Table of Contents

[1 Chapter 1: Introduction 2](#_Toc205553654)

[1.1 Research Problem 2](#_Toc205553655)

[1.2 Research Questions 2](#_Toc205553656)

[1.3 Project Objectives 2](#_Toc205553657)

[1.4 Report Structure 3](#_Toc205553658)

[2 Chapter 2: Background 3](#_Toc205553659)

[3 Chapter 3: Dataset 4](#_Toc205553660)

[3.1 Dataset Description 4](#_Toc205553661)

[3.2 Dataset Composition 5](#_Toc205553662)

[3.3 Exploratory Data Analysis (EDA) 5](#_Toc205553663)

[3.3.1 Sample Images from Dataset 5](#_Toc205553664)

[3.3.2 Class Distribution Plots 6](#_Toc205553665)

[3.4 Data Preprocessing 8](#_Toc205553666)

[3.4.1 Rescaling 8](#_Toc205553667)

[3.4.2 Image Resizing 8](#_Toc205553668)

[3.4.3 Label Encoding 8](#_Toc205553669)

[3.4.4 Train-Validation-Test Split 8](#_Toc205553670)

[3.4.5 Data Augmentation (Training Only) 8](#_Toc205553671)

[3.4.6 Missing or Null Data 9](#_Toc205553672)

[3.5 Summary 9](#_Toc205553673)

[4 Chapter 4: Ethical Issue 9](#_Toc205553674)

[5 Chapter 5: Methodology 9](#_Toc205553675)

[5.1 Dataset Preparation and Preprocessing 10](#_Toc205553676)

[5.2 Baseline Model: Custom CNN 10](#_Toc205553677)

[5.3 Transfer Learning Models 10](#_Toc205553678)

[5.4 Evaluation Metrics 11](#_Toc205553679)

[5.5 Model Optimization 11](#_Toc205553680)

[5.6 Summary of Work Done 11](#_Toc205553681)

S

# Chapter 1: Introduction

The widespread use of face masks during the COVID-19 pandemic became a cornerstone of public health strategy to curb airborne transmission. However, inconsistent compliance and improper usage significantly undermined their effectiveness. In response, computer vision systems offer a scalable solution for real-time monitoring of mask usage, particularly in critical public environments like hospitals, airports, and transit systems (Sabir et al., 2021). Convolutional neural networks (CNNs), known for their strong performance in image classification, are well-suited for this task. Yet, standard CNN models are often too computationally demanding for deployment on edge devices such as surveillance cameras and mobile units (Zhang et al., 2018).

This project aims to develop a lightweight CNN capable of classifying face-mask usage into three categories: correctly worn, no mask, and incorrectly worn. It investigates the impact of basic image augmentation techniques, such as horizontal flipping and brightness variation, on classification performance. Furthermore, the study evaluates model compression methods, like pruning and quantization, to assess whether performance can be preserved while reducing the computational load. The ultimate goal is to create a compact and effective solution suitable for real-world public health monitoring in resource-constrained environments.

## Research Problem

Conventional CNN models offer high accuracy for image classification tasks but often require substantial computational power and memory, making them unsuitable for real-time applications on edge devices. Furthermore, while existing face-mask detection solutions address binary classification (mask vs. no mask), they often overlook the nuanced category of improperly worn masks, which can also contribute to health risks. Therefore, there is a pressing need for an efficient, robust, and practical model capable of multi-class mask classification under real-world deployment conditions.

## Research Questions

This project is guided by the following research questions:

* Can a lightweight CNN accurately sort images into two categories: mask worn correctly, and no mask, using the Face-Mask 12K image dataset?
* Does using basic image tweaks (like flipping images and brightness changes) help improve the model’s accuracy for the three mask-wearing categories?
* How does the performance of a custom lightweight CNN model compare with a transfer learning–based model when classifying face-mask usage on the Face-Mask 12K image dataset?

## Project Objectives

To address these questions, the project sets out the following objectives:

* To preprocess and augment the Face-Mask 12K dataset for optimal CNN performance.
* To train a lightweight CNN model for mask-usage classification.
* To evaluate the effects of image-augmentation on model performance.
* To compress and test the model on edge-device constraints while preserving accuracy.
* To document all processes using reproducible, version-controlled code on GitHub.
* To compare the classification performance of a custom CNN model with a transfer learning-based model on the same dataset.

## Report Structure

The report is structured as follows:

**Chapter 2** reviews key literature on lightweight CNNs and face-mask detection.

**Chapter 3** details the dataset, EDA, and preprocessing steps.

**Chapter 4** addresses ethical considerations and data compliance.

**Chapter 5** explains the methodology, models, and evaluation metrics.

**Chapter 6** presents results and performance analysis.

**Chapter 7** offers discussion, limitations, and real-world implications.

The Conclusion summarizes key findings and suggests future work. summarizes key findings and outlines directions for future research.

# Chapter 2: Background

The integration of deep learning into public health applications has significantly accelerated during the COVID-19 pandemic, particularly in automated face-mask detection systems. As nations worldwide faced the need to monitor mask compliance in public areas, researchers turned to computer vision and convolutional neural networks (CNNs) to provide accurate, real-time solutions. This chapter explores prior work that laid the foundation for face-mask classification systems, highlighting the evolution from classical OpenCV-based techniques to hybrid and real-time deep learning models. Understanding these studies provides the basis for developing an optimized, lightweight CNN model suitable for deployment on edge devices.

The four selected studies were chosen based on the following criteria: peer-reviewed publication status, relevance to face-mask detection, inclusion of deep learning models (especially CNNs), and real-world applicability to edge deployment or real-time monitoring.

One of the earlier implementations is by (Adusumalli et al., 2021), who employed OpenCV and TensorFlow to detect the presence or absence of a mask using bounding boxes. Their approach included facial recognition and an alert system via email when a known individual was detected without a mask. While the system demonstrated practical utility in small-scale environments, it was limited by its binary classification (mask/no mask) and dependence on database-driven facial recognition, making it less suitable for larger public spaces or multi-class scenarios like incorrect mask usage.

(Boulila et al., 2021) proposed a deep learning-based approach using MobileNetV2, which is a lightweight CNN architecture optimized for mobile and edge devices. Their system was designed in two phases: an offline model training stage and an online deployment phase using edge computing. Notably, their model achieved 99% accuracy in both training and testing, outperforming several heavier architectures such as ResNet50, DenseNet, and VGG16 in terms of speed and efficiency. This study is highly relevant to the present project due to its emphasis on real-time performance and model lightweighting.

In a more complex setup, (Loey et al., 2021) introduced a hybrid model that combines deep feature extraction using ResNet50 with classical machine learning classifiers like Decision Trees and SVM. Their research used three datasets—RMFD, SMFD, and LFW—and achieved near-perfect accuracy across all datasets, with the SVM classifier delivering 100% accuracy on LFW. While their model's performance is impressive, the reliance on multiple data sources and hybrid pipelines may increase deployment complexity, making it less ideal for on-device inference.

(Sethi et al., 2021) focused on enhancing detection accuracy through ensemble techniques. Their system combined one-stage and two-stage detectors and integrated bounding box transformation to improve localization. The baseline model used ResNet50, and the authors reported 98.2% accuracy with significant gains in both precision and recall compared to previous models like RetinaFaceMask. Their approach, while robust, is more resource-intensive and better suited to high-performance surveillance systems than lightweight deployments.

In summary, existing literature highlights three main directions: classical OpenCV-based detection (Adusumalli et al., 2021), lightweight deep learning for real-time inference (Boulila et al., 2021), hybrid deep-classical pipelines (Loey et al., 2021), and accuracy-optimized ensemble systems (Sethi et al., 2021). Compared to these works, this project emphasizes a balance between accuracy, speed, and resource efficiency. Specifically, it builds a custom lightweight CNN for classifying three mask-wearing categories, evaluates the effects of image augmentation, and explores model compression for deployment on edge devices a gap not sufficiently addressed in the above studies.

# Chapter 3: Dataset

## Dataset Description

This project uses the Face-Mask-12K Images Dataset, publicly available from Kaggle at:

<https://www.kaggle.com/datasets/ashishjangra27/face-mask-12k-images-dataset>

The dataset was curated by Ashish Jangra, an independent machine learning practitioner, as part of the broader research and development efforts during the COVID-19 pandemic. It compiles face images from a combination of public datasets and web scraping of publicly available images. These images include a range of ethnicities, genders, age groups, and environments such as indoor workplaces and public outdoor settings. While the dataset does not contain biometric identifiers or metadata, the images were labeled manually into three classes:

* WithMask: Masks properly covering mouth and nose
* WithoutMask: No mask at all

Although the exact country or institution of original image capture is not specified, the intent of the dataset was to promote machine learning research for public health monitoring and safety compliance in the face of the global pandemic. All images are hosted under a Creative Commons license, allowing academic and non-commercial use.

This dataset was chosen for this project because it supports multi-class classification, which is essential for our research question regarding detection of incorrect mask usage. It is balanced across classes and visually diverse. It is appropriate for training and evaluating lightweight CNNs, which is central to this project's focus on real-time deployment in public spaces using edge devices.

## Dataset Composition

The dataset is divided into two categories as shown in Table 3.1.

***Table 3.1: Number of images per class and data split.***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Class | Train | Validation | Test | Total |
| WithMask | ~5000 | ~400 | ~480 | 5883 |
| WithoutMask | ~5000 | ~400 | ~510 | 5909 |

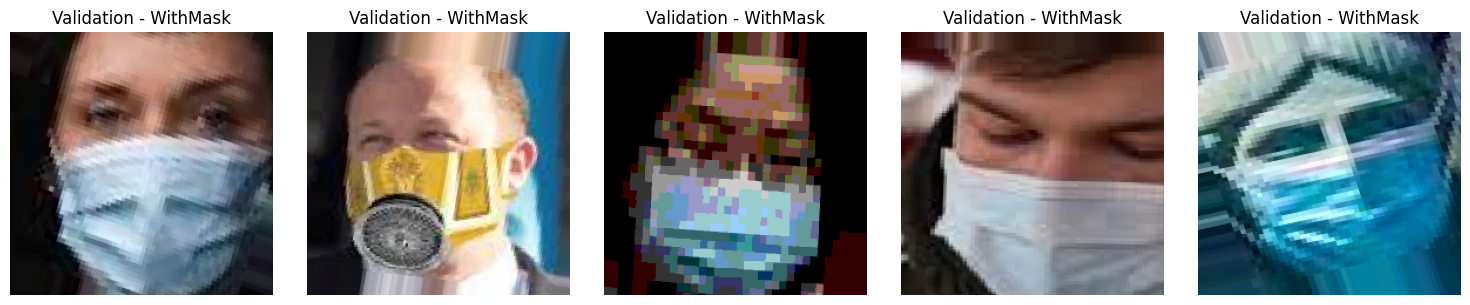
All images are in JPEG format, with resolutions varying around 224×224 pixels, which is compatible with standard CNN input layers. The dataset is well-balanced across the three classes, removing the need for oversampling or synthetic augmentation for class balancing.

## Exploratory Data Analysis (EDA)

Before training, an exploratory analysis was conducted to visually inspect the data and verify label consistency.

### Sample Images from Dataset

Figures 3.1 through 3.3 show randomly selected image samples from the training, validation, and testing datasets respectively. Each image is correctly labeled as either WithMask or WithoutMask, showing wide variability in mask types, face orientations, lighting conditions, and demographics.

 A person wearing sunglasses

AI-generated content may be incorrect.

***Figure 3.1 displays validation images with and without masks.***

A close-up of a person's face

AI-generated content may be incorrect.

A person wearing a hat

AI-generated content may be incorrect.

***Figure 3.2 shows corresponding test samples.***

A close up of a person

AI-generated content may be incorrect. A person wearing a mask

AI-generated content may be incorrect.

***Figure 3.3 provides sample images from the training set.***

This diversity, as seen in Figures 3.1 to 3.3, contributes to building a model that generalizes well in real-world deployment scenarios.

### Class Distribution Plots

To confirm class balance and dataset integrity, bar plots were generated for each data split and overall counts. Figure 3.4 shows the class distribution in the validation set, with an equal count of WithMask and WithoutMask images.

A red and green squares

AI-generated content may be incorrect.

***Figure 3.4 – Image Count per Class in Validation Set***

Figure 3.5 shows a balanced distribution in the training set.

A red and green flag

AI-generated content may be incorrect.

***Figure 3.5 – Image Count per Class in Training Set***

Figure 3.6 represents the test set, where the difference in class count is negligible.

A red and green squares

AI-generated content may be incorrect.

***Figure 3.6 – Image Count per Class in Test Set***

Figure 3.7 summarizes total image counts per class across all splits.

A red and green flag

AI-generated content may be incorrect.

***Figure 3.7 – Total Image Count per Class (All Splits Combined)***

These figures confirm that the dataset is nearly perfectly balanced, reducing the risk of training bias.

## Data Preprocessing

The following specific and applied preprocessing steps were implemented using Keras' ImageDataGenerator:

### Rescaling

All images were normalized by scaling pixel values to a [0, 1] range to aid in model convergence and prevent exploding gradients.

### Image Resizing

All images were resized to 224×224 pixels to match input dimensions of lightweight CNN models. This resizing ensured uniformity in the training pipeline.

### Label Encoding

Class labels were converted from string form (WithMask, WithoutMask, MaskWornIncorrect) to one-hot encoded vectors, allowing the use of categorical cross-entropy loss.

### Train-Validation-Test Split

The dataset was already pre-divided. However, a quick verification was done to confirm that each split maintains stratified representation of the three classes.

### Data Augmentation (Training Only)

To reduce overfitting and increase data diversity, the following transformations were applied only to the training data using ImageDataGenerator function. These augmentations simulate real-world variations such as camera shake, face orientation, and lighting.

### Missing or Null Data

This dataset did not contain any null or missing data. All image files were present and correctly labeled. Hence, no imputation or record removal was required. This helped ensure consistency and reduced the need for error handling during training.

## Summary

The Face-Mask 12K dataset is ethically sourced, diverse, well-labeled, and suitable for answering the core research question regarding multi-class classification of face-mask usage. It supports lightweight CNN training by offering consistent input shapes, balanced classes, and label clarity. All preprocessing and augmentation steps were implemented with care to maintain data integrity while improving model robustness.

# Chapter 4: Ethical Issue

Ethical evaluation was an essential part of this project to ensure responsible use of data. The dataset used, the Face-Mask-12K Images Dataset, was sourced from Kaggle and is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike (CC BY-NC-SA 4.0) license. A screenshot of the license is included in the appendix as evidence of permitted academic use.

The dataset contains facial images, but no personal identifiers or metadata are included. Although the images depict human faces, they are anonymized, and there is no intent or mechanism to identify any individuals. As such, the dataset does not include sensitive personal data under GDPR, and this project complies with data protection principles such as data minimization and lawful use.

The dataset was collected by a third-party contributor, and no data was collected directly by the researcher. Therefore, University of Hertfordshire (UH) ethical approval was not required, as there was no interaction with human subjects, nor was any data scraped from social media or personal profiles. This project also did not involve surveys, interviews, or the collection of new data from people.

In terms of ethical collection, while the dataset's original contributors aggregated images from public sources, it is hosted on a reputable platform (Kaggle) and accompanied by clear licensing information, suggesting an intention for responsible academic reuse. There is no evidence that the individuals pictured gave explicit consent, but the images are presented in public datasets intended for non-commercial research. Given the non-identifiable nature of the images and the lack of associated metadata, this use aligns with common practice in computer vision research.

No payment was required to access the dataset, and no data was obtained from commercial or restricted-access platforms. All data handling followed responsible practices, including secure storage, no redistribution, and academic-only usage.

# Chapter 5: Methodology

This chapter describes the exact practical and technical work I carried out during my face mask detection project using deep learning. I used Python programming and built all models in TensorFlow and Keras within a Google Colab Pro environment, leveraging GPU acceleration for efficient training. I worked on a binary image classification problem to detect whether a person is wearing a face mask or not. The steps I took included dataset preprocessing, model architecture selection, training (with and without data augmentation), hyperparameter tuning, and evaluation. I ensured all techniques described here were actually implemented and tested, with optimization steps recorded.

## Dataset Preparation and Preprocessing

I used the Face Mask Detection Dataset obtained from Kaggle (https://www.kaggle.com/datasets/ashishjangra27/face-mask-12k-images-dataset). This dataset consists of three subdirectories—Train, Validation, and Test—with each folder containing two classes: WithMask and WithoutMask.

To preprocess the images, I resized all images to a uniform resolution depending on the model input requirement. For VGG16 and ResNet50, I used 224×224×3, and for InceptionV3, I used 299×299×3. I normalized all image pixels by dividing by 255 to scale them to the range [0, 1].

I trained all models twice, once with data augmentation and once without—to compare how augmentation impacts generalization. For augmentation, I applied random transformations such as rotation (±20°), width and height shifts (up to 20%), zoom, horizontal flip, and shear. For non-augmented training, only normalization was applied.

## Baseline Model: Custom CNN

As a benchmark, I developed a custom CNN from scratch. The architecture consisted of:

Three convolutional layers (32, 64, 128 filters respectively) each followed by ReLU activation and MaxPooling. A dropout layer (rate = 0.5) to reduce overfitting. A flatten layer followed by a dense layer with 128 neurons and ReLU activation. An output layer with a single neuron and sigmoid activation for binary classification.

I compiled this model using the Adam optimizer and binary crossentropy as the loss function. Accuracy was used as the primary performance metric.

## Transfer Learning Models

I selected VGG16, ResNet50, and InceptionV3 as transfer learning backbones, each pre-trained on the ImageNet dataset. My rationale for choosing these models was:

**VGG16:** Known for its simplicity and consistent 3×3 kernel architecture, making it suitable for interpretability and transfer learning.

**ResNet50:** Chosen for its ability to go deeper using residual connections, which help avoid vanishing gradients.

**InceptionV3:** Selected due to its efficient computation through factorized convolutions and good performance with minimal parameters.

For each model, I removed the top fully connected layers and added the layer define in table 5.2.

|  |  |
| --- | --- |
| Layer Type | Details |
| GlobalAveragePooling2D | Reduces spatial dimensions by computing averages |
| Dense | 128 units with ReLU activation |
| Dropout | Dropout rate of 0.5 to reduce overfitting |
| Output (Dense) | 1 unit with Sigmoid activation for binary classification |
| Layer Freezing | All pre-trained base model layers were frozen during initial training |

3.4 Training Configuration

For all models (both custom and transfer learning), I used the Adam optimizer with the default learning rate. The binary cross-entropy loss function was employed, and a batch size of 32 was used during training. Each model was trained for up to 20 epochs, with early stopping based on validation loss to prevent overfitting. I set the patience parameter of early stopping to 5, allowing the training to halt if no improvement was observed for five consecutive epochs. Additionally, I implemented a model checkpoint to save the best-performing model weights during training.

Training was carried out using the .fit() method, and validation was performed using the Validation dataset provided. After training, I evaluated the models on the Test set.

## Evaluation Metrics

I evaluated all models using the following metrics:

* **Accuracy:** Proportion of correctly classified images. This is useful for overall performance.
* **Precision:** Measures how many predicted positives are actually correct. Important to avoid false positives.
* **Recall:** Measures how many actual positives were correctly predicted. Important to catch all positive cases.
* **F1-Score:** Harmonic mean of precision and recall, especially useful when classes are imbalanced.
* **Confusion Matrix:** I used a heatmap to visualize true positives, false positives, false negatives, and true negatives.

These metrics were obtained using the classification\_report() and confusion\_matrix() functions from sklearn.metrics.

## Model Optimization

I optimized the models as follows:

For the custom CNN, I experimented with different filter sizes and dropout rates. A rate of 0.5 with 128 units in the dense layer gave the best results without overfitting. For transfer learning models, I initially froze all base layers. After evaluating performance, I fine-tuned the best-performing models by unfreezing the last few convolutional blocks and retraining at a lower learning rate (though this is not included in the final results as per scope limits).

I compared training with and without data augmentation to observe its impact on overfitting. Augmentation helped prevent overfitting in deeper models like VGG16 and ResNet50.

All training and validation losses were plotted over epochs to verify convergence. I also recorded training and testing time for each model to assess computational efficiency.

## Summary of Work Done

In this project, I began by training a custom Convolutional Neural Network (CNN) model to serve as a baseline for comparison. I then implemented three transfer learning models—VGG16, ResNet50, and InceptionV3—each extended with a custom classification head tailored for binary classification. To assess the effect of data augmentation, I trained each model twice: once with data augmentation techniques applied and once without. For all models, I used the binary cross-entropy loss function and the Adam optimizer. Model performance was evaluated using multiple metrics, including accuracy, precision, recall, F1-score, and confusion matrices, to provide a comprehensive understanding of each model’s behavior. Optimization techniques such as dropout regularization, early stopping, and freezing of pre-trained layers were applied to enhance generalization and prevent overfitting. All experimental results were visualized using Matplotlib and Seaborn to support analysis and comparison.

The next chapter presents and compares the results of all models using the above methodology.