Assignment 2: Convolutional Neural Networks for CIFAR-10 multiclass image classification, Deep Learning Fundamentals, 2020

Yumin Cao, a1754926
The University of Adelaide

Abstract

This assignment is to classify the well known CIFAR - 10 dataset using a specific Convolutional Neural Network (CNN) algorithm called Visual Geometry Group (VGG) net. The submission will take the Conference on Computer Vision and Pattern Recognition (CVPR) form.

1. Introduction

The CIFAR-10 is a widely-used data set for CNN beginners, it consists of 60000 32x32 color images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images in it. The assignment aims at giving the background of the CNN's algorithms and compare them, then classify these images and getting a high accuracy. VGG Net is the name of a pre-trained convolutional neural network (CNN) invented by Simonyan and Zisserman from Visual Geometry Group (VGG) at University of Oxford in 2014[1] and it was able to be the 1st runner-up of the ILSVRC (ImageNet Large Scale Visual Recognition Competition) 2014 in the classification task. VGG net is used for classification. However, the traditional VGG net has some drawbacks like the convergence speed is slow and the occupation of CPU is huge. So we optimize the VGG net to suit the data set. For the CPU time problem, we reduce the parameters during the process by using the dropout function, which can also prevent the over-fitting problem. We also choose a suitable learning rate as well as epochs. For the convergence speed problem, we use the batch normalization to keep the output of each layer in the normal form. Also we replace the relu function with the leaky-relu function. At last, a conclusion is given to analyze the benefits and drawbacks of VGG with its accuracy performance.

2. Background

Before the CNN came out, scientists use traditional machine learning method like pre-processing, feature selection, classifier to train their model. However, the traditional methods need lots of hand-crafted features and are hard to get the global minimum.

To solve these problems, scientists from England and Japan introduce the CNN as a special neural network. CNN is a hierarchical model, the input is the raw data like RGB(red, green, blue) images or audio data, and CNN processes these data by stacking the convolution, pooling, non-linear activation function, and regularization to extract the features from the raw data. After calculating the loss function, the weights in each layer will be updated to get the global optimize solution.

There are some famous CNN nets like Alex-Net, VGG Net, and ResNet (residual net)[1]. Alex Net is the first CNN net that break the limit of calculating ability and solve the vanishing gradient problem[2]. This network uses a new activation function (tanh) and drop out method to reveal that the CNN has ability to deal with large amount data and avoid over fitting, and it replace the CPU with GPU to accelerate training time. Then VGG brings the innovation with using small size convolutional kernel[1] to increase the ability of learning more complex features. This method can also make the input size (w*h) steady decrease. However, as the depth goes increase, the loss signal becomes harder to convergence and even results in gradient explosion if the gradient is set too large. This will make the training error get increased. Residual Net is set to solve that problem and can reduce the cost of calculation dramatically. And it also replace the fully connected layer with the global average pooling layer.

As for activation function, there are also some commonly used function like tanh, relu, sigmoid, and others. Sigmoid function will bring the saturation effect because some points at the two sides with gradient almost equaling to 0 are hard to pass the loss to the formal layer. The tanh function will also have this problem because their shape are almost same. Relu function eliminate the saturation effect when x is more than 0, and other function like leaky relu also eliminate the saturation effect when x is less than 0, however, the hyper parameter alpha in leaky relu is hard to set.

Considering that our data set is not very big and the layers' structure is also not very deep, so I choose VGG method accompany with leaky relu function.

3. Method

Preprocessing: Download the data set using torch vision, and then transform then transform the image data into RGB form, using the dataloader function to load the data, with setting the batch size equal to 128, because we want to get more generalize optimize solution, if our batch size set too small, it is hard to get the comprehensive information of the data.

we use the standard VGG16 convolutional layer structure.

[(3,64),(64,64),(64,128),(128,128),(128,256),(256,256),(256,256),(256,512),(512,512),(512,512),(512,512),(512,512),(512,512)]

As we have discussed the back propagation (BP) in assignment1, the CNN can be seen as a shared weight and partial connected neural network, so it will also use the partial derivative and learning rate to update the weights.

$$new\ w_{j,k} = old\ w_{j,k} - \alpha . \frac{\alpha E}{\alpha w_{j,k}}$$

Using the formula we can pass the loss to the formal layer:

$$\nabla a^{l-1} = \frac{\partial J(W,b)}{\partial a^{l-1}} = (\frac{\partial z^l}{\partial a^{l-1}})^T \frac{\partial J(W,b)}{\partial z^l} = (\frac{\partial z^l}{\partial a^{l-1}})^T \delta^l$$

Then we can get the weight using:

$$\frac{\partial J(W,b)}{\partial W_{pq}^l} = \sum_i \sum_j (\delta_{ij}^l a_{i+p-1,j+q-1}^{l-1})$$

Our loss function is the cross entropy loss function, it can change the original form into probability and its formula is:

$$l_{cross\ entropy\ loss} = -\frac{1}{N} \sum_{i=1}^{N} log(\frac{e^{h_{yi}}}{\sum_{j=1}^{C} e^{h_j}})$$

The pooling method is the max pool method, which get the biggest value among the (2,2) area and output that value into new layer.

What's more, the inactive rate of dropout function has a slope and that is from 0.1 to 0.5(the possibility that being ignored). The reason is that we want to control the over fitting problem while does not loss much information at the beginning. As known that we may get very much features at the tail of the layers but not that much features at the beginning if we follow the VGG16 network.

Batch normalization is also used to accelerate the convergence speed and avoid gradient dispersion. It is similar to the saturation effect, because the output data will gradually deviate the normal distribution if we do not set the bias value to maintain it.

We also use the antitheses method to compare the method and get the parameter we want. There are three VGG networks:

VGG16: with batch normalization
with drop out function

VGG: without batch normalization
with drop out function

VGG1: without batch normalization
without drop out function

And my learning rate is in 0.001 and 0.01 based on experience.[3]

Also, we set the epoch from 2 to 4, because it is often not enough if we only train the CNN data once. And if the epoch is too big, we may face the over fitting problem and have to change the learning rate. Wei[3] mentions that the learning rate can be decrease like:

$$lr_t = lr_0/(1 + k * epoch)$$

As for the view of fully connected layer, because the input size of CIFAR-100 is 232*232*3 and it correspond view is 7*7*512. Our CIFAR-10 is 32*32*3 which is 1/7 of the CIFAR-100, so I guess the correspond view is 1*1*512.

4. Experiment

At the beginning I want to see whether there exists sample imbalance problem, then we find that each class has 5000 samples, which means there does not have sample imbalance problem.

The parameters that we want to turn are learning rate and epoch, the learning rate has two possible value: 0.001 and 0.01, a suitable learning rate is important because a large learning rate may make the loss increase and a small learning rate may be too hard to converge.

The default value of momentum is 0.9, but a common method is setting it to 0.5 and making a growth with epoch grows. Here I use the default value because my interval of epoch only is 2 to 4. The reason that the epoch should not be very big is that our VGG net is very deep and has too many parameters, it already has the potential of over fitting, so my epoch should be moderate.

And I also tried three different VGG network with same structure. To see the influence of batch normalization and drop out function.

Unfortunately, my laptop is almost break when turning the parameters, and Google jupyter notebook is even more slowly. I don't have device to support my experiment.

So I only tried VGG16, VGG, VGG1 with learning rate set to 0.001 and epoch set to 3.

The results are:

VGG16 -> accuracy 0.59

VGG -> accuracy 0.56

VGG1 -> accuracy 0.57

The worst accuracy are cat and dog with no more than 0.35 accuracy, the best accuracy are plane and truck and they both beyong 0.7 accuracy. The acccuracy of car is also very high, ranging from 0.6 to 0.7.

According to the experiment, the model seems having learned something, but not very accurate, and it performs well in truck, plane and car, but performs poor in cat and dog classification.

5. Code

https://github.com/twofatcat/Deep-learning-assign2/tree/master

6.Conclusion

Using VGG net to classify images requires a good device if calculating based on CPU, and we might need to adjust the structure of VGG constantly because one structure might performs good on one data set but not good on the other data set. The most problem that confuse me is the running speed. I tried a simplified version in my beta code with structure [(3,64),(64,64),(64,128),(128,128),(128,256),(256,256),(256,512),(512,512),(512,512),(512,512)] and the accuracy is 0.55. Which is almost same to the complex VGG16.

Probably I will use the Residual Net in the rest of my life and never touch VGG again.

References

[1] Koustubh. *ResNet, AlexNet, VGGNet, Inception: Understanding various architectures of Convolutional Networks*, https://cv-tricks.com/cnn/understand-resnet-alexnet-vgg-inception/#:~:text=While%20VGG%20achieves%20a%20phenomenal,large%20width%20of%20convolutional%20layers.
[2]Alex, Ilya, Geoffrey E. *ImageNet Classification with Deep Convolutional Neural Networks*, 2012.
[3]XiuCan. Wei. *The principle of Convolutional neural network and the practice of visualization.*