Deep Learning

AI and ML (6CS012)

A2: Project Portfolio

Face Mask Detection

Student Number : 2049855

Full Name : Jenny Rajak

Group : L6CG1

Lecturer : Mr. Siman Giri

Tutor : Mr. Shreejan Shrestha

Submission : 02/05/2022

Abstract

Face mask detection is the detection of masked and unmasked faces. Today, the world has been greatly affected by the COVID-19. To reduce COVID-19 wearing a mask is considered one of the remedies. So, to spread awareness and ensure everybody's safety, we need to promote technology that detects masked and unmasked faces in public places such as colleges, schools, malls, airports, etc.

We have made a model using Keras, OpenCV, and MobileNet to detect a masked and unmasked face in real-time video. We have successfully achieved the output which can predict the masked and unmasked faces correctly. Firstly, for training the data we have used a dataset that is taken from GitHub. It is a task given by the module Artificial Intelligence and Machine learning for us to implement the CNN for performing image classification tasks practically for us to utilize the knowledge which we have gained during the CNN classes.

Table of Contents

Table of Figures

1. Introduction

1.1. Dataset

The data we are going to work on is the data that is captured in real-time. There are going to be two kinds of data that will be captured one is with a mask, and the other is without a mask. Image classification is a task in computer vision for classifying images in accord with their visual content. (Sanghvi, 2020) Here, in face mask detection we are about to detect whether a person is wearing a mask or not in real-time. The dataset is taken from the GitHub.

1.2. Aims and objectives

1.2.1. Aims

The main goal of this project is to detect a person with and without a mask correctly in a real-time video stream.

1.2.2. Objectives

- To grasp about CNN and its importance in image classification.
- To gain and implement knowledge about various deep learning libraries.
- To learn about various layers of CNN and implement them.

1.3. Introduction to CNN

CNN is a sort of artificial brain network utilized for picture acknowledgment and implementation that is explicitly intended to deal with pixel information. (Contributor, 2018)
Importance of CNN for image classification task
Parameters:

The number of parameters in the neural network increases with the number of layers resulting in high time complexity in the training of the model. Additionally, tuning those parameters also takes time, which can be reduced by the use of CNN.

Network:

CNN are capable of reducing the number of parameters without the lessening the nature of the model. (Mishra, 2019)

2. Methodology

2.1. Model summary

The model is built using MobileNetV2 architecture. In our model, we have used 5 different layers, input layer is the initial layer which brings the initial data for further implementation, averagepooling2d the input size of the image is taken over an input window defined by pool_size which was taken as 7. In the dropout layer, it prevents the model from overfitting. Then in the flattened layer, the data are converted into a 1d array for input into the next layer, and the output of the convolutional layers is flattened for creating a single long feature vector. The flattened layer is also connected to the fully-connected layer. Finally, in the dense layer matrix-vector multiplication is performed. It receives output from every neuron of its preceding layer. Softmax is used as a activation function because we are performing a binary classification where we are detecting masked and unmasked faces.

2.1.1. Number of layers used in the model:

The model consists of 5 layers.

2.1.2. Name of the layers:

AveragePooling2D:

average_pooling2d (AveragePooli	(None, 1, 1, 1280)	0	out_relu[0][0]
flatten (Flatten)	(None, 1280)	0	average_pooling2d[0][0]

Figure 1: Model Summary: AveragePooling2D layer

Dropout:

dropout (Dropout)	(None, 128)	0	dense[0][0]

Figure 2: Model Summary: Dropout layer

Flatten:

flatten (Flatten)	(None, 1280)	0	average_pooling2d[0][0]

Figure 3: Model Summary: Flatten layer

Dense:

dense (Dense)	(None, 128)	163968	flatten[0][0]
dropout (Dropout)	(None, 128)	0	dense[0][0]
dense_1 (Dense)	(None, 2)	258	dropout[0][0]

Figure 4: Model Summary: Dense layer

Input:

input_1 (InputLayer)	[(None, 224, 224, 3) 0

Figure 5: Model Summary: Input layer

2.1.3. Hyper parameters used

The hyper parameters that were provided were pool_size, dense layer, dropout layer.

Whereas other hyper parameters used in a models are learning rate, epochs, batch size, random state and test size.

```
headModel = baseModel.output
headModel = AveragePooling2D(pool_size=(7, 7))(headModel)
headModel = Flatten(name="flatten")(headModel)
headModel = Dense(128, activation="relu")(headModel)
headModel = Dropout(0.5)(headModel)
headModel = Dense(2, activation="softmax")(headModel)
```

Figure 6: Hyper parameters

2.1.4. Number of hyper parameter

The model has about 2,388,098 trainable hyper parameters.

```
Total params: 2,422,210
Trainable params: 2,388,098
Non-trainable params: 34,112
```

Figure 7: Number displaying number of trainable parameters

2.2. Training of the model

For training of the model, we have used Binary_crossentropy as the loss function as face mask detection is a binary classification problem consisting of two categories masked and unmasked faces. Adam optimization is used as the optimization algorithm due to its time complexity is faster, and it only needs a few parameters for tuning. Whereas, epochs is the training phase of the neural network where all the data is trained. It is made up of one or more batches. In our case we have provided about 20 epochs and 32 batch size.

2.2.1. Loss function:

Binary_crossentropy is used as the loss function in the model.

2.2.2. Optimization Algorithm:

The algorithm used is Adam optimization algorithm.

2.2.3. Number of epochs

We have trained our model for about 20 epochs.

2.2.4. Image of training and validation loss against the iteration

```
[INFO] training head...
Epoch 1/20
34/34 [==============================] - 71s 2s/step - loss: 0.5040 - accuracy: 0.8268 - val_loss: 0.2315 - val_accuracy: 0.974
6
Epoch 2/20
34/34 [==============================] - 62s 2s/step - loss: 0.2034 - accuracy: 0.9691 - val_loss: 0.1092 - val_accuracy: 0.989
1
Epoch 3/20
34/34 [==============================] - 65s 2s/step - loss: 0.1108 - accuracy: 0.9897 - val_loss: 0.0679 - val_accuracy: 0.989
1
Epoch 4/20
34/34 [==============================] - 62s 2s/step - loss: 0.0832 - accuracy: 0.9869 - val_loss: 0.0490 - val_accuracy: 0.992
8
Epoch 5/20
34/34 [==============================] - 59s 2s/step - loss: 0.0599 - accuracy: 0.9934 - val_loss: 0.0390 - val_accuracy: 0.992
8
Epoch 6/20
34/34 [==============================] - 62s 2s/step - loss: 0.0509 - accuracy: 0.9925 - val_loss: 0.0322 - val_accuracy: 0.992
8
Epoch 7/20
34/34 [==============================] - 61s 2s/step - loss: 0.0397 - accuracy: 0.9934 - val_loss: 0.0294 - val_accuracy: 0.992
8
Epoch 8/20
34/34 [==============================] - 59s 2s/step - loss: 0.0290 - accuracy: 0.9972 - val_loss: 0.0246 - val_accuracy: 0.996
4
Epoch 9/20
34/34 [==============================] - 58s 2s/step - loss: 0.0382 - accuracy: 0.9916 - val_loss: 0.0222 - val_accuracy: 0.992
8
Epoch 10/20
34/34 [==============================] - 59s 2s/step - loss: 0.0263 - accuracy: 0.9953 - val_loss: 0.0203 - val_accuracy: 0.992
8
Epoch 11/20
34/34 [==============================] - 62s 2s/step - loss: 0.0269 - accuracy: 0.9953 - val_loss: 0.0179 - val_accuracy: 0.992
8

Epoch 12/20
34/34 [==============================] - 71s 2s/step - loss: 0.0310 - accuracy: 0.9916 - val_loss: 0.0161 - val_accuracy: 0.996
4
Epoch 13/20
34/34 [==============================] - 63s 2s/step - loss: 0.0262 - accuracy: 0.9953 - val_loss: 0.0155 - val_accuracy: 1.000
0
Epoch 14/20
34/34 [==============================] - 63s 2s/step - loss: 0.0200 - accuracy: 0.9953 - val_loss: 0.0144 - val_accuracy: 1.000
0
Epoch 15/20
34/34 [==============================] - 62s 2s/step - loss: 0.0205 - accuracy: 0.9981 - val_loss: 0.0132 - val_accuracy: 1.000
0
Epoch 16/20
34/34 [==============================] - 52s 2s/step - loss: 0.0193 - accuracy: 0.9963 - val_loss: 0.0123 - val_accuracy: 1.000
0
Epoch 17/20
34/34 [==============================] - 51s 2s/step - loss: 0.0195 - accuracy: 0.9944 - val_loss: 0.0120 - val_accuracy: 1.000
0
Epoch 18/20
34/34 [==============================] - 48s 1s/step - loss: 0.0204 - accuracy: 0.9934 - val_loss: 0.0129 - val_accuracy: 1.000
0
Epoch 19/20
34/34 [==============================] - 49s 1s/step - loss: 0.0149 - accuracy: 0.9972 - val_loss: 0.0115 - val_accuracy: 0.996
4
Epoch 20/20
34/34 [==============================] - 40s 1s/step - loss: 0.0152 - accuracy: 0.9963 - val_loss: 0.0115 - val_accuracy: 0.996
4
```

Figure 8: Image of training and validation loss against iteration

2.3. Findings and Discussion

2.3.1. Evaluation metrics

The evaluation metric we have used is the classification report. We have used the evaluation metrics to find the accuracy of our trained model. As we can observe in figure 9, our model has

obtained an accuracy of 100%. Additionally, in figures 10 and 11, we can observe that the model has successfully detected faces with and without masks acquiring 100% accuracy.

```
print(classification_report(testY.argmax(axis=1), predIdxs,
                            target_names=lb.classes_))
```

	precision	recall	f1-score	support
with_mask	1.00	0.99	1.00	138
without_mask	0.99	1.00	1.00	138
accuracy			1.00	276
macro avg	1.00	1.00	1.00	276
weighted avg	1.00	1.00	1.00	276

Figure 9: Evaluation metrics

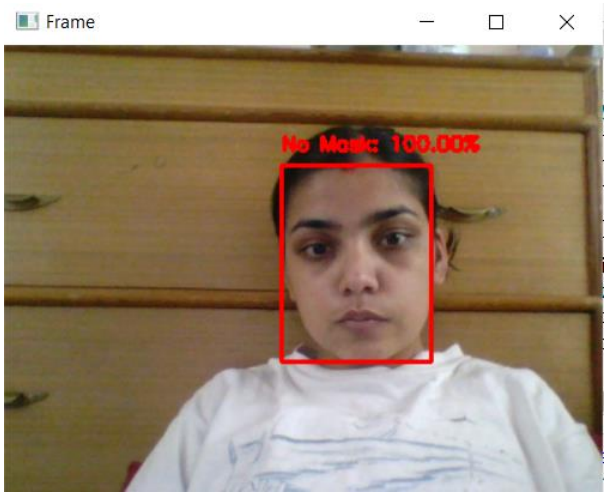

Figure 10: Screenshot of model detecting no mask

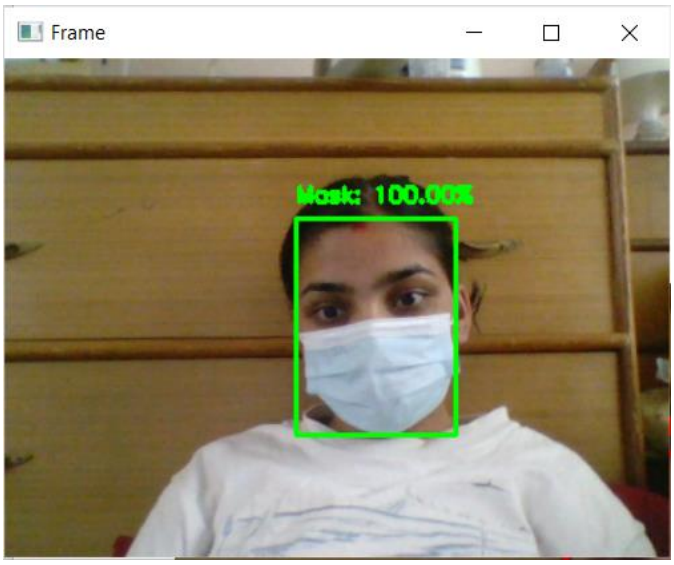

Figure 11: Screenshot of model detecting mask

2.4. Recommendations

Since it is a face mask detector (FMD) that detects faces on a live video camera for now. Besides, if few improvements are made then the model can be integrated into both embedded systems and CCTV cameras. For instance, embedded system applications can use FMD in public areas like schools, colleges, offices, etc to ensure COVID-19 safety guidelines and in CCTV cameras to identify and detect unmasked people.

3. References

Contributor, T. T., 2018. *techtarget.* [Online]
Available at: https://www.techtarget.com/searchenterpriseai/definition/convolutional-neural-network
[Accessed 28 April 2022].

Mishra, P., 2019. *DataDrivenInvestor.* [Online]
Available at: https://medium.datadriveninvestor.com/why-are-convolutional-neural-networks-good-for-image-classification-146ec6e865e8
[Accessed 29 April 2022].

Sanghvi, K., 2020. *Analytics Vidhya.* [Online]
Available at: https://medium.com/analytics-vidhya/image-classification-techniques-83fd87011cac
[Accessed 28 April 2022].