18COC102 - Coursework

Yixiao Zhang
Student ID: B810056

Experiment Description

Experiment Environment

- Hardware
  Intel Xeon(R) CPU X5550, GeForce GTX 1080 Ti, 24GB DDR4 Memory

- Software and runtime
  Ubuntu 16.04.5 LTS 64-bit, conda 4.5.4, python3.6.7, pytorch 0.4.0

Experiment Tools

- Anaconda
  Anaconda is the most popular python environment management software. In the experiment, every package is installed on a Anaconda environment. The package can be searched in Conda Cloud and can be installed by using conda install command. The command conda activate, conda env list, conda list are used for activating environment, showing the environment list and showing the package list respectively.

- NumPy and Matplotlib
  NumPy is a useful package for scientific computing with Python. It provides many classes which PyTorch and Matplotlib needed. Matplotlib is a Python 2D plotting library, it is used for show the loss plot and accuracy plot at the end of training on Jupyter Notebook.

- Pytorch
  PyTorch is a machine learning library for Python. It gives us the ability to do machine learning easily. Some API of PyTorch are used in the experiment for crate network, load the data, do the forward process, automatic backward process, compute loss and set the optimizer.

- Jupyter Notebook
  The Jupyter Notebook is a web application which allows users to create a notebook. The notebook is used for remote writing and execution(use ssh to connect remote machine and port mapping to local), split code segment, execute code step by step, record the output in each step in this experiment.

- Tensorboard
  TensorBoard is a suite of web-based visualization tools, it makes people easier to understand, debug, and optimize. TensorBoard is used for real-time monitor, while Matplotlib cannot support real-time monitor. It also can be used for compare the experiment in the same group on one graph(data in one graph are from different code file).

Dataset and Dataloader

CIFAR-10 dataset is used in this experiment which consists of 60000 colour images in 10 classes. There are 6000 images in each class. The size of each image is 32x32. The 50000 number of the training images is 50000, therefore the number of test images is 10000.

This dataset does not give the validation set, as well as I do not split the training set for two set. It may lead to overfitting, which means the training accuracy would very good but the testing result may show not as good as training.

The dataloader is showed as Listing 1. The torchvision package is used for load the CIFAR-10. From PyTorch documentation we can find that the torchvision package consists of popular datasets, model architectures, and common image transformations for computer vision. Firstly, the class torchvision.datasets.CIFAR10 is used to get the dataset. The parameters are root(dataset directory), train(training set or testing set), transform(transform images), download(download the dataset from the internet). Secondly, the class torch.utils.data.DataLoader is used to load the dataset. The parameters are trainingset(the dataset we prepared), batch size(the size of batch), shuffle(data reshuffled at every epoch), num workers (how many subprocesses to use for data loading)

Listing 1: code segment of dataloader

```
trainingset = torchvision.datasets.CIFAR10(root='./data',
                          train=True, download=True,
                          transform = transforms.ToTensor())

trainingloader = torch.utils.data.DataLoader(trainingset,
                          batch_size = batchSize, shuffle = True,
                          num_workers = workers)
```

Experiment Design

Table 1 shows that experiments are divided into 4 groups. In order to find out how different parameters influence the training process and training result, only one parameter is changed in the each group. AlexNet, LeNet, and VGG16 are used in group A, B, and C.

Table 1: Experiment design

Group	Experiment No.	Network	Learning Rate	Batch Size	Optimizer	Epoch
A	A1 A2 A3	AlexNet	0.0001 0.001 0.01	200	Adam	150
B	B1 B2 B3	LeNet	0.001	100 1000 10000	Adam	800
C	C1 C2 C3 C3	VGG16	0.001	200	Adam SGD Adagrad RMSprop	200
D	D1 D2 D3	LeNet LeNet* LeNet**	0.001	100	Adam	200

AlexNet, a convolutional neural network, was developed by Alex Krizhevesky in 2012. AlexNet contained eight layers, the first five layers are convolutional layers and the last three layers were fully connected layers. (Krizhevsky,2012) It is a simple convolutional neural network and being used in experiment group A. I implement this AlexNet by myself.

According to Singh(2015), LeNet is a great convolutional neural network for image classification task on CIFAR-10 dataset. I implement the network according to the network architecture. In

experiment group B, LeNet is used to find how batch size influences the training process. In experiment group D, LeNet* and LeNet** are designed respectively as Figure 1 shows in order to verify that the network is successfully modified and how the new network performance. The LeNet* based on LeNet, remove a pool layer from LeNet and add a convolution layer and a full connect layer. The LeNet ** based on LeNet, remove a pool layer from LeNet and add two convolution layers.

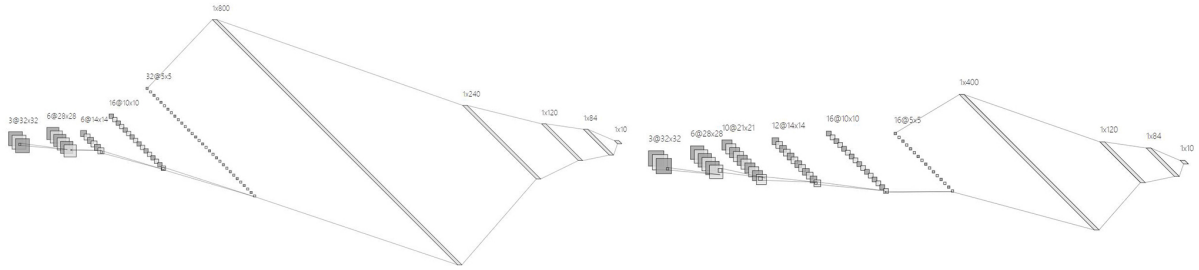

Figure 1: Network of Experiment D2(left) and Network of Experiment D3(right)

VGG16 which also called OxfordNet is a convolutional neural network developed by Visual Geometry Group. It was used to win the ILSVR (ImageNet) competition in 2014. Nowadays, it is still considered to be a good vision model. (Chollet,2016) As this network is a little bit hard to implement, the network source code is download from the Internet. The VGG16 network is used in group C to find out the performance of different optimizers.

Experiment Results

Experiment Group A

Choosing a suitable learning rate is important. Learning rate is a parameter in the learning process which controls how much the weights of the network machine adjusting. In group A, the learning rate is set as 0.0001, 0.001, and 0.01 respectively.

Table 2 and Figure 2 show the result of experiment group A. When the learning rate is too small, the gradient descent would be slow. As experiment A1, it reduces the loss lower than experiment A2 which learning rate is bigger. When the learning rate is too large, the gradient descent would overshoot the minimum. So that the experiment A3 decreased its loss very slow. The experiment A2 has a more appropriate learning rate than experiment A1 and experiment A3, which not only can quickly improve the accuracy but also get good accuracy in the final.

There is a useful and common method to set the learning rate is that reduce the learning rate with increasing of epoch. This method will help machine get a faster and more accurate result.As this task is too easy, this method is not used in the experiment.

Table 2: Results of Experiment Group A

Exp No.	Network	Learning Rate	Time Cost	Training Accuracy	Testing Accuracy
A1	AlexNet	0.0001	1948s	99.352%	99.752%
A2		0.001	1869s	99.192%	99.452%
A3		0.01	2057s	43.726%	44.312%
Other parameters: batch size=200; optimizer=Adam; epoch=150					

Figure 2: Training process of Experiment Group A (screen-shot from TensorBoard)

Experiment Group B

Batch size is a term which refers to the number of training examples loaded in one iteration. Table 3 and Figure 3 describe the training result and training process of group B.

Table 3: Results of Experiment Group B

Exp No.	Network	Batch Size	Time Cost	Training Accuracy	Testing Accuracy
B1	LeNet	100	8768s	97.112%	98.428%
B2		1000	4412s	99.532%	99.658%
B3		10000	5515s	69.690%	69.942%
Other parameters: learning rate=0.001; optimizer=Adam; epoch=800					

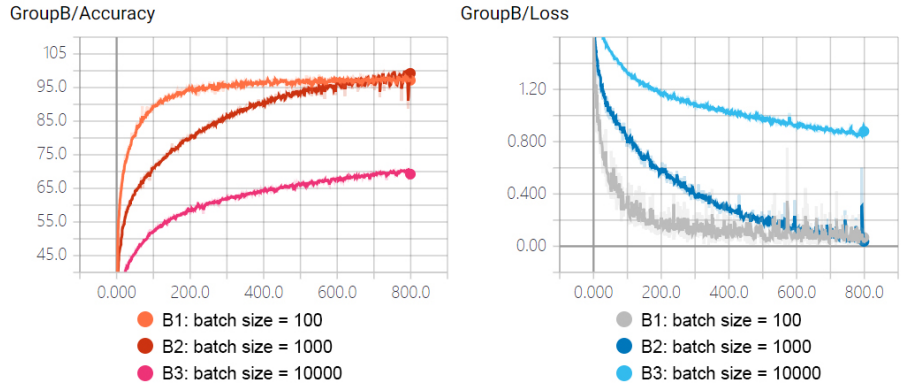

Figure 3: Training process of Experiment Group B(screen-shot from TensorBoard)

From the experiment we can prove some basic principles. There are some advantages to increasing an appropriate number of batch size in training:

- The time cost will reduce dramatically with the decreasing number of iteration. Some data can be find in Figure 3. The experiment B1, batch size equal to 100, cost 8768 seconds in 800 epoch. While the experiment B2 which batch size equal to 1000 only cost 4412 seconds nearly half of experiment B1.
- The larger the batch size, the more accurate the determined direction of decline, and the smaller the training oscillation. Figure 3 shows that the training oscillation in experiment B2(batch size=1000) is much stable than experiment B1(batch size=100).

- Load more data to the GPU will lead to memory utilization improvement. Although the table and figure do not show the memory utilization, but it is obvious. The command "nvidia-smi" could be used for monitor the GPU when training.

Increasing to an over large batch size may also lead some problems. On the one hand, training process will cost lots of GPU memory which may lead out of memory error. On the other hand, training process will cost more time to achieve the same accuracy. Table 3 shows that the experiment B1(batch size=100) cost around 23 minutes to achieve 95% training accuracy while experiment B2(batch size=1000) cost approximately 55 minutes.

Overall, both too small and too large batch size will cause problems. Choosing an appropriate batch size is important which will give you a faster and more accurate result.

Experiment Group C

Optimizer tie model parameters and the loss function together by updating the model in response to the output of the loss function. Experiment group C use 4 different optimizers to training VGG16 network.

From Table 4 and Figure 4, different optimizers have their own features which related to the time cost and the training process. Different optimizer's loss trend are also significantly different. But no matter which optimizer is chosen, the final result of the network training is very familiar.

In this experiment, the Adagrad performance best, it converges fastest and get a good accuracy and loss at the end. The RMSprop is designed to solve the problem of a sharp decline in Adagrad's learning rate, although its performance not as good as Adagrad in this experiment. The Adam added bias-correction and momentum to RMSprop, it shows an average performance. The SGD is one of the most common optimizers in machine learning, it also shows a good result.

Table 4: Results of Experiment Group C

Exp No.	Network	Optimizer	Time Cost	Training Accuracy	Testing Accuracy
C1	VGG16	Adam	5745s	99.378%	99.652%
C2		SGD	5431s	99.992%	99.994%
C3		Adagrad	2851s	99.994%	99.988%
C4		RMSprop	2860s	99.400%	99.110%
Other parameters: batch size=200; learning rate=0.001; epoch=200					

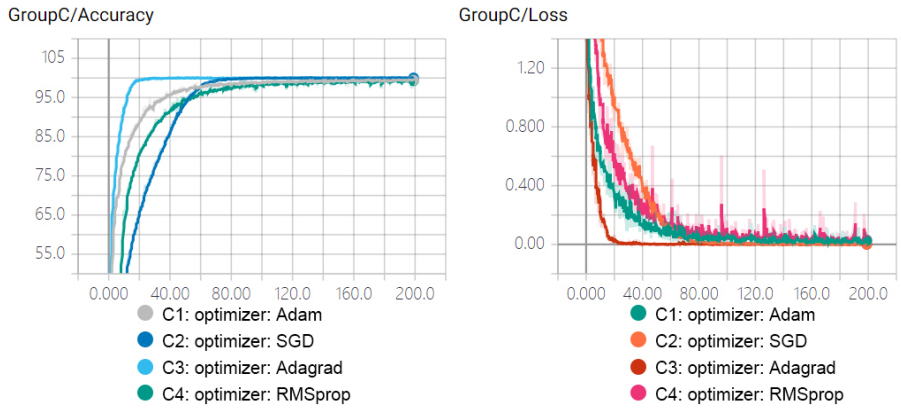

Figure 4: Training process of Experiment Group C(screen-shot from TensorBoard)

Experiment Group D

Experiment group D try to test the new CNN base on LeNet. Experiment D1 is the benchmark of the LeNet. There is no any change on it. Experiment D2 is a new network which base on LeNet, removes the 'MaxPool2' layer and add one convolution layer(kernel=6) and one ReLu activation function. Experiment D3 is another new network which base on LeNet, removes the 'MaxPool1' layer and add two convolution layers(kernel=8) and two ReLu activation function after each convolution layer. Architecture of these two CNN are as shown on Figure 1.

Table 5 and Figure 5 show the training result of experiment group D. Experiment D2 increase accuracy on the testing set by around 5% from the original network(experiment D1). By contrast, Experiment D3 gives a not good result, which make the accuracy decrease by around 10%

Table 5: Results of Experiment Group D

Exp No.	Network	Time Cost	Training Accuracy	Testing Accuracy
D1	LeNet	2439s	93.066%	93.394%
D2	LeNet*	2061s	97.652%	98.072%
D3	LeNet*	2620s	81.452%	83.046%
Other parameters: learning rate=0.001; batch size=100; optimizer=Adam; epoch=200				

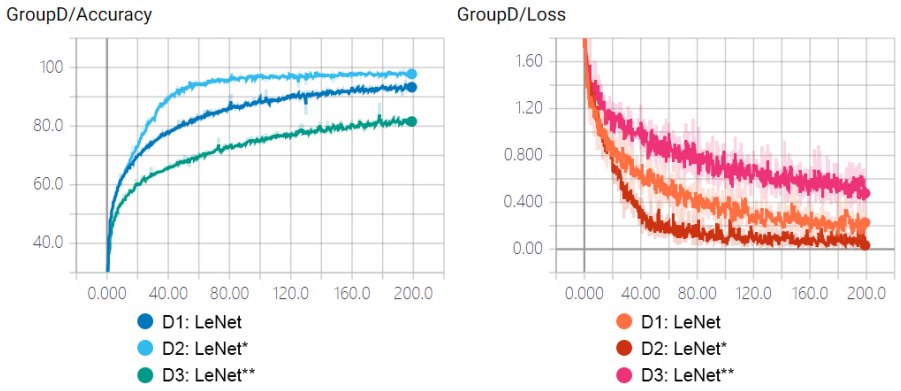

Figure 5: Training process of Experiment Group D(screen-shot from TensorBoard)

Reference

Chollet, F., 2016. How convolutional neural networks see the world. The Keras Blog, 30.

Krizhevsky, A., Sutskever, I. and Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105).

Singh, C.D., 2015. Image Classification: CIFAR-10 Neural Networks vs Support Vector Machines.