Stuart Harley

Lab 8: Convolutional Neural Networks Report

For the following data, I trained on my local CPU so I have no GPU memory usage data.

Fashion-MNIST: 1st architecture

For this network, I arbitrarily chose the architecture and hyperparameters that I did.

(1x28x28) – (5x28x28) – (2x28x28) -> 1568 – 10,

padding=1, filter sizes – 3x3, step size=.001, 10 epochs

no regularization – it was unhelpful in the fully connected network for Fashion-MNIST

training acc – 51.76%, testing acc – 52.90%, training loss – 1.35, testing loss – 1.31

time estimate – 60sec per epoch so 600sec. Based on running my old setup for Fashion-MNIST took about 30sec an epoch and this is a more complicated network.

Actual time – 1248sec

2nd architecture

The 1st architecture was improving, but not by much so I decided to increase the step size.

Same as 1 but step size=.005

training acc – 70.68%, testing acc – 71.08%, training loss - .869, testing loss – .855

time estimate – should be about the same as the 1st run; ~1248sec

Actual time – 1291sec

3rd architecture

The 2nd architecture was improving slower at the end. The original fully connected architecture for Fashion-MNIST got over 80% accuracy in 10 epochs quickly. So I decided to increase the number of filters in each layer so it learned more weights (kernels) and can predict better.

Same as 2 but (1x28x28) – (10x28x28) – (10x28x28) -> 7840 – 10

Training acc – 73.47%, testing acc – 74.07%, training loss - .808, testing loss - .759

Time estimate – 2 times as long as the last. I more than doubled the number of kernels learned and the last fully connected layer is significantly longer.

Actual time – 1484sec

4th architecture

For the 4th architecture I used the same network as 3 but ran it for 100 epochs.

I stopped it after 60 epochs because it had stagnated.

Training acc – 77.99%, testing acc – 75.92%, training loss - .648, testing loss - .697

Time estimate – 6 times as long as the last since 60 epochs is 6x10 – 89040sec

Actual time – 90432sec

CIFAR-10: 1st architecture

For this network, I arbitrarily chose most of the architecture. I chose the same step size, epochs, and lambda that I used in the fully connected layer for CIFAR-10.

(3x32x32) – (10x32x32) – (5x32x32) -> 5120 -10 w/regularization on both kernels & the fully connected layer's weights.

Padding=1, filter sizes – 3x3, step size=.001, 10 epochs, lambda for reg=.01

Training acc – 43.95%, testing acc – 43.7%, training loss – 1.75, testing loss – 1.75

Time estimate – slightly smaller then the 3rd architecture of Fashion-MNIST, so about 1400sec

Actual time – 1444sec

2nd architecture

The last network worked well, so for this one I am going to add some more filters to each layer to hopefully allow it to learn more.

(3x32x32) – (20x32x32) – (20x32x32) -> 20480 – 10 w/same regularization

Training acc – 46.68%, testing acc – 46.93%, training loss – 1.74, testing loss – 1.71

Time estimate – larger than the last architecture, but when I made the architecture larger in the Fashion-MNIST it didn't increase by much since filters are not fully connected. So ~2000sec.

Actual time – 2669sec

3rd architecture

The last network worked well, so I am extended the number of epochs I run to 100. Also, after 20 epochs, I am dropping the step size from .001 to .0001, since in other networks it seems to have plateaued by that point so hopefully this will allow it to continue to increase. I decided to stop training after 30 epochs because the accuracy was not changing much at that point.

Epochs=30, step size - .001 < 20 epochs, .0001 > 20 epochs

Training acc – 55.80%, Testing acc – 51.58%, Training loss – 1.48, Testing loss – 1.56

Time estimate – 3x as long as the last network since 100 epochs = 3x10 - ~8007sec

Actual time – 7923sec

Conclusion:

For the Fashion-MNIST dataset, I was unable to outperform my performance when I used the fully connected implementation from the previous lab. In that lab I achieved an 83% accuracy in only 10 epochs, and it trained quicker than any of the networks using convolution.

For the CIFAR-10 dataset, I was able to achieve a better accuracy when using convolution. In the previous lab, I achieved an accuracy of 43% at best. When using convolution, my best network was able to achieve over 51% accuracy.

For both networks, it is likely that I could achieve better accuracies in my networks if I used more filters per convolutional layer.