Project Proposal

Face Mask Detection using CNN

Introduction

Masks play a crucial role in protecting the health of individuals against respiratory diseases, as is one of the few precautions available for COVID-19 in the absence of immunization. With this dataset, it is possible to create a model to detect people wearing masks, not wearing them, or wearing masks improperly.

Project Scope

- **Features**
  - Mask detection from images.
  - Classification of faces as Masked or Unmasked.
- **Expected Outcomes**
  - A functional and accurate detection system.
  - High performance in terms of speed and accuracy.

Requirements

- Custom image processing techniques
- The development of the model involves leveraging Convolutional Neural Networks (CNNs) as the primary deep learning architecture.
- Neural network architecture

Technologies and Tools

- Python with libraries as TensorFlow,keras, numpy and OpenCV, sklearn, PIL
- Matplotlib to visualize performance.

Datasets

- Leverage publicly available datasets of masked and unmasked faces, obtained from Kaggle, a widely recognized platform for sharing datasets and machine learning resources.
- Collection of real-world images.

Project Pipeline

1. Data Gathering

- We collect number of images and classify to classes “mask” and unmasked”.

2. Preprocessing Data

- Resize Images: 128x128
- Support Format (JPG, PNG,JPEG)
- Convert Image To RGB
- Label Images: mask = 1, unmask = 0
- Convert Images to NumPy Array
- Normalize Data: Dividing by 255
- Shuffle Data

3. Model Design

- Convolution Neural Networks optimized for binary classification tasks to distinguish between masked and unmasked faces.

4. Evaluation

- Evaluate the model's performance using key metrics such as accuracy, data loss,data validation to gain a comprehensive understanding of its effectiveness in classification tasks.

Documentation

Face Mask Detection

The project aims to develop a real-time face mask detection system using Convolutional Neural Networks. This project will ensure compliance with mask wearing in public spaces and classify them into two categories: “with mask” and “without mask.” contributing to public health safety.

Overview

- Features
  - Mask detection from images.
  - Classification of faces as Masked or Unmasked.
- Expected Outcomes
  - A functional and accurate detection system.
  - High performance in terms of speed and accuracy.

Our pipeline consists of two steps:

1. We make a mask/no_mask prediction for each of them
2. We return an annotated image with the predictions

Team Collaboration

- Collaboration
  - We worked together closely, ensuring smooth communication and mutual support throughout the development process.
- Team Meeting
  - meeting one: We proposed the project idea and conducted a comprehensive study on the requirements needed for the project.
  - meeting two: We collected the appropriate dataset to start the project and divided the tasks, including coding and optimization.
  - meeting three: We implemented the necessary algorithms for the project and trained the model on different datasets so that it can classify new datasets as we intend.
  - meeting four: We We had a Zoom meeting before the discussion to review the entire project.

Technical Methodology

- Tools and Technologies

- Python, TensorFlow, OpenCV, and numpy, sklearn.
- Tensorflow / Keras
- Kaggel Notebook

Steps For Building Project

Collecting Data and Datasets

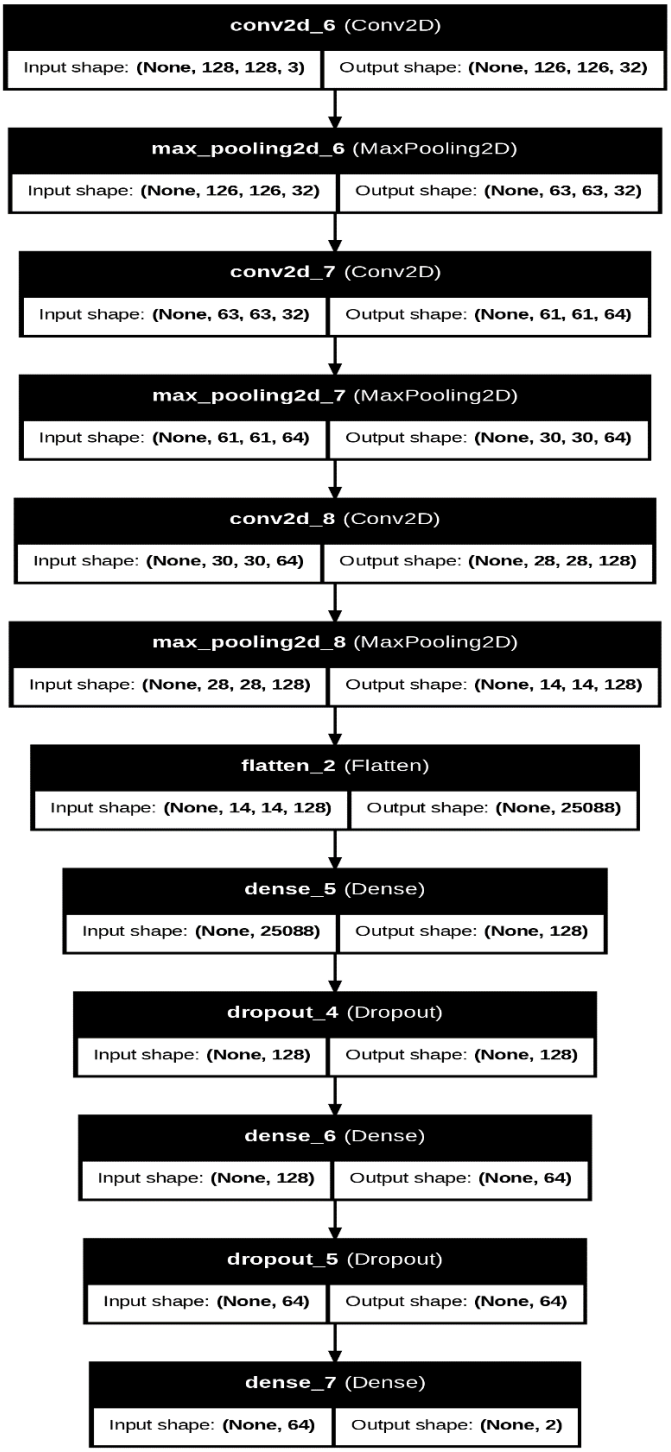
- collecting data from real-world images and supplemented it with datasets sourced from **Kaggle**: link 1,link 2, ensuring a diverse and comprehensive dataset.We used two datasets and combined them together .Datasets consisting of **more than 19,000 RGB images**, which include individuals **wearing masks** and others **without masks**.This dataset provides a solid foundation for training and evaluating our face mask detection model.

Data preprocessing

- **Resize Images**: 128x128
- **Support Format (JPG, PNG,JPEG)**
- **Convert Image To RGB**
- **Label Images**: mask = 1, unmask = 0
- **Convert Images to Numpy Array**
- **Normalize Data**: Dividing by 255
- **Shuffle Data**

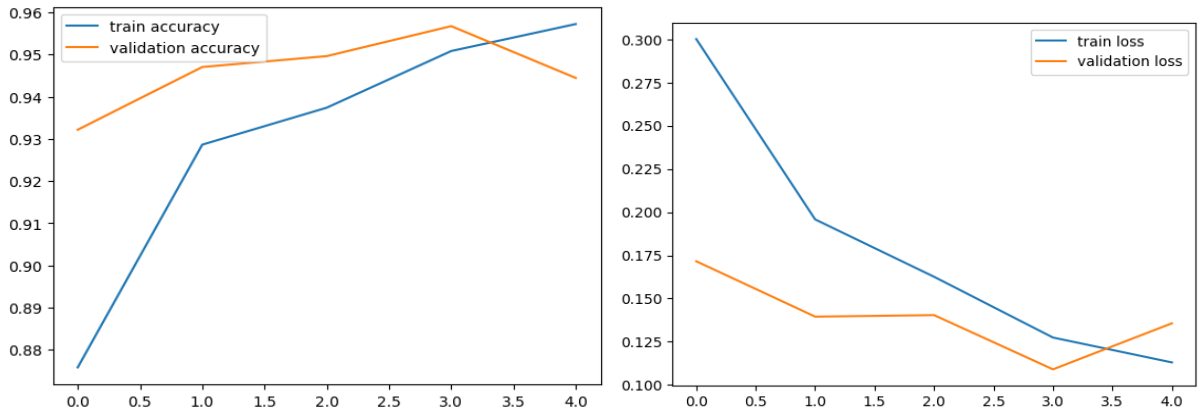
Building Model

- **Train Test Split**: 70% for training, 20% for testing , 10% for validation
- **Model Architecture**: We employed a **Convolutional Neural Network (CNN)** architecture, which optimized for binary classification.



Evaluation

```
121/121 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - acc: 0.9496 - loss: 0.1196
Test Accuracy = 0.9426208138465881
```



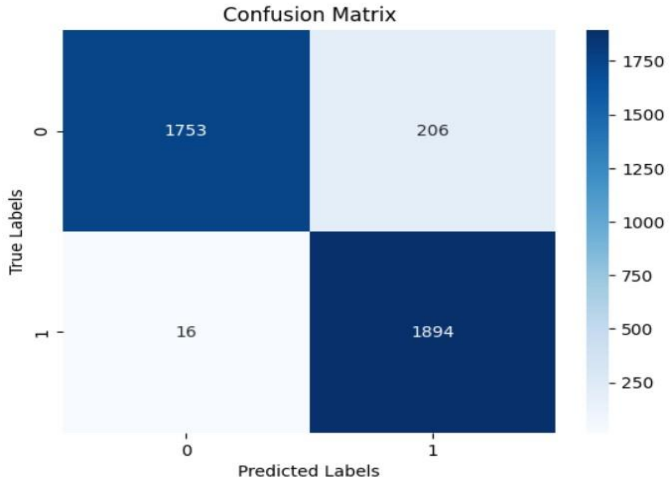
Classification Report

```
121/121 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.89      0.94      1959
           1       0.90      0.99      0.94      1910

    accuracy                           0.94      3869
   macro avg       0.95      0.94      0.94      3869
weighted avg       0.95      0.94      0.94      3869
```

Confusion Matrix



Passing input images to the model for testing