EE569 Digital Image Processing

HOMEWORK #5

CNN Training on LeNet-5

Issued: 3/23/2020 **Due: 4/7/2020**

Xu Kangyan

4876-0109-98

kangyanx@usc.edu

Problem 1: CNN Training on LeNet-5

(a) CNN Architecture

(b) CIFAR-10 Classification

(c) State-of-the-Art CIFAR-10 Classification

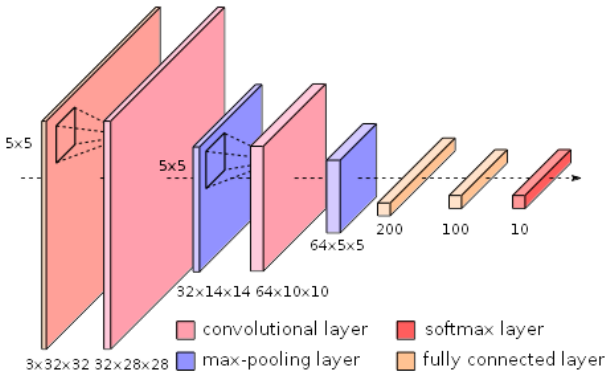

Figure 1. Modified LeNet-5 for CIFAR-10

I. CNN Architecture

1. CNN components

fully connected layer: output is a $n \times 1$ vector, each dimension (a number) comes from a weighted sum of all the outputs from previous layer. Suppose the previous layer has a size of $k \times m \times n$, for each $m \times n$ matrix using a weighted $m \times n$ matrix to do convolution and gets a number, then summing k numbers in total to get one dimension of FC layer. Repeat that for all dimensions in FC layer. $k \times m \times n \times FC\_dimension$ parameters are needed, which is very large. Bias can also be added for each dimension, $FC\_dimension$ in total.

convolutional layer: For each input image with size $k \times m \times n$, different kernels (size: $k \times s \times s$) are used to do convolution, and each output is a feature image, whose size is given by $Output_{size} = \frac{width_{input} - size_{filter} + 2 \times padding}{stride} + 1(bias)$ with $k \times s \times s + 1$ parameters in total.

During each convolution, the kernel is not changed.

Normally many feature images are generated, using different kernels.

max pooling layer: Normally choosing the largest number in a 2×2 block, and the output has a half size ($m \times m\ to\ \frac{m}{2} \times \frac{m}{2}$).

activation function: Normally Sigmoid or ReLU are used. Activation function is used to make the network become a nonlinear system.

softmax function: Normally used in final step for classification. After softmax function, output values are between $(0,1)$ and their sum is 1, so the larger value represents a higher possibility.

2. Over-fitting issue

The model's prediction curve fits the particular data set too much that it may not be able to represent extra data.

Early stopping: stop training if training set's error rate no longer decreases.

Data Augmentation: using methods like flipping, cropping to increase data set without collecting new data.

Dropout: neglecting some neuros under certain probability.

3. Why CNNs work much better

Image Segmentation uses Laws Filters to get 25 filtered images, the parameters are fixed, and the computation is not much. However, CNN uses more kernels to do convolution, and adding some nonlinear steps like max pooling and activation function. Besides, CNN can learn and adjust its parameters epoch after epoch.

4. Explain loss function and classical backpropagation (BP) optimization procedure

Loss Function: used to represent a "cost", so that the optimization procedure can minimize it. MSE and Cross-entropy Loss are commonly used.

BP optimization procedure: BP is based on Chain rule. It uses chain rule to calculate $\frac{\partial E}{\partial \omega_{ij}}$,

and updates the weight ω_{ij} with $-\alpha \frac{\partial E}{\partial \omega_{ij}}$,where α is learning rate.

II. CIFAR-10 Classification (PyTorch)

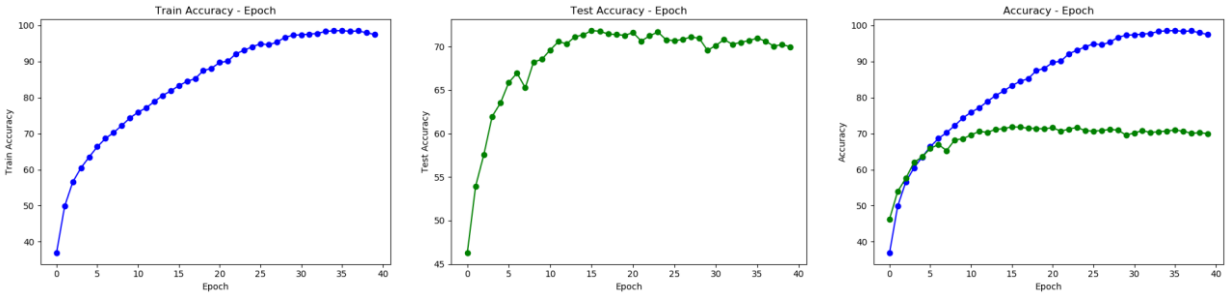

Figure 2. accuracy performance curves

Batch Size (Train): **500** / Batch Size (Test): **100** / Epoch Times: **40** / Learning Rate: **0.001**

Convolution Layer 1: **32** / Convolution Layer 2: **64** / Kernel Size: **5**

Fully Connected Layer 1: **200** / Fully Connected Layer 2: **100**

Epoch 39 - 65.412081 sec

Accuracy on 50000 train images: 97 % / Accuracy on 10000 test images: 69 %

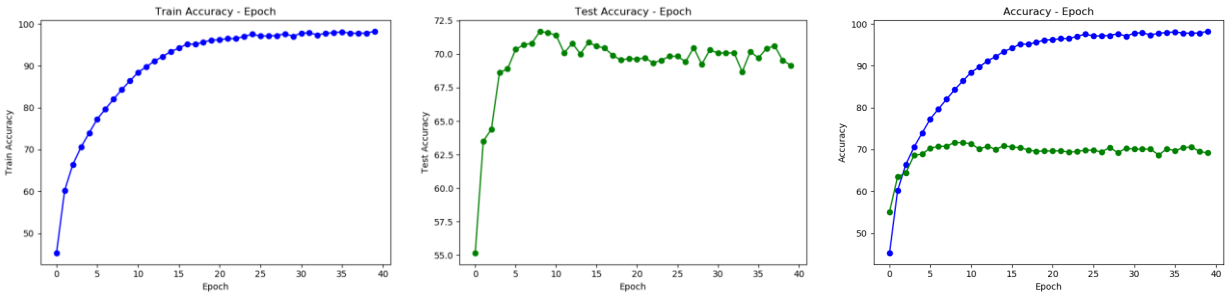

Figure 3. accuracy performance curves

Batch Size (Train): 100 / Batch Size (Test): **100** / Epoch Times: **40** / Learning Rate: **0.001**

Convolution Layer 1: **32** / Convolution Layer 2: **64** / Kernel Size: **5**

Fully Connected Layer 1: **200** / Fully Connected Layer 2: **100**

Epoch 39 - 58.798262 sec

Accuracy on 50000 train images: 98 % / Accuracy on 10000 test images: 69 %

Change the Train batch size, the training converges faster.

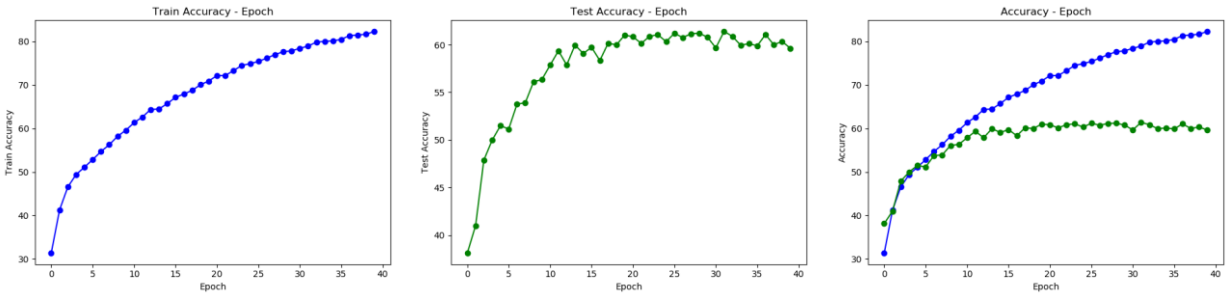

Figure 4. accuracy performance curves

Batch Size (Train): **100** / Batch Size (Test): **100** / Epoch Times: **40** / Learning Rate: **0.005**

Convolution Layer 1: **32** / Convolution Layer 2: **64** / Kernel Size: **5**

Fully Connected Layer 1: **200** / Fully Connected Layer 2: **100**

Epoch 39 - 58.635617 sec

Accuracy on 50000 train images: 82 % / Accuracy on 10000 test images: 59 %

Increases the learning rate, the training converges slower. Besides, test accuracy converges at 59%, which is lower.

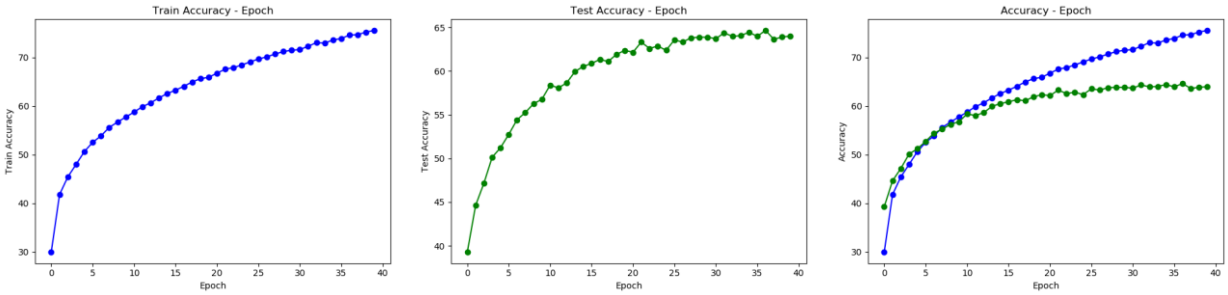

Figure 5. accuracy performance curves

Batch Size (Train): **500** / Batch Size (Test): **100** / Epoch Times: **40** / Learning Rate: **0.001**

Convolution Layer 1: **6** / Convolution Layer 2: **16** / Kernel Size: **3**

Fully Connected Layer 1: **120** / Fully Connected Layer 2: **84**

Epoch 39 - 18.397682 sec

Accuracy on 50000 train images: 75 % / Accuracy on 10000 test images: 64 %

Change the network parameter. A smaller size of convolution layer makes each epoch time shorter. Test accuracy is near 64%.

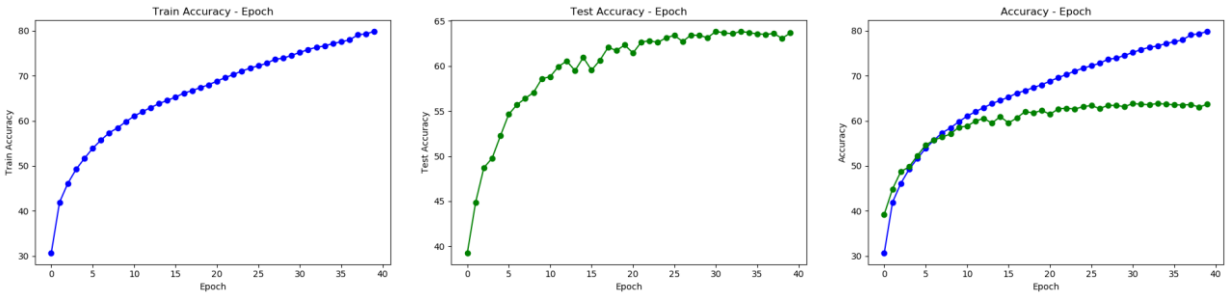

Figure 6. accuracy performance curves

Batch Size (Train): **500** / Batch Size (Test): **100** / Epoch Times: **40** / Learning Rate: **0.001**

Convolution Layer 1: **6** / Convolution Layer 2: **16** / Kernel Size: **5**

Fully Connected Layer 1: **200** / Fully Connected Layer 2: **100**

Epoch 39 - 21.829536 sec

Accuracy on 50000 train images: 79 % / Accuracy on 10000 test images: 63 %

Change the kernel size and fully connected layer. No major difference on the performance.

In conclusion, best parameter I found is:

Batch Size (Train): **100** / Batch Size (Test): **100** / Epoch Times: **40** / Learning Rate: **0.001**

Convolution Layer 1: **32** / Convolution Layer 2: **64** / Kernel Size: **5**

Fully Connected Layer 1: **200** / Fully Connected Layer 2: **100**

Accuracy on 10000 test images: nearly 70 %

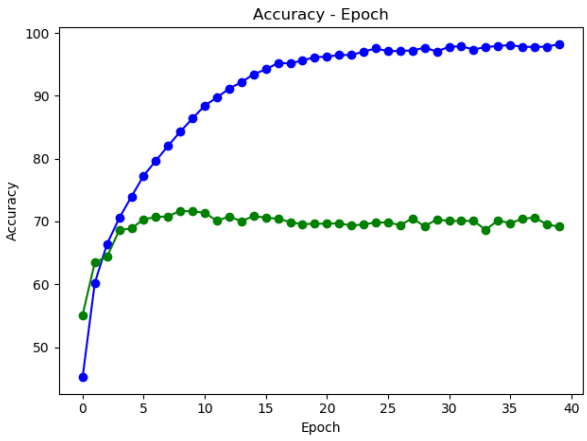

Figure 7. accuracy performance curve

III. State-of-the-Art CIFAR-10 Classification

Paper selected: **Fractional Max-Pooling**

1. what the authors did

Instead of a simple max pooling focused on 2×2 blocks, the author proposed a fractional version $\alpha \times \alpha$ where α can be non-integer value ($1 < \alpha < 2$), called FMP.

There are three choices while implementing FMP: the value α, choosing pooling regions in *random* or *pseudorandom* fashion, whether pooling regions are *disjoint* or *overlapping*.

The author finds that *overlapping* regions work better. In a *disjoint* sample, input square image can be divided into many disjoint rectangles (size:$1 \times 1, 1 \times 2, 2 \times 1, 2 \times 2$) for pooling.

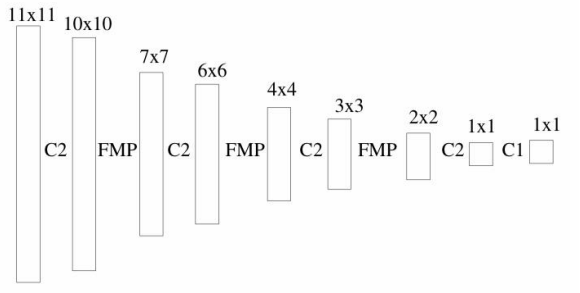

Figure 8. An example network from [1]

The author uses $\frac{10}{7}$, $\frac{6}{4}$, $\frac{3}{2}$ to represent $\sqrt{2}$. (While in MP2 the input size will be half, so using FMP there can be more layers)

2. Compare the solution

In [1], the author says performance on CIFAR-10 gets a test error of 3.47% (100 tests).

The author also added dropout and data augmentation in the training.

For LeNet-5, MP2 is very simple, fast and reduce size quickly, but it's not so "random", and 75% information is lost. It also needs convolution layer to be deep.

For FMP, it is a little complicated, but it decreases the feature image's size much slower, so Convolution-FMP operation can be implemented more times in the network, and the information loses also slower.

References

[1] Graham B. Fractional max-pooling (2014)[J]. arXiv preprint arXiv:1412.6071, 2014.