ECE590-10/11: Lab #2
Construct, Train, and Optimize Neural Network Models

Hai Li and Yiran Chen

ECE Department, Duke University — September 11, 2019

Objectives

Lab #2 is designed to guide you to learn how to design a DNN step by step and to understand how the size of the DNN determines its computational cost. Lab #2 is also designed to help you to master the training of the DNNs. Upon completing Lab #2, you will be able to obtain good understandings of:

- How the size of a DNN determines its memory consumption and computational cost;
- How to construct a DNN for image classification tasks (e.g., CIFAR-10);
- How to train a DNN by optimizing hyperparamters to combat overfitting, and maximize the potential of the DNN.

We encourage you to complete the Lab #2 on the JupyterLab server since it consumes a lot of computing power of GPUs. We will use the kaggle site `https://www.kaggle.com/c/ece590lab-2/` for result submission. You may need to use `https://www.kaggle.com/t/564d968cd0e4443fb36e2c3494153135` to join this competition.

Warning: You are asked to complete the assignment independently.

This lab has **125** points in total. The submission deadline will be **11:59pm, Wednesday, October 2**.

For the coding practice in Assignments 1∼4, we provide a template named `lenet5-cifar10.ipynb` to start with. The exercise of Assignment 5 will be conducted in a competition format. You are asked to start from scratch. The grade will be determined by the performance obtained at the central server.

To summarize, you are asked to submit all the following **4** parts by the deadline: (1) a PDF report that describe your results, analysis and lessons learned in the lab; (2) your original code of `lenet5-cifar10.ipynb` for Assignments 1∼4; (3) your prediction of the 10,000 images in the testset on **Kaggle**; and (4) the training code for your CIFAR-10 model for Assignment 5.

1 LeNet-5

LeNet-5 is a classic convolutional neural network (CNN) for hand-written digits recognition. It has 2 CONV layers, 2 POOL layers and 3 FC layers. Although LeNet-5 is very small, it is a good start to learn from this classic architecture that contains all of the basic components of CNNs. In addition, go through the process of building this model gives you some ideas about how to build a classifier and perform the training from scratch. We will use PyTorch to build LeNet-5. We use LeNet-5 to conduct CIFAR-10 task by replacing the 'tanh' activation with 'ReLU' activation.

Figure 1 gives a visualization of the adapted LeNet-5 model.

Assignment 1 (10 points)

(a) (10 pts) It is important to understand the size of DNNs. We usually measure the size of DNNs by its memory consumption and computational cost. Given the configurations of the LeNet-5 model, please compute its memory consumption (i.e., number of parameters) and computational cost (i.e., number of Multiply-Accumulates, or MACs).

Layer	Type	depth/units	Activation	Strides	Notes
1	Convolution	6	ReLU	1	
2	MaxPool	N/A	N/A	2	Pooling size is the same as strides.
3	Convolution	16	ReLU	1	
4	MaxPool	N/A	N/A	2	Pooling size is the same as strides.
5	Fully-connected	120	ReLU	N/A	Flatten layer is inserted before this op.
6	Fully-connected	84	ReLU	N/A	
7	Fully-connected	10	Softmax	N/A	

Table 1: Adapted LeNet-5 Model

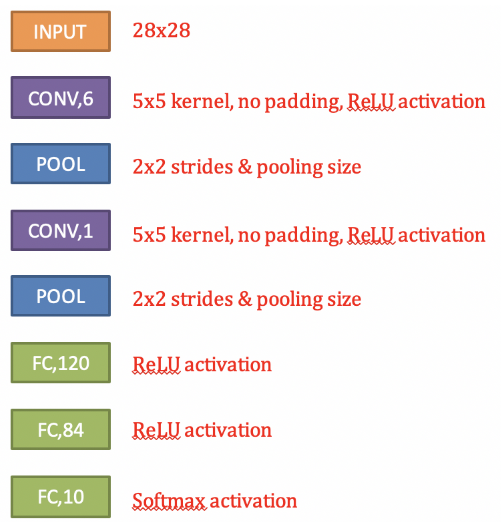
INPUT
28x28
CONV,6
5x5 kernel, no padding, ReLU activation
POOL
2x2 strides & pooling size
CONV,1
5x5 kernel, no padding, ReLU activation
POOL
2x2 strides & pooling size
FC,120
ReLU activation
FC,84
ReLU activation
FC,10
Softmax activation

Figure 1: A visualization of the LeNet-5 model.

2 Constructing DNN

In this lab, we will use the CIFAR-10 dataset for training and evaluation of the LeNet-5 model. The CIFAR-10 dataset [1] consists of 60000 32x32 colour images in 10 classes, each of which has 6000 images. We have modified the original CIFAR-10 dataset for its usage on the ECE590 course. We will release 45,000 images as the training dataset, 5,000 images as the validation dataset, and hold 10,000 images as the final test dataset.

Assignment 2 (25 points)

(a) (10 pts) Build the LeNet-5 model by following table 1 or figure 1.

(b) (10 pts) Write functions to load dataset and preprocess the incoming data. We recommend that the preprocess scheme **must** include normalize, standardization, batch shuffling to make sure the training process goes smoothly. The preprocess scheme may also contain some data augmentation methods (e.g., random crop, random flip, etc.).

(c) (5 pts) In the targeted classification task, we use cross entropy loss with L2 regularization as the learning object. You need to formulate the cross-entropy loss function in PyTorch. You should also specify a PyTorch Optimizer to optimize this loss function. We recommend you to use the SGD-momentum with an initial learning rate 0.01 and momentum 0.9 as a start.

3 Training DNN

After you constructed a DNN and setup the optimizer for the current loss function, you can start the training process.

Assignment 3 (25 points)

(a) (10 pts) Start the training process over the whole CIFAR-10 training dataset. For sanity check, you are required to report the initial loss value at the beginning of the training process and briefly justify this value. Run the training process for **a maximum of 30** epochs and you should be able to reach around **65%** accuracy on the validation dataset.

(b) (10 pts) You are asked to embed data augmentation into the code and describe its effects. You are requied to report validation results (i.e. validation accuracy) with and without data augmentation and briefly explain your findings.

(c) (5 pts) Insert batch normalization into the LeNet-5 architecture. Try to increase the initial learning rate to a larger value. What can you observe from the training process? Apply the same increased initial learning rate to the original LeNet-5 architecture without batch normalization and rerun the training process. Does the training process go smoothly? Briefly explain your findings.

4 Hyperparamter Optimization

Hyperparameter tuning is the most important part during neural network training. In this lab, we will discuss about only two hyperparameters: the learning rate and the regularization strength.

Assignment 4 (25 points)

(a) (10 pts) Try a few hyperparameter combinations to study the effects of hyperparameter selection during the training process. Plot learning curves for different hyperparameter combinations and briefly explain your findings.

(b) (10 pts) Learning rate is an important hyperparameter to tune. Specify a learning rate decay policy and apply it in your training process. Briefly describe its impact on the learning curve during your training process.

(c) (5 pts) As we can see from above, hyperparameter optimization is critical to obtain a good performance of DNN models. Try to fine-tune the model to over 70% accuracy. You may also increase the number of epochs to up to 100 during the process. Briefly describe what you have tried to improve the performance of the LeNet-5 model.

5 Crating your own DNN for CIFAR-10

The performance of LeNet-5 is limited and thus, the validation accuracy is hard to go higher. You can design your own CNN architecture to achieve a higehr performance on the validation dataset.

Assignment 5 (40 points)

(a) (40 pts) You are asked to design your own CIFAR-10 model, setup the training process and train the model to achieve a higher test accuracy on the CIFAR-10 testset. To evaluate your performance, you are asked to submit your predictions to a Kaggle site: `https://www.kaggle.com/c/ece590lab-2/`. We evaluate your submission and release the results based on only 50% of the testset. Your final grade is decided by the remaining 50% of the testset. Please check the Kaggle submission site for more details.

Upon completion of this assignment, you are also asked to submit the training code and include the implementation details (e.g., DNN design, data augmentation, training, hyperparameter optimization, etc.) in your submitted PDF report.

Grading Rubics The grading will largely depend on your final test accuracy on the testset. More specifically:

- You will have to receive at least 75% accuracy to make your submission valid.
- You will receive full credit as long as you achieved a test accuracy higher more than 80%.
- (**Extra Credit**) 5pts are awarded if you achieve an accuracy higher than 85%. Another 5pts are awarded if your achieve an accuracy higher than 90%.

Some additional requirements:

- **DO NOT** train on the testset or use pretrained models to get unfair advantage. We have already conducted special preprocessing on the original CIFAR-10 dataset. As we have tested, "cheating" on the full dataset will give you only 6% accuracy on our final test set, which means that you will fail on this assignment.
- **DO NOT** copy code directly online or from other classmates. We will check it! The result can be severe if your codes fail to pass our check.

As this assignment requires a lot of computing power of GPUs, you are asked to:

- Plan your work in advance and start early. We will **NOT** extend the deadline because of the unavailability of computing resources.
- Be considerate and kill Jupyter Notebook instances when you do not need them.
- **DO NOT** run your program forever. A maximum of 2 GPU hours is enough for you to complete the training process.

References

[1] A. Krizhevsky *et al.*, “Learning multiple layers of features from tiny images,” tech. rep., Citeseer, 2009.