EC3066D – ARTIFICIAL INTELLIGENCE: THEORY AND PRACTICE

COURSE PROJECT REPORT

FACE MASK DETECTION USING CNN ARCHITECTURE

MIRSAB MARWAN K

B180638EC

Department of Electronics & Communication Engineering, National Institute of Technology Calicut

mirsab_b180638@nitc.ac.in

MOHAMMAD FAIZ T

B180698EC

Department of Electronics & Communication Engineering, National Institute of Technology Calicut

mohamedfaiz_b180698ec@nitc.ac.in

RAHUL K R

B180664EC

Department of Electronics & Communication Engineering, National Institute of Technology Calicut

rahul_b180664@nitc.ac.in

PROBLEM STATEMENT:

The COVID-19 pandemic has reshaped our life and it is important to wear face mask in public and even in home in order to prevent the spread of the virus. But the truth is not everyone is wearing the mask properly and it is still a threat to the society. Our project demonstrates how a Convolutional Neural Network (CNN) can detect if a person is wearing a face mask or not. The applications of this method may be very helpful for the prevention and the control of COVID-19 and other similar diseases that may be occur in future as it could be used in public places like airports, shopping malls etc. Detect a face and check whether a person is wearing a mask or not is a classification problem.We have to classify the images between 2 discrete classes: The ones that contain a face mask and the ones that do not.

OVERVIEW OF PROJECT WORK:

- Training Architectures with sample data
- Compare the performance of models and determining best model for facemask detection.
- Train best model for different epochs and reach out into a good final model.
- Use model trained by a large dataset collected to detect facemask by real time video streaming. (Face detection using Haar feature-based cascade classifiers)

COMPARISON OF ARCHITECTURES WITH SAMPLE DATA:

Determining a good architecture for training a model is difficult task. So we have tried lots of commonly using CNN architectures and compared different parameters by training with a sample dataset of smaller size.

Specifications of dataset used for comparison:

Train

With mask – 200 images

Without mask – 200 images

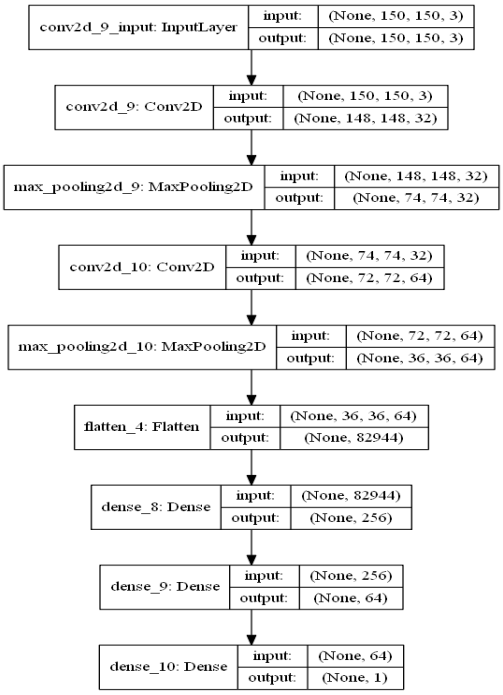
Test

With mask – 50 images

Without mask – 50 images

All photos are of different size and dimension and it is converting into fixed target size before training.

Model – 1




```
model1 = Sequential()

model1.add(Conv2D(32, (3, 3),
activation='relu', input_shape=(150,
150,3)))

model1.add(MaxPooling2D())

model1.add(Conv2D(64, (3, 3),
activation='relu'))

model1.add(MaxPooling2D())

model1.add(Flatten())

model1.add(Dense(256, activation='relu'))

model1.add(Dense(64, activation='relu'))

model1.add(Dense(1, activation='sigmoid'))
```

Model – 1 summary:

```
Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_20 (Conv2D)           (None, 148, 148, 32)      896
_________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 74, 74, 32)        0
_________________________________________________________________
conv2d_21 (Conv2D)           (None, 72, 72, 64)        18496
_________________________________________________________________
max_pooling2d_13 (MaxPooling (None, 36, 36, 64)        0
_________________________________________________________________
flatten_4 (Flatten)          (None, 82944)             0
_________________________________________________________________
dense_8 (Dense)              (None, 256)               21233920
_________________________________________________________________
dense_9 (Dense)              (None, 64)                16448
_________________________________________________________________
dense_10 (Dense)             (None, 1)                 65
=================================================================
Total params: 21,269,825
Trainable params: 21,269,825
Non-trainable params: 0
_________________________________________________________________
```

Here we trained data by splitting data into 16 batches and with a fixed size of (150,150) and trained for 10 epochs. It is a normal sequential model. This architecture has about 21,000,000 parameters. Here the model is incrementing the number of parameters rapidly. We have tried with different activation functions and filter number and kernel sizes.

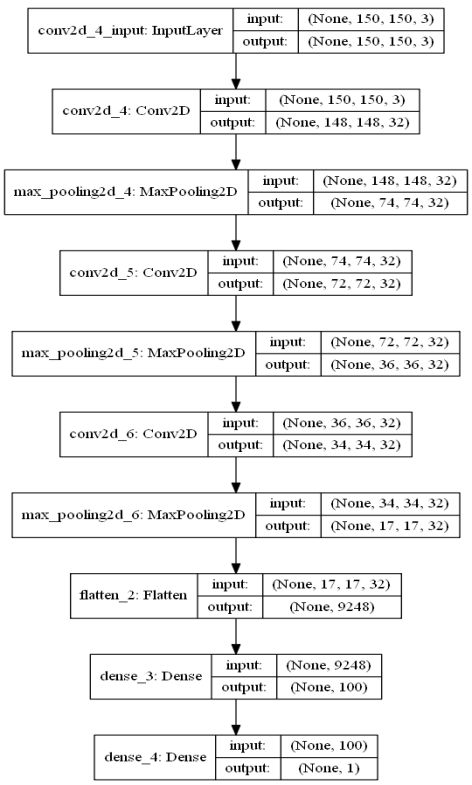
Model – 1 training details:

```
Found 400 images belonging to 2 classes.
Found 100 images belonging to 2 classes.
Epoch 1/10
25/25 [==============================] - 22s 886ms/step - loss: 0.6084 - accuracy: 0.7675 - val_loss: 0.0333 - val_accuracy: 0.
9300
Epoch 2/10
25/25 [==============================] - 14s 556ms/step - loss: 0.1997 - accuracy: 0.9375 - val_loss: 0.0445 - val_accuracy: 0.
9100
Epoch 3/10
25/25 [==============================] - 14s 551ms/step - loss: 0.1528 - accuracy: 0.9525 - val_loss: 1.7713 - val_accuracy: 0.
9300
Epoch 4/10
25/25 [==============================] - 12s 492ms/step - loss: 0.1217 - accuracy: 0.9475 - val_loss: 0.5189 - val_accuracy: 0.
9200
Epoch 5/10
25/25 [==============================] - 12s 469ms/step - loss: 0.0846 - accuracy: 0.9725 - val_loss: 0.0034 - val_accuracy: 0.
9500
Epoch 6/10
25/25 [==============================] - 12s 480ms/step - loss: 0.0884 - accuracy: 0.9775 - val_loss: 0.5504 - val_accuracy: 0.
9200
Epoch 7/10
25/25 [==============================] - 12s 472ms/step - loss: 0.1620 - accuracy: 0.9400 - val_loss: 0.1350 - val_accuracy: 0.
9600
Epoch 8/10
25/25 [==============================] - 12s 478ms/step - loss: 0.0885 - accuracy: 0.9650 - val_loss: 0.0010 - val_accuracy: 0.
9600
Epoch 9/10
25/25 [==============================] - 13s 524ms/step - loss: 0.0740 - accuracy: 0.9675 - val_loss: 0.0066 - val_accuracy: 0.
9800
Epoch 10/10
25/25 [==============================] - 13s 514ms/step - loss: 0.0745 - accuracy: 0.9775 - val_loss: 1.0036 - val_accuracy: 0.
9600
```

Here thevalues correspond to finally trained epochs are:

Loss: 0.0740, train accuracy: 0.9775, test loss: 1.0036, Test accuracy: 0.9600

Model – 2




```
model2 = Sequential()

model2.add(Conv2D(32, (3, 3),
activation='relu', input_shape=(150,
150,3)))

model2.add(MaxPooling2D())

model2.add(Conv2D(32, (3, 3),
activation='relu'))

model2.add(MaxPooling2D())

model2.add(Conv2D(32, (3, 3),
activation='relu'))

model2.add(MaxPooling2D())

model2.add(Flatten())

model2.add(Dense(100, activation='relu'))

model2.add(Dense(1, activation='sigmoid'))
```

Model – 2 summary:

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 148, 148, 32)      896
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 74, 74, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 72, 72, 32)        9248
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 36, 36, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 34, 34, 32)        9248
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 17, 17, 32)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 9248)              0
_________________________________________________________________
dense_1 (Dense)              (None, 100)               924900
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 101
=================================================================
Total params: 944,393
Trainable params: 944,393
Non-trainable params: 0
```

Here also we trained data by splitting data into 16 batches and with a fixed size of (150,150) and trained for 10 epochs.But this model does not increment parameters rapidly as model-1. And kernel size kept remains as it is in the model -1. It is a commonly used normal architecture with relu activation at each layer and sigmoid activation at output layer. This architecture has about 945,000 parameters.

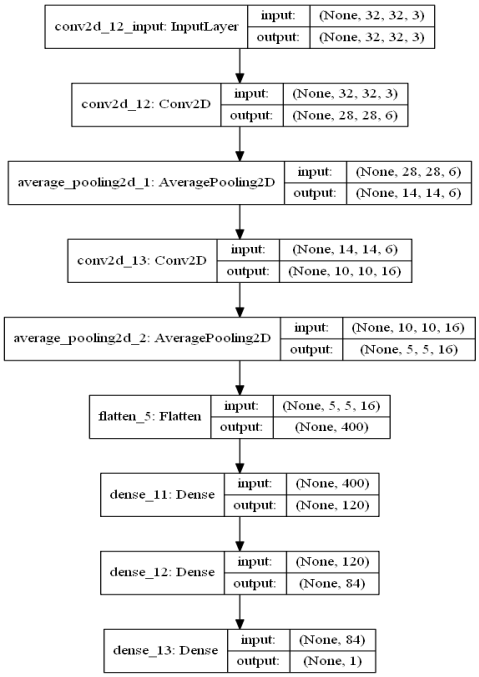
Model – 2 training details:

```
Found 400 images belonging to 2 classes.
Found 100 images belonging to 2 classes.
Epoch 1/10
25/25 [==============================] - 8s 339ms/step - loss: 0.5582 - accuracy: 0.7725 - val_loss: 0.1762 - val_accuracy: 0.8
900
Epoch 2/10
25/25 [==============================] - 7s 267ms/step - loss: 0.2453 - accuracy: 0.9175 - val_loss: 0.0845 - val_accuracy: 0.8
900
Epoch 3/10
25/25 [==============================] - 6s 255ms/step - loss: 0.1498 - accuracy: 0.9475 - val_loss: 0.0648 - val_accuracy: 0.9
400
Epoch 4/10
25/25 [==============================] - 7s 265ms/step - loss: 0.1326 - accuracy: 0.9550 - val_loss: 0.0809 - val_accuracy: 0.8
900
Epoch 5/10
25/25 [==============================] - 6s 245ms/step - loss: 0.1106 - accuracy: 0.9650 - val_loss: 0.0046 - val_accuracy: 0.9
100
Epoch 6/10
25/25 [==============================] - 7s 275ms/step - loss: 0.1475 - accuracy: 0.9400 - val_loss: 0.0075 - val_accuracy: 0.9
500
Epoch 7/10
25/25 [==============================] - 6s 250ms/step - loss: 0.0852 - accuracy: 0.9650 - val_loss: 0.0112 - val_accuracy: 0.9
500
Epoch 8/10
25/25 [==============================] - 7s 260ms/step - loss: 0.0968 - accuracy: 0.9650 - val_loss: 0.2226 - val_accuracy: 0.9
200
Epoch 9/10
25/25 [==============================] - 7s 262ms/step - loss: 0.0993 - accuracy: 0.9700 - val_loss: 0.0033 - val_accuracy: 0.9
500
Epoch 10/10
25/25 [==============================] - 7s 265ms/step - loss: 0.0590 - accuracy: 0.9800 - val_loss: 0.0236 - val_accuracy: 0.9
600
```

Here the values corresponds to finally trained epochs are:

Loss: 0.0590, train accuracy: 0.9800, test loss: 0.0236, Test accuracy: 0.9600

LeNET5 Model




```
model3= Sequential()

model3.add(Conv2D(6,(5,5),activation='tanh'
,input_shape=(32,32,3)))

model3.add(AveragePooling2D())

model3.add(Conv2D(16,(5,5),activation='tanh
'))

model3.add(AveragePooling2D())

model3.add(Flatten())

model3.add(Dense(120, activation='tanh'))

model3.add(Dense(84, activation='tanh'))

model3.add(Dense(1, activation='sigmoid'))
```

LeNET5 Model summary:

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_13 (Conv2D)           (None, 28, 28, 6)         456
_________________________________________________________________
average_pooling2d (AveragePo (None, 14, 14, 6)         0
_________________________________________________________________
conv2d_14 (Conv2D)           (None, 10, 10, 16)        2416
_________________________________________________________________
average_pooling2d_1 (Average (None, 5, 5, 16)          0
_________________________________________________________________
flatten (Flatten)            (None, 400)               0
_________________________________________________________________
dense (Dense)                (None, 120)               48120
_________________________________________________________________
dense_1 (Dense)              (None, 84)                10164
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 85
=================================================================
Total params: 61,241
Trainable params: 61,241
Non-trainable params: 0
```

LeNet-5 is a popular and simple architectures. Here we train the data by splitting data into 16 batchesand with a fixed size of (32, 32) and trained for 10 epochs. It has 2 convolutional and 3 fully-connected layers. In this model kernel size has changed to (5, 5) in each layer and filters. It has average-pooling layer as the sub-samplinglayer. Thisarchitecture has about 60,000 parameters.It has tanh activation at each layer and sigmoid activation at output layer.

LeNET5 Model training details:

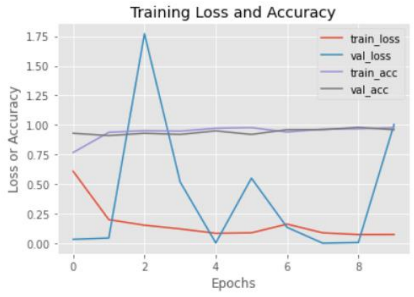
```
Epoch 1/10
25/25 [==============================] - 9s 360ms/step - loss: 0.5168 - accuracy: 0.7650 - val_loss: 0.2913 - val_accuracy: 0.8
800
Epoch 2/10
25/25 [==============================] - 3s 109ms/step - loss: 0.2026 - accuracy: 0.9275 - val_loss: 0.2823 - val_accuracy: 0.9
100
Epoch 3/10
25/25 [==============================] - 3s 117ms/step - loss: 0.2264 - accuracy: 0.9350 - val_loss: 0.2292 - val_accuracy: 0.9
500
Epoch 4/10
25/25 [==============================] - 2s 99ms/step - loss: 0.1569 - accuracy: 0.9425 - val_loss: 0.2229 - val_accuracy: 0.92
00
Epoch 5/10
25/25 [==============================] - 2s 87ms/step - loss: 0.1476 - accuracy: 0.9525 - val_loss: 0.1981 - val_accuracy: 0.95
00
Epoch 6/10
25/25 [==============================] - 2s 89ms/step - loss: 0.1388 - accuracy: 0.9550 - val_loss: 0.2577 - val_accuracy: 0.89
00
Epoch 7/10
25/25 [==============================] - 2s 91ms/step - loss: 0.1728 - accuracy: 0.9450 - val_loss: 0.2132 - val_accuracy: 0.92
00
Epoch 8/10
25/25 [==============================] - 2s 91ms/step - loss: 0.1474 - accuracy: 0.9500 - val_loss: 0.2166 - val_accuracy: 0.95
00
Epoch 9/10
25/25 [==============================] - 2s 91ms/step - loss: 0.1446 - accuracy: 0.9550 - val_loss: 0.1905 - val_accuracy: 0.96
00
Epoch 10/10
25/25 [==============================] - 2s 92ms/step - loss: 0.1438 - accuracy: 0.9450 - val_loss: 0.2087 - val_accuracy: 0.94
00
```

Here the values corresponds to finally trained epochs are:

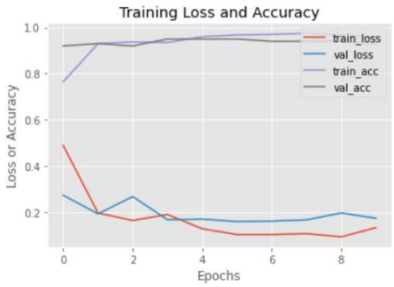
Loss: 0.1438, train accuracy: 0.9450, test loss: 0.2087, Test accuracy: 0.9400

RESULT OF COMPARISON

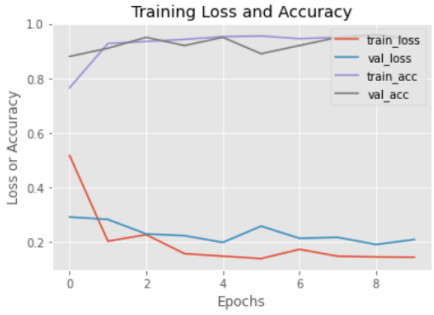
Model – 1:



Model – 2:



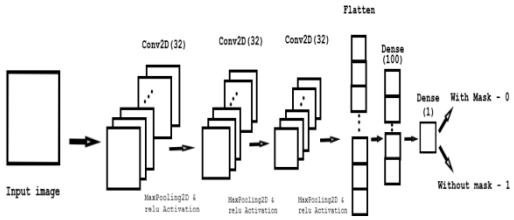
LeNET5 Model:



From the observations of the different architecture it is recommended to choose model 2 for training our dataset. Model 2 has more test accuracy compare to other models given here. And it is found that test accuracy with other standard models like VGG-16 and AlexNet which exhibits low for the given small datasets. Some models show more fluctuation in the values of train_loss, val_loss, train_accuracy and val_accuracy. It is good to follow a normal architecture like Model 2 to build a good trained model which has a test accuracy value of 0.9600 and low variations in Loss or Accuracy with respect to change in epochs.

TRAING MODEL FOR MASK DETECTION

Based on the result of comparison of architecture we come up with a good model and trained it with dataset containing around 8000 images.



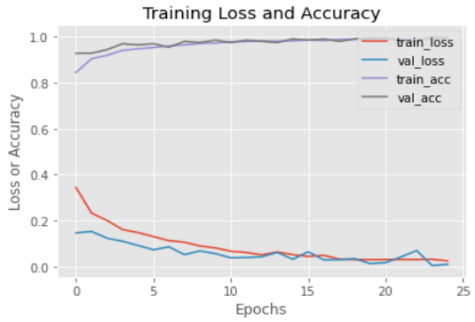
First objective is to train model for large number of epochs and find out the variations with respect to number of epochs. The result of the model trained with large dataset for 25 epochs:

Dataset specifications:

Data set consists of 7553 RGB images in 2 folders as with mask andwithout mask. Images are named as label with maskandwithout mask. Images of faces with mask are 3725 and images of faces without mask are 3828.

Source:

https://www.kaggle.com/omkargurav/face-mask-dataset



From the result of the above training it is obvious that at 15 to 20 epochs loss and accuracy curve became almost flat and attained minimum loss and maximum accuracy values. So we can finalize our model by training around 20 epochs with a large dataset.

Dataset specifications for the final model:

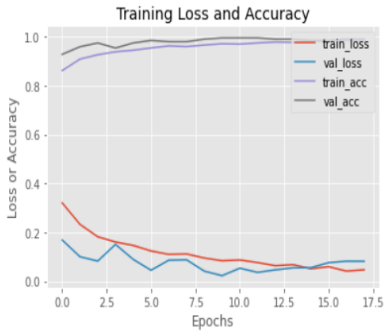
Data set consists of 7553 RGB images in 2 folders as with mask andwithout mask. Images are named as label with mask and without mask. Images of faces with mask are 3725 and images of faces without mask are 3828.

Final Model training details:

```
Epoch 10/18
473/473 [==============================] - 202s 426ms/step - loss: 0.0837 - accuracy: 0.9710 - val_loss: 0.0228 - val_accuracy:
0.9948
Epoch 11/18
473/473 [==============================] - 202s 427ms/step - loss: 0.0868 - accuracy: 0.9695 - val_loss: 0.0532 - val_accuracy:
0.9948
Epoch 12/18
473/473 [==============================] - 202s 427ms/step - loss: 0.0762 - accuracy: 0.9743 - val_loss: 0.0362 - val_accuracy:
0.9948
Epoch 13/18
473/473 [==============================] - 204s 432ms/step - loss: 0.0635 - accuracy: 0.9782 - val_loss: 0.0465 - val_accuracy:
0.9897
Epoch 14/18
473/473 [==============================] - 204s 431ms/step - loss: 0.0678 - accuracy: 0.9768 - val_loss: 0.0548 - val_accuracy:
0.9897
Epoch 15/18
473/473 [==============================] - 204s 431ms/step - loss: 0.0509 - accuracy: 0.9824 - val_loss: 0.0555 - val_accuracy:
0.9845
Epoch 16/18
473/473 [==============================] - 205s 432ms/step - loss: 0.0594 - accuracy: 0.9809 - val_loss: 0.0760 - val_accuracy:
0.9845
Epoch 17/18
473/473 [==============================] - 204s 431ms/step - loss: 0.0413 - accuracy: 0.9844 - val_loss: 0.0818 - val_accuracy:
0.9897
Epoch 18/18
473/473 [==============================] - 204s 432ms/step - loss: 0.0468 - accuracy: 0.9829 - val_loss: 0.0815 - val_accuracy:
0.9897
```

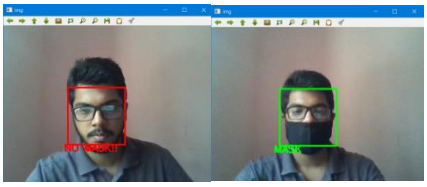
Values at the end of training:

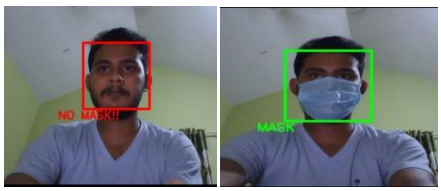
Loss: 0.0468, train accuracy: 0.9829, test loss: 0.0815, Test accuracy: 0.9897.



RESULT OF PROJECT

A normal CNN architecture is used for building the model and trained the model with around 8000 images. Trained model for 18 epochs and tested with real time video streaming. A good architecture is chosen by constructing different architectures and evaluating with sample dataset of smaller size. We come up with a good model to detect face mask with 98.97% test accuracy and detected faces with and without mask by real time video streaming. Also obtained a good loss percentage of 8.15% with respects to other models.





APPLICATIONS

- Face mask detector can be used along with checking temperature in public places such as supermarket, Government offices and other places where people gather that may cause to spreading of disease.
- By adding percentage that indicates proper wearing of face mask can be used to correct errors in wearing mask individually.
- This program can also be connected to the entrance gates and only people wearing face masks can come in.
- An alarm system can also be implemented to sound a beep when someone without a mask enters the area.
- Store images of the people entered the area and make dataset of people to connect them or for further researches.

REFERENCES:

[1]https://www.kaggle.com/datasets

[2]R. Lienhart and J. Maydt, "An extended set of haar-like features for rapid object detection," vol.1, 02 2002, pp.I–900.

[3]https://towardsdatascience.com/illustrated-10-cnn-architectures-95d78ace614d

[4]P. Viola and M. J. Jones, "Robust real-time face detection," Int. J. Comput. Vision, vol. 57, no. 2,p. 137154, May 2004. https://doi.org/10.1023/B:VISI.0000013087.49260.fb

[5]https://towardsdatascience.com/face-detection-with-haar-cascade-727f68dafd08

[6]Paul Viola and Michael Jones in the paper, " Object Detection using Haar feature-based cascade classifiers method" in 2001

[7]https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_objdetect/py_face_detection/py_face_detection.html