Face Expression Recognition Based on Deep Learning

Tingxuan Zhang
International School, Beijing University of Posts and Telecommunications, Beijing, China
*Corresponding author e-mail: zhangtingxuan@bupt.edu.cn

Abstract. Nowadays, the information technology such as artificial intelligence (AI) and the cloud classroom have been increasingly introduced to improve the quality of modern education. It is well known that teachers can adjust the teaching mode through the interactive performance of students in the classroom. For example, if many students show surprise expressions on their faces, it means that students may not understand what the teacher is talking about, and the teacher needs to slow down the rhythm and explain the knowledge point carefully. For another example, if students present positive expressions, it means that the teacher has mobilized the atmosphere of the classroom well and should keep this good teaching style to make students be interested in this subject. All in all, face expression recognition is quite useful in the modern society.

1. Introduction:
Four kinds of expressions (positive, negative, focused, surprise) can be recognized by the system. Through the results of records, we can find that the classification results (mAP) obtained by the traditional machine learning method are low (less than 60%). Although the traditional machine learning method is very short in running operation, its accuracy is not as expected, so after constructing the base line model, the project needs to be processed further using some methods similar to CNN (Convolutional Neural Network).

With the help of the general CNN model, the mAP increased to more than 60% which is 67.8% to be exact, and the average time of recognition was about 0.80s. Although the general CNN method can greatly increase the accuracy of the project, I thought that I could further improve the accuracy of this project before the mid-term inspection, and tried to make mAP to be more than 70%. I further learned about R-CNN (Regions with Convolutional Neural Network), fast R-CNN, faster R-CNN. Through further learning, I found that the deep learning method based on region proposal for object detection can greatly improve the accuracy and speed of the program, but we still needed to find a solution to the problem of real-time object detection and recognition. Because of the large amount of the parameters of the CNN models, the delay of the project is quite long. If we need to use such a software on the classroom interactive system, the delay of more than 0.80 seconds is terrible, so I decided to learn some deep learning method based on regression. Through in-depth study I found that YOLO (You only look Once) and SSD (single shot multibox detector) can solve the problem of excessive delay and achieve real-time object detection and recognition. In this project, I hope to achieve faster running speed by reducing the parameters of the network in the CNN to achieve real-time object detection and recognition. By learning about several papers found on the network, I learned the important idea of reducing network parameters in CNN. Ideas in the NIN (Network in Network) gave me a lot of inspiration, one of which is GAP (Global Average Pooling) for the alternative analysis of the fully connected layer. With the help of GAP, we could remove the fully connected layer, which was achieved by mapping the same number of feature
maps as the number of ultimate classified classes in the last convolutional layer, and transferring each reduced feature map into a softmax activation function, reducing the number of parameters involved in this process. In the end, the part of the expression recognition in this classroom interactive system has achieved the expected results, it can complete real-time (less than 0.2 seconds) expression recognition, and the accuracy rate is above 70%.

2. Background:
Facial expression recognition: Most traditional methods use manually extracted features, or shallow learning. But after 2013, various expression recognition competitions collected a wealth of datasets from challenging real-world scenarios, driving FER technology to the real scene from the lab. Due to the increase in the amount of data, conventional features are insufficient to represent the diversity of factors unrelated to facial expressions. With the dramatic improvement of the performance of GPU and the design of various excellent neural network structures, many fields have begun to turn to deep learning methods, which greatly improve the accuracy of recognition. [1] What’s more, deep learning techniques are increasingly being used to address the challenges of facial expression recognition in real-world environments.

FER-2013 facial expression dataset: FER-2013 facial expressions dataset which was made by Pierre-Luc Carrier and Aaron Courville was used in this project to deal with the part of facial expressions recognition. Some examples of FER-2013 are shown in Figure 1.

![Figure 1 Examples of FER-2013 facial expressions dataset](image)

The Fer-2013 facial expression data set consists of 35,886 facial expression images, including 28,708 training pictures, 3,589 validation pictures and test picture. Each picture is composed of a grayscale image with a fixed size of 48×48. Most of faces in this dataset are centred and most of them occupy about the same amount of space in each image. There are 7 kinds of expressions, corresponding to the digital label 0-6 shown in Figure 2, the specific expression corresponding to the label are as follows: anger=0; disgust=1; fear=2; happy=3; sad=4; surprised=5; neutral=6.[2]

![Figure 2 Seven kinds of expressions corresponding to the digital label 0-6](image)
CNN: "The CNN (Convolutional Neural Network) consists of neurons. And each neuron has its own weight and bias. CNN is a feedforward neural network whose artificial neurons can respond to a surrounding area of a part of the coverage. Each neuron receives some input and does some dot product calculations. The output is the score for each category. It has excellent performance for large image processing. Some calculation techniques in the common neural network are still applicable here. The convolutional neural network default input is an image that allows us to encode specific properties into the network structure, making our feedforward function more efficient and reducing a large number of parameters." said by Feifei Li who is the professor of Department of Computer Science in Stanford University.

Traditional neural network is shown in Figure 3:

![Figure 3 Traditional neural network](image1)

Convolutional neural network is shown in Figure 4:

![Figure 4 Convolutional neural network](image2)

It can be observed that a convolutional neural network consists of many layers. The inputs of this kind of network are three-dimensional, and the outputs are also three-dimensional. Some layers have parameters, and some layers do not have parameters. CNN is mainly composed of the following layers:

- Convolutional layer: Usually in a convolutional neural network, there are lots of convolutional units in the convolutional layer. The parameters of the convolution unit can be obtained by a back-propagation algorithm. Through convolution operations, we can extract many features which are different from each other. The lower the level of the layer is, the lower the features we can extract; conversely, higher-level networks allow us to extract more complex features from low-level features. An example of the convolutional layer is shown in Figure 5.
Figure 5 An example of the convolutional layer

Pooling layer: We will get the feature with a large dimension after the convolutional layer. This large dimension feature is not convenient for us to operate later, so we need to cut the feature into several regions and take the maximum value or average value or some other methods to obtain new and smaller-dimensional features. The most common pooling layer is 2*2 with a stride of 2, and the maximum value is selected in each region. The example of pooling layer is shown in Figure 6.

Figure 6 The example of pooling layer

The fully connected layer: This layer combines all local features into global features to calculate the score for each category.

3. Face expression recognition:
In the part of facial expression recognition, the system should judge the students’ expressions (positive, negative, surprise, focused). In addition, the system should send the detected expressions of the students to the teacher, and score the students' expressions at this time to reflect the performance of each student in the class. It can help teachers to have an intuitive understanding of the students' class performance.

Convolutional neural network: The general CNN is consisted of an input and an output layer and a plurality of hidden layers. These hidden layers can be divided into convolution layers, pooling layers, RELU layers, and dense (fully connected) layer.

I will introduce the function of each layer in the project one by one. The first is the input layer. Just as it literally means, we need to input fixed-dimensional data in the input layer. I have processed the image data in Dataset pre-processing. For the purpose of preventing the network from over-fitting, some image transformations such as flipping, rotating, cutting, etc. can be done artificially which are called data augmentation. Figure 7 shows the process of the data augmentation. The project uses OpenCV which is a computer vision library for image processing. The input layer gets a 48x48 pixel grayscale image and this image should be cropped randomly to 42*42 pixel and mirrored randomly so as to enhance the robustness of this network. The output of the input layer is a 42x42 pixel grayscale image.
Next, let’s talk about the function of the convolutional layer. I can pass the numpy array of the input layer to the Convolution2D layer. Here I need to define some hyperparameters in the model. The hyperparameters of the convolutional layer mainly include the size of the convolution kernel, the stride of the convolution operation, the number of convolution kernels, and whether padding is needed. And then I use weight sharing which is to use the same filter to scan the obtained feature map to generate a new feature map. The feature map generated by the convolutional layer represents the strength of the pixel value.

And then we will explore the pooling layer. In this model, each convolutional layer is connected to the pooling layer to retain the main features while reducing the parameters and calculations of the next layer. For each pooling layer, we use 3×3 MaxPooling2D and set stride to 2.

Before the final output layer, there are two fully connected layers that play an important role in this model just like the classifier. Here, I need to set the appropriate dropout which is to discard the neural network unit temporarily from the network with a certain probability, and I set the dropout of both fully connected layers to 40%.

In the output layer, I will get a 1 × 1 × 6 output. I use the softmax function which is a function that each element has a range between (0,1) and the sum of all elements is 1. The softmax function is: \( S_i = \frac{e^i}{\sum_j e^j} \).

With the softmax function, I can get the probability of outputting each type of expression. What’s more, I need to set a function which is the cross-entropy loss function: \( \text{Loss} = -\sum y_i \ln a_i \). Where \( y \) represents our true value, \( a \) represents the value obtained by softmax function and \( i \) represents the node number. I also need to use SGD (Stochastic Gradient Descent) which is a method using one batch of samples per iteration to calculate the parameters in the network.

The CNN model used in this project is shown in Table 1.

| Layer         | Kernel | Stride | Padding | Output       |
|---------------|--------|--------|---------|--------------|
| Input         |        |        |         | 42×42×1      |
| Convolution1  | 5×5    | 1      | 2       | 42×42×32     |
| Pooling1      | 3×3    | 2      |         | 21×21×32     |
| Convolution2  | 4×4    | 1      | 1       | 20×20×32     |
| Pooling2      | 3×3    | 2      |         | 10×10×32     |
| Convolution3  | 5×5    | 1      | 2       | 10×10×64     |
| Pooling3      | 3×3    | 2      |         | 5×5×64       |
| Full-connected4 |      |        |         | 1×1×2048    |
| Full-connected5 |      |        |         | 1×1×1024    |
| Output        |        |        |         | 1×1×6       |
Through the above CNN model, this system can achieve a good accuracy in the part of facial expression recognition, but still needs to be improved in terms of speed. Since there are millions of parameters in the model, it will take a long time for this system to do the recognition and cannot achieve real-time effects. In order to make this system faster, I decided to improve the model by reducing the parameters.

As we all know, most of the people percept something from its local to global. We expect that the working mode of the neural network can perceive the picture from the local as humans, instead of observing the global picture. The neuron network only needs to observe the local part of the image, and then integrate the local information at a higher level in order to obtain global information. After the local connection process, the number of connections between neurons has been reduced. However, it has not actually decreased a lot, and the number of parameters is still large. The number of parameters can be greatly reduced by weight sharing. We can treat each convolution kernel as a feature extraction method, which is independent of the position of the data such as images. This means that for the same convolution kernel, the features it extracts in one region can also be applied to other regions. Finally, after the pooling layer, the model parameters are much reduced.

I expected to use two methods to reduce parameters in the model, one is to use GAP (Global Average Pooling), another one is to use depth-wise separable convolution.

GAP can mainly reduce parameters in the full connected layers. Basically, all neural network-based machine learning algorithms add one or more fully connected layers in order to the vectorization of features before the output layer. Sometimes designing several fully connected networks can improve the classification performance. However, scientists also noticed that the fully connected layer has a very fatal weakness, that is, there are too many parameters in the model because of the fully connected layer. On the one hand, the calculation amount of training and testing is increased, and the speed is reduced; on the other hand, too many parameters may cause over-fitting. The fully connected layer expands the convolution layer into vectors and then classifies each feature map. GAP is a method to combine the above two processes into one step, which means that traditional fully connected layers can be replaced by using GAP. This operation is its literal meaning: the feature map is globally averaged to output a value, that is, a tensor of $W \times H \times D$ is changed to a tensor of $1 \times 1 \times D$ and the final vector is sent to the softmax layer. [3] By using GAP we can remove the fully connected layer, which was achieved by mapping the same number of feature maps as the number of ultimate classified classes in the last convolutional layer, and transferring each reduced feature map into a softmax activation function, reducing the number of parameters involved in this process. Before using GAP, the number of parameters we need to use in Full-connected4 is $2048 \times (5 \times 5 \times 6411) = 3278848$, which is a quite large number more than 3 million. But we can use GAP to make the output of Convolution3 $10 \times 10 \times 64$ convert to $1 \times 1 \times 64$, and then send it to softmax for classification. Figure 8 shows the comparison of fully connected layers and global average pooling.

I can use depth-wise separable convolution to reduce parameters in the convolution layers. The depth-wise separable convolution is a convolution of different convolution kernels for different input channels, which decomposes the normal convolution operation into depth-wise convolution and point-wise convolution. Figure 9 shows the framework of the depth-wise separable convolution.
Figure 9 The framework of the depth-wise separable convolution

General convolution (shown in Figure 10): Suppose there is an input of N×H×W×C, and there are k 3×3 kernels. If padding=1, stride=1, then the output of ordinary convolution is N×H×W×k.

Figure 10 General convolution

Depth-wise convolution (shown in Figure 11): Depth-wise convolution means that the input of N×H×W×C is divided into C groups, and then each group uses 3×3 kernel to do the convolution, which is the same as collecting the spatial characteristics of each channel, that is, the depth-wise feature.

Figure 11 Depth-wise convolution

Point-wise convolution (shown in Figure 12): Point-wise convolution means that it makes k 1×1 convolutions on the input of N×H×W×C, which is equivalent to collecting the features of each point, that is, the point-wise feature. The output obtained by depth-wise convolution and point-wise convolution is also N×H×W×K which is the same as the output of the ordinary convolution.

Figure 12 Point-wise convolution
For the CNN model I designed before, using depth-wise separable convolution can greatly reduce the amount of computation. For example, for Convolution3 layer, its input is $10 \times 10 \times 32$, the output is $10 \times 10 \times 64$, and the size of the kernel is $5 \times 5$. If I use general convolution, the amount of calculation of this process is $5 \times 5 \times 32 \times 64 \times 10 \times 10 = 5120000$. If I use depth-wise separable convolution, the amount of calculation of this process is $5 \times 5 \times 32 \times 10 \times 1001 \times 32 \times 64 \times 10 \times 10 = 284800$. In contrast, the latter convolution method can reduce the amount of calculation to 0.055 times, which greatly increases the speed of operation.

In summary, I used GAP (Global Average Pooling) to reduce the number of parameters in the fully connected layer, and used depth-wise separable convolution to reduce the computation of the convolution layer. Through these two methods, the CNN model I designed has been greatly modified, and the part of facial expression recognition can achieve the expected effect in terms of time and accuracy.[4]

By using GAP and depth-wise separable convolution, the system has greatly improved the running speed, and the expression recognition can be completed within 0.2s, and the accuracy rate can reach 70.2%. Since our final system only needs to recognize four expressions instead of the seven expressions in the dataset, the final expression recognition accuracy rate is greatly improved to 73.8%. The test accuracy confusion matrix of the four expression recognitions is shown in Table 2.

Table 2 The test accuracy confusion matrix of the four expression recognitions

| Actual value | Predicted value | Positive | Negative | Surprise | Focused |
|--------------|----------------|----------|----------|----------|---------|
| Positive     | Positive       | 87.7     | 6.9      | 2.1      | 3.3     |
|              | Negative       | 2.4      | 64       | 11.4     | 22.2    |
|              | Surprise       | 4.1      | 9.5      | 84.7     | 1.7     |
|              | Focused        | 5.8      | 19.6     | 1.8      | 72.8    |

4. Conclusion:
I first merged the data and did the data augmentation to complete the data pre-processing. And then I used traditional machine learning methods such as logistic regression, SVM, and random forest to realize the function, but the traditional machine learning method does not meet the accuracy required by the cloud classroom system. And then I switched to the deep learning method to deal with this problem. I first used the general convolutional neural network to build the base line model, but the parameters of this model are too large, which seriously affects the running speed of the system and does not meet the real-time detection effect in the classroom. I need to use some methods to lighten the model as much as possible to make it run faster. I have reviewed a lot of information and learned about methods such as faster RCNN, YOLO, SSD, etc. Finally, I used GAP to reduce parameters in the fully connected layers in the neural network and used depth-wise separable convolution to reduce parameters in the convolution layers. Finally, the system can recognize four expressions (positive, negative, focused, surprise) within 0.2s, and the recognition accuracy can reach 73.8%.

Reference
[1] Li, J., Zhang, D., Zhang, J., Zhang, J., Li, T. and Xia, Y. et al. (2017). Facial expression recognition with faster r-cnn. Procedia Computer Science, 107, pp. 135-140.
[2] Kaggel, (2013). Challenges in Representation Learning: Facial Expression Recognition Challenge. [online] Available from: https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge/data [Accessed 5th April 2019].
[3] Lin, M., Chen, Q. and Yan, S. (2013). Network in network. arXiv preprint arXiv:1312.4400.
[4] Li, F. (2017). CS231n: Convolutional Neural Networks for Visual Recognition. [Stanford University]. San Francisco.