Image Classification of Fashion-mnist Data Set Based on VGG Network

Chao Duan[a], Panpan Yin[b], Yan Zhi[c], Xingxing Li[d]

Department of electronics and information engineering, Guangzhou College of Technology and Business, Foshan, 528138, China

[a]MDL0857a@126.com, [b]abc0604@126.com, [c]sparks666888@163.com, [d]lixingxing_Edooy@163.com

Keywords: Image Classification, VGG, Fashion-mnist, Batch Normalization

Abstract: Image classification is the core of computer vision, which plays a pioneer role in other visual tasks. In this paper, we use VGG-11 network to classify Fashion-mnist data sets. We use several consecutive 3×3 convolution cores to replace the larger convolution cores (11×11, 7×7, 5×5) in AlexNet. For a given receptive field, small cumulative convolution kernel is better than large convolution kernel. Multilayer nonlinearity layer increases network depth to ensure learning more complex patterns, and the cost is relatively small. At the same time, batch normalization layer is added after each pooling layer, which makes it easier to train effective models by standardizing input data to make the distribution of each feature similar. Finally, the classification accuracy of this paper on Fashion-mnist Data Set is 91.5%.

1. Introduction

The research of object classification and detection is the cornerstone of the whole computer vision research, and is the basis of solving other complex visual problems such as tracking, segmentation, scene understanding and so on. In order to analyze and understand the real complex scene automatically, it is necessary to determine what objects exist in the image (classification problem) or what objects exist in the image (detection problem). In view of the importance of object classification and detection in the field of computer vision, it is undoubtedly of great theoretical and practical significance to study robust and accurate object classification and detection algorithms. With the wide application of in-depth learning, many well-known in-depth networks have emerged. LeNet [1] is an early convolution neural network used to recognize handwritten digital images. The basic unit of the convolution layer is the convolution layer followed by the maximum pooling layer: the convolution layer is used to identify spatial patterns in the image, such as lines and objects, while the later maximum pooling layer is used to reduce the sensitivity of the convolution layer to position. Convolutional blocks consist of two such basic units stacked repeatedly. In a convolution layer block, each convolution layer uses a 5×5 window, and sigmoid activation function is used in the output. The shape of convolution window in the first layer of AlexNet is 11×11. Because the height and width of most images in ImageNet are more than 10 times higher than that of MNIST images, objects in ImageNet images occupy more pixels, so larger convolution windows are needed to capture objects. The shape of convolution window in the second layer is reduced to 5×5, and then 3×3 is used. In addition, after the first, second and fifth convolution layers, the maximum pooling layer with a window shape of 3×3 and a step of 2 is used. Moreover, AlexNet uses dozens of more convolution channels than LeNet [1] does. AlexNet changed the sigmoid activation function to a simpler ReLU activation function. On the one hand, the calculation of ReLU activation function is simpler, for example, it does not have power calculation in sigmoid activation function. On the other hand, the ReLU activation function makes the model easier to train under different parameter initialization methods. The VGG [2] network used in this paper can construct depth model by reusing simple foundation blocks. Several convolution layers with the same filling 1 and window shape 3×3 are successively used, followed by a maximum pooling layer with 2 steps and 2×2 window shape. The convolution layer keeps the input height and width unchanged, while the pooling layer halves it.

2. VGG network

2.1 VGG Principle

VGGNet uses all 3×3 convolution cores and 2×2 pooling cores to improve performance by deepening the network structure. The increase of network layers will not lead to the explosion of parameters, because the parameters are mainly concentrated in the last three full connection layers. Meanwhile, the cascade of two 3×3 convolution layers corresponds to a 5×5 convolution layer, and three 3×3 convolution layers correspond to a 7×7 convolution layer, that is, the receptive field of three 3×3 convolution layers corresponds to a 7×7 convolution layer. However, the convolution layer parameters of 3×3 are only about half of that of 7×7. At the same time, the former can have three non-linear operations, while the latter has only one non-linear operation, which makes the former more capable of learning features. Using 1×1 convolution layer to increase linear transformation, the number of output channels has not changed. Other uses of 1×1 convolution layer are mentioned here. 1×1 convolution layer is often used to extract features, i.e. multi-channel features are combined to condense the output of larger or smaller channels, while the size of each picture remains unchanged. Sometimes 1×1 convolutional neural network can also be used to replace the full connection layer.

2.2 Batch Normalization

Batch normalization [3] is proposed to meet the challenges of in-depth model training. In model training, batch normalization utilizes the mean and standard deviation in small batches to continuously adjust the intermediate output of the neural network, so that the output value of the whole neural network in the middle of each layer is more stable. Usually the batch normalization layer is placed between the affine transformation and the activation function in the full connection layer. Let the input of the full connection layer be mu, the weight parameter and the deviation parameter are w and b, and the activation function is Φ. Let the batch normalization operator be BN. Then, the output of the full connection layer using batch normalization is:

$$\Phi(BN(x)) \tag{1}$$

The batch normalized input x is transformed by affine transformation:

$$x=w\times u+b \tag{2}$$

Considering a small batch consisting of m samples, the output of affine transformation is a new small batch $\beta=\{x(1),\ldots,x(m)\}$. They are inputs to the batch normalization layer. For any sample $x^i \in R^i$ in small batch β, the output of batch normalization layer is also d-dimensional vector:

$$y^i = BN(x^i) \tag{3}$$

It can be obtained from the following steps. Firstly, the mean and variance of small batch B are calculated.

$$\mu_b \leftarrow \frac{1}{m}\sum_{i=1}^{m} x^i \tag{4}$$

$$\delta_\beta^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x^i-\mu_\beta)^2 \tag{5}$$

The square calculation is based on the square of the elements. Next, standardize $x^{(i)}$ by element prescription and by element division:

$$\hat{x}^i \leftarrow \frac{x^{(i)}-\mu_\beta}{\sqrt{\delta_\beta^2+\epsilon}} \tag{6}$$

Here $\epsilon>0$ is a very small constant, guaranteeing that the denominator is greater than 0. On the basis of the above standardization, two learnable model parameters, tensile parameter γ and migration parameter β, are introduced into the batch normalization layer. These two parameters have the same shape as $x^{(i)}$ and are d-dimensional vectors. They are calculated by element multiplication ($\odot$) and addition respectively with $x^{(i)}$: w

$$y^i \leftarrow r\odot\hat{x}^i + \beta \tag{7}$$

Therefore, the formula obtains the batch normalized output y^i of $\hat{x}^i$.

2.3 Network Structure

The basic structure of VGG-11 network is shown in the figure. One or two 3×3 convolution cores are followed by a pool layer, followed by a Batch Normalization layer. In this paper, the BN layer is added after the pool layer, not before the pool layer after the convolution layer, because the network is shallow, the phenomenon of gradient disappearance or gradient explosion is not easy to occur, and it is added after the pool layer. The calculation will be reduced.

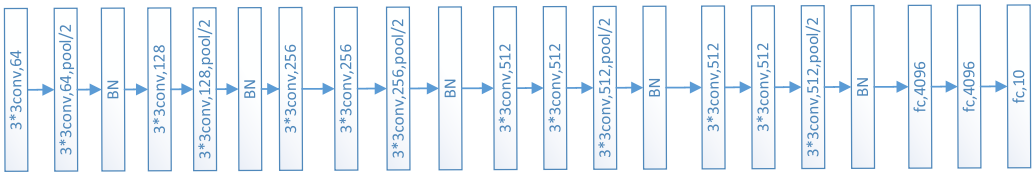

Figure 1. VGG11 network architecture

3. Experiments

3.1 Data Set

Fashion-MNIST includes 10 categories: t-shirt, trouser, pullover, dress, coat, sandal, shirt, sneaker, bag and ankle boot. The number of images of each category in training and test sets is 6,000 and 1,000 respectively. Because there are 10 categories, the number of samples in the training set and the test set is 60,000 and 10,000, respectively, as shown in Figure 2.

Figure 2. Fashion-MNIST DataSet

3.2 Experimental results

In this paper, classical LeNet [1], AlexNet, NiN [4], DenseNet [5] and the most primitive VGG-11 [2] networks are used for experimental comparison. The algorithms are iterated for five cycles on the training set. The experimental results are as follows:

Table.1. Deep network experiment comparison

Method	Accuracy
LeNet	0.753
AlexNet	0.871
NiN	0.667
DenseNet	0.864
VGG-11	0.905
Ours	0.915

The experimental results show that the BN layer is very effective on Fashion-mnist after the pooling layer. Especially for the shallow neural network such as VGG-11, the accuracy is 1% higher than that of the original VGG-11.

4. Conclusion

VGG network constructs depth model by reusing simple foundation blocks. Several convolution layers with the same window shape of 3×3 are successively used, followed by a maximum pooling layer with the window shape of 2×2. The convolution layer keeps the input height and width unchanged, while the pooling layer halves it. In Fashion-mnist dataset, adding BN layer after pooling layer is very effective.

Acknowledgments

This work was financially supported by fund project, that is, Guangzhou Institute of industry and commerce college level research project in 2019 "Research and design of S bandradio frequency front-end" KA201939.

References

[1] Lecun Y, Bottou L, Bengio Y, et al. Gradient-based learning applied to document recognition [J]. Proceedings of the IEEE, 1998, 86(11):2278-2324.

[2] Smirnov E, Timoshenko D, Andrianov S, et al. Comparison of Regularization Methods for ImageNet Classification with Deep Convolutional Neural Networks [J]. AASRI Procedia, 2014:89-94.

[3] Ioffe S, Szegedy C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift [J]. International conference on machine learning, 2015:448-456.

[4] Chang J, Chen Y. Batch-normalized Maxout Network in Network [J]. arXiv: Computer Vision and Pattern Recognition, 2015.

[5] Huang G, Liu Z, Der Maaten L V, et al. Densely Connected Convolutional Networks [J]. Computer vision and pattern recognition, 2017:2261-2269.