ENGINEERING DIVISION | NYU ABU DHABI

ENGR-UH 4560
Selected Topics in Information and Computational Systems

Deep Learning

Project 06 - Image Classifier (Pytorch)

Introduction

Image classification is the process of taking an input (like a picture) and outputting a class (like “cat”) or a probability that the input is a particular class (“there’s a 90% probability that this input is a cat”). You can look at a picture and know that you’re looking at a terrible shot of your own face, but how can a computer learn to do that? With a convolutional neural network!

Goals

We would like you to establish a neural network involving advance DNN modules (i.e. convolution layers, RELU, pooling and fully connection layers and etc.) to distinguish the specific category of an input image.

Dataset

You may find the dataset in the following link: CIFAR-10
The CIFAR-10 dataset consists of 60000 32x32 color images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.
The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class.

Requirements

1. Program a data loader for CIFAR-10 dataset:
    a. You are supposed to split the dataset into train data and test data
    b. Dataset predefined by Pytorch can be applied in your implement, but other three-party codes for loading CIFAR-10 are not allowed
    c. Both training data loader and testing data loader need to be established by *torch.utils.data.DataLoader*.
2. Built a CNN model for classification:
    a. The model should be established based on *torch.nn.Module* and *torch.nn.functional*
    b. The model (base model) need consist of at least two convolution layers, pooling layers, RELU and one fully connection layer
    c. You can add more layers in the base model to push performance, but improvement should be written in your report.
3. Write training frameworks to implement the classification pipeline:

   a. Apply the dataloader you have written in the first step
   b. Apply the CNN model you have written in the second step
   c. Define a Negative Log Likelihood loss for your implement
   d. Create an optimizer (SGD/Adam) based on Pytorch to implement Autograd mechanism (calculate gradients and update parameters within your model)
4. Write testing frameworks to evaluate your classification pipeline:
   a. Apply the dataloader you have written in the first step
   b. Apply the CNN model you have written in the second step
   c. Print out the accuracy of your model's prediction
5. Write a report to introduce the structure your implement and how it works:
   a. You are supposed to report where each modules (e.g. dataloader, CNN model, loss etc.) is in your code
   b. You are supposed to plot out the loss convergence course by Tensorboard or lera.ai.

Deliverables

A zip file containing the following:

1. a working project (source code, makefiles if needed, etc)
2. a report for the detailed description of the project
    a. explain the main aspects of your code
    b. how to run your project
    c. plots and diagrams

Before submitting your project, please make sure to test your program on the given dataset.

Notes

You may discuss the general concepts in this project with other students, but you must implement the program on your own. ***No sharing of code or report is allowed.*** *Violation of this policy can result in a grade penalty.*

Late submission is acceptable with the following penalty policy:
10 points deduction for every day after the deadline