South China University of Technology

The Experiment Report of *Deep Learning*

SCHOOL: SCHOOL OF SOFTWARE ENGINEERING

SUBJECT: SOFTWARE ENGINEERING

Author:
Yihan Zheng, Qi Chen and Jing Liu

Supervisor:
Mingkui Tan

Student ID:
201720145105
201720144924
201720145099

Grade:
Graduate

December 24, 2017

Image Classification based on Convolutional Neural Network

***Abstract*—Convolutional neural network (CNN or ConvNet) is a class of deep, feed-forward artificial neural networks that has successfully been applied to analyzing image recognition. In this experiment, we train a CNN model to classify images in CIFAR-10 dataset. The experimental results show that CNN can learn features and classify the images effectively.**

I. Introduction

CONVOLUTIONAL neural network is inspired by biological processes in which the connectivity pattern between neurons is inspired by the organization of the animal visual cortex. Individual cortical neurons respond to stimuli only in a restricted region of the visual field known as the receptive field. The receptive fields of different neurons partially overlap such that they cover the entire visual field.

A CNN consists of an input and an output layer, as well as multiple hidden layers. The hidden layers of a CNN typically consist of convolutional layers, pooling layers and fully connected layers.

A. Convolutional layer

A fully connected feed forward neural networks can be used to learn features as well as classify data, but it is not practical to apply this architecture to images. A very high number of neurons would be necessary, even in a shallow architecture, due to the very large input sizes associated with images, where each pixel is a relevant variable.

Convolutional layer applies a convolution operation to the input, passing the result to the next layer. Each convolutional filter is shared in the whole feature map and processes data only for its receptive field. In this way, the convolution operation brings a solution to high computational complex problem as it reduces the number of parameters, which allows the network to be deeper with fewer parameters.

B. Pooling layer

Convolutional neural network may include pooling layer, which combines the outputs of neuron clusters at one layer into a single neuron in the next layer.For example, max pooling uses the maximum value from each of a cluster of neurons at the prior layer.

C. Fully connected layer

After several convolutional and max pooling layers, the high-level feature map extracted by the prior layers will be fed into fully connected layers. Fully connected layer connects every neuron in one layer to every neuron in another layer. It is in principle the same as the traditional multi-layer perceptron neural network (MLP).

II. Methods and Theory

In this experiment, we use the model VGG16[3] to do the classification. The input of the VGG16 is a fixed-sized 32×32 RGB image. The preprocessing we do is to random crop the images padding each edge with 4 pixels and substract the mean RGB value for each pixel. The image is passed through a stack of convolutional layers, where the filters with a very small receptive field: 3×3. The convolution stride is fixed to 1 pixel. Spatial pooling is carried out by five max-pooling layers, which follow some of the convolutional layers. Max-pooling is performed over a 2×2 pixel window, with stride 2.

A stack of convolution layers is followed by a fully-connected layer, which performs 10 way CIFAR10 classification. The final layer is the softmax layer. Note that all hidden layers are equipped with the rectification non-linearity(ReLU)[2].

III. Experiments

A. Dataset

The CIFAR-10 dataset[1] consists of 60000 32×32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. And the classes in CIFAR-10 are completely mutually exclusive.

B. Implementation

All the experimental code are written in python, with the help of torch package.

1) Create the data loader, which preprocesses and loads CIFAR-10 dataset.
2) Build convolutional neural network VGG16, which is illustrated in the last section.
3) Instantiate torch.optim.SGD to get the optimizer. The learning rate of the optimizer is firstly defined as 0.01, and decaies with the iterative process.
4) Instantiate VGG16 to get the net.
5) Input data into the network with batch size=128, and do a forward propagation to get predict target.
6) Use torch.nn.CrossEntropyLoss to calculate the loss between predict target and ground truth target.
7) Calculate gradient using loss.backward and optimize the net using optimizer.step.
8) Calculate the accuracy.
9) Repeat step 5 to 8 until all data are inputed to the net.
10) Define training epoch = 150, repeat the training process from step 5 to 9
11) Save the best model as a .pkl file.

C. Results

In this section, we illustrate the experimental results of VGG16. Figure 1 shows the loss descent process during the training. In the first 15 epochs, the loss declines rapidly. After that, the loss curve is oscillating and can not achieve the best performance. In the ninetieth epoch, the learning rate reduce from 0.01 to 0.001. The loss falls again and finally settles in 0.3685.

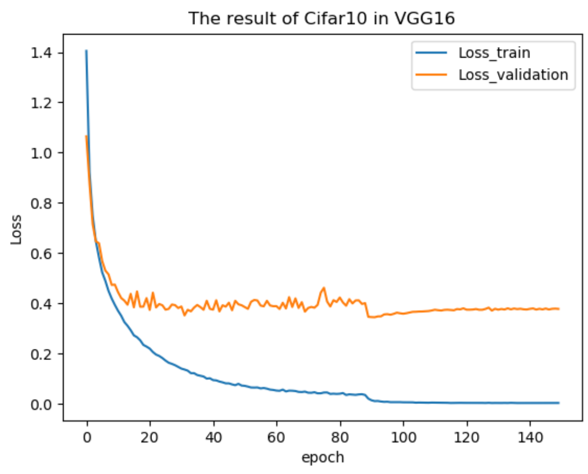

Fig. 1: The loss descent process of the model.

Figure 2 illustrates the process of rising accuracy. And the final accuracy of the model is reach to 92.25%.

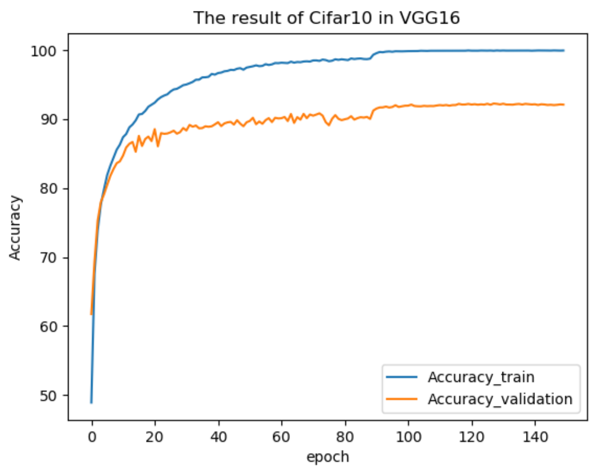

Fig. 2: The accuracy of the model.

IV. Conclusion

In this experiment, we train a CNN model to classify images in CIFAR-10 dataset. The experimental results show that CNN can learn features and classify the images effectively.

References

[1] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. 2009.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[3] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.