ATLANTIS PRESS

Advances in Engineering Research, volume 149

2nd International Conference on Mechanical, Electronic, Control and Automation Engineering (MECAE 2018)

Implementation of Face Recognition Based on Deep Learning Framework Caffe

Zijiang Zhu [a, *], Xiaoguang Deng [b], Yi Hu [c], Dong Liu [d] Junshan Li [e]

South China Business College of Guangdong University of Foreign Studies , Guangzhou 510545, China.

[a]zzjdwh2002@163.com, [b]94143082@qq.com, [c]444187113@qq.com, [d]78972493@qq.com, [e]lijunshan403@163.com

Keywords: Deep learning, Neural network, Caffe, Face recognition.

Abstract. At present, deep learning has been recognized as the best research direction of face recognition. This paper describes the basic principle of neural network technology, uses deep learning framework Caffe to build neural network, and realizes face recognition by experiment. It also puts forward the major issues for face recognition research by the use of neural network, which provides reference for construction and implementation of Caffe Face Recognition.

1. Introduction

Deep learning is a branch of characterization learning in machine learning, which uses unsupervised or semi-supervised feature learning and efficient algorithms for hierarchical feature extraction instead of manual feature retrieval. To date, many deep learning frameworks have been developed, such as Convolutional Neural Network (CNN), Deeper Belief Network and Recurrent Neural Network, which have been applied in computer vision, speech recognition, natural language processing and achieved good results [1]. In 2012, Hinton's team won the first place in ImageNet Image Classification Contest with deep learning, in which the accuracy rate was more than 10%. Subsequently, Google released a search engine based on image content with deep learning model, and found that the accuracy of image search has been greatly improved. In ImageNet ILSVRC competition of 2013, all the top 20 teams used deep learning techniques. In this paper, a deep learning framework Caffe is used to build a suitable neural network for face detection on Windows platform to train the available Caffe model for face recognition.

2. Neural network theory

2.1 Neural Networks.

Neural network is the recurrence of human brain visual mechanism, and the basic idea is: by some means, a specific class of objects can be extracted relatively abstract features [2]. On this basis, these extracted features can be extracted "features of features". In this way, features that are very abstract and can represent such objects can be extracted. However, a kind of objects have many different features, and two different kinds of objects may also have the same feature. Therefore, multiple features of the object need to be extracted in many ways to make the correct classification. In practice, to extract the appropriate features needs many layers of feature extraction, which is a very deep feature extraction network, that is, neural network [3]. Taking an image as an example, the main method of feature extraction is to convolve an image through a convolution kernel (which can also be considered as passing through a filter) to obtain another image (possibly accompanied by a change in image size). In essence, the image is a feature image extracted from original image. And then extract the feature image to obtain a more abstract nature feature image. If a number of different filters will be used, a number of different features can be extracted. These feature images necessarily reflect the features of the original image, but not necessarily the common features of such kind of objects. So the key question is how to ensure that these features can well show such kind of objects [4]. There

are many ways to achieve this, such as SGD (Stochastic Gradient Descent) algorithm used in this paper.

Based on this principle, many advanced neural networks have emerged. The convolutional neural network used in this paper is a further deep multi-layer neural network which extracts the features of images of different levels from the shallow to deep by convolution operation, and uses neural network training process to automatically adjust the convolution kernel parameters of the entire network, which produces the most suitable classification features unsupervisedly. Its weight sharing network structure reduces the complexity of the network model and reduces the number of weights.This advantage is more obvious when the input of the network is multi-dimensional image, so the image can be directly used as the input of the network, which avoids the complicated feature extraction and data reconstruction process in the traditional recognition algorithm.

2.2 SGD algorithm principle [5].

One of the input samples from the classification is taken and output after passing through the filter. The output characteristic is predicted as the classification of the input $\hat{y} = f(x)$. For the set of sample $z(x,y)$, there is a loss function $l(f(x),y)$ that measures the accuracy of this filter. is related to the weight of this filter structure. Since the structure of the filter is fixed, is essentially determined by its weight, which can be written as f_w, and the loss function can also be written as $l(f_w(x),y)$. Considering from another perspective, this loss function can also be regarded as a two-dimensional function of and, so there is formula (1).

$$Q(z,w) = \ell(f_w(x),y) \tag{1}$$

The smaller the two-dimensional function is, the more appropriate the filter will be, that is, the more appropriate the filter weights will be. To be more accurately, it is the weight of this layer is more appropriate. However, is a two-dimensional function, it is obviously unreasonable to look for a set of input to change and to find out the minimum value of , so another approach is needed to modify to make be approximation minimum. Given that the gradient of Q(z, w) at point (z, w) ∇Q is the fastest Q vector at this point, negative gradient -∇Q is necessarily the fastest declining vector. And because is an input sample, adjusting here is pointless and needs adjusting. For convenience, the following 鋈? Q refer to the component of direction. Therefore, on this basis, at least adjust to this direction, see formula (2).

$$w_{t+1} = w_t - \gamma \frac{1}{n}\sum_{i=1}^{n} \nabla_w Q(z_i, w_t) \tag{2}$$

The above formula (2) is Gradient Descent. Negative gradients of are required to all the points of the current input z_i and weight w_t groups. The average value is multiplied by coefficient and is taken as the offset factor of w_t. This obviously makes it possible to make to a more suitable value. However, the amount of computation is relatively large, and a set of z_t can be selected randomly from all the current inputs to obtain gradients to simplify the process[6]. This is the random gradient descent method SGD algorithm:

$$w_{t+1} = w_t - \gamma_t \nabla_w Q(z_t, w_t) \tag{3}$$

In Caffe framework, use SGD algorithm for driving:

$$w_{t+1} = \mu w_t - \alpha \nabla_w Q(z_t, w_t) \tag{4}$$

However, the fact is that some input deviations are large, and the gradient directions may be different for each sample. Therefore, if one sample is taken at a time to update, the scattered gradient directions may lead to that the network cannot converge[7]. So when real training, each training at the same time, test multiple samples. In Caffe framework, in a training, the number of test samples is determined by parameter batch_size.

3. Deep Learning Framework Caffe

It is very difficult to realize a full neural network. Deep learning framework can greatly improve efficiency in building a neural network. In this paper, deep learning framework Caffe of BVLC is chosen and Microsoft-Caffe on Windows platform is used to develop. Caffe is the most commonly-used one in open source deep learning framework.

In Caffe, the neural network consists of multiple layers. There are many types of these Layers, which are responsible for a variety of different functions. Parameters can be used to adjust Layer functions. The inputs and outputs of Layer are generally Blob, and data flow between layers is in the form of Blob. The filter shall correspond to a Layer in Caffe (a Layer often contains one or more filters). The feature images generated by the filter shall correspond to Blob in Caffe. Build a network of multiple Layer, and then adjust the parameters, feature extraction can be achieved.

The main Layed their parameters of Caffe are as follows: 1) Data layer: Entrance of data to be input to the network. This layer often carry out some data preprocessings, such as subtracting the mean, normalization, mirroring, reduction, etc. These features can be configured by parameters. Generally use Data layer, this type of Layer reads LevelDB or LMDB forms, but there is data layer which directs read image form. 2) Visual layer: mainly is Convolution layer and Pooling layer. The former is the core of convolutional neural network and also the main core of feature extraction through convolutional arithmetic. The latter is mainly used to reduce the computational complexity and data dimension, and achieve the effect by sampling the largest maximum and mean filter. 3) Activation layer: Conduct function transformation to Blob's data (that is, the feature graph group). The function is to make certain features be more obvious. 4) Inner Product layer: Convert input to one-dimensional variables, especially when converting network data to category labels; SoftmaxWithLoss layer and Accuracy layer are all reference values when outputs are taken as measure training and test results.

4. Face recognition using Caffe

It is very difficult to realize a full neural network. Deep learning framework can greatly improve efficiency in building a neural network. In this paper, deep learning framework Caffe of BVLC is chosen and Microsoft-Caffe on Windows platform is used to develop. Caffe is the most commonly-used one in open source deep learning framework.

For network training, the choice of data sets is very important. The database does not provide a special test set, which needs to be divided. There is no explicit requirement on the ratio of training set to test set. In general, it is sufficient to ensure that the amount of data in both sets is enough, and the amount of data in each type is enough to meet the requirements [8].

4.1 Pretreatment of face data.

Ace data pretreatment is very important, which can be divided into face pretreatment and general image pretreatment. Face pretreatment includes face detection, face alignment and face cutting [9]. Face detection refers to the detection of the part of the face in the image. Face alignment refers to align all the faces in all the images of the data set to the same angle and cut face part as shown in Fig. 1 This can be done with FaceTools, a Python-implemented tool (by RiweiChen: https:// github. com/ RiweiChen/FaceTools).

Fig. 1 From left to right: original image, face detection, face alignment, face cutting

The general pretreatment of images is mainly in the following situations:

1) Convert images to LevelDB or LMDB forms. Although Caffe can read image files directly through ImageData layer, it consumes a very large amount of IO. Generally, it shall be formatted firstly and then be read in with Data layer.

2) Subtract the image file from the dataset mean. This work is mainly to make sample pixels fall near 0, reducing the amount of computation. And some initialization algorithms require that the input data have a mean value of zero. In some special cases, it also prevents certain network weights from becoming #QNAN (Infinity).

3) Normalized. Normalizing the pixel values to [0, 1) interval by setting the scale parameter at Data layer to be 0.00390625 (ie 1/256), which in some cases can prevent some of the network's weights from becoming #QNAN.

4) Change size. In Caffe training image files, each image requires the same size. resize_height and resize_width can be set when converting images into LevelDB/LMDB to modify the size[2].

5) Convert to grayscale. In most classifications, color is not an important feature, but it consumes more computation. If RGB image is converted to grayscale, three channels can be converted to one channel, which can reduce the amount of computation.

6) Cut. During training, image of a certain size is cut randomly from the sample image (but in the test, the image is cut out from the middle of the sample image), and the purpose is also to increase the amount of data.

4.2 Neural network structure.

Since this is a simple dichotomizer, features can be extracted without the need for a particularly complex network[10]. Reference Caffe routines, build a face recognition network, as shown in Fig. 2.

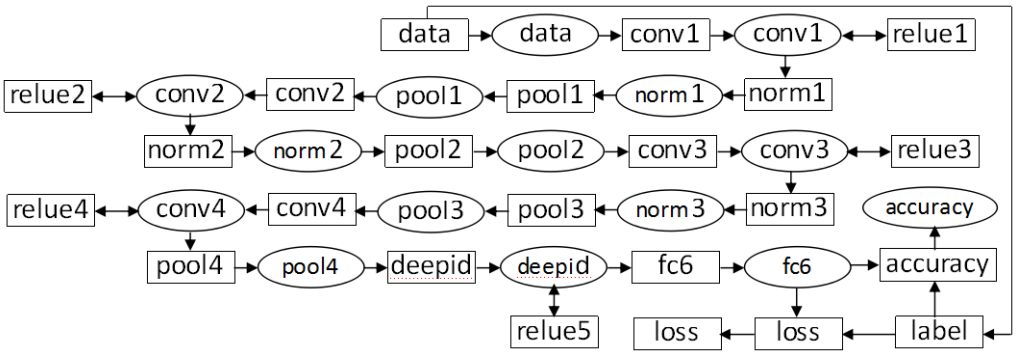

Fig. 2 Neural network structure

4.3 Neural network settings.

Building a neural network is very difficult and requires a lot of time. In general, it can be continuously modified on networks that have achieved higher accuracy, followed by training[11]. In order to speed up the training rate, the already trained Caffe model can be loaded for fine tuning.When using Caffe for neural network training, the most important setting is batch_size parameters of Data layer in the network file. The experiment results which show the training results of LeNet on MNIST data set are shown in Table 1 below.

Table 1 The impact on training results after changing batch

Batch_Size	5000	1000	500	100	50	10
Total Epoches	200	200	200	200	200	200
Total Iterations	1999	9999	19999	99999	199999	999999
Time of 200 Epoches	1	1.16	1.38	3.016	5.027	13.773
Achieve 0.99 Accuracy at Epoches	-	135	78	45	24	9
Time of Achieve 0.99 Accuracy	-	2.12	1.48	1.874	1.7	1.729
Best Validation Score	0.015	0.01	0.01	0.009	0.0098	0.01
Best Score Achieved at Epoches	182	198	100	111	38	51
Best Test Score	0.014	0.01	0.01	0.008	0.0083	0.008
Final Test Error(200 epoches)	0.013	0.01	0.01	0.009	0.0082	0.008

It can be seen from Table 1: Adding batch_size can make the network training be more stable and the number of iterations needed be less, but the time to complete one iteration be longer; reducing batch_size may lead to the wrong gradient declining, reduce the network training stability and make the number of iterations needed be more, but the time to complete one iteration be shorter. When batch_size is increased to a certain extent, the product of the iteration times and the iteration time reaches the minimum, and batch_size is the most reasonable.

In addition, parameter settings of base_lr, gamma and momentum in solver.prototxt are also very important. base_lr and momentum are respectively correspond and of SGD. Learning rate cannot be

chosen too much, otherwise it will lead to some parameters quickly toward infinity. Because it would cause the gradient of offset in the direction be too large, causing it to shift beyond the pole to a larger value than itself. Then repeat this process to make it rapidly tend to infinity, in order to make the network be in an unusable state. Gamma parameter is to adjust base_lr when each stepsize is achieved under the lr_policy: "step". Because after a certain degree of iteration, it is difficult for to achieve better performance at a higher learning rate 伪, which requires a smaller learning rate. Finally, momentum parameter is related to the amount of computation, generally taking 0.5 to 0.9. When adjusting base_lr to be smaller, it shall be adjusted accordingly multiples.

5. Experiment results

Neural networks are usually used for face recognition in the way of classification. In fact, neural network can be used in face recognition. Images can be converted into "face" and "non-face" to be identified. After training the corresponding caffe model, each scale and each possible sliding window can be classified. The windows which are classified into "face" are the detected face areas. This set of faces uses 1570 photos in Dataset Face under ImageNet, a non-face set uses a non-face set provided by MITEx and a total set of seven different Datasets randomly selected in ImageNet.

1) Non-Face sets select from MITEx. All the data is converted to grayscale. The experiment results are shown in Fig. 3 After 4000 times of iteration, the accuracy is over 90%.

```
solver.cpp:244]     Train net output #0: loss = 0.23852 (* 1 = 0.23852 loss)
sgd_solver.cpp:106] Iteration 4006, lr = 0.001
solver.cpp:228] Iteration 4007, loss = 0.180454
solver.cpp:244]     Train net output #0: loss = 0.180454 (* 1 = 0.180454 loss)
sgd_solver.cpp:106] Iteration 4007, lr = 0.001
solver.cpp:228] Iteration 4008, loss = 0.119537
solver.cpp:244]     Train net output #0: loss = 0.119537 (* 1 = 0.119537 loss)
sgd_solver.cpp:106] Iteration 4008, lr = 0.001
solver.cpp:228] Iteration 4009, loss = 0.125092
solver.cpp:244]     Train net output #0: loss = 0.125092 (* 1 = 0.125092 loss)
sgd_solver.cpp:106] Iteration 4009, lr = 0.001
solver.cpp:337] Iteration 4010, Testing net (#0)
solver.cpp:404]     Test net output #0: accuracy = 0.938125
solver.cpp:404]     Test net output #1: loss = 0.171779 (* 1 = 0.171779 loss)
solver.cpp:228] Iteration 4010, loss = 0.290778
solver.cpp:244]     Train net output #0: loss = 0.290778 (* 1 = 0.290778 loss)
sgd_solver.cpp:106] Iteration 4010, lr = 0.001
solver.cpp:228] Iteration 4011, loss = 0.0823777
solver.cpp:244]     Train net output #0: loss = 0.0823777 (* 1 = 0.0823777 loss)
sgd_solver.cpp:106] Iteration 4011, lr = 0.001
solver.cpp:228] Iteration 4012, loss = 0.0534595
```

Fig. 3 Screenshot of the experimental results a

2) Non-Face sets select 7 different photos randomly from ImageNet which are firstly converted to grayscale. The experiment results are shown in Fig. 4 After 8000 times of iteration, the accuracy is over 99.9%.

```
solver.cpp:228] Iteration 8007, loss = 2.68966e-006
solver.cpp:244]     Train net output #0: loss = 2.68971e-006 (* 1 = 2.68971e-006 loss)
sgd_solver.cpp:106] Iteration 8007, lr = 0.001
solver.cpp:228] Iteration 8008, loss = 5.20132e-005
solver.cpp:244]     Train net output #0: loss = 5.20133e-005 (* 1 = 5.20133e-005 loss)
sgd_solver.cpp:106] Iteration 8008, lr = 0.001
solver.cpp:228] Iteration 8009, loss = 0.00362401
solver.cpp:244]     Train net output #0: loss = 0.00362401 (* 1 = 0.00362401 loss)
sgd_solver.cpp:106] Iteration 8009, lr = 0.001
solver.cpp:337] Iteration 8010, Testing net (#0)
solver.cpp:404]     Test net output #0: accuracy = 0.999375
solver.cpp:404]     Test net output #1: loss = 0.00336357 (* 1 = 0.00336357 loss)
solver.cpp:228] Iteration 8010, loss = 8.50158e-006
solver.cpp:244]     Train net output #0: loss = 8.50162e-006 (* 1 = 8.50162e-006 loss)
sgd_solver.cpp:106] Iteration 8010, lr = 0.001
solver.cpp:228] Iteration 8011, loss = 1.03045e-005
solver.cpp:244]     Train net output #0: loss = 1.03045e-005 (* 1 = 1.03045e-005 loss)
sgd_solver.cpp:106] Iteration 8011, lr = 0.001
```

Fig. 4 Screenshot of the experimental results b

3) Non-Face sets select 7 different photos randomly from ImageNet to preserve the third channel of RGB. The experiment results are shown in Fig. 5 After 8000 times of iteration, the accuracy is also 99.9%.

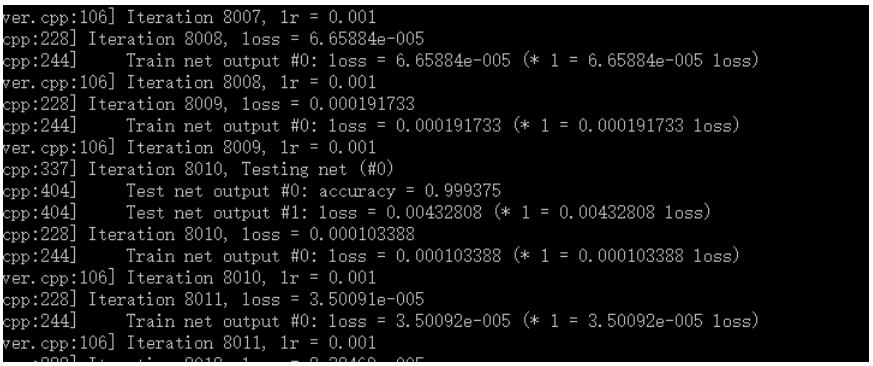

Fig. 5 Screenshot of the experimental results c

6. Conclusion

In this paper, face recognition is achieved through building a neural network with deep learning framework Caffe. From the experimental results, it achieves the expected goal. In the specific experiment, we should pay attention to the following three questions[12]: First, deep networks consume large amounts of computing and memory resources and require very long training time, if the situation allows, smaller networks shall be used; Second, overly simple networks may be difficult to extract the appropriate classification features. Select or build the appropriate network based on the situation; Third, the number of data setting samples shall be sufficient or it is easy to fall into the case of over fitting. As it can be seen that to choose the right neural network is the key in deep learning.

Acknowledgements

This article has received the support of the Characteristics innovation project of colleges and universities of Guangdong Province (Natural Science), 2016, No.2016KTSCX182; and also received the support of the Youth Innovation Talent Project of colleges and universities of Guangdong Province,2016, No. 2016KQNCX230.

References

[1] L. Bottou:*Stochastic Gradient Descent Tricks Neural Networks Tricks of the Trade*(Springer Berlin Heidelberg, Germany 2012),p.421-436.
[2] G.E. Hinton, S. Osindero,Y.W. Teh: Neural Computation, Vol. 18 (2006) No.7,p.1527-1554.
[3] Y. Bengio: Foundations and Trends® in Machine Learning, Vol. 2(2009) No.7,p.1-127.
[4] G. Hinton: Momentum, Vol. 9(2010) No.1,p.926.
[5] O.M. Parkhi, A. Vedaldi, A. Zisserman: Proceedings of the British Machine Vision, Vol. 1(2015) No.3,p.6-12.
[6] Y. Sun, X.D. Wang, X. Tang: *26th IEEE International Conference on Computer Vision and Pattern Recognition*(Portland, Oregon, USA, Jun 23-28, 2013), Vol. 1, p.1891-1898.
[7] L. Best-Rowden, H. Han and C. Otto: IEEE Transactions on Information Forensics and Security, Vol. 9(2014) No.12, p.2144–2157.
[8] M.Z. Lin:*Face Recognition Based on Depth Learning*(MS.,Dalian University of Technology, China 2013), p.16-45.
[9] Z.J. Sun, X. Lei, Y.M. Xu: Journal of Computer Applications, Vol. 29(2012) No.8, p. 2806-2810.
[10] Y. Liu, S.S. Zhou and Q.C. Chen: Pattern Recognition, Vol. 44(2011) No.10, p.2287–2296.
[11] F. Schroff, D. Kalenichenko and J. Philbin: *IEEE Conference on Computer Vi-sion and Pattern Recognition*(Salt Lake City, Utah, USA, June 17-22, 2015), Vol. 1, p.815–823.
[12] P.N. Belhumeur, D.W. Jacobs and D.J. Kriegman: IEEE Transactions on Pattern Nalysis and Machine In-Telligence, Vol. 35(2013) No.12, p.2930–2940.