HT2021: IL2230 HADL Lab 2
Handwritten Digits Recognition from MLP to CNN

IL2230 Course Team

November 2, 2021

1 Introduction

The deep learning framework we use in IL2230 is PyTorch, which is free and open-source software released under the Modified BSD license. It is primarily developed by Facebook's artificial intelligence research group.

PyTorch is a Python based library built to provide flexibility as a deep learning development platform. The workflow of PyTorch is as close as you can get to python's scientific computing library – numpy.

If you lack knowledge on Python, please go through an excellent online tutorial: The Python Tutorial `https://docs.python.org/3/tutorial/Python` is an interpretive powerful programming language which is easy to learn and convenient to use. It has high-level data structures with object-oriented programming. It is excellent for scripting and rapid application development.

Hand-written digits recognition has been a long standing problem since the beginning of machine learning and pattern recognition. There are certainly many different approaches. In the lab, we try to use neural networks to address this problem. In particular, we use a CNN called LeNet-5 [1] to do this task. LeNet-5 is perhaps the first successful application of convolutional neural networks (CNNs) in real-life problems. CNNs can automatically extract features in a way much more efficient and effective than manually-made features. After feature extraction by convolutional layers, the neural network can perform classification using fully connected layers.

2 Purpose

This lab is composed of two parts. The first part is for the beginners of PyTorch users and deep learning learners. It has the following objectives:

- Understand PyTorch's Tensor library and neural networks at a high level
- Train a small neural network to classify images
- Get the first-hand experience on the complexity of CNN design

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition." Proceedings of the IEEE, 86(11):2278-2324, November 1998.

The second part is to use and contrast different neural networks to solve an image recognition problem. It has a few intended learning outcomes after completing the project:

- be able to design and train MLPs.
- be able to design and train CNNs, which is a special kind of MLPs.
- be able to compare the performance of different neural networks, in particular, understand the benefits of convolutional layer in automatic feature extraction.

This lab is a group work with typically four students per group. Each group submits one short technical report.

3 Task: Part 1 CNN in PyTorch

PyTorch is a well documented deep learning framework. Its official website provides a rich source of very good tutorials. Your task in this part is to learn from scratch, starting from installation to use.

3.1 Installation

Install the package manager Anaconda `https://www.anaconda.com/distribution/` if you are new to Python. You can use it to create your Python environment and manage all dependencies.

1. Download and install Anaconda (Python will be installed as part of the installation)
2. Open a command window with Anaconda
3. Run the following command to install PyTorch

   ```
   conda install pytorch torchvision -c pytorch
   ```

   Optionally, you can create a virtual environment for pytorch. You may refer to "Create virtual environments for Python with conda". `https://uoa-eresearch.github.io/eresearch-cookbook/recipe/2014/11/20/conda/`

3.2 Tutorial

Follow the tutorial "Deep Learning with PyTorch: A 60 Minute Blitz" to run all the given code. `https://pytorch.org/tutorials/beginner/deep_learning_60min_blitz.html` We run through all the following four parts:

1. What is PyTorch?
2. Autograd: Automatic Differentiation
3. Neural Networks

4. Training a classifier with CPU (the GPU part is optional. You can skip.)

   To know more about related functions, you may go through another tutorial "Learning PyTorch with Examples", which introduces Tensors, Autograd, and NN modules.

   `https://pytorch.org/tutorials/beginner/pytorch_with_examples.html`

What is the accuracy of the image classifier? What is the accuracy for each image class?

3.3 Exploration

After the tutorial, you are going to design new CNNs in order to improve the classification accuracy by considering the following adjustments:

1. Increase the number of feature maps. For example, C1 has 6 features maps. What if you increase it to 8, 10, 12 feature maps? The same goes for C3, with 20, 24, 28 feature maps.

2. Add more convolutional layers, turning the 7-layer neural network into a 9-layer, 11-layer, or 13-layer neural network. You are free to modify the filter size.

3. Experiment with different nonlinear activation functions. Instead of ReLu, try to use sigmoid, tanh.

   `https://pytorch.org/docs/stable/nn.html#non-linear-activations-weighted-sumnonlinearity`

4. Draw accuracy figures with the different setups (number of feature maps, number of convolutional layers, nonlinear activation function) above. Use the original design as the baseline for comparison.

   Give your comments on the CNN designs and their accuracy. Which factor has more significant impact on the performance of the classifier? Why?

4 Task: Part 2 Handwritten Digits Recognition

Part 2 of this lab intends to solve an image recognition problem, specifically, handwritten digits. The data set is the well-known MNIST, which consists of only 10 classes from 0, 1, 2, ... to 9.

The MNIST database of handwritten digits has a training set of 60,000 examples, and a test set of 10,000 examples. The digits have been size-normalized and centered in a fixed-size image. Figure 1 shows an example of ten digits in MNIST. It is a good database for testing learning techniques and pattern recognition methods on real-world data while spending minimal efforts on data pre-processing and formatting.
Your tasks are as follows:

1. Design and train an MLP. You can consider two variants: (1) MLP1: MLP with 1 hidden layer. (2) MLP2: MLP with 2 hidden layers.

   For MLP1, you can try different number of neurons for the hidden layer, say from 30 to 300 neurons with a step size of 30. Certainly you can try your own options.

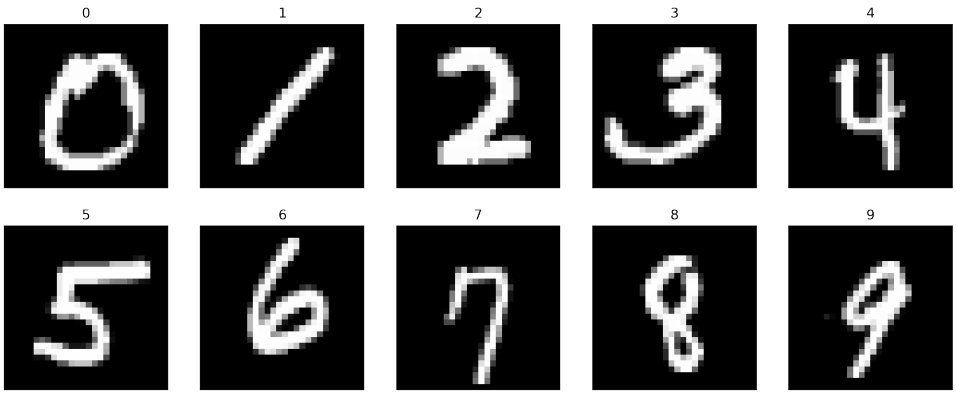

Figure 1: MNIST example.

For MLP2, you can try different numbers of neurons for the two hidden layers. One possible choice is 300 neurons for the first hidden layer and 100 neurons for the second hidden layer.

2. Write and train a neural network implementing the LeNet-5 model. LeNet-5 is perhaps the first CNN which successfully deals with hand-written digits recognition problem. Note that the images were centered in a 28x28 image. The standard LeNet-5 assumes the image size 32x32.

3. Record and compare the inference performance, namely, classification accuracy, runtime, memory consumption, of the two approaches.

   Hint: Use PyTorch Profiler to characterize execution time and memory consumption.

5 Documentation

- Your final task is to write a short technical report mainly about the "Section 3.3 Exploration" and "Section 4 Handwritten Digits Recognition" tasks you have done.

  For part 1 of the report, you present your new CNN designs (network structure and per-layer details), and discuss the impact of feature map, layer depth, nonlinear functions on the classification accuracy according to the results you have obtained. Try to use figures and tables to assist your explanations and discussions.

  For part 2 of the report, you describe your procedures for implementing the tasks, reporting results you obtained, and discussing your results.

  You need to answer the questions when your lab results are checked by TAs and write them in your report.

- Upload your source code together with your group report to the Canvas course page. The source code should accompany a simple Readme.txt for how to run the code and obtain the graphs/numbers in your report.

After your lab results have been approved by the lab assistant, submit your report via the Canvas course website: `https://kth.instructure.com/courses/26565`

6 Appendix

Links to online materials.

- PyTorch recipes and tutorials.

  Tutorials: `https://pytorch.org/tutorials/`

  Recipes: `https://pytorch.org/tutorials/recipes/recipes_index.html`

  PyTorch profiler: `https://pytorch.org/tutorials/recipes/recipes/profiler.html`

- The MNIST dataset. `http://yann.lecun.com/exdb/mnist/`

  This website gives detailed description of the MNIST dataset. Read this as the first step of understanding the dataset and its data structure.

  You can also load MINIST by

```
import torchvision.datasets as datasets
mnist_trainset = datasets.MNIST(root='./data',
        train=True, download=True, transform=None)
```