Assignment 2

CMPUT 328

Fall 2023

[**Total Weightage:** 10%]

1 In-lab: Implement a Fully Connected Network using PyTorch

[2% **of the total weight**]

Implement a Fully-connected network architecture using the built-in functions of PyTorch. Train this network on the MNIST dataset with two different losses – *crossentropy* and *L2*. Please refer to the **Network Architecture** section for the specifications.

You need to complete the `class FNN` in `FNN_submission.py` to implement the forward pass as well as the loss computation.

- `__init__(self, loss_type, num_classes)` initializes your network layers
- `forward(self, x)` takes a batch of images as a tensor of size $N \times 784$ and returns the class probabilities as a tensor of size $N \times 10$ where N is the batch size
- `get_loss(self, output, target)` takes the output of the forward pass and ground truth labels of the corresponding images and returns a tensor containing the loss computed according to the loss_type argument of `__init__`

1.1 Quiz

[2% **of the total weight**]

You are expected to have completed this part by the end of the lab, though the submission deadline for this part is the same as the other parts. There will be an in-lab quiz at the end of that lab with questions related to this part. We will also collect attendance in the lab at some point during the lab hours using your ONEcard for identity verification.

If you are not present in the lab (or fail to produce your ONEcard) when the attendance is taken, you will lose the marks for the quiz (2% weight) entirely.

1.2 Network Architecture

$$Y_p = Softmax(Relu(Tanh(XW_1 + b_1)W_2 + b_2))W_3 + b_3)$$

You can use built-in torch functions for defining the layers (`nn.Linear`) and activations. Dimensions of the vectors and matrices are as follows: X contains the input images having a shape $(N \times 784)$. N is the batch size. W_1 is (784×64), b_1 is (1×64), W_2 is (64×32), b_2 is (1×32), W_3 is (32×10), b_3 is (1×10). Output probabilities Y_p has the shape $(N \times 10)$. Note that for each of N indices in the first dimension, the $softmax$ function is applied along the second dimension of its input matrix.

1.2.1 Loss function

In your code, you will use two loss functions: $cross - entropy$ and $L2$. You can use Pytorch builtin implementations.

2 Part 2 (Out of lab): Classification using CNNs for MNIST

[3% **of the total weight**]

You are going to implement a Convolutional Neural Network (CNN) to classify the MNIST and the CIFAR-10 dataset. The network architecture is not fixed. You will design the network architecture yourself.

However, there are some requirements that your network architecture must satisfy:

1. Your network architecture must have at least 4 layers (which is either a convolution or fully connected layer passed into an activation function like Relu, Tanh, etc). Max pooling or strided convolution layers used to downsample activation maps do not count. The number of layers in your network will be very close to the number of activation functions.
2. Must have at least one convolution layer.
3. Must have at least 1 Max Pooling.
4. Must have at least 1 fully connected layer at the end

If your network architecture doesn't satisfy any of the above requirements, **marks will be deducted**. For every requirement that is not satisfied, you lose **10%** of your marks.

Similar to Assignment 1, make sure to test the trained model on the validation set every few epochs to prevent overfitting.

3 Part 3 (Out of lab): Classification using CNNs for CIFAR-10

[3% **of the total weight**]

Repeat Part 2 for the CIFAR-10 dataset. Make sure that your submitted code works for both datasets (they have different number of channels!).

4 Additional Information

4.1 Template Code

4.1.1 Part 1

You are given `FNN_main.py` and `FNN_submission.py`. Please do not make any changes to `FNN_main.py` as you will not be submitting this file. You will only submit `FNN_submission.py`. All you need to do is finish the `class FNN`.

4.1.2 Part 2 and Part 3:

You are given `CNN_main.py` and `CNN_submission.py`. Please do not make any changes to `CNN_main.py` as you will not be submitting this file. You will only be submit `CNN_submission.py`.

`CNN_submission.py` has some template code provided. Please do not make any changes to `load_dataset()`. You are free to make changes to the other functions/classes and add your own but make sure `CNN(dataset_name, device)` returns the model within the dict as shown in the template. This is how `CNN_main.py` expects it to be returned.

4.2 Running the Code

1. **Part 1:** `python3 FNN_main.py`
2. **Part 2/3:** `python3 CNN_main.py`

Optionally, if you have `paramparse` installed, you can then use command line arguments.

1. **Part 1:** `python3 FNN_main.py loss_type=ce` or `python3 FNN_main.py loss_type=l2`
2. **Part 2:** `python3 CNN_main.py dataset=MNIST`
3. **Part 3:** `python3 CNN_main.py dataset=CIFAR10`

4.3 Grading

Please keep in mind that there are no partial marks in this assignment. Any submission that fails to run or does not satisfy **Accuracy** and **Runtime** will get no marks. All runtimes are with respect to Colab GPU.

4.3.1 Part 1

1. A correct implementation will have a test accuracy of around 92 to 97%. You will get a full score if it falls within this range. Any submission with a test accuracy $< 92\%$ will get no marks.
2. **Runtime:** Should be less than 300 seconds.

4.3.2 Part 2

1. MNIST: should be minimum 97% to get the maximum score. Score will scale linearly from 93-97% on the test set. Any submission with a test accuracy $< 93\%$ will get no marks.
2. **Runtime:** Should be less than 400 seconds

4.3.3 Part 3

1. CIFAR 10: should be minimum 65% to get the maximum score. Score will scale linearly from 55-65% on the test set. Any submission with a test accuracy $< 55\%$ will get no marks.
2. **Runtime:** Should be less than 400 seconds

4.4 Submission Guidelines

Submit only `FNN_submission.py` and `CNN_submission.py`. Do not submit the `_main.py` files.

4.5 Collaboration Policy

This must be your own work. Do not share or look at the code of other students (whether they are inside or outside the class). Do not copy the code from the Internet. You can talk to others in the class about solution ideas (but detailed enough that you are verbally sharing, hearing or seeing the code). You must cite whom you talked with in the comments of your programs.