HW2 Short Answers (Not Collab)

Note: After finishing my code and optimizing it, I found that my MNIST network trains at a rate of 2 it/s. Each epoch is 600 its, hence this mean I spend about 5 minutes per epoch. For 20 epochs this comes out to 1 hr 40 mins. For MNISTRes network my speed was a little over 2 s/it. This means that each epoch takes 20 minutes and 20 epochs take about 7 hours. Hence for the MNIST network I trained my model for 20 epochs but for MNISTRes network I trained my model for 5 epochs due to the time constrains. My learning rate for the last assignment was 0.1 instead of 0.01, this learning rate yielded the desired response. Hence after some trial and error the learning rate for this assignment is 0.025. This learning rate is giving the desired accuracy.

1) The MNISTRes network without any modifications gave me an accuracy of 98.7% in 5 epochs. After adding more Resnet blocks my accuracy was 98% after 5 epochs. My network architecture was:

MNISTResNetwork:

(layers): SequentialLayer:

(0): ConvLayer: Kernel: (5, 5) In Channels 1 Out Channels 6 Stride 1

(1): MaxPoolLayer: kernel: 2 stride: 2

(2): ReLULayer:

(3): ConvLayer: Kernel: (5, 5) In Channels 6 Out Channels 16 Stride 1

(4): ResNetBlock:

(conv_layers): SequentialLayer:

(0): ConvLayer: Kernel: (3, 3) In Channels 16 Out Channels 16 Stride 1

(1): ReLULayer:

(2): ConvLayer: Kernel: (3, 3) In Channels 16 Out Channels 16 Stride 1

(add_layer): AddLayer:

(relu2): ReLULayer:

(5): ResNetBlock:

(conv_layers): SequentialLayer:

(0): ConvLayer: Kernel: (3, 3) In Channels 16 Out Channels 16 Stride 1

(1): ReLULayer:

(2): ConvLayer: Kernel: (3, 3) In Channels 16 Out Channels 16 Stride 1

(add_layer): AddLayer:

(relu2): ReLULayer:

(6): MaxPoolLayer: kernel: 2 stride: 2

(7): ResNetBlock:

(conv_layers): SequentialLayer:

(0): ConvLayer: Kernel: (3, 3) In Channels 16 Out Channels 16 Stride 1

(1): ReLULayer:

(2): ConvLayer: Kernel: (3, 3) In Channels 16 Out Channels 16 Stride 1

(add_layer): AddLayer:

(relu2): ReLULayer:

(8): ResNetBlock:

(conv_layers): SequentialLayer:

(0): ConvLayer: Kernel: (3, 3) In Channels 16 Out Channels 16 Stride 1

(1): ReLULayer:

(2): ConvLayer: Kernel: (3, 3) In Channels 16 Out Channels 16 Stride 1

(add_layer): AddLayer:

(relu2): ReLULayer:

(9): ReLULayer:

(10): FlattenLayer:

(11): LinearLayer: (784, 120)

(12): ReLULayer:

(13): LinearLayer: (120, 84)

(14): ReLULayer:

(15): LinearLayer: (84, 10)

(loss_layer): SoftmaxCrossEntropyLossLayer:

2) I tried LeakyRelu, my network architecture is:

MNISTResNetwork:

(layers): SequentialLayer:

(0): ConvLayer: Kernel: (5, 5) In Channels 1 Out Channels 6 Stride 1

(1): MaxPoolLayer: kernel: 2 stride: 2
(2): LeakyReLULayer:
(3): ConvLayer: Kernel: (5, 5) In Channels 6 Out Channels 16 Stride 1
(4): ResNetBlock:
(conv_layers): SequentialLayer:
(0): ConvLayer: Kernel: (3, 3) In Channels 16 Out Channels 16 Stride 1
(1): LeakyReLULayer:
(2): ConvLayer: Kernel: (3, 3) In Channels 16 Out Channels 16 Stride 1
(add_layer): AddLayer:
(relu2): LeakyReLULayer:
(5): ResNetBlock:
(conv_layers): SequentialLayer:
(0): ConvLayer: Kernel: (3, 3) In Channels 16 Out Channels 16 Stride 1
(1): LeakyReLULayer:
(2): ConvLayer: Kernel: (3, 3) In Channels 16 Out Channels 16 Stride 1
(add_layer): AddLayer:
(relu2): LeakyReLULayer:
(6): MaxPoolLayer: kernel: 2 stride: 2
(7): LeakyReLULayer:
(8): FlattenLayer:
(9): LinearLayer: (784, 120)
(10): LeakyReLULayer:
(11): LinearLayer: (120, 84)
(12): LeakyReLULayer:
(13): LinearLayer: (84, 10)
(loss_layer): SoftmaxCrossEntropyLossLayer:
The accuracy I got after 5 epochs was: 98.8%.

3) With the regular Resnet network I am getting about 99% accuracy in 5-6 epochs, this was the upper limit. I could not do any better than that. I tried 3 different architectures:
    a. The regular MNIST network gave me 95.7% accuracy in 1 epoch and 98.9% accuracy in 20 epochs at epoch 5 I was at 97.7% accuracy:
    MNISTNetwork:
    (layers): SequentialLayer:
    (0): ConvLayer: Kernel: (5, 5) In Channels 1 Out Channels 6 Stride 1
    (1): MaxPoolLayer: kernel: 2 stride: 2
    (2): ReLULayer:
    (3): ConvLayer: Kernel: (5, 5) In Channels 6 Out Channels 16 Stride 1
    (4): MaxPoolLayer: kernel: 2 stride: 2
    (5): ReLULayer:
    (6): FlattenLayer:
    (7): LinearLayer: (784, 120)
    (8): ReLULayer:

(9): LinearLayer: (120, 84)
(10): ReLULayer:
(11): LinearLayer: (84, 10)
(loss_layer): SoftmaxCrossEntropyLossLayer:

b. I added another Resnet block:
MNISTResNetwork:
(layers): SequentialLayer:
(0): ConvLayer: Kernel: (5, 5) In Channels 1 Out Channels 6 Stride 1
(1): MaxPoolLayer: kernel: 2 stride: 2
(2): ReLULayer:
(3): ConvLayer: Kernel: (5, 5) In Channels 6 Out Channels 16 Stride 1
(4): ResNetBlock:
(conv_layers): SequentialLayer:
(0): ConvLayer: Kernel: (3, 3) In Channels 16 Out Channels 16 Stride 1
(1): ReLULayer:
(2): ConvLayer: Kernel: (3, 3) In Channels 16 Out Channels 16 Stride 1
(add_layer): AddLayer:
(relu2): ReLULayer:
(5): ResNetBlock:
(conv_layers): SequentialLayer:
(0): ConvLayer: Kernel: (3, 3) In Channels 16 Out Channels 16 Stride 1
(1): ReLULayer:
(2): ConvLayer: Kernel: (3, 3) In Channels 16 Out Channels 16 Stride 1
(add_layer): AddLayer:
(relu2): ReLULayer:
(6): ResNetBlock:
(conv_layers): SequentialLayer:
(0): ConvLayer: Kernel: (3, 3) In Channels 16 Out Channels 16 Stride 1
(1): ReLULayer:
(2): ConvLayer: Kernel: (3, 3) In Channels 16 Out Channels 16 Stride 1
(add_layer): AddLayer:
(relu2): ReLULayer:
(7): MaxPoolLayer: kernel: 2 stride: 2
(8): ReLULayer:
(9): FlattenLayer:
(10): LinearLayer: (784, 120)
(11): ReLULayer:
(12): LinearLayer: (120, 84)
(13): ReLULayer:
(14): LinearLayer: (84, 10)
(loss_layer): SoftmaxCrossEntropyLossLayer:

c. The last architecture I tried was:

```
MNISTResNetwork:
  (layers): SequentialLayer:
    (0): ConvLayer: Kernel: (5, 5) In Channels 1 Out Channels 6 Stride 1
    (1): MaxPoolLayer: kernel: 2 stride: 2
    (2): ResNetBlock:
      (conv_layers): SequentialLayer:
        (0): ConvLayer: Kernel: (3, 3) In Channels 6 Out Channels 6 Stride 1
        (1): ReLULayer:
        (2): ConvLayer: Kernel: (3, 3) In Channels 6 Out Channels 6 Stride 1
      (add_layer): AddLayer:
      (relu2): ReLULayer:
    (3): ReLULayer:
    (4): ConvLayer: Kernel: (5, 5) In Channels 6 Out Channels 16 Stride 1
    (5): ResNetBlock:
      (conv_layers): SequentialLayer:
        (0): ConvLayer: Kernel: (3, 3) In Channels 16 Out Channels 16 Stride 1
        (1): ReLULayer:
        (2): ConvLayer: Kernel: (3, 3) In Channels 16 Out Channels 16 Stride 1
      (add_layer): AddLayer:
      (relu2): ReLULayer:
    (6): ResNetBlock:
      (conv_layers): SequentialLayer:
        (0): ConvLayer: Kernel: (3, 3) In Channels 16 Out Channels 16 Stride 1
        (1): ReLULayer:
        (2): ConvLayer: Kernel: (3, 3) In Channels 16 Out Channels 16 Stride 1
      (add_layer): AddLayer:
      (relu2): ReLULayer:
    (7): ResNetBlock:
      (conv_layers): SequentialLayer:
        (0): ConvLayer: Kernel: (3, 3) In Channels 16 Out Channels 16 Stride 1
        (1): ReLULayer:
        (2): ConvLayer: Kernel: (3, 3) In Channels 16 Out Channels 16 Stride 1
      (add_layer): AddLayer:
      (relu2): ReLULayer:
    (8): MaxPoolLayer: kernel: 2 stride: 2
    (9): ReLULayer:
    (10): FlattenLayer:
    (11): LinearLayer: (784, 120)
    (12): ReLULayer:
    (13): LinearLayer: (120, 84)
    (14): ReLULayer:
    (15): LinearLayer: (84, 10)
```

(loss_layer): SoftmaxCrossEntropyLossLayer:

The accuracy I got was around the same as the original Resnet architecture.