Available online www.jsaer.com

***Journal of Scientific and Engineering Research*, 2020, 7(9):81-86**

Research Article

ISSN: 2394-2630
CODEN(USA): JSERBR

Face Recognition System Design based on Convolutional Neural Network

Zhouxu Zhan, Hongli Zhu

School of Information and Electronic Engineering, Zhejiang University City College, Hangzhou, China
zhuhl@zucc.edu.cn

Abstract In recent years, the problem of garbage classification has gradually entered our vision. More and more cities have begun to implement garbage classification policies. However, relying on citizens' manual classification has the problems of low efficiency and low accuracy. It is harmful to young children and older people. There are certain difficulties for the elderly in, and it is difficult for citizens to distinguish the types of rubbish when the light is low near the trash can. In response to this problem, this article combines deep learning and garbage classification, and uses the Inception-ResNet-v2 model in the convolutional network for modeling. The two thousand or so data sets are divided into training sets and test sets. Types of trash: plastic, glass, metal, waste cardboard, and paper have undergone in-depth learning. After multiple modeling, a model with relatively good accuracy has been obtained, and certain research has been conducted on the intelligent classification of urban trash cans. The intelligent classification of trash cans, which uses deep learning technology in machine learning, brings a certain improvement in efficiency and accuracy compared to manual recognition. Compared with traditional image recognition, deep learning has less impact on image quality, high concurrency, and too many types. It can recognize images better, which is the main direction of image recognition in recent years.

Keywords Convolutional Neural Network, Face recognition, LeNet-5

1. Introduction

With the rapid development of technologies such as big data, cloud computing, and mobile communication systems, the research on artificial intelligence has once again triggered an upsurge in the world. The "face recognition system", as a combination of artificial intelligence, machine recognition, machine learning, model theory, video image processing and other professional technologies, has gradually entered people's field of vision, becoming the most integrated artificial intelligence and deep learning. One of a wide range of applications.

Compared with other biological recognition methods, face recognition has the advantages of ease of use, intuitiveness, non-contact and so on. This also makes the commercial use and demand of face recognition increase day by day. In order to be better applicable to various occasions of life and commercial use, people also put forward more stringent requirements for the robustness and non-restriction of face recognition technology. Convolutional neural network, as one of the core algorithms of computer vision, has characteristics of representation learning and biological similarity that make it shine in the application of face recognition. Applying the convolutional neural network to the face recognition system helps to simplify the complicated problems caused by the diversity of face representations, and is beneficial to the application of face recognition technology in actual scenes.

Nowadays, the application of face recognition has also entered our lives. Alipay's facial recognition payment and face login; airports and train stations use face recognition to achieve intelligent and automated entry; communities use face recognition technology to identify residents and Open the door. Such applications make face recognition technology widely used and promoted in daily life.

2. Processing of image samples

The convolutional neural network has strict requirements for the input, that is, the image should be consistent in size. In this design, the size of the input face image is set to 64×64 to prevent the image pixels from becoming too large. Too much calculation. The processing method is also to use Python to process each image, requiring recognition of the face position and cropping and saving the image. Among them, the frontal_face_detector model in the Dlib library can be used to complete the recognition of the face position, and then the walk function in the OS library is used to obtain the path of the original sample. After obtaining the path, you can use OpenCV to read the image sample. In order to save the face position obtained through model processing, we use Python's own enumerate function to obtain the upper, lower, left, and right boundaries of the face, and re-adjust the image to a size of 64×64 according to the boundary value. This step is still using the OpenCV library To be done.

The processed samples have reached the 64×64 requirements to be set, and the proportion of face images has reached 70% to 80%. The processed sample is shown in Figure 1.

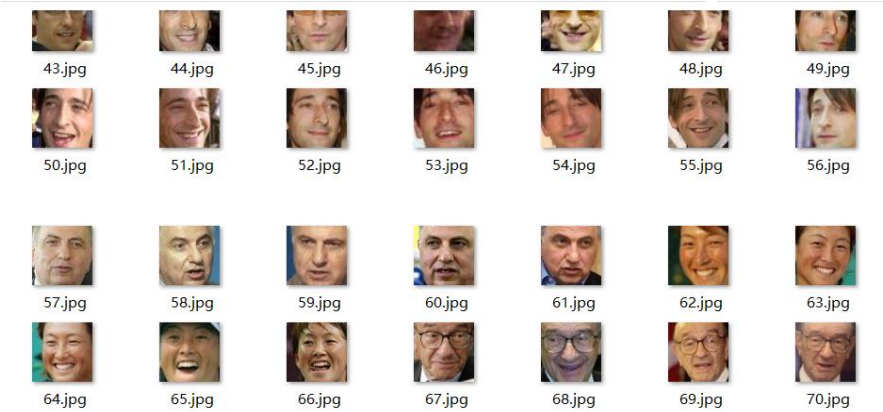

Figure 1: Sample after processing

3. Model construction and training

The most important part of this design is the construction of the entire convolutional neural network. The reasonable setting of each single-layer network parameter can also reduce the amount of calculation overhead to a certain extent, thereby speeding up the training speed of the model and making the model reach convergence faster.

The convolutional network structure designed this time is based on the LeNet-5 model, and its specific structure is shown in Figure 2.

Regarding the number of convolution kernels in each layer and the number of neurons required by the fully connected layer, because there is no theoretical support for the time being, we refer to the configuration parameters of some successful cases in the network. Next, the remaining parameter settings of each layer of the network will be described.

The data is first input into the first convolutional layer, which has 32 convolution kernels, and the size of each convolution kernel is 3×3. According to the theory of convolution kernel and the method of convolution calculation, the smaller the convolution kernel, the smaller the amount of calculation for a single input. And due to the need to improve the receptive field (the amount of information contained in a pixel), the size of the convolution kernel should be greater than 1. And because when the size of the convolution kernel is an even number, the efficiency is low, and when the convolution kernel is an even number, it cannot be guaranteed that the size of the feature image can remain unchanged after the edge filling, so the convolution kernel with a size of 3 is the best. The convolutional layer uses relu as the activation function of the convolutional layer, as shown in Figure 3.

Layer (type)	Output Shape	Param #
conv2d_27 (Conv2D)	(None, 62, 62, 32)	896
max_pooling2d_20 (MaxPooling	(None, 31, 31, 32)	0
dropout_29 (Dropout)	(None, 31, 31, 32)	0
conv2d_28 (Conv2D)	(None, 29, 29, 64)	18496
max_pooling2d_21 (MaxPooling	(None, 14, 14, 64)	0
dropout_30 (Dropout)	(None, 14, 14, 64)	0
flatten_10 (Flatten)	(None, 12544)	0
dense_18 (Dense)	(None, 512)	6423040
dropout_31 (Dropout)	(None, 512)	0
dense_19 (Dense)	(None, 572)	293436

Total params: 6,735,868
Trainable params: 6,735,868
Non-trainable params: 0

Figure 2: CNN network structure

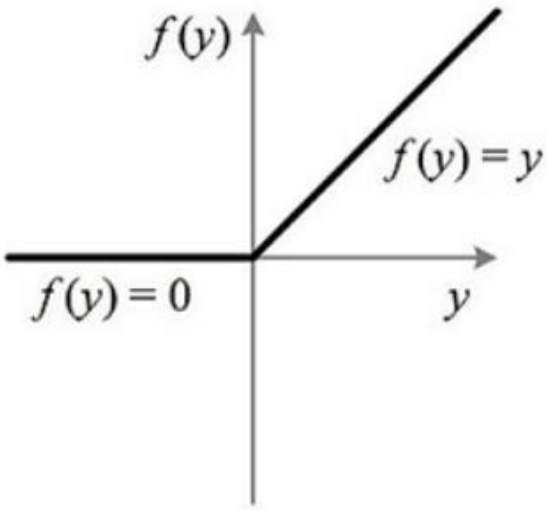

Figure 3: Relu activation function

The unilateral suppression of the relu function increases the sparsity of the network, which enables the trained model to better mine relevant features and effectively prevent data overfitting. But this also makes the convergence speed of the model slower, and the gradient of change will increase significantly after a certain training.

This method can effectively avoid the mean shift caused by the convolutional layer error. Because of this method, the filter size of this layer is 2×2.

Then the data enters a dropout layer, the setting parameter of this layer is p, the specific function realized is to stop a certain neuron with the probability of p during forward transmission. The effect is shown in Figure 4.

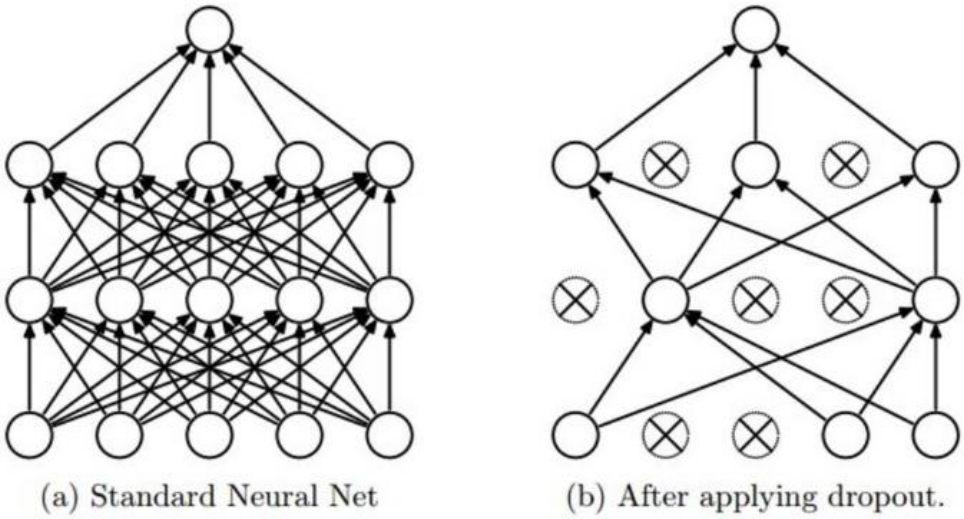

Figure 4: Schematic diagram of the dropout layer

Adding the dropout layer also prevents over-fitting of the data, but again slows the convergence speed of the model. Then the data passes through a network consisting of a convolutional layer, a pooling layer, and a dropout layer to compress the data dimension again.

The data that completes this series of operations is input to a Flatten layer. Because the fully connected layer can only operate on one-dimensional data, the Flatten layer is required to first make the data processed by the convolutional layer and the pooling layer one-dimensional, so as to achieve data entry The transition of the fully connected layer.

Each feature that enters the fully connected layer is selected and combined, and finally the feature of the entire face is obtained. The setting parameter of this layer is the number of neurons in this layer, and the relu function is used as the activation function.

Finally, the data enters the output layer to complete the classification of each face. The number of neurons in the consolidation layer corresponds to the number of objects used in this design. The activation function used in this layer is the sigmoid function, as shown in Figure 5.

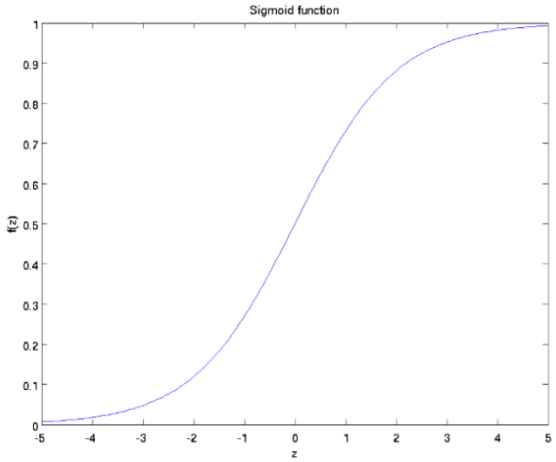

Figure 5: Sigmoid function

It has very good symmetry and has a good application in judging right from wrong, that is, two classification problems.

4. Model training and result testing

Before model training, you need to configure the parameters of the optimizer. In this design, stochastic gradient descent (SGD) is used as the optimization method of back propagation. It can support large-scale data set training and sparse Machine learning is in line with the scale of the data set used this time and the construction of the convolutional neural network. Among the various configuration parameters, lr is the learning rate of the optimizer, which will be gradually updated during the model training process. Here is an initial value. A lower initial value helps the model update iteration and reduces The model may oscillate during training.

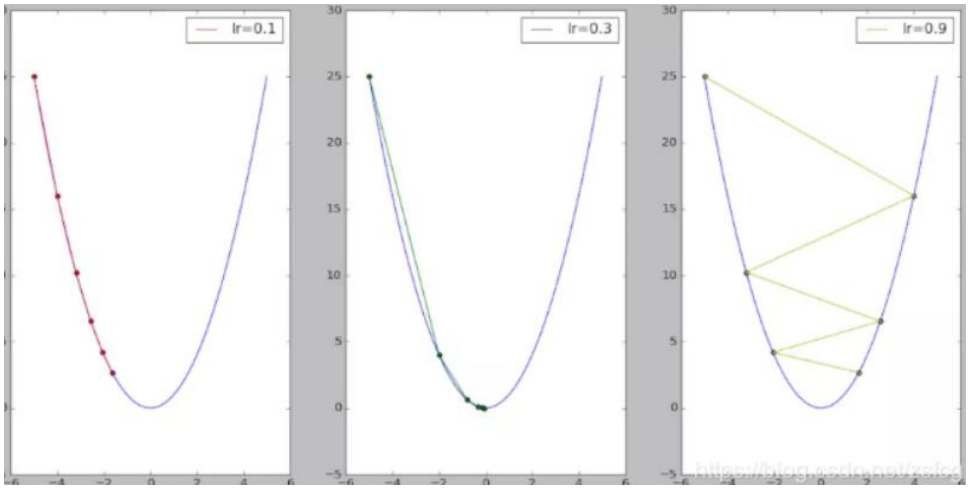

Figure 6: The learning rate gradually increases and the model oscillates

Decay is the updated learning rate decay value. Its specific function is to avoid too large changes in the learning rate update to reach the extreme point. Its introduction also effectively alleviates the shock that occurs during model training.

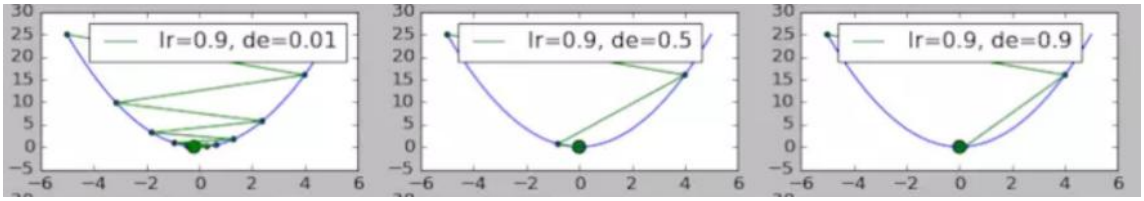

Figure 7 The learning rate attenuation gradually increases, and the model oscillation phenomenon is alleviated

Because of the existence of the learning rate decay value, the learning rate will always decrease during model iteration, but if it does not reach the extreme value at this time, it will greatly increase the time and number of model training, thereby increasing useless computing overhead. Therefore, the introduction of momentum (impulse) is particularly important. He calculates the derivative of the last value point and compares it with the derivative of this value point. If the change does not reach the set value, the value of the learning rate is not attenuated.

Then the optimizer is introduced and the model is compiled. The loss function used in the compilation is cross entropy. The calculation formula is shown in (1)

$$H(p,q) = -\sum_x p(x) \log q(x) \tag{1}$$

It is used to express the distance between the probability distributions of two events. The smaller the cross-entropy, the greater the probability that the two events are closer together. If the cross entropy is large, the model will increase the penalty for the weight, and promote it to reduce the cross entropy, so as to achieve the convergence of the model.

After the compilation of the model is completed, the data is imported into the model, and the training of the model is started, where X_train and y_train correspond to the training set of the data and the label respectively. Epochs is the number of model training. It should be noted that due to the use of the relu activation function and the introduction of the dropout layer, in the early stage of model training, the loss of the model is large and the accuracy is low, and a large number of samples are needed to increase the number of training to promote the loss Optimization of value and accuracy.

```
Epoch 1/40
4785/4785 [==============================] - 63s 13ms/step - loss: 6.1781 - acc: 0.0738 - val_loss: 5.9844 - val_acc: 0.0852
Epoch 2/40
4785/4785 [==============================] - 61s 13ms/step - loss: 6.0461 - acc: 0.0742 - val_loss: 5.9530 - val_acc: 0.0852
Epoch 3/40
4785/4785 [==============================] - 61s 13ms/step - loss: 5.9236 - acc: 0.0742 - val_loss: 5.8028 - val_acc: 0.0803
Epoch 4/40
4785/4785 [==============================] - 62s 13ms/step - loss: 5.7531 - acc: 0.0692 - val_loss: 5.6311 - val_acc: 0.0852
Epoch 5/40
4785/4785 [==============================] - 61s 13ms/step - loss: 5.4919 - acc: 0.0679 - val_loss: 5.5386 - val_acc: 0.0852
Epoch 6/40
4785/4785 [==============================] - 61s 13ms/step - loss: 5.1622 - acc: 0.0644 - val_loss: 5.4258 - val_acc: 0.0767
Epoch 7/40
4785/4785 [==============================] - 61s 13ms/step - loss: 4.7807 - acc: 0.0658 - val_loss: 5.3620 - val_acc: 0.0788
Epoch 8/40
4785/4785 [==============================] - 61s 13ms/step - loss: 4.3447 - acc: 0.0602 - val_loss: 5.4315 - val_acc: 0.0803
Epoch 9/40
4785/4785 [==============================] - 61s 13ms/step - loss: 3.9888 - acc: 0.0706 - val_loss: 5.3530 - val_acc: 0.0732
Epoch 10/40
4785/4785 [==============================] - 61s 13ms/step - loss: 3.5596 - acc: 0.0748 - val_loss: 5.4919 - val_acc: 0.0874
Epoch 11/40
4785/4785 [==============================] - 61s 13ms/step - loss: 3.1518 - acc: 0.0865 - val_loss: 5.8216 - val_acc: 0.0859
Epoch 12/40
4785/4785 [==============================] - 61s 13ms/step - loss: 2.7773 - acc: 0.1007 - val_loss: 5.9618 - val_acc: 0.0888
Epoch 13/40
4785/4785 [==============================] - 61s 13ms/step - loss: 2.3614 - acc: 0.1187 - val_loss: 5.9990 - val_acc: 0.1044
Epoch 14/40
4785/4785 [==============================] - 61s 13ms/step - loss: 2.0447 - acc: 0.3204 - val_loss: 5.8666 - val_acc: 0.1768
Epoch 15/40
4785/4785 [==============================] - 61s 13ms/step - loss: 1.6420 - acc: 0.5755 - val_loss: 6.2635 - val_acc: 0.1839
Epoch 16/40
4785/4785 [==============================] - 61s 13ms/step - loss: 1.2617 - acc: 0.6798 - val_loss: 5.9429 - val_acc: 0.2188
Epoch 17/40
4785/4785 [==============================] - 61s 13ms/step - loss: 1.0073 - acc: 0.7565 - val_loss: 6.3432 - val_acc: 0.2166
Epoch 18/40
4785/4785 [==============================] - 61s 13ms/step - loss: 0.8537 - acc: 0.7879 - val_loss: 6.4971 - val_acc: 0.2109
Epoch 19/40
4785/4785 [==============================] - 61s 13ms/step - loss: 0.6766 - acc: 0.8362 - val_loss: 6.5305 - val_acc: 0.2244
Epoch 20/40
4785/4785 [==============================] - 61s 13ms/step - loss: 0.6171 - acc: 0.8514 - val_loss: 7.0039 - val_acc: 0.2038
Epoch 21/40
4785/4785 [==============================] - 61s 13ms/step - loss: 0.5570 - acc: 0.8646 - val_loss: 7.1340 - val_acc: 0.2116
Epoch 22/40
4785/4785 [==============================] - 61s 13ms/step - loss: 0.4571 - acc: 0.8959 - val_loss: 6.9157 - val_acc: 0.2116
Epoch 23/40
4785/4785 [==============================] - 61s 13ms/step - loss: 0.3940 - acc: 0.9099 - val_loss: 7.7080 - val_acc: 0.2095
```

Figure 8: Changes in loss value and accuracy during training

As shown in Figure 8, the loss value of the model gradually decreased in the first 13 training sessions, but the trend of increasing accuracy was very small. After completing the 13th training, the accuracy of the model was only 11.87%, but after the 15th training, the accuracy of the model reached 57.55%. This is because the sparseness of the data increases the model's convergence. The time required, although the computational overhead is increased, this method effectively avoids the overfitting of the training results, making the model more versatile.

batch_size is the number of batches that the training is divided into, and shuffle indicates whether to shuffle the samples in the next training to prevent overfitting. The input of validation_data is the validation set. The model trained in this way has high versatility. Even if the test set is resampled by adjusting the sampling parameters of the test set, the new test set can still achieve high accuracy in the model obtained by the previous training.

Reference

[1]. R Brunelli, T Poggio. Face recognition: features versus templates [J]. IEEE Trans. PAMI. 1993, 15(10):1042-1052.

[2]. Bengio Y., Simard P. Learning long-term dependencies with gradient descent is difficult [J]. IEEE Transactions on Neural Networks.1994, 2(2):1-35.

[3]. Y. Lecun, L. Bottou, Y. Bengio, et al. Gradient-based learning applied to document recognition [J]. Proceedings of the IEEE.1998, 11(11): 2278-2324.

[4]. Yang Jian. Research and application of face recognition method based on convolutional neural network [D]. Chang'an: Chang'an University, 2018