Lewis Arnsten

I apologize for this homeworks slight lateness. I had COVID up until the middle of this week (I tested positive Feb. 22). This significantly impaired my ability to do work over the course of this assignment.

Image of gradient_checker results:

```
(base) Lewiss-MacBook-Pro:hw3 lewisarnsten$ python3 gradient_checker.py
Testing Conv2d:
Shape Match = True
Result Match = True
Result Match = False
Result Match = True
Result Match = True


Testing Linear:
Shape Match = True
Result Match = True
Result Match = True
Result Match = True
Result Match = True


Testing AvgPool2d:
Shape Match = True
Result Match = True
Result Match = True


Testing ELU:
Shape Match = True
Result Match = True
Result Match = True


Testing BatchNorm2d:
Shape Match = True
Result Match = True
Result Match = True
Result Match = True
Result Match = True


Testing cross_entropy_loss_with_softmax:
Shape Match = True
Result Match = True
Result Match = True
```

Logic behind pytorch CNN architecture for segmentation:

I first created a series of sequences (4 sequences) to conduct convolutions, batch normalization, and ReLU activation--adding layers for feature extraction. When each sequence was called, it was followed by max pooling with stride two to cut image resolution in half. In my first convolution I increased the number of layers from 3 to 64. I then followed a common practice of doubling the number of layers in each convolution until I felt it would be trivial to add more (512). I then created and executed 4 more sequences utilizing convolutions, batch normalization, ReLU activation, and transpose convolution resulting in output of shape (64, 20, 32, 32). I tested the SGD and Adam optimizers, and decided upon Adam due to better performance. I also decided upon a learning rate of 3e-4 experimentally.

Results: achieved ~80% accuracy in 20 epochs

```
Epoch 18, Iteration 0, loss = 0.1236
Epoch 18, Iteration 100, loss = 0.1896
Epoch 18, Iteration 200, loss = 0.1141
Epoch 18, Iteration 300, loss = 0.1157
Epoch 18, Iteration 400, loss = 0.1532
Epoch 18, Iteration 500, loss = 0.1926
Epoch 18, Iteration 600, loss = 0.3634
Epoch 18, Iteration 700, loss = 0.2351
Epoch 19, Iteration 0, loss = 0.1206
Epoch 19, Iteration 100, loss = 0.1905
Epoch 19, Iteration 200, loss = 0.0894
Epoch 19, Iteration 300, loss = 0.0819
Epoch 19, Iteration 400, loss = 0.1238
Epoch 19, Iteration 500, loss = 0.1598
Epoch 19, Iteration 600, loss = 0.1443
Epoch 19, Iteration 700, loss = 0.0831
Got 8165912 / 10240000 correct (79.75)
```