COMP 5900V
Advanced Machine Learning
Winter 2021
Assignment 1

Background

This assignment is about Convolutional Neural Networks. You will do object recognition using CNN. The implementation is to be done with PyTorch and the dataset is CIFAR10. The easiest way to set up your environment is to use Colab.

Data set: You will use CIFAR-10 dataset. It consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

Deep Learning Framework: Backpropagation needs to compute gradients. It is simple for a one-layer fully connected network but it gets too complicated for larger networks. That's why we use frameworks such as PyTorch (Facebook), TensorFlow (Google), Caffe (UC Berkeley), MXNET (Open-Source), Theano (UdeM), Keras (higher level APIs). In this assignment, you will use PyTorch. It is an open source machine learning library primarily developed by Facebook. It allows Tensor computing (like NumPy) with strong acceleration via graphics processing units (GPU). Tensors can be treated as multidimensional array data structures (arrays). Tensors in PyTorch are similar to NumPy arrays, but can also be operated on a CUDA-capable Nvidia GPU.

Environment: You will use Colaboratory (also known as Colab), a free Jupyter notebook environment from Google that runs in the cloud and stores its notebooks on Google Drive. It is more suitable for interactive jobs rather than long runs. It is free. If you use it for long runs your access to the resources will be reduced, for example your request for GPU may be rejected and you end up using CPU which is still fine for doing this assignment. In fact, since your neural network is not big, you may not notice any improvement by switching from GPU to CPU.

First time using PyTorch? Check out the PyTorch tutorial posted on the course website.

Some useful modules in PyTorch

Autograd module: PyTorch uses a method called automatic differentiation. A recorder records what operations have performed, and then it replays it backward to compute the gradients. You don't need to compute the gradient yourself!

Optim module: torch.optim is a module that implements various optimization algorithms used for building neural networks. Most of the commonly used methods are already supported, so there is no need to build them from scratch.

nn module: PyTorch autograd makes it easy to define computational graphs and take gradients, but autograd is still too low level. The nn Module Pre-implements network layers already. No need to implement each layer yourself.

Where to start?

- Find Assignment01.ipynb on the course website and copy it to your Google Drive.
- Open it in Colab. You may need to search for Colab and associate it with ipynb file type in your GoogleDrive. You need to do it just once.
- Run the cells in the notebook one by one.

Questions

Q1 (3 points) Run the CNN for 10 epochs (# of passes through the training set) and record the accuracy of the test set per epoch. Then, drop the conv layers and modify the code to get a simple neural network. Change the number of hidden layers from 0 to 4 and record the models accuracy per epoch. Compare the accuracy of the CNN with the simple neural networks with 0,1,2,3,4 hidden layers. Each hidden layer has 120 nodes with Relu as non-linearity. Plot the test accuracy (computed on the test set) versus epochs. 10 epochs may not be enough to reach the convergence, but it is enough to see the trend. Create a single figure. Your figure must have 6 curves, one for CNN and five for the simple neural network with 0,1,2,3,4 hidden layers. Explain why some models perform better than others?

Q2 (3 points) Create a new model by replacing the Relu unites with Sigmoid unites in the CNN. Plot the test accuracy versus epoch number for 10 epochs. Create a single figure. Your figure should contain two curves. One for the model with Relu units and one for the model with Sigmoid units. Explain the results.

Q3 (4 points) The provided CNN network uses 5x5 filters with valid convolution. Design three new models by changing the filter size to 3x3 and/or the convolution type to same convolution. Run the new models for 10 epochs. Plot the accuracy on the test set versus epoch number for the four models. Create a single figure. Your figure should contain four curves corresponding to 1- (5x5 filters, valid convolution), 2-(3x3 filters, valid convolution), (5x5 filters, same convolution) and 4-(3x3 filters, same convolution). Explain why some models perform better than others? Use print(Net()) to generate a summary of the structure of your model. Repeat this for the four models and attach the output generated by print(Net()).

Note: Submit your report as a single pdf in CuLearn. Also submit your codes as a single zip file. Do not include the pdf in the zip code.