WEARMASKNET: REAL-TIME FACE MASK DETECTION

Ethan Robert A. Casin
Master of Science in Data Science
Asian Institute of Management
ethancasin.msds2021@aim.edu

Daniel Rey D. Cimafranca
Master of Science in Data Science
Asian Institute of Management
danielcimafranca.msds2021@aim.edu

Hilbert S. Lazatin
Master of Science in Data Science
Asian Institute of Management
hilbertlazatin.msds2021@aim.edu

Rea P. Tanguilig
Master of Science in Data Science
Asian Institute of Management
reatanguilig.msds2021@aim.edu

Maynard F. Vargas
Master of Science in Data Science
Asian Institute of Management
maynardvargas.msds2021@aim.edu

Thursday 28th January, 2021

ABSTRACT

With the advent of the COVID-19 pandemic, face masks have become an integral part in combating the transmission of the virus. As outdoor activity increased due to more relaxed restrictions, people tend to wear face masks as a means of entry to establishments which calls for a proactive approach in regulating violators. In this project, we have created a mobile deep learning model for face mask detection. It identifies three classes, namely: with mask, no mask, and incorrectly worn. This model would help regulators identify violators and ensure public health proactively. Additionally, the model can be deployed to CCTVs or even phones. Our implementation uses MobileNetV2 to make the training faster. Moreover, our model has a 99.26% test accuracy for our dataset and about 2.3 million parameters, which is lighter than other models in literature.

***K*eywords** COVID-19 · Deep Learning · Face Mask Detection · MobileNetV2

1 Introduction

During the COVID-19 lockdown, we see that there is a significant decrease in human activity worldwide. People are staying at home and only going out to purchase necessities. Governments effectively closed down its borders and imposed a nation-wide restriction that limit human activity. In fact, these two aspects of human activity are critical in reducing the transmission of the virus [1].

However, we cannot completely remove human activity outdoors. Face masks and other hygienic items (i.e. alcohol sprays and face shields) are recommended measures for going to public spaces and establishments. Currently, we

observe more human activity as nations started declaring to be COVID-19 free and more efficient response methods. This is also part of the problem, since people tend to wear masks improperly or not at all, which make it seem that face masks are a means of entry. It also was found that exposing the nose is already harmful to public health [2, 3].

These reasons require proactive measures, especially for large crowds. Manual inspection from security officers tend to be inefficient and prone to errors. This study aims to create a mobile deep learning model to identify people who wear masks, who does not wear a mask, and who wears it improperly. The main benefit of the improperly worn masks is that people tend to expose their nose while wearing face masks. Moreover, this would be beneficial for establishments that normally have a large population of customers (i.e. malls or parks).

2 Methodology

Figure 1: WearMaskNet Pipeline

Overview. As shown in our pipeline in Figure 1, given a video frame, our architecture, called "WearMaskNet", generates a modified video frame with bounding boxes on detected faces with a corresponding face mask classification as mask-on, masked-off and incorrect-mask (incorrectly worn masks). Our method is composed of two different modules which are the Face Detection Module and the Face Mask Classification Module.

2.1 Dataset

There are two data sources that we utilized. The first data source we used was the **Face Mask 12K Images Dataset** by Ashish Jangra from Kaggle. This dataset contains only two classes of images, namely: faces with masks and faces without a mask.

The other data source we used was from the **MaskedFace-Net** group [4], where we downloaded incorrectly masked images for our third face mask classification class. The incorrectly masked class is composed of different variations of incorrectly worn masks. The first variation is composed of faces with uncovered nose only while the second variation is composed of faces with uncovered nose and mouth. Lastly, the last variation shows a face with a face mask but it only covers the chin. There are around 80% of the of the first variant in the incorrectly masked class while the other variations compose 10% each.

We then separated the images into training, validation and test sets. Below in Table 1, you can see the number of images per class and split.

Split	Class	Number of Images
Training	With Mask	5,000
	Without Mask	5,000
	Incorrectly Worn Mask	4,820
Validation	With Mask	400
	Without Mask	400
	Incorrectly Worn Mask	400
Test	With Mask	483
	Without Mask	509
	Incorrectly Worn Mask	500

Table 1: Overview of the dataset

2.2 Pre-processing

All of the images of faces were preprocessed and resized to 108x108 RGB image. Data augmentation was also done such as scaling, offsetting, mirroring and zooming but for our final model, we didn't do any data augmentation because it didn't improve the performance.

2.3 Face Detection Module Creation

For the Face Detection Module, we experimented with different face detection algorithms. This module's objective is to extract the faces from the video frames/images it is fed as seen in Figure 1. The first one we used was the Haar Casscade which is a face detection algorithm using haar-like features as explained previously. We used this for its speed of face-detection. We also used MTCNN which is as neural network that was developed as a solution for both face detection an alignment. MTCNN balances the speed and robustness in terms of face detection compared to the other face detection algorithms that was used in this study. Lastly, we used RetinaFace which is a robust single-stage face detector which performs pixel-wise face localization in different scales of faces. This was the most robust solution for face detection but it takes a lot longer than the previous methods. Furthermore, detecting faces wearing masks incorrectly was challenging for the three methods as the bounding box obtained may not capture the masks located below the chin or nose. As such, we made some tweaks such that the bounding boxes dimensions are increased to captuure the entire face with the mask.

2.3.1 Haar Casscade

Implementing an object detection method is important for the model to sift through various object images and classify them. For this purpose, an effective method for object detection is a *Haar Cascade* classifier.

This classifier uses a cascade function, which is trained in numerous positive and negative images. Thus, a Haar Cascade classifier identifies objects in images based on learned features [5, 6]. In this study, a pre-trained Haar Cascade Classifier from the OpenCV python library was used.

2.3.2 MTCNN

Multi-task Cascaded Convolutional Network (MTCNN) is a widely known algorithm used in face detection and alignment. The algorithm utilizes three stages of deep convolutional neural networks with a cascaded structure that predicts face and landmark locations in a coarse-to-fine manner [7]. It was also shown that MTCNN can achieve higher accuracies than state-of-the-art models, while maintaining real-time performance.

The model begins by using a convolutional neural network (CNN) to obtain the candidate windows and bounding box regression vectors. The bounding box regression vectors are then used to calibrate the candidates. Highly overlapping candidates are merged using non-maximum suppression (NMS).

Resulting candidates from the first stage are then fed into another CNN, known as Refine Network, which removes the remaining false candidates. This stage also calibrates the bounding box regression and further merges highly overlapping candidates through NMS.

The last stage works similar to the second stage, but the network will output five facial landmark positions of the image. A pre-trained MTCNN model was implemented via the MTCNN library which is also based on Tensorflow.

2.3.3 RetinaFace

RetinaFace is a single stage face detector which performs pixel wise localisation and employs a multi-task learning strategy to simultaneously predict face score, face box, five facial landmarks, and 3D position and correspondence of

each facial pixel [8]. The RetinaFace detection algorithm was implemented via the insight face library in Python, in which a pretrained retinaface model is readily available.

2.4 Face Mask Classification Module Creation

The Face Mask Classification Module classifies images of faces extracted by the Face Detection Module from Video Frames as shown in Figure 1. This module is crucial to flag the faces detected as people who are wearing masks, not wearing masks or incorrectly wearing a mask. We created this module using a MobileNetV2 as the base model and we then used a max pooling layer and fully connected layer to predict the three classes.

2.4.1 Architecture

As explained previously, we made use of MobileNetV2 as the base model for our image classifier. We used this because of its lightweight nature and because we found that using pre-trained models have better classification accuracy. We just used the Keras' implementation and weights for MobileNetV2. We changed the input shape to 108x108x3 as this is the shape of the images that we will feed to the classifier. We just added a simple max pooling layer and flattened the output to feed it to a fully connected layer after. In total, we have around 2,340,163 parameters but the only trainable ones are 82,179 which greatly increases the training speed. The full architecture can be summarized by the image in Figure 2.

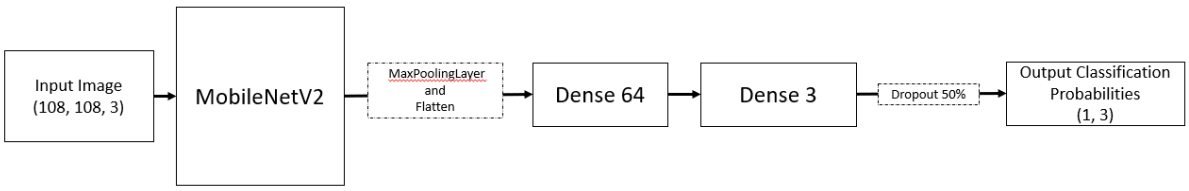

Figure 2: Model architecture

2.4.2 Training

The model was trained with 20 epochs only given that we are only training the head of the model and the base model (MobileNetV2) was set to not trainable. We used an Adam optimizer with 0.0001 as the initial learning rate and with a decay of initial learning rate / number of epochs. We used the loss of categorical cross-entropy given we have three classes for our classifier. We saved the model weights in our models folder named final_model.h5.

2.4.3 Adam Optimizer

In model building, the choice of an optimization algorithm can make or break results and training time. An effective and efficient optimizer is the Adaptive Moment Estimation, also known as, *Adam* which computes for adaptive learning rates of each parameter. From [9], Adam is a method for stochastic optimization that updates exponential moving averages of the gradient (m_t) and the squared gradient (v_t). These moving averages, m_t and v_t, are also the estimates of the first moment (mean) and the second moment (un-centered variance) of the gradient. These moments are defined as

$$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$$
$$v_t = \beta_2 \cdot m_{t-1} + (1 - \beta_2) \cdot g_t^2.$$

The hyper-parameters $\beta_1,\ \beta_2 \in [0, 1)$ are the exponential decay rates for the moment estimates and $g_t = \nabla_\theta f_t(\theta)$ is the gradient of a noisy objective function f_t with parameter θ. The authors note that the moving averages are initialized as zero vectors, which biases the estimates towards zero. This can be resolved using the bias-corrected estimates $\hat{m}_t$ and $\hat{v}_t$ for the first and second moment, respectively. These estimates are given by

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$
$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t},$$

where β_1^t and β_2^t are the exponential decay rates raised to the power t. Furthermore, Adam updates the parameters using the rule

$$\theta_t = \theta_{t-1} - \alpha \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon},$$

where α is the learning rate and ϵ is a constant that avoids a division by zero error and prevents the gradients from blowing up. The authors suggest to set $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, and $\alpha = 0.001$ as these were the settings that produced good results based on their testing. For our project, we set the learning rate to 1×10^{-4} and a decay rate of 5×10^{-6}.

2.4.4 Categorical Cross-entropy Loss

For a multi-class classification problem it is appropriate to use the categorical cross-entropy loss function for model learning. This loss function tries to differentiate between two probability distributions. The categorical cross-entropy loss L is defined as

$$L = -\sum_{i=1}^{N} y_i \cdot \hat{y}_i,$$

where N is the output size, $\hat{y}_i$ is the i-th predicted output of the corresponding target value y_i.

2.4.5 Accuracy

Given the balanced nature of the dataset, the primary measure would be the model's accuracy. Given a $n \times n$ confusion matrix C, the accuracy for a multi-class classifier is defined as

$$\text{Accuracy} = \frac{\text{tr}(C)}{\sum_{ij}^{n} c_{ij}},$$

where $\text{tr}(C) = \sum_{i}^{n} c_{ii}$ is the trace of C. The denominator is the sum of all elements c in C for each column j and row i.

3 Results and Discussion

3.1 Face Detection

In this section, we explored three pre-trained face detection algorithms namely, the HAAR cascade, MTCNN and RetinaFace and compared these in terms of average inference time and face detection capability. The average inference

time is calculated based on ten 720p images for varying crowd sizes. This would also help us gauge the ability of the model to scale with large crowds. The average inference time are summarized in Table 2.

3.1.1 Inference Time

Model	1 person	3 persons	10 persons
Haar Cascade	0.156	0.196	0.277
MTCNN	1.110	1.219	1.305
RetinaFace	5.779	6.377	5.823

Table 2: Average inference time (in seconds) for a 720p image with different crowd sizes

From the results in Table 2, we see that the HAAR cascade performed best in terms of speed. MTCNN is second and RetinaFace performed the worst. However, run time is not the only metric that needs to be considered. The success of the face detection algorithm would also depend on the face detection accuracy which we demonstrate in the next section.

3.1.2 Sample Results

Sample results for the face detection algorithm are shown in the figure below.

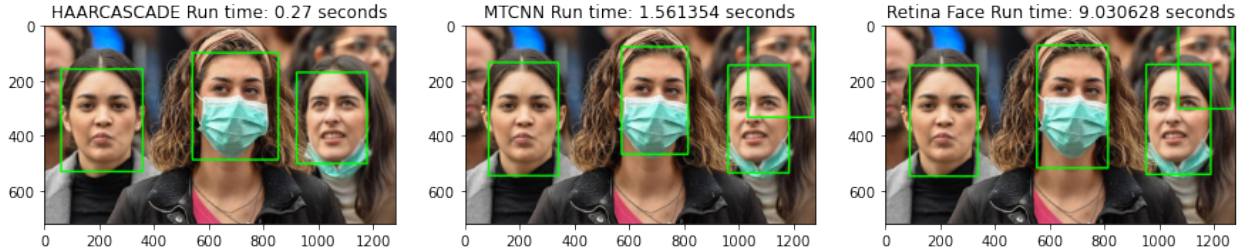

Figure 3: Comparing Face Detection Results for an Image with 3 persons

From the image, we see that all three models can detect the faces properly. The model performance disparity, however, becomes more apparent for a larger crowd as shown below.

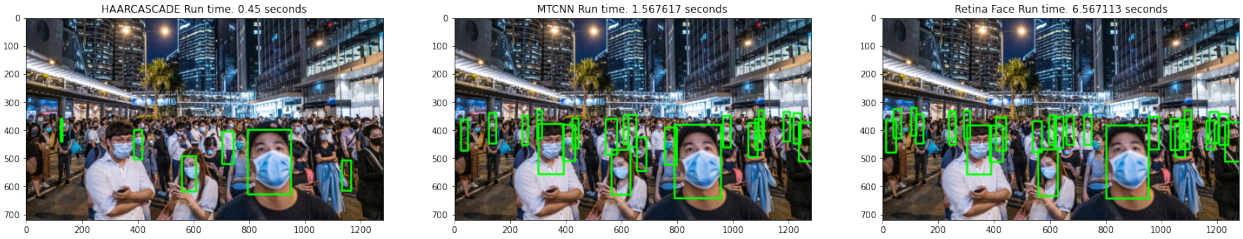

Figure 4: Comparing Face Detection Results for an Image with 10 persons

As shown in the figure above, the HAAR cascade fails to detect a number of faces in the crowd. MTCNN and RetinaFace was able to detect more faces, with RetinaFace having the most number of faces detected. This however comes at the expense of model prediction speed.

3.2 Face Classification

3.2.1 Model Performance

The model performance of our face mask classifier is shown below.

	precision	recall	f1-score	support
Mask worn incorrectly	0.9920	0.9960	0.9940	500
With Mask	0.9876	0.9917	0.9897	483
No Mask	0.9980	0.9902	0.9941	509
accuracy			0.9926	1492
macro avg	0.9926	0.9926	0.9926	1492
weighted avg	0.9926	0.9926	0.9926	1492

Test Accuracy: 99.26%

Figure 5: Model Performance Summary

The model performs well in predicting any class type with precision, recall, F_1-score and accuracy all at approximately 99%. As a point of reference, the Proportional Chance Criterion (PCC) is determined to be 33% and 1.25xPCC at 42%. Our results is at 3 times better than the PCC making this a highly successful classifier.

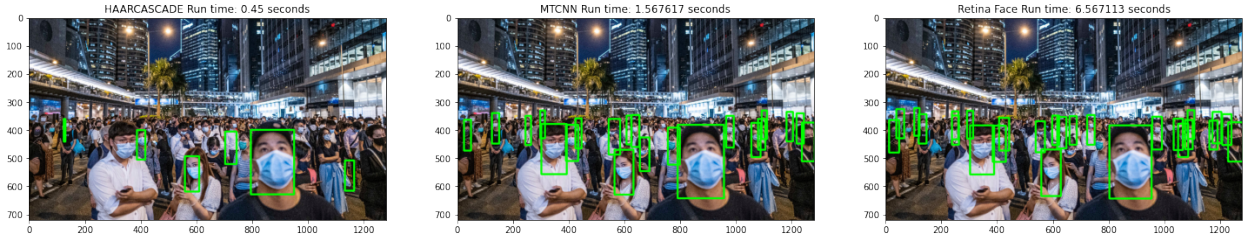

Figure 6: Comparing Face Detection Results for an Image with 10 persons

We also compare the results of our face mask classifier in comparison to other existing studies is also shown in Table 3. Due to differences in how model performance our reported, we only compared our model's F_1-score to existing works as this is the only common metric used. It can be observed that the model we have has an average F_1-score comparable to these studies at 99.26%. Additionally, these studies were only limited to predicting whether a person is wearing or not wearing a mask, where as in our model, three classes are being predicted. It should be noted however that the data sets used for the other studies are different. As such, we can not conclude that our model is better.

Work	**Average F_1 Score**
Chavda, et al	99.12%
Lipper, et al	99.00%
RetinaFaceMask	93.99%
Ours	**99.26%**

Table 3: Comparing our work with other similar studies

3.2.2 Runtime

Our classifier has an average run time of 54ms per face with a standard deviation of 70ms. Comparing to the final model Chavda, et al used (refer to Table 4), that our model perform 6 times faster and has less than half of network parameters.

Model	Average Inference Time	No. of Parameters (in millions)
NASNetMobile	295 ms	4.88
Ours	**53 ms**	**2.34**

Table 4: Face classification comparison table

3.2.3 Sample Classification Results

You can see below (Figure 7) that our model can classifier images with mask easily even with different kinds of mask. There are cases that our model misclassifies maybe because of the design of the mask as seen on the rightmost image. Figure 8, shows how our model easily classifies images without mask. Lastly, Figure 9, shows the variation of images that could be classified as incorrectly worn mask.

Figure 7: Sample images with mask

Figure 8: Sample images with no mask

Figure 9: Sample images with incorrectly worn mask

3.3 Combined Results

Combining the results of our face detection and face classifier, we were able to get a face mask detection system which could identify faces and classify if a particular face is wearing mask, not wearing mask or is not properly wearing mask. A sample result of our face mask detection system is shown.

Figure 10: WearMasknet Detection System Results

We also test the result of our face mask detection system on people wearing masks that do not look like mask. It was shown to still work in such cases:

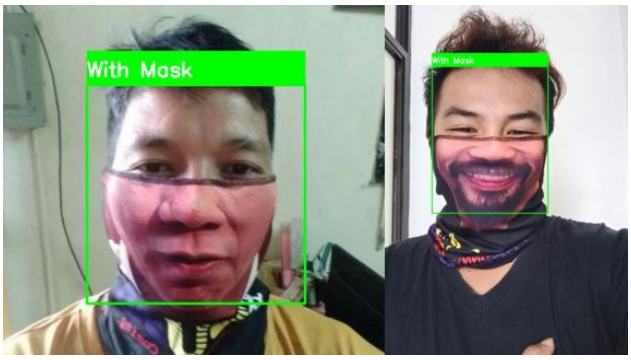

Figure 11: WearMasknet Detection System Results

4 Conclusion and Recommendations

We created **WearMaskNet** which doesn't just detect faces from an image but also classifies the faces as having a face mask, having no face mask or having a face mask but incorrectly worn. We utilized a multi-step approach in tackling this problem by using face detector algorithms such as Haar Casscade, MTCNN and RetinaFace to get the bounding boxes of the images and then used a classifier to classify the faces using a pre-trained model named MobileNetV2. We found out that our classification model architecture was significantly lighter than other architectures but we have comparable performances than other works.

Unfortunately, we weren't able to compare our model directly to the other solutions because of the lack of specifications and weights of the other work's models. Thus we recommend to implement other group's works and compare our solution to them with the same dataset.

We also found that the bottleneck for the speed of processing of our solution is largely based on the detector and not the classifier. Thus we recommend that we found other solutions that would make detecting of faces faster. In relation to this, we found that it is challenging for our detector to find faces in crowds and thus we recommend to explore face/head detection algorithms that is for crowds.

In the Philippines, face shields are required by the government as a protective measure against COVID-19. Thus, we recommend to add another classification to our face mask classifier algorithm for the one with face shields.

We recommend that we add more dataset for the face masks so that our classifier will be robust for face masks with different designs.

Lastly, we recommmend to optimize our solution for videos. Maybe it could be improved by utilizing algorithms that utilizes motion. We could also optimize the classification by just classifying faces only every other frame.

References

[1] Nundu Sabiti Sabin, Akintije Simba Calliope, Shirley Victoria Simpson, Hiroaki Arima, Hiromu Ito, Takayuki Nishimura, and Taro Yamamoto. Implications of human activities for (re) emerging infectious diseases, including covid-19. *Journal of physiological anthropology*, 39(1):1–12, 2020.

[2] Waradon Sungnak, Ni Huang, Christophe Bécavin, Marijn Berg, Rachel Queen, Monika Litvinukova, Carlos Talavera-López, Henrike Maatz, Daniel Reichart, Fotios Sampaziotis, et al. Sars-cov-2 entry factors are highly expressed in nasal epithelial cells together with innate immune genes. *Nature medicine*, 26(5):681–687, 2020.

[3] Yixuan J Hou, Kenichi Okuda, Caitlin E Edwards, David R Martinez, Takanori Asakura, Kenneth H Dinnon III, Takafumi Kato, Rhianna E Lee, Boyd L Yount, Teresa M Mascenik, et al. Sars-cov-2 reverse genetics reveals a variable infection gradient in the respiratory tract. *Cell*, 182(2):429–446, 2020.

[4] Adnane Cabani, Karim Hammoudi, Halim Benhabiles, and Mahmoud Melkemi. Maskedface-net – a dataset of correctly/incorrectly masked face images in the context of covid-19. *Smart Health*, 2020.

[5] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, volume 1, pages I–I. IEEE, 2001.

[6] Tanwir Khan. Computer vision — detecting objects using haar cascade classifier, 2019.

[7] Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, and Yu Qiao. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, 23(10):1499–1503, 2016.

[8] Jiankang Deng, Jia Guo, Yuxiang Zhou, Jinke Yu, Irene Kotsia, and Stefanos Zafeiriou. Retinaface: Single-stage dense face localisation in the wild, 2019.

[9] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.