was there, then results show that YOLOv7 could perform 120x faster than YOLOv5.

To make YOLOv7 a beast in object detection, the author used four key area improvements over the other YOLO algorithms, such as Extended efficient layer aggregation, model scaling techniques, Re-parameterization planning and auxiliary head coarse to fine. In the final layer for aggregation, the author chooses E-Elan an extended version of the ELAN computational block. While scaling author used concatenating layers together by scaling the width and depth of the network. In Re-parameterization planning the author used gradient flow propagation paths to identify the Re-parameterization methods that we must use and the ones which we should avoid. With these changes to the architecture, the author makes YOLOv7 an object detection benchmark compared to the other YOLO versions.

From this discussion, we could conclude with the two proposed models based on the comparison and how they could probably enhance the detection capabilities of the prototype with some valid metrics and architecture differences. Even though this could enhance the prototype, still, there are a few future works that need to be performed in these areas to make this one a robust prototype that could work well in every scenario or location where it deployed.

FUTURE WORKS

Dataset

Improving the dataset would naturally improve the quality of detection and its learning of doing detection of a variety of people from different regions. People from different regions would vary slightly by their colour, facial structure and physical appearance. Expanding the dataset with a variety of people could also expand the model's detection capabilities. Also, it is better to add them as natural images instead of artificially created ones based on real-life images which could match the real-life scenario better, which makes the model perform better in real-life situations.

Mask

Whenever some piece of technology was introduced in monitoring, people would always find a way to identify its weakness and deceive it with those weaknesses. One of the challenges in identifying face masks is that it needs to identify when people are covering their faces with some other things without a mask. To tackle this, we may include some other label called improper mask and add data like them and train the machine accordingly to predict them as an improper mask which could help them identify them in real-time.

Optimization & Calibration

While examining the prototype, the distance detection information on the prototype was not accurate because of its lack of optimization and camera calibration of pixel-to-distance mapping and depth information. Some fine-tuning work was needed in this area to make those predictions accurate.

Facial Temperature Detection

The facial temperature detection part of the prototype still needs much work to do in terms of optimization and accuracy. Based on the examination of the prototype in the lab, the infrared sensor gives some extreme results in some scenario which needs some proper positional value and conditional structure to determine the temperature variation of the forehead and lacrimal caruncle area.

CONCLUSION

Here, we researched a few pre-trained models including the ResNet101 for the image classification where the ResNet101 was used in the prototype and a few object detection techniques including YOLOv5 which was used in the prototype to determine their performance based on a given Custom Face mask dataset for image classification and the WIDER face dataset & COCO dataset for the object detection. Based on the observed results and evaluation, we conclude this research with two proposed models which could enhance the detection and classification capabilities of the prototype. They are YOLOv7 for the object detection (Face detection & Social Distancing part) and InceptionV3 for the image classification (Mask Classification) part based on the results evaluated with the evaluation metrics such as Confusion Matrix, Accuracy, Precision, Recall, F1 Score and Parameters training speed for Image classification and Intersection over Union, Mean Average Precision, and inference time for the object detection and given some ideas about how these models perform better than their competitors in the given tests by exploring their architectural advantages in the discussion report and conclude with some of the future works that still needs to be done on this research.

REFERENCES

Boyko, N., Basystiuk, O., & Shakhovska, N. (2018, August). Performance evaluation and comparison of software for face recognition, based on dlib and opencv library. In *2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP)* (pp. 478-482). IEEE.

Brown, R., Rocha, A., & Cowling, M. (2020). <? covid19?> Financing entrepreneurship in times of crisis: exploring the impact of COVID-19 on the market for entrepreneurial finance in the United Kingdom. *International Small Business Journal*, *38*(5), 380-390.

Cengil, E., Çinar, A., & Yildirim, M. (2021, September). A Case Study: Cat-Dog Face Detector Based on YOLOv5. In *2021 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)* (pp. 149-153). IEEE.

Comas-Herrera, A., Glanz, A., Curry, N., Deeny, S., Hatton, C., Hemmings, N., ... & Suarez-Gonzalez, A. (2020). The COVID-19 long-term care situation in England. *LTCcovid. org, International Long-Term Care Policy Network, CPEC-LSE*.

Coronavirus disease (COVID-19) pandemic. (2022, December 1). https://www.who.int/europe/emergencies/situations/covid-19

Daly, M. (2020). COVID-19 and care homes in England: What happened and why?. *Social Policy & Administration*, *54*(7), 985-998.

De Vos, J. (2020). The effect of COVID-19 and subsequent social distancing on travel behavior. *Transportation Research Interdisciplinary Perspectives*, *5*, 100121.

Hsiao, T. C., Chuang, H. C., Griffith, S. M., Chen, S. J., & Young, L. H. (2020). COVID-19: An aerosol's point of view from expiration to transmission to viral-mechanism. *Aerosol and Air Quality Research*, *20*(5), 905-910.

Jadhav, A., Lone, S., Matey, S., Madamwar, T., & Jakhete, S. (2021). Survey on face detection algorithms. *International Journal of Innovative Science and Research Technology*, *6*(2).

Jignesh Chowdary, G., Punn, N. S., Sonbhadra, S. K., & Agarwal, S. (2020, December). Face mask detection using transfer learning of inceptionv3. In *International Conference on Big Data Analytics* (pp. 81-90). Springer, Cham.

Kostyál, L. Á., Széman, Z., Almási, V. E., Fabbietti, P., Quattrini, S., Socci, M., ... & Gagliardi, C. (2021). Impact of the COVID-19 pandemic on family carers of older people living with dementia in Italy and Hungary. *Sustainability*, *13*(13), 7107.

Krstinić, D., Braović, M., Šerić, L., & Božić-Štulić, D. (2020). Multi-label classifier performance evaluation with confusion matrix. *Comput Sci Inf Technol*, *10*, 1-14.

Machida, M., Nakamura, I., Saito, R., Nakaya, T., Hanibuchi, T., Takamiya, T., ... & Inoue, S. (2020). Incorrect use of face masks during the current COVID-19 pandemic among the general public in Japan. *International Journal of Environmental Research and Public Health*, *17*(18), 6484.

Nagrath, P., Jain, R., Madan, A., Arora, R., Kataria, P., & Hemanth, J. (2021). SSDMNV2: A real time DNN-based face mask detection system using single shot multibox detector and MobileNetV2. *Sustainable cities and society*, *66*, 102692.

O'Caoimh, R., O'Donovan, M. R., Monahan, M. P., Dalton O'Connor, C., Buckley, C., Kilty, C., ... & Cornally, N. (2020). Psychosocial impact of COVID-19 nursing home restrictions on visitors of residents with cognitive impairment: a cross-sectional study as part of the engaging remotely in care (ERiC) project. *Frontiers in psychiatry*, *11*, 585373.

Rahmad, C., Asmara, R. A., Putra, D. R. H., Dharma, I., Darmono, H., & Muhiqqin, I. (2020). Comparison of Viola-Jones Haar Cascade classifier and histogram of oriented gradients (HOG) for face detection. In *IOP conference series: materials science and engineering* (Vol. 732, No. 1, p. 012038). IOP Publishing.

Rich, E. (1983). *Artificial intelligence*. McGraw-Hill, Inc..

Sanyaolu, A., Okorie, C., Marinkovic, A., Patidar, R., Younis, K., Desai, P., ... & Altaf, M. (2020). Comorbidity and its impact on patients with COVID-19. *SN comprehensive clinical medicine*, *2*(8), 1069-1076.

Schellack, N., Coetzee, M., Schellack, G., Gijzelaar, M., Hassim, Z., Milne, M., ... & Gray, A. L. (2020). COVID-19: Guidelines for pharmacists in South Africa. *Southern African Journal of Infectious Diseases*, *35*(1), 1-10.

Shapiro, L. G., & Stockman, G. C. (2001). *Computer vision* (Vol. 3). New Jersey: Prentice Hall.

Taqi, A. M., Awad, A., Al-Azzo, F., & Milanova, M. (2018, April). The impact of multi-optimizers and data augmentation on TensorFlow convolutional neural network performance. In *2018 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)* (pp. 140-145). IEEE.

Tripathi, R., Alqahtani, S. S., Albarraq, A. A., Meraya, A. M., Tripathi, P., Banji, D., ... & Alnakhli, F. M. (2020). Awareness and preparedness of COVID-19 outbreak among healthcare workers and other residents of South-West Saudi Arabia: a cross-sectional survey. *Frontiers in public health*, *8*, 482.

Viola, P., & Jones, M. (2001, December). Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001* (Vol. 1, pp. I-I). Ieee.

Walia, I. S., Kumar, D., Sharma, K., Hemanth, J. D., & Popescu, D. E. (2021). An Integrated Approach for Monitoring Social Distancing and Face Mask Detection Using Stacked ResNet-50 and YOLOv5. *Electronics*, *10*(23), 2996.

Wang, C. Y., Bochkovskiy, A., & Liao, H. Y. M. (2022). YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *arXiv preprint arXiv:2207.02696*.

Xiao, J., Wang, J., Cao, S., & Li, B. (2020, April). Application of a novel and improved VGG-19 network in the detection of workers wearing masks. In *Journal of Physics: Conference Series* (Vol. 1518, No. 1, p. 012041). IOP Publishing.

Yang, W., & Jiachun, Z. (2018, July). Real-time face detection based on YOLO. In *2018 1st IEEE international conference on knowledge innovation and invention (ICKII)* (pp. 221-224). IEEE.

Yang, X., & Zhang, W. (2022). Heterogeneous face detection based on multi-task cascaded convolutional neural network. *IET Image Processing*, *16*(1), 207-215.