Face Mask Detection

Yang Liu (yl8240), Junzhou Liu (jl1222), Yiran Ma (ym2360),
CV Final Project link

Abstract

At the end of 2019, the sudden outbreak of coronavirus quickly spread rapidly to most countries around the world. The covid seriously threatens people's life and hinders the development of the economy. During the outbreak of the covid-19, masks play a crucial role in protecting people from diseases. With masks on, the spread of the virus can be interrupted though sometimes we cannot always keep the safe social distance. In this case, it is useful to develop techniques to enforce people wearing masks in the public if necessary.

1 Introduction

In this project, We we explore some existed detection modules and want to eventually develop modules capable of detecting whether wearing masks on given pictures.

1. Familiarize with Face Mask Detection dataset and learn how to analyze XML inputs and implement normalization.
2. Familiarize with existing detection neural network like VGG, Faster R-CNN.
3. Build up a baseline using current model on Face Mask Detection dataset and compare different models' performances.
4. Propose and implement new methods to improve the performance of current models.

Section two will explain the data analyze and preprocess and the structures of two modules. Sections three will discuss the results and improvements of the modules.

2 Methodology

2.1 Data Processing

Before detecting masks, we need to get familiar with the datasets and filter out some improper images. After that, in order to make it easier for the modules to read the inputs, standard normalization will be performed.

2.1.1 Image Cleaning

We download the datasets from Kaggle, which contains 853 pictures. Annotations can be analyzed from XML files. In each XML file, "object" represents for one people.In this part, we keep pictures that contain less than ten people. Also, we delete pictures the size of which are less than 40 x 40 pixels.

2.1.2 Image Normalization

After cleaning the data, we use some algorithms to normalize the data. Following are the methods used during normalization:

- Geometric Augmentation [1] :
  random_flip_left_right
  random_flip_up_down
- Intensity Augmentation [2] :
  random_brightness
  random_contrast
  random_saturation

There are some samples after processing the images(Figure 1).

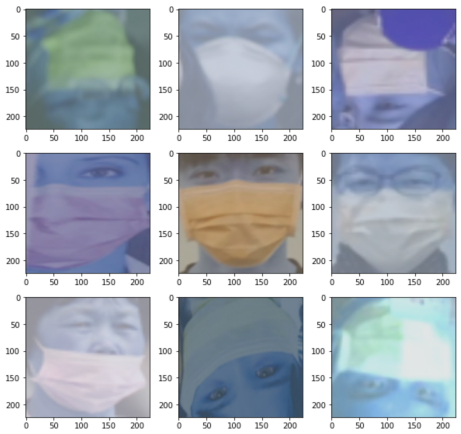

Figure 1: Samples after processing

2.2 Model

2.2.1 VGG16

VGG16 is one of the most popular and clean architecture convolutional neural networks. It was originally published from a paper called 'Very Deep Convolutional Networks for Large Scale Image Recognition' [3] in 2014. It is essentially based on just using three by three convolutions. It has several advantages such as the architecture is quite simple in terms of building layers, 3*3 convolutions, same padding and 2*2 max pooling. The disadvantages could be taking quite a long time to train the model since it has an enormous amount of parameters and numerous layers [4].

1. VGG has 16 layers with weights. In VGG16 there are 13 convolutional layer
2. VGG16 takes input tensor size as 224, 244 with 3 RGB channel
3. The three FC (Fully Connected) Layers follow a stack of convolutional layers
   - The first two has 4096 channels each
   - The 3rd performs 100-way ILSVRC classification about 1000 channels.
4. The final layer is the soft-max layer [5]

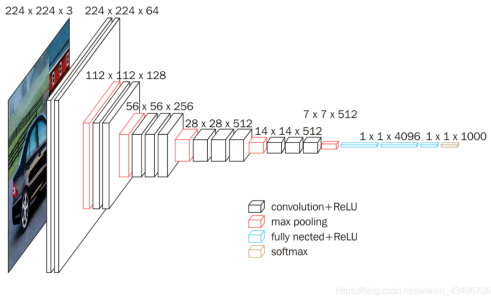

Figure 2: VGG16 Model

2.2.2 Faster-RCNN

RCNN is attempting to overcome the challenge of object detection. The distinction between an object detection algorithm and a classification algorithm is that detection techniques need us to construct a box around the object in the picture that we discovered. Instead of the output of a standard CNN model, the output of RCNN varies with the object and position of the object. The following graphic can help us understand better [6].

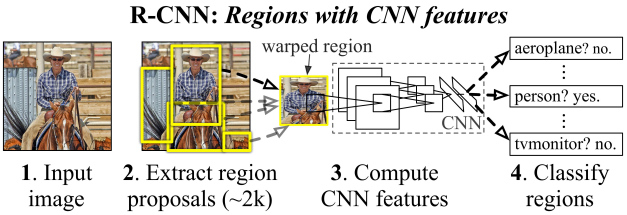

Figure 3: Better Understanding of RCNN

However, RCNN has some problems. It will take a long time to train the network. The selective search algorithm is a fixed algorithm. As a result, no learning occurs at that point which will result in the creation of bad candidate region proposals. Therefore, we consider to select Faster RCNN as our model compared with RCNN and Fast RCNN accocrding to the relevant experimental data results. A better understanding can be illustrated in the following image

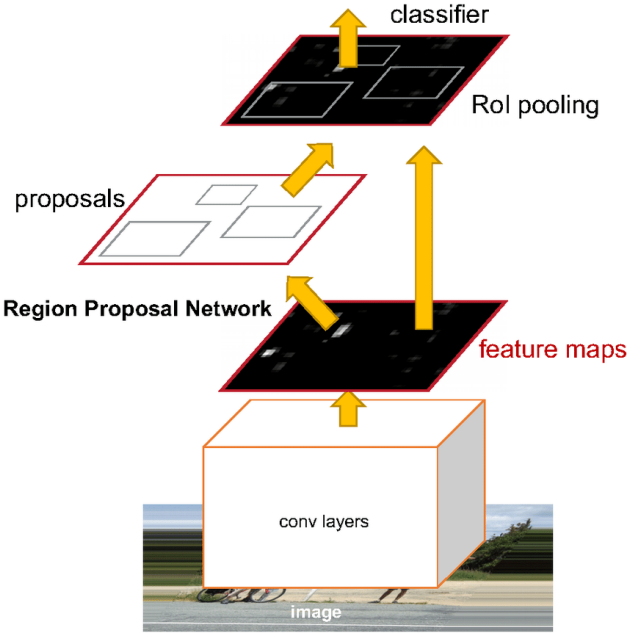

Figure 4: Faster-RCNN

Both R-CNN and Fast R-CNN use selective search to find region proposals. Selective search is a slow and time-consuming operation that degrades network performance. As a result, Shaoqing Ren [7] proposed an object identification algorithm that replaces the selective search strategy and allows the network to learn the region proposals. Similarly to Fast R-CNN, the image is fed into a convolutional network, which outputs a convolutional feature map. A separate network is used to predict the region proposals rather than using a selective search algorithm on the feature map to identify the region proposals. After reshaping the predicted region proposals with a RoI pooling layer, the image is classified within the proposed region and the offset values for the bounding boxes are predicted.

2.2.3 VGG16 Model Parameter

To improve the training and testing results, we tuned the input parameters of the dataset. The total parameters of our model are 134,289,731.

Our model forced on having 13 convolution layers that each each nodes that varies in 64, 128, 256, 512 in a 3*3 filter with stride 1 and always use the same padding. We also applied batch normalization to standardize the inputs to the layer for each mini-batch, stabilize our model if applying a high learning rate and dramatically reduce the number of training epochs required to train deep networks.. In between these convolution layers, maxpool layer was implemented in a 2*2 filter with stride 2. It follows the arrangement of convolution and max pool layers consistently throughout the whole structure. In the end it has a fully connected layer with 4096 nodes. Dropout with rate 0.5 is applied inside the fully connected layer, which offers a very computationally cheap and remarkably effective regularization method to reduce overfitting and improve generalization error in deep neural networks of all kinds.

2.2.4 Faster-RCNN Model Parameter

We distributed the dataset into a separate model, with the training dataset being around 683 and the testing dataset being around 170. The model parameters in detail can be found in the github link: Faster-RCNN.ipynb. Instead of creating our own model, we used the Pytorch model.

2.3 Accuracy and Loss

Following the construction of the model, we must compute the loss and accuracy for validation. For this model, we used cross entropy loss, which assesses the performance of a classification model whose output is a probability value between 0 and 1. As the anticipated probability diverges from the actual label, cross-entropy loss grows. The log loss of a perfect model would be zero. We later added L2 norm loss in the tensorflow environment to the training and testing loss. The sum of all squared discrepancies between the true and anticipated values is the L2 norm loss. All of the results of these loss and accuracy computations would be displayed in the tensorboard graph, with examples provided in the next section.

3 Experiment Results

3.1 The VGG16 Model's Results

Due to the insufficient number of images in the dataset, we set the batch size of VGG16 to 16 for training. After training for 100 epochs with a learning rate of 0.01, our training accuracy is around 90 percent and testing accuracy is 97.5 percent. The following image is the training, validation and testing graph for loss and accuracy.

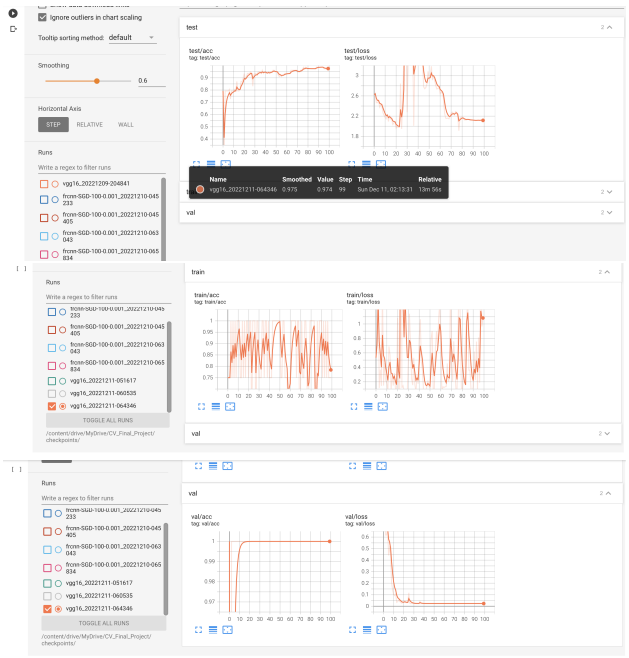

Figure 5: VGG16-accuracy

3.2 The Faster-RCNN Model's Results

Prior to optimization, the recognition still had some discrepancy between the real mask and other comparable picture pixels.

3.2.1 Optimization model

1. In the Faster-RCNN model, we utilize Stochastic Gradient Descent (SGD) to and the learning rate 0.001.

$$optimizer = torch.optim.SGD(params, lr = 0.001) \quad (1)$$

Figure 6: model with SGD optimizer

As can be seen from the prediction results in the figure, some people's ears were wrongly identified as masks, and some people's facial masks were identified multiple times, which caused confusion. After Optimization, we have improved the recognition accuracy.

2. The second is an implementation of the ADAM optimizer, which combines two separate gradi-

ent descent algorithms. Taking into consideration the exponential weighted average frequently speeds up the gradient descent. We set the learning rate to 0.0001 in our results. In this example, we were able to successfully optimize the mis-recognition.

$$optimizer = torch.optim.Adam(params, lr = 0.0001) \quad (2)$$

Figure 7: model with ADAM optimizer

The accuracy of the prediction effect optimized by adam optimizer is much more accurate than that of the former, which can accurately identify whether everyone wears a mask or not and even whether the mask is worn in the right way.

There are a few reasons why the ADAM optimizer may perform better than the SGD (stochastic gradient descent) optimizer:

- First, the ADAM optimizer uses adaptive learning rates, which means that it automatically adjusts the learning rate for each parameter in the model. This can help the model converge faster and avoid getting stuck in local optima. In contrast, the SGD optimizer uses a fixed learning rate, which can make it more difficult to find the global optimum.

- Second, the ADAM optimizer also incorporates momentum, which helps the model continue moving in the same direction even if the gradient changes. This can help the model avoid oscillating around the optimum and converge more quickly. In contrast, the SGD optimizer does not use momentum and may take longer to converge.

- Finally, the ADAM optimizer also uses an exponentially weighted average of the gradients, which can help the model to smooth out the noise in the gradients and avoid overfitting. In contrast, the SGD optimizer uses the gradients from the most recent training batch, which can make it more susceptible to noise and overfitting.

Due to the minimal number of pictures in the dataset, we utilized batch size 10 for our Faster-RCNN model. Then we selected to train 100 epochs with a learning rate of 0.0001. Finally, we achieve a 0.47 mAP [8] score for both validation and testing. The mAP score is calculated by combining the accuracy of both box placement and mask categorization. The validation and testing graph for accuracy and the train graph for loss are shown in the Figure.

Figure 8: The mAP score for validation and test model with ADAM optimizer

4 Future work

4.1 Model selection

Normally, we use CNN modules to detect objects. However in this project, the boundaries of masks have been assigned in the dataset. Thus, instead of using modules like CNN, we decide to use classifier model(VGG16 specifically) to achieve our goal. In the meanwhile, the results of Faster-RCNN will be used as comparison with that of VGG16 [9].

4.2 Dataset issue

There are about 1000 pictures used in this project. In order to improve the performance of the modules, we should train and test on more images. However, it is not an easy thing to expand the dataset [10]. At first, though there are a lot of pictures with or without masks, we need to transform the format of the locations of masks into XML objects. Also, there is no defined rules toward the label "Mask worn incorrectly". So confusing data may result in confusing results. Overall, we should make sure the qualities of the dataset and expand more images into the modules.

4.3 Hardware Limitations

Due to the fact that our dataset was not enormous, the parameter for our model was humongous. The training process in this case was quite long. Even though we used Google Colab Pro+ to complete the project, the basic feature was insufficient for training the model. Furthermore, we only resized our image to a reasonable size. As a result, the checkpoint that we saved for each epoch was around 500 MB, which took up a lot of space. And even in VGG16 model, if I want to train 300 epoches, there will appear that there is no enough disk space for us to save some relevant checkpoints information. Therefore, I think there is still room for further optimization of the model and related hyperparameters.

5 Conclusion

Mask detection can be used in various conditions. In this project, the highest accuracy of the VGG16 module is 97.40% and the mAP of the Faster-RCNN module is 0.47, which perform well on detecting masks. What's more, we can also improve the performance of the modules by choosing fitter modules and expanding dataset etc..

References

[1] M. Elgendi, M. U. Nasir, Q. Tang, D. Smith, J.-P. Grenier, C. Batte, B. Spieler, W. D. Leslie, C. Menon, R. R. Fletcher *et al.*, "The effectiveness of image augmentation in deep learning networks for detecting covid-19: A geometric transformation perspective," *Frontiers in Medicine*, vol. 8, p. 629134, 2021.

[2] L. S. Hesse, G. Kuling, M. Veta, and A. L. Martel, "Intensity augmentation to improve generalizability of breast segmentation across different mri scan protocols," *IEEE Transactions on Biomedical Engineering*, vol. 68, no. 3, pp. 759–770, 2020.

[3] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[4] A. Sengupta, Y. Ye, R. Wang, C. Liu, and K. Roy, "Going deeper in spiking neural networks: Vgg and residual architectures," *Frontiers in neuroscience*, vol. 13, p. 95, 2019.

[5] Q. Guan, Y. Wang, B. Ping, D. Li, J. Du, Y. Qin, H. Lu, X. Wan, and J. Xiang, "Deep convolutional neural network vgg-16 model for differential diagnosing of papillary thyroid carcinomas in cytological images: a pilot study," *Journal of Cancer*, vol. 10, no. 20, p. 4876, 2019.

[6] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *arXiv e-prints*, p. arXiv:1311.2524, Nov. 2013.

[7] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *arXiv e-prints*, p. arXiv:1506.01497, Jun. 2015.

[8] R. M. Gottsdanker, "The accuracy of prediction motion." *Journal of Experimental Psychology*, vol. 43, no. 1, p. 26, 1952.

[9] D. Theckedath and R. Sedamkar, "Detecting affect states using vgg16, resnet50 and se-resnet50 networks," *SN Computer Science*, vol. 1, no. 2, pp. 1–7, 2020.

[10] M. Jiang, X. Fan, and H. Yan, "Retinamask: A face mask detector," *arXiv preprint arXiv:2005.03950*, 2020.