

PDE4433 – Coursework 2 Report

**by AIsWARYA SHAJAN**

**MISIS: M01035110**

**COurse: MSc. RObotics**

**Faculty: Ms. MAha Saadeh**

**Index**

1. **Introduction……………………………………………………………………3**
2. **Task Overview…………………………………………………………………3**
3. **Preparation…………………………………………………………………….3**
   * 3.1 Trajectory and Point Marking………………………………………3
   * 3.2 Rough Sketch and Measurements…………………………………..3
   * 3.3 Program Steps………………………………………………………3
4. **Execution………………………………………………………………………4**
   * 4.1 Setup………………………………………………………………..4
   * 4.2 Tool Mounting…………………………………………………….. 4
   * 4.3 Robot and Paper Setup…………………………………………….. 5
   * 4.4 Teaching the Points………………………………………………... 5
   * 4.5 Programming……………………………………………………….6
5. **Results………………………………………………………………………… 7**
6. **Future Improvements………………………………………………………... 7**
7. **Citations………………………………………………………………………..8**

### **List of Figures**

1. **Test sheet………………………………………………………………………3**
2. **Rough sketch…………………………………………………………………..4**
3. **Tool Offset……………………………………………………………………..4**
4. **Points…………………………………………………………………………...5**
5. **Program………………………………………………………………………..7**

**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.

**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 each category.

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

Figure 1: With mask

1





Figure 2: Without mask

1

**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.



Figure 3: Augmented images from the train dataset in colab

**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.

**Model Development**

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.

1. **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%.

1. **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.

#### **Evaluation**

* The model achieved high accuracy in distinguishing between mask and non-mask scenarios.
* Performance metrics such as **validation accuracy** and **loss** were tracked during training.

#### **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

#### **Screenshots**



* + Examples of successful detections.
  + Edge cases where detection was challenging (e.g., poor lighting).

#### **Conclusion**

* 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.

**Citations**

* 1. "Real-Time Face Mask Detector with Python, OpenCV, Keras." DataFlair, <https://data-flair.training/blogs/face-mask-detection-with-python/>
  2. [Image Augmentation. Improving Deep learning models | by Sanchit Tanwar | Analytics Vidhya | Medium](https://medium.com/analytics-vidhya/image-augmentation-9b7be3972e27)

**NOTES**

### 2. **Machine Learning Model**

* **Model Selection:**
  + Explain why you chose a CNN for this project (highlight its strengths in image recognition).
* **Architecture:**
  + Describe the architecture of the CNN (e.g., layers like Conv2D, MaxPooling2D, Dropout, Dense).
  + Include a diagram or visualization of the model (if possible).
* **Training Process:**
  + Discuss how the model was trained, including the dataset used (size, content, and any preprocessing/augmentation techniques).
  + Mention hyperparameters (e.g., optimizer, epochs, loss function).

### 3. **Dataset Analysis**

* Provide an overview of the dataset (number of images, categories: mask and no-mask).
* Explain the role of augmented images (if any), to diversify the dataset.
* If relevant, include a script or method to check the dataset (e.g., counting images).

### 4. **Evaluation**

* **Metrics:**
  + Present evaluation metrics (e.g., accuracy, precision, recall, F1 score).
  + If you evaluated the model on a separate test dataset, include those results.
* **Confusion Matrix:**
  + Add a confusion matrix to show true positives, false positives, etc.
* **Scenarios:**
  + Include screenshots of the model in action (e.g., detecting masked/unmasked faces using your laptop camera).

### 5. **Results and Discussion**

* Interpret your findings:
  + How accurate is the model in detecting masks?
  + Are there any common errors (e.g., false negatives or false positives)?
* Discuss real-world applicability and limitations of the system.

### 6. **Screenshots of Scenarios**

* Show practical examples:
  + Model identifying a person wearing a mask.
  + Model detecting a person without a mask.
  + Mention specific examples of edge cases (e.g., incorrect classifications or faces with partial masks).

### 7. **Conclusion**

* Summarize key findings and achievements of the project.
* Suggest future improvements (e.g., handling partial occlusions, scaling to multiple people in an image).

### 8. **References**

* Cite the blog post or other sources you used (e.g., "Real-Time Face Mask Detector with Python, OpenCV, Keras" by DataFlair).