Middlesex University Dubai

PDE4433 Coursework 2 – Face Mask Detection using CNN

BY AISWARYA SHAJAN

MISIS: M01035110

COURSE: MSC. ROBOTICS

FACULTY: MS. MAHA SAADEH

Index

List of Figures

1. Introduction

Face masks have become an essential measure in post-pandemic times. Their use is mandatory in critical areas of hospitals such as operating rooms and isolated wards. However, mask compliance is often unsupervised, which could be addressed by integrating mask detection through hospital CCTV systems.

This project focuses on analyzing and testing an existing face-mask detection system developed by DataFlair, as described in their blog post *"Real-Time Face Mask Detector with Python, OpenCV, Keras."* By utilizing the methodologies and code provided in the system, the project employs an image dataset for training and testing the model, which is subsequently utilized to detect individuals wearing masks in real-time using a laptop camera.

2. Exploring the dataset

The dataset comprises a total of 1,376 images in JPG format, with 690 depicting individuals wearing masks and 686 showing individuals without masks. It includes both standard images of people wearing masks and augmented images, where mask cutouts have been superimposed onto faces. It is a ZIP folder that has two sub zip folders for both training and validation inside of which they are sub-divided into mask/no mask categories.

The inputs are the images itself and the output labels to predict are mask and no mask.

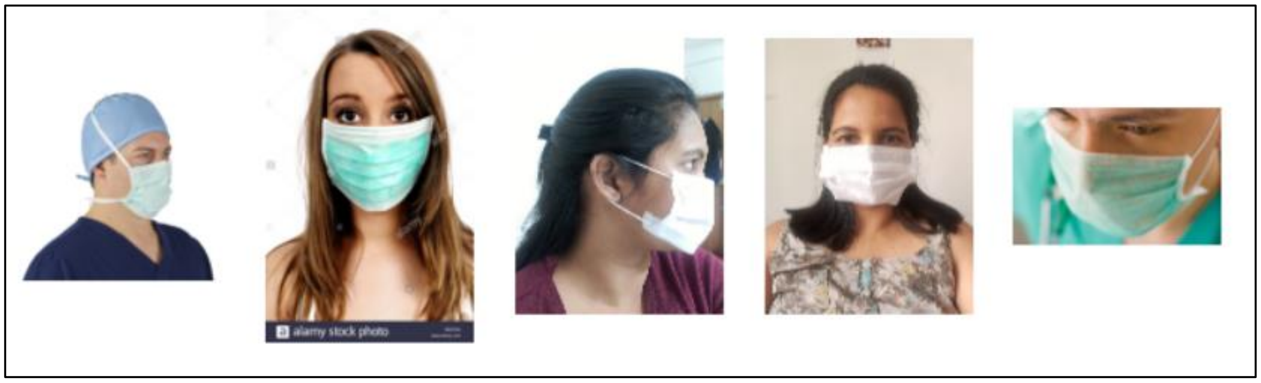


Figure 1: With mask

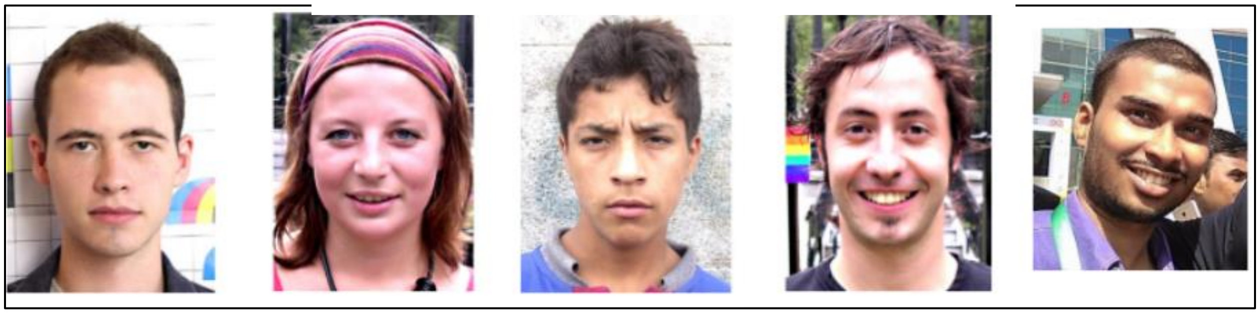
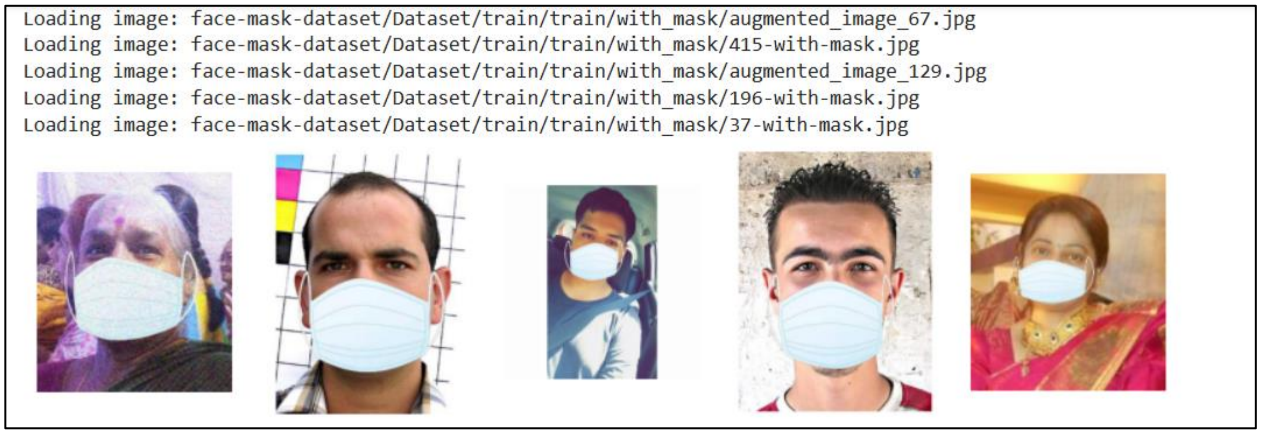


Figure 2: Without mask

Augmentation (Pre-processing)

Image augmentation is a crucial preprocessing technique used in this project to enhance the dataset's diversity and improve model robustness. In general, it is the process of generating new images for training a deep learning model. These new images are generated using the existing training images and done applying transformations such as rotation, flipping, scaling, or in this case overlaying mask cutouts onto faces. This helps in:

- Mimicking real-world scenarios by adding variations (lighting, angles, and facial orientations).
- Balancing the dataset by increasing specific categories (e.g., images with masks).
- Making the model more reliable and effective at handling diverse and new data.



```
Loading image: face-mask-dataset/Dataset/train/train/with_mask/augmented_image_67.jpg
Loading image: face-mask-dataset/Dataset/train/train/with_mask/415-with-mask.jpg
Loading image: face-mask-dataset/Dataset/train/train/with_mask/augmented_image_129.jpg
Loading image: face-mask-dataset/Dataset/train/train/with_mask/196-with-mask.jpg
Loading image: face-mask-dataset/Dataset/train/train/with_mask/37-with-mask.jpg
```

Figure 3: Augmented images from the train dataset in colab

3. Model selection

The project utilizes a Convolutional Neural Network (CNN) model, which has consistently demonstrated superior performance across various image recognition benchmarks. This makes CNNs the preferred choice for tasks such as image classification, object detection, and image segmentation, and thus a reliable tool for real-world applications like face mask detection. The following reasons justify the use of CNN:

- **Feature Extraction:** CNNs automatically learn and extract essential features from images, such as edges, patterns, and textures, which are critical for distinguishing masked faces from unmasked ones.
- **Local Connectivity:** The convolutional layers in a CNN focus on smaller, localized regions of an image, allowing the model to capture spatial patterns, such as the placement and shape of a face mask.
- **Translation Invariance:** Pooling layers in the network ensure that the model can recognize face masks, even if they appear in different positions or orientations within the image.

- **Parameter Sharing:** By reusing filters across the image, CNNs reduce the total number of parameters, making the model more efficient and less prone to overfitting.

4. Model Development

4.1. CNN Architecture:

- **Input Layer -** Accepts images resized to 150x150 with RGB channels.
- **Feature Extraction Layers (2 pairs each):**

  Conv2D Layer (100 filters, 3x3 kernel, ReLU activation):

  - Extracts feature maps from input images.
  - Followed by **MaxPooling2D** (2x2 pooling) to reduce spatial dimensions.

- **Flatten Layer**: Converts 2D feature maps into a 1D vector.
- **Dropout Layer (rate = 0.5)**: Prevents overfitting during training by randomly dropping 50% of the data.
- **Fully Connected Layers**
  - **Dense Layer (50 neurons, ReLU activation)**: Learns complex relationships between features. Combines the 1D data from Flatten to make sense of the data.
  - **Output Dense Layer (2 neurons, Softmax activation)**: Decision making layer that provides probabilities for two classes: "Mask" and "No Mask."

- **Compilation Details**
  - **Optimizer**: Adaptive Moment Estimation (Adam)– adjusts the learning rate dynamically for faster and more efficient training – tells the model how to learn

- **Loss Function**:
  - Crossentropy – used for binary classification tasks (mask/no mask)
  - Measures difference between predicted output and actual class label. The aim of the function is to minimise this loss. – tells the model what to optimize.
- **Metric**: Accuracy – measures the number of correctly predicted instances out of the total instances. Tells the model what to track.

4.2. Image Data Generation/Augmentation:

- The **ImageDataGenerator** class from TensorFlow/Keras is employed to perform image augmentation and allows real-time transformation of images during the training process.
- Following the rescaling process, images undergo a rotation of ±40 degrees, random horizontal and vertical translations, shearing, zooming and horizontal flipping of upto 20%.

4.3. Training Details:

- Model trained using the dataset split into training and validation sets.
- Callback checkpoints were used to save the best-performing models after each epoch.

4.4. Evaluation

- **Confusion Matrix:** The model accurately classified 96 masked and 97 unmasked faces, with only 1 misclassification. This demonstrates high reliability in distinguishing between the two classes.

- **Precision:** Both "with_mask" and "without_mask" classes achieved a precision score of **1.0**, indicating perfect accuracy with no false positives.

- **Recall:** The model achieved a recall score of **1.0** for both classes, successfully identifying all relevant instances without missing any.

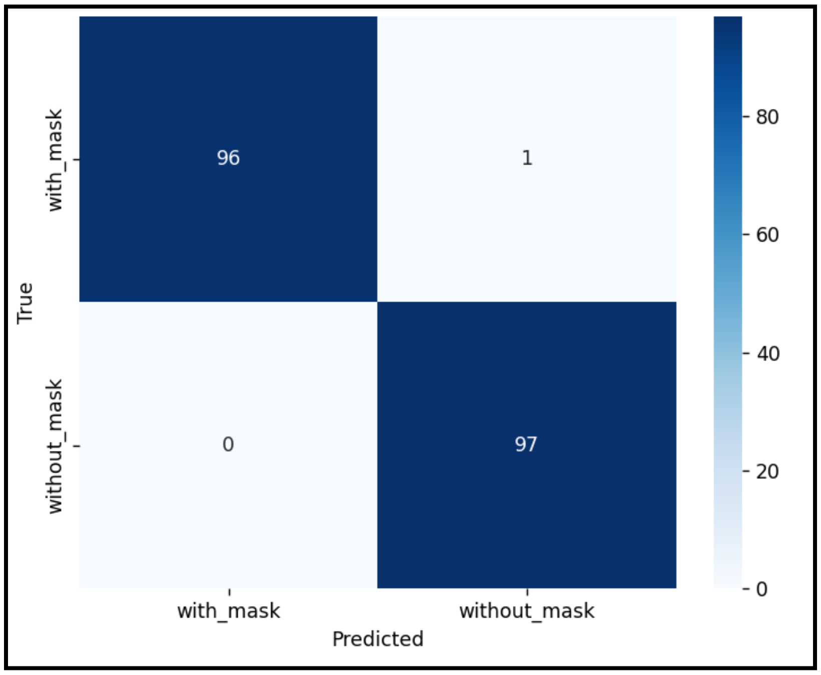


Figure 4: Confusion matrix

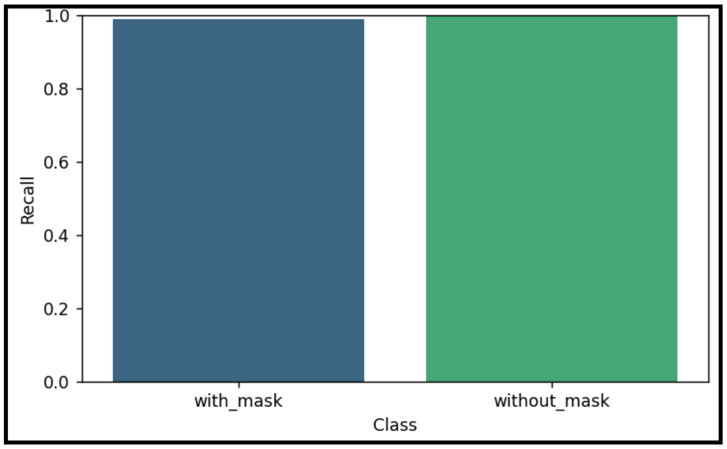


Figure 5: Recall per class

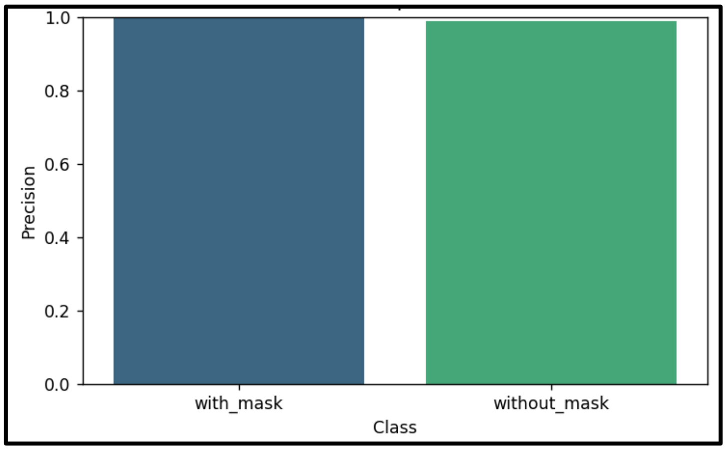


Figure 6: Precision per class

4.5. Implementation

- Testing was performed using OpenCV with a live webcam feed.
- Pre-trained Haar Cascade classifiers were used for face detection.
- The final model categorized faces as either:
  - "With Mask" (green bounding box).
  - "Without Mask" (red bounding box).
- **Training Accuracy (acc)**: 93.41%
- **Training Loss (loss)**: 0.1789
- **Validation Accuracy (val_acc)**: 99.48%
- **Validation Loss (val_loss)**: 0.0351

5. Results

- The model demonstrates satisfactory results when the webcam is positioned against a plain background (Figure 7).
- However, certain displayed results tend to fluctuate and lack stability.
- Testing in different backgrounds led to few incorrect outputs, as illustrated in Figure 8.

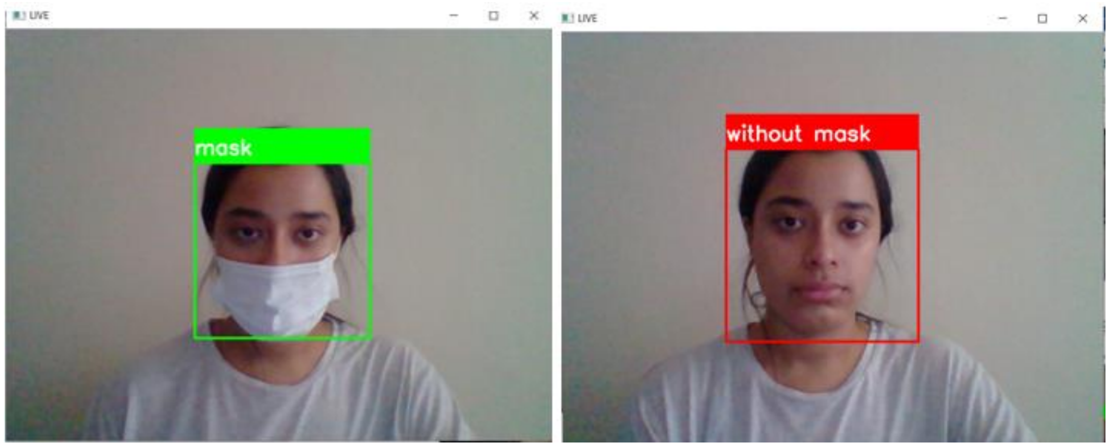


Figure 7: Successful detection

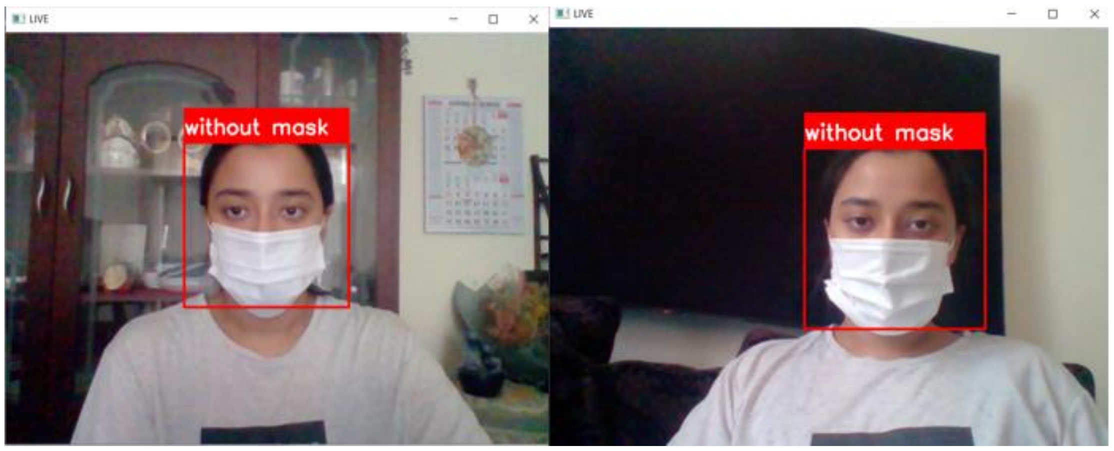


Figure 8: Incorrect detection against other backgrounds

6. Comparison with pre-trained YOLO model

- The pre-trained YOLOv5 model from the GitHub repository YOLOv5 Face Mask Detection (link cited under references) was used to benchmark the performance of a custom CNN model for face mask detection.

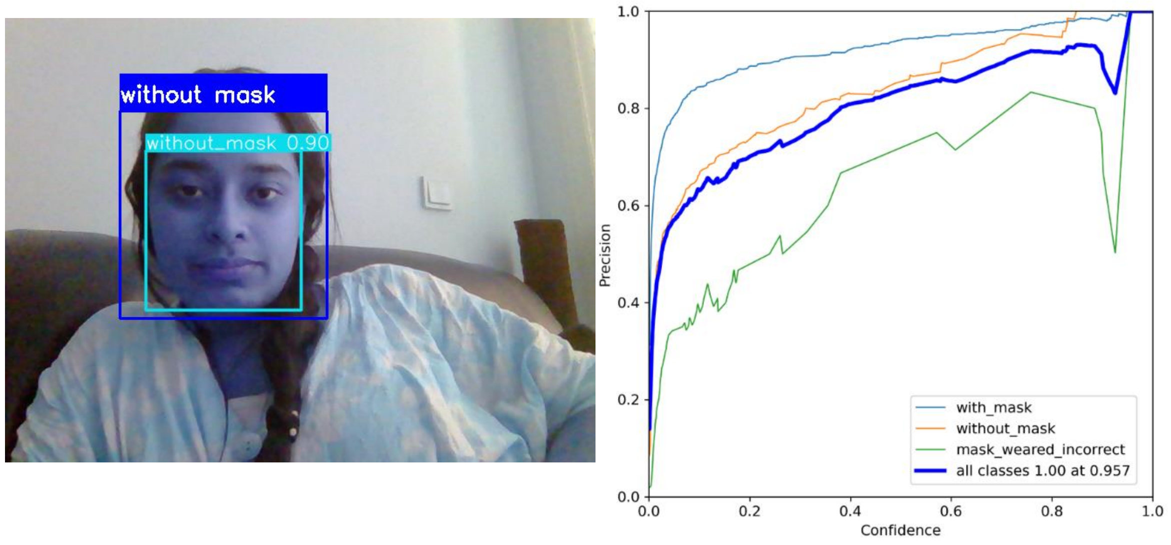


Figure 9: YOLO model prediction

- Metrics such as precision, recall, and mean average precision (mAP) were compared to evaluate the strengths and weaknesses of each model in handling diverse detection scenarios.

7. Conclusion and Future Improvements

- This project demonstrates the implementation of a deep learning model to enhance public health and safety measures. It highlights the potential for AI-driven solutions in addressing real-world challenges.

- **Background Adaptability:** Training on a diverse dataset with varied backgrounds and applying preprocessing techniques like background subtraction can enhance the model's performance across different scenarios.

- **Result Stability:** Introduce smoothing techniques, such as temporal smoothing, to stabilize predictions over consecutive frames in real-time detection scenarios. Adjusting hyper-parameters and using ensemble methods could also improve result consistency.

- **Multiple Face Mask Detection:** Modify the model to accommodate multi-object detection frameworks by utilizing anchor boxes for detecting multiple faces in a single frame. Expanding the dataset to include images with multiple individuals wearing masks would further enhance its accuracy.

8. References

- DataFlair. (n.d.). Real-Time Face Mask Detector with Python, OpenCV, Keras. Retrieved from https://data-flair.training/blogs/face-mask-detection-with-python/

- Tanwar, S. (n.d.). Image Augmentation: Improving Deep Learning Models. Analytics Vidhya. Retrieved from Medium: Link

- GitHub. (n.d.). YOLOv5 Face Mask Detection Repository. Retrieved from spacewalk01/yolov5-face-mask-detection: Face Mask Detection using YOLOv5

9. Links

- Video Demonstration - https://youtu.be/MjmzO18RPJE
- GitHub - AiswaryaShajan/face_mask_detection