# FACE MASK RECOGNITION SYSTEM USING CNN

**2024**

## ABSTRACT

## FACE MASK RECOGNITION SYSTEM USING CNN

**S. M. Gunwardane**

This research paper delves into the challenges and solutions concerning face mask detection in real-world scenarios characterised by varying lighting conditions. The study aims to answer three fundamental research questions:

1. How do real-world scenarios, particularly diverse lighting conditions, impact the accuracy of face mask detection models trained on synthetic data?
2. What preprocessing techniques can be employed to enhance the accuracy of face mask detection models in real-world scenarios?
3. Among different transfer learning architectures, which yields the best face mask detection outcomes in real-world settings?

The research methodology experiment involved training classification models on synthetic datasets from celebrity faces and evaluating their performance on a controlled test dataset representing real-world lighting conditions (sunny, cloudy, shadow, indoor artificial light, and outdoor artificial light). The models were based on pre-trained architectures such as VGG-16, VGG-19, ResNet, DenseNet, and MobileNet, with experiments focusing on preprocessing techniques and transfer learning strategies.

The findings revealed significant challenges when transitioning from synthetic training data to real-world scenarios. Despite achieving high accuracy during training and validation, models struggled to perform well on the real-world test dataset, particularly under different lighting conditions. This emphasises the importance of addressing environmental factors in model training and deployment.

Preprocessing techniques such as sharpness enhancement and face elongation showed promise in improving model performance in real-world scenarios. However, further exploration of advanced preprocessing methods and data augmentation strategies is recommended to enhance robustness and adaptability.

Transfer learning architectures, particularly DenseNet and ResNet, demonstrated superior feature extraction capabilities even under adverse conditions. Fine-tuning transfer learning strategies and investigating ensemble methods could enhance model accuracy and generalisation.

Overall, this research contributes valuable insights and practical recommendations for improving face mask detection models in real-world scenarios, with implications for public health, safety, and AI.

# TABLE OF CONTENTS

List of Figures

<u>Figure No.</u> <u>Page No.</u>

## LIST OF ABBREVIATIONS

| | |
|---|---|
| Artificial Intelligence | AI |
| Computer Vision | CV |
| Convolutional Neural Network | CNN |
| Contextual Multi-Scale Region-based CNN | CMS-RCNN |
| Coronavirus Disease 2019 | COVID-19 |
| Deformable Part Model | DPM |
| frames per second | fps |
| Mean Squared Error | MSE |
| Rectified Linear Unit | ReLU |
| Single Shot MultiBox Detector | SSD |
| World Health Organization | WHO |

# CHAPTER 1

# INTRODUCTION

The outbreak of *Coronavirus Disease 2019* (COVID-19) has presented a global health crisis with severe implications for human life. In 2020, the *World Health Organization* (WHO) declared COVID-19 a pandemic, as it rapidly spread across 188 countries, infecting millions of people within a span of just six months [1]. The emergence of new virus mutations has further exacerbated the situation, demanding urgent and effective measures to combat the disaster. Among the various preventive measures, the proper wearing of protective face masks has been recognised as a scientific and accessible way to curb the transmission of COVID-19. Masks act as a barrier, filtering and blocking virus particles in the air, thereby reducing the risk of droplet and aerosol transmission during social interactions [2].

The enforcement of face mask usage in public spaces has posed significant challenges and risks for policymakers in mitigating the spread of COVID-19. Many countries have implemented laws mandating the use of face masks due to the exponential growth in cases and deaths. However, monitoring compliance with these regulations, particularly in large groups, has become increasingly complex. The detection of individuals not wearing face masks is a crucial aspect of this monitoring process. To address this issue, advanced *artificial intelligence* (AI) software tools have been integrated into surveillance cameras in the Paris Metro system in France [3]. Developed by the French startup DatakaLab [4], this software aims to produce anonymous statistical data rather than identifying or apprehending individuals without masks. The collected data can assist authorities in predicting potential outbreaks and implementing proactive measures against COVID-19.

One branch of AI which has witnessed significant advancements, primarily because of the rapid progress in deep learning techniques, is *computer vision* (CV). CV has several branches depending on the type of solution it is used to provide. For example, object classification includes facial expression recognition [5-7], Alzheimer's disease and mild cognitive impairment diagnosis [8], temporal object detection for collision avoidance systems, and pedestrian counting. Often, a series of algorithms work together in CV to build a solution. For example, an object detector's output is often a feeder to an object classification algorithm. It is because the former provides a region of interest around the objects. At the same time, the latter can classify objects. In the facial mask classification problem with around 70 percent of image being the background behind the person, we also need to build a solution using a pipeline. This is particularly important because the detector would identify the region of interest containing the human features and reduced background. At the same time, the face and mask classifier will use this as input to perform its classification further.

Within the domain of object detection, there exist various classical algorithms such as Faster RCNN [9], YOLOv3 [10], YOLOv4 [11], and various anchor-free algorithms, which can be broadly categorised into one-stage and two-stage methods. The algorithms' foundational components, known as the backbone networks, play a pivotal role in determining the accuracy and efficiency of object detection systems. Among the widely adopted backbone networks are VGG [12], ResNet [13], DenseNet [14], and MobileNet [15], each with its unique advantages and applications. ResNet has gained significant popularity due to its ability to overcome the degradation problem associated with training deep neural networks by employing residual blocks.

In the face and mask detection domain, existing algorithms can be classified into two primary categories: real-time detection methods, which prioritise rapid inference speed. And, high-performance detection methods, which emphasise accuracy [16-18]. These algorithms have seen extensive research, partly driven by the challenges posed by the COVID-19 pandemic. For instance, researchers have developed lightweight models, such as those utilising MobileNet and SSD architectures, achieving sub-millisecond face detection speeds in face mask detection applications. This level of speed renders them suitable for deployment on resource-constrained devices like mobile phones, catering to the evolving needs of our increasingly interconnected world [16,18-19].

Despite the impressive progress made by existing face and face mask detectors, it is crucial to acknowledge a common trend in the literature. Many papers in this domain often rely on subsets of images from the training set to serve as the testing data for model validation [16,18-19]. Although test images are novel to the model, their similarity to the training data in terms of data collection and preprocessing methods leads to inflated accuracy during testing. As a result, these studies often overlook real-world scenarios where images are acquired in diverse contexts beyond the training data distribution. Consequently, the accuracy of these detectors significantly diminishes in practical settings. This decline in performance can be attributed to various factors, including the quality of the training data and the variability in lighting conditions encountered in real-world image capture scenarios. In light of this, the following research questions have been formulated to bridge this gap:

**RQ1:** How does the detection accuracy of models get impacted by capturing images in real-world scenarios with varying lighting conditions outside the scope of the training data context?

**RQ2:** What preprocessing techniques can be employed to enhance detection accuracy in real-world scenarios characterised by diverse lighting conditions?

**RQ3:** Among the transfer learning architectures such as VGG-16, VGG-19, ResNet, DenseNet, and MobileNet, which architecture yields the best outcomes for scenarios encountered in real-world settings?

These research questions are the guiding principles that steer this study. To address these queries, the research endeavours to execute several steps, encompassing the design and training of transfer learning models using a public dataset, which is an artificially created dataset superimposing masks on celebrity images [20] for classifying face masks, creating a unique real-world dataset for testing classification models, optimising preprocessing steps to enhance accuracy with the MobileNet pre-trained network, and finally, comparing models based on their accuracy on the testing dataset. Consequently, these steps, which will be elaborated into research objectives, are closely connected to the discoveries that assist in resolving the research problems outlined. Therefore, the research objectives of this paper are as follows:

**RO1:** Develop an optimised facemask recognition model based on pre-trained networks such as VGG-16, VGG-19, ResNet, DenseNet, and MobileNet using a public dataset.

**RO2:** Create a novel real-world test dataset to assess the models created in RO1. The novel dataset comprises images with varying illumination conditions, such as sunny, cloudy, shadowy, and artificial light.

**RO1:** Employ experimental methodology to enhance the accuracy of the MobileNet model and identify the best preprocessing techniques for the novel dataset.

**RO2:** Conduct experiments to determine the best-pretrained network that yields higher accuracy with the novel real-world dataset.

The selection of pre-trained network architectures in this study is informed by insights from previous research experiences, aiming to find an optimal combination of architectures based on factors like network complexity, parameters, and model size. In RO3, the MobileNet model has been chosen to refine the preprocessing steps. This decision is influenced by MobileNet's performance, which has demonstrated higher accuracy and features a lightweight architecture.

## 1.1 THESIS ORGANISATION

This thesis is structured into five core chapters, each contributing significantly to the overall research process and discussion:

- Chapter 1: Introduction
- Chapter 2: Literature Review
- Chapter 3: Methodology
- Chapter 4: Results and Discussion
- Chapter 5: Conclusion

The introductory chapter introduces the reader to the research by offering a comprehensive overview. It delves into the historical background, elucidates contemporary developments, and emphasises the necessity for further exploration, particularly focusing on the creation of a novel test dataset under various lighting conditions such as sunny, cloudy, shadow, and artificial lighting. Moving on to chapter two, the literature review critically examines existing scholarly works within the research's context. It systematically synthesises diverse literature, encompassing facemask detection and recognition techniques. This chapter concludes by addressing the challenges encountered in detecting real-world images under varying lighting conditions. Chapter three, Methodology, details the application of the Experiment paradigm tailored to the research's specific context. It outlines the procedural sequence and the parameter choices, including both dependent and independent variables, made at each experiment stage. Chapter four consolidates the experiment findings and engages in in-depth discussions. It presents the research outcomes and conducts detailed discussions, aligning the findings with the primary research questions. Lastly, the conclusion chapter encapsulates the key findings, elucidates their broader implications, and extends recommendations for potential future work in the field, providing a comprehensive wrap-up to the research journey.

# CHAPTER 2

# LITERATURE REVIEW

# CHAPTER 3

# METHODOLOGY

The research methodology chapter delves into the experimental framework utilised in this study. It provides a detailed exposition of how the experiment is structured and conducted. Various variables are incorporated into the experiment to elucidate the characteristics of the model and preprocessing steps under investigation. These variables are instrumental in establishing cause-and-effect relationships, yielding reliable data for addressing the research question. Furthermore, this chapter comprehensively outlines the different stages of the research that were undertaken based on research objectives.

## 3.1 EXPERIMENT

This research adopts an experimental design approach to address the research question effectively. The experimental design allows the researcher to describe and predict outcomes and establish cause-and-effect relationships by manipulating variables and controlling the experimental environment [34]. The fundamental principle underlying the experimental approach is that it provides control over the research environment to ascertain potential cause-and-effect relationships among the variables under scrutiny. In this experimental process, specific factors are intentionally selected and varied in a controlled manner to observe their impact on the desired outcome [35]. These manipulated factors are known as independent variables.

This study centres on assessing model accuracy in real-world images under varying lighting conditions; the independent variables include pre-trained networks and preprocessing steps. These variables play a crucial role in evaluating the accuracy of multiple models, which in turn aids in addressing the research question. Conversely, the dependent variables stem from parameters used to assess the classification model, such as precision, recall, F1 score, response speed, and deployment size. These factors are essential for comparing the models in terms of their practical application.

The research involves analysing the performance of trained models using the novel test dataset. Detailed measurements are collected for each model, focusing on specific lighting conditions. Various preprocessing techniques are implemented to enhance classification accuracy across the novel test dataset. Finally, the performance of all models is compared, culminating in answers to the research question derived from reliable data obtained through a controlled experiment.

### 3.1.1 Experiment Design

The primary focus of this research is the evaluation of pre-trained models' accuracy using a real-world dataset influenced by various lighting conditions. The development of simple classification models involved utilising pre-trained networks for feature extraction and fully connected layers to learn weights specific to recognising masks or non-masked individuals. This streamlined approach to network design enables more evident observation of changes in the dependent variables.

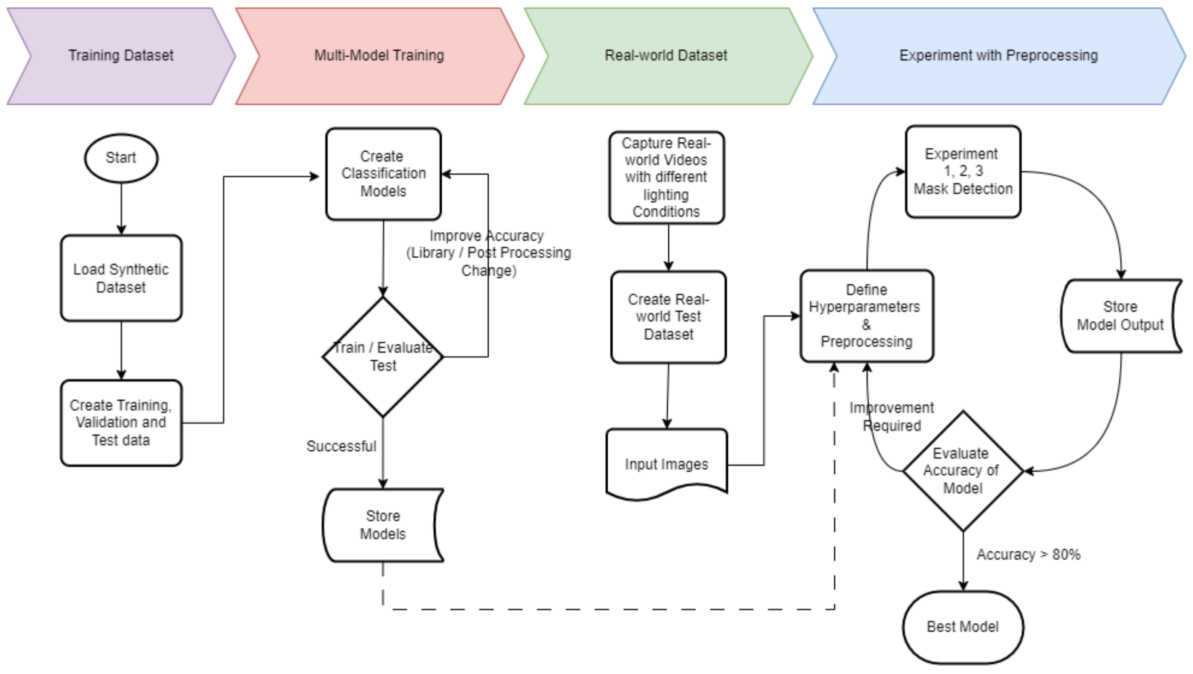

Training Dataset
Multi-Model Training
Real-world Dataset
Experiment with Preprocessing
Start
Load Synthetic Dataset
Create Training, Validation and Test data
Create Classification Models
Improve Accuracy (Library / Post Processing Change)
Train / Evaluate Test
Successful
Store Models
Capture Real-world Videos with different lighting Conditions
Create Real-world Test Dataset
Input Images
Define Hyperparameters & Preprocessing
Experiment 1, 2, 3 Mask Detection
Store Model Output
Improvement Required
Evaluate Accuracy of Model
Accuracy > 80%
Best Model

Figure 3-1: Research Phases

The research study progresses through distinct phases, as depicted in Figure **3-1**, each contributing crucial insights towards addressing the overarching research questions. The initial phase focuses on the Training dataset, where a pivotal task involves sourcing a public dataset closely resembling the pandemic scenario, characterised by challenges in acquiring authentic training data. Consequently, synthetic datasets are predominantly utilised. The subsequent phase entails training multiple models employing diverse pre-trained network architectures such as MobileNet, DenseNet, VGG, and ResNet through transfer learning. These models are trained using synthetic datasets and then evaluated using a real-world dataset. The real-world dataset is meticulously curated to encompass various lighting conditions encountered in everyday scenarios, including sunny, cloudy, shadowy, indoor natural, and artificial light conditions.

In the third phase, a real-world dataset is meticulously created by capturing video recordings of individuals wearing and not wearing masks under different lighting conditions. These videos serve as the basis for extracting images, thus ensuring a balanced dataset for subsequent evaluation. Each model's performance is rigorously assessed for accuracy in the evaluation phase. Notably, three preprocessing techniques are employed, primarily focusing on enhancing the accuracy of the MobileNet model. This experimentation with different preprocessing techniques aims to effectively provide insights into addressing research questions 1 and 2.

Two pivotal factors underpin the selection of MobileNet for these experiments. Firstly, MobileNet is purposefully designed as a *Convolutional Neural Network* (CNN) architecture optimised for efficient computation on mobile and lightweight devices. Secondly, compared to alternative models, MobileNet exhibits satisfactory performance in training and validation within the midrange. Its architecture prioritises computational efficiency, boasting a small memory footprint and low latency, making it an ideal candidate for deployment on resource-constrained devices like smartphones [36-37].

Lastly, through preprocessing techniques and comparative analysis of various model architectures, the study aims to answer research question 3, elucidating which architecture proves more adept at extracting features from the curated real-world dataset. This comparative analysis is instrumental in understanding the capabilities of different architectures in handling real-world data intricacies.

### 3.1.2 Experiment Variable

The experiment encompasses several variables crucial for understanding the accuracy of the models under evaluation. The upcoming sections will delve into a comprehensive discussion of these variables utilised in the experiment.

#### 3.1.2.1 Control Variables

In order to assess the effects of the new test dataset under various lighting conditions, this study employs a controlled experimental methodology. Control is maintained at two distinct phases of the experiment: during the design and training of the models and the creation of the novel dataset.

#### *3.1.2.1.1 Model design and training*

The design of the models in this study has been influenced by transfer learning, where pre-trained weights are incorporated and kept frozen during the network's learning phase. This approach ensures that all pre-trained networks retain their feature extraction capabilities acquired from being trained on the ImageNet dataset, a renowned benchmark dataset comprising more than 14 million images categorised into over 20,000 classes [38].

Additionally, an artificially created dataset has been chosen to mirror real-world conditions during a pandemic, where collecting or generating real images may be challenging. The training dataset utilised in this research is the "Facemask Detection Dataset 20,000 Images" curated by Jathin Pranav Singaraju and Lavik Jain, sourced from Kaggle [20]. This dataset encompasses two distinct classes: "with_mask" and "without_mask," each consisting of 10,000 images. Images for both classes were sourced from the CelebFace dataset on Kaggle, which features a collection of over 200,000 celebrity images for the "without_mask" class; additional preprocessing was conducted to filter out images containing face masks or other face coverings. Conversely, the "with_mask" images were synthetically generated by applying masks to faces using Python scripting.

#### *3.1.2.1.2 Novel test dataset*

The new test dataset was crafted to replicate a range of lighting scenarios commonly encountered in daily life, including conditions like cloudy weather, direct sunlight, shadows, indoor environments with natural lighting, and indoor spaces illuminated with artificial light. This diverse set of lighting conditions was chosen to evaluate the model's adaptability to various real-world situations.

Specific guidelines were followed during the dataset creation process to ensure consistency and control during the testing phase. Firstly, the same individual was used across all images to eliminate variations that might arise from different people. Additionally, each image featured only one person to maintain clarity and prevent potential confusion caused by multiple individuals in a single frame. Surgical masks were selected for inclusion in this dataset to simplify the task and focus on detecting the presence of masks rather than dealing with the complexities of various mask types.

In subsequent sections 3.1.3.1, 3.1.3.2 and 3.1.3.3, further details will be provided regarding the training dataset, model design, and the intricacies of the novel test dataset.

#### 3.1.2.2 Independent Variables

In addressing the core research question, it is paramount to explore how the introduction of a novel dataset featuring diverse lighting conditions impacts classification models. The role of lighting conditions in the image detection process is pivotal, as it directly influences the accuracy and reliability of model outcomes. Illumination, which encompasses the overall brightness of a scene, holds substantial sway over image detection. Detection algorithms tend to excel by capturing detailed, clear information from images in well-illuminated settings characterised by uniform lighting. Conversely, environments with low-light conditions or uneven illumination can pose challenges like noise, loss of detail, and reduced contrast. These factors can impede detection algorithms from accurately identifying and categorising objects or features within images.

Furthermore, the presence of shadows introduces another layer of complexity to lighting conditions, impacting the image detection process significantly. Shadows can distort object appearances, create misleading shapes or patterns, and introduce variations in contrast and colour. Such effects can present hurdles for detection algorithms as shadows may obscure or alter the features crucial for identification and recognition. Accurately detecting objects amidst shadows necessitates robust algorithms capable of discerning actual objects from shadow artefacts.

This research undertakes the development of a novel dataset that encompasses a comprehensive array of lighting conditions encountered in real-world settings. It is imperative to encompass diverse settings, including environments characterised by cloudy conditions, which produce diffused and soft lighting, and sunny conditions featuring bright and direct sunlight. Additionally, shadowy areas with uneven and partially obstructed illumination, indoor spaces receiving natural light through windows, and indoor environments illuminated with artificial light sources are considered. Each lighting condition presents distinct challenges and variations that can significantly influence the performance of image detection algorithms.

Recognising that successful classification by deep learning models hinges on two key factors—the quality of the input image received by the model and the model's ability to extract features from that image—it is crucial to address these factors. As previously discussed, lighting conditions profoundly influence the first factor, image quality. Therefore, preprocessing the image before it is fed into the model becomes paramount to mitigate the adverse effects of lighting conditions, such as overexposure or underexposure, leading to loss of features. Consequently, preprocessing steps emerge as the primary variable of interest under the independent variables in this research, aiming to tackle the research question effectively. The second factor pertains to the complexity embedded in the model’s architecture, which determines its feature extraction capabilities. Hence, this research compares several pre-trained architectures to analyse the variability introduced by network architecture.

##### *3.1.2.2.1 Preprocessing techniques*

Preprocessing techniques play a pivotal role in improving the accuracy and effectiveness of deep learning models, particularly CNNs. These techniques involve manipulating and enhancing images before they are fed into the model, thereby optimising feature extraction and improving prediction accuracy. One key aspect is how preprocessing can significantly enhance feature extraction in CNNs. By applying preprocessing techniques, such as histogram equalisation, contrast adjustment, brightness enhancement, and pixel intensity manipulation, the input images undergo transformations that amplify relevant features and suppress noise or irrelevant information. These transformations ensure that the CNN can focus on extracting meaningful patterns and structures from the images, leading to more accurate predictions.

Histogram equalisation is a technique used to enhance the contrast of an image by redistributing pixel intensities, making the image clearer and more visually appealing. Contrast adjustment involves altering the range of pixel intensities to improve the distinction between different objects or regions within an image. Brightness enhancement increases the overall brightness of an image, making it easier for the model to identify objects or features in low-light conditions. Pixel intensity manipulation involves adjusting the values of individual pixels to highlight specific details or characteristics in the image. Additionally, techniques like sharpening and resizing can positively impact prediction accuracy by refining edges and details in the images and ensuring uniformity in image dimensions for consistent processing by the model.

In this research, three experiments are conducted utilising cropping, grayscale conversion, sharpening, and resizing techniques as part of preprocessing. These techniques are further elaborated in sections 3.1.3.4.2, 3.1.3.4.3, where each experiment is described in detail, including the specific preprocessing steps applied and their impact on the model's performance.

Overall, these preprocessing techniques serve as independent variables in the experiment, allowing for a comprehensive analysis of their effectiveness in improving feature extraction and prediction accuracy in CNNs.

##### *3.1.2.2.2 Network architecture and transfer learning*

In the realm of feature extraction models for image recognition and classification tasks, pre-trained models such as VGG16, VGG19, ResNet101V2, DenseNet201, and MobileNetV2 stand out as excellent choices for comparison and evaluation. Each of these models brings unique strengths and characteristics to the table, making them suitable candidates for diverse experimental setups.

Starting with VGG16 and VGG19, these models are renowned for their deep architectures comprising multiple convolutional layers. They excel in capturing intricate features and patterns within images, making them ideal for tasks requiring detailed analysis and discrimination between complex visual elements. ResNet101V2, on the other hand, introduces residual connections that alleviate the vanishing gradient problem, enhancing the model's training efficiency and overall performance. This makes ResNet101V2 a robust choice for tasks demanding high accuracy and stability.

DenseNet201 adopts a dense connectivity pattern, where each layer is directly connected to every other layer within a dense block. This architecture fosters feature reuse and facilitates gradient flow, leading to strong feature representations and improved model generalisation. Lastly, MobileNetV2 is known for its lightweight design and efficient utilisation of computational resources, making it suitable for deployment in resource-constrained environments such as mobile devices or edge computing platforms.

In an experimental research context, these pre-trained models serve as independent variables for comparison and evaluation. They represent different approaches to feature extraction and representation learning, each with its strengths and weaknesses. By including these models as independent variables, researchers can systematically assess and compare their performance across various metrics such as accuracy, speed, and resource utilisation.

#### 3.1.2.3 Dependent Variables

In experimental research, dependent variables serve as the benchmarks for comparison, as they can be quantitatively measured and compared empirically, thereby aiding in evaluating the models' performance. The selection of evaluation metrics depends on the specific characteristics of the task at hand. Standard metrics include accuracy, precision, recall, and F1 score for classification tasks, while regression tasks often utilise *Mean Squared Error* (MSE). Precision measures the ratio of true positives to all positive detections, while recall gauges the ratio of true positives to all actual positive instances. The F1 score, acting as a comprehensive indicator of algorithmic performance, represents the harmonic mean of recall and precision. An essential tool in this evaluation is the confusion matrix, which offers a detailed breakdown of a model's performance, enabling the computation of precision, recall, and F1 score [39] The chosen classification task metrics for evaluating model performance are elaborated as follows.

$$Accuracy \ = \frac{Tp + Tn}{(Tp + Fp + Fn + Tn)}$$

$$Precision \ = \frac{Tp}{(Tp + Fp)}$$

$$Recall \ = \frac{Tp}{(Tp + Fn)}$$

$$f1 \ score \ = 2 * \frac{Recall * Precision}{(Recall + Precision)}$$

Where *Tp = True positive, Tn = True negative, Fp = False positive* and *Fn = False negative*.

Furthermore, the experimental trials were conducted in the Google Colab environment, leveraging GPU acceleration for model training to expedite the process. However, the environment was configured without GPU utilisation during the testing and validation phases. This setup was maintained to ensure consistent resource allocation across all tests, aiming to measure each model's prediction time reliably. In addition to prediction time, the model's size was measured, which was directly influenced by the number of parameters or weights defined within the model. This metric offers insights into the model's complexity and deployability, helping assess its weightiness and computational requirements.

### 3.1.3 Experiment Stages

After finalising the experiment design and determining all variables, the subsequent phase of the research involves implementation. This implementation phase encompasses three distinct stages: coding, deployment, and execution. As mentioned earlier in the discussion on dependent variables, Google Colab serves as the primary environment for deploying, training, and testing the models. Integrated with Colab, Google Drive acts as the storage repository for datasets, trained models, and predicted outcomes. The experimental trials were executed within the Google Colab environment. Initially, model training was carried out using GPU acceleration to accelerate the training process. However, GPU usage was turned off during the subsequent test and validation phases to maintain consistent resource allocation across all tests. This standardisation ensured that the resources allocated remained uniform throughout the experiments, contributing to reliable and comparable results. The remaining sections of this chapter delve into the coding stage, providing a detailed exploration of each aspect of the implementation process.

#### 3.1.3.1 Training dataset

A synthetic dataset was chosen to mimic real-world conditions for training. The training data employed in this research is the "Facemask Detection Dataset 20,000 Images" sourced from Kaggle, created by Jathin Pranav Singaraju and Lavik Jain [20]. This dataset comprises two distinct classes: "with_mask" and "without_mask," each containing 10,000 images. The "without_mask" class is derived from the CelebFaces dataset on Kaggle, while the "with_mask" class was generated using a script that superimposed mask images onto faces. This dataset selection is reflective of the challenges posed during a pandemic period, where obtaining specific training datasets may be limited, necessitating the generation of synthetic data for training purposes. Figure **3-2** visually represents the two categories: masked and unmasked faces.

The training dataset was meticulously organised into subfolders for training, testing, and validation, encompassing 14,000 images for training, 4,000 for validation, and 2,000 for testing. Within these groups, subfolders were created, each designated with the category names “With_Mask” and “Without_Mask”, delineating the distinct classes of images. All images in the dataset were standardised to a resolution of 256 × 256 pixels. The dataset was effectively loaded and managed using TensorFlow and Keras libraries, renowned for their comprehensive data loading and manipulation functionalities. As a fundamental aspect of this research, both the training and validation datasets underwent augmentation using the ‘***ImageDataGenerator***’ function during loading.

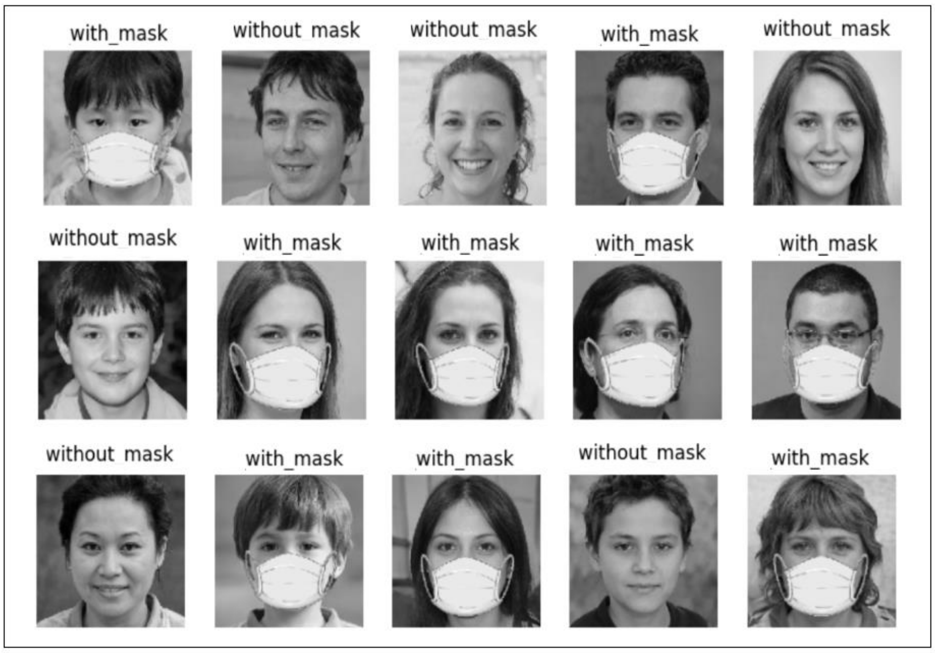
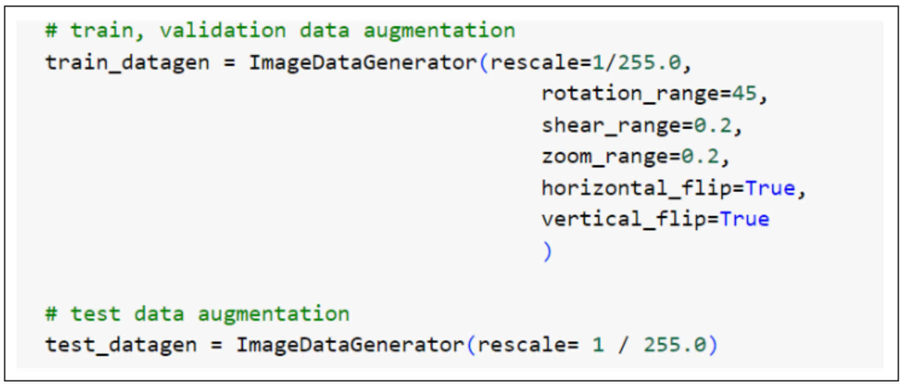


Figure 3-2: Training Dataset

The '***ImageDataGenerator***' function holds significant importance in deep learning, especially when dealing with image datasets. Its primary function revolves around augmenting and preprocessing images, enhancing dataset diversity and robustness, which benefits model training. Among its critical parameters is rescale, which standardises pixel values within images, typically transforming them into a range between 0 and 1. This normalisation procedure stabilises the learning process and facilitates model convergence during training, ensuring that input data remains consistent within a defined numerical range.

Another pivotal parameter of the ImageDataGenerator function is rotation_range, dictating the range of random rotations that can be applied to images. This parameter introduces variability in image orientation, aiding the model in developing rotational invariance and enhancing its generalisation capability. Similarly, the shear_range parameter controls the level of shearing applied to images, introducing diverse perspectives and augmenting the dataset with varying viewpoints, ultimately bolstering the model's resilience against distortions.

The zoom_range parameter significantly determines the extent of random zooming that can be applied to images. This zooming functionality alters the scale of objects within images, simulating variations in perspective and enriching the dataset with a spectrum of scales. Additionally, the horizontal_flip parameter allows for random horizontal flipping of images, introducing variations in spatial orientation and improving the model's adaptability to different scenarios.



```
# train, validation data augmentation
train_datagen = ImageDataGenerator(rescale=1/255.0,
                                   rotation_range=45,
                                   shear_range=0.2,
                                   zoom_range=0.2,
                                   horizontal_flip=True,
                                   vertical_flip=True
                                   )

# test data augmentation
test_datagen = ImageDataGenerator(rescale= 1 / 255.0)
```

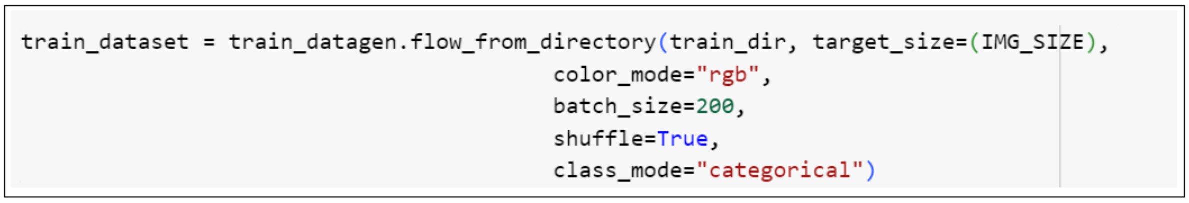
Figure 3-3: Training data augmentation

Similarly, the vertical_flip parameter contributes to augmenting image orientation by randomly mirroring images along the horizontal axis. These augmentation strategies provided by the '***ImageDataGenerator***' function significantly enhance dataset diversity, preparing the data for training models that exhibit robustness to spatial transformations and variations commonly encountered in real-world applications. Figure **3-3** visually depicts the implementation of training data augmentation techniques, showcasing the diverse transformations applied to augment the dataset effectively.

This augmentation strategy exposes the model to diverse transformations, aiding in acquiring robust features and patterns that remain consistent despite variations. However, applying augmentation to the test dataset risks data leakage. The test dataset should represent unseen, real-world scenarios, and augmenting it may cause the model to recognise augmented patterns instead of genuine features, potentially distorting evaluation outcomes.

Additionally, assessing the model's generalisation ability is paramount during testing. The evaluation focuses on the model's capability to make precise predictions on unseen, unaltered data by maintaining the test images unaltered. This approach offers a practical assessment of the model's performance in real-world scenarios outside the training context. Furthermore, preserving test images in their original state ensures a fair and impartial comparison. Utilising augmented test data could misrepresent the model's actual performance on real-world images, leading to erroneous conclusions about its capabilities.

Hence, the decision to apply augmentation solely to training and validation images while leaving test images unchanged. This strategic approach aligns with industry best practices in evaluating deep learning models, striving for dependable and authentic results that precisely reflect the model's effectiveness in practical applications.



```
train_dataset = train_datagen.flow_from_directory(train_dir, target_size=(IMG_SIZE),
                                                  color_mode="rgb",
                                                  batch_size=200,
                                                  shuffle=True,
                                                  class_mode="categorical")
```

Figure 3-4: Training data load

Figure **3-4** visually illustrates the implementation of data loading functionality. Organising training dataset images into subfolders facilitates their management and utilisation in deep learning workflows. The '***flow_from_directory***' function within TensorFlow's '***ImageDataGenerator***' module is instrumental in efficiently loading image data directly from directories, simplifying the process of dataset organisation and accessibility. In this specific implementation, several crucial parameters are employed to tailor the loading and preprocessing of image data.

Firstly, the '***train_dir***' parameter delineates the directory path housing the training images to be loaded. This directory structure typically includes subfolders representing various image classes or categories, such as “masked” and “without_masked”, ensuring clear categorisation. The '***target_size***' parameter defines the dimensions to which the loaded images will be resized. Specifying target_size=(256 × 256) indicates the transformation of images into a square format with dimensions of 256 × 256 pixels, facilitating uniformity in image processing. The '***color_mode***' parameter governs the colour representation of loaded images. Opting for color_mode="rgb" denotes the loading of images in RGB colour space, a standard in computer vision tasks. This decision aligns with the use of pre-trained architectures suitable for RGB colour representations. The '***batch_size***' parameter dictates the number of images processed in each training or validation batch. For instance, with a batch size of 200, the model processes 200 images concurrently, enhancing computational efficiency during training iterations. The '***shuffle***' parameter influences the randomisation of image order before feeding them into the model. Enabling shuffle=True prevents the model from learning sequential patterns in the data, promoting better generalisation and robustness. Lastly, the '***class_mode***' parameter specifies the label encoding method used for loaded images. Setting class_mode= “categorical” signifies that image labels correspond to distinct categorical classes or categories, enabling accurate classification during model training.

### 3.1.3.2 Classification Models

This study focused on feature extraction and image recognition for classification tasks; several pre-trained models were employed: VGG16, VGG19, ResNet101V2, DenseNet201, and MobileNetV2. These models were chosen for their robustness and efficacy in extracting features, making them ideal candidates for comparison and evaluation in various experimental setups. Notably, these models have undergone pre-training on the ImageNet dataset, renowned in image classification with its extensive collection of over 14 million images spanning over 20,000 categories [38].

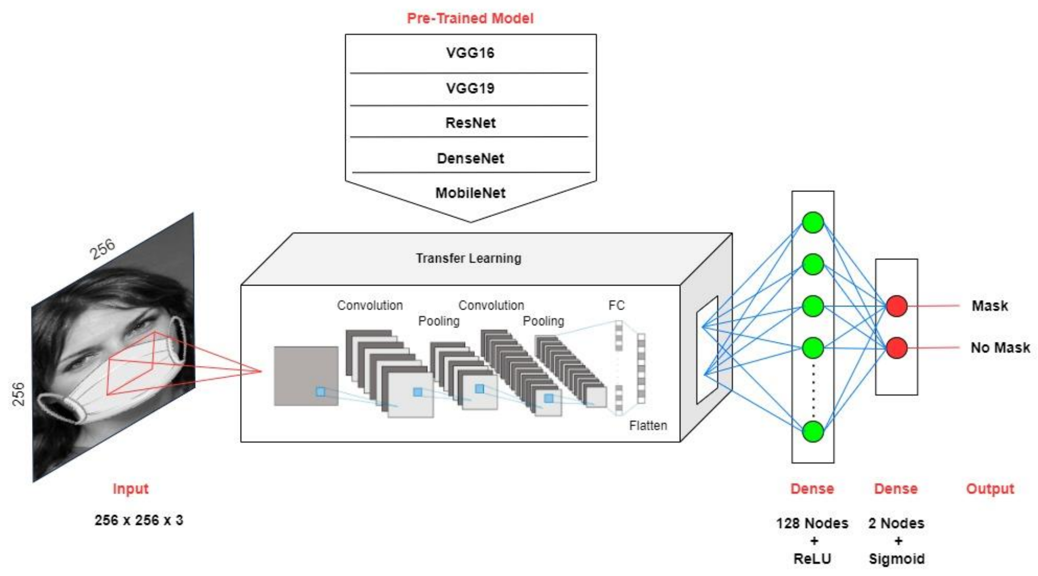
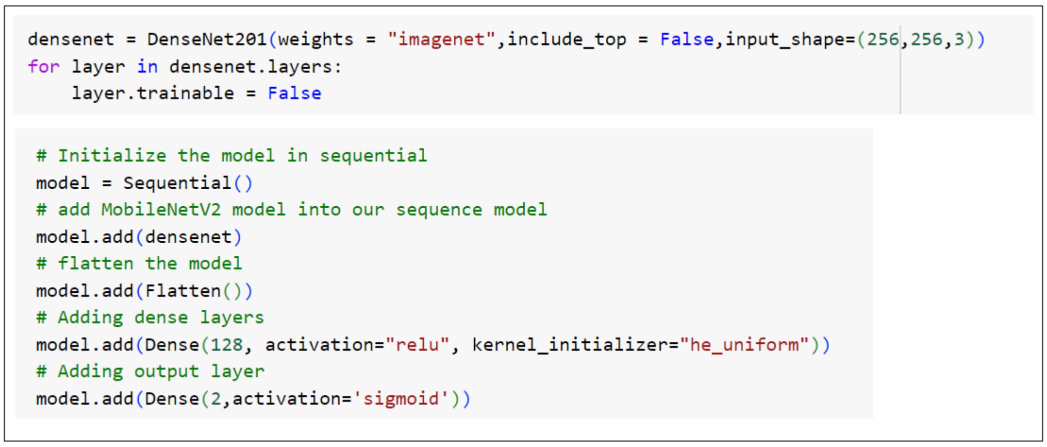


Figure 3-5: Model architecture

The methodology employed in this research leverages transfer learning, harnessing the pre-trained weights of CNN models trained on ImageNet. These weights are kept frozen during training to ensure that the feature extraction layer retains its learned capabilities from the vast ImageNet dataset without specific adaptation to the masked and without-masked dataset. Frozen feature extraction networks enable answering research questions related to pre-trained networks. To adapt these pre-trained models for the classification task, a flattened layer is introduced to transform the output of the convolutional layers into a one-dimensional vector. This vector is then connected to a dense layer comprising 128 neurons with *Rectified Linear Unit* (ReLU) activation, tasked with learning the classification weights based on the extracted features from the CNN layers. Additionally, a final dense layer with two neurons represents the two classes, masked and without masked images, employing the Sigmoid activation function to yield probability values for each class, reflecting the network's confidence in its predictions. Figure **3-5** visually illustrates the high-level architecture of the model, showcasing the utilisation of the mentioned pre-trained models individually to train distinct models capable of predicting whether a person is wearing a mask or not.

Figure **3-6** illustrates the implementation of the DenseNet201 pre-trained CNN architecture. The initial step involves loading the DenseNet201 architecture along with its ImageNet weights, after which all layers are frozen to ensure that the weights remain unchanged throughout the training process. Subsequently, the Keras sequential model creation step is executed, delineating the unique complexities in feature extraction design introduced by different architectures. This is notably reflected in the parameters introduced by each network.



```
densenet = DenseNet201(weights = "imagenet",include_top = False,input_shape=(256,256,3))
for layer in densenet.layers:
    layer.trainable = False
```

```
# Initialize the model in sequential
model = Sequential()
# add MobileNetV2 model into our sequence model
model.add(densenet)
# flatten the model
model.add(Flatten())
# Adding dense layers
model.add(Dense(128, activation="relu", kernel_initializer="he_uniform"))
# Adding output layer
model.add(Dense(2,activation='sigmoid'))
```

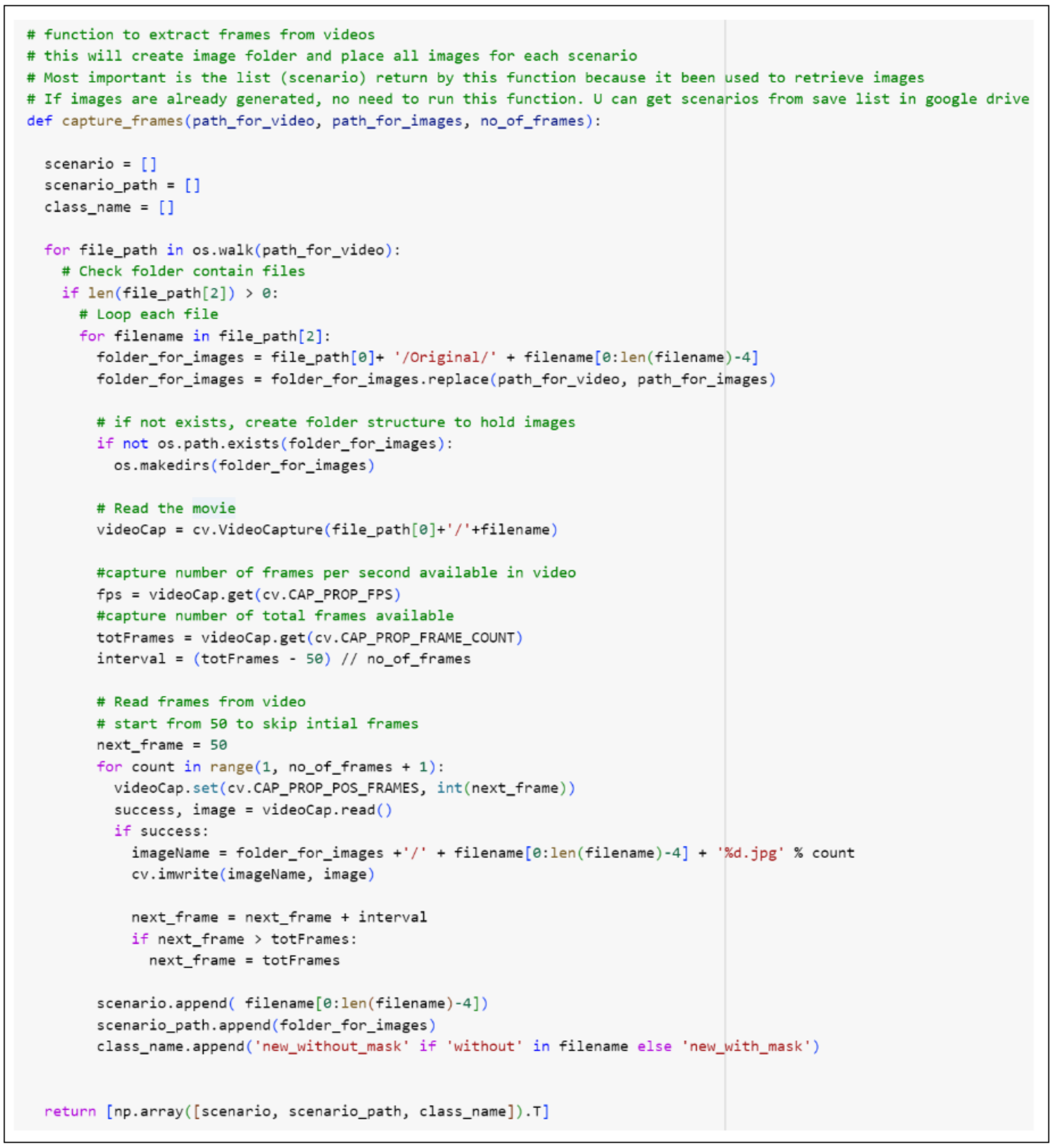
Figure 3-6: Model implementation

For instance, the DenseNet201 model designed for facemask classification comprises a total of 34,051,010 parameters, out of which 15,729,026 are trainable. In contrast, MobileNetV2 is characterised by 12,744,130 total parameters, with 10,486,146 being trainable, making it a lightweight network suitable for deployment on mobile devices. On the other hand, ResNet101V2 features 59,404,162 total parameters, with 16,777,602 being trainable, VGG19 has 24,219,074 total parameters and 4,194,690 trainable parameters, while VGG16 comprises 18,909,378 total parameters, with 4,194,690 being trainable. The variation in trainable parameters among these models stems from differences in their CNN output shapes despite having a fixed number of neurons (128) in the subsequent fully connected layer.

During training, the models are optimised using the Adam optimiser and trained with the Categorical Cross Entropy loss function, with accuracy as the evaluation metric. Each model undergoes training for ten epochs, allowing for comprehensive learning and refinement of the classification tasks.

### 3.1.3.3 Real-world test dataset

A comprehensive test dataset was meticulously curated to validate the hypothesis and assess the practical performance of the face mask detection model. This dataset aimed to emulate a wide range of lighting conditions typically encountered in everyday environments, encompassing scenarios such as cloudy weather, direct sunlight, shadowed areas, indoor spaces with natural lighting, and environments illuminated by artificial light sources. The objective was to evaluate the model's efficacy in handling diverse lighting conditions, thus gauging its adaptability to real-world scenarios.



```python
# function to extract frames from videos
# this will create image folder and place all images for each scenario
# Most important is the list (scenario) return by this function because it been used to retrieve images
# If images are already generated, no need to run this function. U can get scenarios from save list in google drive
def capture_frames(path_for_video, path_for_images, no_of_frames):

  scenario = []
  scenario_path = []
  class_name = []

  for file_path in os.walk(path_for_video):
    # Check folder contain files
    if len(file_path[2]) > 0:
      # Loop each file
      for filename in file_path[2]:
        folder_for_images = file_path[0]+ '/Original/' + filename[0:len(filename)-4]
        folder_for_images = folder_for_images.replace(path_for_video, path_for_images)

        # if not exists, create folder structure to hold images
        if not os.path.exists(folder_for_images):
          os.makedirs(folder_for_images)

        # Read the movie
        videoCap = cv.VideoCapture(file_path[0]+'/'+filename)

        #capture number of frames per second available in video
        fps = videoCap.get(cv.CAP_PROP_FPS)
        #capture number of total frames available
        totFrames = videoCap.get(cv.CAP_PROP_FRAME_COUNT)
        interval = (totFrames - 50) // no_of_frames

        # Read frames from video
        # start from 50 to skip intial frames
        next_frame = 50
        for count in range(1, no_of_frames + 1):
          videoCap.set(cv.CAP_PROP_POS_FRAMES, int(next_frame))
          success, image = videoCap.read()
          if success:
            imageName = folder_for_images +'/' + filename[0:len(filename)-4] + '%d.jpg' % count
            cv.imwrite(imageName, image)

            next_frame = next_frame + interval
            if next_frame > totFrames:
              next_frame = totFrames

        scenario.append( filename[0:len(filename)-4])
        scenario_path.append(folder_for_images)
        class_name.append('new_without_mask' if 'without' in filename else 'new_with_mask')


  return [np.array([scenario, scenario_path, class_name]).T]
```

Figure 3-7: Implementation of capture video frames

Creating this test dataset involved capturing ten-second videos under each specified lighting condition. Subsequently, 1000 images were extracted from these videos to construct the test dataset. Careful attention was paid to balancing the dataset, ensuring an equal distribution of masked and unmasked images, each class comprising 500 images. Moreover, a uniform distribution of 200 images was maintained within each lighting category, guaranteeing equitable representation across various lighting scenarios.

Figure **3-7** provides a visual representation of the implementation of the image extraction function, known as '***capture_frames***', which is designed to extract images from a video based on a specified frame ratio. The function requires inputs such as the file path of the video, the destination folder for the extracted images, and the desired number of frames to be extracted. The function outputs these extracted images into the designated image folder based on the specified frame ratio.

Within the '***capture_frames***' function, the *cv.VideoCapture()* function is utilised to initialise a video capture object, granting access to the specified video file located at the given path provided as part of the input parameters. Subsequently, the script employs various OpenCV functions to extract crucial information about the video file. For instance, *videoCap.get(cv.CAP_PROP_FPS)* retrieves the video's *frames per second* (fps), offering insights into its temporal characteristics. Similarly, *videoCap.get (cv.CAP_PROP_FRAME_COUNT)* helps determine the total number of frames in the video, facilitating frame-by-frame analysis.

This information is utilised to calculate the interval between frames based on the desired number of frames to be extracted, ensuring that the output matches the input request for the number of images. The script then employs a loop structure to iterate through the specified number of frames, setting the video frame position using *videoCap.set (cv.CAP_PROP_POS_FRAMES, int(next_frame))*, and subsequently read the corresponding frame using *videoCap.read()*. This iterative process enables the systematic and sequential extraction of frames from the video file according to the specified frame ratio.

Figure **3-8** presents a visual representation of a sample from the extracted test dataset, showcasing the diversity and balance achieved within the dataset. This meticulously curated test dataset plays a pivotal role in evaluating the model's real-world performance, comprehensively evaluating its accuracy and robustness across various lighting conditions.

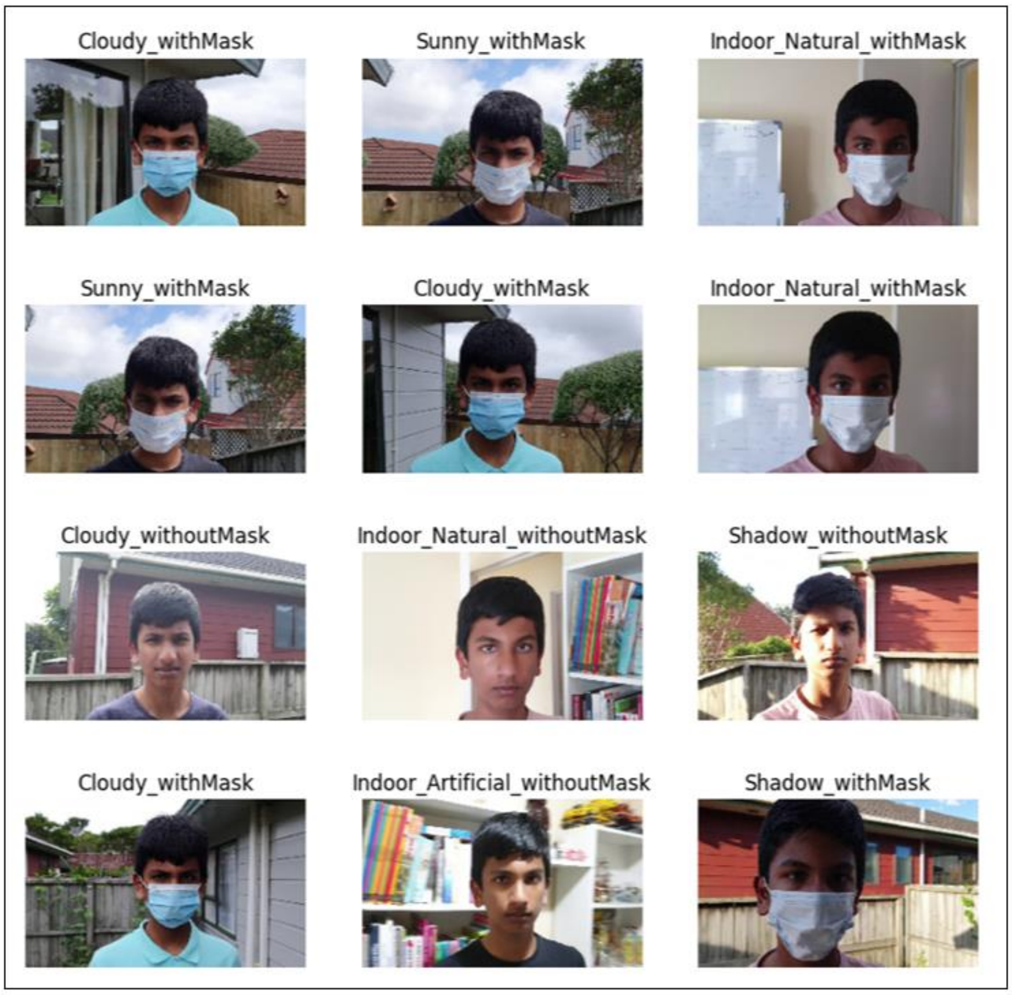


Figure 3-8: Real-world test dataset.

#### 3.1.3.4 Experiments

Figure **3-9** visually represents the three experiments conducted using real-world test images that were constructed for evaluation. Each experiment in the series is associated with specific preprocessing steps, as illustrated in the figure. After applying these preprocessing steps, the resulting data is inputted into the trained classification model, and the performance metrics, including overall accuracy and behaviour under varying lighting conditions, are assessed.

Experiment 1 focuses on achieving an output similar to the training dataset, which primarily comprises grayscale portrait images of individuals. The *Single Shot MultiBox Detector* (SSD) convolutional neural network is employed to localise and identify the person's face within the image. Subsequently, the identified face undergoes cropping and conversion to grayscale, aiming to closely resemble the samples present in the training dataset. Moreover, this technique can be extended to detect and process multiple individuals if they are present in the captured image.

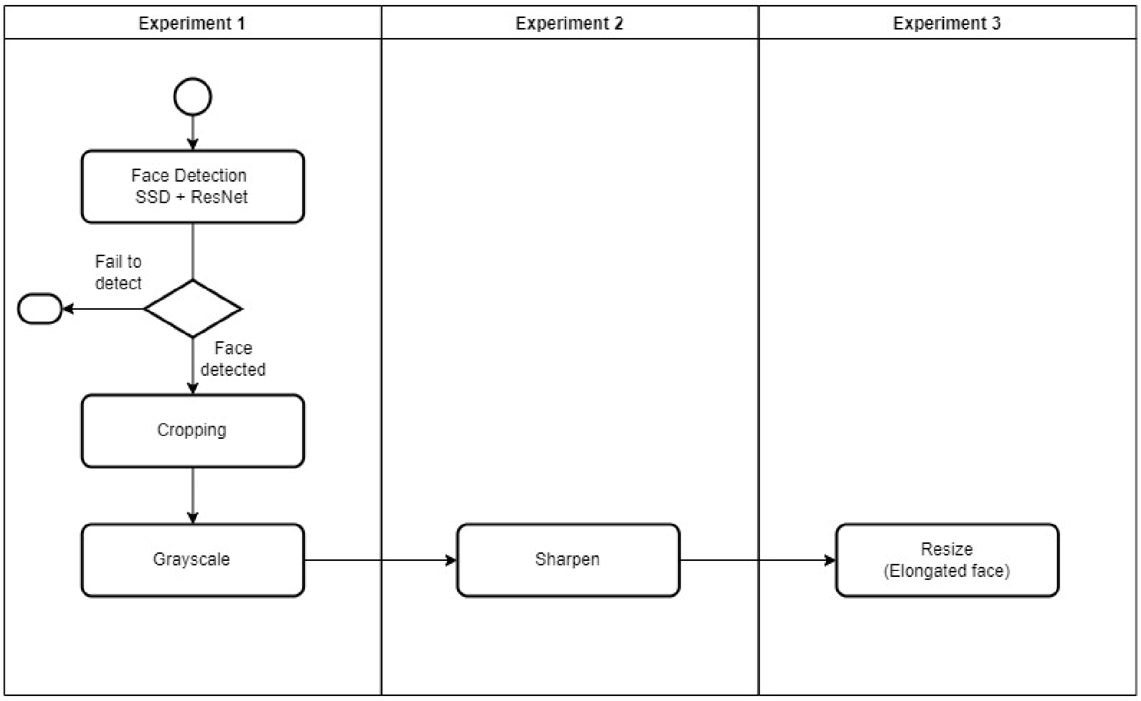
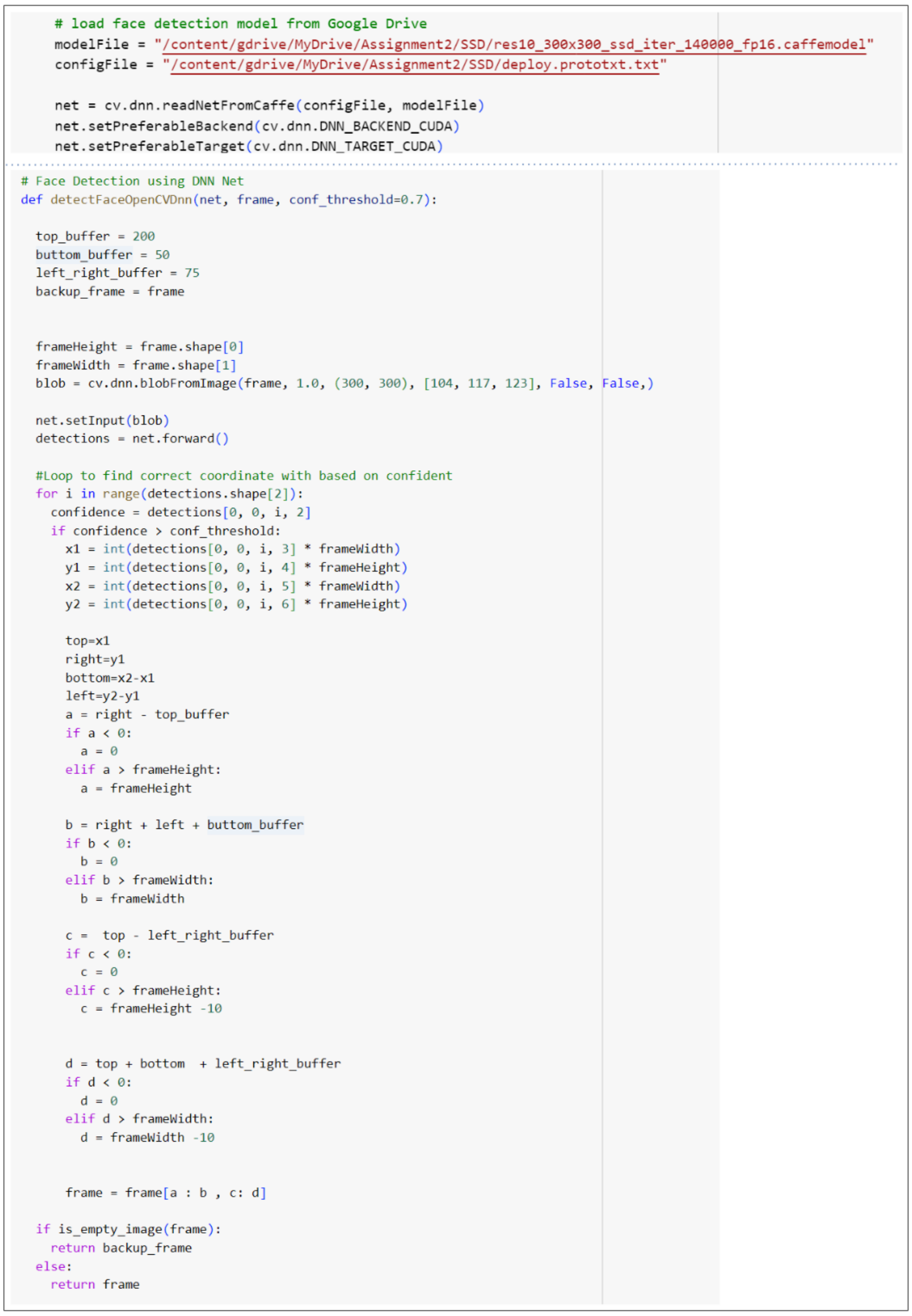


Figure 3-9: Experiments

The preprocessing techniques utilised in Experiments, including cropping, grayscale conversion, sharpening, and resizing, are extensively discussed in this section. However, it is important to note that these are not the only techniques explored within this research. Additional techniques such as histogram equalisation, contrast adjustment, brightness enhancement, and pixel intensity manipulation were also tested on the real-world dataset. However, the results indicated that these techniques did not significantly improve the accuracy compared to the sharpening and resizing effects. Therefore, for the experiments, the focus was placed on implementing sharpening and resizing techniques due to their more substantial impact on accuracy improvement.

##### *3.1.3.4.1 SSD for face detection*

The SSD model is a deep learning architecture designed explicitly for image object detection. It excels in real-time object detection, providing fast and accurate identification of objects without compromising accuracy. This experiment captured videos in a controlled environment to ensure that only one person was present in each frame. The objective of cropping the images was to isolate the person's face and exclude other environmental objects, making the images more closely aligned with the trained images. To achieve this, the "res10_300x300_ssd_iter_140000_fp16.caffemodel" was a designed pre-trained model for detecting faces in images. This model is based on a CNN architecture that consists of multiple layers of convolutional, pooling, and activation functions.



```python
# load face detection model from Google Drive
modelFile = "/content/gdrive/MyDrive/Assignment2/SSD/res10_300x300_ssd_iter_140000_fp16.caffemodel"
configFile = "/content/gdrive/MyDrive/Assignment2/SSD/deploy.prototxt.txt"

net = cv.dnn.readNetFromCaffe(configFile, modelFile)
net.setPreferableBackend(cv.dnn.DNN_BACKEND_CUDA)
net.setPreferableTarget(cv.dnn.DNN_TARGET_CUDA)
```

```python
# Face Detection using DNN Net
def detectFaceOpenCVDnn(net, frame, conf_threshold=0.7):

  top_buffer = 200
  buttom_buffer = 50
  left_right_buffer = 75
  backup_frame = frame


  frameHeight = frame.shape[0]
  frameWidth = frame.shape[1]
  blob = cv.dnn.blobFromImage(frame, 1.0, (300, 300), [104, 117, 123], False, False,)

  net.setInput(blob)
  detections = net.forward()

  #Loop to find correct coordinate with based on confident
  for i in range(detections.shape[2]):
    confidence = detections[0, 0, i, 2]
    if confidence > conf_threshold:
      x1 = int(detections[0, 0, i, 3] * frameWidth)
      y1 = int(detections[0, 0, i, 4] * frameHeight)
      x2 = int(detections[0, 0, i, 5] * frameWidth)
      y2 = int(detections[0, 0, i, 6] * frameHeight)

      top=x1
      right=y1
      bottom=x2-x1
      left=y2-y1
      a = right - top_buffer
      if a < 0:
        a = 0
      elif a > frameHeight:
        a = frameHeight

      b = right + left + buttom_buffer
      if b < 0:
        b = 0
      elif b > frameWidth:
        b = frameWidth

      c =  top - left_right_buffer
      if c < 0:
        c = 0
      elif c > frameHeight:
        c = frameHeight -10


      d = top + bottom  + left_right_buffer
      if d < 0:
        d = 0
      elif d > frameWidth:
        d = frameWidth -10


      frame = frame[a : b , c: d]

  if is_empty_image(frame):
    return backup_frame
  else:
    return frame
```

Figure 3-10: Implementation of SSD face detection

When an input image is provided to the CNN SSD model, it simultaneously analyses the image at multiple scales and locations using a set of predefined anchor boxes. These anchor boxes define potential locations and scales of objects within the image. The model then predicts the probability of each anchor box containing a face and refines the bounding box coordinates to accurately fit the detected face. In this case, the input image has dimensions of 1280 x 720 pixels. To process the image, the model resizes it to 300 x 300 pixels. Subsequently, the model applies its detection algorithm to the resized image and generates a set of bounding boxes around the detected faces and a confidence score for each box. Figure **3-10** visualises the implementation of the SSD face detection model.

The confidence threshold (‘***conf_threshold***’) plays a crucial role in object detection algorithms, as it determines the minimum confidence score required for a detection to be considered valid. In this experiment, a confidence threshold of 0.7 is used, meaning only detections with a confidence score of 0.7 or higher will be considered valid. Research conducted by Ke et al. [40] has shown that a threshold of 0.7 performs well for detecting faces at a closer proximity. Finally, the detected face regions are cropped to 430 x 600, and grayscale is applied for further processing. Figure **3-11** visually depicts the performance of the SSD model both with and without a face mask.

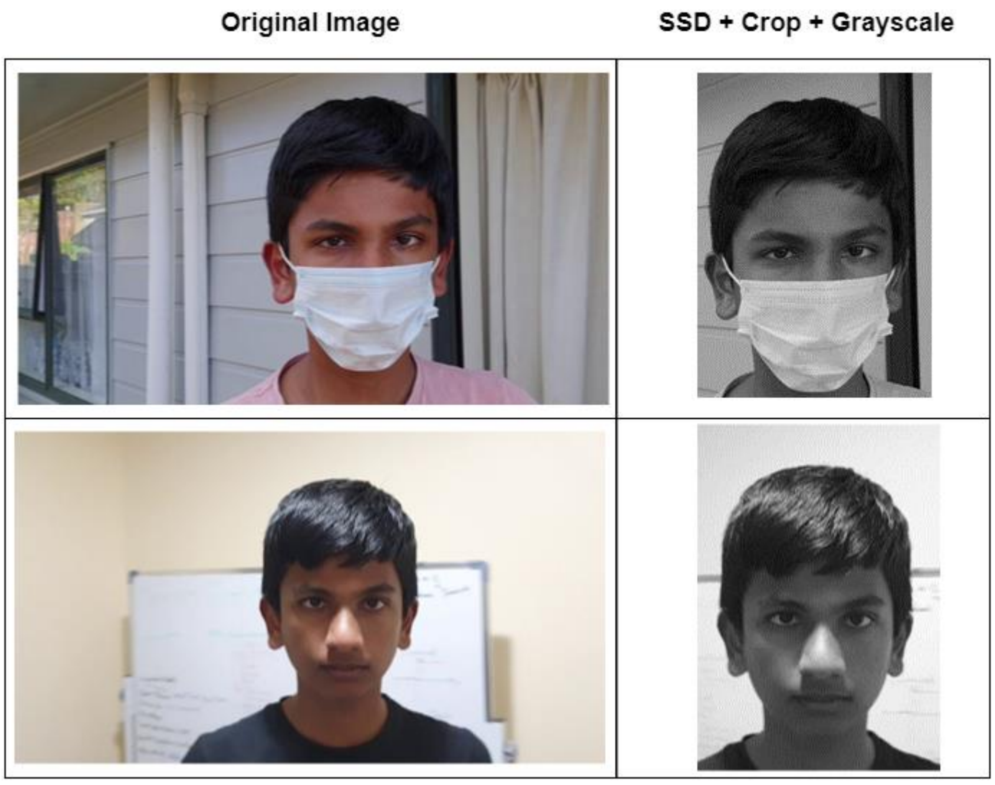


Figure 3-11: Sample of SSD + Crop + Grayscale

*3.1.3.4.2 Sharpness*

The *cv2.filter2D* function from the OpenCV library is employed to improve image sharpness in this experiment. This function applies a convolution kernel to the input image, enhancing its clarity and definition, particularly in terms of edges and details.

The kernel used for sharpening, as illustrated in Figure 8, is a 3x3 matrix:

| | | |
|---|---|---|
| -1 | -1 | -1 |
| -1 | 9 | -1 |
| -1 | -1 | -1 |

Figure 3-12: Kernel for Sharpness

This kernel, known as a Laplacian filter, serves two purposes: edge detection and sharpening. By subtracting the average value of the neighbouring pixels from the central pixel, the Laplacian filter accentuates edges and high-frequency features within the image. The values within the kernel are carefully selected to achieve this operation. The centre pixel is assigned a weight of 9, while the surrounding pixels receive a weight of -1. The implementation of the sharpness function is illustrated in Figure 3-14.

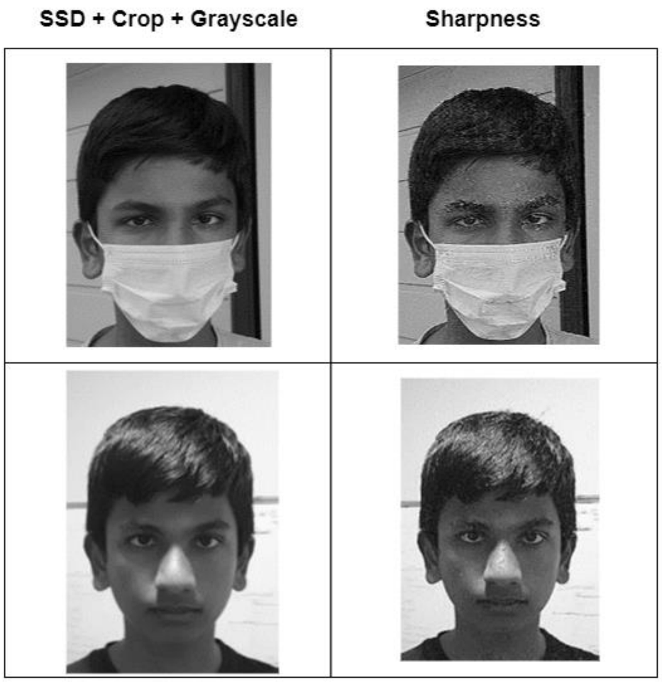
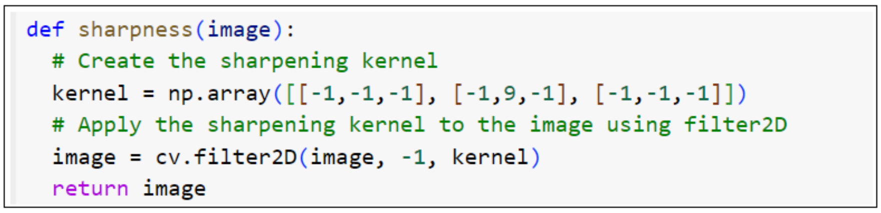


Figure 3-13: Sample of sharpness

When the filter2D function is applied to the input image frame using this kernel, it produces an output image that exhibits enhanced sharpness. The edges and details in the image are intensified, resulting in a clearer and more defined appearance. The visual impact of this sharpening filter can be observed in Figure **3-13**.



```python
def sharpness(image):
  # Create the sharpening kernel
  kernel = np.array([[-1,-1,-1], [-1,9,-1], [-1,-1,-1]])
  # Apply the sharpening kernel to the image using filter2D
  image = cv.filter2D(image, -1, kernel)
  return image
```

Figure 3-14: Implementation of Sharpness

#### *3.1.3.4.3 Size*

In the preprocessing stage, resizing the image before feeding it to the classification model does not provide any significant benefits since its input size is already 256 × 256. As a result, the resized image would be reverted back to the model input size. However, in this experiment, one of the objectives is to evaluate the performance of elongated faces. Figure 10 provides a visualisation of an elongated face after the resizing technique.

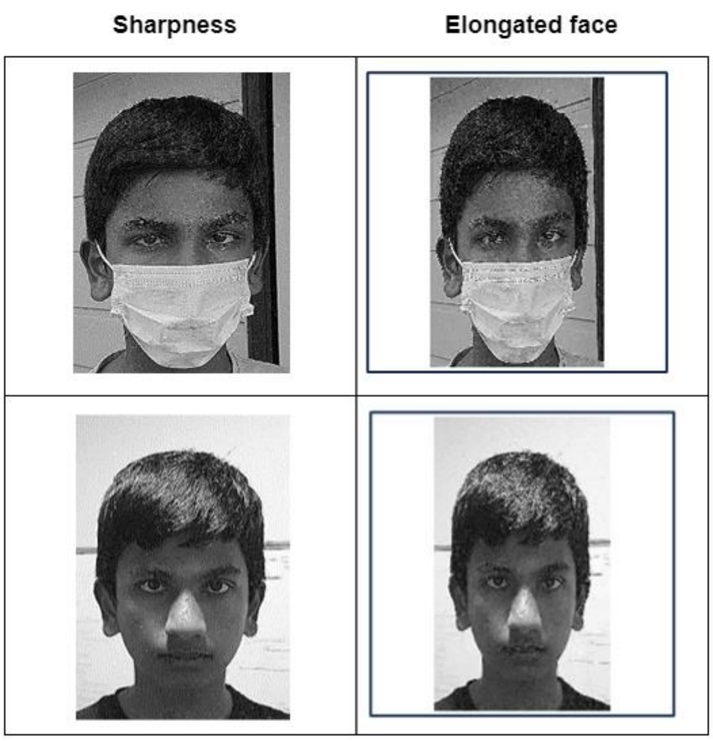
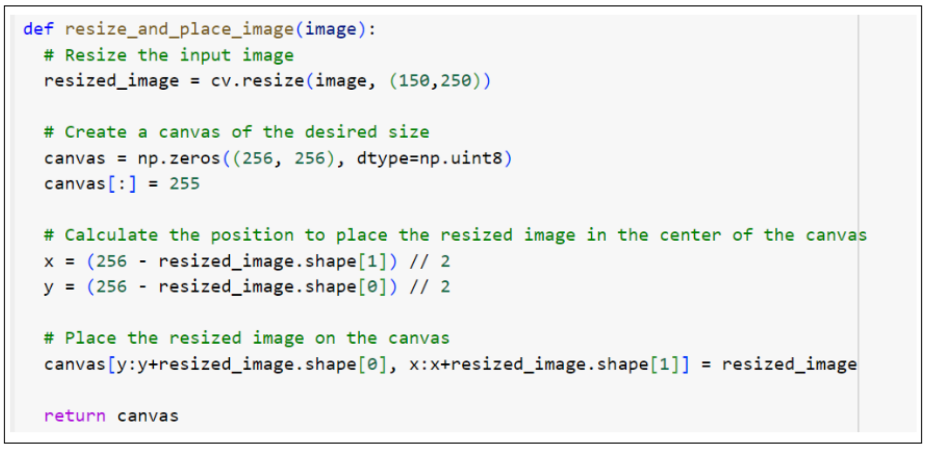


Figure 3-15: Sample of elongated face

To achieve this, the face was resized to 150 x 250 (width x height) and positioned on a white canvas with dimensions of 256 x 256. This resizing approach was implemented to maintain the aspect ratio of the face and mitigate any distortions introduced during the model input resizing process. Placing the resized face on a larger canvas helps retain its elongated appearance as it progresses through the classification model. Figure **3-16** provides a visualisation of the implementation of this resizing technique.



```python
def resize_and_place_image(image):
  # Resize the input image
  resized_image = cv.resize(image, (150,250))

  # Create a canvas of the desired size
  canvas = np.zeros((256, 256), dtype=np.uint8)
  canvas[:] = 255

  # Calculate the position to place the resized image in the center of the canvas
  x = (256 - resized_image.shape[1]) // 2
  y = (256 - resized_image.shape[0]) // 2

  # Place the resized image on the canvas
  canvas[y:y+resized_image.shape[0], x:x+resized_image.shape[1]] = resized_image

  return canvas
```

Figure 3-16: Implementation of elongated face

## 3.2 SUMMARY

This chapter has outlined the various stages of the research process and their practical execution, beginning with selecting training data, creating multiple models, developing a real-world test dataset, and conducting experiments to address the research questions effectively. The subsequent chapter will delve into the results and findings of these experiments, providing insights and conclusions based on the outcomes.

# CHAPTER 4

# RESULTS AND DISCUSSION

# CHAPTER 5

# CONCLUSION AND FUTURE WORK

## 5.1 CONCLUSIONS

In conclusion, this research study was structured to address several key research questions related to developing a classification prediction model to identify whether a person is wearing a mask or not.

The first research question (RQ1) focused on investigating how the accuracy of detection models is affected when images are captured in real-world scenarios with varying lighting conditions that are beyond the scope of the training data context. To simulate such conditions, the study utilised a synthetic dataset created from celebrity faces and then developed a control test dataset with diverse lighting scenarios, including sunny, cloudy, shadow, artificial indoor light, and outdoor light. The experiments conducted provided crucial insights into how different lighting conditions impact the accuracy of prediction models. The findings revealed that models trained solely on synthetic data struggled to recognise real-world face masks under varying lighting conditions, highlighting the challenge of transferring models from synthetic to real-world environments.

Research question 2 (RQ2) aimed to determine which preprocessing techniques could be employed to improve detection accuracy in real-world scenarios characterised by diverse lighting conditions. The results indicated that preprocessing steps such as sharpness enhancement and resizing positively impacted model performance. These specific techniques were selected based on their demonstrated effectiveness in enhancing model accuracy with the dataset created for this study. Other preprocessing techniques like histogram equalisation, contrast adjustment, brightness enhancement, and pixel intensity manipulation were also explored, showing potential for improving prediction accuracy in different lighting situations.

Furthermore, the study compared multiple pre-trained architectures using transfer learning, including VGG-16, VGG-19, ResNet, DenseNet, and MobileNet, to determine which architecture yielded the best outcomes in real-world settings. The analysis considered factors such as feature extraction capabilities, performance under adverse conditions, computational speed, and model size. The results highlighted the effectiveness of architectures like DenseNet and ResNet in feature extraction, even in challenging conditions, demonstrating their potential for real-world applications.

This research underscores the importance of addressing real-world challenges in model development, particularly in environments with varying lighting conditions. By leveraging appropriate preprocessing techniques and robust pre-trained architectures, the performance of face mask detection and recognition models can be significantly improved, leading to more reliable outcomes in practical scenarios where training data may not fully capture the complexities of the target environment.

## 5.2 RECOMMENDATIONS AND FUTURE WORK

Based on the findings and conclusions drawn from the research study, several recommendations and potential areas for future work can be suggested:

***Exploring Advanced Preprocessing Techniques***

Further investigation into advanced preprocessing techniques beyond sharpness enhancement and resizing could be beneficial. Techniques such as deep learning-based image enhancement algorithms or adaptive image normalisation methods could be explored to further enhance model performance in real-world scenarios.

***Data Augmentation Strategies***

Experimenting with more diverse data augmentation strategies during training could help improve model robustness. Techniques such as random rotations, translations, and colour augmentations can add variability to the training data and enhance the model's ability to generalise to different lighting conditions.

***Transfer Learning Optimisation***

Fine-tuning transfer learning strategies to adapt pre-trained models more effectively to real-world scenarios could be explored. This may involve adjusting learning rates, exploring different optimiser configurations, or implementing domain adaptation techniques to bridge the gap between synthetic and real-world data.

***Integration of Environmental Factors***

Considering additional environmental factors beyond lighting conditions, such as background complexity, occlusions, and varying distances from the camera, could further refine the model's performance and make it more applicable in diverse settings.

***Ensemble Methods***

Investigating ensemble learning methods by combining predictions from multiple models or variations of the same model trained with different preprocessing techniques could lead to improved overall accuracy and robustness.

***Real-time Implementation***

Implementing the face mask detection model in real-time applications, such as on embedded systems or mobile devices, would be an important step towards practical deployment. Optimisation for real-time performance, resource efficiency, and scalability would be key considerations in such implementations.

By addressing these recommendations and exploring future avenues of research, the face mask detection and recognition field can continue to evolve, leading to more reliable, accurate, and ethically sound solutions for real-world applications.

## REFERENCES

[1] "WHO Coronavirus (COVID-19) Dashboard | WHO Coronavirus (COVID-19) Dashboard With Vaccination Data." Accessed: Jun. 26, 2023. [Online]. Available: https://covid19.who.int/

[2] M. Liao *et al.*, "A technical review of face mask wearing in preventing respiratory COVID-19 transmission," *Curr Opin Colloid Interface Sci*, vol. 52, p. 101417, Apr. 2021, doi: 10.1016/J.COCIS.2021.101417.

[3] "Paris Tests Face-Mask Recognition Software on Metro Riders - Bloomberg." Accessed: Jun. 26, 2023. [Online]. Available: https://www.bloomberg.com/news/articles/2020-05-07/paris-tests-face-mask-recognition-software-on-metro-riders?leadSource=uverify%20wall#xj4y7vzkg

[4] "Datakalab - Neural Network Compression." Accessed: Jun. 26, 2023. [Online]. Available: https://datakalab.com/

[5] N. Zeng, H. Zhang, B. Song, W. Liu, Y. Li, and A. M. Dobaie, "Facial expression recognition via learning deep sparse autoencoders," *Neurocomputing*, vol. 273, pp. 643–649, Jan. 2018, doi: 10.1016/J.NEUCOM.2017.08.043.

[6] D. K. Jain, Z. Zhang, and K. Huang, "Multi angle optimal pattern-based deep learning for automatic facial expression recognition," *Pattern Recognit Lett*, vol. 139, pp. 157–165, Nov. 2020, doi: 10.1016/J.PATREC.2017.06.025.

[7] D. K. Jain, Z. Zhang, and K. Huang, "Random walk-based feature learning for micro-expression recognition," *Pattern Recognit Lett*, vol. 115, pp. 92–100, Nov. 2018, doi: 10.1016/J.PATREC.2018.02.004.

[8] N. Zeng, H. Li, and Y. Peng, "A new deep belief network-based multi-task learning for diagnosis of Alzheimer's disease," *Neural Comput Appl*, vol. 35, no. 16, pp. 11599–11610, Jun. 2021, doi: 10.1007/S00521-021-06149-6/METRICS.

[9] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *Adv Neural Inf Process Syst*, vol. 28, 2015, Accessed: Jun. 26, 2023. [Online]. Available: https://github.com/

[10] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," Apr. 2018, Accessed: Jun. 26, 2023. [Online]. Available: https://arxiv.org/abs/1804.02767v1

[11] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," Apr. 2020, Accessed: Jun. 26, 2023. [Online]. Available: https://arxiv.org/abs/2004.10934v1

[12] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, Sep. 2014, Accessed: Jun. 26, 2023. [Online]. Available: https://arxiv.org/abs/1409.1556v6

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778. Accessed: Jun. 26, 2023. [Online]. Available: http://image-net.org/challenges/LSVRC/2015/

[14] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely Connected Convolutional Networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 4700–4708. Accessed: Jun. 26, 2023. [Online]. Available: https://github.com/liuzhuang13/DenseNet.

[15] I. B. Venkateswarlu, J. Kakarla, and S. Prakash, “Face mask detection using MobileNet and global pooling block,” *4th IEEE Conference on Information and Communication Technology, CICT 2020*, Dec. 2020, doi: 10.1109/CICT51604.2020.9312083.

[16] M. Loey, G. Manogaran, M. H. N. Taha, and N. E. M. Khalifa, “Fighting against COVID-19: A novel deep learning model based on YOLO-v2 with ResNet-50 for medical face mask detection,” *Sustain Cities Soc*, vol. 65, p. 102600, Feb. 2021, doi: 10.1016/J.SCS.2020.102600.

[17] U. Mahbub, S. Sarkar, and R. Chellappa, “Partial face detection in the mobile domain,” *Image Vis Comput*, vol. 82, pp. 1–17, Feb. 2019, doi: 10.1016/J.IMAVIS.2018.12.003.

[18] V. Bazarevsky, Y. Kartynnik, A. Vakunov, K. Raveendran, and M. Grundmann, “BlazeFace: Sub-millisecond Neural Face Detection on Mobile GPUs,” Jul. 2019, Accessed: Jun. 26, 2023. [Online]. Available: https://arxiv.org/abs/1907.05047v2

[19] S. Singh, U. Ahuja, M. Kumar, K. Kumar, and M. Sachdeva, “Face mask detection using YOLOv3 and faster R-CNN models: COVID-19 environment,” *Multimed Tools Appl*, vol. 80, no. 13, pp. 19753–19768, May 2021, doi: 10.1007/S11042-021-10711-8/FIGURES/8.

[20] “Facemask Detection Dataset 20,000 Images | Kaggle.” Accessed: Jun. 29, 2023. [Online]. Available: https://www.kaggle.com/datasets/pranavsingaraju/facemask-detection-dataset-20000-images

[21] T. Ojala, M. Pietikäinen, and T. Mäenpää, “Multiresolution gray-scale and rotation invariant texture classification with local binary patterns,” *IEEE Trans Pattern Anal Mach Intell*, vol. 24, no. 7, pp. 971–987, Jul. 2002, doi: 10.1109/TPAMI.2002.1017623.

[22] T. H. Kim, D. C. Park, D. M. Woo, T. Jeong, and S. Y. Min, “Multi-class classifier-based adaboost algorithm,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7202 LNCS, pp. 122–127, 2012, doi: 10.1007/978-3-642-31919-8_16/COVER.

[23] S. Yang, P. Luo, C.-C. Loy, and X. Tang, “WIDER FACE: A Face Detection Benchmark,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 5525–5533. Accessed: Jun. 27, 2023. [Online]. Available: http://mmlab.ie.cuhk.edu.hk/projects/

[24] B. F. Klare *et al.*, “Pushing the frontiers of unconstrained face detection and recognition: IARPA Janus Benchmark A,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 07-12-June-2015, pp. 1931–1939, Oct. 2015, doi: 10.1109/CVPR.2015.7298803.

[25] B. Yang, J. Yan, Z. Lei, and S. Z. Li, “Fine-grained evaluation on face detection in the wild,” *2015 11th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition, FG 2015*, Jul. 2015, doi: 10.1109/FG.2015.7163158.

[26] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, 2001, doi: 10.1109/CVPR.2001.990517.

[27] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, “Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks,” *IEEE Signal Process Lett*, vol. 23, no. 10, pp. 1499–1503, Oct. 2016, doi: 10.1109/LSP.2016.2603342.

[28] D. Chen, S. Ren, Y. Wei, X. Cao, and J. Sun, “Joint cascade face detection and alignment,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8694 LNCS, no. PART 6, pp. 109–122, 2014, doi: 10.1007/978-3-319-10599-4_8/COVER.

[29] H. Li, Z. Lin, X. Shen, J. Brandt, and G. Hua, “A Convolutional Neural Network Cascade for Face Detection.” pp. 5325–5334, 2015.

[30] B. Yang, J. Yan, Z. Lei, and S. Z. Li, "Fine-grained evaluation on face detection in the wild," *2015 11th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition, FG 2015*, Jul. 2015, doi: 10.1109/FG.2015.7163158.

[31] C. Zhu, Y. Zheng, K. Luu, and M. Savvides, "CMS-RCNN: Contextual multi-scale region-based CNN for unconstrained face detection," *Advances in Computer Vision and Pattern Recognition*, vol. PartF1, pp. 57–79, 2017, doi: 10.1007/978-3-319-61657-5_3/COVER.

[32] M. Opitz, G. Waltner, G. Poier, H. Possegger, and H. Bischof, "Grid loss: Detecting occluded faces," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9907 LNCS, pp. 386–402, 2016, doi: 10.1007/978-3-319-46487-9_24/FIGURES/7.

[33] H. Li, Z. Lin, X. Shen, J. Brandt, and G. Hua, "A Convolutional Neural Network Cascade for Face Detection." pp. 5325–5334, 2015.

[34] S. L. Jackson, "Research Methods: A Modular Approach," 2010, Accessed: Mar. 26, 2024. [Online]. Available: http://83.136.219.140:8080/handle/123456789/72

[35] J. Wang and W. Wan, "Experimental design methods for fermentative hydrogen production: A review," *Int J Hydrogen Energy*, vol. 34, no. 1, pp. 235–244, Jan. 2009, doi: 10.1016/J.IJHYDENE.2008.10.008.

[36] D. Haase and M. Amthor, "Rethinking Depthwise Separable Convolutions: How Intra-Kernel Correlations Lead to Improved MobileNets." pp. 14600–14609, 2020.

[37] D. Sinha and M. El-Sharkawy, "Thin MobileNet: An Enhanced MobileNet Architecture," *2019 IEEE 10th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference, UEMCON 2019*, pp. 0280–0285, Oct. 2019, doi: 10.1109/UEMCON47517.2019.8993089.

[38] M. Shorfuzzaman and M. Masud, "On the detection of COVID-19 from chest X-Ray images using CNN-based transfer learning," *Materials & Continua CMC*, vol. 64, no. 3, pp. 1359–1381, 2020, doi: 10.32604/cmc.2020.011326.

[39] K. Wadhawan and E. Gajendran, "Automated recognition of text in images: A survey," *Int J Comput Appl*, vol. 127, no. 15, pp. 975–8887, 2015.

[40] X. Ke, J. Li, and W. Guo, "Dense small face detection based on regional cascade multi-scale method," *IET Image Process*, vol. 13, no. 14, pp. 2796–2804, Dec. 2019, doi: 10.1049/IET-IPR.2018.6571.