CONTENT

ABSTRACT

The Face Mask Detection project aims to develop a machine learning model capable of accurately identifying whether individuals are wearing face masks in images. This task is especially relevant for public health scenarios where mask-wearing compliance is essential. The project involves data collection, preprocessing, model training, and evaluation. The final deliverable is a model that can classify images into "masked" and "unmasked" categories, demonstrating the potential of machine learning in supporting public health initiatives.

INTRODUCTION

This project focuses on detecting whether individuals in images are wearing face masks using machine learning techniques. The dataset contains labeled images of individuals with and without masks. The objective is to create a robust model that can correctly classify these images. The project includes steps such as data preprocessing, model selection, training, and evaluation to determine the best model for this task.

TECHNOLOGIES USED

Programming Language:

- **Python**: Chosen for its simplicity and rich libraries for machine learning and data manipulation.

Machine Learning Libraries:

- **TensorFlow/Keras**: Used for building, training, and evaluating the Convolutional Neural Network (CNN) model.
- **OpenCV**: Used for image processing tasks, including loading and resizing images.

- **Matplotlib**: Utilized for visualizing data, training performance, and model predictions.

Development Environment:

- **Google Colab**: Provides free access to powerful GPUs and a cloud-based interface for running the machine learning tasks. It allows easy collaboration and facilitates running computationally intensive tasks without local resource constraints.

DATASET INFORMATION

Dataset Description:

The dataset comprises images labeled as "with_mask" and "without_mask". It includes variations in lighting, angles, and backgrounds to simulate real-world conditions.

Features:

- **Images**: Input data consisting of pixel values.
- **Labels**: Output labels categorizing each image as "with_mask" or "without_mask".

METHODOLOGY

The methodology for face mask detection includes the following steps:

1. **Data Collection**:
    - Images are collected from a dataset stored on Google Drive. The dataset includes two categories: "with_mask" and "without_mask".
2. **Data Preprocessing**:
    - **Resizing Images**: All images are resized to 128x128 pixels to ensure uniform input size for the CNN model.
    - **Normalization**: Pixel values are normalized to the range [0, 1] by dividing by 255.0.
    - **Data Splitting**: The dataset is split into training (80%) and testing (20%) sets.
3. **Model Selection**:
    - **Convolutional Neural Network (CNN)**: A CNN is selected for this task due to its effectiveness in image classification tasks. The model architecture includes convolutional layers followed by pooling layers, dense layers, and a final sigmoid output layer.
4. **Model Training**:

- The model is trained on the training dataset with validation on the testing dataset over several epochs.

5. **Model Evaluation**:
   - The model's performance is evaluated using metrics such as accuracy, precision, recall, F1-score, and confusion matrix.
6. **Deployment**:
   - The trained model is saved and can be deployed to predict mask status on new images.

CODE SNIPPETS

Importing Necessary Libraries:

```
# Import necessary libraries
import os
import cv2
import numpy as np
from sklearn.model_selection import train_test_split
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report, confusion_matrix
```

Loading the Dataset:

```
# Define the path to the dataset in Google Drive
dataset_dir = '/content/drive/MyDrive/dataset/archive (1)/data'
```

```python
# Load and process images
for category in categories:
    path = os.path.join(dataset_dir, category)
    label = categories.index(category)  # 0 for "with_mask", 1 for "without_mask"

    for img in os.listdir(path):
        try:
            img_path = os.path.join(path, img)
            image = cv2.imread(img_path)
            if image is None:
                print(f"Failed to load image {img}")
                continue

            # To check all images are resized to 128x128 pixels
            image = cv2.resize(image, (img_size, img_size))
            data.append(image)
            labels.append(label)
        except Exception as e:
            print(f"Error loading image {img}: {e}")
```

Data Preprocessing:

```python
# Convert lists to numpy arrays
data = np.array(data, dtype='float32') / 255.0
labels = np.array(labels)

# Split the data into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(data, labels, test_size=0.2, random_state=42)
```

Model Training and Evaluation:

```python
# Building a CNN model
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(img_size, img_size, 3)),
    MaxPooling2D(pool_size=(2, 2)),

    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),

    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(1, activation='sigmoid')  # Sigmoid for binary classification
])
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning:
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```python
# Compile the model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

Saving the Model:

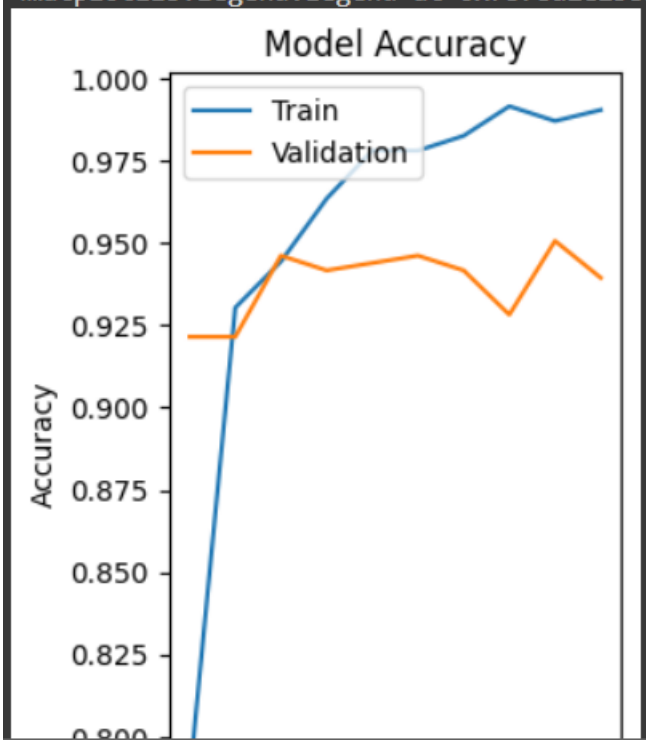
```python
# Save the model
model.save("/content/drive/MyDrive/face_mask_detector_model.keras")
```

Data Visualization

Training and Validation Performance:

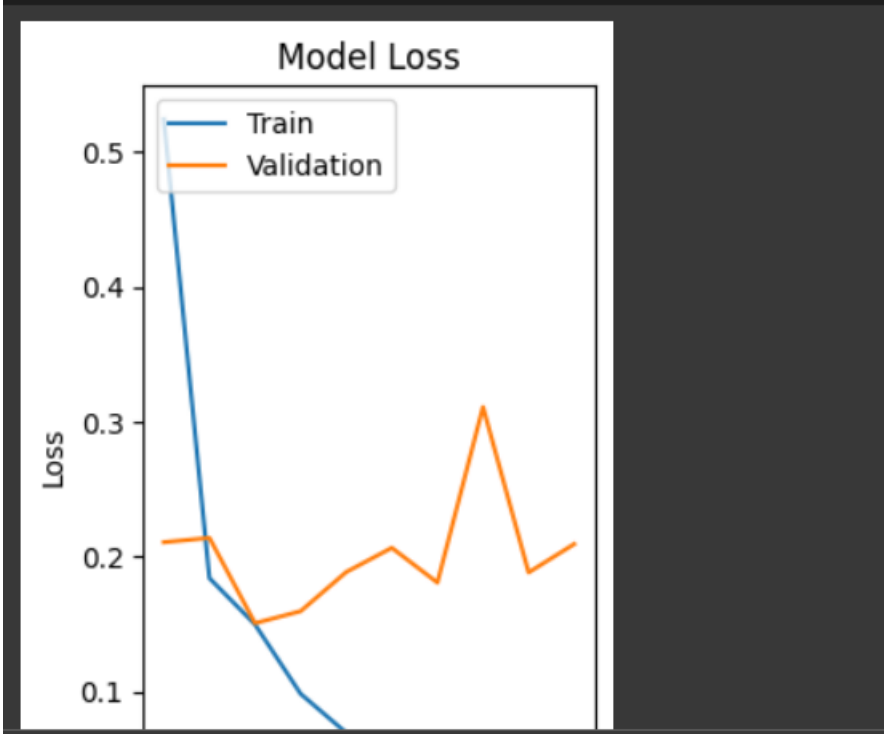
```python
# Plot training & validation accuracy values
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(['Train', 'Validation'], loc='upper left')
```

```
<matplotlib.legend.Legend at 0x7b78a2e138b0>
```



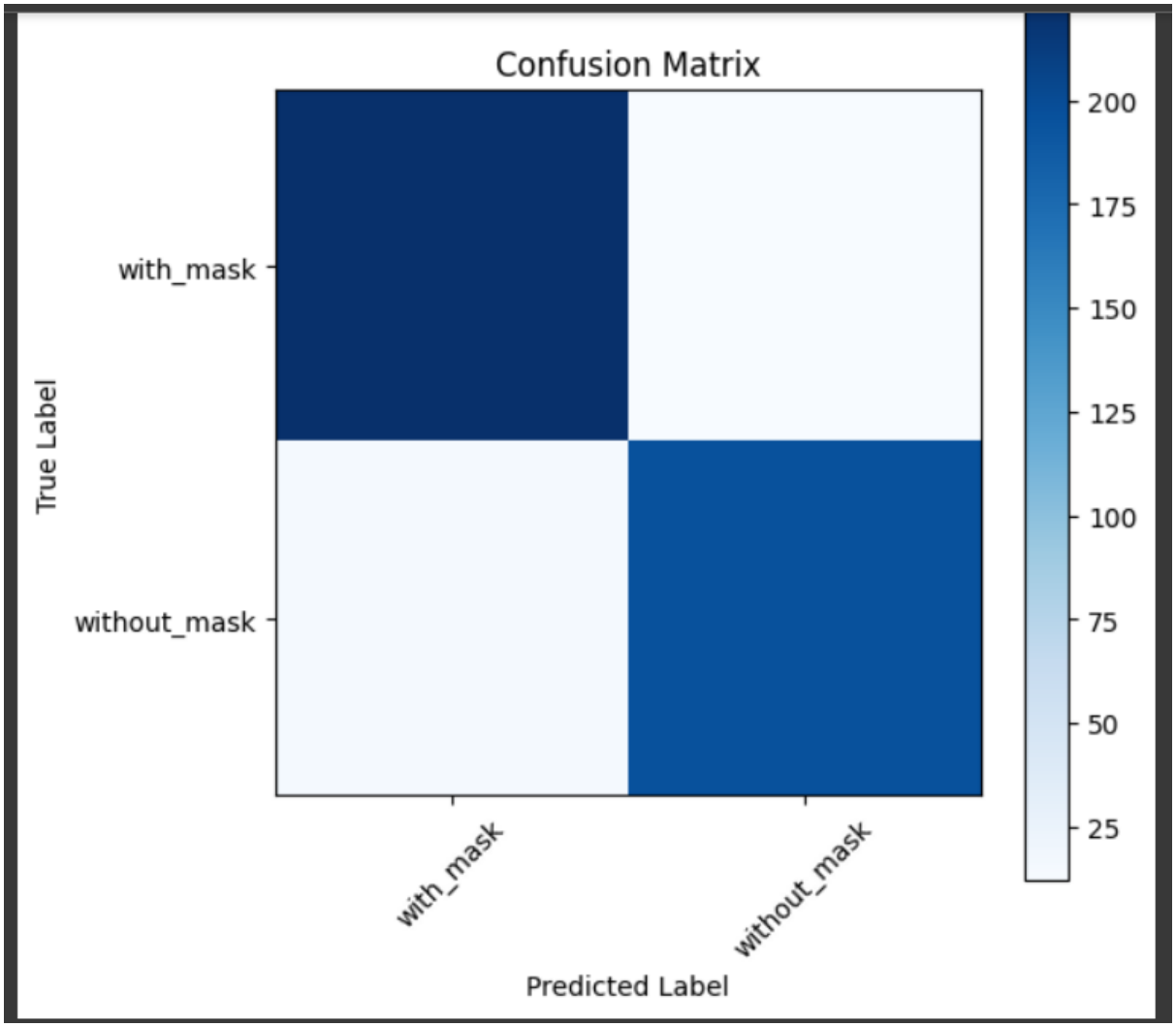
```
# Plot training & validation loss values
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(['Train', 'Validation'], loc='upper left')

plt.show()
```



Confusion Matrix:

```
plt.figure(figsize=(6, 6))
plt.imshow(cm, interpolation='nearest', cmap=plt.cm.Blues)
plt.title('Confusion Matrix')
plt.colorbar()
tick_marks = np.arange(len(categories))
plt.xticks(tick_marks, categories, rotation=45)
plt.yticks(tick_marks, categories)
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
```



RESULT

- **Accuracy**: The accuracy of the model on the test dataset.
- **Precision, Recall, F1-Score**: Detailed performance metrics.

- **Confusion Matrix**: A confusion matrix to illustrate the number of true positives, true negatives, false positives, and false negatives.

CONCLUSION

The Face Mask Detection project successfully identifies whether individuals in images are wearing masks using a CNN model. The model demonstrates high accuracy, making it suitable for deployment in public health monitoring systems. This project underscores the importance of proper data preprocessing, model selection, and evaluation in achieving reliable machine learning outcomes.