Real-time Face Mask Detection

Kashi Vishwanath
Dept. of Electrical and Computer Engineering
Virginia Tech
Blacksburg, VA
kkashivishwanath@vt.edu

***Abstract*— Deep Learning based system that can detect instances where face masks are not used properly. This system consists of a dual-stage Convolutional Neural Network (CNN)architecture capable of detecting masked and unmasked faces and can be integrated with cameras. We are going to use computer vision with a convolution for better accuracy. This will help track safety violations, promote the use of facemasks, and ensure a safe environment in malls, public places etc., This will help people in a way that it can ensure that everyone is wearing a mask to reduce the risk of transmission.**

***Keywords*— Object detection, CNN, PyTorch, Computer Vision, real-time object tracking.**

I. Introduction

Computer Vision has proven to be a revolutionary aspect of modern technology. In a world battling against the Novel Coronavirus Disease (COVID-19) pandemic, technology has been a lifesaver. Multiple studies have shown that the use of face masks reduces the risk of viral transmission. A two-stage CNN architecture, where the first stage detects human faces, while the second stage uses a lightweight image classifier to classify the faces detected in the first stage as either 'Mask' or 'No Mask' faces and draws bounding boxes around them along with the detected class name. This algorithm can be further extended to videos as well. The detected faces are then tracked between frames using an object tracking algorithm. We will be using a pretrained face detection model from caffe and a mobilenet_v2 model in the second stage of the classifier.

II. Related Work

The problem of detecting multiple masked and unmasked faces in images can be solved by a traditional object detection model. The process of object detection mainly involves localizing the objects in images and classifying them (in case of multiple objects). Traditional algorithms like Haar Cascade and HOG have proved to be effective for such tasks, but these algorithms are heavily based on Feature Engineering.

Convolutional Neural Networks (CNNs) (LeCun et al., 1998) is a key aspect in modern Computer Vision tasks like pattern object detection, image classification, pattern recognition tasks, etc. A CNN uses convolution kernels to convolve with the original images or feature maps to extract higher-level features, thus resulting in a powerful tool for Computer Vision tasks.
(Ejaz et al., 2019) have performed facial recognition on masked and unmasked faces using Principal Component Analysis (PCA). However, the recognition accuracy drops to less than 70% when the recognized face is masked.
The work by (Nieto-Rodríguez et al., 2015) presented a method that detects the presence or absence of a medical mask. The primary objective of this approach was to trigger an alert only for medical staff who do not wear a surgical mask, by minimizing as many false- positive face detections as possible, without missing any medical mask detections.

CNN based object detection algorithms can be classified into 2 categories: Multi-Stage Detectors and Single-Stage Detectors. Multi-Stage Detectors: In a multi-stage detector, the process of detection is split into multiple steps. A two-stage detector like RCNN first estimates and proposes a set of regions of interest using selective search. The CNN feature vectors are then extracted from each region independently. Multiple algorithms based on Regional Proposal Network like Fast RCNN and Faster RCNN have achieved higher accuracy and better results than most single stage detectors.

A single-stage detector performs detections in one step, directly over a dense sampling of possible locations. These algorithms skip the region proposal stage used in multi-stage detectors and are thus considered to be generally faster, at the cost of some loss of accuracy. One of the most popular single-stage algorithms, You Only Look Once (YOLO) (Redmon et al.,2016), was introduced in 2015 and achieved close to real-time performance. Single Shot Detector (SSD) (Liu et al., 2016) is another popular algorithm used for object detection, which gives excellent results.

III. Dataset

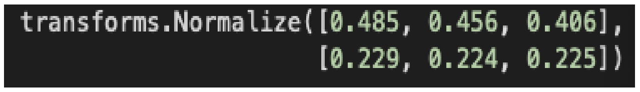
A publicly available dataset from Kaggle is used. It consists around 12k images divided Train, Test and Validation. We have used Train and Test dataset for our training of the model. Train dataset has two classes – mask and no mask. Each class has 5000 images. Test dataset contains 992 images split into 483 images for mask and 509 for no mask. Example figures from these datasets are shown below. Images with mask are labelled 'mask', unmasked faces are labelled as 'no_mask'.

Samples from mask train dataset

Samples from no_mask train dataset

A. *Preprocessing*

The training and the test sets are preprocessed accordingly based on the model we are using. Since we have ImageNet model used in our implementation, we have to normalize the train and test dataset according to the ImageNet standards. In the figure below you can see the normalization metrics.



```
transforms.Normalize([0.485, 0.456, 0.406],
                     [0.229, 0.224, 0.225])
```

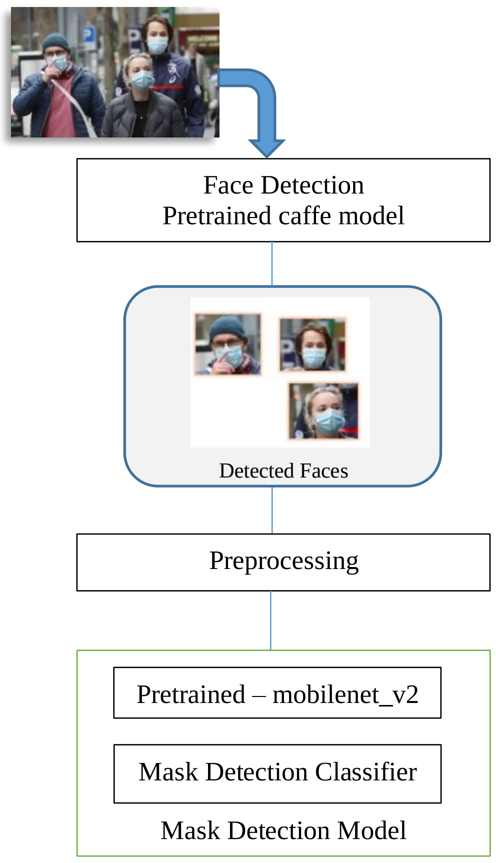
B. *Resizing*

We resize all the datasets to be of shape (224,224). This size is determined by the pretrained model – MobilenetV2 which we use for classification of mask.

Since there are not many public datasets which are not biased towards a particular race and best image quality than the current one. We have performed augmentation of the images in the dataset by doing random rotation (degrees = 15), color jitter and random horizontal flip. This provided us with more images in the dataset.

IV. Model Design

In the two-stage classifier first we detect the faces from the images and these faces are fed to the classifier to classify if a person has worn a mask or not.



Model Framework

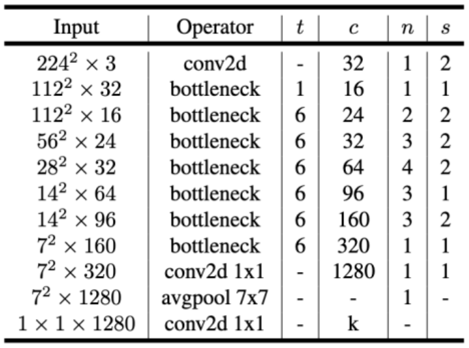
The output of the Mask detection model is an image where we detect face mask, and a bounding box is created over the face mentioning the class it belongs.

V. Mask Detection Architecture

Mask Detection model takes faces as input and determines which class the image belongs to – mask or no_mask. This model consists of two components: Pretrained mobilenet_v2 and fully connected classifier. We will look into it in more detail.

A. *Mobilennet_v2*

A pretrained mobilnet_v2 model [11] is implemented with an addition of a custom fully connected Layer. It introduces a new convolutional neural. Network with inverted residual and a linear bottleneck layer. This provides better performance and high accuracy in mobile and object detection applications.



Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

Figure 1 Pretrained mobilenetv2 architecture [11]

Each line describes a sequence of 1 or more identical (modulo stride) layers, repeated *n* times. All layers in the same sequence have the same number *c* of output channels. The first layer of each sequence has a stride *s* and all others use stride 1. All spatial convolutions use 3×3kernels. The output of this is 1280 and is given as input to the classifier.

B. Classifier

We have implemented our custom classier shown in the figure below with five hidden layers. The hidden nodes in each layer are of size [512,256,128,64,32]. Each layer is accompanied by a ReLU activation. Dropout layer is implemented after the first and last hidden layers. The probability for an element to be zeroed is given as 0.3. The classification problem is a binary classification as we have only two classes. So, the output of the classifier is the probability of which class the input belongs to.

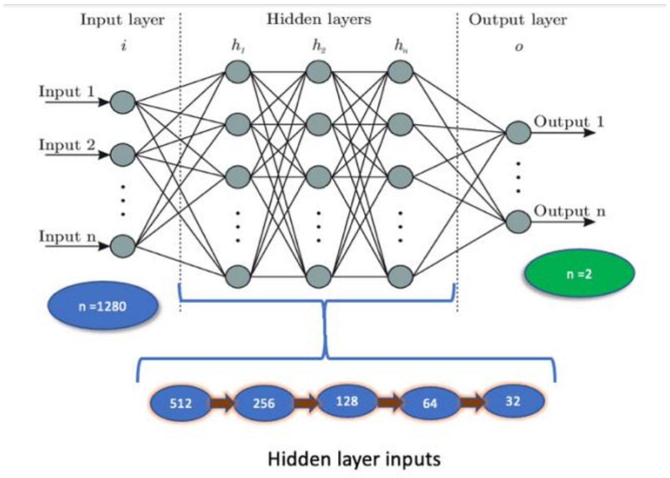


Figure 3 Classifier

VI. Face Detection Architecture

This is the first stage of our network. This stage determines if the input contains a human face. We have used a pretrained face detection model from caffe library. It is a Single shot multibox detector with a backbone of ResNet10 architecture. Below figures from [12] provides a representation of SSD based on Resnet10. The output of this architecture would be faces found in the input and their positions.

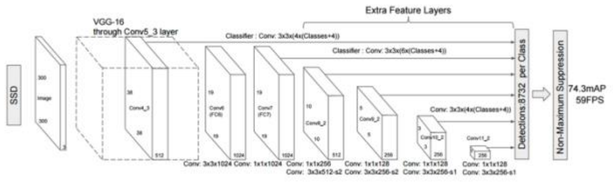


Figure 2 SSD

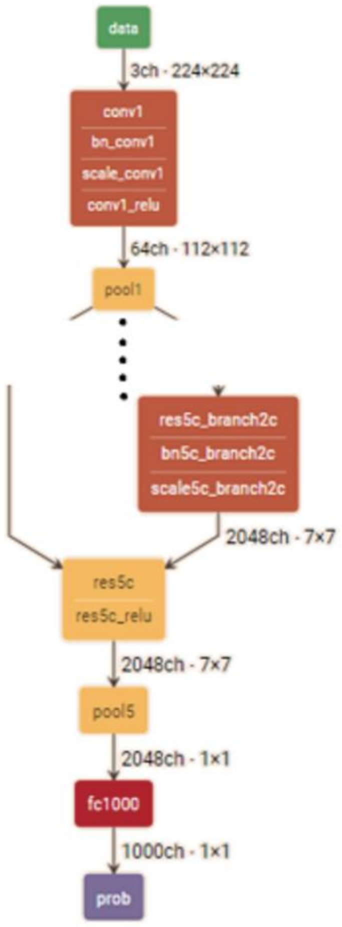


Figure 4 ResNet10

VII. Training

Now that we have defined and implemented our network. We train the model the model on the dataset previously discussed. We will discuss the training parameters here. We resize all the images to be of size (224,224). We have trained the model using Adam optimizer with *learning rate = 0.001*. As our network is binary classification, we have implemented binary cross entropy loss as the loss function. The training was implemented on PyTorch framework. Binary cross entropy loss is defined as BCELoss () in pyTorch. We have trained the model for *100* epochs. The batch size for the train dataset is *64* and test is *8*. The total number of parameters the model consists including the mobilenet_v2 is *830498*. In the below figure 5 we can observe the training loss and the validation loss.

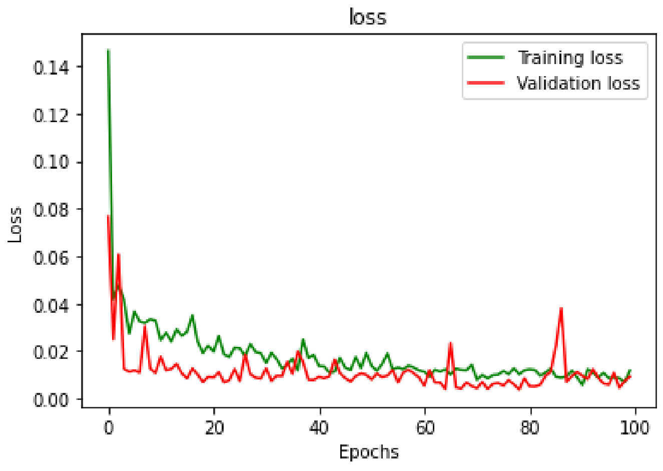


Figure 5 Loss vs Epochs

VIII. Testing

We save the model with all the weights in it. Testing the model gave an accuracy of 99.69% where 989 images were predicted correctly out of 992 images. For testing the model on real images, we have chosen a few images where components of blurriness and position of the face is not in symmetry.

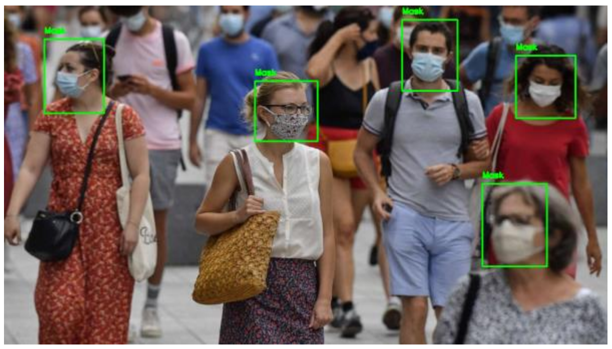


Figure 6 Test

We can observe from the image (*fig 6.*) the model was correctly able to predict the mask even when the position of the faces is not forward facing. We can observe that the person's face is blurry, but the model had detected correctly.

Another test case shown in figure below (*fig 7.*). We can observe that person at the left end is undetected because our training dataset doesn't contain significant number of images where some part of the face mask is hidden.

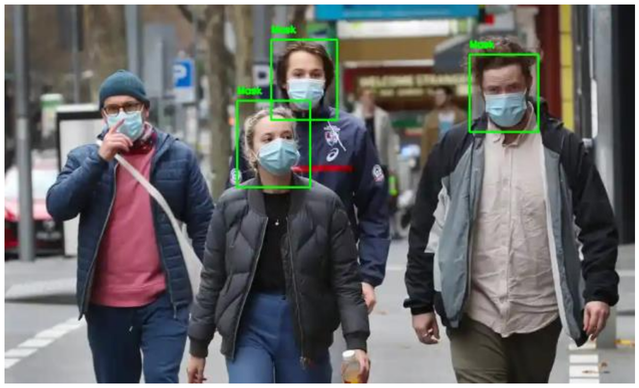


Figure 7 Test

IX. Conclusion and Future Scope

In this project we have successfully implemented a real time face mask detection by a two stage CNN based classifier. We have implemented this using a pretrained face detection model and our custom classifier with pretrained mobilenet_v2 model. We have tested the model on images, but the work can be extended to videos as well. This network can be installed in any public place and can be useful for monitoring the people not wearing masks. To improve the model and better the efficiency, for face detection we can use a supposedly better model like MTCNN (Multi-task Cascaded Convolutional Neural Network). Another way to improve the accuracy is to train the model on a dataset which contains images which are not biased, and the color of the mask is more diversified. The datasets which are available currently are more augmented and artificial. For better network model we can train on a better dataset.

This project can be further extended by adding another class where a mask is incorrectly worn. Many times, we observe that people don't wear a mask as expectedly this can result in the transmission of virus. This will help to determine more efficiently whether a person has worn a mask or incorrectly worn a mask.

REFERENCES

[1] Dalal, N., and Triggs, B., Histograms of oriented gradients for human detection, CVPR '05, 2005.

[2] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L., ImageNet: A Large-Scale Hierarchical Image Database, IEEE Computer Vision and Pattern Recognition (CVPR), 2009.

[3] Ejaz, M.S., Islam, M.R., Sifatullah, M., Sarker, A., Implementation of principal component analysis on masked and non-masked face recognition, 1stInternational Conference on Advances in Science, Engineering and Robotics Technology (ICASERT),2019, pp. 1–5.

[4] Girshick, R., Fast R-CNN, Proceedings of the IEEE International Conference on Computer Vision,2015, pp. 1440–1448.

[5] https://developer.nvidia.com/blog/implementing-a-real-time-ai-based-face-mask-detector-application-for-covid-19/

[6] [Amit Chavda, Jason Dsouza, Sumeet Badgujar, Ankit Damani., Multi-Stage CNN Architecture for Face Mask Detection, 2020.

[7] https://github.com/NVlabs/ffhq-dataset

[8] https://www.kaggle.com/andrewmvd/face-mask-detection

[9] Cabani, A. et al. "MaskedFace-Net – A dataset of correctly/incorrectly masked face images in the context of COVID-19." Smart Health (Amsterdam, Netherlands) 19 (2021): 100144 - 100144.

[10] https://medium.com/@tomstaite1/face-mask-detection-algorithm-using-convolutional-neural-network-ai-computer-vision-15f08988533e

[11] Sandler, Mark & Howard, Andrew & Zhu, Menglong & Zhmoginov, Andrey & Chen, Liang-Chieh. (2018). MobileNetV2: Inverted Residuals and Linear Bottlenecks. 4510-4520. 10.1109/CVPR.2018.00474.

[12] https://pytorch.org/hub/nvidia_deeplearningexamples_ssd/

[13] https://pytorch.org/docs/stable/generated/torch.nn.Dropout.html

[14] https://towardsdatascience.com/a-guide-to-face-detection-in-python-3eab0f6b9fc1

[15] https://github.com/timesler/facenet-pytorch

[16] B. Yang, J. Yan, Z. Lei, and S. Z. Li, "Aggregate channel features for multi-view face detection," in IEEE International Joint Conference on Biometrics, 2014, pp. 1-8.

[17] Q. Zhu, M. C. Yeh, K. T. Cheng, and S. Avidan, "Fast human detection using a cascade of histograms of oriented gradients," in IEEE Computer Conference on Computer Vision and Pattern Recognition, 2006, pp. 1491-1498.