Vision based automatic door entry system

Rachit Yagnik
MT2020078
rachit.yagnik@iiitb.org

Shourabh Payal
MT2020054
shourabh.payal@iiitb.org

Sujith Kumar
MT2020106
sujit.kumar@iiitb.org

Anshul kumar garg
MT2020154
anshulkumar.garg@iiitb.org

***Abstract*—We will attempt to build a Vision based automatic door entry system. In the times where pandemics will be a common phenomenon across the globe in the coming years we can automate some aspects of day to day living with the help of technology. A vision based automatic door entry system aims at providing an accurate and safe way to control locking mechanisms of a building door. If a visitor is detected by the system, it can automatically decide to allow or deny the visitor entry based on a simple fact: is he/she wearing a mask?**

***Index Terms*—Convolutional neural network, YOLO, Haarcascade frontal face classifier, corona, mask**

I. Dataset

Our data set will consists of two classes.

- With mask: images of various people with a augmented face mask.

- Without mask

II. Data Pre-processing

- Random resize crop : Random resize crop is a data augmentation technique. This is done to create new training example from our existing dataset that will help our model to generalize better. This will randomly extract patches of given size. We used patch size of (224x224). So it will extract patches of size (224x224) from images randomly. It might be from top corner, bottom or centre.
- Random horizontal flip : Once we have images of size (224x224), we can choose to flip it. This is another part of data augmentation.
- Normalization : This is just input data scaling. We will normalize our tensor image with given mean and standard deviation. Both mean and standard deviation should be sequence.

III. Model

We will be dividing the problem into three modules.

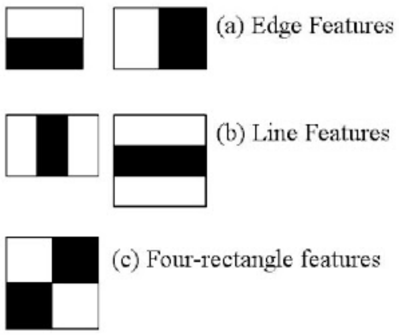
- Detection of human if any using YOLO
- Detection of face if any
- Detection of mask if any

A. Human Detection using YOLO

YOLO is a one-stage detector strategy that is little less accurate than two-stage detectors i.e R-CNN but significantly faster. It apply a single neural network to the full image. This network divides the image into regions and predicts bounding boxes and probabilities for each region. These bounding boxes are weighted by the predicted probabilities. For human detection, we have used pretrained YOLOv3 model trained on coco dataset. Our premodel will return the bounding box over image with class name over it.

B. Face detection using Haarcascade frontal face classifier

After human detection, we will use haarcascade frontal face classifier to count and detect faces in the image. Haarcascade classifier needs a lot of positive images (images of faces) and negative images (images without faces) to train the classifier. Then we need to extract features from it. For this, Haar features shown in the below image are used.



They are just like our convolutional kernel. Each feature is a single value obtained by subtracting sum of pixels under the white rectangle from sum of pixels under the black rectangle. OpenCV already contains many pre-trained classifiers for face, eyes, smile etc. The output of classifier is pass for mask detection.

C. Mask detection using CNN

For detecting the mask we mainly used 2 kind of models as discussed below

1) **Transfer learning with Alexnet**

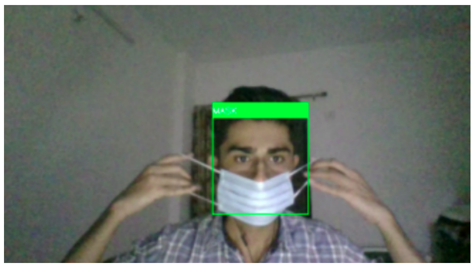
   We initially used a transfer learning approach to quickly build our initial baseline model which could point us to the correct directions for using other models.
   This model was very quick to train and gave us an accuracy of 92% on the validation data.

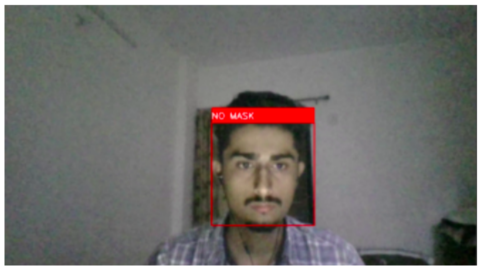
2) **Transfer learning with MobileNetv2**

   Having tried the transfer learning approach we came up with custom CNN architecture added on top of MobileNetv2. Here we use 4-5 ReLU activated fully connected layers beginning with 1024 nodes in the first layer and going till 32 nodes in the last layer. Each layer has the number of nodes cut by half. This is combined with dropout layer and soft max layer at the end to help prevent over fitting.
   We obtained an accuracy of 99% on validation data.

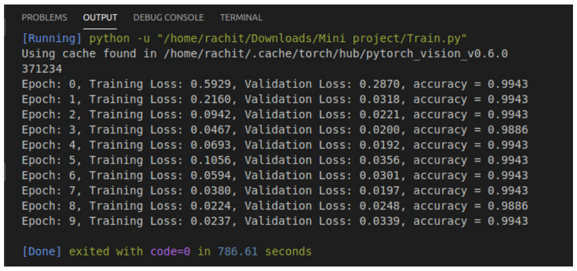
IV. Results

1) **Alexnet base**





2) **MobileNetv2 base & custom layers**
   After training this model, we achieve the accuracy of 99% and successfully detected mask and non-mask. Results and video are attached in zip file.



```
[Running] python -u "/home/rachit/Downloads/Mini project/Train.py"
Using cache found in /home/rachit/.cache/torch/hub/pytorch_vision_v0.6.0
371234
Epoch: 0, Training Loss: 0.5929, Validation Loss: 0.2870, accuracy = 0.9943
Epoch: 1, Training Loss: 0.2160, Validation Loss: 0.0318, accuracy = 0.9943
Epoch: 2, Training Loss: 0.0942, Validation Loss: 0.0221, accuracy = 0.9943
Epoch: 3, Training Loss: 0.0467, Validation Loss: 0.0200, accuracy = 0.9886
Epoch: 4, Training Loss: 0.0693, Validation Loss: 0.0192, accuracy = 0.9943
Epoch: 5, Training Loss: 0.1056, Validation Loss: 0.0356, accuracy = 0.9943
Epoch: 6, Training Loss: 0.0594, Validation Loss: 0.0301, accuracy = 0.9943
Epoch: 7, Training Loss: 0.0380, Validation Loss: 0.0197, accuracy = 0.9943
Epoch: 8, Training Loss: 0.0224, Validation Loss: 0.0248, accuracy = 0.9886
Epoch: 9, Training Loss: 0.0237, Validation Loss: 0.0339, accuracy = 0.9943

[Done] exited with code=0 in 786.61 seconds
```

V. Conclusion

We have successfully implemented vision based automatic door entry system by training our model on given dataset. Our next goal will be detecting mask when multiple person are present.

Challenges and Future Work

Integrate the model with an end to end system and test for practical usage. Also we will try to integrate the model with sensors and perform locking and controlling mechanisms.

Acknowledgment

We would like to thank Professor Dinesh Babu Jayagopi and also the Teaching Assistants for giving us the opportunity to work on this project and help us whenever we were struck by giving ideas and pointing out to specific resources that we could learn from. Also, we would like to thank the all teams for sharing their ideas and having open ended discussions. We really had a great learning experience while working on this project.

References

[1] https://www.youtube.com/watch?v=EMXfZB8FVUA&list=PLqnslRFeH2UrcDBWF5m
[2] https://www.youtube.com/watch?v=d3DJqucOq4g
[3] https://github.com/prajnasb/observations/tree/master/experiements/data

Code link

Full Code link