



ISTANBUL AREL UNIVERSITY FACULTY OF ENGINEERING

LEEN364-DEEP LEARNING AND CLASSIFICATION TECHNIQUES

PROJECT REPORT

Project Title: Face Mask Detection Using Deep Learning (MobileNetV2)

Group Members:

ELMOATASEM ALY (*Built the model, EDA, preprocessing, training, evaluation*)

AHMED GAAFAR (*Implemented face detection, real-time video prediction*)

.....

1. INTRODUCTION

- **Objective:** The goal of this project is to build a deep learning model that can detect whether a person is wearing a face mask or not. We classify each detected face into two categories: with mask and without mask.
- **Motivation:** Face mask detection became very important during the COVID-19 pandemic. Automatic systems can help increase safety in public areas such as hospitals and airports. This project combines deep learning and computer vision, which are widely used today in security and healthcare applications.

2. LITERATURE REVIEW

- **CNNs (Convolutional Neural Networks):** These are the most successful models for image classification because they learn features automatically from raw images.

- **Transfer Learning:** Models like **MobileNetV2** trained on ImageNet can be reused for new tasks. This reduces training time and improves accuracy.
- **Face Detection:** Haar cascades and deep face detectors (OpenCV DNN) are commonly used to detect faces before classification.

3. DATASET DESCRIPTION

- **Dataset Name:** Custom Mask Detection Dataset

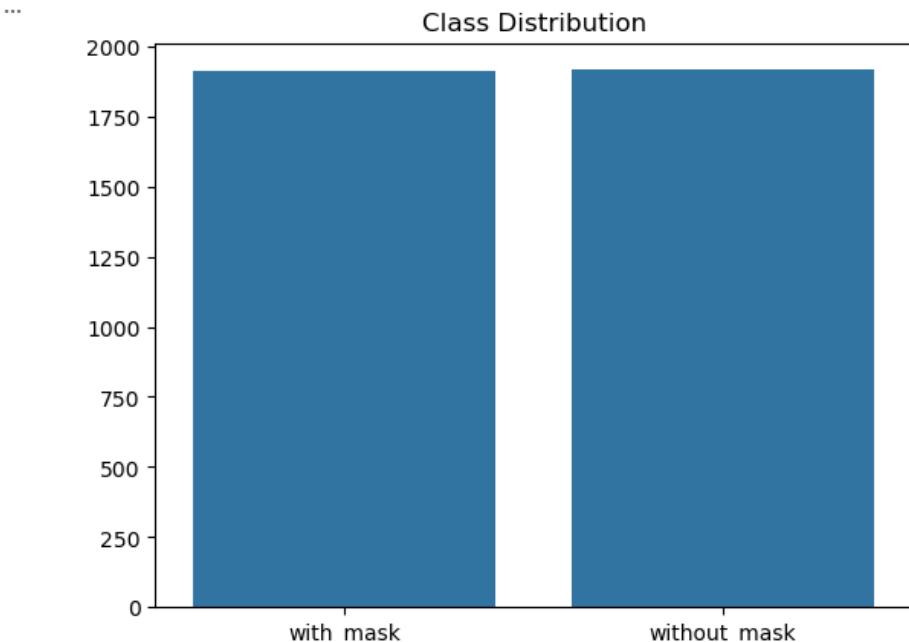


- **Source:** <https://github.com/balajisrinivas/Face-Mask-Detection/tree/master/dataset>
- **Features:**
 - Each sample is an RGB image of a face.
 - Images are resized to **224×224×3** for MobileNetV2.

```
[22]
data = np.array(data)
print("Dataset Shape:", data.shape)
print("Target Shape:", target.shape)

...
Dataset Shape: (3833, 224, 224, 3)
Target Shape: (3833, 2)
```

- Two labels: **with_mask** (1) and **without_mask** (0).
with_mask = 1915 , without_mask = 1918



- **Preprocessing:**
- • Images resized to 224×224

```
image = load_img(img_path , target_size= (224 , 224))
```

- Converted to arrays

```
image = img_to_array(image)
```

- Preprocessed using preprocess_input()

```
image = preprocess_input(image)
```

- Labels encoded using one-hot encoding

```
## Binary Encoder to convert ['with_mask' , 'without_mask'] into [0,1]
lb = LabelBinarizer()
target = lb.fit_transform(target) ## return Vector
target = to_categorical(target) ## convert Vector into Metric
data = np.array(data)
target = np.array(target)
print(data , target)
```

- Data augmented (rotation, zoom, shift, flipping) to improve generalization

```
## func to generate new data images
aug = ImageDataGenerator(
    rotation_range = 20,
    zoom_range = 0.10,
    width_shift_range = 0.1,
    height_shift_range = 0.1,
    shear_range = 0.2,
    horizontal_flip=True,
    fill_mode = 'nearest')

)
```

4. METHODOLOGY

- **Model Selection:**

We used **MobileNetV2**, a pretrained CNN on ImageNet. We removed the top layers and added:

- MaxPooling
- Flatten
- Dense(128, relu)
- Dropout
- Dense(2, softmax)

This allows the model to classify mask vs. no mask efficiently.

```
## func to generate new data images
aug = ImageDataGenerator(
    rotation_range = 20,
    zoom_range = 0.10,
    width_shift_range = 0.1,
    height_shift_range = 0.1,
    shear_range = 0.2,
    horizontal_flip=True,
    fill_mode = 'nearest'
)

0.0s
```

- **Algorithms/Frameworks Used:**

TensorFlow / Keras

OpenCV

NumPy

Scikit learn

ImageDataGenerator for augmentation

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report

from tensorflow.keras.preprocessing.image import ImageDataGenerator, load_img, img_to_array
from tensorflow.keras.layers import *
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import Model

from imutils import paths
import os
```

- **Hyperparameters:**
 - Learning rate: **0.001**
 - □ Optimizer: **Adam**
 - □ Batch size: **32**
 - □ Epochs: **10**
 - □ Loss: **Binary Crossentropy**
 - □ Split: **80% train, 20% test**

```

● ## Compile Our Model

opt = Adam(learning_rate = 0.001)
model.compile(optimizer=opt , loss = 'binary_crossentropy' , metrics=['accuracy'])

train_generator = aug.flow(x_train , y_train , batch_size= 32)

M = model.fit(
    train_generator ,
    epochs=10 ,
    validation_data = (x_test , y_test)
)
✓ 7m 37.1s

```

5. CODE IMPLEMENTATION : The code is divided into several parts:

1. Data Loading & Preprocessing

- Load images from two folders
- Resize and normalize
- Encode labels
- Split into train/test
- Apply data augmentation

2. Model Definition

- Load MobileNetV2
- Freeze base layers
- Add custom classification head
- Compile the model

3. Training

- Train using augmented data
- Track accuracy and loss

4. Evaluation

- Plot accuracy/loss
- Confusion matrix
- Predictions

5. Real-Time Detection Code

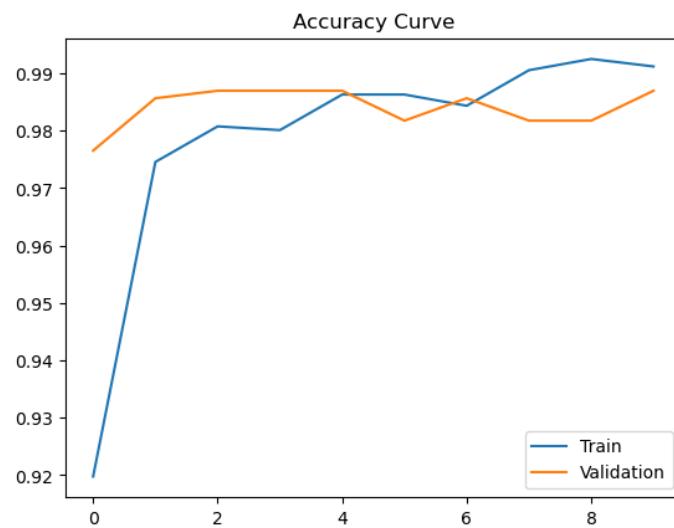
- Load trained model
- Detect faces with OpenCV
- Predict mask/no mask
- Draw bounding boxes
- Show output video

6. RESULTS and ANALYSIS

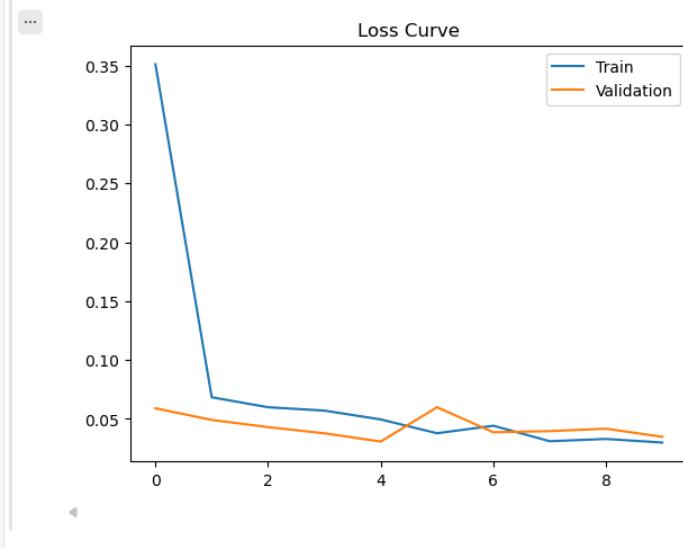
- General Results

```
... 24/24 ━━━━━━ 6s 235ms/step
          precision    recall   f1-score   support
with_mask       0.98      0.99      0.99      383
without_mask    0.99      0.98      0.99      384
accuracy        0.99      0.99      0.99      767
macro avg       0.99      0.99      0.99      767
weighted avg    0.99      0.99      0.99      767
```

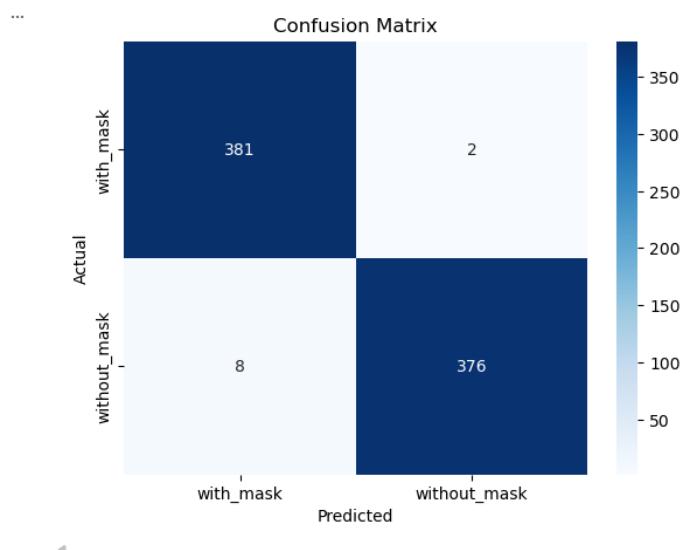
- Accuracy Curve



- Loss Curve



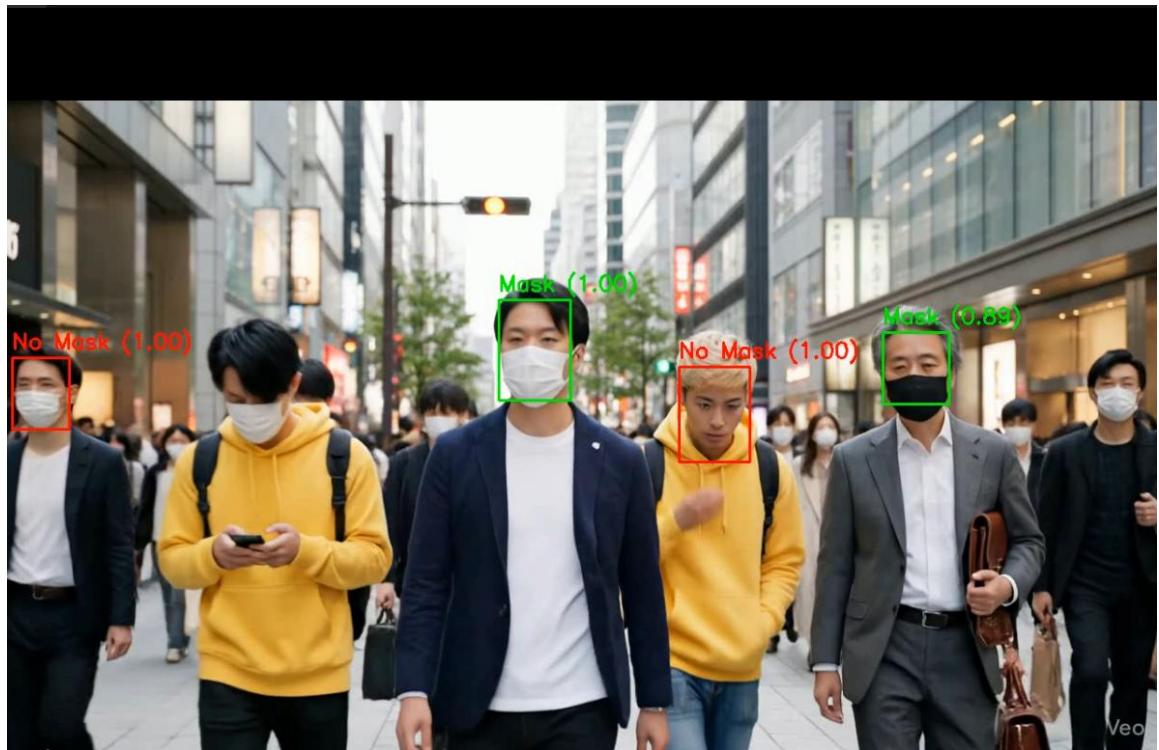
- Confusion Matrix



Many faces with masks correctly classified.

Incorrect cases happen when:

- Mask only covers chin
- Poor lighting
- Face partially visible



7. DISCUSSION and CONCLUSION

Performance Evaluation

The model performed well and reached high accuracy.

MobileNetV2 worked efficiently for this task and allowed fast inference.

Challenges

- Some images were low quality.
- Face detector sometimes missed very small faces.

Conclusion

We successfully created a deep learning system that detects face masks in real-time video.

Future improvements may include:

- Using a better face detector (MTCNN or RetinaFace)
- Training longer
- Adding more mask types (half mask, incorrect mask)
- Deploying the system as a mobile or web app

8. REFERENCES

8. REFERENCES

- TensorFlow Documentation

- OpenCV Documentation
- MobileNetV2 Paper
- Kaggle Mask Datasets
- ChatGPT (AI Assistance)