Face Mask Detector

Ruofan Yao
ryao9@wisc.edu

Shijie Huang
shuang362@wisc.edu

Yanxu Guo
yguo249@wisc.edu

Abstract

Considered the outbreak of Covid-19 pandemic and the proven effectiveness of masks for protecting humans from viruses, more and more governments require their citizens to wear masks in public places. Thus, it is meaningful and necessary to check the situation of wearing a mask in public places and send notifications to people. Fortunately, the neural network models provide a cheap and effective way to automatically detect face masks. In this paper, we chose three models, MobileNet-v2, Resnet-18 and Inception-v3, to compare their performance at this task. We collected 11042 face images from the Internet which were with masks or without mask as our dataset. After several experiments, we found that MobileNet-v2 can achieve a high training precision of 90.43% with a faster training process by training the model with 35 epochs, a batch size of 128 and a learning rate of 0.0005. While the test accuracy is 63.36%, which was relatively low. The limitation of our experiment is that we failed to train our models with full dataset and large epochs due to the weakness of computation resources we have. Hence, it may get a better accuracy result if the dataset is larger and the computation resources are more abundant. Also, the implementation of a face detector would help to extract face images and improve detection performance.

1. Introduction

Due to the sudden outbreak of Covid-19 pandemic, face mask detection has become a popular research and application area. It has been confirmed that wearing masks in public areas is the most powerful and convenient way to prevent the virus from spreading by air and droplets. Therefore, people are required to wear a mask in public areas that have high population density, especially in stores, airports, and offices. However, people are not comfortable or unwilling to wear masks all the time and may take off masks when no one is supervising. It is impossible to hire staff to check everyone all the time, but by monitors it is possible. A good face mask detection system is able to recognize whether a person is wearing a mask or not instantly. For example, at the entrance of the airport, if a traveler is found without wearing a face mask, the system should immediately detect the target, send a report to the airport guards and remind the traveler.

By automatic mask detecting algorithms, systems can send alarms to people when the mask is off. This tool will largely reduce the labor cost, and the risk of staff getting infected during work time. Therefore, under current circumstances, face mask detection is a popular and powerful topic in deep learning studies, also with a huge application demand. Fortunately, with the rapid advancement of deep learning and convolutional neural networks (CNN), precise facial recognition and image classification have been well addressed. However, it is challenging to develop an efficient model because [1] "faces with masks have varied accommodations, various degrees of obstructions, and diversified mask types."

The goal of our project is to compare different deep learning models and find the best model to detect whether a person is wearing a mask or not. Since face mask detection is a treading study that is based on image cognition, it is a great way for us to learn and apply advanced deep learning methods to solve this public health problem. We would like to find a model with high accuracy, which can correctly identify if an individual is wearing a mask within a second.

The rest of the paper is organized as follows. The upcoming section 2 discusses some related articles and former studies about mask detection. Section 3 introduces the sources of our dataset and the processes of cleaning and cropping to them. Section 4 illustrates the architectures of different models, Mobilnet-v2, Resnet-18, and Inception-v3, that we are going to compare. Section 5 illustrates the process of our experiment and model training. And section 6 shows our results and discusses the problems we faced during the training.

2. Related Work

Boosting-based classification Since 2002, researchers have begun working on detecting face images. Viola-Jones Detector firstly made real-time face detection possible, which offered great progress in face detection technology[1]. However, it still had limitations such as the

orientation and brightness of the face. Thus, it was not sophisticated with detecting faces in dull and dim light. Motivated by Viola-Jones Detector, a multiview face mask detector using decision trees algorithms was developed.

Deformable Part Model-based classification The DPM-based face mask detector using a random forest tree model can reach high accuracy[1]. It can precisely detect face structure and facial features, while the cost of DPM-based computation can be relatively high.

Convolutional Neural Network-based classification In CNN, the face detector model can directly gain information and learn from the raw data and there are many powerful deep learning algorithms that can train good models. As technology advances, CNN is especially good at image classification, object detection, face recognition, and image synthesis. It is widely used because of its relatively low computation cost and remarkable spatial feature extraction capability.

In an article talking about OpenCV, Keras / TensorFlow[2], and deep learning, Rosebroc introduces potential ways of training a COVID-19 face mask detector with Keras/TensorFlow. There are two steps of training a face mask detector. The first step is to artificially build a dataset with masks on people's faces, and the second step is to train the model using the dataset to detect and classify whether people in the photos are wearing face masks or not. The face mask dataset is created by detecting the facial landmark of the original photo with no face masks and artificially adding face masks to those photos to create a category with face masks. Then, it needs to gather unused photos to form the no face mask category. After data processing, MobilenetV2 is implemented with fine-tuning. During the fine-tuning process, the base layer is frozen and the head layer will be turned. In the end, it introduced the potential implementation of adding the photo detector in a real-time video stream with OpenCV.

In another research paper[6], the researchers used YOLO algorithm and Faster R-CNN algorithm to train the face mask detection models and compared the outcomes of these two architectures. They found that Faster R-CNN had a higher accuracy but YOLO can train much faster. Considering the trade-off between speed and precision, YOLO was preferred because it performs single-shot detection and has a much higher frame rate[6], which can better fit in real-world applications.

3. Dataset Used

3.1. Data Overview

The dataset used in the project contains 11042 images separated into two categories, with labels with-mask and without-mask. The original data was gathered from a variety of sources, accommodating different conditions. The ready-to-use images were coming from the Real-Time-Medical-Mask-Detection on github[4]. One main source of the data was 1376 images created by Bhandary P.[2] by taking standard images of faces and applied facial landmarks. An artificial method was implemented to add masks to people without masks. During the process, the images would not be overused. Another source of images is Kaggle. It includes 678 raw pictures of people wearing medical masks. The full dataset was divided into two parts, people wearing masks and without masks. In order to make the test more accurate and include more possibilities, each of the group members searched 40 images without masks and 40 images with masks from different searching engines, Baidu, Bing, and Google, respectively. After the data cleaning process, the team separates the images into training, testing, and validation datasets according to the 7:2:1 ratio. It ends up having 7688 images in the training dataset, 1184 images in the testing dataset, and 2288 images in the validation dataset. To separate images with masks and without masks in the same folder, all images with masks are added "1000" at the head of their filenames, and all images with no masks are added "2000" at the head of their filenames. And these filenames are recorded in train, valid, test CSV files with their class label, 1 means with mask and 0 means with no mask.

Figure 1. Figure expected to be labeled as 1.

Figure 2. Figure expected to be labeled as 0.

Figure 3. Displacement of images with masks after transformation.

Figure 4. Displacement of images without masks after transformation.

3.2. Dataset Cleaning

First, we tried to remove duplicate images, which may come from different sources but replicated incidentally. We transferred all images into their unique hash code to compare whether they are the same. Then, during the runtime of code, we found several images are broken and cannot be opened by python Image class. So we read all images at first and print the filenames of broken images, then delete these filenames in both CSV files and image folders.

3.3. Dataset Cropping

In the beginning, the data were continuously trained with high loss and low validation accuracy. To figure out the problem, we displayed the first 48 images of the train images and found that some faces of people were hard to detect. Some of the people were large and located in the middle of the images while others were tiny and hard to identify. Considering this issue, we cropped the part of people's faces from photos with tiny faces to increase the accuracy of detection. The program inbac was implemented in the terminal and allowed the author to crop photos with ease by tapping the z command[4]. The training dataset was then re-established with most people's faces displayed in the middle of the image.

4. Proposed Models

4.1. Transfer learning

Transfer learning is a powerful technique to speed up the training process and improve model accuracy. Our model will learn and update the weights based on the weights and biases of a pre-trained model, so we do not need to train the model from scratch. A pre-trained model has been trained on a dataset by others before and contains learned features such as layers or weights that are applicable to our dataset. When the dataset is not big enough, transfer learning will help us save time and also improve the performance of the models since the pre-trained model, such as ImageNet, has extracted some useful features and built a good foundation for training.

4.2. Mobilenet-v2

Mobile Network (MobileNet) architecture is proposed to solve the situation when computational resources are very limited, such as detecting objects by mobile or embedded devices. This model uses depthwise convolution to extract features and channel-wise convolutions to adjust channel numbers, so the computational cost of MobileNet is much lower than the networks using standard convolutions[6]. The structure of Mobilenet-v2 comes up with 1 feature block, 17 inverted residual blocks, 1 activation block and 1 classification block. Inside of them, they have convolutional layers, max and average pooling layers, dropout layers, ReLu activations and fully-connected layers. The convolutional layer uses a sliding window mechanism to extract features from an image, and then by a pooling mechanism, it reduces the size of input data to speed up later computations. The dropout layer is to randomly drop out biased neurons from the model to avoid overfitting problems. The non-linear layer and fully-connected layer is to classify images into multi-class or binary classifications.

The innovation of MobileNet-V2 is to create an interesting residual block, which is opposite to the traditional residual block. The residual block has a wide-narrow-wide structure with many channels, and it uses 1x1 convolution to compress and widen. But the inverted residual block has an opposite structure, which is narrow-wide-narrow. It uses a 1x1 convolution to widen, then uses a 3x3 depthwise convolution (which greatly reduces the number of parameters), then a 1x1 convolution is implemented to reduce the num-

ber of channels so input and output can be added.

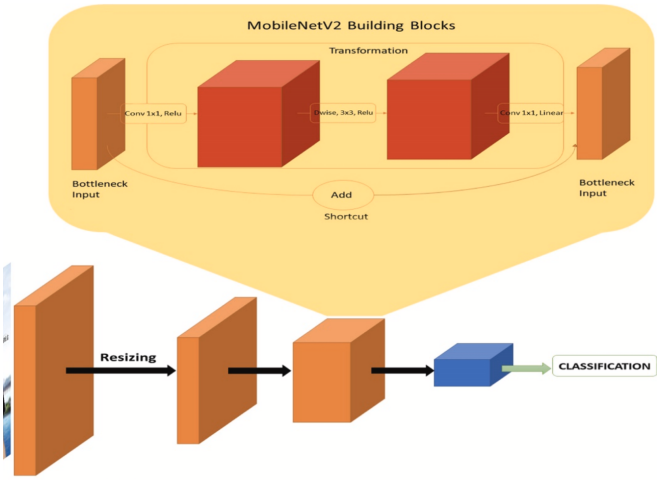

Figure 5. The architecture of MobileNetV2. Image source:[1]

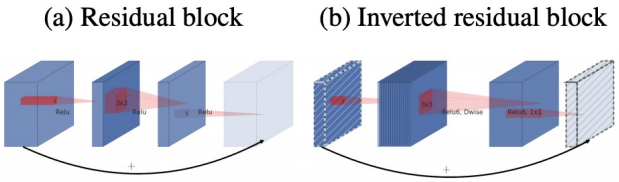

Figure 6. Comparison between residual and inverted residual. Image source: https://arxiv.org/pdf/1801.04381.pdf

4.3. Resnet-18

Residual neural network (ResNet) allows researchers to implement very deep architectures by utilizing skip connections. Since it has so many layers, skip connections can help it reduce unnecessary layers and also help avoid the problem of gradient descent. In each layer, the model includes batch normalization and nonlinear functions like ReLU. ResNet-18 is a CNN architecture that contains 18 layers. The pre-trained version of the network trained on more than a million images is available from the ImageNet database. The pre-trained network can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. As a result, the network has learned rich feature representations for a wide range of images.

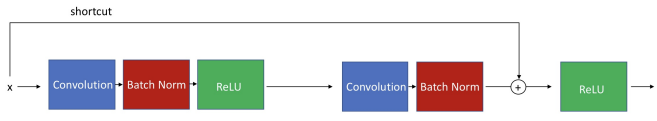

Figure 7. Skip connection. Image source:https://canvas.wisc.edu/courses/244897/files/folder/L14

4.4. Inception-v3

The name of Inception came from a popular quote in a film saying "we need to go deeper". In order to maximize the learning outcome, Inceptionv3 adds more convolutions in a layer, which results in a large number of parameters. The model is both wider and deeper. To make the training easier, the architecture generally reduces the sizes of images, and the precision is improved. Developed from GoogleNet, Inception is remarkably good at image analysis and object classification. It can reach very high precision if the computational resources are available.

5. Experiments

5.1. Experiment Process

The experiment was conducted based on a small dataset consisting of 80 photos including 40 with masks and 40 without masks are randomly selected to form the cleaned training, testing, and validation dataset. The graphs and a CSV file including their file name and labels were loaded into the data loader. Then, the photos experienced a transformation. The images were resized to 70 and center cropping to [70, 70]. The photos would be trained by the official model algorism online and an add-on output layer to classify the photos into two categories. As for hyperparameters, we used a learning rate of 0.0005, a batch size of 128 and 35 epochs to train.

Backpropagation and a loss function were implemented in the helper file (course material). Backpropagation is an algorithm for the supervised learning of artificial neural networks using gradient descent. Given an artificial neural network and an error function, the method calculates the gradient of the error function with respect to the neural network's weights. In addition, the cross-entropy was used as the loss function. We compared both the cross-entropy loss function and BCE loss function and it turns out that the cross-entropy loss function resulted in higher accuracy. Cross-entropy loss, or log loss, measures the performance of a classification model whose output is a probability value between 0 and 1. Cross-entropy loss increases as the predicted probability diverge from the actual label.

Then, to optimize the model, we chose Adaptive Moment Estimation(Adam) as our optimizer. The Adam optimizer is a combination of the Adaptive Gradient Algorism(AdaGrad), which is used to improve performance on problems with sparse gradients, and Root Mean Square Propagation(RMSProp), which is used to adapt the average of recent magnitudes of the gradients for the weight (e.g. how quickly it is changing) and reduce noise. In Adam, the adaptive learning rates will decrease if the gradient changes its direction, and will increase if the gradient stays consistent.

$$\mathcal{L}_i = -\sum_k y_k \log(\sigma(l_i)) + (1 - y_k) \log(1 - \sigma(l_i))$$

Figure 8. Cross-entropy loss for classification. Image source:https://subscription.packtpub.com

$$For\ each\ Parameter\ w^j$$
$$(j\ subscript\ dropped\ for\ clarity)$$
$$\nu_t = \beta_1 * \nu_{t-1} - (1 - \beta_1) * g_t$$
$$s_t = \beta_2 * s_{t-1} - (1 - \beta_2) * g_t^2$$
$$\Delta\omega_t = -\eta \frac{\nu_t}{\sqrt{s_t + \epsilon}} * g_t$$
$$\omega_{t+1} = \omega_t + \Delta\omega_t$$

$\eta : Initial\ Learning\ rate$
$g_t : Gradient\ at\ time\ t\ along\ \omega^j$
$\nu_t : Exponential\ Average\ of\ gradients\ along\ \omega_j$
$s_t : Exponential\ Average\ of\ squares\ of\ gradients\ along\ \omega_j$
$\beta_1, \beta_2 : Hyperparameters$

Figure 9. ADAM. Image source: https://blog.paperspace.com/intro-to-optimization-momentum-rmsprop-adam

Method	Accuracy	Training Time
Mobilenet-v2	82.5 %	2.20 min
Resnet-18	85.0%	2.82 min

Table 1. Comparison of Mobilenet-v2 and Resnet-18

5.2. Comparison

Mobilenet-v2, Resnet-18, and Inception-v3 models were implemented in the experiment to determine which model had the fastest speed and the highest testing accuracy.

According to Table 1, Resnet-18 had a higher precision but a longer training time. Mobilenet-v2 had a little lower precision and a shorter training time. Inception-v3 is slow in the training process and has a lower test accuracy since it is a deeper and more complicated architecture. The time it took to run one epoch was 50 times longer than the other two architectures'. Because of the limitation of the hardware, the team put the training time as the first consideration so that it is able to make adjustments in time. Modilenet-v2 needs fewer computational resources while reaching a relatively high accuracy, so it is the main training model.

5.3. Software and Hardware

All the code was written by PyTorch and ran in python3. We had tried to run the code in Google Colab and took advantage of GPU, but google drive, where we wanted to upload our dataset, was not cooperative. So we changed to run our code on laptops. In addition, when doing data cropping, we utilized a program called "inbac" which was provided in GitHub. This program effectively helped us crop the images and kept only human's face remaining in the dataset.

Hardware was a challenge met by the group during the projects. All the team members use MacBooks, which are not as powerful as desktop computers, and GPUs are not accessible. It took an average of 5 hours to conduct a full training process, then made adjustments according to results. By having examinations during the training process, the team members detected the trend of loss fluctuations at each epoch before running the whole dataset. To increase the performance speed, a method thought by the group is to take advantage of the GPU through the Google Colab. However, the uploading issue induced the missing value of the image that did not allow the code to run through the beginning. Thus, the hardware project increased the difficulty of the project.

6. Results and Discussion

6.1. Training full dataset

We end up achieving a test accuracy of 63.36%. The training accuracy and validation accuracy are 90.43% and 84.40% respectively. From the displacement of the testing result, It seems that our learning model could only identify most of the masks with the blue color but not the white one. For future improvement, we could adjust the brightness of each photo to make the white mask not blend with the background.

In the former experiment of model building, we noticed that the loss entropy should be used as the loss function and Mobilenet-v2 was the right model to choose. Then, we planned to implement the model to a large dataset. It turned out to have a low learning accuracy and low validation accuracy at 50% level. Several methods have been tried to improve the testing accuracy.

Firstly, by displaying the photo after resizing and cropping the dataset, we found that in the original dataset, people are displayed nonuniformly in the dataset, with some were very small while others were very large. Then, the author artificially cropped the dataset and replaced the original dataset with the new one. Because there were originally 11042 images in the dataset, it was hard to crop all of them. By quickly moving through figures to figures, the author cropped most but may not all the photos. The improvement was not significant for the large dataset but was significant for our small dataset.

Finding the training data set to be volatile, the author thought about another way of improvement–only using the training and testing dataset. The original validation dataset

was directly changed to be the training dataset with 2288 figures and the original testing dataset was divided into 433 testing figures and 751 validation figures. We then train the model with our new dataset. Noticing that the epoch achieved the highest training and validation rate at the 35th epoch, we changed the training epoch from 50 to 35. By doing that, the testing accuracy improved from 50% to 63.36%.

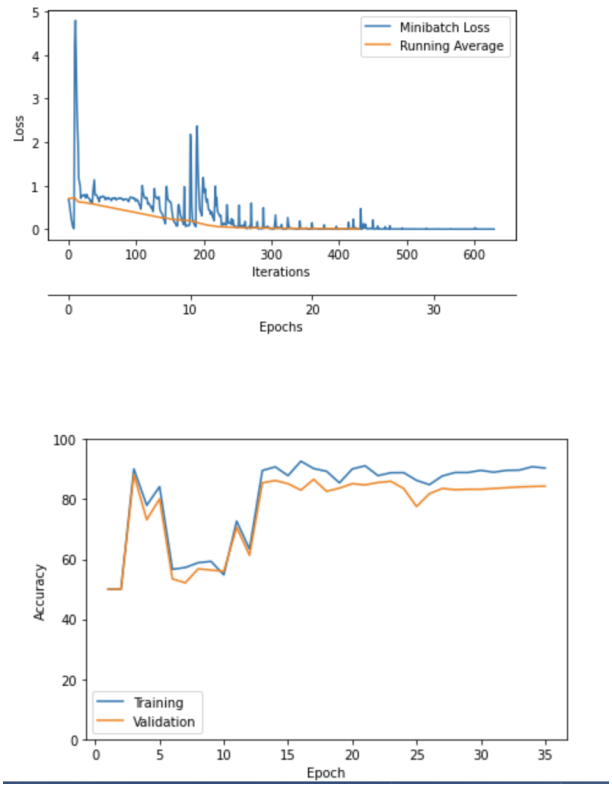

Figure 10. Training outcome of Mobilenet-v2

6.2. Discussion

The difficulties we met during the process is that the mask detecting method is relatively weak, which may cause the accuracy rate to keep continuously low and the loss never decreased. The classification of the images of wearing masks and without wearing is based on artificial class labels of 0s and 1s instead of the detection based on facial landmarks. Since the model worked pretty well when using smaller data sets including only 80 images, the uncertainty of the large dataset is brought into consideration. The similarities and the accessibility of the main features are crucial for the detecting process. Thus, for the next time, it is crucial to increase and check the quality of the dataset at the beginning. For future study, to increase the accuracy of the model, the detection based on fiscal landmarks could be implemented to directly target the human's face in each photo.

7. Conclusions

Due to the limitation of computational resources, Mobilenet was the optimal training model for us because it was fast and precise as well. While the small dataset preformed well with 90 percent accuracy, the full dataset had a hard time to train and could hardly exceed the accuracy rate of 70 percent. In order to overcome this obstacle, we edited our model and tested for several times. For future improvements, if we have enough computational resources and time, we could firstly develop a face detector, which can detect human faces directly from different kinds of photos under various conditions, such as small figures, far vision, or dark light. Also, we would make our dataset more organized so the model can easily read and run through it. Moreover, besides static images, we hope to implement a face mask detector in real-time video streams. When a person passes by, the detection system can automatically focus on the person's face and check whether the individual is wearing a mask properly. If this technique becomes sophisticated, it can be a good helper to prevent the virus from spreading.

8. Acknowledgements

We want to Thank professor Sebastian Raschka and TA Zhongjie Yu for the help during the project. The cropping method Inbac helps to increase the testing accuracy for the small dataset, and the easy-to-use feature of it would be helpful for future projects whenever we want to train a specific part of the photo. In addition, the helper files and the class materials provide us with the insight of how to perform and what to do to improve efficiency.

9. Contributions

During the preparation and planning process, the authors communicate weekly to discuss methods for implementing the model and shared information. In the beginning, each of the authors read 5 relevant articles which are related to mask detection to form a basic idea in mind. In the process of building the dataset, the authors obtained a large dataset from GitHub, then each of the authors found 40 photos with masks and 40 photos without masks in different search engines including Baidu, Bing, and google to make the dataset more concrete. Ruofan Yao is mainly responsible for building the architecture and writing the report paper. Shijie Huang is mainly responsible for organizing code and developing the model. Yanxu Guo is mainly responsible for cleaning the data, classifying the photos into CSV files, and adjusting the size of the images, as well as running experimentation of the full dataset.

References

[1] P. Nagrath, R. Jain, A. Madan, R. Arora, P. Kataria, and J. Hemanth. Ssdmnv2: A real time dnn-based face mask detection system using single shot multibox detector and mobilenetv2. *Sustainable Cities and Society*, 66:102692, 2021.

[2] Rosebrock,A., COVID-19: Face Mask Detector with OpenCV, Keras/TensorFlow, and Deep Learning (May 2020) https://www.pyimagesearch.com/2020/05/04/covid-19-face-mask-detector-with-opencv-keras-tensorflow-and-deep-learning/

[3] Data source:https://github.com/TheSSJ2612/Real-Time-Medical-Mask-Detection/releases/download/v0.1/Dataset.zip

[4] Data source:https://github.com/TheSSJ2612/Real-Time-Medical-Mask-Detection/

[5] https://github.com/weclaw1/inbac

[6] Singh, S., Ahuja, U., Kumar, M. et al. Face mask detection using YOLOv3 and faster R-CNN models: COVID-19 environment. Multimed Tools Appl (2021). https://doi.org/10.1007/s11042-021-10711-8

[7] Link to our code: https://github.com/YanxuGuo/453-Facemask-Detection