

Baku, Azerbaijan

+994 50 4918219

mahammad.ojagzada@outlook.com

**Mask Wear Detector**

Low Level Design

Technology: Deep Learning & Computer Vision

Creator: Mahammad Ojagzada

Date: 13.11.2022

|  |  |  |  |
| --- | --- | --- | --- |
| Date issued | Version | Description | Author |
| November 12, 2022 | 1.1 | First Draft | Mahammad Ojagzada |
| November 13, 2022 | 1.2 | Added unit test cases | Mahammad Ojagzada |

**Document Version Control**

Contents

[Introduction 2](#_Toc119105843)

[What is Low-Level Design Document? 2](#_Toc119105844)

[Scope 3](#_Toc119105845)

[Architecture 3](#_Toc119105846)

[Architecture Description 4](#_Toc119105847)

[Data Preparation 4](#_Toc119105848)

[Data Description 4](#_Toc119105849)

[Data Preprocessing 4](#_Toc119105850)

[Exploratory Data Analysis 4](#_Toc119105851)

[Feature Engineering 4](#_Toc119105852)

[Model Development 5](#_Toc119105853)

[Model implementation 5](#_Toc119105854)

[Unit Test cases 5](#_Toc119105855)

# Introduction

## What is Low-Level Design Document?

The goal of LLD or a low-level design document is to give the internal logical of the actual program code for Face Mask Detection model. It will explain the purpose and features of the system, the interfaces of the system, what the system will do, the constraints under which it must operate and how the system will react to external stimuli.

The main objective of the project is to detect whether a person wears a face mask or not. Transfer Learning is used in this project to get the best results with limited the dataset,

## Scope

Low-level design (LLD) is a component-level design process that follows a step-by-step refinement 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.

# Architecture

**Data Preparation**

**Model**

**Development**

# Architecture Description

## Data Preparation

### Data Description

We downloaded the dataset from github.com/cabani/MaskedFace-Net. There are 67,049 images with Correctly Masked Face Dataset (CMFD) at 1024×1024 and 66,734 images with Incorrectly Masked Face Dataset (IMFD) at 1024×1024. However, for time limit, I used only one folder of correctly and incorrectly Masked Face Dataset. Overall, i used 1143 images for training and 286 images for validation.

### Data Preprocessing

In the data preprocessing step, using ImageDataGenerator we scaled, shuffled, inserted target size, which was 150x150, and chose appropriate class mode (binary) with specified batch size.

### Exploratory Data Analysis

Using matplotlib, we visualized images with 4x4 configuration.

### Feature Engineering

In this part, datatypes of “date\_time”, “rain\_1h”, “snow\_1h”, “temp” were corrected. Values is “temp” column are in Kelvin, they are converted to Celsius for convenience. Outliers were checked using boxplot and removed from the data. Lastly, some new columns (weekday, hour, month, year) were extracted from “date\_time” column. Newly derived “hour” column is modified like “early morning”, “morning”, and etc. As “weather\_main” column contains major data, “weather\_description” feature was dropped. Moreover, most of values in “snow\_1h” and, ”rain\_1h” columns are almost 0, so they are also dropped from the data.

## Model Development

### Model implementation

As we had very few data for reliable model output, I used Transfer Learning method with InceptionV2 model. I imported local weights for the model from the API. I split the model from “mixed7” layer and merged the model with my own Functional custom layers.

Model Prediction

Having used Callbacks, the model stopped at the 2nd epoch with 100% accuracy and 2.9379e-04 on training dataset and 100% accuracy, 2.4872e-04 loss on validation dataset. At the end, I saved the as 'model\_v1.h5'.

Real time face mask detection

That was the interesting part of all coding part. I used my webcam to test the results and the model very well on real time face mask prediction very well. The important point is that using ‘haarcascade\_frontalface\_default.xml’, the model is able to identify faces.

# Unit Test cases

|  |  |  |
| --- | --- | --- |
| **Test Case Description** | **Pre-Requisite** | **Expected Result** |
| Verify whether the model identifies faces | “Haarcascade” library should be used | The model correctly identifies faces on each frame. |
| Verify whether the model correctly classifies whether a person wears a face mask or not. | Model should be trained to a high accuracy with good data validation. | The model correctly identifies faces and shows if a person wears a face mask or not. |
| Saved model works on real time detection | Application is accessible | Using webcam, the model also classifies our face. |