Lab 2pre

CNN and Image Classification Basics

Instructions

- There is no required submission for this pre-lab.
- Complete the two tutorials in *Playing with CNNs* and *Train an Image Classifier with CNNs*, respectively. If you are not familiar with Google Colab, it may be useful to go through the quick introduction. Other information is provided here for your reference *only*.
- This is a preparatory step for Lab 2, which uses deep learning models, specifically, CNNs implemented with PyTorch, for a classification task. If you are not familiar with deep learning, it may take time to get familiar with CNNs and PyTorch. It is always good to **start early**.

This preparatory exercise helps you to get started with CNNs and PyTorch. After completing it, you should be able to build your own CNN model and learn a classifier from data.

The Convolutional Neural Network (CNN) is designed specifically for image classification and regression. A CNN is a neural network composed of several convolutional layers. Each layer uses learnable kernels to perform convolution over input images. To perform classification or regression, output features of the final convolutional layer are often flattened to a 1D vector and fed into additional fully connected (FC) layers. The final output is also a 1D vector. In the case of classification, the output represents the likelihoods of the input belonging to the various classes. In the case of regression, the output directly predicts the target. CNNs are typically trained using gradient descent algorithms.

PyTorch is a commonly used deep learning library. It is a Python package that provides convenient functions for implementing and training CNN models.

We will use Google Colab for both Lab 2 and this preparatory exercise. In Colab, you can edit, run, and share Python codes on webpages, with GPU access and free storage on the cloud.

The Colab sheets used in this exercise are official tutorials provided by PyTorch.

Play with CNNs

Follow this tutorial to play with a toy CNN model using PyTorch:

- Click on “Run in Google Colab” to open the Colab sheet.
- Run through the code blocks.
- Modify the parameters of the convolutional and FC layers to make the CNN wider or narrower.
- Add or remove convolutional and FC layers to make the CNN deeper or shallower.

The example model in this tutorial is a regression model to be trained with the mean square error (MSE) loss. Try to adapt it for a classification model with 5 classes, and change the loss to cross-entropy loss.

Train an Image Classifier with CNNs

Follow this tutorial for an image classification example. The goal is to train a CNN to identify the class of the object in input images using a labeled dataset. The dataset is CIFAR10, which consists of 60,000 color images categorized into 10 classes. Each image has size 60 × 60, and each class contains 6000 images. Play with the depth and the width of the CNN model as well as the hyperparameters of the optimizer, such as the learning rate, batch size, *etc.*, to see how they affect the loss and accuracy of predictions.

Use the SoC Computer Cluster (optional)

If you wish to train larger neural network models and require additional computing resources, consider SoC's computer cluster. To use the school cluster's for model training:

- Apply an MySoC account at this portal if you do not have one already.
- Enable the "SoC Compute Cluster" through the portal.
- You may find out the number of compute nodes available here.
- Connect to a specific host using ssh within SoC. For example,

  ```
  ssh xlog0
  ```

  Additional details on access to the computer cluster is available here.
- The computer cluster is managed by Slurm. You may find this guide useful.

Additional Information

If you wish to learn more about deep learning with PyTorch, read up on

- *Deep learning with PyTorch: a 60-minute blitz*.
- Datasets and data loaders
- Optimizing model parameters
- Save and load the model

They may be useful for Lab 2 and the course project.