Face-mask Detection using Deep Neural Nets

Yash More Sehajpreet Kaur Shweta Prasad

I. Introduction

Facial detection and recognition are well-posed learning problems. There are several state of the art facial detection and recognition software widely in use and now are commonplace in several applications that run on our mobile devices. While its ubiquitous nature raises several serious privacy and security concerns, work is still being done to recognize faces that are partially occluded, or are from low-resolution input sources, with the cheapest possible compute. Partial occlusion - be it by means of wearing masks or hiding certain facial features to evade detection makes facial recognition an interesting problem. False-positive identification of faces when there aren't any in the input image, or non-identification of faces when presented with a semi-occluded image of a face are two commonly met problems when training facial detection networks.

As a result of the coronavirus pandemic, wearing masks has become an imperative and the new norm. It is one of the cheapest ways to keep oneself as well as others safe from possible infection. This change in scenario presents facial recognition with new challenges, as well as possible use cases. Since countries all over are relaxing the clampdown, re-opening public spaces, and letting people get back to their physical work spaces, it is of utmost importance to keep as many controls as possible on preventing a resurgence of spread. Of the several measures enforced in public spaces, wearing masks is a crucial one. In workplaces where automatic sign-in and entry procedures are done via facial recognition, this becomes a challenge. On the other hand, recognizing whether a face has a mask or not, becomes a useful tool in surveilling the spread of the disease. Particularly, warning people that someone is not wearing a mask in their vicinity; or delivering personal alerts and reminders to put on masks in public spaces if the user of the service themselves are not wearing masks; or using public surveillance cameras to identify the number of people in a given area that are (not) wearing masks and calculate the probability of a spread originating from that location, become possible use cases. While increased surveillance is worrying, superficial surveillance as mentioned in the last use case is indeed the need of the hour, especially in developing countries with large populations.

In this interest, we took up the task of trying to build a face-mask detector that is capable of identifying whether people in a given image are wearing masks or not. We then wanted to put the detector to use in live video streams, so that one, live feedback could be given and two, unique identification of the number of faces wearing masks at a given instance was possible. To do this, we made use of convolutional neural nets and transfer learning. Additionally, we made it a point to use computationally efficient architectures such as MobileNetV2 so that we can possibly extend its usage to smaller, common devices such as smartphones and laptops.

II. Related Works

Several works with elements related to our proposed task exist. However, not many of these explicitly focus on developing a face mask detection model for limited compute resources.

Loey et. al proposed a hybrid model based on deep transfer learning for feature detection using ResNet50 and classical machine learning algorithms like decisions trees, Support Vector Machines (SVM) and the ensemble algorithm for face mask detection. They concluded that the SVM classifier performed the best by achieving the highest possible accuracy with the least time consumed in the training process. They used three datasets including the RMFD dataset used by us. Training was done on a specific dataset and the model was then tested on the other two. The SVM classifier in RMFD (Real-World Masked Face Dataset) achieved 99.64% testing accuracy; in SMFD (Simulated Masked Faced Dataset), it achieved 99.49%; while in LFW (Labelled Faces in the Wild), it reached 100% testing accuracy [1].

Petrovic and Kocic proposed a computer vision subsystem based on Raspberry Pi for detecting face masks as part of a larger IoT-based system to monitor Indoor Safety against COVID-19. They used Haar-cascade Detection in OpenCV to first, detect the human face frontally and then, detect the mouth and nose specifically. A camera frame with a frontal face where the mouth and nose cannot be detected implies a face mask. They extended this to work with multi-person video streams. They achieved an accuracy of 84-91% for face mask detection [2].

Upon further review of existing literature, it became evident to us that for an application aimed at public use, if we were to deploy our models, it should be able to run efficiently on cheap resources. MobileNet and its more recent variants, originally developed by Google Inc. scientists, are the current standard when it comes to CNN based mobile

vision applications. It is computationally much less expensive than many other network architectures, particularly because it relies on depth-wise separable convolutions, rather than spatially separable convolutions. The use of depth-wise separable convolutions, as explained in Howard, et. al (2017) guarantees several orders fewer mult-adds, which makes it relatively more feasible for embedded systems and low compute machines, than architectures where the convolutions are spatially separable. Increasing the channel depth of the output of a convolution becomes computationally viable for deep networks because depth-wise separation of the operations reduces the total number of multiplications and additions in calculating the output of the convolution [3]. Further improvements were made to this architecture in MobileNetV2, such as the addition of linear bottlenecks and inverted residual layers that add shortcut connections between the thin bottleneck layers [4].

For the first part of our model, the face detector, we turned to the OpenCV DNN module implementation of a ResNet10 based Single Shot MultiBox Detector model [5] which uses a single deep network to detect objects. Their approach (named SSD) generates scores for the presence of each object category at prediction time and correspondingly, adjusts to the box to match the shape of the object better. SSD outperformed state-of-the-art models such as Faster R-CNN. Compared to other single stage methods, SSD has much better accuracy even with a smaller input image size.

III. Experimental Design

The aim of this experiment is to efficiently perform face-mask detection, taking into consideration the limited availability of computation.
We can split the problem into two sub-tasks:

- Perform face detection on 2D images
- Mask detection and recognition on the detected faces

The first task of this pipeline entails using pre-trained models to perform face-detection. The second task involves us preparing a custom classifier for the binary classification task of detecting whether or not a face is masked. We describe the setup and techniques used to tackle our second task below.

IV. Setup and Techniques

1) Model: Our model is comprised of two sub-parts. The primary part relies on MobileNetV2, and a custom model, which gets appended at the end of the base model.
MobileNetV2 retains the core idea of the MobileNetV1 i.e of depth-wise separable convolutions. The depth wise-separable convolutions divide the job a full-convolutional layer into two sub-stages. The first stage performs a depth-wise convolution which doesn't combine all the channels of the input into one, instead it performs the convolution on each channel separately. This is followed by a point-wise convolution which performs a weighted addition of the channels which results in a convolution which is almost same but computationally much faster. MobileNet V2 however makes certain alterations to make the process more computationally efficient. The MobileNetV2 wraps the depth-wise convolution with an expansion layer, and a projection layer. This layer expands the input tensors to higher dimensions, via an expansion factor which can be regulated. This is then passed on to the depth-wise convolution layer. Finally, the output is shrunken to lower dimensions via the projection layer. Apart from the addition of new layers, the MobileNetV2 also uses a residual connection to help with the flow of gradients through the network(akin to the ResNet architecture).

Our custom model replaces the top layers of the base (MobileNetV2) with an average pooling layer, followed by a fully-connected block comprising of a two dense layers. We use dropout right after the dense layer, to prevent the model from over-fitting. Finally, the output is passed through the soft-max activation function, to suit our binary-classification task. We customise this model by adding a very small number of layers as compared to our base model model, to tune it to perform classification, while the heavy lifting is done by the feature extractor i.e the base model.

2) Datasets: We use two datasets namely D_A, and D_B, in our experiments. D_A is a publicly available dataset with 1,376 total images, labeled as with 'with_mask' and 'without_mask', corresponding to images with and without masks respectively. [1]

D_B is derived from another publicly available dataset named `RWFMD` or Real-World Face Mask Dataset [6]. We curated a custom selection of images from RWFMD dataset, as it isn't completely noise free, and contains incorrect labelling, redundant pictures etc. We joined this curated dataset with our earlier dataset to make it larger, with a total of 3230 'with_mask' and 3250 'without_mask' images.

Fig. 1. Sample images from D_A

[1] https://github.com/prajnasb/observations/tree/master/experiements/data

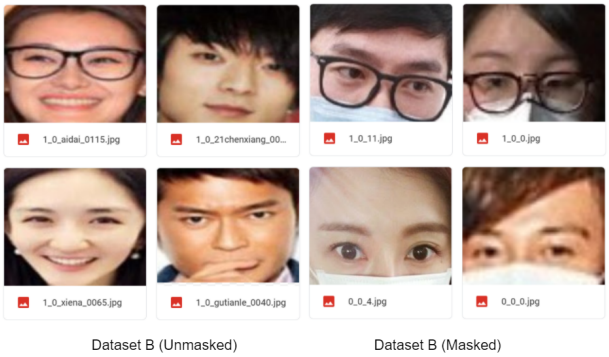

Fig. 2. Sample images from D_B

In all, our model was trained on roughly 7000 images of masked and unmasked faces taken from different angles and of various resolutions. We noticed two possible discrepancies that could affect model output, especially while working with live video streams. Firstly, there was very little variety in terms of the types and colours of masks. Most masked images in both datasets (augmented or otherwise) used light coloured surgical or N95 masks. Secondly, almost all images were taken in artificial or natural light with barely any in poor/dim lighting.

3) Pre-processing: Pre-processing involves preparation of a relatively clean dataset, along with steps of data-augmentation and coercing the data into standard formats and dimensions.

In our case, data augmentation becomes paramount, as we only have access to a limited amount of data. Data augmentation involves various transformations applied to the input space, such as rotation, shear, color space manipulation, random erasing etc. At the end of this, we obtain a larger dataset with each augmented copy varying enough to be representative of the different types of images that we can encounter in the real world.

Next, we need to ensure that all of our data was of valid input dimensions, so as to ensure compatibility with the different pre-trained models we utilised, such as MobileNetV2. In this step, we standardise the input dimensions to 224×224, which is compliant with the input dimensions of the first layers of our pre-trained models.

The images were then put into respective folders for 'with mask' and 'without mask' for easy access, and one-hot encoded with these labels.

A. Training

In this section, we will focus on the specifics of training different models for our task. To reiterate, we tested several models and combinations so as to pick the one best suited to our task.

For the entire training process we rely on Google Colaboratory and Jupyter Notebooks, each equipped with NVIDIA Tesla K80, coupled with 16GB RAM.

To train our face-mask classifier, we primarily utilised a baseline model using the MobileNetV2 architecture, trained on the ImageNet dataset. This model was fine-tuned to our two class classification problem on the different datasets mentioned.

Before training, we perform a 80-20 split of the datasets D_A and D_B. The 20% is used for testing purposes later. As our input labels are already one-hot encoded in the pre-processing stages, we train the model in lieu with the expectation of performing well on the binary classification task of detecting whether or not a face has a mask. The training process remains fairly similar across other models we tried, such as:

- Efficient Net
- Shuffle Net
- VGG-16
- SVM

The Results section will delve further into our findings with respect to each model.

Once a satisfactory classification accuracy was reached in each case, we serialized the resultant model and transferred it to be used in conjunction with a face detection model. For the face detection part, we tested several pre-trained models such as the default Haar-cascades model for full frontal face detection, dlib frontal face detector and a ResNet10 based Caffe model using OpenCV's DNN module. This way, we were not only able to compare the performance of different combinations of models, we were also able to observe which ones were suitable to what level of resource availability.

V. Observations and Results

A. The Face-Mask Classifier model as a stand-alone

We observe that the MobileNetV2 model performed the best amongst others we had tried, keeping the training process unchanged.

- MobileNetV2:
  - D_A: After training the model for 20 Epochs, we achieved a good training accuracy of 0.99 and and a validation accuracy of 0.979. We also noticed that the confusion matrix (Figure 4) via the test dataset was mis-classifying only a very low percentage of cases.
  - D_B: While testing out the model on the second dataset, we observed that the model start over-fitting after the 9th Epoch. Hence, we limited the number of epochs to 5-8. The model performed decently on the larger dataset reporting a training accuracy of 98.2 and validation accuracy of 96.02, however

Fig. 3. 3d representation of our model

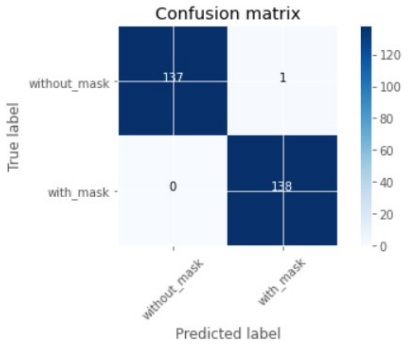

Fig. 4. Confusion Matrix for D_A on MNV2

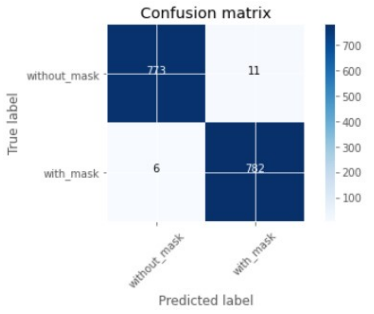

Fig. 5. Confusion Matrix for D_B on MNV2

there was fair increase in the mis-classified images as observed from the confusion matrix in Figure 5.

- Efficient Net:
  - D_A : The model gave a training accuracy of 0.6 and a validation accuracy of 0.49, when tested on around 15 Epochs. Reducing the number of epochs didn't affect these results, and we inferred it might be because the model is under-fitting.
  - D_B : Shifting to a comparatively larger dataset didn't change the training and validation accuracy, as training accuracy was reported at a staggering 0.74, and validation of 0.55.
- VGG-16: The model performed very poorly on both the datasets giving an accuracy of 0.55 training. This might be explainable due to the large number of parameters in comparison to Efficient Net and MobileNetV2.

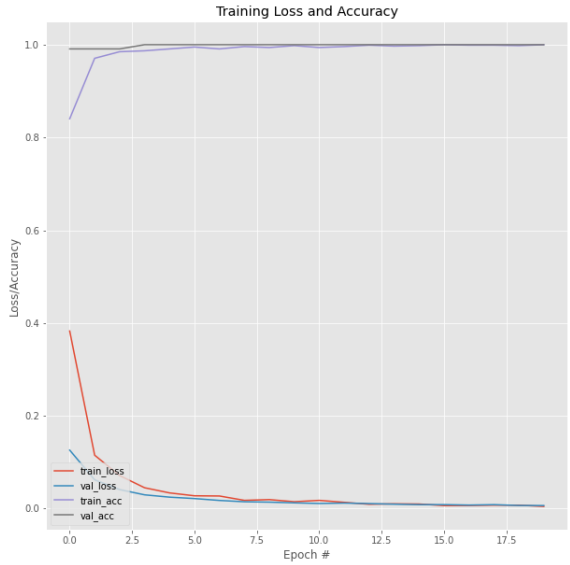

Fig. 6. Training and Validation metrics for D_A on MNV2

- SVM: To compare classical image classification methods to DNNs, we tried fitting an SVM model to D_A and D_B. We observe that even though we were getting an accuracy of around 0.7 in this case, there was a significantly large mis-classification error in comparison to other methods.

After testing out the other models, we observed that our model generalizes fairly well despite the limitations of the size of datasets we trained on. We also observe that keeping our classifier part of the model architecture restricted to few layers helped us in getting a better accuracy – including more layers as the head of the model was not beneficial to our training, and was giving lower accuracy as compared to the smaller head.

B. The live-stream face-mask detector

On the video streams, the ResNet10-MobNetV2 model trained on the smaller dataset D_A performed significantly

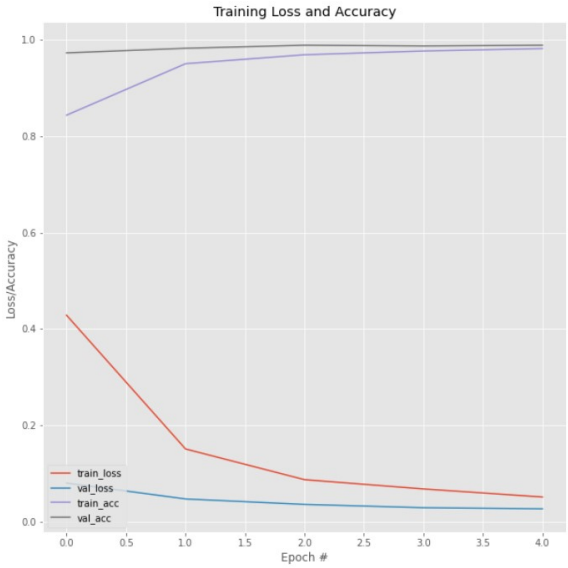

Fig. 7. Training and Validation metrics for D_B on MNV2, after stopping at 8 epochs

TABLE I
PERFORMANCE OF OUR PRIMARY MODEL

Dataset	Training Accuracy	Testing Accuracy
D_A	0.979	0.99
D_B	0.96	0.98

better and the bounding boxes produced were far more stable, than in case of the other model combinations trained on the same dataset, as well as the same model combination trained on larger datasets;

Testing just for the face-detection task, the Haar-cascades frontal face detector was better than the ResNet10 face detection model only at detecting multiple faces in images of crowds, and more than 5 people. In all other scenarios, the latter model gave face detections with confidences higher than 95%, with a lower number of observed false positve detections. However, in the multiple face scenario, this model failed to identify all faces in the image unless the minimum confidence for face identification was significantly lowered (in some cases, even to 20%). It is worth mentioning that the standard minimum confidence we used for face detection was 65%. While we initially tested the dlib frontal face detector, its poor performance on face-detection given multiple faces in static images suggested to us that we not expend time on using it further.

With respect to the mask classification task, we output the confidence of classification along with the bounding box identifying the face, so we could gauge performance in real time. Our metrics as observers were the stability of the bounding boxes, confidence of classification and false positives and false negatives, under varied conditions presented in the video stream.

Models other than the ResNet10 based Caffe model (including the Haar-cascades) did not perform well on quick movements, varying angles, and non-frontal face detection. The ResNet10 model performed better in different lighting conditions. However, it gave false positives. Additionally, ResNet10 performed better than Haar-cascades in a scenario where the face was occluded by hands or different objects. We did not test the effect of different frame rates on the performance of the models.

The models primarily tested on video streams performed only mediocrely well at detecting multiple faces with masks, especially when occlusions such as hair, hands, clothes, transparent bottles, etc were brought in. Additionally, we also observed a marked difference in reported confidence of classification when the mask used was of darker shades. Some times, dark shades of masks served to offset face detection itself. A clear bias towards white masks and mask-like object placement was observed and it could throw the machine off very easily to give a false positive classification of "Mask on". This could possibly be explained by the fact that the datasets we trained the face-mask classifier on, primarily contained images where the masks were white/of lighter shades. We were also able to observe that a central placement of the face in the video frame guaranteed better chances of detection and proper classification, as opposed to faces in the far left or right corners.

Fig. 8. Model classification of random images outside of primary datasets

VI. CONCLUSION

While we reached our end goal of shaping and applying our model to real scenarios, the rate of positive face detection in case of multiple faces, especially if images are blurred or if the quality of the input image varies in different regions, was too low to be of any real use in devices such as street CCTV cameras, where we envisioned the use of such a classifier. For this, we would have to focus on low resolution images as input, collect appropriate data and identify augmentations or changes to the current state of the art pipelines to suit the

problem better. This classifier, however, is useful in an office scenario, where identifying workers becomes paramount. It could be put to use in an alarm/warning system where people who are not wearing masks can be requested to wear them, or to report it to other workers to be careful, especially in our current contexts.

For this project, we made use of pre-trained models cognizant of our limited compute, storage and time resources. The deep networks that are suitable to object identification and image classification tasks such as these are very compute intensive, and we simply do not have the resources. On the other hand, we were able to learn about different architectures, approaches and optimizations offered by them.

In continuum with the task undertaken here, we think that this model could be extended to report the extent to which the mask covers the face, giving us the opportunity to give corrective feedback to the user of such an application. It should not be a difficult task, especially given how ubiquitous and well functioning face filter applications are currently. [7]

In this report, we described a two-phase pipeline to identify and classify whether a given face is masked or not. In the future, work can be done to reduce this to a single phase task, instead of first identifying the face, and then running a classifier on it to detect object X's presence. Much like the Single Shot Multiple Box Detector reduced the face-identification pipeline to a single model, we could try to build a model that does not require multiple phases to be able to do the task at hand.

ACKNOWLEDGMENT

We would like to thank Prof. Ravi Kothari for his guidance and enthusiasm. We are grateful for the opportunity to explore a lot of different techniques in lieu of this project, and capitalise our learnings from the Introduction to Machine Learning Class. We would like to thank the course Teaching Fellow – Chanda Grover, and Teaching Assistant – Vedansh Priyadarshi for all the help through out this course, and assistance in picking an approach for this project.

Finally, we would like to thank Google Colaboratory for existing, without which this project would have been impossible.

REFERENCES

[1] M. Loey, G. Manogaran, M. H. N. Taha, and N. E. M. Khalifa, "A hybrid deep transfer learning model with machine learning methods for face mask detection in the era of the COVID-19 pandemic," *Measurement*, vol. 167, p. 108288, Jan. 2021. [Online]. Available: https://doi.org/10.1016/j.measurement.2020.108288

[2] N. Petrovic and Kocić, "Iot-based system for covid-19 indoor safety monitoring," 09 2020.

[3] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," 2017.

[4] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," 2018.

[5] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," 2015.

[6] Z. Wang, G. Wang, B. Huang, Z. Xiong, Q. Hong, H. Wu, P. Yi, K. Jiang, N. Wang, Y. Pei, H. Chen, Y. Miao, Z. Huang, and J. Liang, "Masked face recognition dataset and application," 2020.

[7] A. Rosebrock, *Deep Learning for Computer Vision with Python*, 1st ed. PyImageSearch, 2017, vol. 3-ImageNetBundle. [Online]. Available: http://gen.lib.rus.ec/book/index.php?md5=7db52e410ee84ec5a30b3190e73b61d3