CSC 4780/6780
Fall 2022
Homework 11

November 5, 2022

This homework is due at 11:59 pm on Sunday, Nov 13. It must be uploaded to iCollege by then. No credit will be given for late submissions. A solution will be released by noon on Monday, Nov 8.

it is always a good idea to get this done and turned in early. You can turn it in as many times as you like – iCollege will only keep the last submission. If, for some reason, you are unable to upload your solution, email it to me before the deadline.

Incidentally, I rarely check my iCollege mail, but I check my `dhillegass@gsu.edu` email all the time. Send messages there.

Be sure to rename your solution directory to match your name.

1 Classifying Images with a Neural Net

The first place where deep neural nets excelled was image classification. One of the standard image classification tasks is Fashion-MNIST. The data set has small images (28×27) with 255 levels of gray scale:

`https://github.com/zalandoresearch/fashion-mnist`

The images are in 10 categories:

t-shirt	0
trouser	1
pullover	2
dress	3
coat	4
sandal	5
shirt	6
sneaker	7
bag	8
angle boot	9

This has a lot in common with the example I did in class: `https://colab.research.google.com/drive/15y0ZySFucnUxHpVAUbWVK1-fGD_2pPgT`

1.1 Dealing with the data

You can get the FashionMNIST data from the keras project:

```
from keras.datasets import fashion_mnist

(X_train,  y_train_np), (X_test,  y_test_np) = fashion_mnist.load_data()
```

These are numpy arrays. You must convert them to pytorch tensors before they are useful. Also the shape of the input is (60,000, 28, 28); You will need to reshape it to (60,000, 784). And you need to make sure that X_train is of type Float32. For the training data this would look like this:

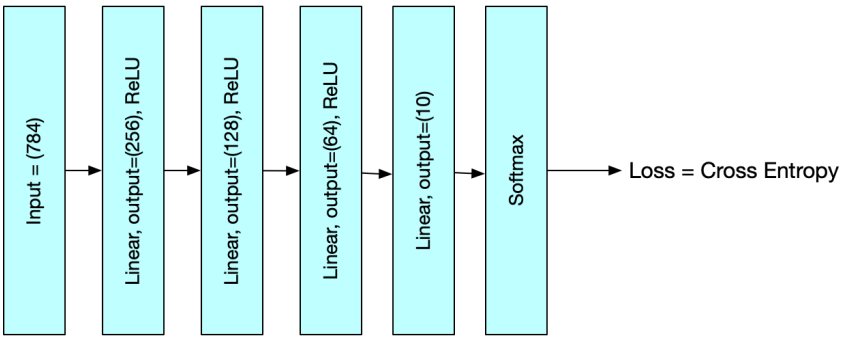
```
# Make each image into a long vector with range 0 to 1.0
X_train = X_train.reshape((-1, IN_D)).astype(np.float32) / 255.0

# Convert to pytorch tensors
X_train = torch.from_numpy(X_train)
y_train = torch.from_numpy(np.copy(y_train_np))
```

Now the data is ready to be used with pytorch.

1.2 The Model

Create a file called `FashNet.py` that has the subclass of `torch.nn.Module`. Here is a diagram of the model you will create:



Your model will instantiate instances of the following classes in its `__init__` method

- `torch.nn.Linear`
- `torch.nn.ReLU`
- `torch.nn.Softmax`

Then it will run the data through those layer in its `forward` method.

1.3 Training

You will write a program called `fashion_train.py` that reads in the training data, instantiates an instance of `FashNet`, and trains it. It will print out the information about its layers. Every 50th iteration of training, it will print out its cross entropy loss and accuracy. You will train it for a total of 501 iterations. The output will look like this:

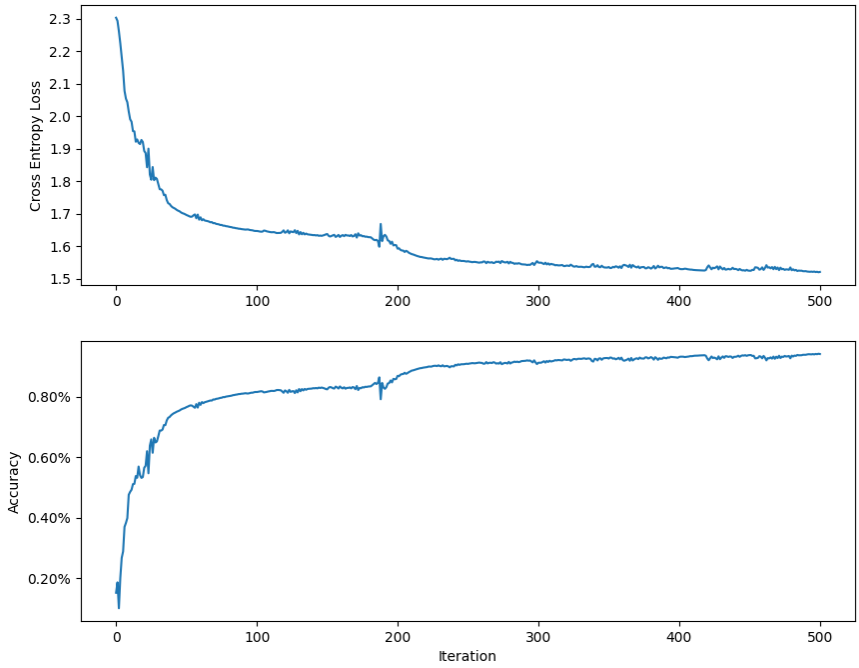
```
> python3 fashion_train.py
Input: (60000, 28, 28)
Model parameters:
fc1.weight:[256, 784]
fc1.bias:[256]
fc2.weight:[128, 256]
fc2.bias:[128]
fc3.weight:[64, 128]
fc3.bias:[64]
fc_last.weight:[10, 64]
fc_last.bias:[10]
Total parameters: 242,762
```

```
Training:
   0: loss: 2.302428,  accuracy: 15.13%
  50: loss: 1.696137,  accuracy: 76.52%
 100: loss: 1.647088,  accuracy: 81.50%
 150: loss: 1.637731,  accuracy: 82.39%
 200: loss: 1.592654,  accuracy: 86.85%
 250: loss: 1.554131,  accuracy: 90.79%
 300: loss: 1.549938,  accuracy: 91.18%
 350: loss: 1.536394,  accuracy: 92.56%
 400: loss: 1.531210,  accuracy: 93.09%
 450: loss: 1.524864,  accuracy: 93.71%
 500: loss: 1.521189,  accuracy: 94.06%
Training took 411.79 seconds
Wrote weights.pt
```

(Your output may not be exactly the same as mine. The weights are initialized randomly, and you can get different results based on that initialization.) Save this output to `train_out.txt`

Use an ADAM optimizer. The learning rate of 0.01 worked well for me, but you should play with it.

`fashion_train.py` will also plot its cross-entropy and accuracy on the training data as it is trained:



(This plot will be saved as `learning.png`.)

Finally, it will save the model's state dictionary to a `weights.pt` using `torch.save`.

2 Testing

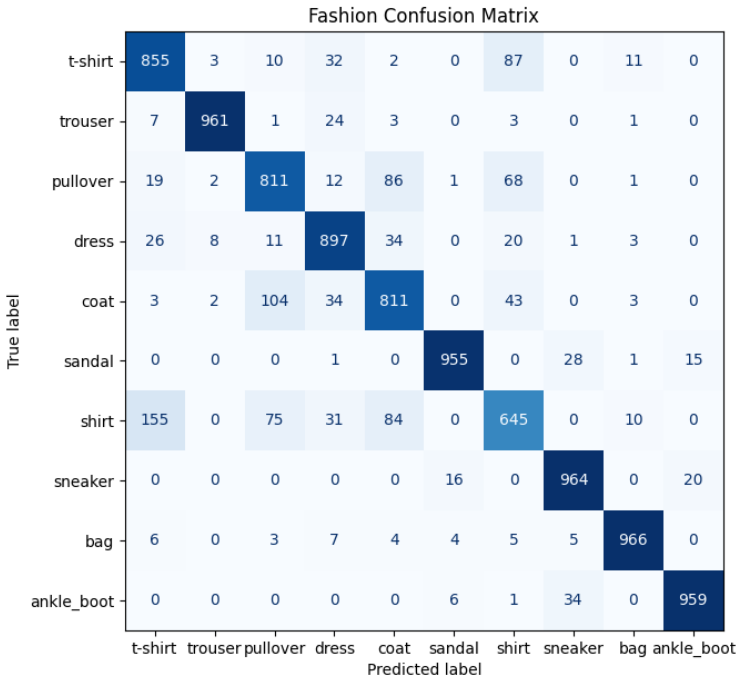
Create a program called `fashion_test.py` that reads in the weights and the test data. It should compute the accuracy and create a confusion matrix in `confusion.png`.

It will look like this when it runs:

```
> python3 fashion_test.py
Input: (10000, 28, 28)
Accuracy on test data: 88.24%
Confusion:
[[855   3  10  32   2   0  87   0  11   0]
 [  7 961   1  24   3   0   3   0   1   0]
 [ 19   2 811  12  86   1  68   0   1   0]
 [ 26   8  11 897  34   0  20   1   3   0]
 [  3   2 104  34 811   0  43   0   3   0]
 [  0   0   0   1   0 955   0  28   1  15]
 [155   0  75  31  84   0 645   0  10   0]
 [  0   0   0   0   0  16   0 964   0  20]
 [  6   0   3   7   4   4   5   5 966   0]
 [  0   0   0   0   0   6   1  34   0 959]]
Wrote confusion.png
```

Save this output to `test_out.txt`

The confusion matrix will look like this:



3 Batching

Sometimes the whole training data set is too big to work with efficiently. When this happens, we break the training data into "mini-batches". You should break the 60,000 rows into 600 batches, each with 100 rows. The rows should be randomly selected. They should appear exactly once in the resulting batches.

Sometimes when you move to a batched training model you need to tweak your hyperparameters.

Copy `fashion_train.py` to `fashion_train_batches.py`. Add batching to it.

In the original `fashion_train.py`, you generated statistics using the whole training dataset. If the dataset is too big, you typically are content with using the stats from the last batch of the epoch. In this case, it means that the stats represent the loss and accuracy for 100 samples instead of the full 60,000.

You will probably need to adjust the learning rate of the optimizer.

The weighs generated by `fashion_train_batches.py` will work fine with `fashion_test.py`

4 Criteria for success

If your name is Fred Jones, you will turn in a zip file called `HW11_Jones_Fred.zip` of a directory called `HW11_Jones_Fred`. It will contain:

- `fashion_train.py`
- `fashion_train_batches.py`
- `fashion_test.py`
- `FashNet.py`
- `weights.pt`
- `confusion.png`
- `learning.png`
- `train_out.txt`
- `test_out.txt`

Be sure to format your python code with black before you submit it.

We would run your code like this:

```
cd HW11_Jones_Fred
python3 fashion_train.py
python3 fashion_test.py
python3 fashion_train_batches.py
python3 fashion_test.py
```

Do this work by yourself. Stackoverflow is OK. A hint from another student is OK. Looking at another student's code is *not* OK.

The template files for the python programs have import statements. Do not use any frameworks not in those import statements.