Face Mask Detection Using Deep Learning (MobileNetV2)

ISTANBUL AREL UNIVERSITY

FACULTY OF ENGINEERING

LEEN364-DEEP LEARNING AND CLASSIFICATION TECHNIQUES

Project Report

Group Members:

ELMOATASEM ALY (Model, EDA, preprocessing, training, evaluation)

AHMED GAAFAR (Face detection, real-time video prediction)

Introduction: Mask Detection Objective

Objective

Build a deep learning model to classify faces as "with mask" or "without mask."

Motivation

Increase safety in public areas like hospitals and airports, combining deep learning and computer vision for security and healthcare.

Literature Review: Key Technologies

CNNs

Convolutional Neural Networks excel at image classification by learning features automatically.

Transfer Learning

Reusing models like MobileNetV2 (trained on ImageNet) reduces training time and boosts accuracy.

Face Detection

Techniques like Haar cascades and OpenCV DNN are crucial for locating faces before classification.

Dataset Description

Our custom dataset includes RGB face images, resized to 224x224x3 for MobileNetV2.

- Source: https://github.com/balajisrinivas/Face-Mask-Detection/tree/master/dataset


Labels

Two categories: with_mask (1915 samples) and without_mask (1918 samples).

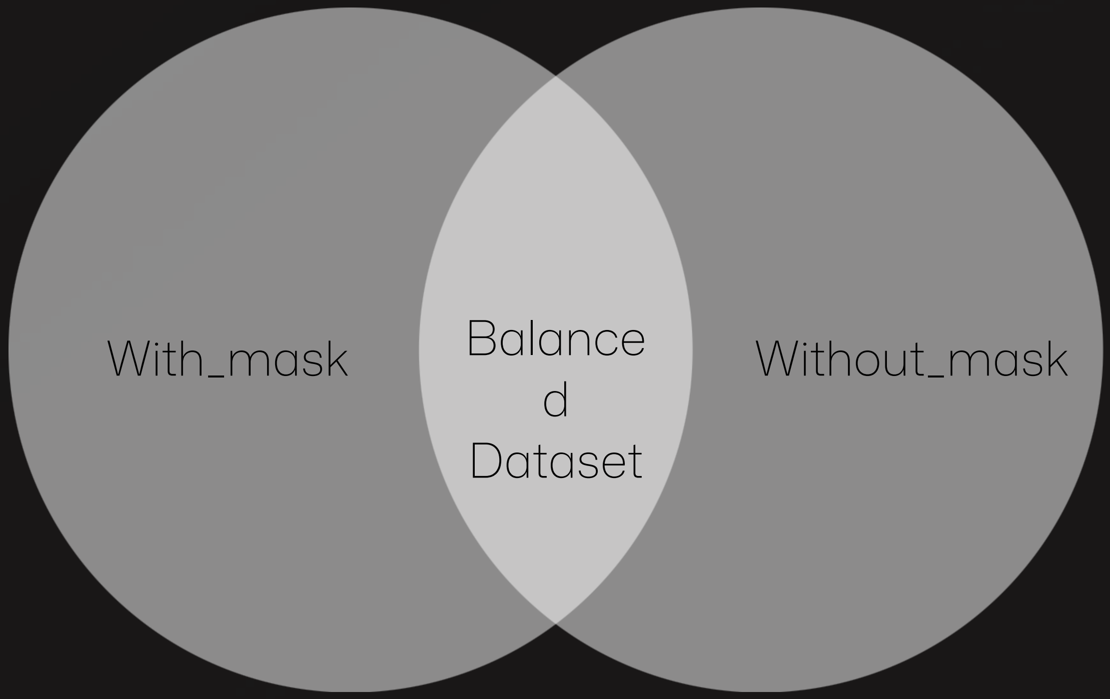
Untitled

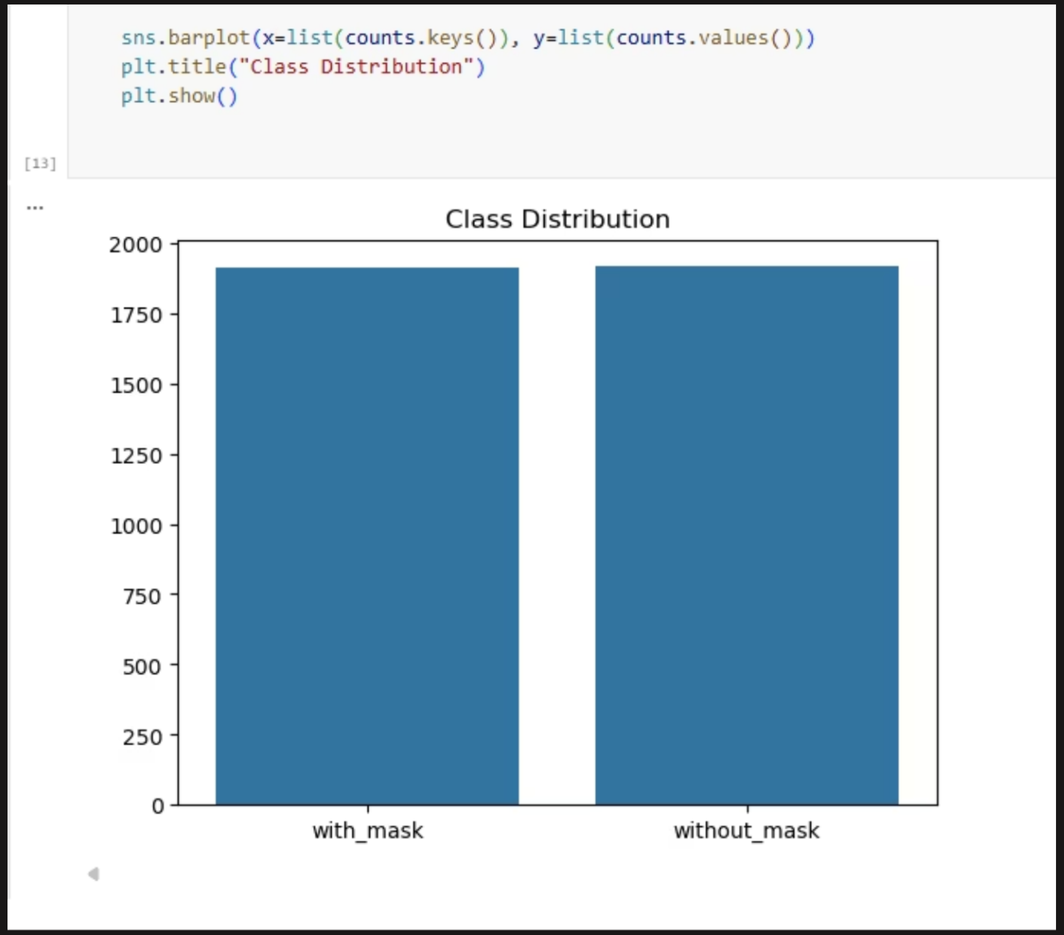
```
[22] data = np.array(data)
     print("Dataset Shape:", data.shape)
     print("Target Shape:", target.shape)

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

Dataset Distribution

The chart illustrates a balanced dataset distribution, with nearly equal numbers of images in both the "with_mask" and "without_mask" classes.





Dataset Preprocessing

Images resized to 224×224 pixels to ensure uniform input dimensions for the model.

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

2

Preprocessed using a specialized function, `preprocess_input()`, to normalize pixel values.

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

Dataset Preprocessing

1

Labels encoded using one-hot encoding, converting categorical labels into a binary vector format.

Untitled

```
## 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)
```

2

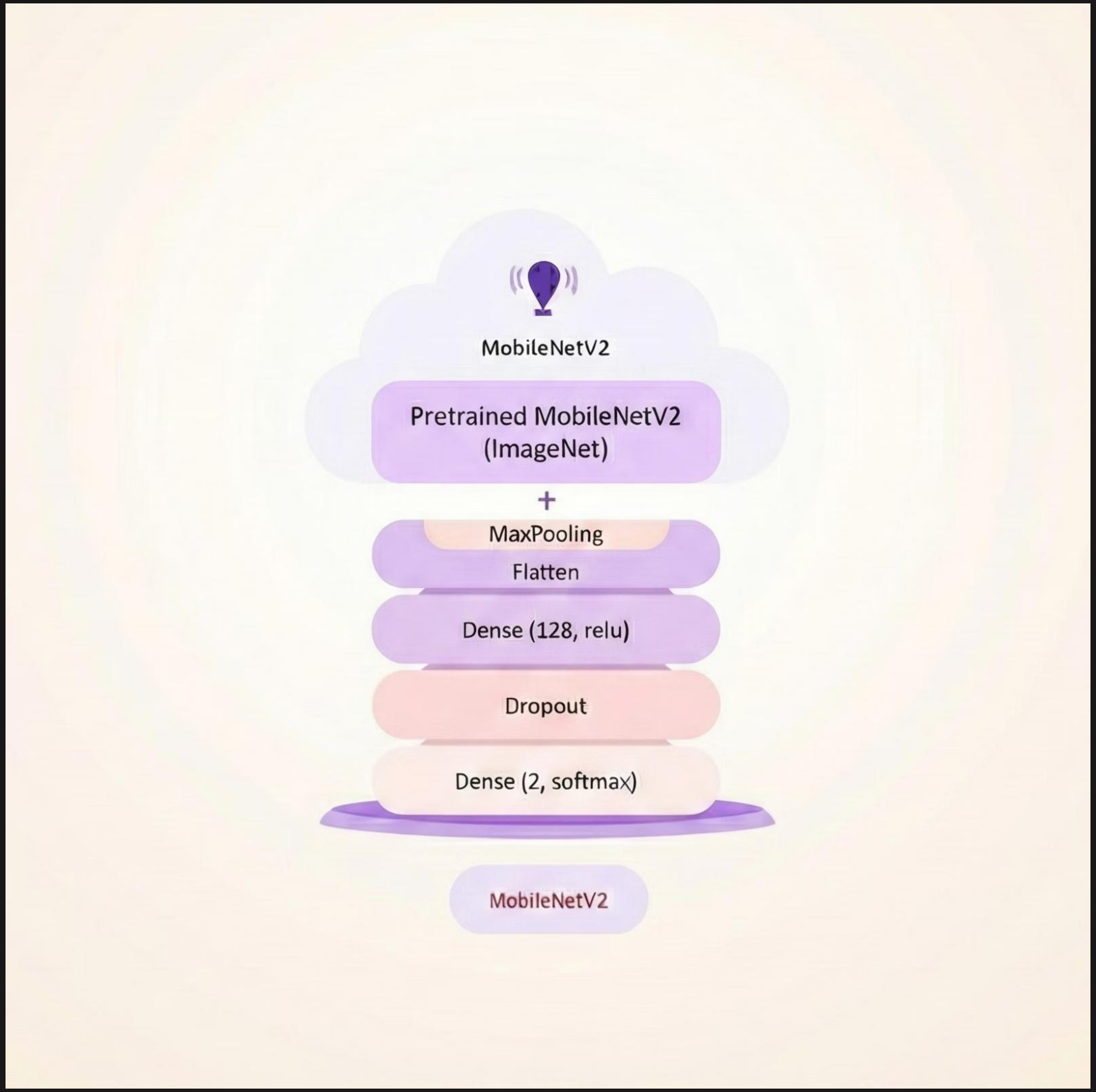
Data augmented through techniques like rotation, zoom, shift, and flipping to enhance model generalization.

Untitled

```
## 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'
)
```

Made with GAMMA

Model Archtícture



Model Selection: MobileNetV2

Utilized a pre-trained MobileNetV2 CNN, customizing its top layers for mask classification:

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

Methodology: Hyperparameters

Hyperparameters

Learning Rate: 0.001	Optimizer: Adam	Batch Size: 32
Epochs: 10	Loss: Binary Crossentropy	Data Split: 80% Train, 20% Test

Methodology: Model & Hyperparameters

Algorithms & Frameworks

- TensorFlow / Keras
- OpenCV
- NumPy
- Scikit-learn
- ImageDataGenerator (for augmentation)

CODE IMPLEMENTATION

Data Loading & Preprocessing

- Load, resize, normalize images
- Encode labels
- Split data
- Apply augmentation

Untitled

```
## split the data ⟶ train- test
x_train , x_test , y_train , y_test = train_test_split(
      data , target ,
      test_size= 0.2 ,
      stratify=target ,
      random_state=42)
```

CODE IMPLEMENTATION

Model Definition

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

Untitled

```
baseModel = MobileNetV2(weights='imagenet' , include_top=False ,input_tensor=Input(shape=(224,224,3)))

head_model = baseModel.output ## this is the last layer in base Model
head_model = MaxPooling2D(pool_size=(7,7))(head_model) ## connect with head_model →Build related connected layers
head_model = Flatten()(head_model)
head_model = Dense(128 , activation= 'relu')(head_model)
head_model = Dropout(0.2)(head_model)
head_model = Dense(2 , activation='softmax')(head_model) ## Output Layer

model = Model(baseModel.input , outputs = head_model )
```

CODE IMPLEMENTATION

Training

- Train the model using augmented data
- tracking accuracy and loss metrics.

Untitled

```
from sklearn.metrics import classification_report

pred = model.predict(x_test)
pred = np.argmax(pred, axis=1)
true = np.argmax(y_test, axis=1)

print(classification_report(true, pred, target_names=categories))
```

CODE IMPLEMENTATION

Evaluation

- Plot accuracy/loss curves
- generate confusion matrix
- make predictions

Untitled

```
from sklearn.metrics import confusion_matrix
import seaborn as sns

cm = confusion_matrix(true, pred)

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=categories,
                                                    yticklabels=categories)
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
```

CODE IMPLEMENTATION

Real-Time Detection

- Load model
- detect faces with OpenCV
- predict mask status
- draw bounding boxes
- display video output

Untitled

```
        # predict
        pred = model.predict(face_input)[0]
        class_id = np.argmax(pred)
        confidence = pred[class_id]

        # label + color
        color = (0, 255, 0) if class_id == 0 else (0, 0, 255)
        text = f"{labels[class_id]} ({confidence:.2f})"

        # draw box
        cv2.rectangle(frame, (x, y), (x+w, y+h), color, 2)
        cv2.putText(frame, text, (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.8, color, 2)

    cv2.imshow("Mask Detection", frame)
```

Results and Analysis

The system has a near-perfect score of 99%, showing it handles both groups equally well.

Untitled

```
              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       767
   macro avg       0.99      0.99      0.99       767
weighted avg       0.99      0.99      0.99       767
```

With Mask Performance

The model is excellent at finding masks, catching 99% of the people who are wearing one.

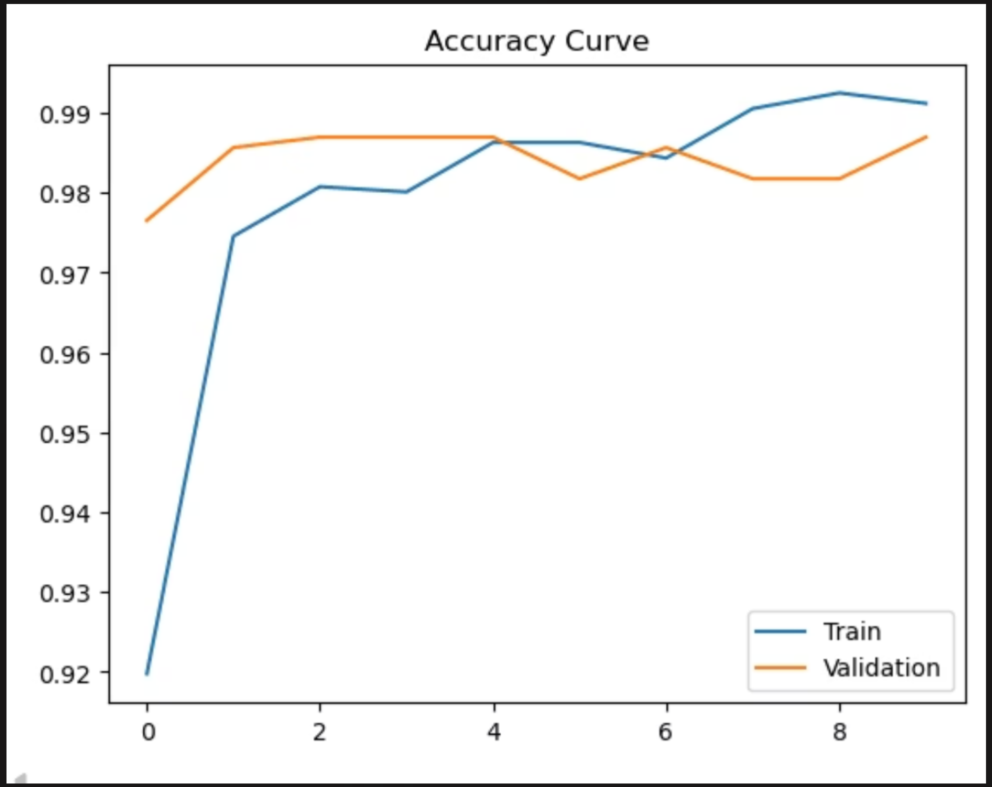
Without Mask Performance

When the model says someone is not wearing a mask, it is correct 99% of the time.

Overall Accuracy

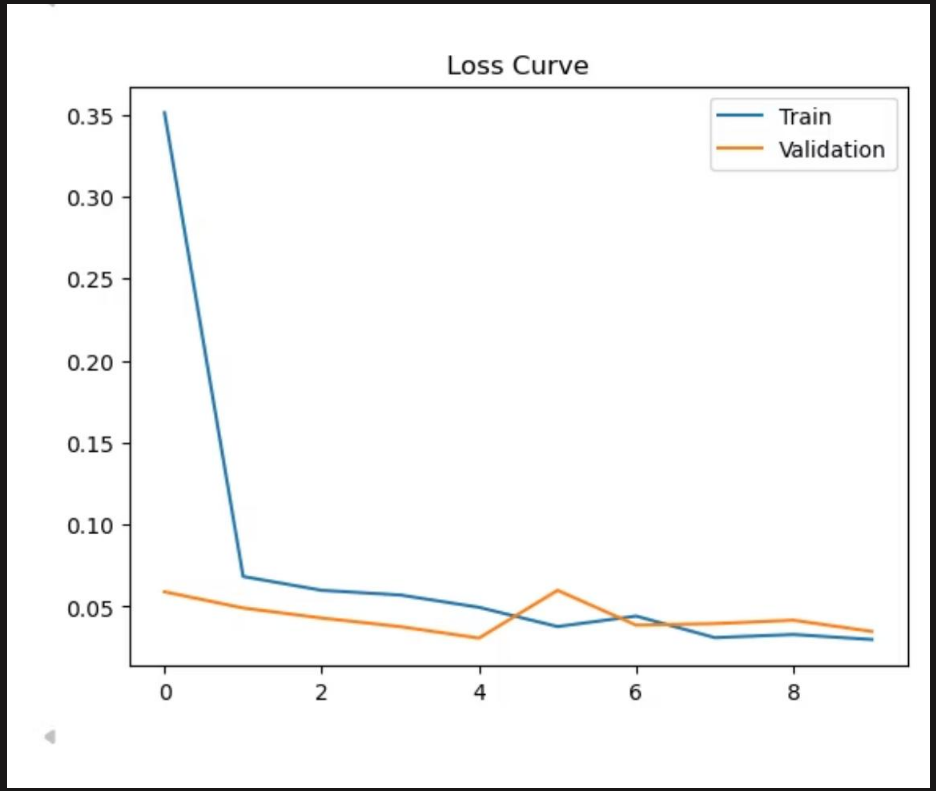
The system has a near-perfect score of 99%, showing it handles both groups equally well.

Results and Analysis



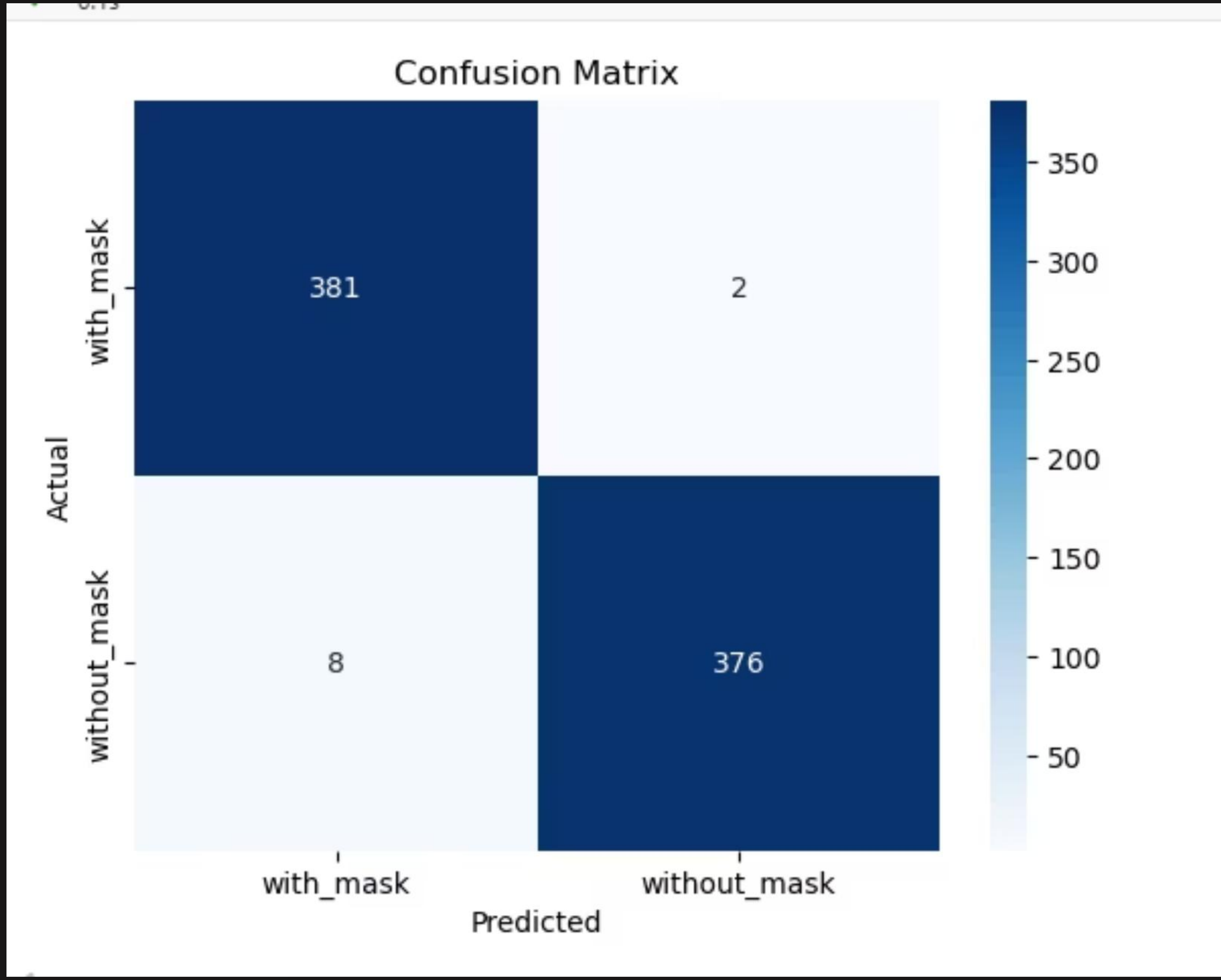
Accuracy Curve

The model learns quickly to reach very high accuracy on both training and testing data, the overall results remain stable and excellent.



Loss Curve

The blue line drops down quickly to show the model learned to make very few mistakes on the training data.the model is reliable and accurate on new information.



Confusion Matrix Insights

Many faces with masks were correctly classified.

- Incorrect classifications typically occurred due to:
- Masks only covering the chin
- Poor lighting conditions
- Faces being partially visible

Discussion and Conclusion

Performance Evaluation

The model performed well, achieving high accuracy with MobileNetV2 for fast inference.

Challenges Encountered

Low-quality images and missed small faces by the detector were primary challenges.

Future Improvements

Consider better face detectors (MTCNN/RetinaFace), longer training, more mask types, and mobile/web app deployment.

We successfully developed a deep learning system for real-time face mask detection.

Real-Time Detection Demo

Observe real-time mask detection in action on a bustling street scene.

Real-Time Detection

Google Docs

mask_detection_output.mp4

References

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

Thank You

We appreciate your time and attention today. Your engagement is invaluable as we strive for innovation.

Mask Detection Project

Presented by ELMOATASEM ALY , AHMED GAAFAR