1. See if you can improve the MNistResNetwork architecture using more ResNet-Blocks. What's the highest accuracy you achieve? What is the architecture?

I tested the MNistResNetwork architecture with different numbers of ResNetBlocks being 1,2,3 and 4 each trained using the same setting. The highest accuracy achieved was by the one with 3 ResNetBlocks, having **98.9% Test Accuracy**, however the performance of the 2 and 4 ones are actually quite comparable. Thus I found it more reasonable to use only two ResNetBlocks as the accuracy trade-off for the time is very acceptable.

1 Blocks	2 Blocks	3 Blocks	4 Blocks
98.2%	98.7%	98.9%	98.6%

```
MNISTResNetwork:
    (layers): SequentialLayer:
        (0): TorchConvLayer: Kernel: (5, 5) In Channels 1 Out Channels 6 Stride 1
        (1): MaxPoolLayer: kernel: 2 stride: 2
        (2): ReLULayer:
        (3): TorchConvLayer: Kernel: (5, 5) In Channels 6 Out Channels 16 Stride 1
        (4): ResNetBlock:
            (conv_layers): SequentialLayer:
                (0): TorchConvLayer: Kernel: (3, 3) In Channels 16 Out Channels 16 Stride 1
                (1): ReLULayer:
                (2): TorchConvLayer: Kernel: (3, 3) In Channels 16 Out Channels 16 Stride 1
            (add_layer): AddLayer:
            (relu2): ReLULayer:
        (5): ResNetBlock:
            (conv_layers): SequentialLayer:
                (0): TorchConvLayer: Kernel: (3, 3) In Channels 16 Out Channels 16 Stride 1
                (1): ReLULayer:
                (2): TorchConvLayer: Kernel: (3, 3) In Channels 16 Out Channels 16 Stride 1
            (add_layer): AddLayer:
            (relu2): ReLULayer:
        (6): ResNetBlock:
            (conv_layers): SequentialLayer:
                (0): TorchConvLayer: Kernel: (3, 3) In Channels 16 Out Channels 16 Stride 1
                (1): ReLULayer:
                (2): TorchConvLayer: 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, 128)
        (11): ReLULayer:
        (12): LinearLayer: (128, 64)
        (13): ReLULayer:
        (14): LinearLayer: (64, 10)
    (loss_layer): SoftmaxCrossEntropyLossLayer:
```

Figure 1: Architecture of MNistResNetwork with 3 ResNet Blocks

2. Do you get any improvement using a different non-linearity?

Among all non-linearity activation function, the **ReLU outperform others** using the same MNistResNetwork architecture and hyper-parameters. The table below shows the highest test accuracy achieved in 20 epochs.

ReLU(default)	Leaky ReLU (s=0.1)	PReLU
98.7%	98.0%	98.4%

3. Can you come up with an architecture which gets even higher accuracy?

The best result I get throughout assignment 2 is **99.23% Test Accuracy** using the following architecture and hyper-parameters.

```
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): ReLULayer:
        (8): FlattenLayer:
        (9): LinearLayer: (784, 128)
        (10): ReLULayer:
        (11): LinearLayer: (128, 64)
        (12): ReLULayer:
        (13): LinearLayer: (64, 10)
    (loss_layer): SoftmaxCrossEntropyLossLayer:
```

Figure 2: Architecture of MNistResNetwork giving 99.43% Test Accuracy

batch size	total epochs	learning rate	momentum	weight decay
100	40	0.01 for epoch [0,20) 0.001 for epoch [20,40)	0.9	0.0005

We set the first 20 epochs' learning rate to 0.01 and the later 20 epochs' learning rate to 0.001 which make our architecture improve by an amount of $\approx 0.3\%$.