Homework 5: Image Classification

December 1, 2023

Due Date: December 15 by 23:59:59

Introduction

In this assignment, you will implement and test various image classification models on the CIFAR-10 dataset. The goals of this assignment are as follows:

- Implement and compare the linear classifier and the full-connected neural network.
- Implement and compare the SVM loss and the cross-entropy loss.
- Implement and compare the AdamW and the SGD optimizer.
- Implement and compare the StepLR and the CosineAnnealingLR scheduler.
- Train and test two types of classifiers.

You can learn how to create, train, and test a model using PyTorch here. You are highly encouraged to go through this tutorial before you start.

Here are some other supplementary materials that may help you:

- PyTorch Documentation
- PyTorch Chinese Documentation
- Dive into deep learning

1 Define Classifiers (20 pts.)

1.1 Linear classifier (10 pts.)

Add your own code to the **LinearClassifier** class to define a linear classifier. Your classifier is required to process a mini-batch data.

1.2 Full-connected neural network classifier (10 pts.)

Add your own code to the **FCNN** class to define a full-connected neural network classifier. You are responsible for choosing the network depth, width, and activation type.

2 Define loss function (20 pts.)

You need to implement the SVM loss and the cross-entropy loss from scratch. The weight decay term for regularization is also required, but you are not required to implement it explicitly since you can use **weight_decay** defined in optimizer for this purpose.

2.1 SVM loss function (10 pts.)

Add your own code to the **svmloss()** function to define an SVM loss. You need to implement it from scratch instead of calling pre-implemented PyTorch functions that directly finish the task.

2.2 Cross-entropy loss function (10 pts.)

Add your own code to **crossentropyloss()** function to define cross-entropy loss. You also need to implement it from scratch instead of calling pre-implemented PyTorch functions that directly finish the task, such as **torch.nn.functional.cross_entropy** and **torch.nn.functional.nll_loss**.

3 Implement the training and testing function (30 pts.)

There is a whole training code in PyTorch Tutorial: train a classifier, you can learn from it. In this task, you need to implement the **train()** and **test()** function that can choose a model, optimizer, scheduler, and so on; see the end of the **main.py** for details.

4 Compare AdamW and SGD optimizer (10 pts.)

Train the classifiers you implemented using the AdamW (**torch.optim.AdamW**) and SGD (**torch.optim.SGD**) optimizer and compare the loss and accuracy curves. Put the results in your report.

5 Compare SVM and Cross-entropy loss (10 pts.)

Train the classifiers you implemented using the SVM and Cross-entropy loss and compare the loss and accuracy. Put the loss and accuracy curves and the final classification accuracies in your report. You can use **TensorBoard** in PyTorch to record and visualize the loss and accuracy curves. Here is a tutorial introducing **TensorBoard**.

6 Compare StepLR and CosineAnnealingLR scheduler (10 pts.)

Train the classifiers you implemented using two learning rate schedulers, including the StepLR (**torch.optim.lr_scheduler.StepLR**) and CosineAnnealingLR (**torch.optim.lr_scheduler.CosineAnnealingLR**) scheduler and compare the loss and accuracy curves. Put the results in your report.

7 Report

You have now completed the entire process of this project. Put all the visualizations and results in your report. More experiments and discussions are encouraged.

8 Submit

Be sure to zip your code and final report; Name it as **StudentID_YourName_HW5.zip**.