**Ahmed Magdy Ali**

**Mask Detection**

**Using Deep Learning Object Detection techniques**

**Table of Contents**

[**Table of Figures** 1](#_Toc90714212)

[**Problem Overview** 2](#_Toc90714213)

[**Dataset** 2](#_Toc90714214)

[**Methodology** 3](#_Toc90714215)

[1. YOLOV4 3](#_Toc90714216)

[2. SSD and EfficientDet: 4](#_Toc90714217)

[**Challenges** 5](#_Toc90714218)

[**Results** 5](#_Toc90714219)

# **Table of Figures**

[Figure 1:Output convolution No. of filters formula 3](#_Toc90714232)

[Figure 2:SSD Loss 5](#_Toc90714233)

[Figure 3:EfficientDet Loss 6](#_Toc90714234)

[Table 1:Models Results 5](#_Toc90714237)

# **Problem Overview**

During the coronavirus pandemic, people had to wear masks to lower the risks of being infected with this disease, and they were not allowed to enter any governmental departments or public places without wearing a mask.

As we live in the Artificial intelligence revolution many companies and engineers hurried up to help to get out of this crisis in many ways, so many datasets were collected and annotated to help engineers to do their job and help. So, I decided to develop a face mask detection to increase my skills in object detection and computer vision field as the dataset was available and annotated so that I can save a lot of time related to data collection and annotation, also it is easy to test the model on myself to have the freedom to experiment with anything that comes to my mind.

# **Dataset**

The Dataset consists of 1370 annotated images of people wearing face masks and others who don’t.

The annotation format for the data was two types, the first one is Pascal VOC which is XML files, the other was text file, and we have two classes “with\_mask” and “without\_mask”.

# **Methodology**

I uploaded the dataset to google drive and used google Colab to read data and build the models to be able to train them using free Colab GPU.

As the dataset was small, I used transfer learning with three models which are:

1. You Only Look Once (YOLOV4)
2. Single Shot Multibox Detector (SSD)
3. EfficientDet

## 1. YOLOV4

I cloned the YOLOV4 model of AlexeyAB darknet implementation and used its optimal pre-trained weights and used the ‘.txt’ annotation format, then altered the make file so that I can train it on GPU, then I split the data into 90% train and 10% test and created ‘.names’ file; which contains classes names and ‘.data’ file; which contains the number of classes and train and validation paths, then I edited the configuration file to change some hyperparameters like the batch size which I set to 64 and the three Yolo layers to put my number of classes which in this case only 2 and also edited the number of filters in the last convolution layers before [yolo] layers to be 21 as I have only 2 classes by this formula:

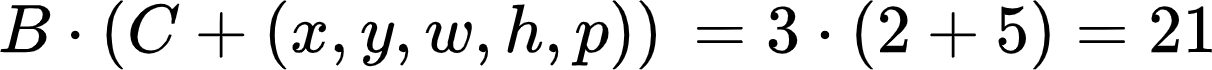


Figure 1:Output convolution No. of filters formula

Where:

B: Number of Bounding Boxes

C: Number of classes

x, y: Center of the box   
W: width of the box

h: height of the box

p: probability of the object

Then I started training and after getting an acceptable loss I used some images from google to test it and two photos of myself.

## 2. SSD and EfficientDet:

For both models I used the pascal VOC annotation format and followed the TensorFlow version 1.15 API tutorial guide for SSD and V2 for EfficientDet to implement the models.

I split the data into 90% train and 10% test then used their provided code to create the ‘label\_map.pbtxt’ file from the XML files then created the ‘train.record’ and ‘test.record’ files.

Then I cloned their models from their GitHub repository and also used their mobilenet version 2 SSD weights trained on coco dataset for SSD, while for the EfficientDet model I used D1 (alpha =1) model.

After that I edited the configuration files which I changed the number of classes into 2, and batch size to 64 for SSD and 4 for EfficientDet due to the lack of RAM in colab, then changed the checkpoint type to detection and entered the path to the train and validation data and the checkpoint.

Then I run the ‘model\_main.py’ script of TensorFlow to train the model, and after training I exported the inference graph and used it to test the model on some photos I downloaded from google and 2 photos of myself.

# **Challenges**

* I trained the models on Google Colab using free GPU, which has limitations of the usage as I had to train the models for 6 hours only each day, which took so much time for each model to train enough to get acceptable results, however it is not the best results.
* At first I tried to use EfficientDet 2,3 but the Colab crashed due to the limitation of available RAM, so I decided to use D1 which worked with batch size of 4.

# **Results**

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Iterations | Loss | Hours |
| YOLOV4 | 1300 | 1.29 | 18 hours |
| SSD | 21409 | 4.64 | 18 hours |
| EfficientDet D1 | 9200 | 0.33 | 18 hours |

Table 1:Models Results

Chart, line chart, histogram

Description automatically generated

Figure 2:SSD Loss

Chart, line chart

Description automatically generated

Figure 3:EfficientDet Loss