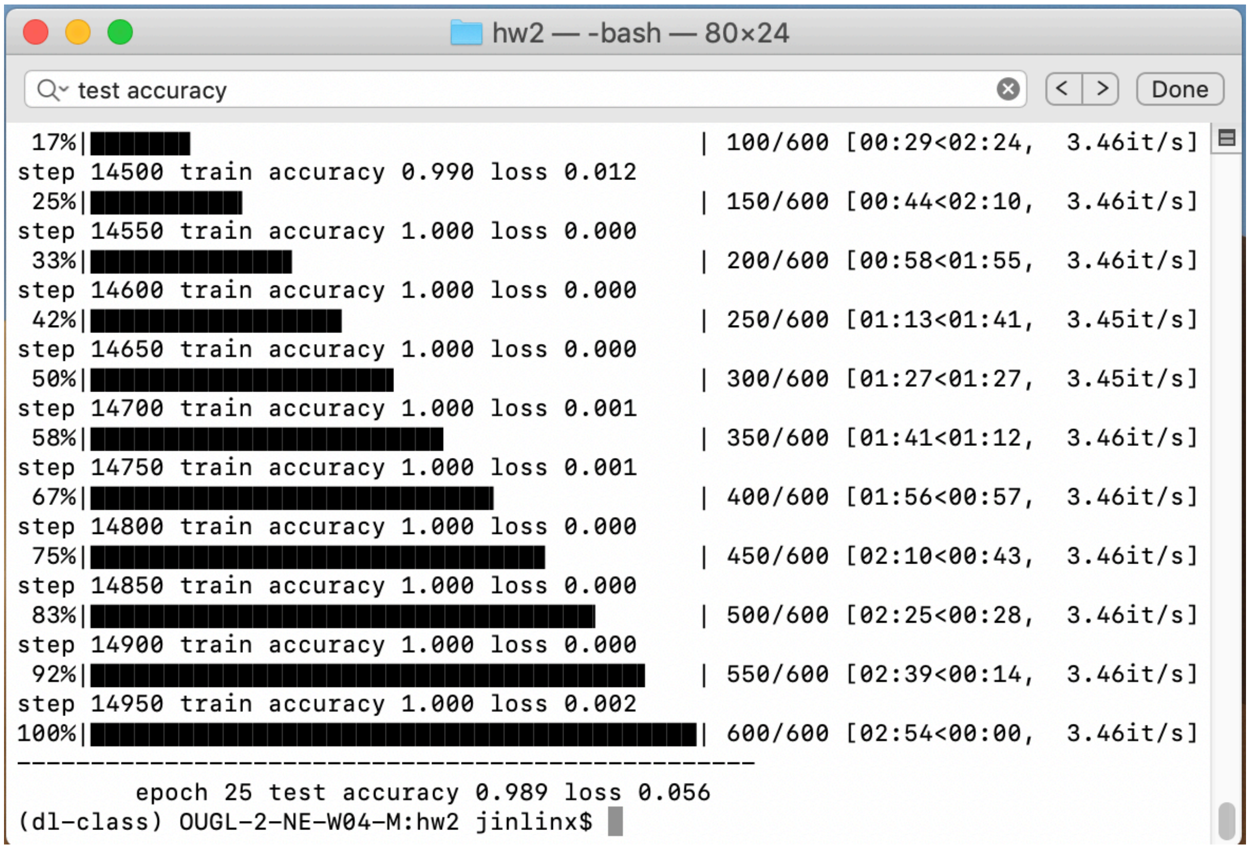
Short Answer：

1. See if you can improve the MNistResNetwork architecture using more ResNetBlocks. What's the highest accuracy you achieve? What is the architecture (you can paste the output from print(network)).

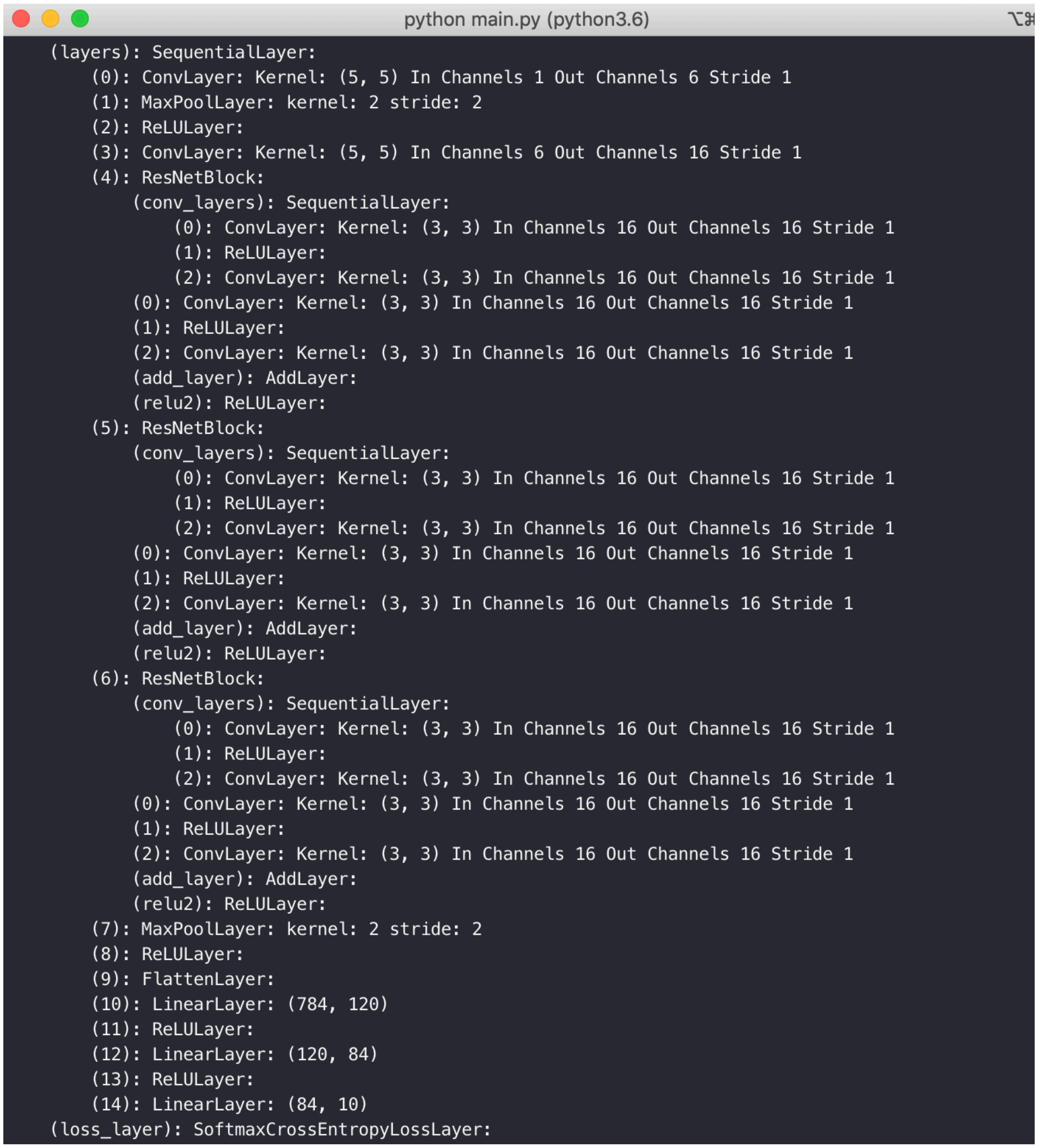
   I can improve the accuracy by using more ResNetBlocks, the highest is 0.991(after 25 epochs, the original one is 0.989 accuracy).(add one Resnet layer)

final result:



```
 17%|███████                                  | 100/600 [00:29<02:24,  3.46it/s]
step 14500 train accuracy 0.990 loss 0.012
 25%|██████████                               | 150/600 [00:44<02:10,  3.46it/s]
step 14550 train accuracy 1.000 loss 0.000
 33%|██████████████                           | 200/600 [00:58<01:55,  3.46it/s]
step 14600 train accuracy 1.000 loss 0.000
 42%|█████████████████                        | 250/600 [01:13<01:41,  3.45it/s]
step 14650 train accuracy 1.000 loss 0.000
 50%|█████████████████████                    | 300/600 [01:27<01:27,  3.45it/s]
step 14700 train accuracy 1.000 loss 0.001
 58%|████████████████████████                 | 350/600 [01:41<01:12,  3.46it/s]
step 14750 train accuracy 1.000 loss 0.001
 67%|████████████████████████████             | 400/600 [01:56<00:57,  3.46it/s]
step 14800 train accuracy 1.000 loss 0.000
 75%|███████████████████████████████          | 450/600 [02:10<00:43,  3.46it/s]
step 14850 train accuracy 1.000 loss 0.000
 83%|███████████████████████████████████      | 500/600 [02:25<00:28,  3.46it/s]
step 14900 train accuracy 1.000 loss 0.000
 92%|██████████████████████████████████████   | 550/600 [02:39<00:14,  3.46it/s]
step 14950 train accuracy 1.000 loss 0.002
100%|█████████████████████████████████████████| 600/600 [02:54<00:00,  3.46it/s]
--------------------------------------------------
        epoch 25 test accuracy 0.989 loss 0.056
(dl-class) OUGL-2-NE-W04-M:hw2 jinlinx$ 
```

MNISTResNetwork Print:

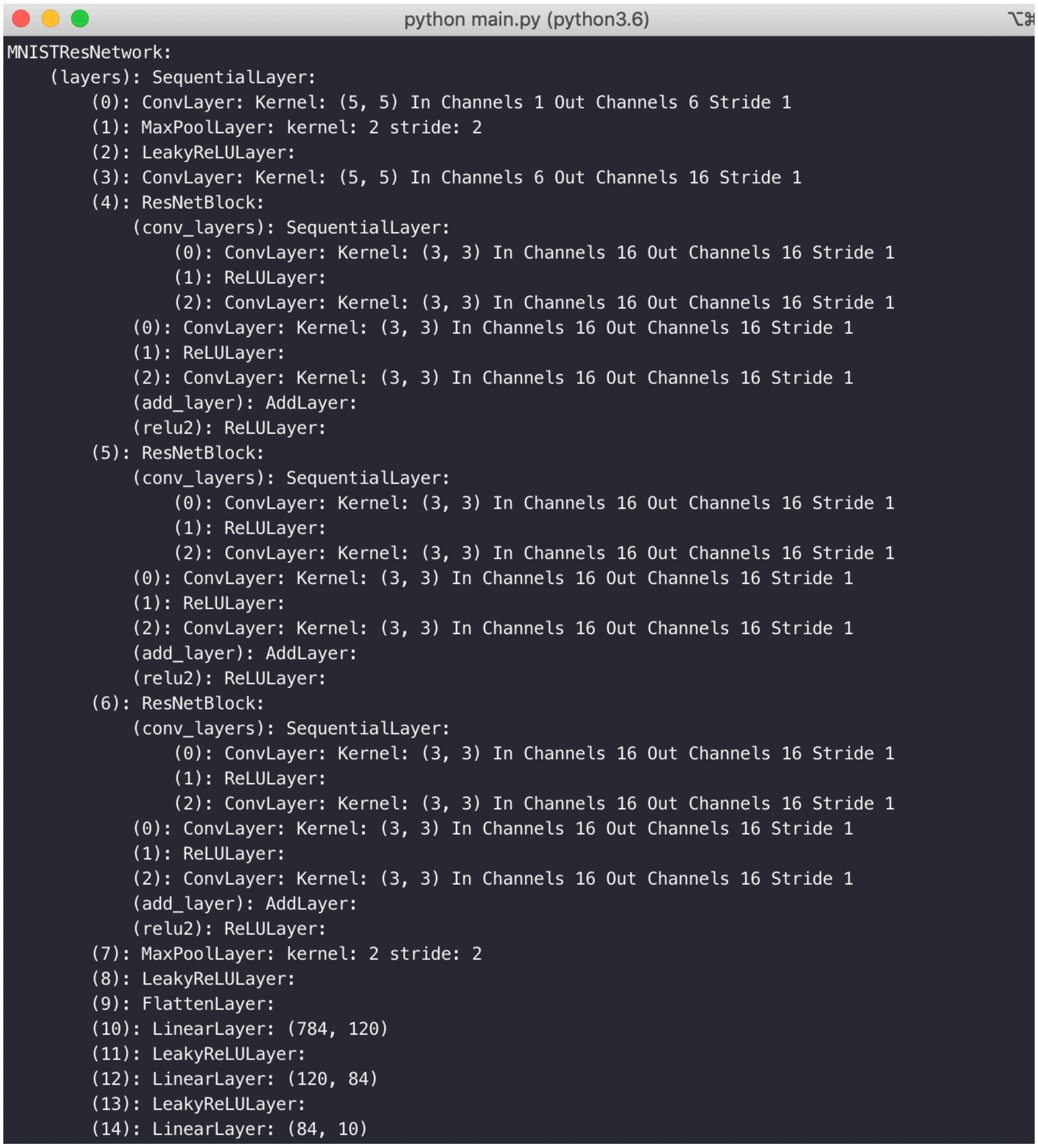


```
    (layers): SequentialLayer:
        (0): ConvLayer: Kernel: (5, 5) In Channels 1 Out Channels 6 Stride 1
        (1): MaxPoolLayer: kernel: 2 stride: 2
        (2): ReLULayer:
        (3): ConvLayer: Kernel: (5, 5) In Channels 6 Out Channels 16 Stride 1
        (4): ResNetBlock:
            (conv_layers): 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
            (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
            (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
            (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:
```

2. Do you get any improvement using a different non-linearity? Be sure to change it back to ReLU before you turn in your final code.

   I try other non-linerity like prelu, leaky-relu, but almost same accuracy with relu or a little better than the relu. Achieve 98.9%

MNISTResNetwork Print:

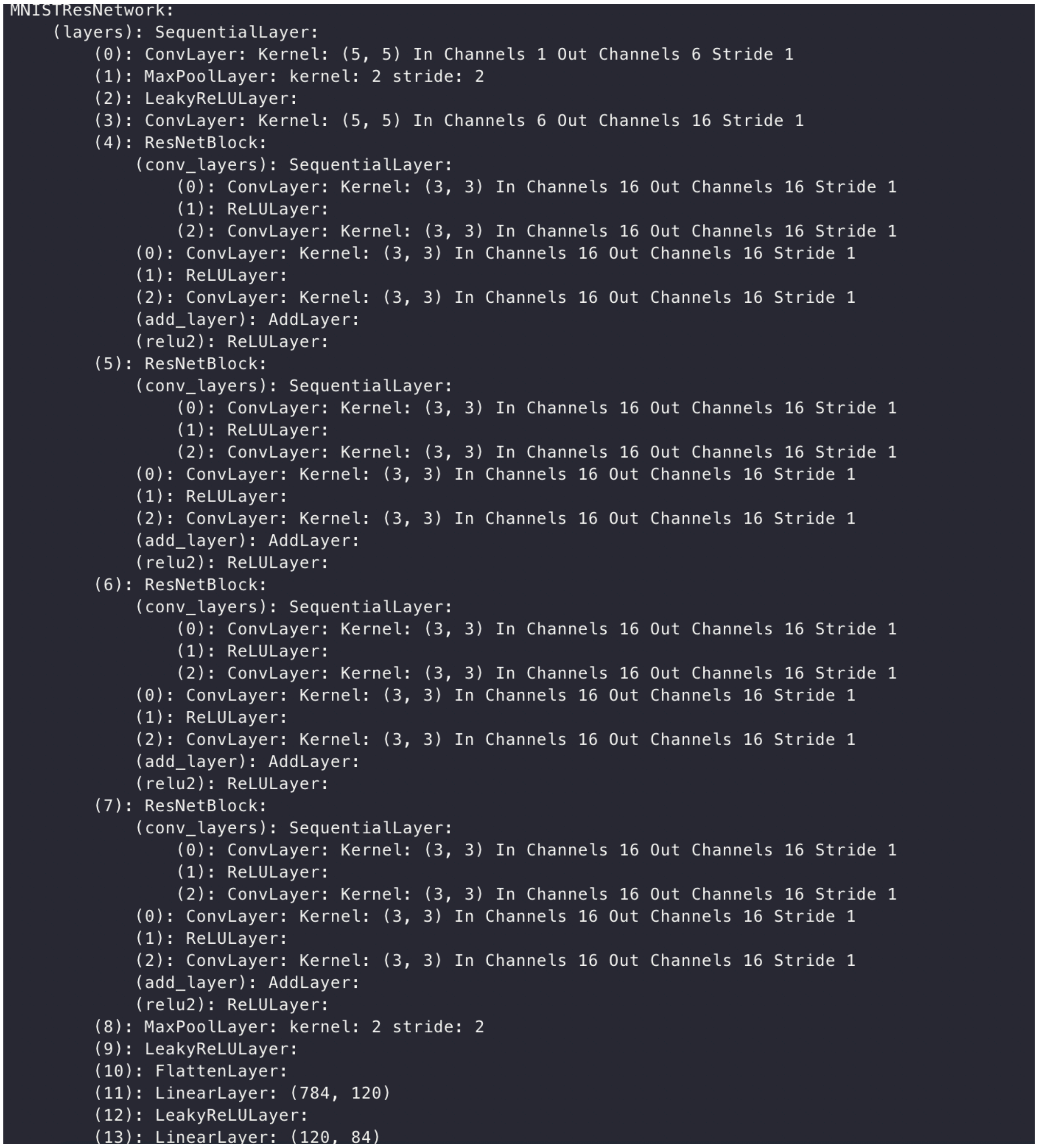


```
MNISTResNetwork:
    (layers): SequentialLayer:
        (0): ConvLayer: Kernel: (5, 5) In Channels 1 Out Channels 6 Stride 1
        (1): MaxPoolLayer: kernel: 2 stride: 2
        (2): LeakyReLULayer:
        (3): ConvLayer: Kernel: (5, 5) In Channels 6 Out Channels 16 Stride 1
        (4): ResNetBlock: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

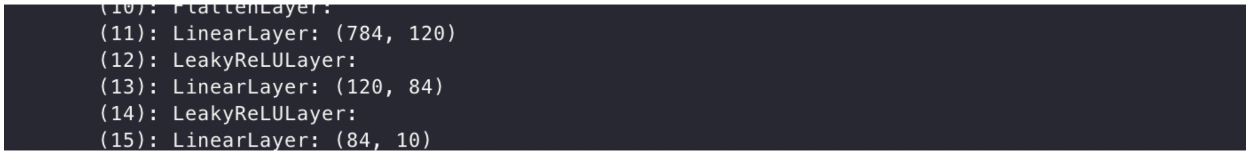
3. Can you come up with an architecture which gets even higher accuracy? Again, include the output from print(network).

   add one more resblock layer and change all relu to leaky rule layer.

like



```
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): ReLULayer:
                (2): ConvLayer: Kernel: (3, 3) In Channels 16 Out Channels 16 Stride 1
            (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
            (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
            (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
            (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): LeakyReLULayer:
        (10): FlattenLayer:
        (11): LinearLayer: (784, 120)
        (12): LeakyReLULayer:
        (13): LinearLayer: (120, 84)
```



```
        (10): FlattenLayer:
        (11): LinearLayer: (784, 120)
        (12): LeakyReLULayer:
        (13): LinearLayer: (120, 84)
```

```
        (14): LeakyReLULayer:
        (15): LinearLayer: (84, 10)
```