Architecture Document

Face Mask Detector

Revision No: 1.0

Last Date of Revision: 05/11/2023.

Document Version Control-

Date Issued	Version	Description	Author
05/11/2023	1.0	LLD – V- 1.0	Mahesh. A

Contents -

Abstract

By using the pre-trained of Facemask detector user can understand one is with proper facemask or not by this we can reduce the risk involved in corona virus transmission.

1 Introduction

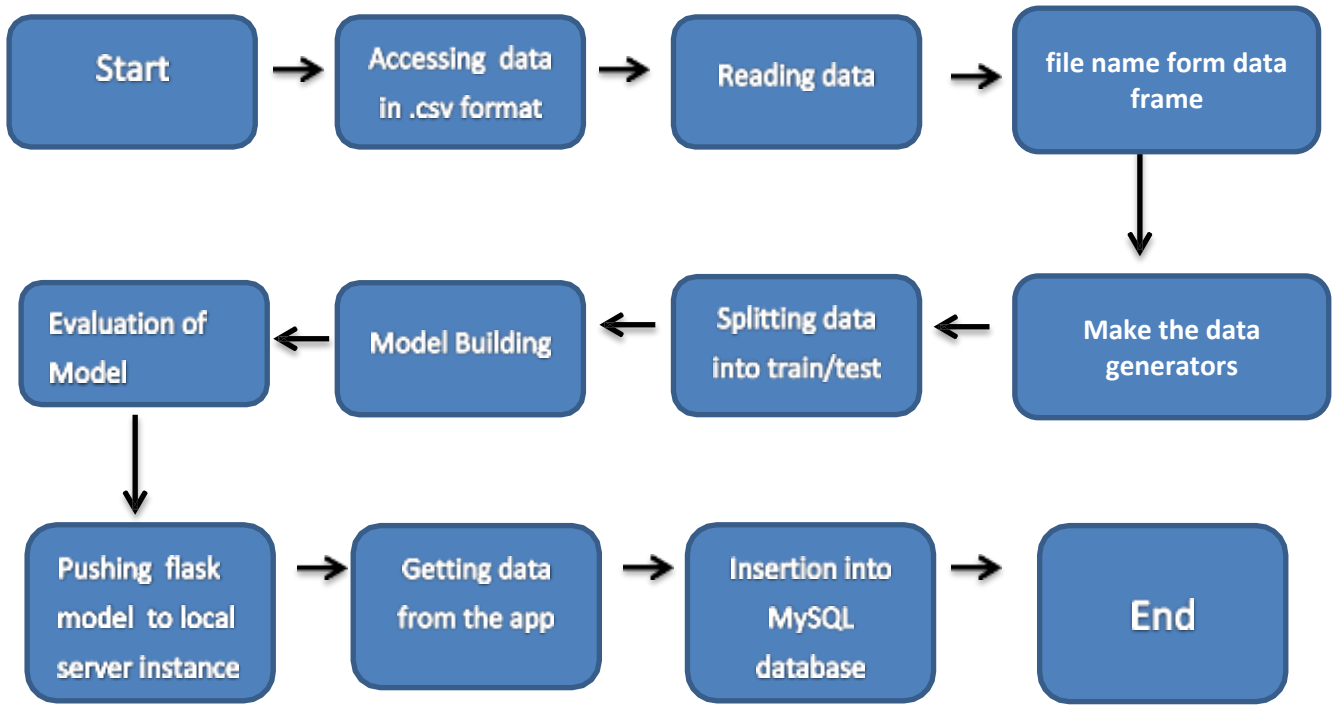
1.1. What is document required?

The goal of LLD or a low-level design document (LLDD) is to give the internal logical design of the actual programmed code for Facemask Detector. LLD describes the class relations with predictors .It describes the modules so that the programmer can directly code the program from the document.

1.2. Scope

Low-level design (LLD) is a component-level design process that follows a step-by-step r process. This process can be used for designing data structures, required software architecture, source code and ultimately, performance algorithms. Overall, the data organization may be defined during requirement analysis and then refined during data design work.

2 Architecture



3 Architecture Description

3.1 Data Accessing

We can access the data in the from the download link as available in the project.

We load the data to the framework using the pandas read function.

Attribute Information: (classes: Mask identified=1, No Mask identified=0)

file_name	labels
without_mask. (1).jpg	0
without_mask. (10).jpg	0
without_mask. (100).jpg	0
without_mask. (101).jpg	0
without_mask. (102).jpg	0

file_name	labels
without_mask. (1).jpg	0
without_mask. (10).jpg	0
without_mask. (100).jpg	0
without_mask. (101).jpg	0
without_mask. (102).jpg	0
with_mask. (280).jpg	1
with_mask. (281).jpg	1
with_mask. (282).jpg	1
with_mask. (283).jpg	1
with_mask. (284).jpg	1
with_mask. (285).jpg	1
with_mask. (286).jpg	1
with_mask. (287).jpg	1
with_mask. (288).jpg	1

3.2 Data Pre-Processing

By the usage of the data preprocessing features to get the binary output.

```
[ ] import os

[ ] image_names = os.listdir('/content/mydrive/MyDrive/DeeplearningImages')

[ ] labels = []
    for image in image_names:
      name = image.split('.')[0]
      if name == 'with_mask':
        labels.append('1')
      else:
        labels.append('0')
```

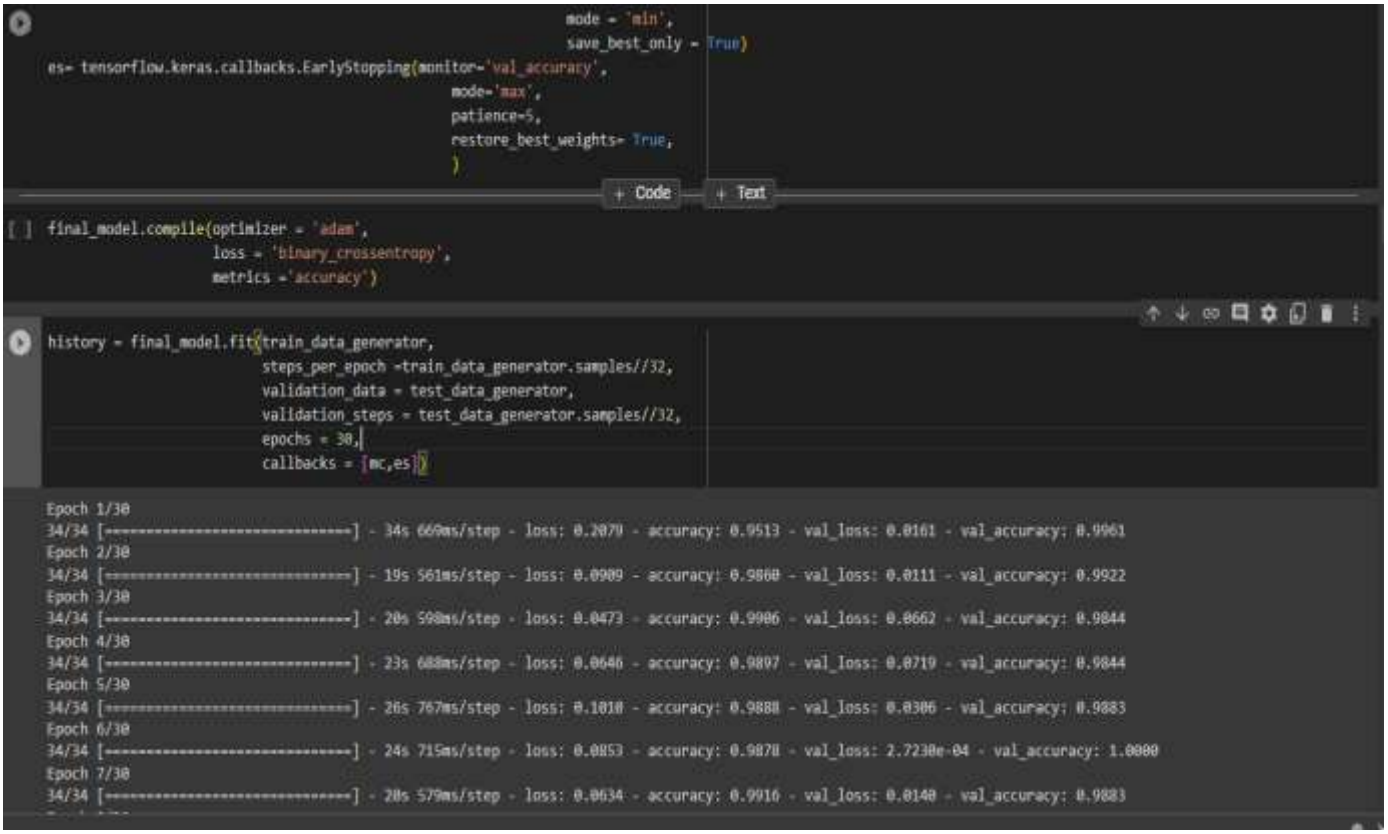
3.3 Splitting the Data

```
[ ] train_data = ImageDataGenerator(rescale = 1./255,
                                    rotation_range =20,
                                    horizontal_flip= True,
                                    vertical_flip = True)
    test_data =ImageDataGenerator(rescale =1./255)

    train_data_generator = train_data.flow_from_dataframe(dataframe = train_df,
                                                          directory='/content/mydrive/MyDrive/DeeplearningImages',
                                                          target_size =(224,224),
                                                          x_col = 'file_name',
                                                          y_col = 'labels',
                                                          color_mode = 'rgb',
                                                          class_mode = 'binary',
                                                          batch_size = 32,
                                                          seed = 42,
                                                          shuffle = True,
                                                          validate_filenames = True
                                                          )

    Found 1100 validated image filenames belonging to 2 classes.

[ ] test_data_generator = test_data.flow_from_dataframe(dataframe = test_df,
                                                        directory='/content/mydrive/MyDrive/DeeplearningImages',
                                                        target_size =(224,224),
                                                        x_col = 'file_name',
                                                        y_col = 'labels',
                                                        color_mode = 'rgb',
                                                        class_mode = 'binary',
                                                        batch_size = 32,
                                                        seed = 42,
                                                        shuffle = True,
```

We use train test split Sklearn function to split the data for training and validation.

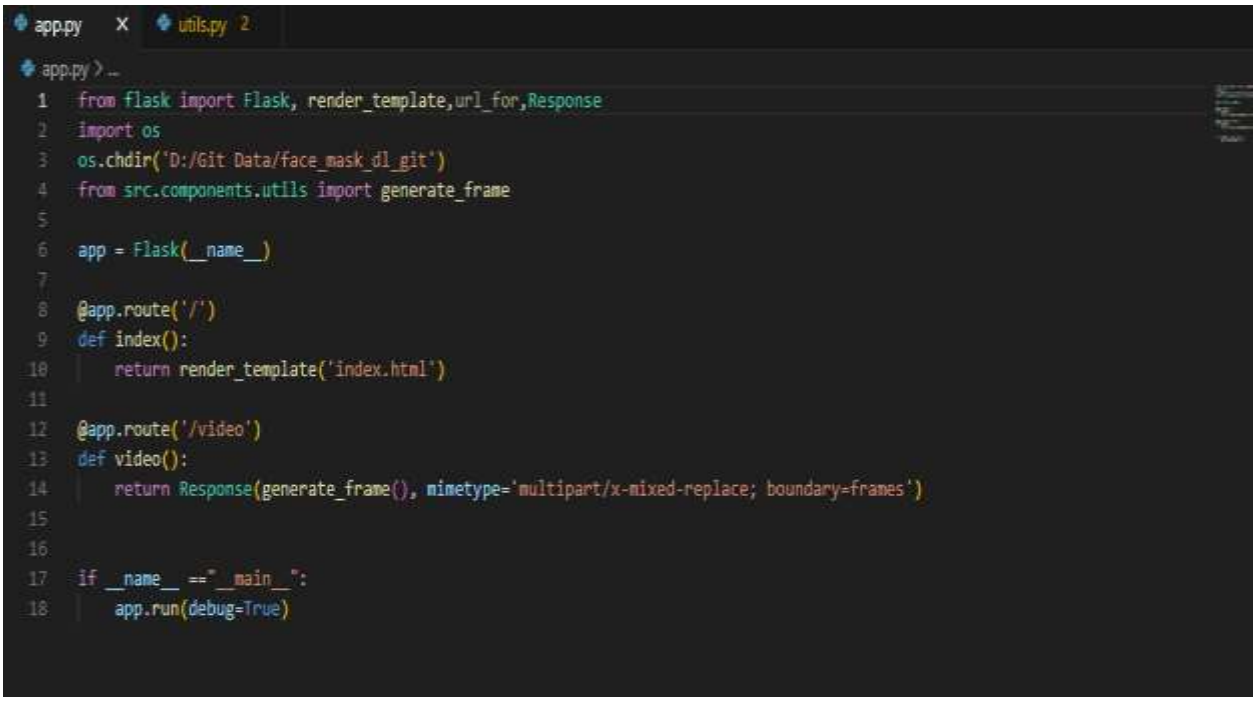
3.4 Model Building

We will deploy as many models as possible and fine tune them using GridsearchCV select the model which are performing with highest accuracy and select those models and ensemble them using voting classifier for the best model. Perform the model evaluation.



3.5 Create Front End User Module using flask

Once the model is created download and save the model and now we create GUI for front end user using the flask incorporated with HTML, CSS. Align and map the user data to the data base created. From user data create the data frame and load it to the model for the prediction the same prediction is send back to the user GUI and well saved in the data base (MySQL).



app.py | index.html | utils.py 2

templates > index.html

```
<!DOCTYPE html>
<html>
    <body>
        <h1>Live streaming</h1>
        <div>
            <img src="{{ url_for('video') }}" width="40%"/>
        </div>

        </body>
</html>
```

3.6 Testing the Model

- ✓ Verify whether the application is the loading on the local server and render server.
- ✓ Verify whether the user can access the application.
- ✓ Verify the user can see the live feed is visible
- ✓ Once it is visible check for the rectangle boxes around the face
- ✓ Check the user can get the result or prediction on the top of boxes.
- ✓ Once he gets the prediction.
- ✓ Check the data form the user and prediction from the model is loaded into the local MySQL and Cassandra data base

- ✓ Verify whether the application is the loading on the web service instance and also Render web hosting service.

LIVE HOST WEB ADDRESS: https://face-mask-detection-ggc0.onrender.com

4. Key performance indicators (KPI)

- Time and work load reduction by using the flask model.
- Compare the accuracy of model using prediction and actual results.
- Check for the wrong predictions
- If found any wrong predictions again train the model with the new data along with previous data
- Retest the model unless the productions attain the good results.