Masked Face Recognition Based on Convolutional Neural Network

Zhang Yifei* Sang Rui†

June 7, 2020

*Student ID: 5180xxxxxxxx
†Student ID: 5180xxxxxxxx

Abstract

This article is for project 2 of the Course of Machine Learning, School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University. In this paper, we will show that how to implement a masked face classification task by using convolutional neural network(CNN). Also, the advantages of different CNNs are illustrated.

Please note that the codes in this article are based on Python 3.7, TensorFlow 2.0.

Keywords: CNN, classification, masked face recognition

1 Introduction

1.1 Classification For Masked Face

2020 is destined to be an historical year. Since the outbreak of the COVID-19 at the beginning of the year, people all over the world begin to wear masks to protect themselves from the virus. Wearing a mask is an effective way to prevent the spread of the virus. Particularly, in areas where the epidemic is developing rapidly, a lot of data and pictures of people wearing masks emerge as people are wearing masks everywhere they travel. These massive data can be generated and open to public, turning it out to be valuable resources for AI based face recognition and preparing for similar public health emergencies that may occur in the future.

In this article, we will implement a simple model using the convolutional neural network, which performs the classification job of whether a person is wearing a mask.

1.2 Convolutional Neural Network

Convolutional neural network (CNN) is one of the representative algorithms of deep learning. It imitates the biological visual perception mechanism. CNN is mainly composed of three basic layers: convolution layer, pooling layer and fully connected layer (dense layer). Among these, the convolution layer and the fully connected layer have parameters, and the parameter update is achieved through back propagation.

1.2.1 Convolution Layer

The convolution layer uses a convolution kernel (a series of filters) to extract some features from the original image (Figure 1 is the schematic diagram). For example, the vertical Sobel filter (shown in Table 1) is convolved with the original image to detect the vertical edges of the original image (shown in Figure 2). The result obtained by convolution from the original image is called ”feature map”.

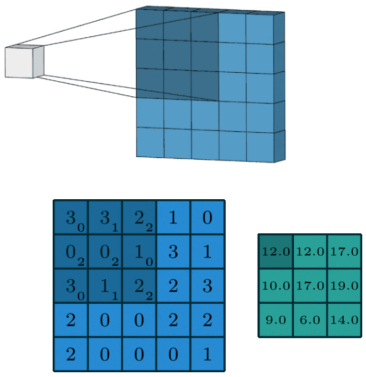


Figure 1: Convolution operation

-1	0	1
-2	0	2
-1	0	1

Table 1: Vertical Sobel filter

Figure 2: Original image and the feature map after convolution with vertical Sobel filter[1]

1.2.2 Pooling Layer

Adjacent pixels in the image tend to have similar values, this means that most of the information contained in the output of the convolution layer is redundant. That's why we have pooling layers in CNN.

The pooling operation is shown in Figure 3. The pooling layer is actually down-sampling, which will continuously reduce the size of data, so the number of parameters and calculation amount will also decrease, which can also controls overfitting[1]. At the same time, useful information is retained.

The most commonly used pooling methods are average-pooling and max-pooling. Generally speaking, max-pooling performs better at saving edge information, and average-pooling is good at saving overall background information.

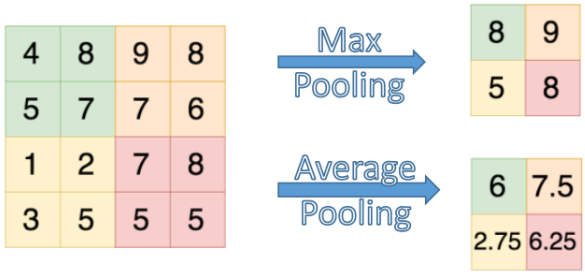


Figure 3: Pooling operation

[1]Figure 1 and 3 are taken from Professor Chen's slides

1.2.3 Fully Connected Layer

Fully connected layers (FC, also called ”dense layer”) play the role of ”classifier” in the entire CNN. The convolution layer and the pooling layer map the original image data to the feature space, and the fully connected layer functions map the features to the label space through certain weights and bias.

However, there are generally more parameters for full connection layers, which can account for most of the entire network parameters.

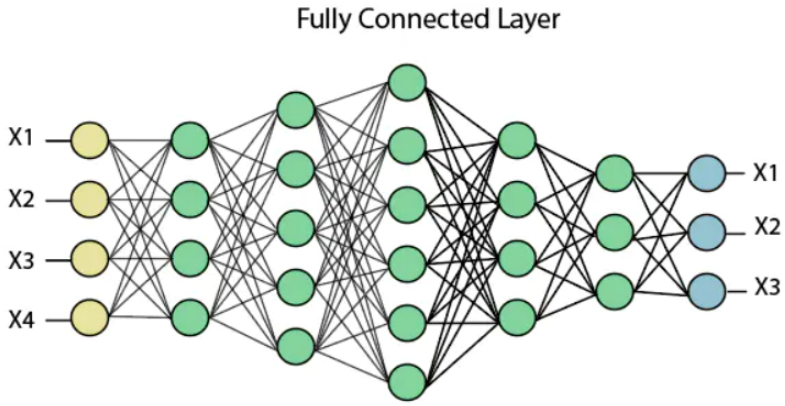


Figure 4: Fully connected layer

1.3 TensorFlow

In the project, we choose TensorFlow to build the model. TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in machine learning and developers easily build and deploy machine learning powered applications[2].

Tensorflow has two function interfaces: TensorFlow and Keras. We use Keras interface this time.

2 Algorithm and Code Explanation

2.1 Dataset of Real World Masked Face

2.1.1 Basic Features of The Dataset

The dataset[2] of real world masked face was kindly offered by Professor Chen. The dataset contains 90929 unmasked face and 2729 masked face. Considering the huge gap in number between the two types of samples, we have expanded the dataset ourselves on the basis of the dataset provided. The dataset[3] that we used mainly consists of 15763 pictures. The detailed information of the dataset is illustrated in figure 5.

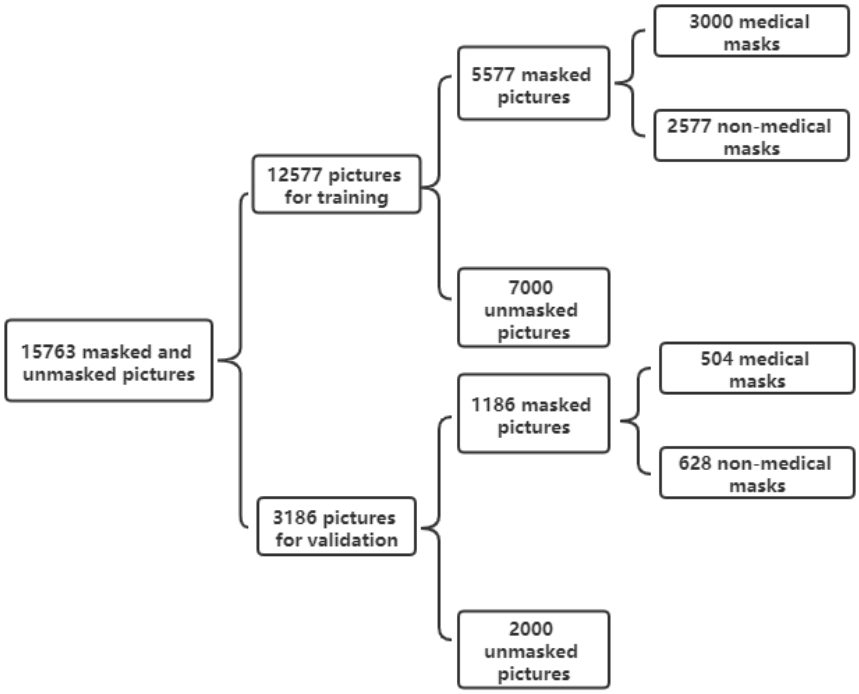


Figure 5: The composition of the dataset

2.1.2 Dataset Pretreatment

To simplify the training process, we place the dataset at our working directory with appropriate order. The composition of the folders are shown in figure 6.

[2]link:https://pan.baidu.com/share/init?surl=XvGepj84SCA9rlVb9rGhEQ password:j3aq
[3]link:https://pan.baidu.com/s/1WQBIqgcvfqrCykJGR3hFgg password:m2ue

dataset			
train		validation	
masked	unmasked	masked	unmasked

Figure 6: The directory of data files

We first divide the data into training data and validation data, inside which has two classes of pictures labeled as masked or unmasked respectively. Putting data this way provide much conveience for us to load dataset using the function ”ImageDataGenerator” and method ”flow_from_directory”. To gain more diversity and variety, we randomly changed the pictures by rotating, width/height shrifting, shearing/zooming and horizontal flip. It can be done by specifying the parameters in function ”ImageDataGenerator”.

2.2 Build Model Based On Basic Convolution Layers

Given the simplicity of this two classes classification, we build a relatively simple but high efficient convolutional neural network. The CNN is connected by three independent sets of convolutional layer and max pooling layer, which pass through a flatten layer and followed by two dense layers to the output as shown in the figure 7. This basic CNN contains 17,374,049 parameters in total, all of which are trainable.

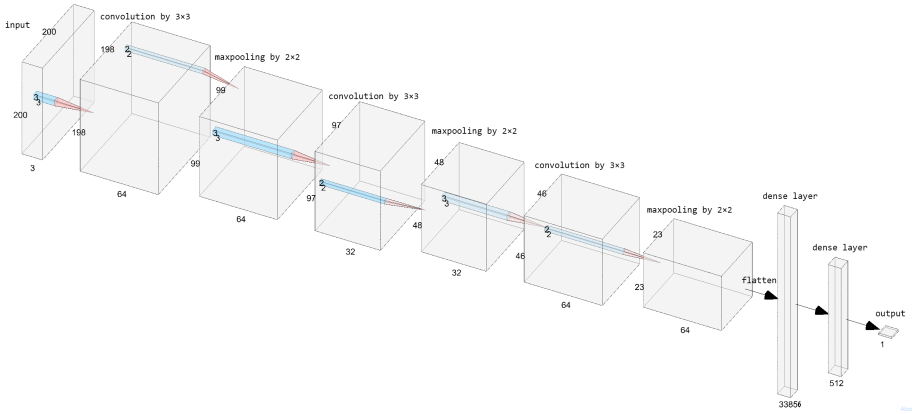


Figure 7: The structure of basic convolutional neural network

For convolution layers and the first dense layer, we use the activation function RELU which perfectly introduces a nonlinearity into the neurons. Also, we use the activation function Sigmoid for the output layer, as Sigmoid is great for binary classifications.

When compiling the model, we specify the loss function as ”binary_crossentrofy”. For optimizer, ”adam” performs well but we set it as ”RMSprop”, thus providing conditions for manually chosen of the learning rate. Later in this paper, we will analyze how the learning rate impact on the validation accuracy.

2.3 Build Model Based On AlexNet

2.3.1 AlexNet

AlexNet was designed by the 2012 ImageNet competition winner Hinton and his student Alex Krizhevsky. Its proposal detonated the upsurge of neural network applications, making CNN the core algorithm model in image classification. After its proposal, more and more deeper neural networks were raised, such as VGG and GoogleLeNet.

The model is divided into eight layers as shown in figure 8: 5 convolutional layers and 3 fully connected layers. Each convolutional layer contains the activation function RELU and local response normalization (LRN) processing, and then undergoes pooling.

There have been many breakthroughs in AlexNet at that time, such as the adoption of RELU functions, the use of the Dropout layer, and the introduction of local response normalization (LRN), and data augmentation.

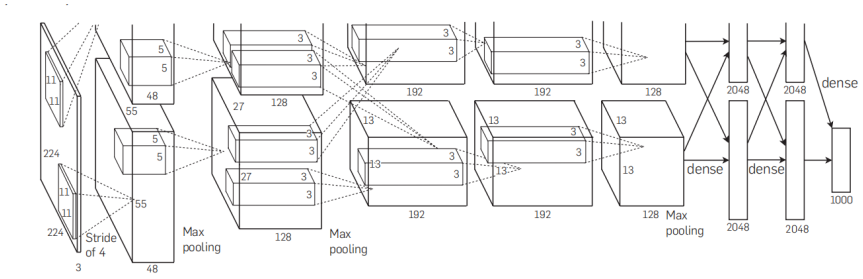


Figure 8: The structure of AlexNet[3]

2.3.2 The Implementation of AlexNet in This Case

According to the definition of AlexNet, we implement AlexNet by adjusting the size of input layer as ”(224, 224, 3)”, after which follows by sets of convolution layers and pooling layers. To gain more accuracy, we tried to use dropout layers alongside with dense layers, as dense layer has great chances of overfitting.

Although we added additional dropout layers, the model still not performs well. So we used ”batch_normalization” after each pooling layers. However, the final accuracy remains still, slightly over 55% on the validation data. Many reasons may lead to this phenomenon, among which the degradation is the most possible one.

We also use VGG and other advanced and more deepd CNN models in this case. However, the final accuracy is still not ideal. We think that degradation is also an important reason.

2.4 Inspiration from ZFNet

2.4.1 ZFNet

ZFNet was first proposed in the famous paper *Visualizing and Understanding Convolutional Networks*[4]. It is based on AlexNet and only have modification in some details, but no big breakthrough in the whole network structure.

The main contribution of this model is to reveal what the layers of the neural network are doing by using visualization method called "feature visualization", as shown in figure 9. A multi-layer deconvolution and unpooling network is used to visualize the evolution of features during training and identify potential problems[4]. After all, if you don't know why the CNN has achieved such good results, you can only rely on continuous experiments to find a better model. Meanwhile, according to the influence of coverd image parts on classification results, the input information of which part is more important for classification task is also discussed.

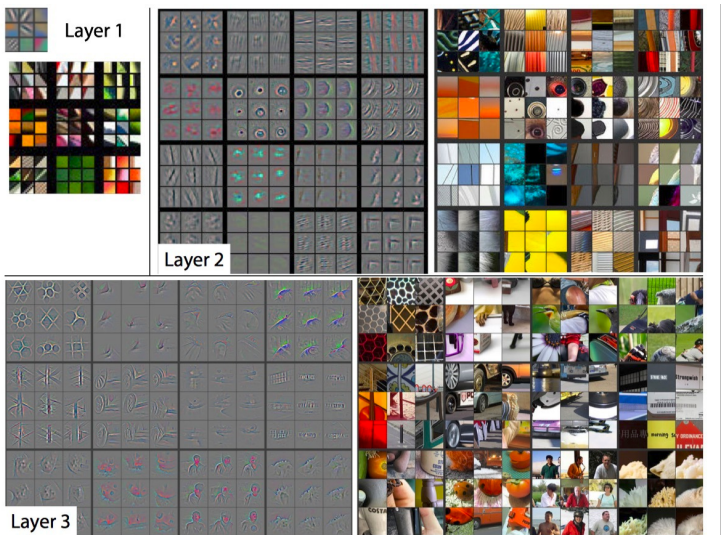


Figure 9: "feature visualization"[4]

2.4.2 The Inspiration from ZFNet

The visualization method "feature visualization" used in ZFNet are too difficult to repetition. It depends on deconvolution and unpooling to map the feature maps to the input space. However, we are still greatly inspired. In this paper, instead of using feature visualization, we conduct two other visualization methods: convolution output visualization and class activation map(CAM) visualization. Both we will discuss in the next part of this paper in details.

3 Model Optimization and Evaluation

3.1 The Impact of Learning Rate

At the beginning, we set learning rate as 0.001, which has both high efficiency and accuracy of 98.62% on the validation data. By adjusting it as 0.01, the accuracy on training data is hovering at 55.66% rather than converged at a bigger value. The loss and accuracy at "lr=0.01" is shown in table 2.

	epoch				
	1	2	3	4	5
accuracy	0.5437	0.5592	0.5566	0.5566	0.5566
loss	51.2649	2.3551	0.6870	0.6872	0.6870

Table 2: The loss and accuracy on training data for each epoch at lr=0.01

On the contrary, if we assign "lr=0.0001", the result is much closer to the 0.001 one. The loss and accuracy at "lr=0.01" is shown in the table 3.

	epoch				
	1	2	3	4	5
accuracy	0.9420	0.9770	9814	0.9843	0.9874
loss	0.1604	0.0708	0.0591	0.0486	0.0422

Table 3: The loss and accuracy on training data for each epoch at lr=0.0001

3.2 The Impact of Optimizer

In the previous training, we use optimizer RMSprop. Now we set "optimizer='adam' " to see the impact of different optimizers on the accuracy. The accuracy on the training data is shown in table 4. The accuracy on the validation data is 98.05%.

	epoch				
	1	2	3	4	5
accuracy	0.9503	0.9826	0.9867	0.9896	0.9899
loss	0.1509	0.0587	0.0432	0.0357	0.0086

Table 4: The loss and accuracy on training data for optimizer adam

From table 4, the accuracy on the training data approaches 95.03% in the first epoch. So we can conclude that adam has higher efficiency than the RMSprop optimizer.

3.3 The Impact of Batch Size

By default, we set batch_size to 64. Reducing its value can reduce the amount of data per step, improve memory efficiency while it also increases the training steps per epoch and even makes the model unable to convergence.

To verify how batch_size influence the accuracy, we reset it 32 to see the result. The training process are shown in table 5 below. The accuracy on the validation data is 98.49%.

	epoch				
	1	2	3	4	5
accuracy	0.9568	0.9780	0.9816	0.9854	0.9890
loss	0.1351	0.0770	0.0622	0.0489	0.0353

Table 5: The loss and accuracy on training data for batch_size=32

3.4 Visualization——Inspired by ZFNet

As we have already mentioned in this paper, we will use convolution output visualization and class activation map (CAM) visualization to show how CNN process and recognize images.

3.4.1 Convolution Output Visualization

Figure 10 shows the visualization of the convolution output images in the intermediate process after two photos are respectively input into the CNN of three convolution layers that have been trained (described in 2.2).

Figure 10 (a) and 10 (b) are input photos to be identified, in which 10 (a) wears a mask while 10 (b) does not wear a mask. Both of them can be classified correctly.

Figure 10 (c), 10 (e) and 10 (g) are image visualization of the first 16 feature maps after the first, second and third convolution and pooling with image 10 (a) as the input. Figure 10 (d), 10 (f), 10 (h) and the input 10 (b) have the same relationship.

For 10 (a), we can find that with the increase of convolution layers, the features of the original picture are gradually extracted, such as eyes, mask, mask edge, hair, etc. For 10 (b), we can also see eyes, hair and the contour lines of face being gradually extracted.

In a word, through the visualization of convolution output, the feature extraction effect of the convolution pooling layer on the original image is intuitively shown.

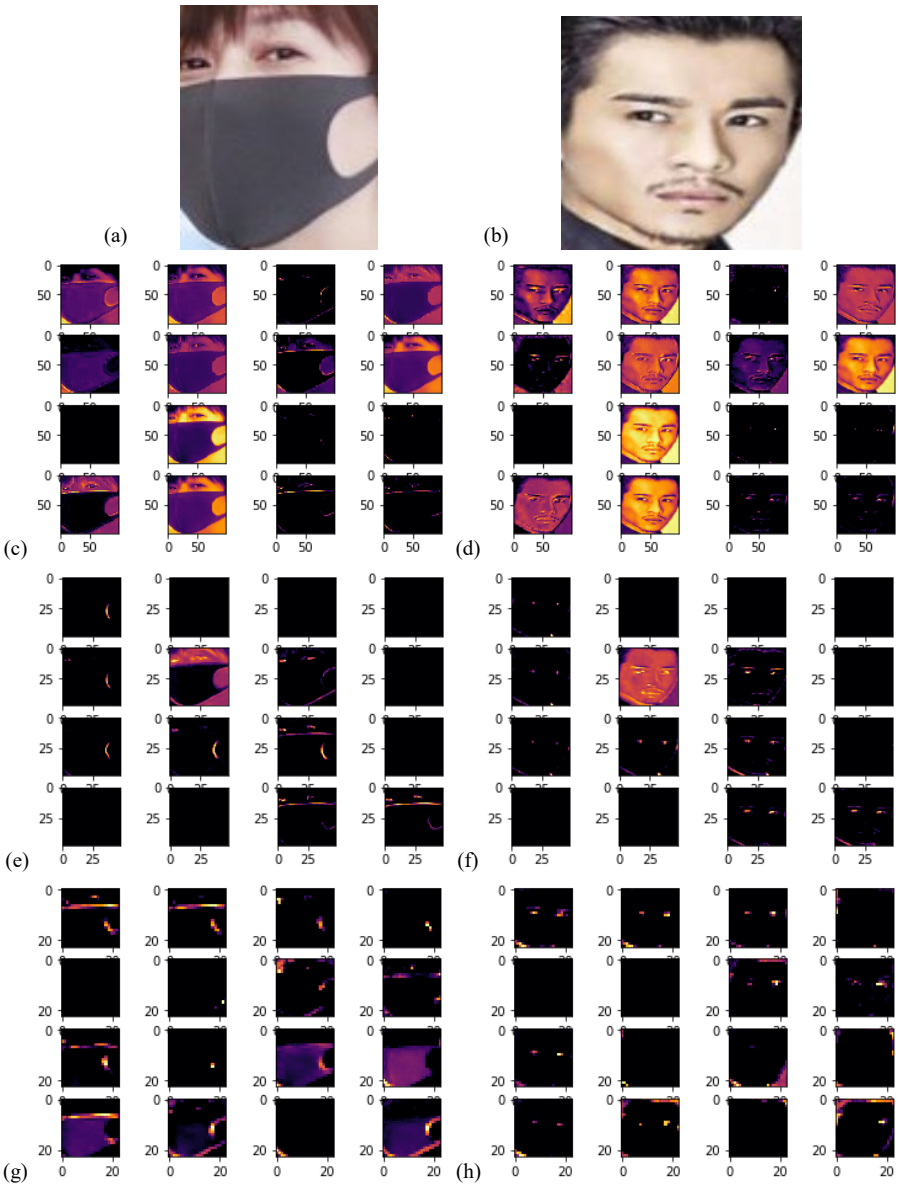


Figure 10: Convolution output visualization

3.4.2 Class Activation Map (CAM) Visualization

A class activation heat map is a 2D grid of scores associated with a particular output class. As shown in figure11, we calculate the weight of feature maps of the last convolution layer on the final judgment result. These feature maps are weighted by corresponding weights and then superposed into a feature map, so that it can be intuitively seen which area the model is judging by. For example, in figure 11, we can know by CAM that the CNN judges the input image as a cat mainly by the face area of the cat in the input image.

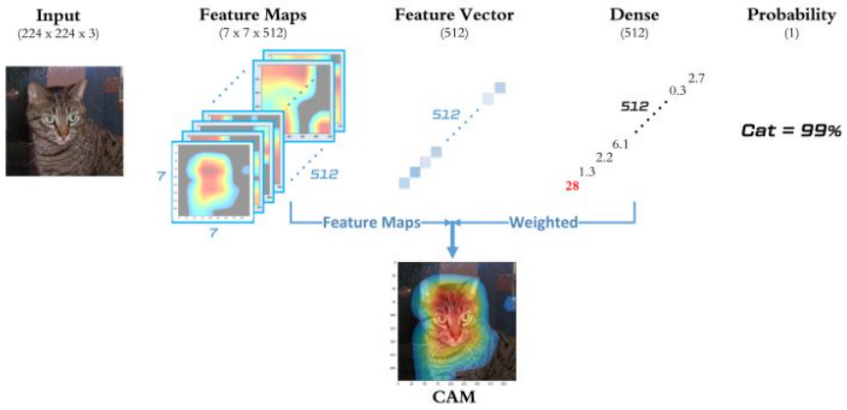


Figure 11: The principle diagram of CAM[5]

In figure 12, 12 (c) and 12 (d) are CAM heat maps of photo 12 (a) and 12 (b). 12 (e) and 12 (f) are also superposition of the output image of the last pooling layer, but all of the weights equal to 1, which is the only difference from 12 (c) and 12 (d). In other words, in 12 (e) and 12 (f), all feature maps of the last pooling layer have the same contribution to the result. By comparing 12 (c), 12 (d) with 12 (e), 12 (f), we can see more clearly how the full connection layer distinguishes between wearing a mask and not wearing a mask by these multiple features.

From 12 (e), we can see that hair, eyes and the edge of the mask are features of the photo 12 (a) extracted by convolution and pooling layers. Compare 12 (c) and 12 (e), we can see that when judging masked faces, eyes and hair are most important features while the edge of the mask seems to have little contribution, which is really unexpected.

In the same way, we can see that eyes, mouth and the edge of the face are features of the photo 12 (b) extracted by convolution and pooling layers. All of the features especially eyes and the edge of the face contribute to the final judgement.

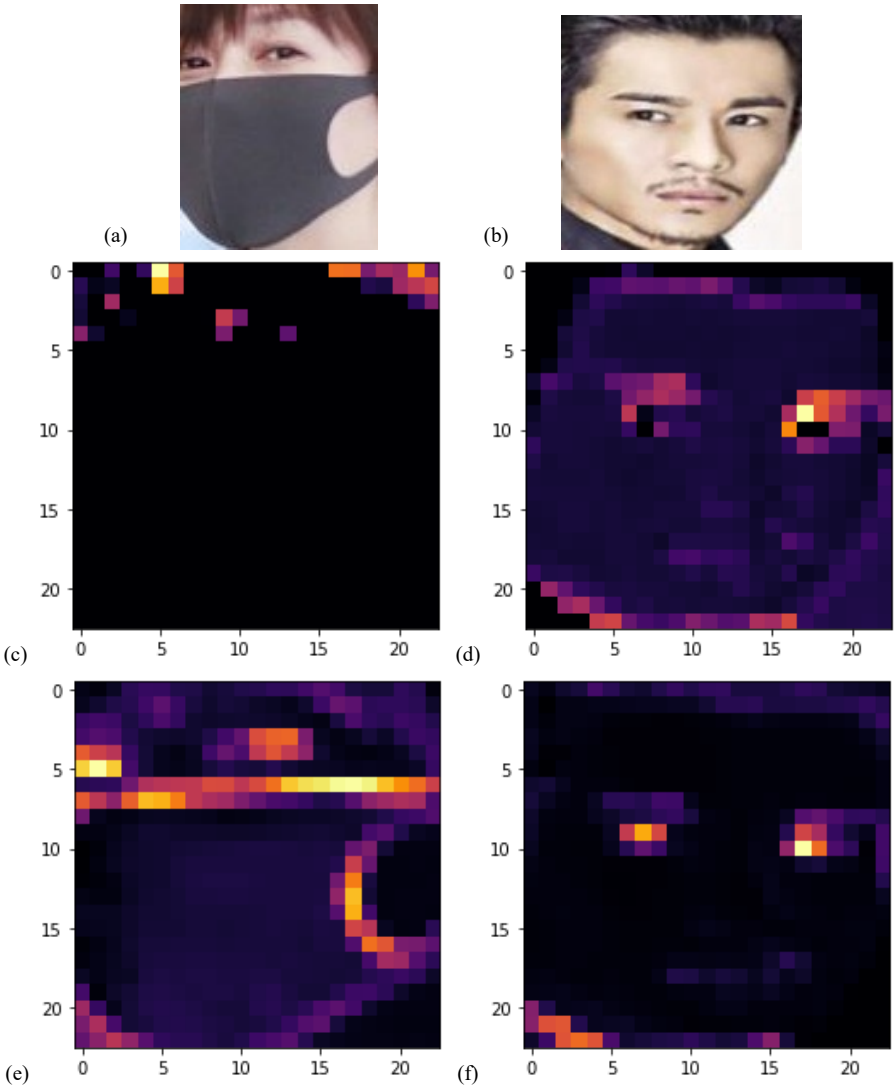
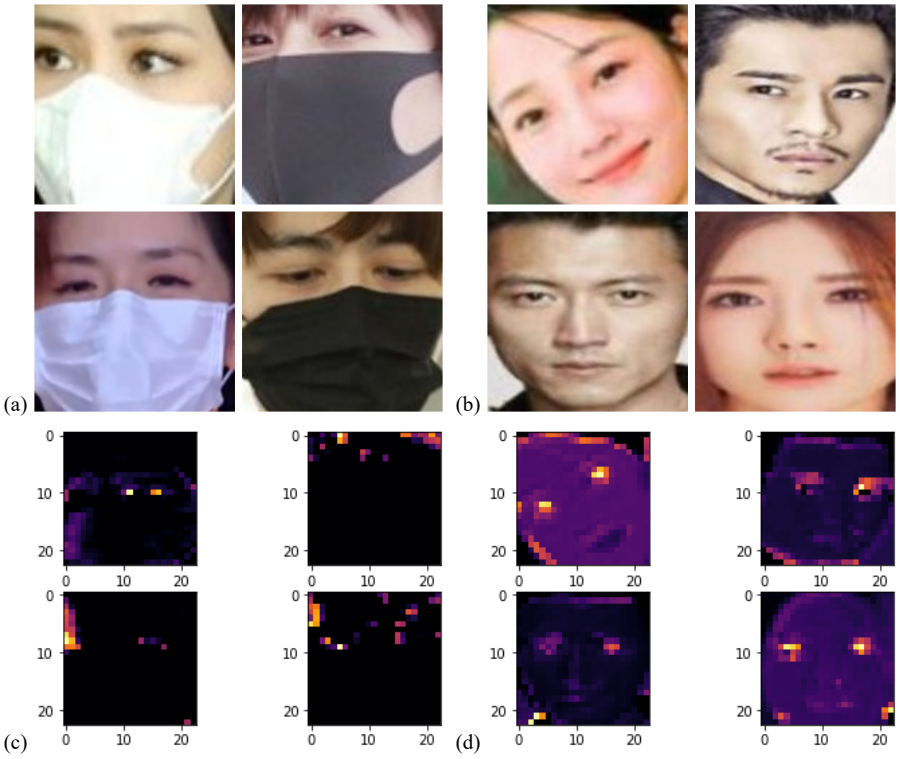


Figure 12: Original photos, class activation heat maps and figures integrating feature maps in a 1:1 ratio

Figure 13 further confirms our findings. Eyes are important features both in masked and unmasked face images. However, the existance of mask destroys the integrity of the contours of the face, thus the edge of the face cannot be extracted. So as for masked faces, hair and eyes are the most important while features in the bottom half of the picture are of little importance. But as for unmasked faces, the whole face edge and all the facial features such as eyes, mouth and nose are detected and play key roles in the judgement.



(a):Top Left: face with white mask in train dataset; Top Right: face with black mask in train dataset; Bottom Left: face with white mask in test dataset; Bottom Right: face with white mask in test dataset
(b):Top Left and Top Right: unmasked faces in train dataset; Bottom Left and Bottom Right: unmasked faces in test dataset
(c): CAM heat maps corresponding to (a)
(d): CAM heat maps corresponding to (b)

Figure 13: CAM heat maps of masked and unmasked faces

4 Conclusion

After multiple parameters adjustments and trail runs, we finally ended up with the best results. For the basic convolutional neural network, it obtains an accuracy of 98.74% on the training data and 98.62% on the validation data with eight epochs. Table 6 shows the training process.

	epoch							
	1	2	3	4	5	6	7	8
accuracy	0.9141	0.9657	0.9743	0.9790	0.9848	0.9851	0.9861	0.9874
loss	0.3889	0.1290	0.0918	0.0700	0.0534	0.0698	0.0556	0.0486

Table 6: The loss and accuracy on training data for each epoch at lr=0.001

Acknowledgements

There does not always have smooth sledding in the field of machine learning. Through this project, we have experienced difficulties that may be big or small, but with much nonreimbursable assistance from the following people, these problems were readily solved. Here, we would like to appreciate those who have offered us support during the project.

First, our sincere thanks is given to Professor Chen Li and assistant Zhang Xueheng. During this semester, it is our honor to have such an inspiring and enlightening course.

Also, particular acknowledgement is given to Andrew Ng, for providing the course of TensorFlow on the coursera. It did help us go deeper into the practice of TensorFlow.

Last but not least, we would like to express our most heartfelt gratitude to Wuhan citizens, it is your braveness and persistance that defeated the coronavirus.

References

[1] 量子位. 如何理解卷积神经网络（CNN）中的卷积和池化？[Z]. [EB/OL]. https://www.zhihu.com/question/49376084. Accessed June 7, 2020.

[2] [EB/OL]. https://tensorflow.google.cn/. Accessed June 4, 2020.

[3] KRIZHEVSKY A, SUTSKEVER I, HINTON G E. ImageNet Classification with Deep Convolutional Neural Networks[J/OL]. Commun. ACM, 2017, 60(6): 84-90. https://doi.org/10.1145/3065386. DOI: 10.1145/3065386.

[4] ZEILER M D, FERGUS R. Visualizing and Understanding Convolutional Networks[Z]. 2013. arXiv: 1311.2901 [cs.CV].

[5] YUAN M. 如何利用 CAM（类激活图）动态可视化模型的学习过程？[Z]. [EB/OL]. https://zhuanlan.zhihu.com/p/29567314. Accessed June 7, 2020.

Division of Work

Zhang Yifei:

- Write code of the basic convolutional neural network
- Write code of the visualization

Sang Rui:

- Optimize the code
- Collect dataset