CS 4100 5100

What Neural Networks See

Instructions

- **ALERT:** Expect high runtimes. Start early.

Academic Integrity

- If you discuss concepts related to this programming assignment with one or more classmates, all parties must declare collaborators in their individual submissions within a code comment. Such discussions must be kept at a conceptual level, and no sharing of actual code is permitted.
- You may use any generative AI tool available to you, as long as it is appropriately cited in a code comment. **However, for this assignment I strongly urge you not to rely on GenAI tools, but rather write your own code based on PyTorch documentation for a meaningful learning experience.**
- Failure to disclose collaborations or use of generative AI will be treated as a violation of academic integrity, and penalties listed in the course syllabus will be enforced.

Reach Out!

If at any point you feel stuck with the assignment, please reach out to the TAs or the instructor, and do so early on! This lets us guide you in the right direction in a timely fashion and will help you make the most of your assignment.

Fashion-MNIST Classification

Implement and train Neural Networks (20)

Relevant files: `fashionmnist.py`, `ffn.py`, `cnn.py`

In this section, you will be working with a dataset called Fashion-MNIST. Your task is to design and train two neural networks - one with a fully feedforward architecture, and one with convolutional layers - that classify each image of the dataset into one of 10 possible output classes. Data is downloaded directly from within the script (using PyTorch). You are expected to experiment with hyperparameters such as the number of layers, the number of neurons in each layer, the number and sizes of convolution kernels, etc.

Fashion-MNIST contains grayscale images of 28 x 28 pixels representing images of clothing. The dataset has 60000 training images, and 10000 testing images, and each image comes with an associated label (e.g. t-shirt, coat, bag, etc.). There are 10 classes, just like the MNIST handwritten digits dataset we discussed in class, so that it may serve as a direct drop-in replacement to test neural networks. Read the full details about this dataset at the repository.

The starter code for this part of the assignment is divided into three files. The skeleton code in `fashionmnist.py` serves as a general guide for the end-to-end training process. The `ffn.py` and `cnn.py` files contain skeleton code for the two model architectures, which you must also complete. There are some sections in the code without too many guidelines where you will be expected to research and experiment with various techniques to find a good approach. You must use PyTorch in this section.

Your accuracy on the testing dataset must be greater or equal to 80% for both models. Grading is based on test accuracy. The Gradescope leaderboard displays mean accuracy across the test set and a random subset of the training set.

Additional Resources

Please refer to the following for more information about convolutions, pooling, and convolutional layers in PyTorch. The first link is an especially good write-up about how convolutional networks work, and mirrors the format of our class notes quite closely:

- DeepLizard - Convolutional Neural Nets
- CNN Demo
- Max Pool Demo
- Image Kernels
- PyTorch - Conv2d

Instructions: Add the generated plots for all questions in both sections below to a single PDF file, and submit it along with your code to Gradescope.

Plotting Losses, Prediction Examples (8)

1. For **each** neural network, submit a figure containing one image that is classified incorrectly by the model, and one image classified correctly by the model. Include clear labels that indicate the network used, the *predicted* classes from the model and the *true* classes the images belong to (both human-readable labels, not just the class number). (3)

2. For **each** neural network, submit one plot showing your training loss over time. (2)

3. For **each** neural network, submit its total number of parameters. (1)

4. For **each** neural network, submit a confusion matrix plot on the test data. Please include clear labels indicating *predicted* and **true** classes. Here are some resources to help you learn more: (2)

- The concept and the usefulness of a confusion matrix: Confusion Matrix
- Generating a confusion matrix: sklearn-metrics: confusion matrix
- Plotting a confusion matrix: seaborn: heatmap

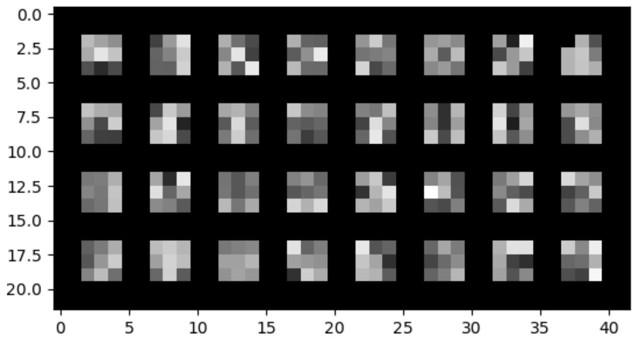
Exploring Kernels (12)

Relevant files: `kernel_vis.py`

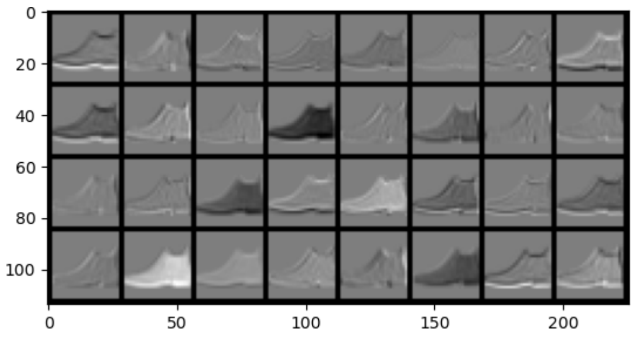
Finally, we will learn to visualize kernels, and the features they extract from images in a convolutional neural network. The `kernel_vis.py` file contains skeleton code for this process, where you must complete the missing sections. The goal is to a) visualize the kernels themselves and b) applying the kernel to an image to extract features. Since we are dealing with grayscale images, each kernel in the convolutional neural network can be thought of as a single-channel feature extractor, and is simply a matrix of a chosen size. The numbers in this matrix, when normalized to the range $[0, 1]$, can be plotted as an image. Then, this kernel, when applied to a sample image, produces an output that is similar in shape to the original image (may be slightly smaller, depending on the stride), which can also be normalized and plotted as an image.

5. Submit a plot of the kernels your model learns (i.e., after model training) at the first convolutional layer, arranged in a grid, like the example below (the number of kernels may differ based on your implementation). (4)

6. Submit a plot of the features extracted from the sample image by each of the kernels at the first convolutional layer, arranged in a grid, like the example on the next page (the number of images depends on the number of kernels). (4)



(a) Kernels learnt at the first conv. layer



(b) Kernels applied to sample image

7. Process the provided sample image through each layer of your CNN (both convolutional and pooling layers) and visualize the feature maps at each stage. Create a single figure showing the sample image and the output of each layer's convolution + activation and/or pool. For each visualization, display the first channel of the feature map and clearly label which layer produced it. This demonstrates how features evolve from low-level edge detection to more abstract representations. (4)

Conceptual Questions (5)

8. Why does the CNN achieve better performance on image classification tasks? Consider the role of convolutional layers in feature extraction versus the fully-connected layers when answering this question. (2)

9. Looking at your feature map progression visualization, describe how the features evolve as they pass through the CNN layers. What types of features does each layer appear to be detecting, and how does this hierarchical feature extraction help with classification? (3)