AI 825 / Visual Recognition Mini-project-1

Dev Patel
IMT2018021
Dev.Patel@iiitb.org

Ishmeet Singh
IMT2018030
Ishmeet.Singh@iiitb.org

Prateek Kamboj
IMT2018057
Prateek.Kamboj@iiitb.org

Ishaan Sachdeva
IMT2018508
Ishaan.Sachdeva@iiitb.org

I. INTRODUCTION

After the breakout of the worldwide pandemic COVID-19, there arises a severe need for protection mechanisms, face masks being the primary one. The basic aim of the project is to detect the presence of a face mask on human faces on live streaming video, images as well as recorded video. We have used deep learning to develop our face detector model.

Face mask detection means identifying whether a person is wearing a mask or not. The first step to recognize the presence of a mask on the face is to detect the person if present then face, and finally, check if the person is wearing a mask. Which makes the strategy divided into three parts: to detect persons, detect faces, and to detect masks on those faces.

II. DATASET

The dataset which we have used consists of 1376 total images out of which 690 are of masked faces and 686 are of unmasked faces. We have split our dataset into three parts: training dataset(%72), test dataset(%10), and validation dataset(%18).

III. ALGORITHM

1. The first step is to load images from the 'dataset' folder into a NumPy array. For this, we are listing all image names in the folder into a list using os.listdir(data_path). Then we are looping over this list: grab the image, convert it to grayscale, and resizing it(100x100). Finally appending the pre-processed image and associated label to the data and labels Numpy array respectively.

2. The next step is to define our CNN model architecture. we have tried different models with a different number of layers with a combination of Activation function, Maxpooling, and Batch normalization. But finally, the Best performing model was consist of 3-convolutional layers and 2-Fully connected layers.

   Each Convolution Layer contains:

   - Conv2d -> It is a linear operation used for feature extraction. It will convolve the input signal with the kernel (used size 3). It will also take out_channel.
   - ReLU -> It is a nonlinear activation function. It will output the input directly if it is positive, otherwise, it will output zero.
   - MaxPool2d -> It is a pooling layer. It reduces the in-plane dimensionality of the feature maps in order to decrease the number of subsequent learnable parameters.

   The final fully connected layer typically has the same number of output nodes as the number of classes. In our case, it was 2 one for each with and without a mask.

   Below is the model summary:

```
_________________________________________________________________
Layer (type)                  Output Shape             Param #
=================================================================
conv2d (Conv2D)               (None, 98, 98, 256)      2560
_________________________________________________________________
activation (Activation)       (None, 98, 98, 256)      0
_________________________________________________________________
max_pooling2d (MaxPooling2D)  (None, 49, 49, 256)      0
_________________________________________________________________
conv2d_1 (Conv2D)             (None, 47, 47, 128)      295040
_________________________________________________________________
activation_1 (Activation)     (None, 47, 47, 128)      0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2  (None, 23, 23, 128)      0
_________________________________________________________________
conv2d_2 (Conv2D)             (None, 21, 21, 128)      147584
_________________________________________________________________
```

```
activation_2 (Activation)          (None, 21, 21, 128)     0
_________________________________________________________________
flatten (Flatten)                  (None, 56448)           0
_________________________________________________________________
dropout (Dropout)                  (None, 56448)           0
_________________________________________________________________
dense (Dense)                      (None, 64)              3612736
_________________________________________________________________
dense_1 (Dense)                    (None, 2)               130
=================================================================
Total params: 4,058,050
Trainable params: 4,058,050
Non-trainable params: 0
```

3. The next step is to train our CNN model. For that, we have used Categorical_Crossentropy as loss function and Adam as an optimizer. And used model.fit() to train the model on the training dataset.

4. The next step is to detect if there is a person present in the current live feed from the webcam. For that, we have used YOLOv3. First, we load the YOLOv3 weight and config file using the cv2.dnn module. Then we convert the input frame (read from webcam) into a blob using dnn.blobFromImage. This function will perform scaling, mean subtraction, and channel swapping. Next, we pass blob to forward() function, which will return a nested list containing information ([x,y,h,w], score, and confidence) about all the objects detected. Then we loop over all the objects detected, calculate the bounding box for that object with a confidence of more than 50%. Finally, we loop over all bounding boxes, check if the object is a person or not. And if it is a person then we draw a bounding box on the frame and put text around it.

5. The next step is to detect faces. If in the above step, there is at least one person detected then we pass the frame to haarcascade_frontalface classifier. which will return a list of (x,y,h,w) for each face present in the image.

6. The next step is to detect if there is a mask on the faces detected in the above step. For this, we loop over the list from the previous step extract out the part of the face from the frame using (x,y,h,w) values and pass the extracted image to our trained CNN model. which will return a label indicating Mask/No-Mask. Finally, we can draw a bounding box and put an appropriate text using a label for each face in the frame.

IV. YOLO

People detection is an important task for surveillance. Like the human visual system, we are here trying to automate the task of person detection using YOLO. It is a CNN that applies a single NN to the full image, and then divides the image into regions and predicts bounding boxes and probabilities for each region. The task we have in hand is identifying what is there in the image and where it is located. YOLO can achieve high accuracy while also being able to run in real-time which is required in our LIVE person detection system. The algorithm “only looks once” as in it requires only one forward propagation pass through the neural network to make predictions.

Here a single CNN simultaneously predicts multiple bounding boxes and class prediction of these boxes which is helpful in our case also if there are more than one people approaching the door system.

We have used the recent advancement in the YOLO version i.e. YOLOv3 pre-trained model on ImageNet 1000 class classification dataset. Which seemed like a much better improvement upon testing our model using this and the previous versions of YOLO as it performed accurately in the live scenario while the YOLOv2 and YOLO-tiny seemed to be inaccurate and hit or miss in many cases.

V. VIOLA JONES

Next up is to detect the person’s face, this task is achieved using Viola-Jones Object Detection Framework which can quickly and accurately detect objects in images and works particularly well with human faces. The framework combines the concept of haar-like features, integral images, the AdaBoost algorithm, and the cascade classifier.

In the model, we used haarcascade_frontalface default as our cascade classifier. These classifiers have been trained on images from the frontal CALTECH dataset and detect whether the object is a face or not a face. The first step it does is to collect Haar features which are calculations that are performed on adjacent rectangular regions at a specific

location in a detection window. The calculation involves summing the pixel intensities in each region and calculating the differences between the sums. For larger images, integral images come into play. Integral images speed up calculations of Haar features on larger images. Then to determine the best features that represent an object, AdaBoost is used which essentially uses a combination of weak classifiers to create a strong classifier that the algorithm can use to detect objects. The last step is to implement cascading classifiers which use a series of steps containing collections of weak learning classifiers and boosting them to give high accuracy classifiers.

It works on multiple faces simultaneously as well and we use it to detect the face area of the person which is passed in the model and then used to determine whether the person is wearing a mask or not.

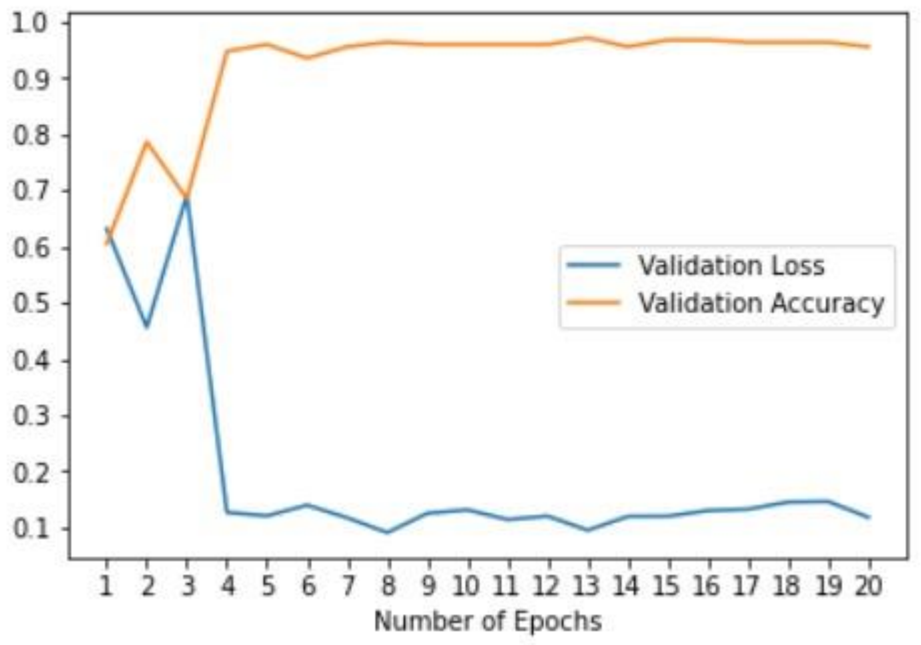
VI. OBSERVATIONS

1. The algorithm works 100% accurately with a white color mask. The reason could be that the dataset consists of all images with a white mask.
2. If the background contains a plain wall/screen then accuracy increases drastically.
3. If a person is wearing glasses with a mask then the algorithm has a hard time detecting their face.
4. The algorithm works perfectly fine with multiple persons, some wearing masks and some not.
5. We have tried Batch Normalization in CNN architecture, but it was reducing CNN as well as overall system accuracy.

VII. RESULT AND CONCLUSION

We have modelled a face mask detector using YOLO, haarcascade_frontalface, and CNN. The model was inferred on images and live video streams. To select a base model, we evaluated the metrics like accuracy, precision, and loss function.

Accuracy of CNN Architecture:



validation accuracy = 0.9718   validation loss = 0.0948

Overall system real-time accuracy:

We have tried our model on three different people and in one case with two people (1 wearing a mask and another not) and it worked perfectly fine in all 4 cases (it is also shown in the demo video).

In conclusion our VR system will detect if person is wearing a mask or not, and respectively opens/closes the door.

VIII. REFERRENCES

[1]https://www.pyimagesearch.com/2020/05/04/covid-19-face-mask-detector-with-opencv-keras-tensorflow-and-deep-learning/

[2] https://data-flair.training/blogs/face-mask-detection-with-python/

[3] https://www.pyimagesearch.com/2018/11/12/yolo-object-detection-with-opencv/

[4] https://towardsdatascience.com/object-detection-using-yolov3-and-opencv-19ee0792a420

[5] https://www.mygreatlearning.com/blog/yolo-object-detection-using-opencv/

[6] https://machinelearningmastery.com/how-to-perform-object-detection-with-yolov3-in-keras/