2020/2021

# FACE-MASK DETECTOR

## USING CONVOLUTION NEURAL NETWORK

FATIMAH NASSER
PRESENTED TO DR. MOHAMMAD AOUDE
Mini project

# Table of Contents

## General introduction:

In the past two years, the world has witnessed a pandemic known as the Corona Virus (covid-19). The Corona virus is a dangerous virus that is known for causing serious respiration problems, high fever, muscles and back pain; and when it's extreme it may lead to death.

Even though not many have died because of the virus, it has caused serious damages to our human bodies as well as our life routines in general.

In the early stages of an infection the virus is able to deceive the body. One might be holding the virus and transmit it without him knowing; causing anonymous wide spreading of the virus.

That’s why the World Health Organization has announced a group of precautions and health instructions to follow in order to maintain healthy body and stray away from the virus.

Wearing a mask is one of the most important instructions that one should not take lightly. However, some individuals do not follow the W.H.O precautions and consequently neglect wearing a mask; claiming that they don't feel sick and it's of no importance.

Our project deals with this problem by detecting if the individual is wearing a mask or not. How is it possible though?

With the help of convolution neural networks, we were able to implement machine deep learning and recognize if the individual is wearing a mask or not.

In the report you will go through a general description of the convolution of neural networks, and how machine learning has helped detect if a person is wearing the mask or not. Then there will be a brief overview of the project result to be discussed and reach a conclusion.

# Chapter 1: CNN- convolution neural networks

## 1.1. Introduction:

This chapter represents the foundation stone of the project. The name "convolutional neural network" indicates that the network employs a mathematical operation called convolution. Convolutional networks are a specialized type of neural networks that use convolution in place of general matrix multiplication in at least one of their layers. The CNN architecture has several layers that are going to be introduced in this chapter.

## 1.2. CNN architecture:

A convolutional neural network consists of an input layer, hidden layers and an output layer. In any feed-forward neural network, any middle layers are called hidden because their inputs and outputs are masked by the activation function and final convolution. In a convolutional neural network, the hidden layers include layers that perform convolutions.

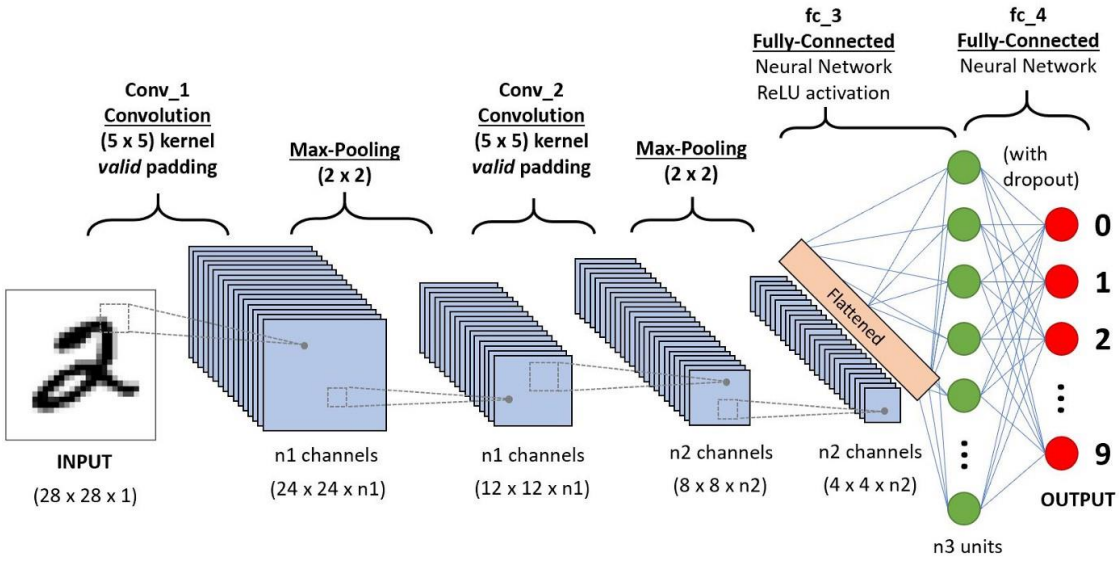


Figure1- CNN layers

### Convolution Layer:

Convolution is the first layer to extract features from an input image. Convolution preserves the relationship between pixels by learning image features using small squares of input data. It is a mathematical operation that takes two inputs such as image matrix and a filter or kernel.

- An image matrix (volume) of dimension **(h x w x d)**
- A filter **($f_h$ x $f_w$ x d)**
- Outputs a volume dimension **(h - $f_h$ + 1) x (w - $f_w$ + 1) x 1**

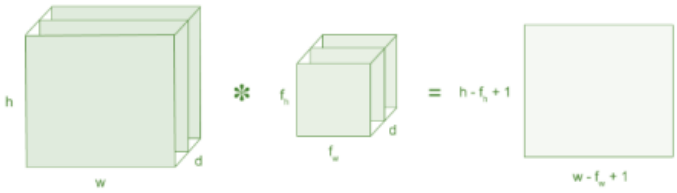


Figure 2- Image matrix multiplies kernel or filter matrix

## Activation Function:

There are several nonlinear functions such as tanh, sigmoid, or ReLU. Most of the data scientists use ReLU since performance wise ReLU is better than the other two. ReLU's purpose is to introduce non-linearity in our ConvNet. Since, the real-world data would want our ConvNet to learn would be non-negative linear values.

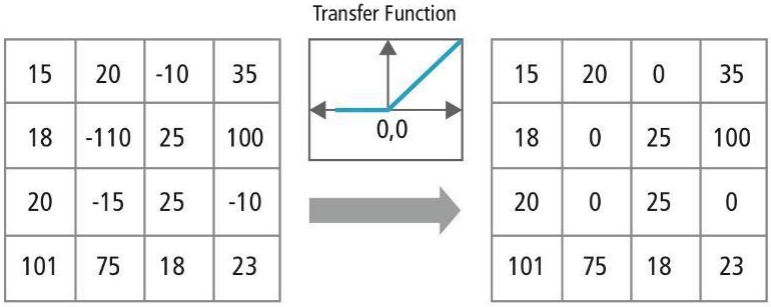


Figure 3- ReLU operation

## Pooling Layer:

Pooling layers section would reduce the number of parameters when the images are too large. Spatial pooling also called subsampling or down sampling which reduces the dimensionality of each map but retains important information. Spatial pooling can be **Max pooling** where it takes the largest element from the rectified feature map.

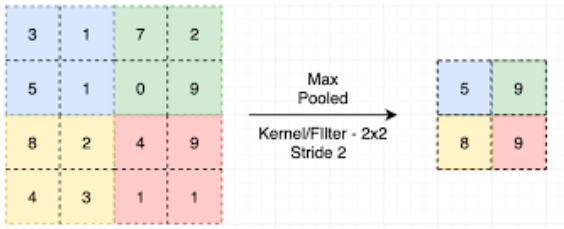


Figure 5- Max Pooling

### Fully Connected Layer:

The layer we call as FC layer, we flattened our matrix into vector and feed it into a fully connected layer like a neural network.

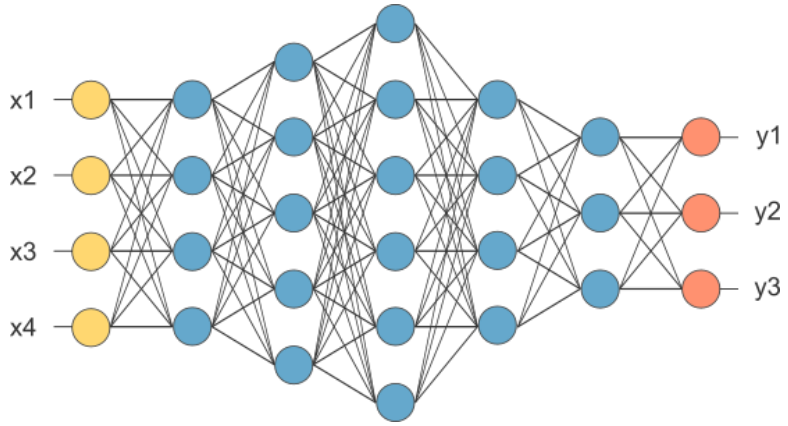


Figure 6- After pooling layer, flattened as FC layer

In the above diagram, the feature map matrix will be converted as vector (x1, x2, x3, …). With the fully connected layers, we combined these features together to create a model.

## 1.3. Conclusion

To sum up, convolutional neural networks use learnable filters in its different layers; the convolutional and pooling layers. The input image will go through more than one layer in order for the CNN to extract a feature from it. The output of those layers will then be fully connected, we will be able to have a good and accurate decision.

# Chapter 2: Machine Training

## 2.1. Introduction:

Machine learning algorithms build a model based on sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to do so.

## 2.2. Dataset:

The dataset used here has a good number of pictures taken before hand. This dataset consists of set of images of people with masks and has around 1915 images, and another without masks with 1918 images.

We proceed to define a DIRECTORY for the set, and CATEGORIES with the 2 desired categories “with mask” and “without mask”.

The next step is to go over the whole set and load image by image, with doing frame adjustments to each; we try to feed the requirements of our CNN.

## 2.3. Requirements:

- ✓ Targeted image size will be adjusted to become 224x224
- ✓ We will need to convert the image into an array in order to work with it
- ✓ Preprocess the image;
  The preprocess_input function is meant to adequate your image to the format the model requires.
- ✓ Define “Data” as a list and add the modified images to it. We also define with the data another set and put our pre-defined categories in it and name that list as “Label”

We cannot use the data as it is. We need to go ahead and binaries the data to be able to give it to the machine. Label Binarizer is an SciKit Learn class that accepts Categorical data as input and returns an Numpy array. Unlike Label Encoder, it encodes the data into dummy variables indicating the presence of a particular label or not.

The next step is to apply fit_transform which means to do some calculation and then do transformation. It is used on the training data so that we can scale the training data and also learn the scaling parameters of that data. ... These learned parameters are then used to scale our test data.

Finally, we convert a class vector (integers) to binary class matrix;

| | target |
|---|---|
| 1 | Iris-setosa |
| 2 | Iris-virginica |
| 3 | Iris-setosa |
| 4 | Iris-versicolor |
| 5 | Iris-virginica |

Apply One-hot Encoding →

| | Iris-setosa | Iris-versicolor | Iris-virginica |
|---|---|---|---|
| 1 | 1 | 0 | 0 |
| 2 | 0 | 0 | 1 |
| 3 | 1 | 0 | 0 |
| 4 | 0 | 1 | 0 |
| 5 | 0 | 0 | 1 |

Figure 7-iris example

## 2.4. Model Construction:

MobileNets are a family of mobile-first computer vision models for TensorFlow, designed to effectively maximize accuracy while being mindful of the restricted resources for an on-device or embedded application.

MobileNets are small, low-latency, low-power models parameterized to meet the resource constraints of a variety of use cases. They can be built upon for classification, detection, embeddings, and segmentation, similar to how other popular large-scale models, such as Inception, are used.

Counting depth wise and pointwise convolutions as separate layers, MobileNet has 28 layers.

On top of our base model the following has been added:

- AVERAGEPOOLING2D of pooling size 7x7
- Flattening Layers
- Use Activation function: RELU
- Drop function to avoid overfitting of model; set to 0.5

## 2.5. Network Training:

To train the model a number of epochs is needed. one epoch is the act of going through the dataset forward and backward. In this project a total of 20 epochs has been used.

## 2.6. Live-stream processing:

After the training of the machine, we got our model that we are going to use in our video stream code.

The plan is that once we get the video livestream from our camera, we will search for the “face” of the individual; the face will be our only interest. This can be achieved with the use of cascade classifier which is already trained, and it can be applied to a region of an image and detect the object in question (used haarcascade_frontalface_default.xml imported from opencv library).

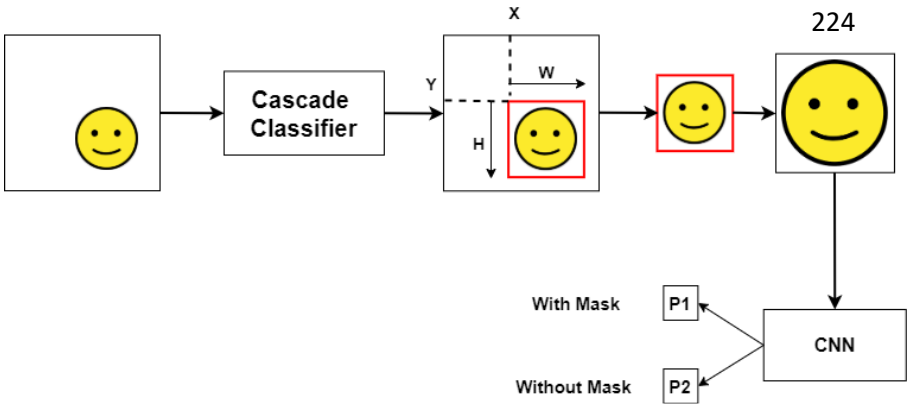


Figure 8- live-stream process

We then reframe the obtained desirable image, the face, and we resize the photo accordingly. In our code we make our frame a 224x224 to suite the CNN as explained before. After we isolate the face that we need to apply our model on, we go ahead and start our convolutional network to detect and make

predictions. The result will be either a “with mask” or a “without mask” one. We also load our model that we created in the training part of our project.

Our next step will be to define a dictionary that marks off the mentioned two categories: “with mask” and “without mask”.

Webcam will thereby be launched, and draws a box around the face it detects; you will be able to see a red box for no mask with accuracy percentage and a green one for mask results.

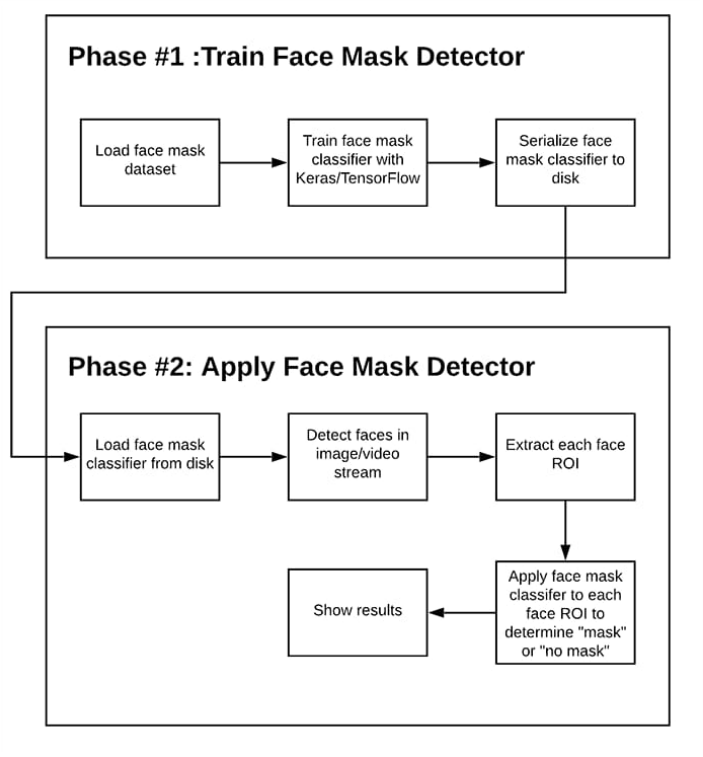


Figure 9- FlowChart

## 2.7. Conclusion:

Through this chapter we were able to understand and illustrate how the whole program is functioning and connected together. In addition, applying the model on livestream video needs few steps to get results of mask or without mask.

# Chapter 3: Result and discussion

## 3.1. introduction:

After understanding the project's concept in the previous chapter, now we will see the results obtained after running the program.

## 3.2. Machine training:

As mentioned before in this project we use a total of 20 epochs to train the machine. And as a consequence, we get the following:

```
Epoch 16/20
95/95 [==============================] - 159s 2s/step - loss: 0.0308 - accuracy: 0.9904 - val_loss: 0.0274 - val_accuracy: 0.9922
Epoch 17/20
95/95 [==============================] - 155s 2s/step - loss: 0.0218 - accuracy: 0.9952 - val_loss: 0.0322 - val_accuracy: 0.9922
Epoch 18/20
95/95 [==============================] - 153s 2s/step - loss: 0.0294 - accuracy: 0.9911 - val_loss: 0.0261 - val_accuracy: 0.9922
Epoch 19/20
95/95 [==============================] - 152s 2s/step - loss: 0.0350 - accuracy: 0.9893 - val_loss: 0.0311 - val_accuracy: 0.9922
Epoch 20/20
95/95 [==============================] - 154s 2s/step - loss: 0.0281 - accuracy: 0.9917 - val_loss: 0.0284 - val_accuracy: 0.9922
[INFO] evaluating network...
              precision    recall  f1-score   support

   with_mask       0.99      0.99      0.99       383
without_mask       0.99      0.99      0.99       384

    accuracy                           0.99       767
   macro avg       0.99      0.99      0.99       767
weighted avg       0.99      0.99      0.99       767

[INFO] saving mask detector model...
```

Figure 10- training phase

Let's take for example the last epoch (20/20), we can understand from it the several points. For, every epoch gives a value for the loss; a scalar value that we attempt to minimize during our training of the model the lower the loss the closer our predictions are to the true labels. Loss of 0.0281 in figure 10 epoch 20. This loss is how much the model fits the data that we have. Validation loss is also computed; this loss is more important since it gives the error percentage in labeling the testing part of the dataset. Validation loss should be less than the loss value for accurate prediction in a new image not already found in our dataset.

It also provides knowledge for the accuracy, 0.9917 for epoch 20, which is pretty high. Notice also the validation accuracy is evaluated; 0.9922 in epoch 20.

## 3.3. result's plot:

We then use MATPLOTLIB to plot the results and analyze the results. (Commanded the program to plot this while training).

As the plot below shows, we have a high accuracy that reaches at the end of training an approximate value of 100% verifying the good model generation.

Not only did the model give high accuracy, but also the losses are significantly very low.


Figure 11- epoch results graph

## 3.4. Live-Stream Video

The webcam will be activated, presenting a livestream video that will display the results with the generated model help. Multiple individuals can be on same livestream and the model will still analyze each case individually with displaying accuracy percentage above each face. The individual with the mask will be specified with a green box around his face, while with the one who is not wearing a mask red box will be drawn.

## 3.5. MobileNetv2 Supremacy

It was explained in previous chapters that the mobilenet is a pre-trained architecture.

Mobilenet introduces two simple global hyper-parameters that efficiently trade-off between latency and accuracy. The mobile net hyper-parameters allow the model builder to choose the right sized model for their application based on the constraints of the problem.

Depth wise Separable Convolutions are a key building block aiming to reduce the size while maintaining the efficiency. The basic idea is to replace a full convolutional operator with a factorized version that splits convolution into two separate layers.

- The first layer is called a depth wise convolution; it performs lightweight filtering by applying a single convolutional filter per input channel.
- The second layer is a 1x1 convolution, called a pointwise convolution, which is responsible for building new features through computing linear combinations of the input channels.

It is worth mentioning that mobilenet has been proven to be among the best in image vision recognition tasks. Nowadays, mobilenet is gaining more popularity in this domain.

## 3.6. How Dataset Affects the Modeling

What specializes our model is that we used a dataset that contains actual images taken for people wearing a mask and ones who aren't. Some model builders use a dataset where the with mask class is being artificially generated. This approach is possible but will give a less efficient model.

Figure 12- artificially generated

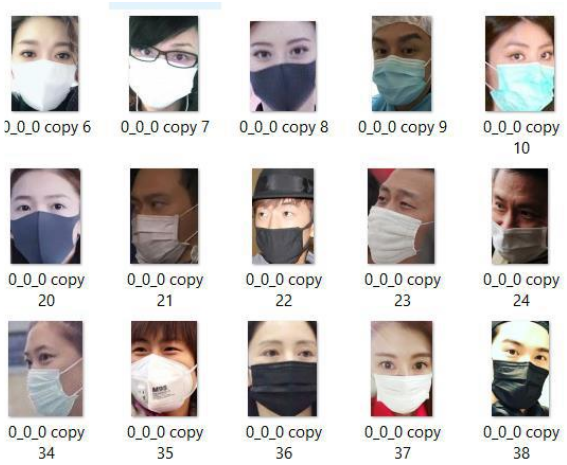


Figure13- real masks

## 3.7. Conclusion

After training the machine, and getting each epoch results of loss and accuracy. The plot generated gives an idea of the whole process' accuracy and how were the data lost. As a final result, the model was used to recognize multiple case from a livestream video. Also, Mobilenet played an important role to have such model too. finally, the chosen dataset with the quite a large number of images with real mask led to have a good learning.

## General Conclusion

In a nutshell, Earth today is troubled with the Corona virus; and it is suffering more as the days are passing.

The prevention of the virus' spreading can be established by different means; namely by wearing a mask. Our project proposes a solution to monitor the actual commitment and see if individuals are wearing the mask as should, or they are not following the W.H.O. health instructions. This was possible by machine deep learning using convolution neural network.

The CNNs are a special type of the simplest artificial neural networks. They have special layers called convolutional layers and pooling layers that allow the network to encode certain images properties. An example of a simple convolutional neural network architecture consists of an input layer at first followed by sequence of convolutional, pooling, and fully-connected layers.

With this knowledge on CNNs we were able to train our machine and generate a good, accurate model. For starters, an important requirement will be a good dataset to use while training and testing. A dataset is considered good when the number of images is sufficient and has a variety of angles and positions to compare with. Our dataset has satisfied the two desired necessities: (mask (around 1915 image) and no mask (around 1918 images)). The model was also built with the help of MobileNetv2 gives high accuracy results with minimal losses.

Our model's importance and usage appears in daily activities and routines. The commitment in wearing a mask will now be a must in several crowded areas; like the mall, the supermarkets, job companies, as well as the universities. Furthermore, we seek to make the society gain back its normal life with minimal covid spreading; hence returning to a better economical, psychological, and a safe life.

Still, the model can be further improved and have 3 labels instead of 2. We can have a third possibility: mask worn incorrectly. This will acquire images of people putting the mask in a wrong position (like under their noses) in our new dataset.

## References:

[1] MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam

[2] Mobilenetv2 source code. Available from https://github.com/tensorflow/ models/tree/master/research/slim/ nets/mobilenet. 1

Mobilenetv2 source code. Available from https://github.com/tensorflow/ models/tree/master/research/slim/ nets/mobilenet. 1

[3] https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148

[4] Deep Learning for Computer Vision with Python Dr.Adrian Rosebrock