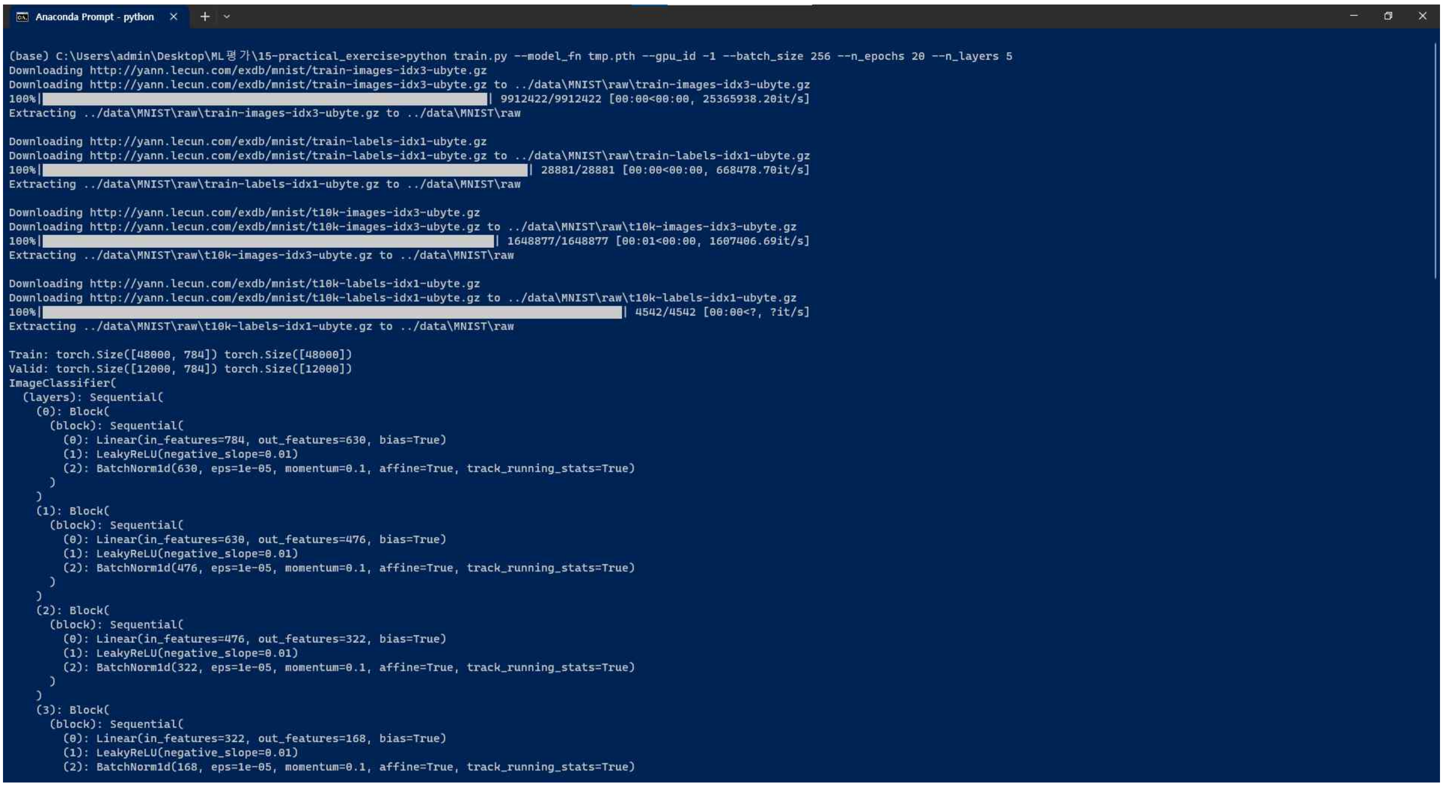
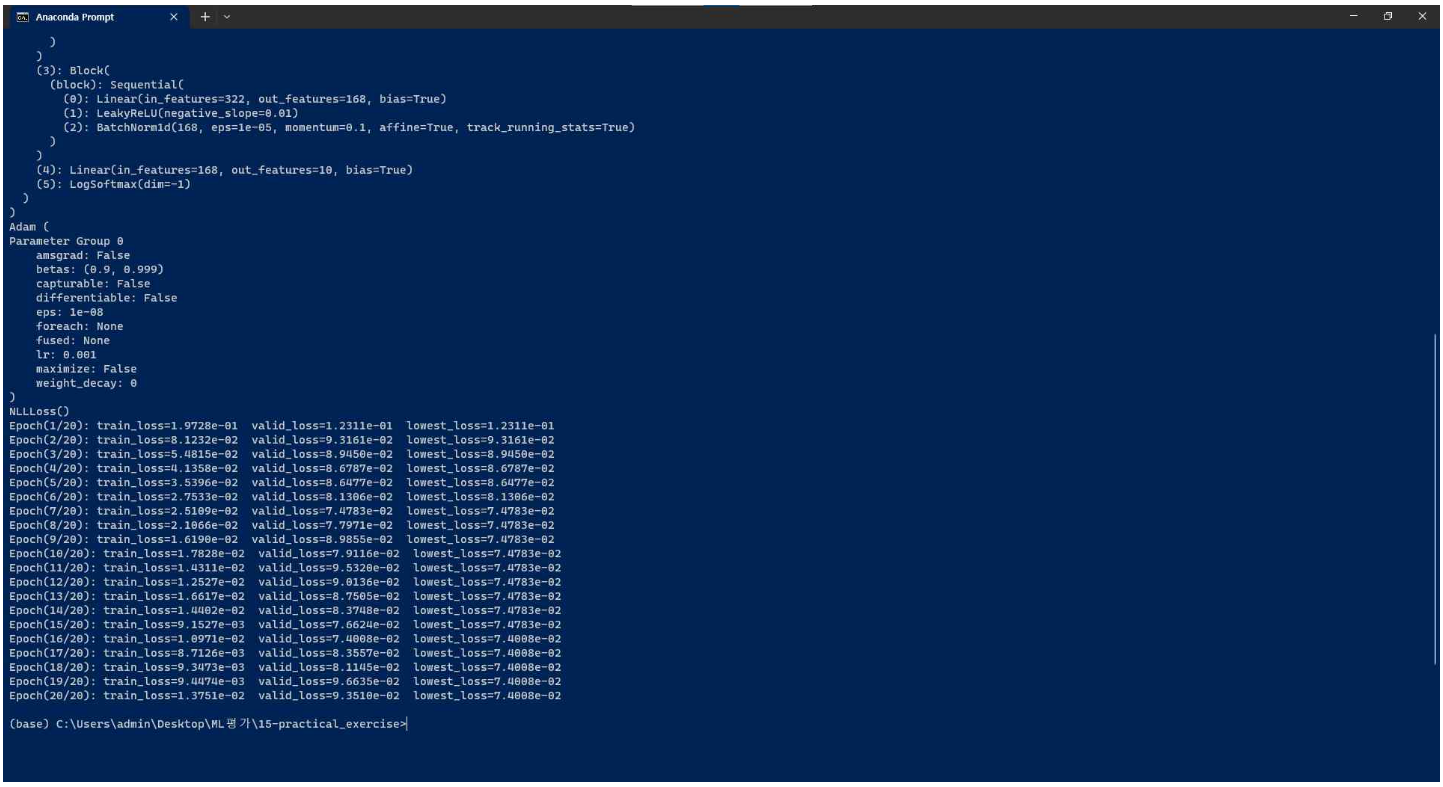
빅데이터 머신러닝(BigData ML) 평가

15장

<모델 학습 - tmp.pth>

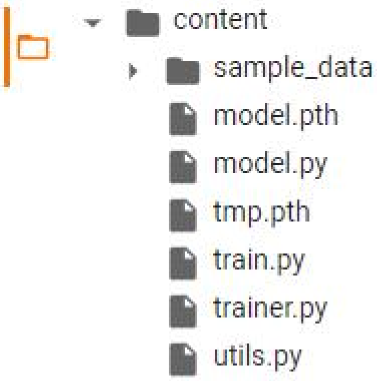


```
(base) C:\Users\admin\Desktop\ML평가\15-practical_exercise>python train.py --model_fn tmp.pth --gpu_id -1 --batch_size 256 --n_epochs 20 --n_layers 5
Downloading http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz to ../data\MNIST\raw\train-images-idx3-ubyte.gz
100%|██████████████████████████████████████████████████████████████| 9912422/9912422 [00:00<00:00, 25365938.20it/s]
Extracting ../data\MNIST\raw\train-images-idx3-ubyte.gz to ../data\MNIST\raw

Downloading http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz to ../data\MNIST\raw\train-labels-idx1-ubyte.gz
100%|████████████████████████████████████████████████████████████████████| 28881/28881 [00:00<00:00, 668478.70it/s]
Extracting ../data\MNIST\raw\train-labels-idx1-ubyte.gz to ../data\MNIST\raw

Downloading http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz to ../data\MNIST\raw\t10k-images-idx3-ubyte.gz
100%|███████████████████████████████████████████████████████████████| 1648877/1648877 [00:01<00:00, 1607406.69it/s]
Extracting ../data\MNIST\raw\t10k-images-idx3-ubyte.gz to ../data\MNIST\raw

Downloading http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz to ../data\MNIST\raw\t10k-labels-idx1-ubyte.gz
100%|█████████████████████████████████████████████████████████████████████████████████| 4542/4542 [00:00<?, ?it/s]
Extracting ../data\MNIST\raw\t10k-labels-idx1-ubyte.gz to ../data\MNIST\raw

Train: torch.Size([48000, 784]) torch.Size([48000])
Valid: torch.Size([12000, 784]) torch.Size([12000])
ImageClassifier(
  (layers): Sequential(
    (0): Block(
      (block): Sequential(
        (0): Linear(in_features=784, out_features=630, bias=True)
        (1): LeakyReLU(negative_slope=0.01)
        (2): BatchNorm1d(630, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
    (1): Block(
      (block): Sequential(
        (0): Linear(in_features=630, out_features=476, bias=True)
        (1): LeakyReLU(negative_slope=0.01)
        (2): BatchNorm1d(476, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
    (2): Block(
      (block): Sequential(
        (0): Linear(in_features=476, out_features=322, bias=True)
        (1): LeakyReLU(negative_slope=0.01)
        (2): BatchNorm1d(322, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
    (3): Block(
      (block): Sequential(
        (0): Linear(in_features=322, out_features=168, bias=True)
        (1): LeakyReLU(negative_slope=0.01)
        (2): BatchNorm1d(168, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
```



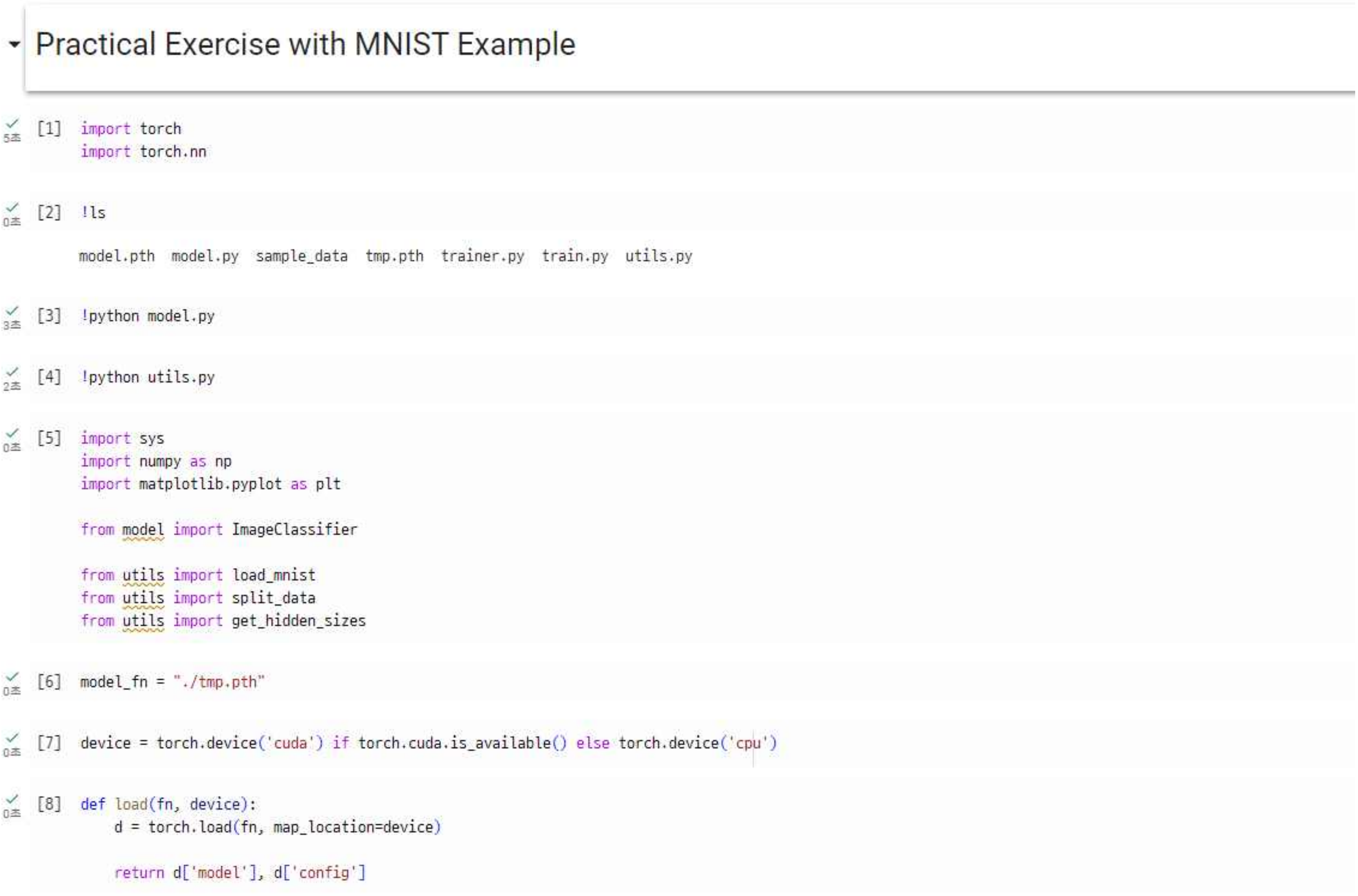
```
      )
    )
    (3): Block(
      (block): Sequential(
        (0): Linear(in_features=322, out_features=168, bias=True)
        (1): LeakyReLU(negative_slope=0.01)
        (2): BatchNorm1d(168, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
    (4): Linear(in_features=168, out_features=10, bias=True)
    (5): LogSoftmax(dim=-1)
  )
)
Adam (
Parameter Group 0
    amsgrad: False
    betas: (0.9, 0.999)
    capturable: False
    differentiable: False
    eps: 1e-08
    foreach: None
    fused: None
    lr: 0.001
    maximize: False
    weight_decay: 0
)
NLLLoss()
Epoch(1/20): train_loss=1.9728e-01  valid_loss=1.2311e-01  lowest_loss=1.2311e-01
Epoch(2/20): train_loss=8.1232e-02  valid_loss=9.3161e-02  lowest_loss=9.3161e-02
Epoch(3/20): train_loss=5.4815e-02  valid_loss=8.9450e-02  lowest_loss=8.9450e-02
Epoch(4/20): train_loss=4.1358e-02  valid_loss=8.6787e-02  lowest_loss=8.6787e-02
Epoch(5/20): train_loss=3.5396e-02  valid_loss=8.6477e-02  lowest_loss=8.6477e-02
Epoch(6/20): train_loss=2.7533e-02  valid_loss=8.1306e-02  lowest_loss=8.1306e-02
Epoch(7/20): train_loss=2.5109e-02  valid_loss=7.4783e-02  lowest_loss=7.4783e-02
Epoch(8/20): train_loss=2.1066e-02  valid_loss=7.7971e-02  lowest_loss=7.4783e-02
Epoch(9/20): train_loss=1.6190e-02  valid_loss=8.9855e-02  lowest_loss=7.4783e-02
Epoch(10/20): train_loss=1.7828e-02  valid_loss=7.9116e-02  lowest_loss=7.4783e-02
Epoch(11/20): train_loss=1.4311e-02  valid_loss=9.5320e-02  lowest_loss=7.4783e-02
Epoch(12/20): train_loss=1.2527e-02  valid_loss=9.0136e-02  lowest_loss=7.4783e-02
Epoch(13/20): train_loss=1.6617e-02  valid_loss=8.7505e-02  lowest_loss=7.4783e-02
Epoch(14/20): train_loss=1.4402e-02  valid_loss=8.3748e-02  lowest_loss=7.4783e-02
Epoch(15/20): train_loss=9.1527e-03  valid_loss=7.6624e-02  lowest_loss=7.4783e-02
Epoch(16/20): train_loss=1.0971e-02  valid_loss=7.4008e-02  lowest_loss=7.4008e-02
Epoch(17/20): train_loss=8.7126e-03  valid_loss=8.3557e-02  lowest_loss=7.4008e-02
Epoch(18/20): train_loss=9.3473e-03  valid_loss=8.1145e-02  lowest_loss=7.4008e-02
Epoch(19/20): train_loss=9.4474e-03  valid_loss=9.6635e-02  lowest_loss=7.4008e-02
Epoch(20/20): train_loss=1.3751e-02  valid_loss=9.3510e-02  lowest_loss=7.4008e-02

(base) C:\Users\admin\Desktop\ML평가\15-practical_exercise>
```

<content 폴더 파일 삽입>



```
content
  sample_data
  model.pth
  model.py
  tmp.pth
  train.py
  trainer.py
  utils.py
```

<predict.ipynb 구동>



Practical Exercise with MNIST Example

```
[1] import torch
    import torch.nn
```

```
[2] !ls

    model.pth  model.py  sample_data  tmp.pth  trainer.py  train.py  utils.py
```

```
[3] !python model.py
```

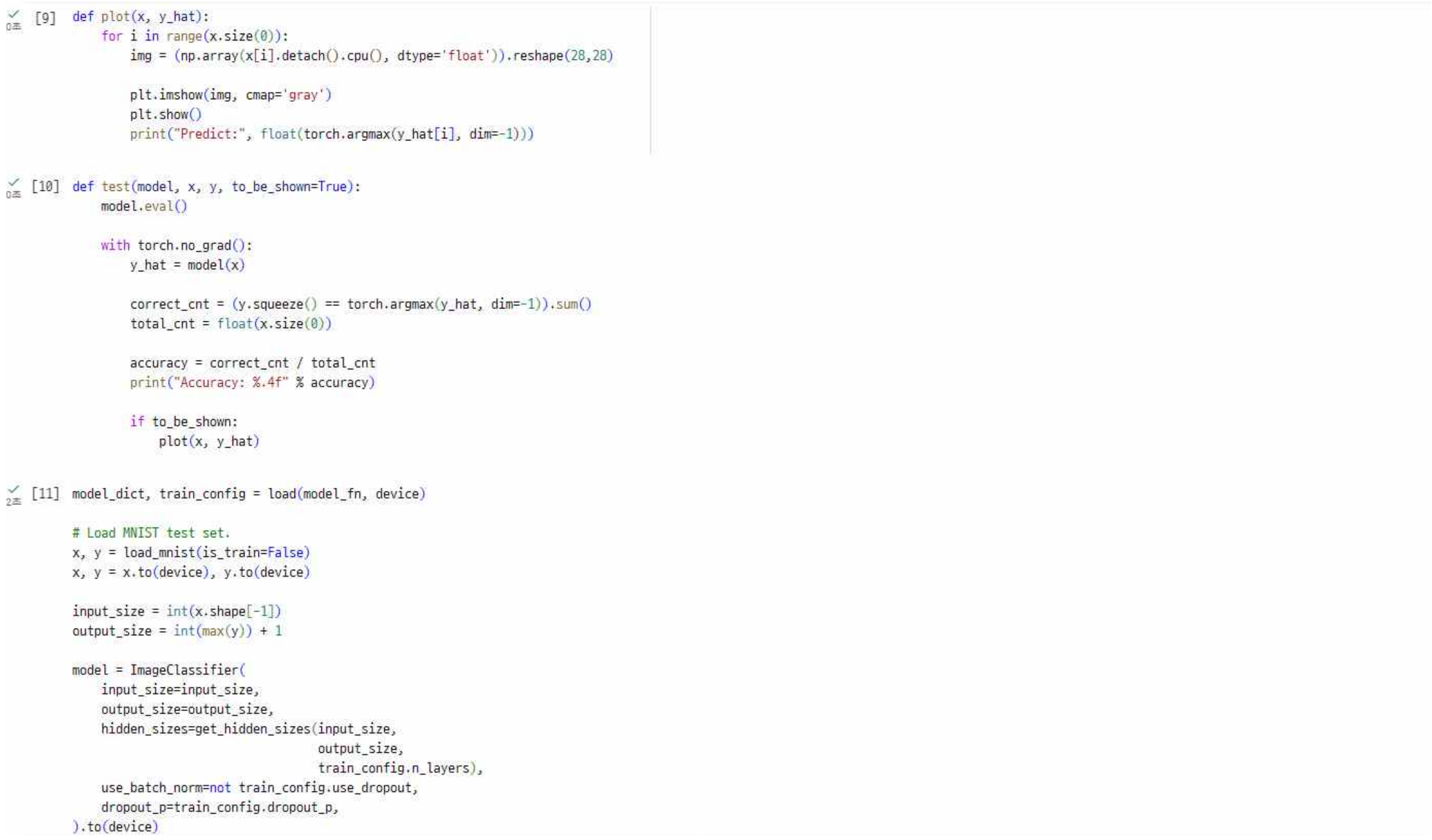
```
[4] !python utils.py
```

```
[5] import sys
    import numpy as np
    import matplotlib.pyplot as plt

    from model import ImageClassifier

    from utils import load_mnist
    from utils import split_data
    from utils import get_hidden_sizes
```

```
[6] model_fn = "./tmp.pth"
```

```
[7] device = torch.device('cuda') if torch.cuda.is_available() else torch.device('cpu')
```

```
[8] def load(fn, device):
        d = torch.load(fn, map_location=device)

        return d['model'], d['config']
```



✓ 0초 [9]

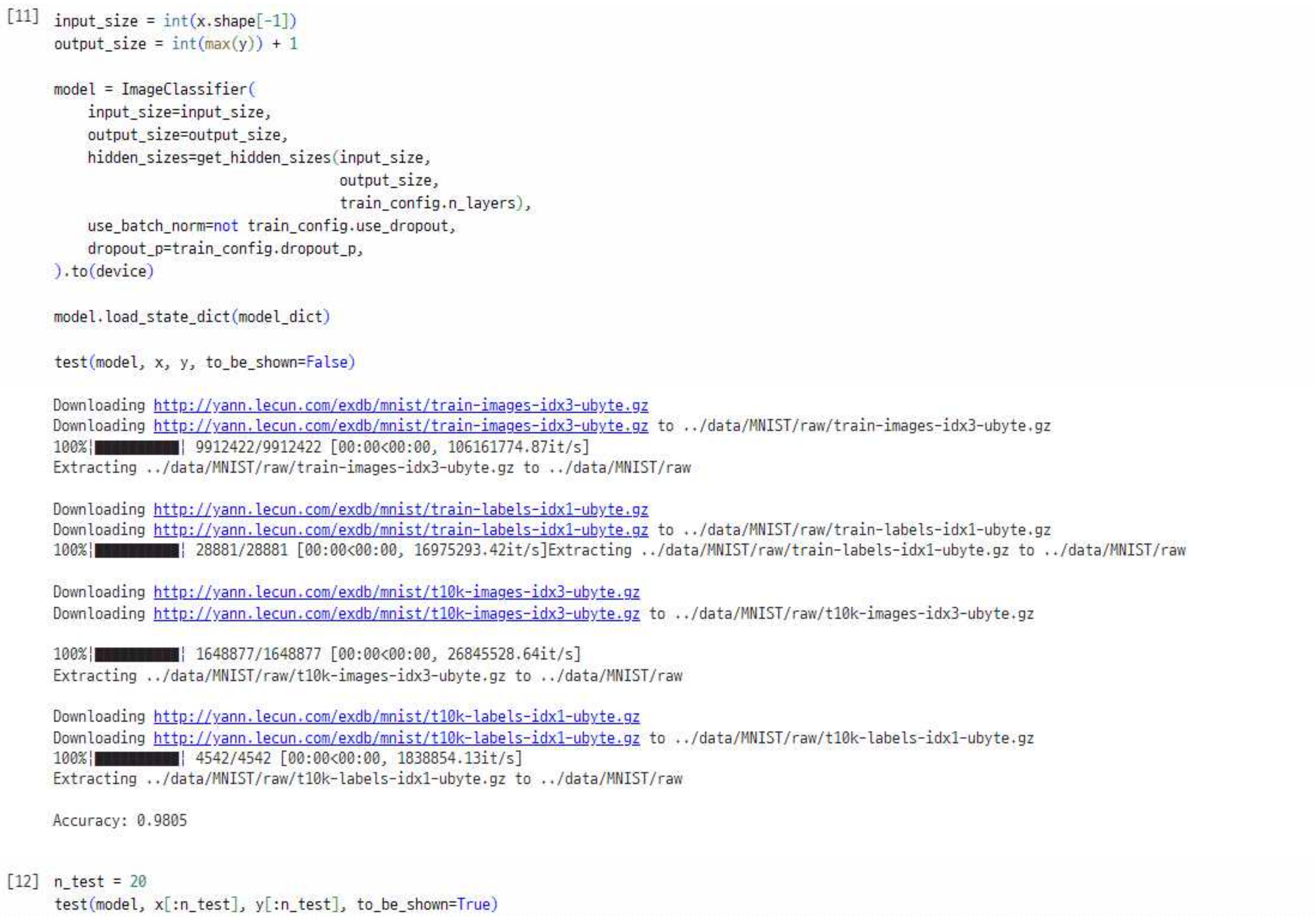
```
def plot(x, y_hat):
    for i in range(x.size(0)):
        img = (np.array(x[i].detach().cpu(), dtype='float')).reshape(28,28)

        plt.imshow(img, cmap='gray')
        plt.show()
        print("Predict:", float(torch.argmax(y_hat[i], dim=-1)))
```

✓ 0초 [10]

```
def test(model, x, y, to_be_shown=True):
    model.eval()

    with torch.no_grad():
        y_hat = model(x)

        correct_cnt = (y.squeeze() == torch.argmax(y_hat, dim=-1)).sum()
        total_cnt = float(x.size(0))

        accuracy = correct_cnt / total_cnt
        print("Accuracy: %.4f" % accuracy)

        if to_be_shown:
            plot(x, y_hat)
```

✓ 2초 [11]

```
model_dict, train_config = load(model_fn, device)

# Load MNIST test set.
x, y = load_mnist(is_train=False)
x, y = x.to(device), y.to(device)

input_size = int(x.shape[-1])
output_size = int(max(y)) + 1

model = ImageClassifier(
    input_size=input_size,
    output_size=output_size,
    hidden_sizes=get_hidden_sizes(input_size,
                                  output_size,
                                  train_config.n_layers),
    use_batch_norm=not train_config.use_dropout,
    dropout_p=train_config.dropout_p,
).to(device)
```



[11]

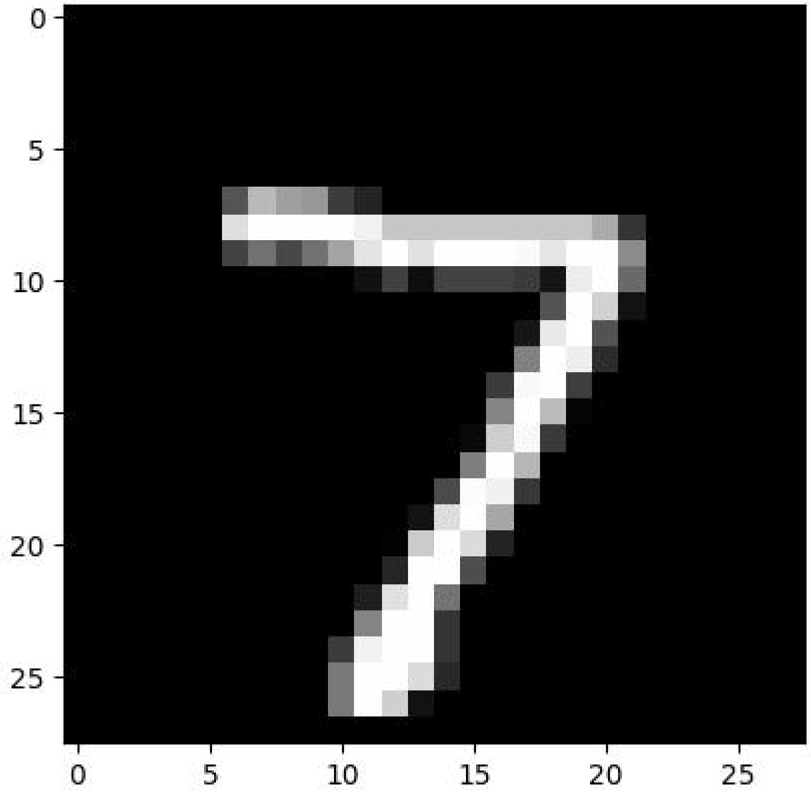
```
input_size = int(x.shape[-1])
output_size = int(max(y)) + 1

model = ImageClassifier(
    input_size=input_size,
    output_size=output_size,
    hidden_sizes=get_hidden_sizes(input_size,
                                  output_size,
                                  train_config.n_layers),
    use_batch_norm=not train_config.use_dropout,
    dropout_p=train_config.dropout_p,
).to(device)

model.load_state_dict(model_dict)

test(model, x, y, to_be_shown=False)
```

```
Downloading http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz to ../data/MNIST/raw/train-images-idx3-ubyte.gz
100%|██████████| 9912422/9912422 [00:00<00:00, 106161774.87it/s]
Extracting ../data/MNIST/raw/train-images-idx3-ubyte.gz to ../data/MNIST/raw

Downloading http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz to ../data/MNIST/raw/train-labels-idx1-ubyte.gz
100%|██████████| 28881/28881 [00:00<00:00, 16975293.42it/s]Extracting ../data/MNIST/raw/train-labels-idx1-ubyte.gz to ../data/MNIST/raw

Downloading http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz to ../data/MNIST/raw/t10k-images-idx3-ubyte.gz

100%|██████████| 1648877/1648877 [00:00<00:00, 26845528.64it/s]
Extracting ../data/MNIST/raw/t10k-images-idx3-ubyte.gz to ../data/MNIST/raw

Downloading http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz to ../data/MNIST/raw/t10k-labels-idx1-ubyte.gz
100%|██████████| 4542/4542 [00:00<00:00, 1838854.13it/s]
Extracting ../data/MNIST/raw/t10k-labels-idx1-ubyte.gz to ../data/MNIST/raw

Accuracy: 0.9805
```

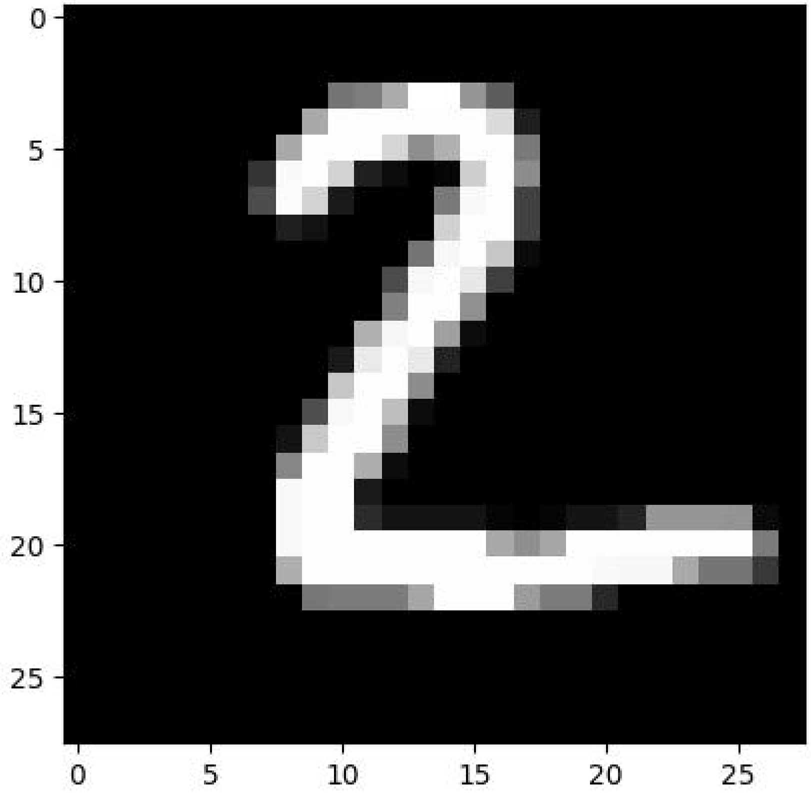
[12]

```
n_test = 20
test(model, x[:n_test], y[:n_test], to_be_shown=True)
```

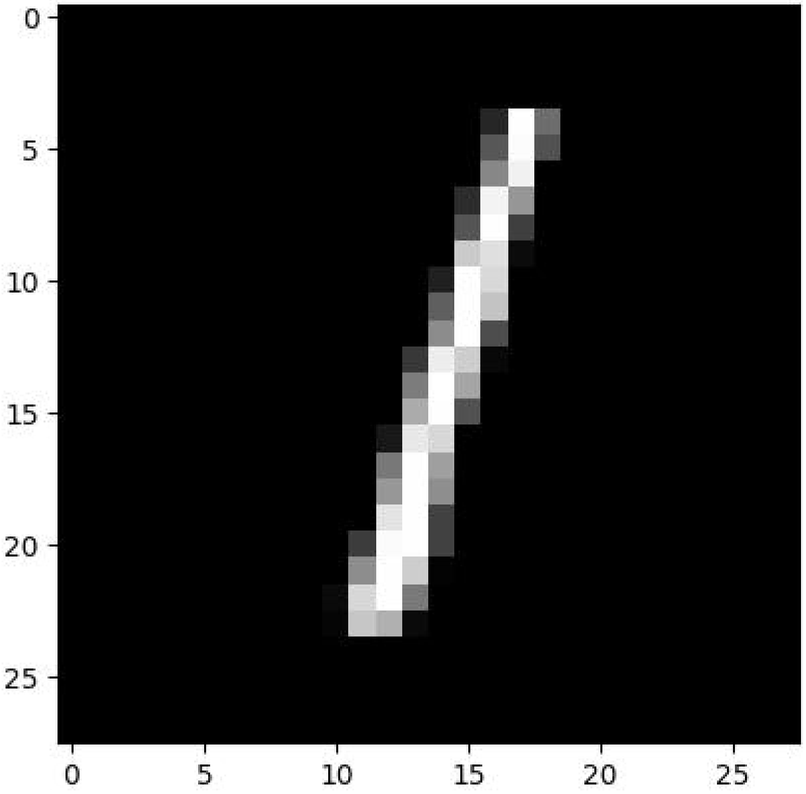
```
Accuracy: 0.9500
```



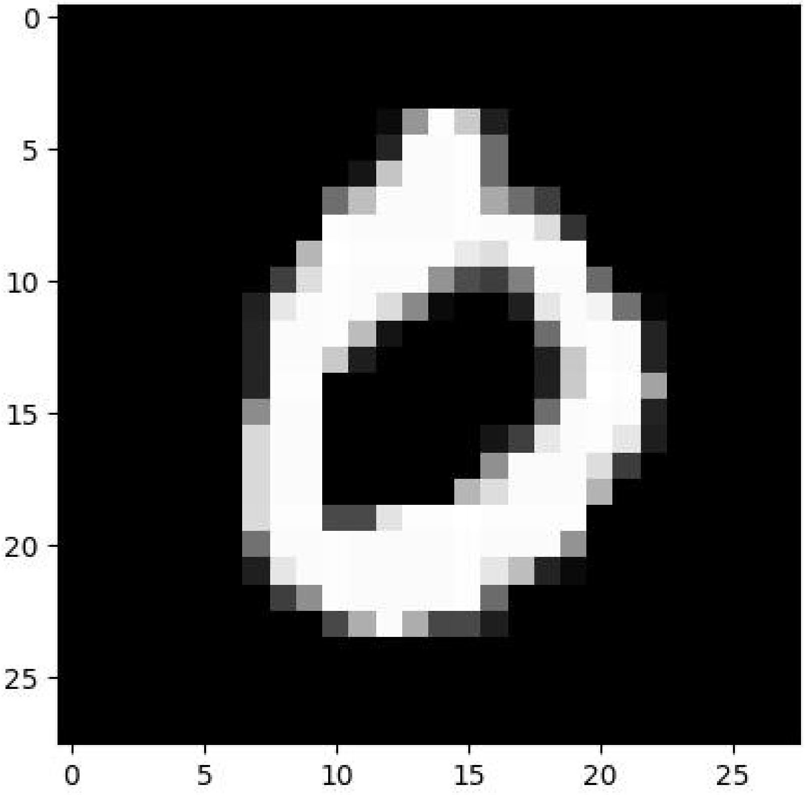
```
Predict: 7.0
```



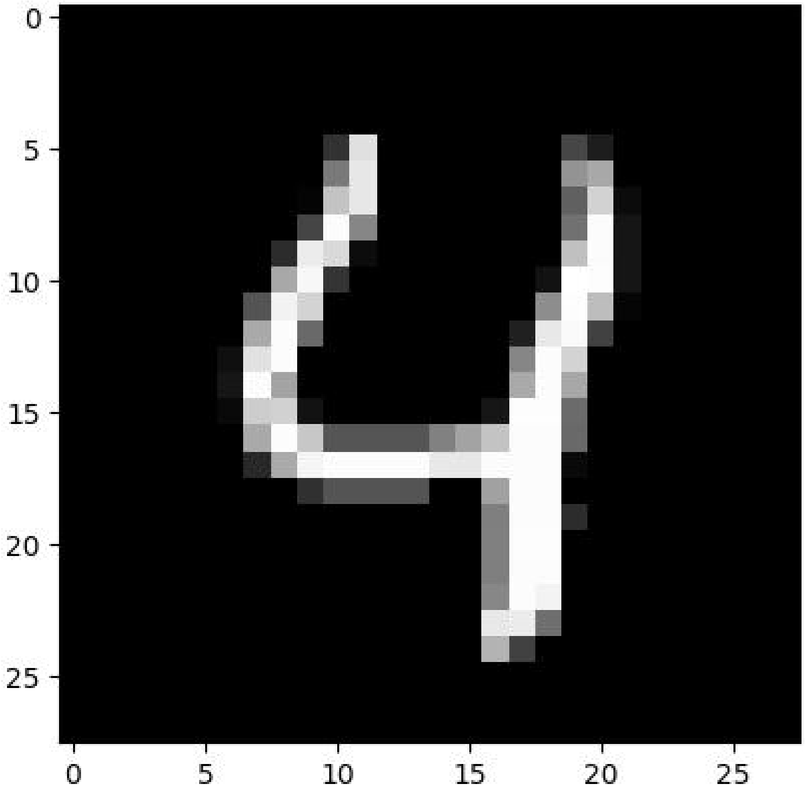
```
Predict: 2.0
```



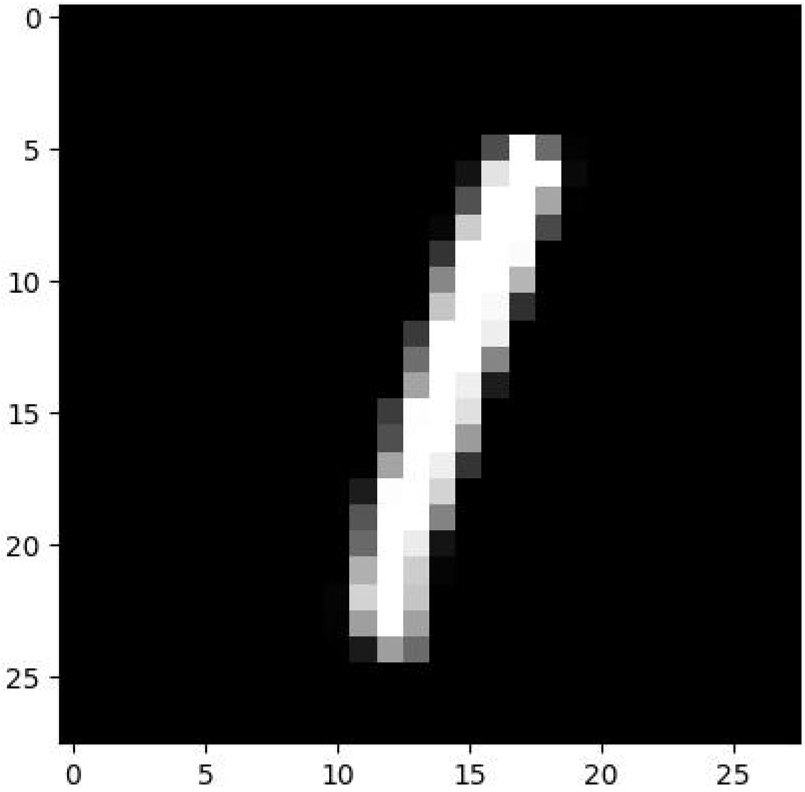
```
Predict: 1.0
```



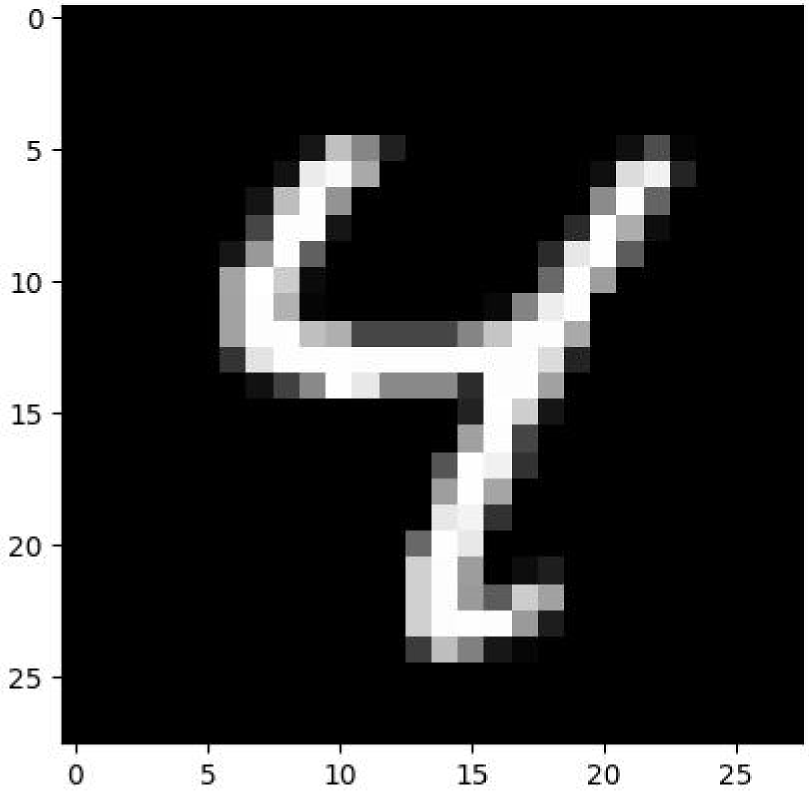
Predict: 0.0



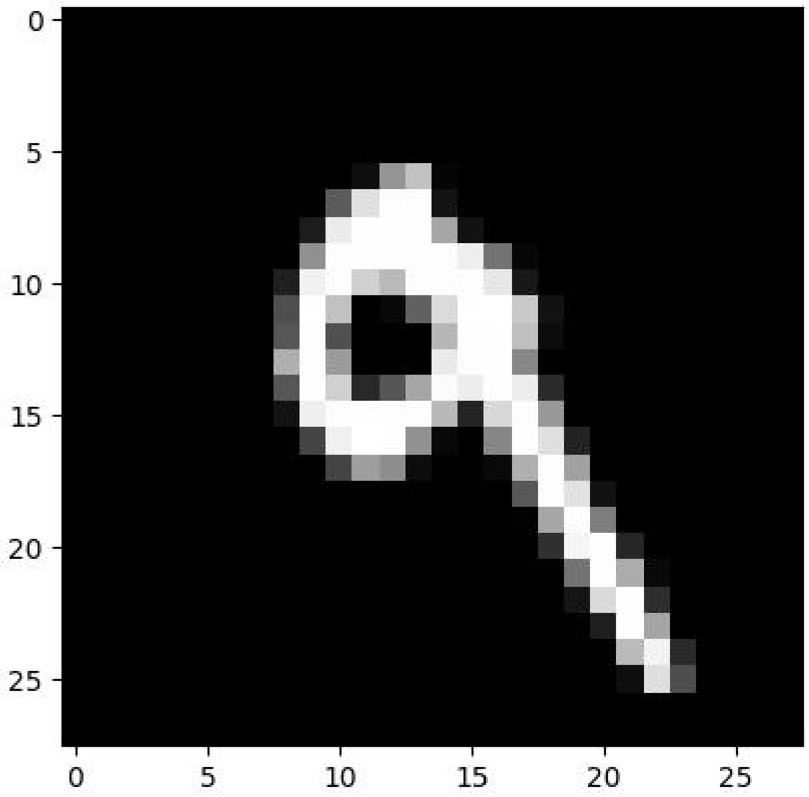
Predict: 4.0



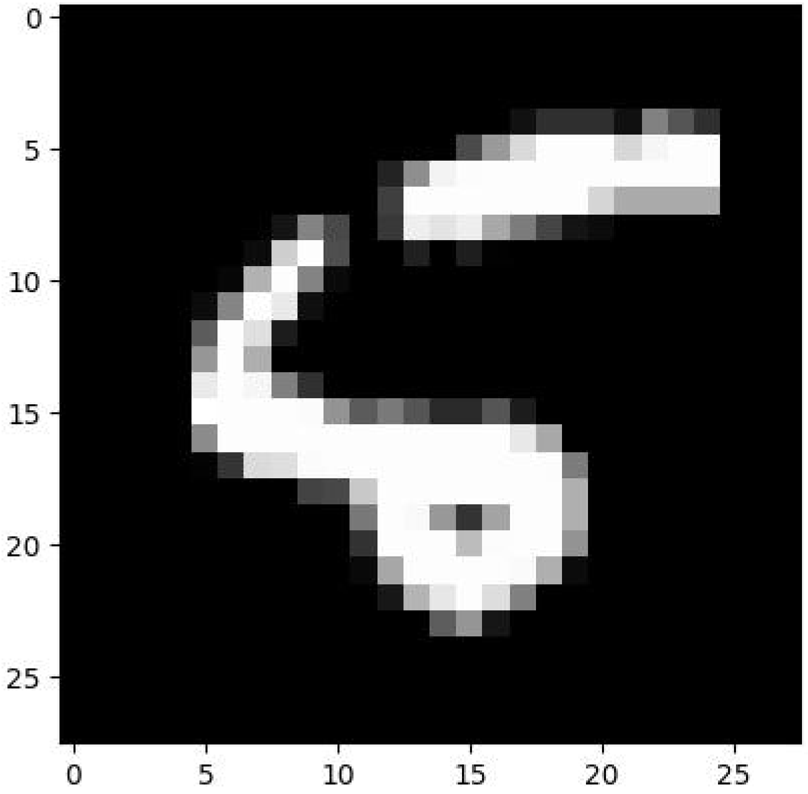
Predict: 1.0



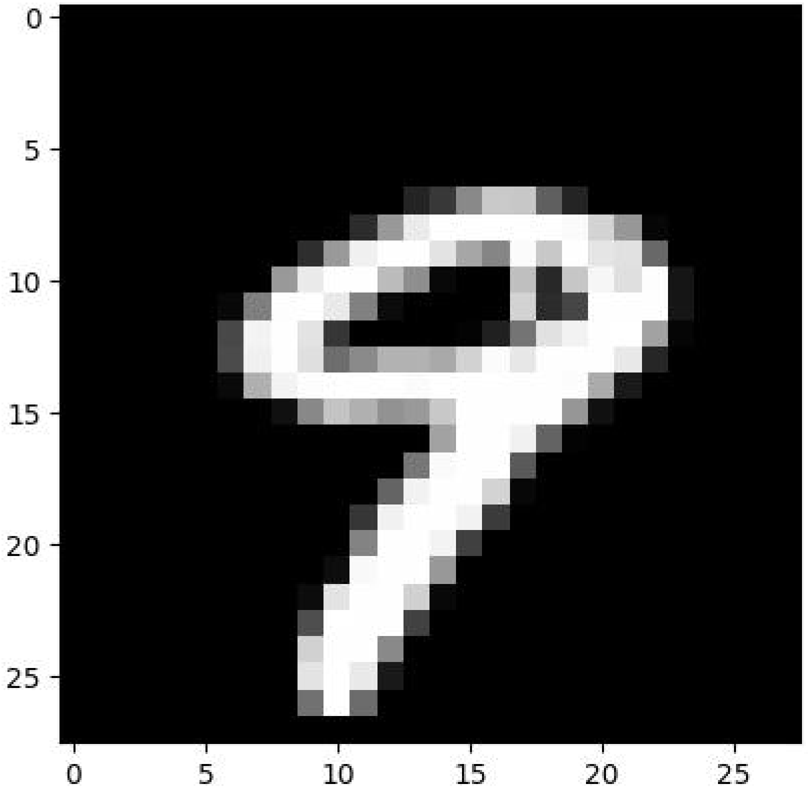
Predict: 4.0



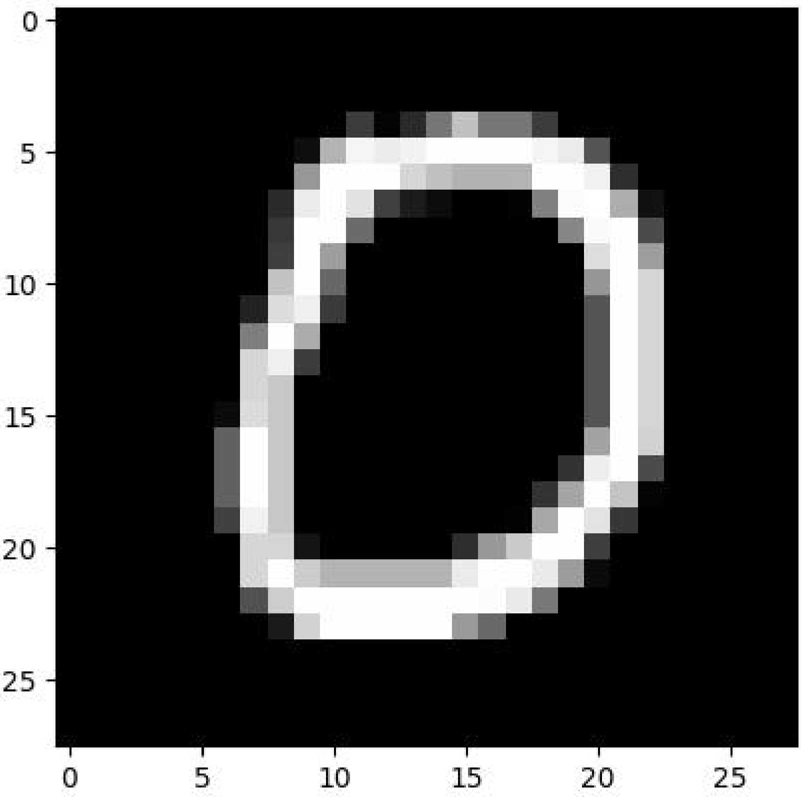
```
Predict: 9.0
```



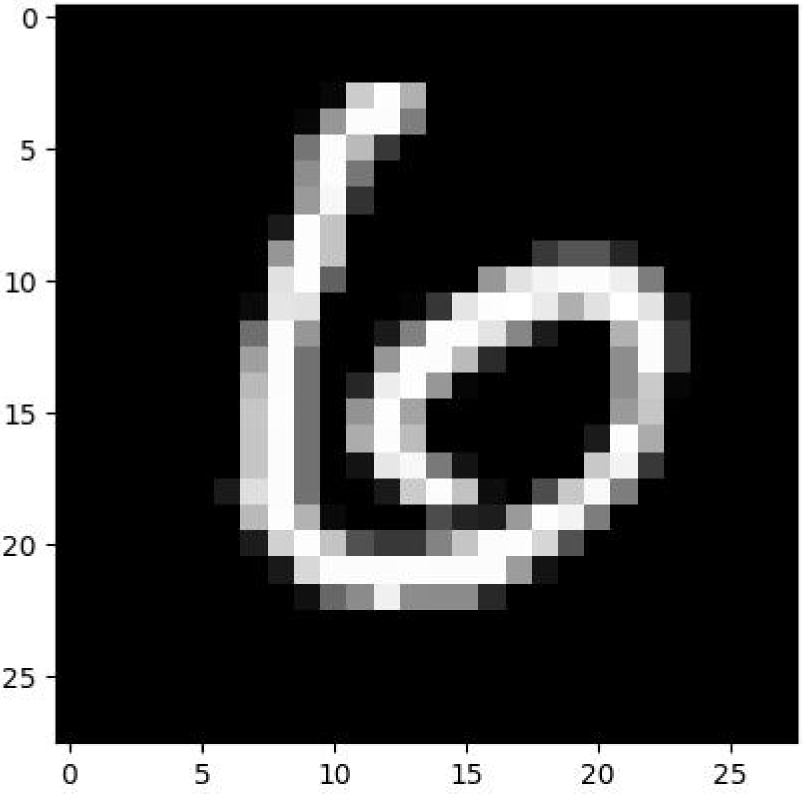
```
Predict: 5.0
```



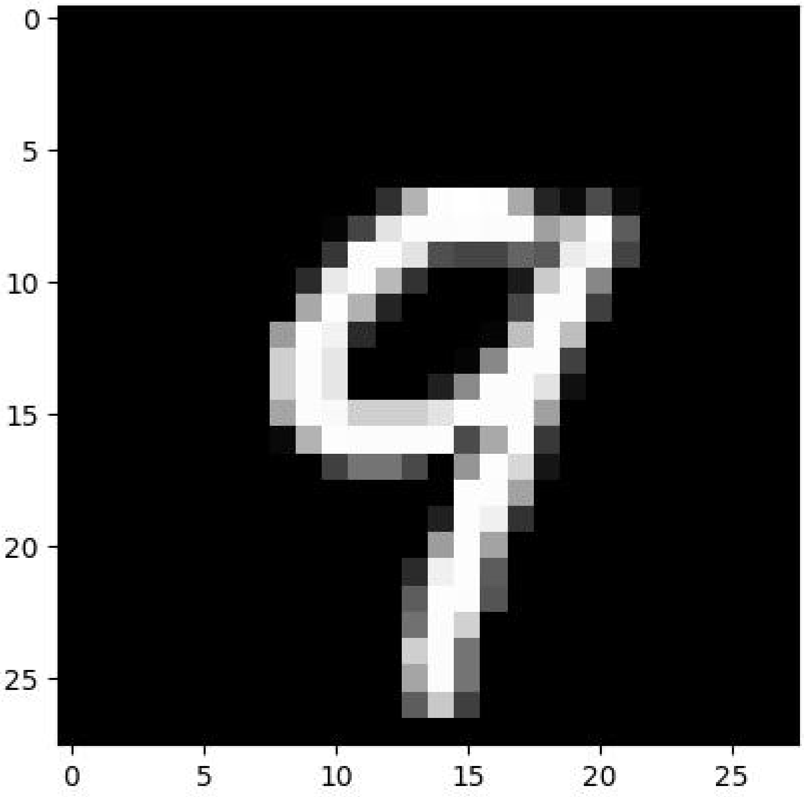
Predict: 9.0



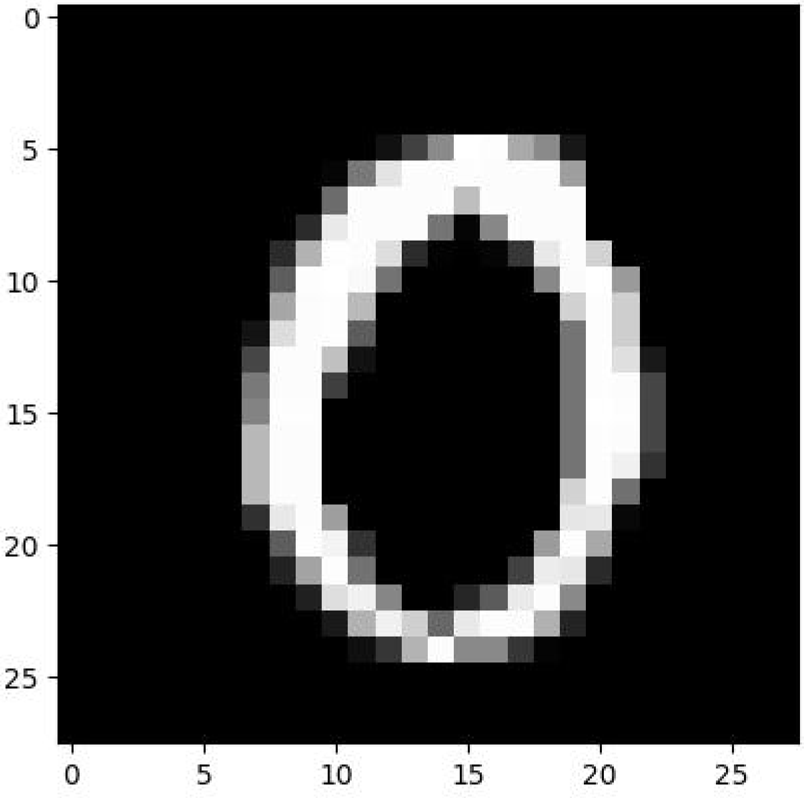
Predict: 0.0



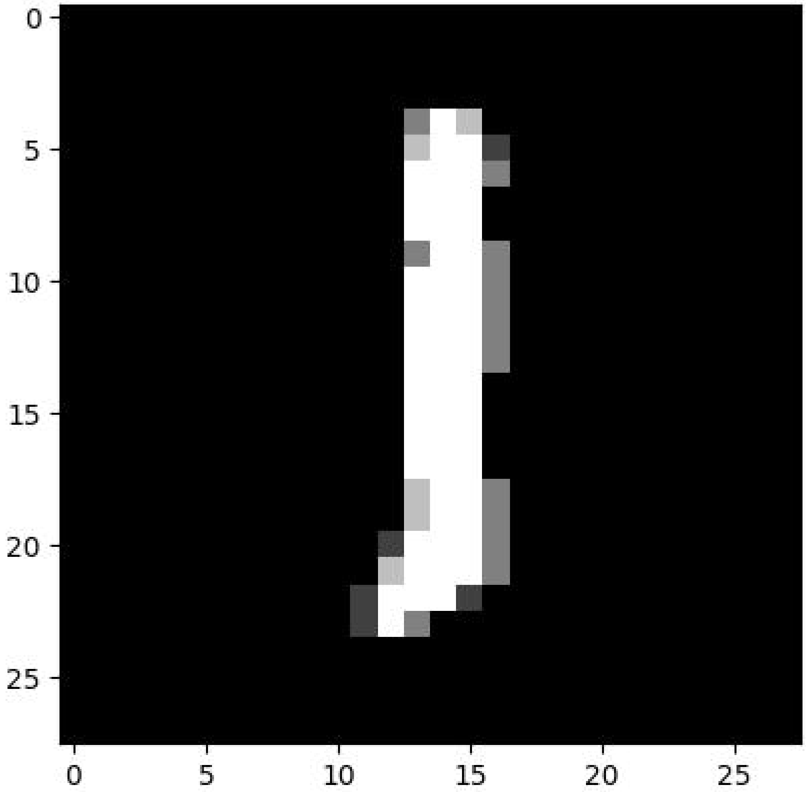
```
Predict: 6.0
```



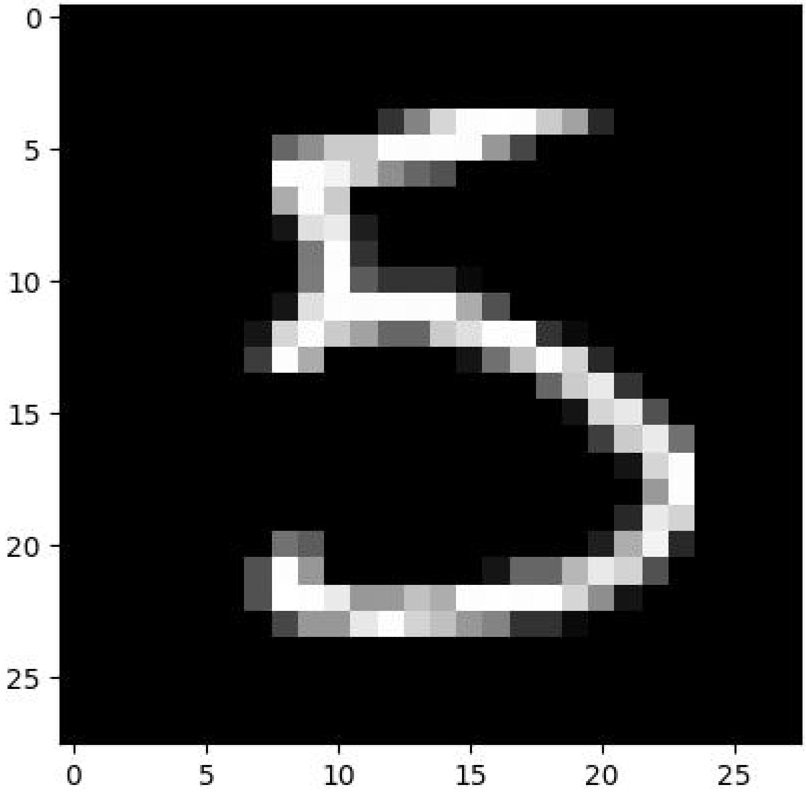
```
Predict: 9.0
```



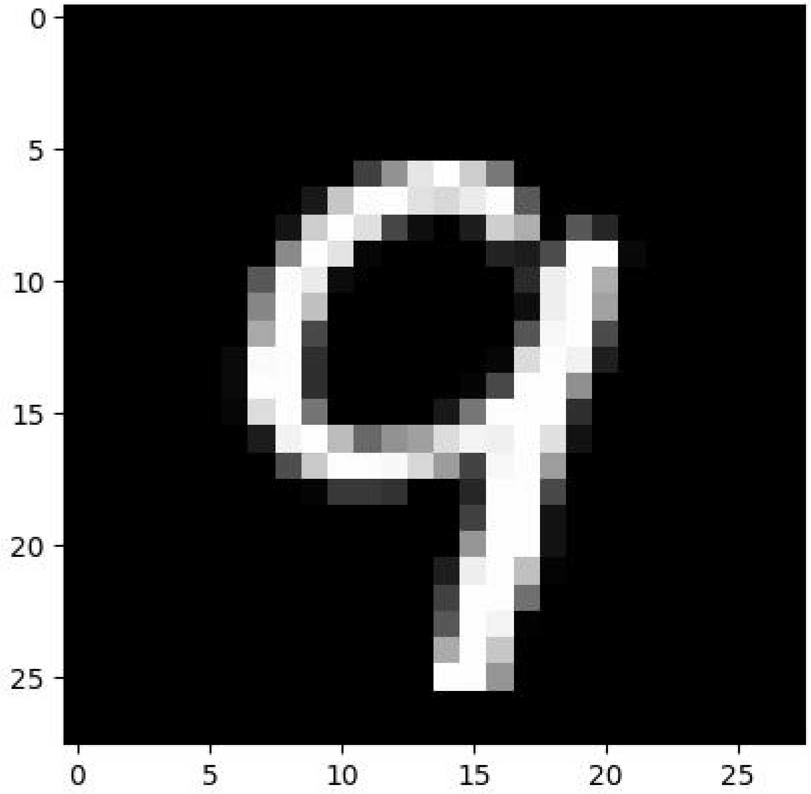
Predict: 0.0



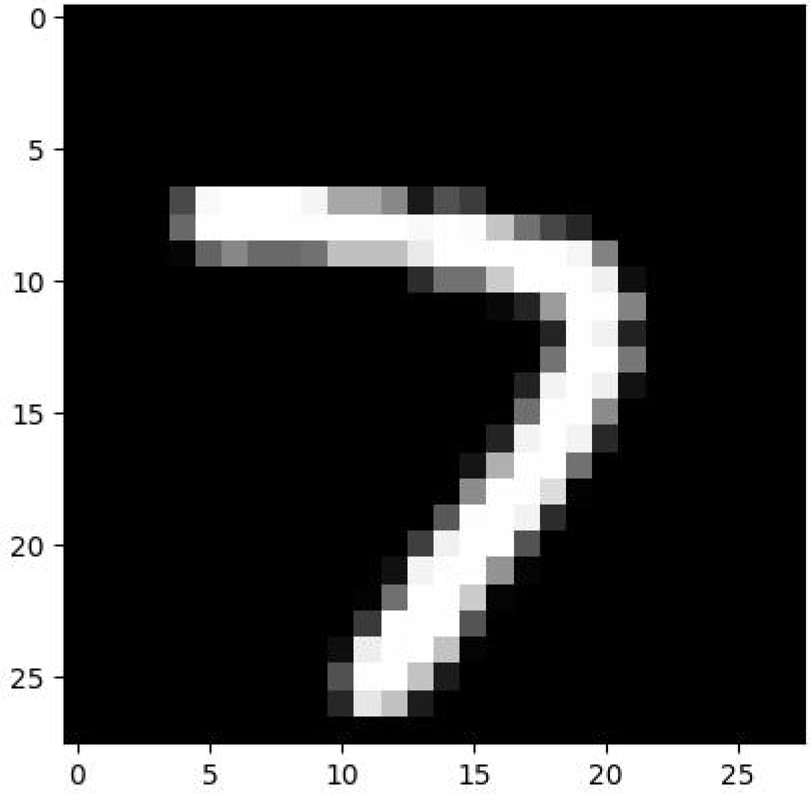
Predict: 1.0



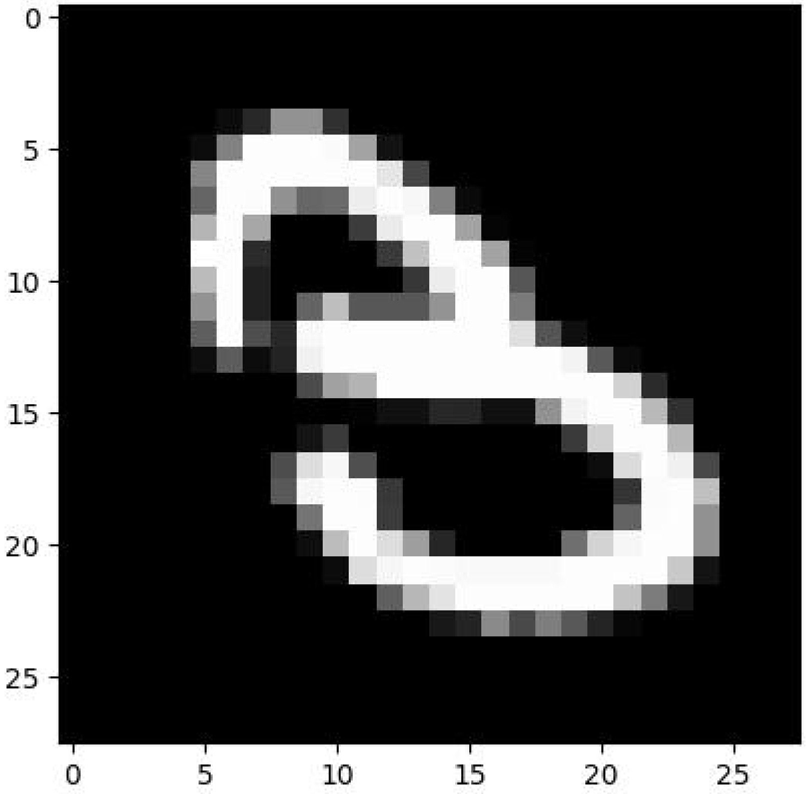
```
Predict: 3.0
```



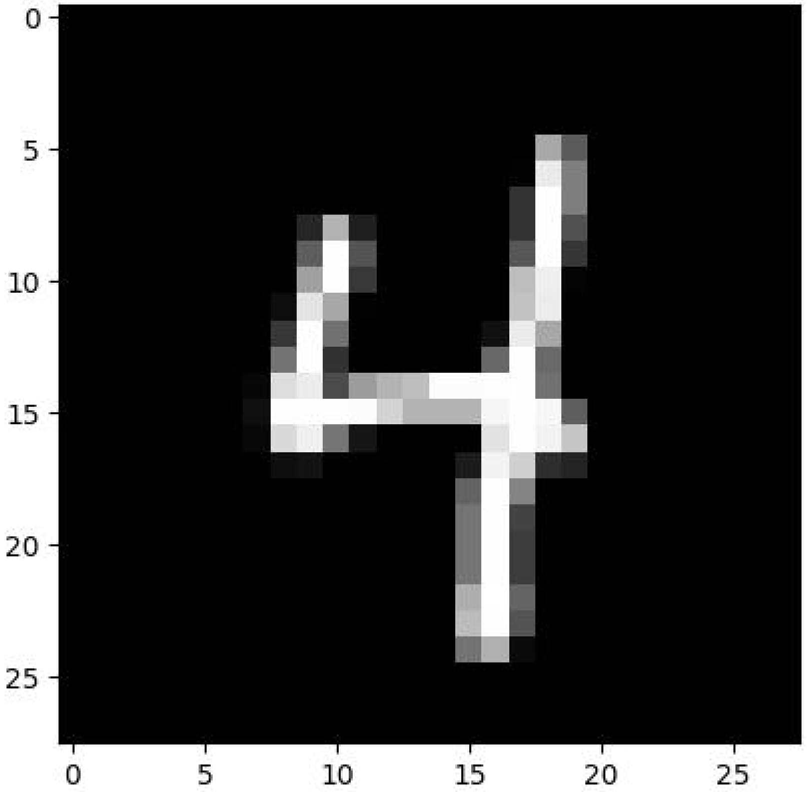
```
Predict: 9.0
```



```
Predict: 7.0
```

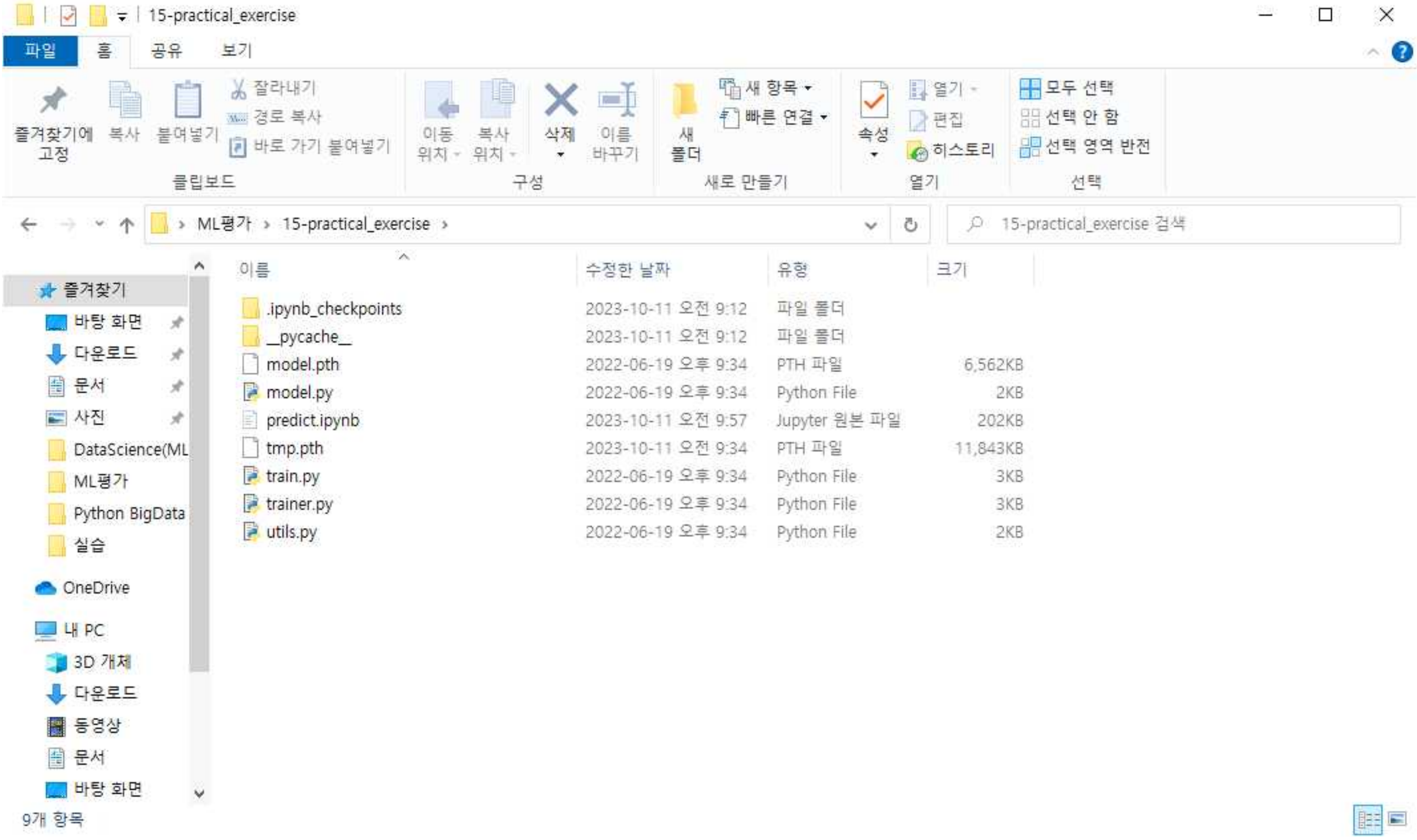


```
Predict: 3.0
```



```
Predict: 4.0
```

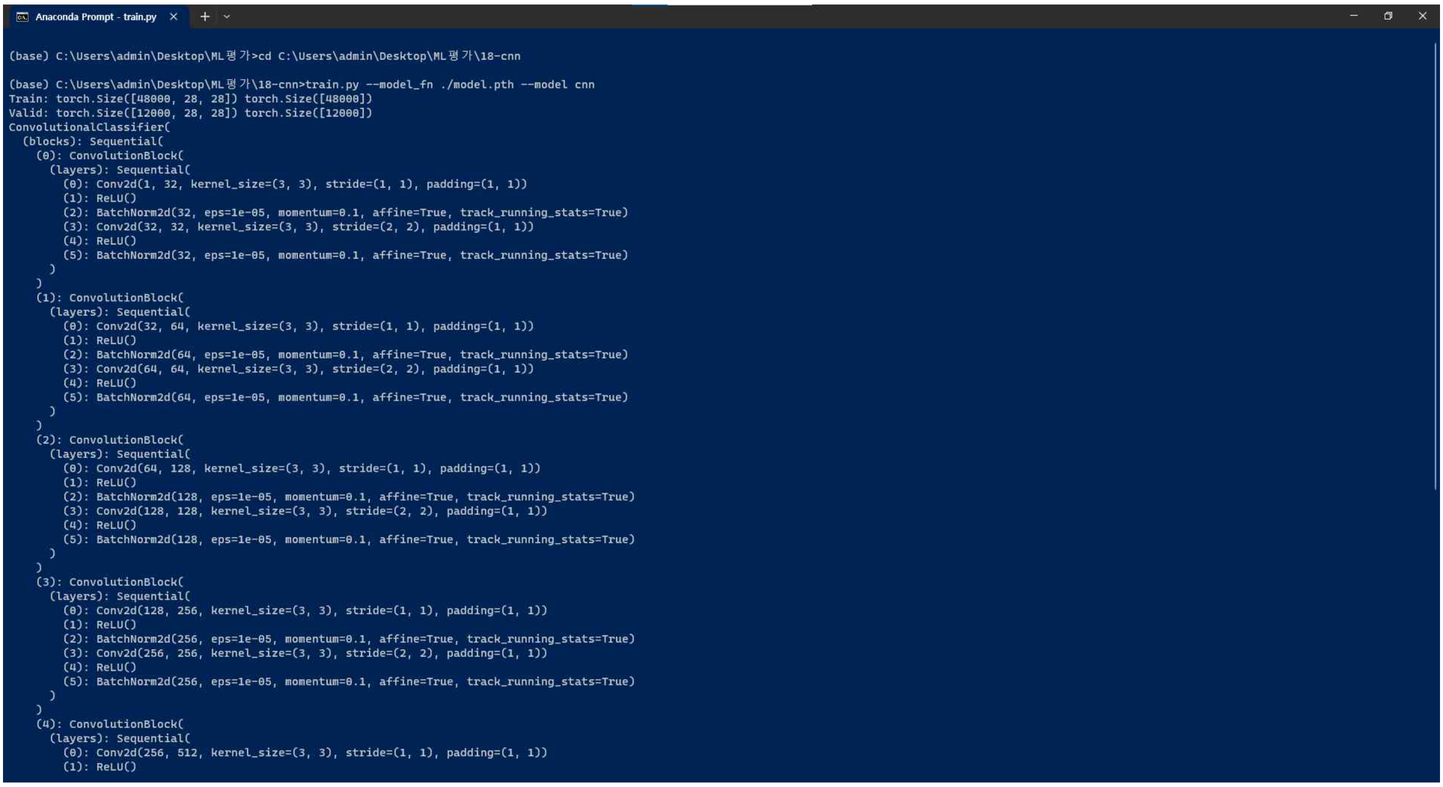
<작업폴더 캡처화면>



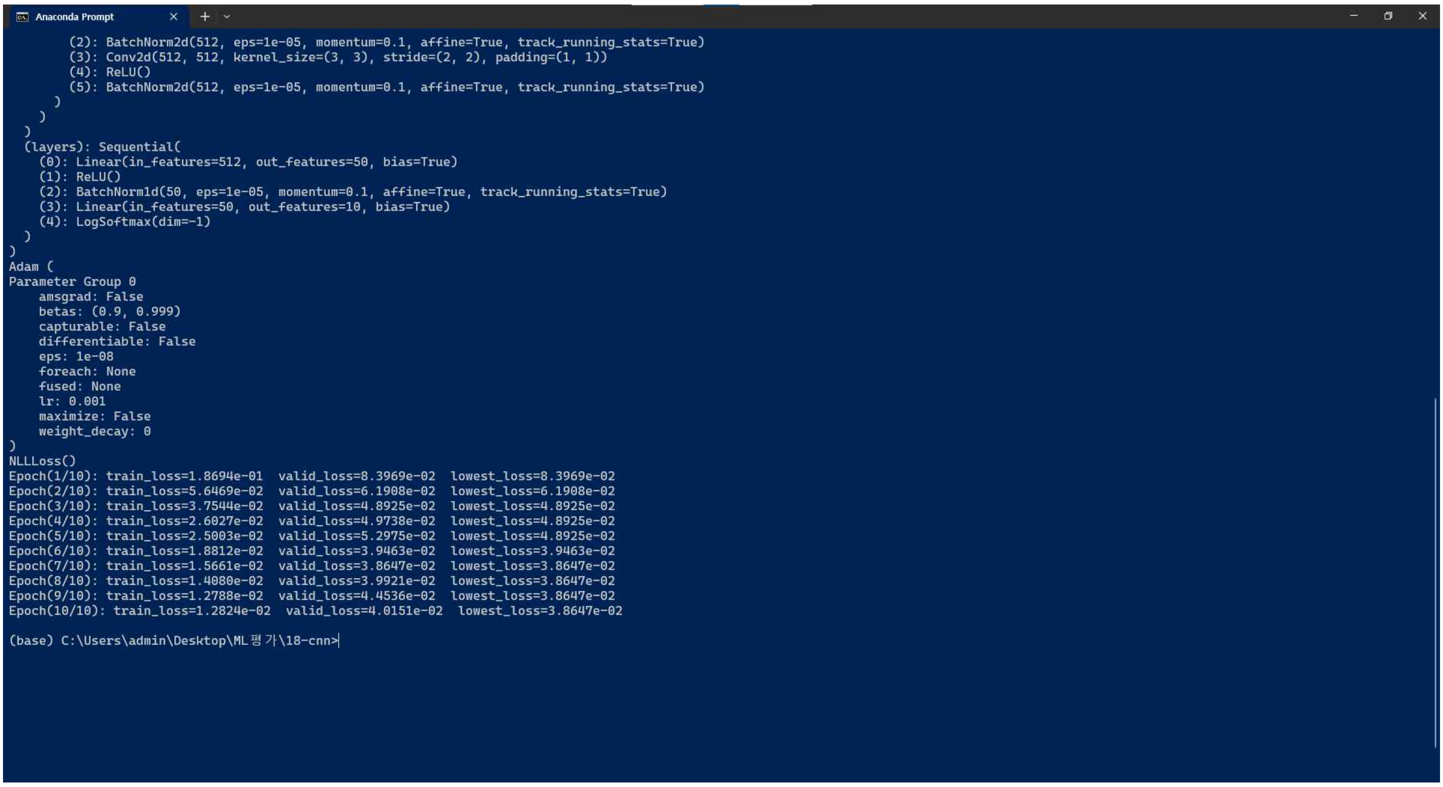
이름	수정한 날짜	유형	크기
.ipynb_checkpoints	2023-10-11 오전 9:12	파일 폴더	
__pycache__	2023-10-11 오전 9:12	파일 폴더	
model.pth	2022-06-19 오후 9:34	PTH 파일	6,562KB
model.py	2022-06-19 오후 9:34	Python File	2KB
predict.ipynb	2023-10-11 오전 9:57	Jupyter 원본 파일	202KB
tmp.pth	2023-10-11 오전 9:34	PTH 파일	11,843KB
train.py	2022-06-19 오후 9:34	Python File	3KB
trainer.py	2022-06-19 오후 9:34	Python File	3KB
utils.py	2022-06-19 오후 9:34	Python File	2KB

18장

<모델 학습 - model.pth>

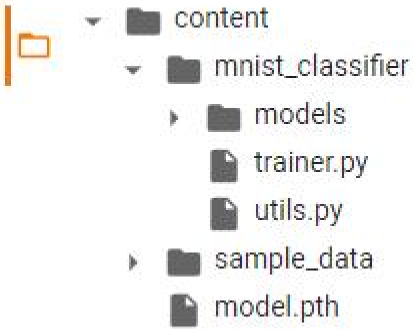


```
(base) C:\Users\admin\Desktop\ML평가>cd C:\Users\admin\Desktop\ML평가\18-cnn

(base) C:\Users\admin\Desktop\ML평가\18-cnn>train.py --model_fn ./model.pth --model cnn
Train: torch.Size([48000, 28, 28]) torch.Size([48000])
Valid: torch.Size([12000, 28, 28]) torch.Size([12000])
ConvolutionalClassifier(
  (blocks): Sequential(
    (0): ConvolutionBlock(
      (layers): Sequential(
        (0): Conv2d(1, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (1): ReLU()
        (2): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (3): Conv2d(32, 32, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
        (4): ReLU()
        (5): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
    (1): ConvolutionBlock(
      (layers): Sequential(
        (0): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (1): ReLU()
        (2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (3): Conv2d(64, 64, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
        (4): ReLU()
        (5): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
    (2): ConvolutionBlock(
      (layers): Sequential(
        (0): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (1): ReLU()
        (2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (3): Conv2d(128, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
        (4): ReLU()
        (5): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
    (3): ConvolutionBlock(
      (layers): Sequential(
        (0): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (1): ReLU()
        (2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (3): Conv2d(256, 256, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
        (4): ReLU()
        (5): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
    (4): ConvolutionBlock(
      (layers): Sequential(
        (0): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (1): ReLU()
```

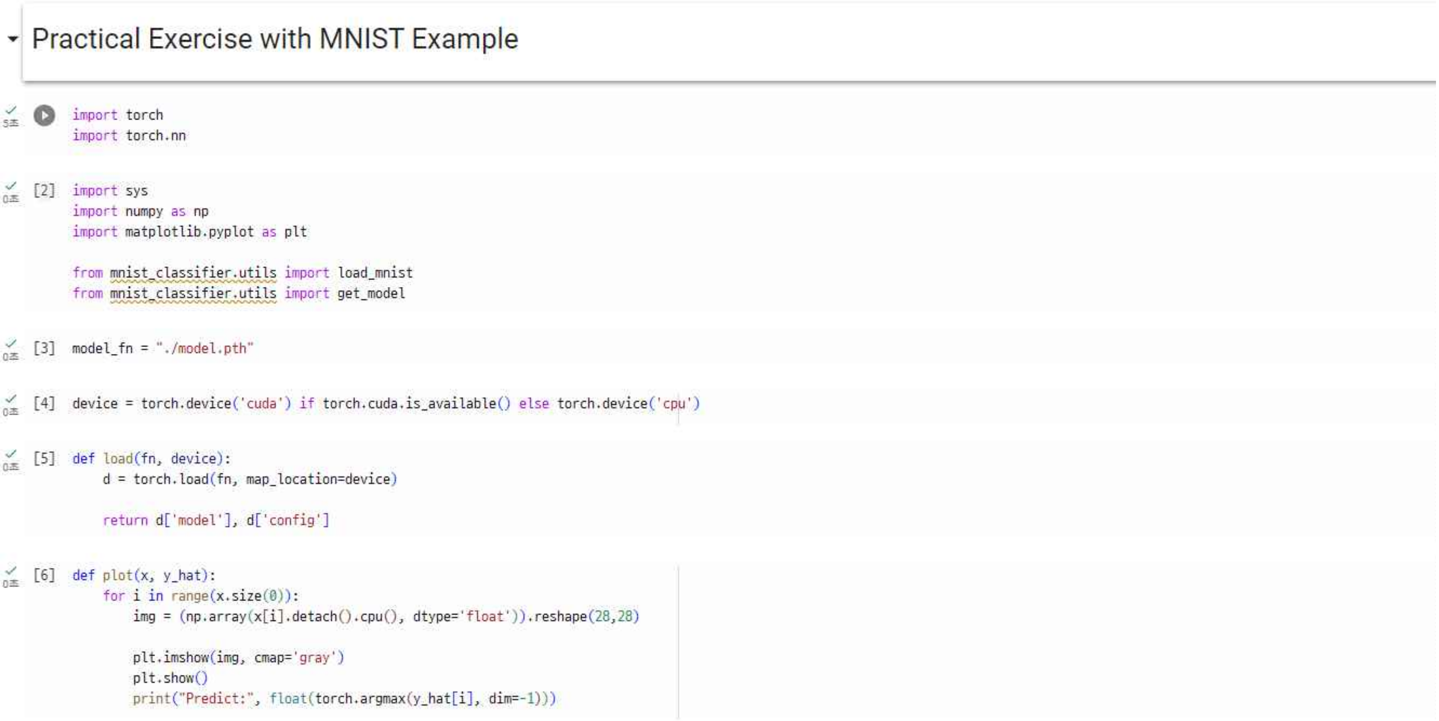


```
        (2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (3): Conv2d(512, 512, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
        (4): ReLU()
        (5): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
  )
  (layers): Sequential(
    (0): Linear(in_features=512, out_features=50, bias=True)
    (1): ReLU()
    (2): BatchNorm1d(50, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (3): Linear(in_features=50, out_features=10, bias=True)
    (4): LogSoftmax(dim=-1)
  )
)
Adam (
Parameter Group 0
    amsgrad: False
    betas: (0.9, 0.999)
    capturable: False
    differentiable: False
    eps: 1e-08
    foreach: None
    fused: None
    lr: 0.001
    maximize: False
    weight_decay: 0
)
NLLLoss()
Epoch(1/10): train_loss=1.8694e-01  valid_loss=8.3969e-02  lowest_loss=8.3969e-02
Epoch(2/10): train_loss=5.6469e-02  valid_loss=6.1908e-02  lowest_loss=6.1908e-02
Epoch(3/10): train_loss=3.7544e-02  valid_loss=4.8925e-02  lowest_loss=4.8925e-02
Epoch(4/10): train_loss=2.6027e-02  valid_loss=4.9738e-02  lowest_loss=4.8925e-02
Epoch(5/10): train_loss=2.5003e-02  valid_loss=5.2975e-02  lowest_loss=4.8925e-02
Epoch(6/10): train_loss=1.8812e-02  valid_loss=3.9463e-02  lowest_loss=3.9463e-02
Epoch(7/10): train_loss=1.5661e-02  valid_loss=3.8647e-02  lowest_loss=3.8647e-02
Epoch(8/10): train_loss=1.4080e-02  valid_loss=3.9921e-02  lowest_loss=3.8647e-02
Epoch(9/10): train_loss=1.2788e-02  valid_loss=4.4536e-02  lowest_loss=3.8647e-02
Epoch(10/10): train_loss=1.2824e-02  valid_loss=4.0151e-02  lowest_loss=3.8647e-02

(base) C:\Users\admin\Desktop\ML평가\18-cnn>
```

<content 파일 삽입 및 폴더 생성 - mnist_classifier 폴더, models 폴더>



<predict.ipynb 구동>



Practical Exercise with MNIST Example

```python
import torch
import torch.nn
```

[2]

```python
import sys
import numpy as np
import matplotlib.pyplot as plt

from mnist_classifier.utils import load_mnist
from mnist_classifier.utils import get_model
```

[3]

```python
model_fn = "./model.pth"
```

[4]

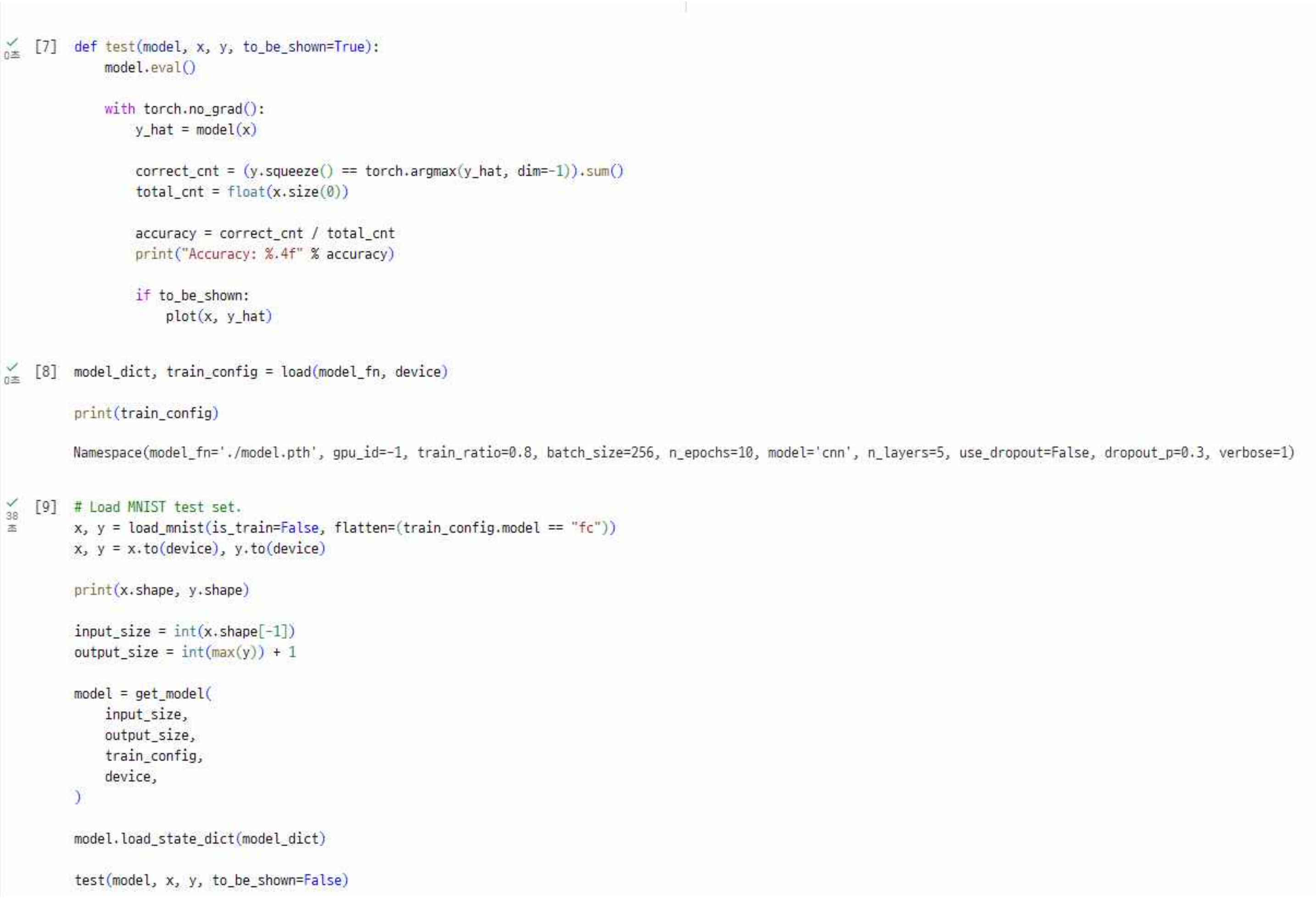
```python
device = torch.device('cuda') if torch.cuda.is_available() else torch.device('cpu')
```

[5]

```python
def load(fn, device):
    d = torch.load(fn, map_location=device)

    return d['model'], d['config']
```

[6]

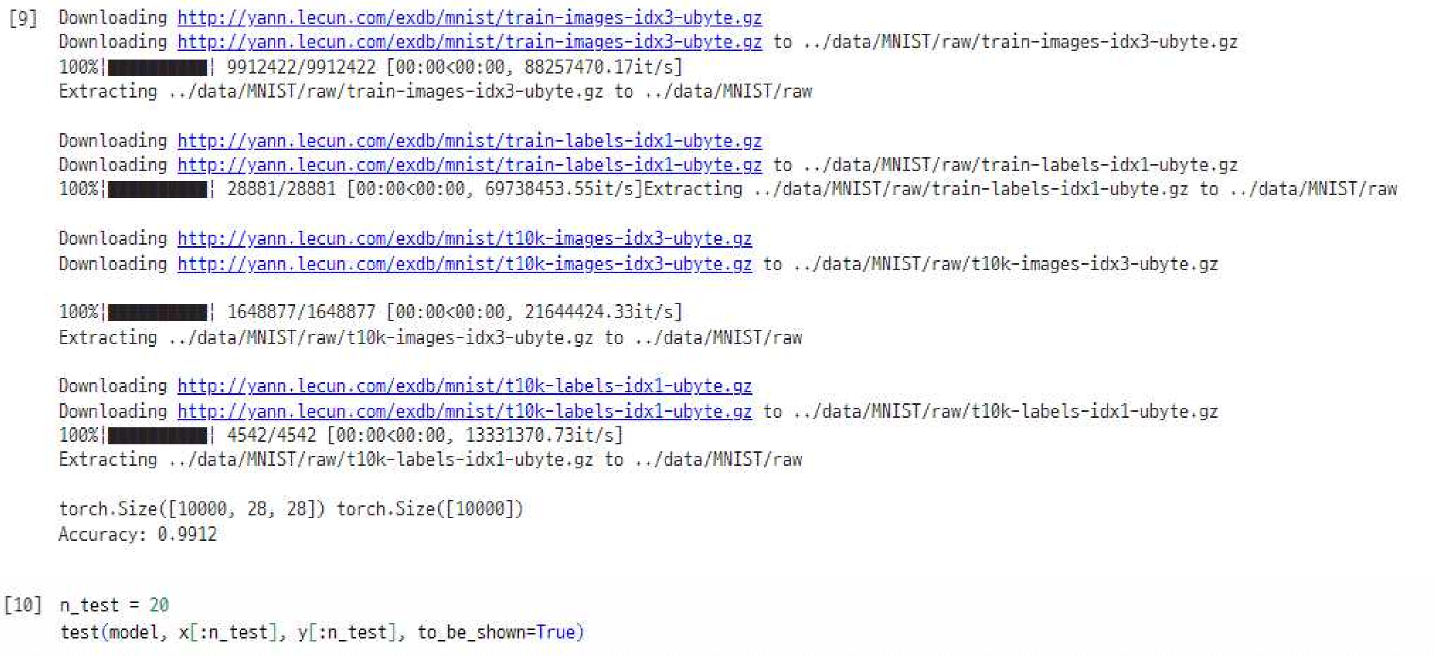
```python
def plot(x, y_hat):
    for i in range(x.size(0)):
        img = (np.array(x[i].detach().cpu(), dtype='float')).reshape(28,28)

        plt.imshow(img, cmap='gray')
        plt.show()
        print("Predict:", float(torch.argmax(y_hat[i], dim=-1)))
```



```
[7] def test(model, x, y, to_be_shown=True):
        model.eval()

        with torch.no_grad():
            y_hat = model(x)

            correct_cnt = (y.squeeze() == torch.argmax(y_hat, dim=-1)).sum()
            total_cnt = float(x.size(0))

            accuracy = correct_cnt / total_cnt
            print("Accuracy: %.4f" % accuracy)

            if to_be_shown:
                plot(x, y_hat)
```

```
[8] model_dict, train_config = load(model_fn, device)

    print(train_config)

    Namespace(model_fn='./model.pth', gpu_id=-1, train_ratio=0.8, batch_size=256, n_epochs=10, model='cnn', n_layers=5, use_dropout=False, dropout_p=0.3, verbose=1)
```

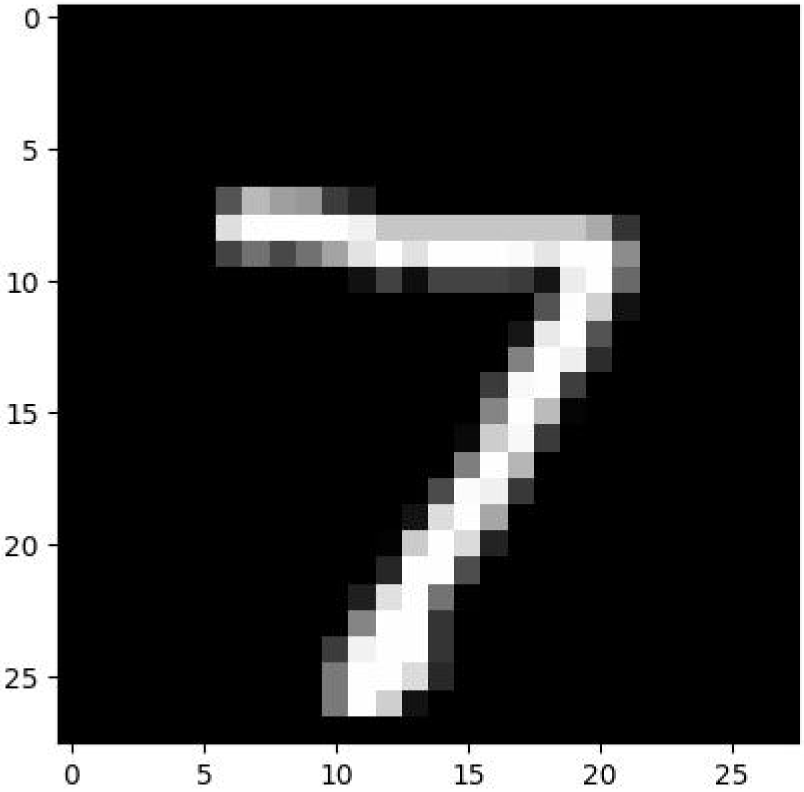
```
[9] # Load MNIST test set.
    x, y = load_mnist(is_train=False, flatten=(train_config.model == "fc"))
    x, y = x.to(device), y.to(device)

    print(x.shape, y.shape)

    input_size = int(x.shape[-1])
    output_size = int(max(y)) + 1

    model = get_model(
        input_size,
        output_size,
        train_config,
        device,
    )

    model.load_state_dict(model_dict)

    test(model, x, y, to_be_shown=False)
```



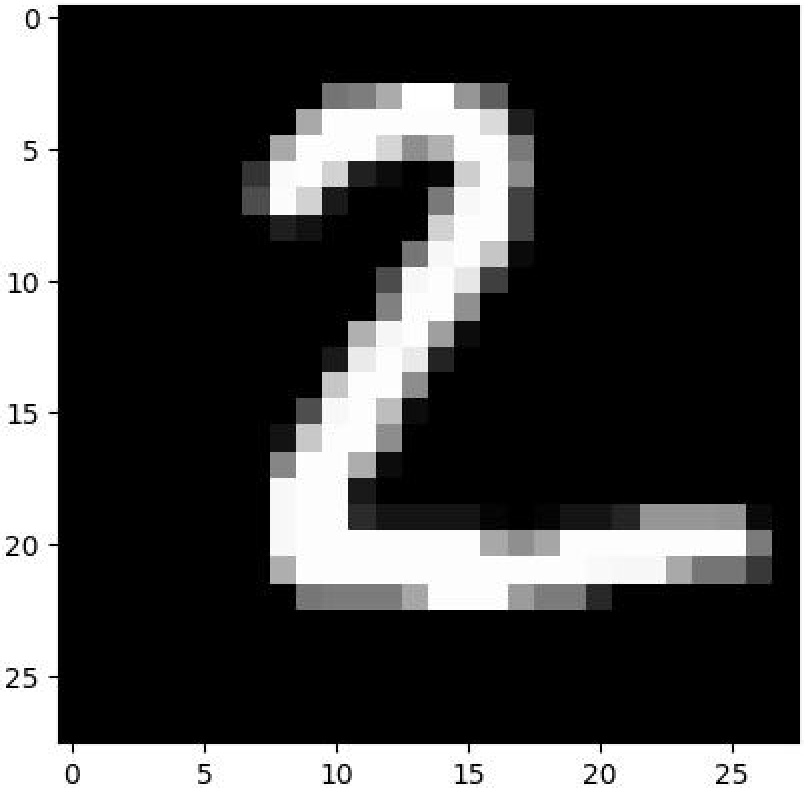
```
[9] Downloading http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz
    Downloading http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz to ../data/MNIST/raw/train-images-idx3-ubyte.gz
    100%|██████████| 9912422/9912422 [00:00<00:00, 88257470.17it/s]
    Extracting ../data/MNIST/raw/train-images-idx3-ubyte.gz to ../data/MNIST/raw

    Downloading http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz
    Downloading http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz to ../data/MNIST/raw/train-labels-idx1-ubyte.gz
    100%|██████████| 28881/28881 [00:00<00:00, 69738453.55it/s]Extracting ../data/MNIST/raw/train-labels-idx1-ubyte.gz to ../data/MNIST/raw

    Downloading http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz
    Downloading http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz to ../data/MNIST/raw/t10k-images-idx3-ubyte.gz

    100%|██████████| 1648877/1648877 [00:00<00:00, 21644424.33it/s]
    Extracting ../data/MNIST/raw/t10k-images-idx3-ubyte.gz to ../data/MNIST/raw

    Downloading http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz
    Downloading http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz to ../data/MNIST/raw/t10k-labels-idx1-ubyte.gz
    100%|██████████| 4542/4542 [00:00<00:00, 13331370.73it/s]
    Extracting ../data/MNIST/raw/t10k-labels-idx1-ubyte.gz to ../data/MNIST/raw

    torch.Size([10000, 28, 28]) torch.Size([10000])
    Accuracy: 0.9912
```

```
[10] n_test = 20
     test(model, x[:n_test], y[:n_test], to_be_shown=True)
```

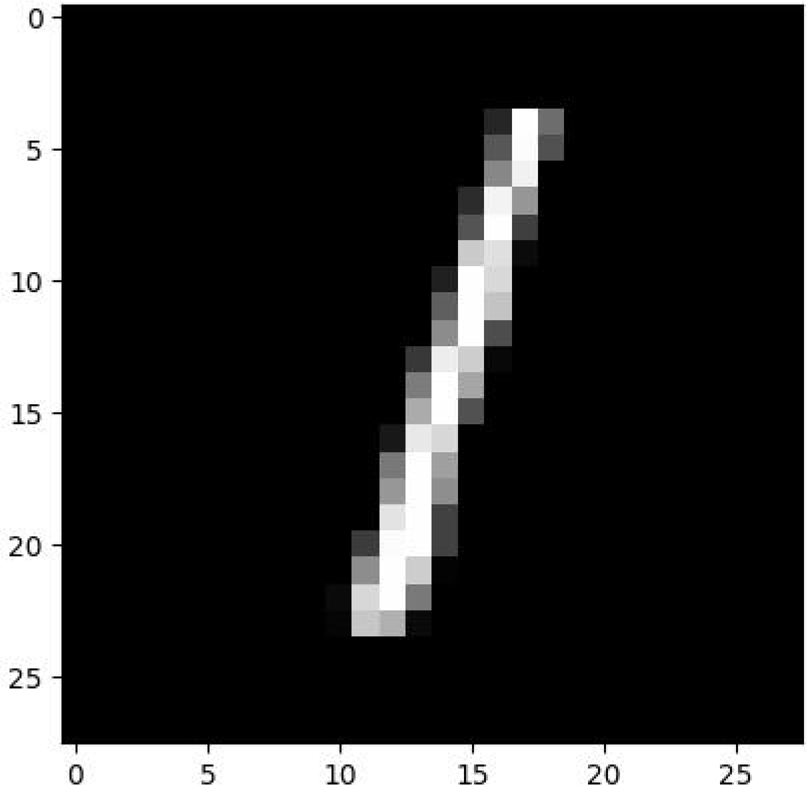
```
Accuracy: 1.0000
```



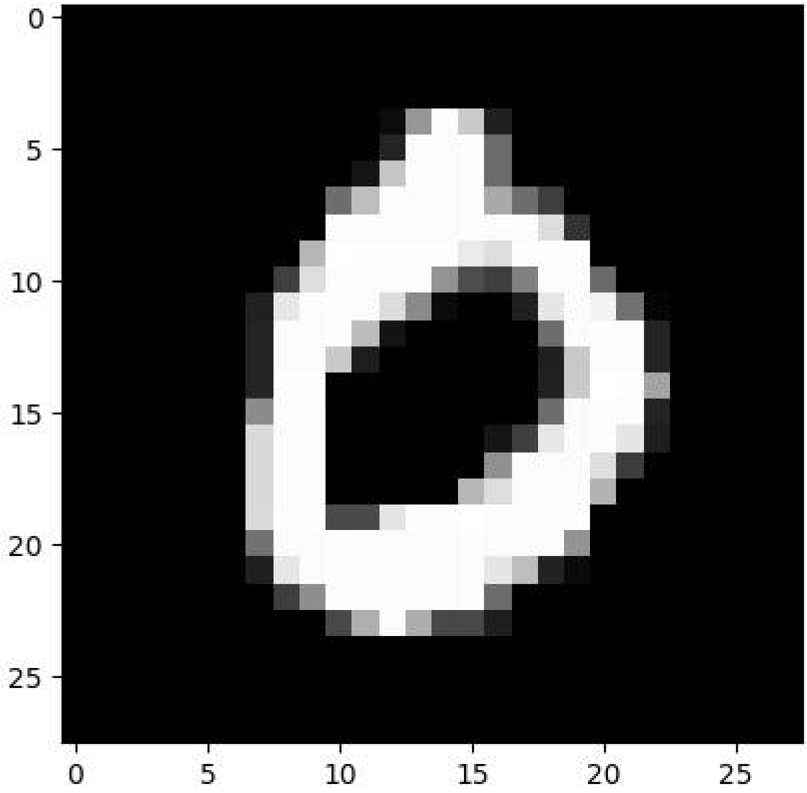
```
Predict: 7.0
```



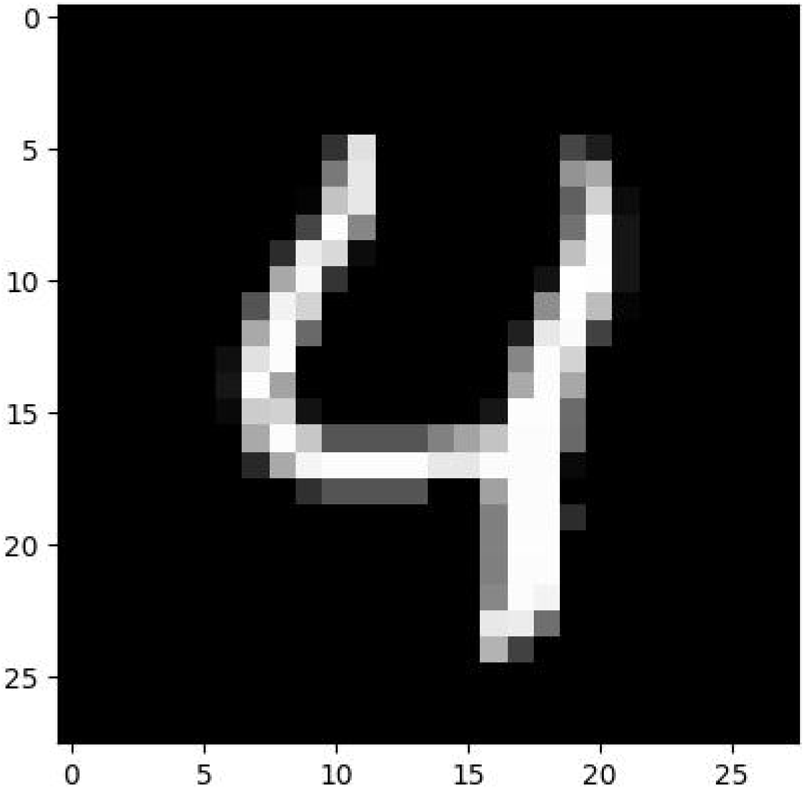
Predict: 2.0



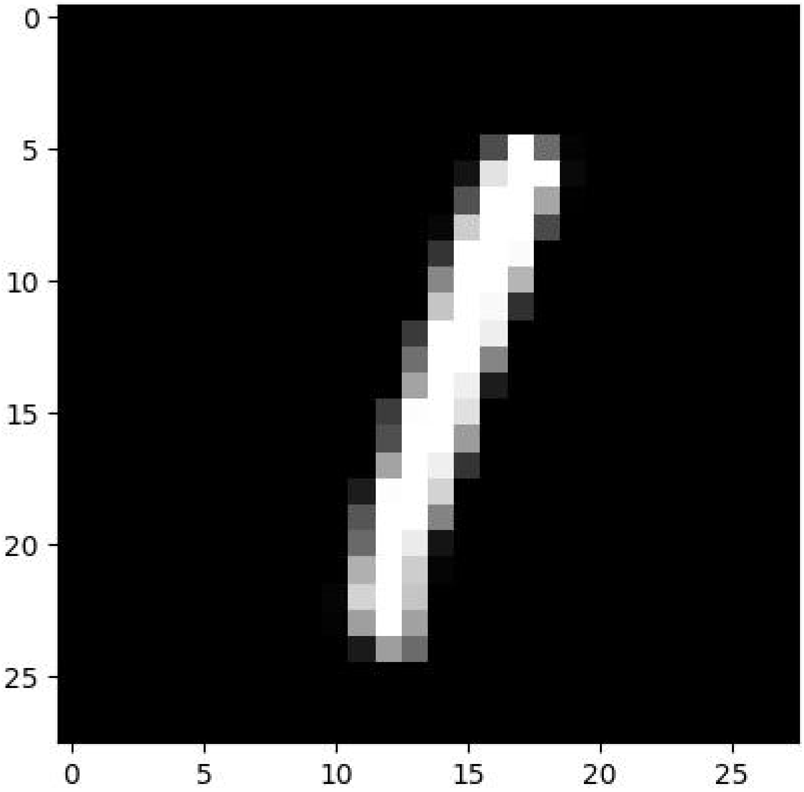
Predict: 1.0



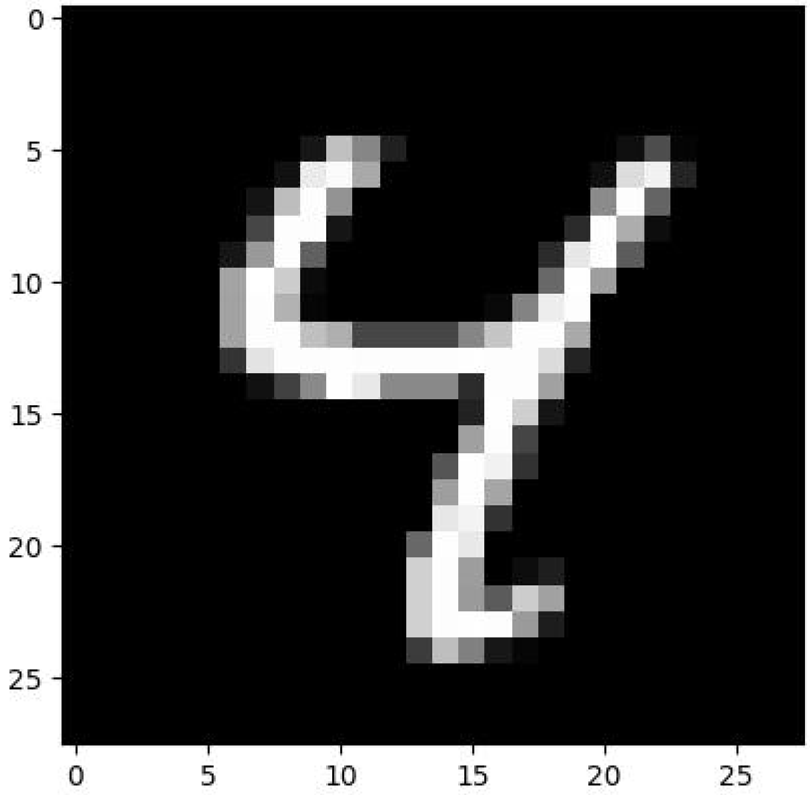
Predict: 0.0



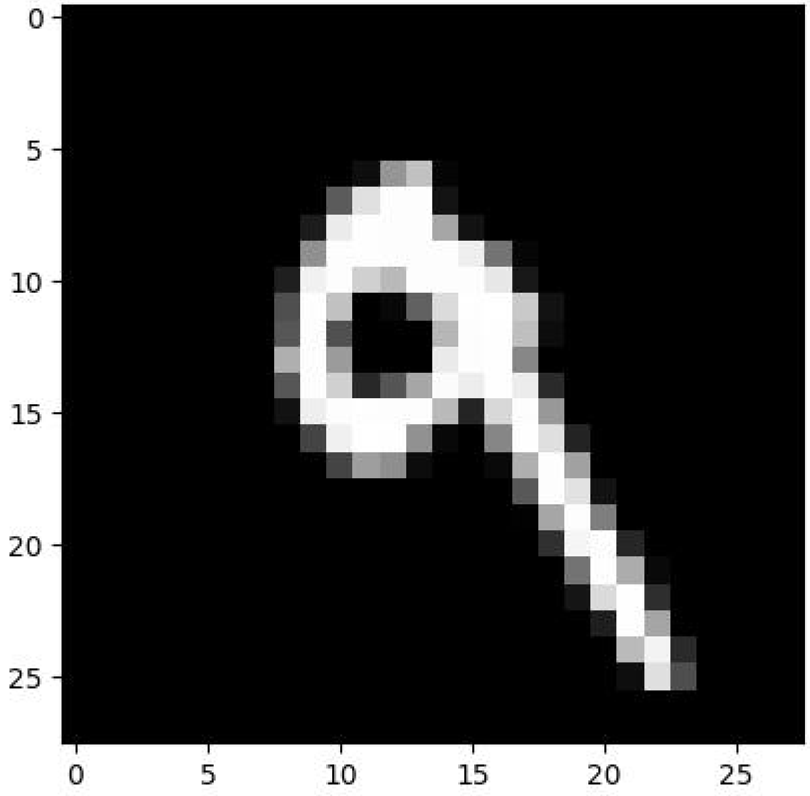
Predict: 4.0



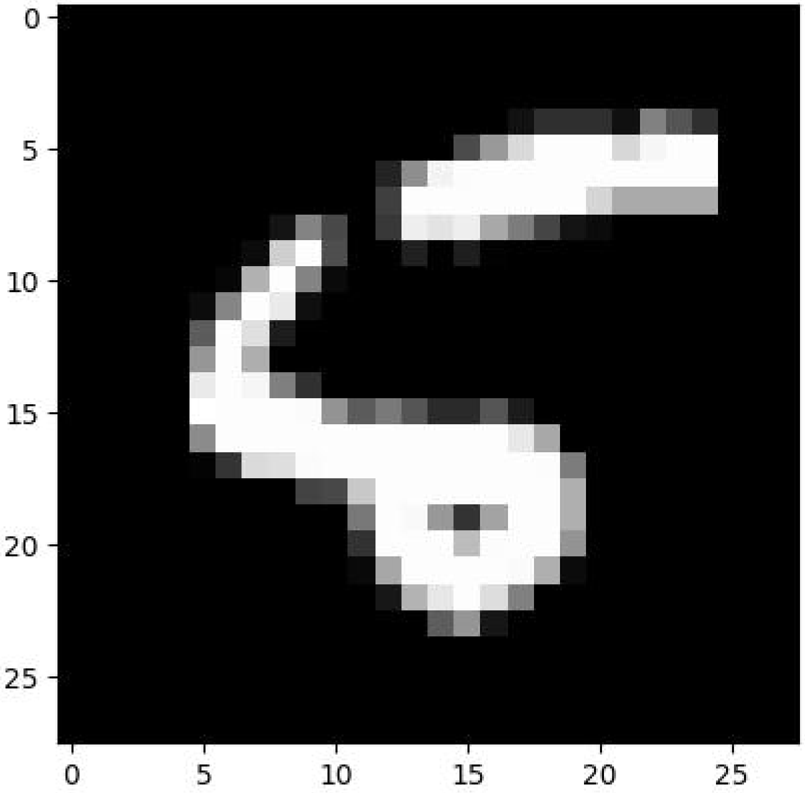
Predict: 1.0



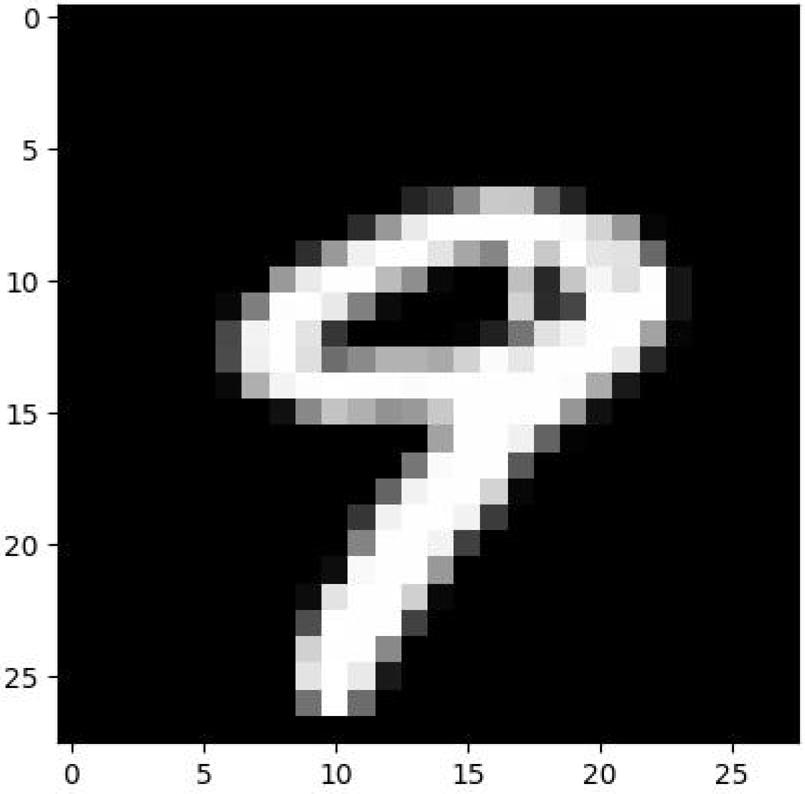
Predict: 4.0



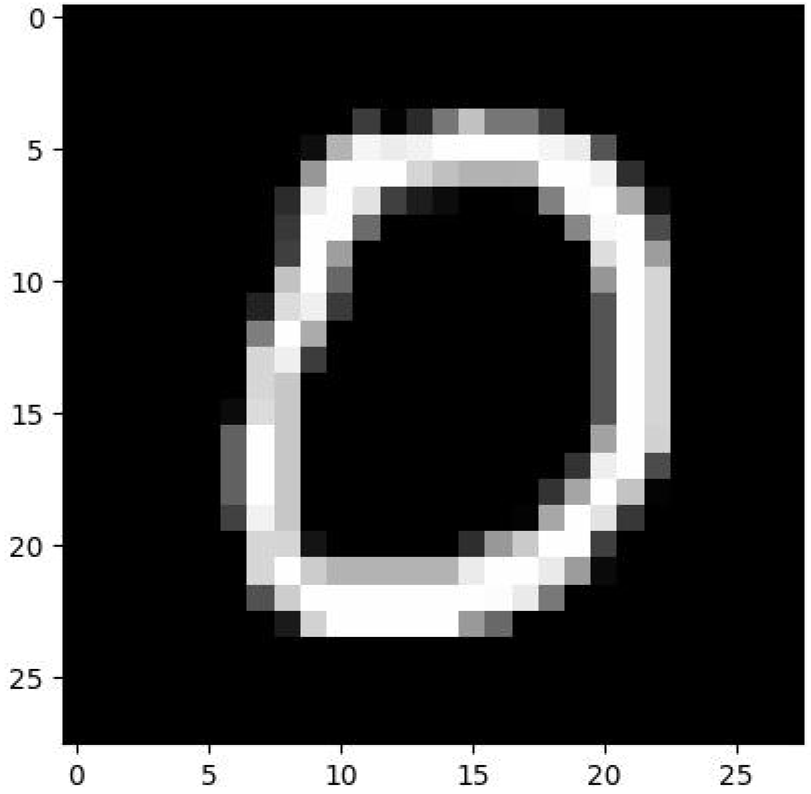
```
Predict: 9.0
```



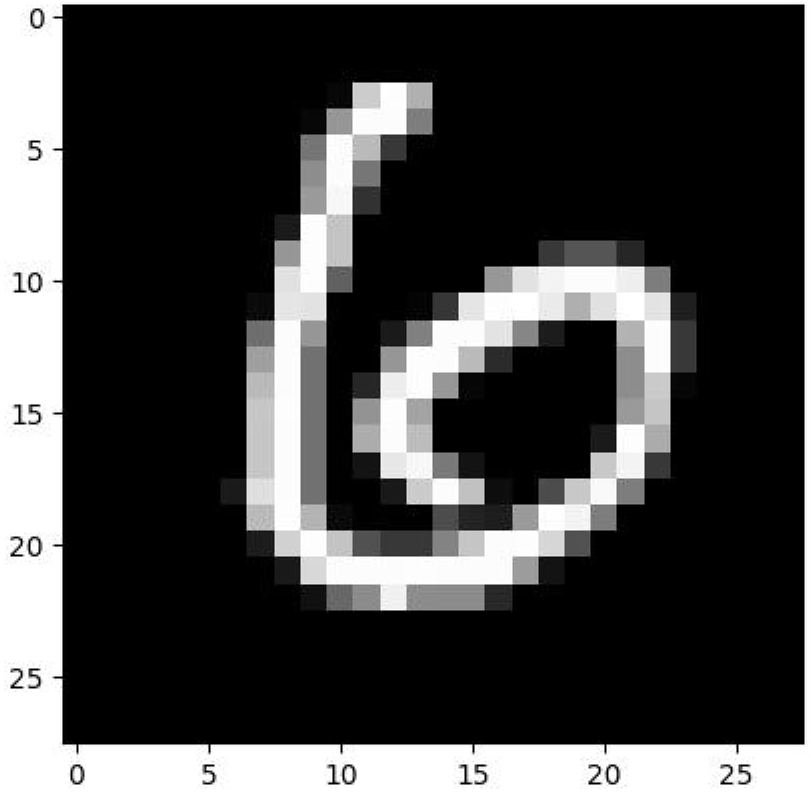
```
Predict: 5.0
```



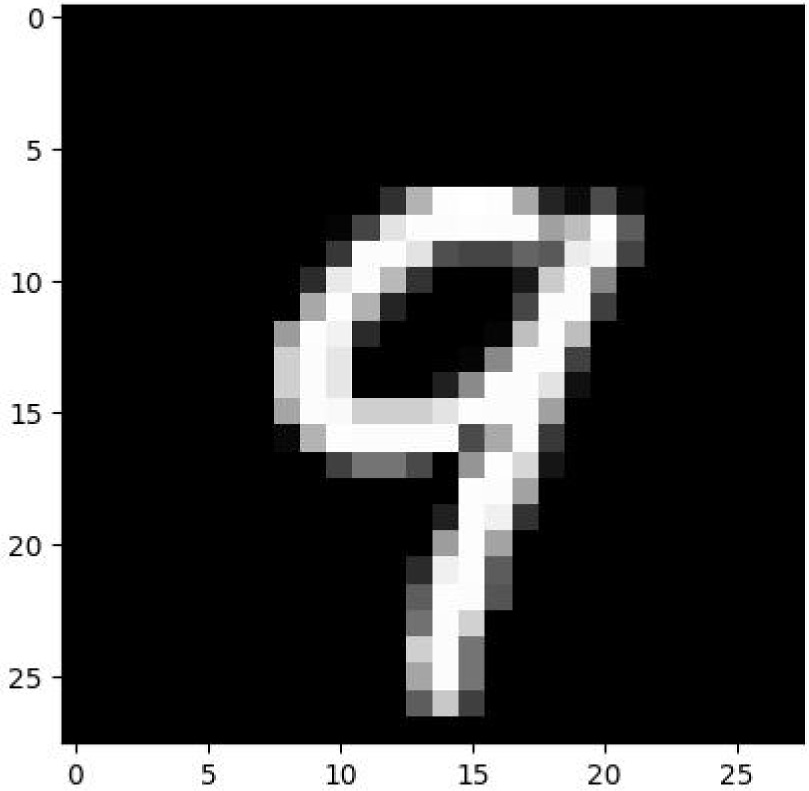
Predict: 9.0



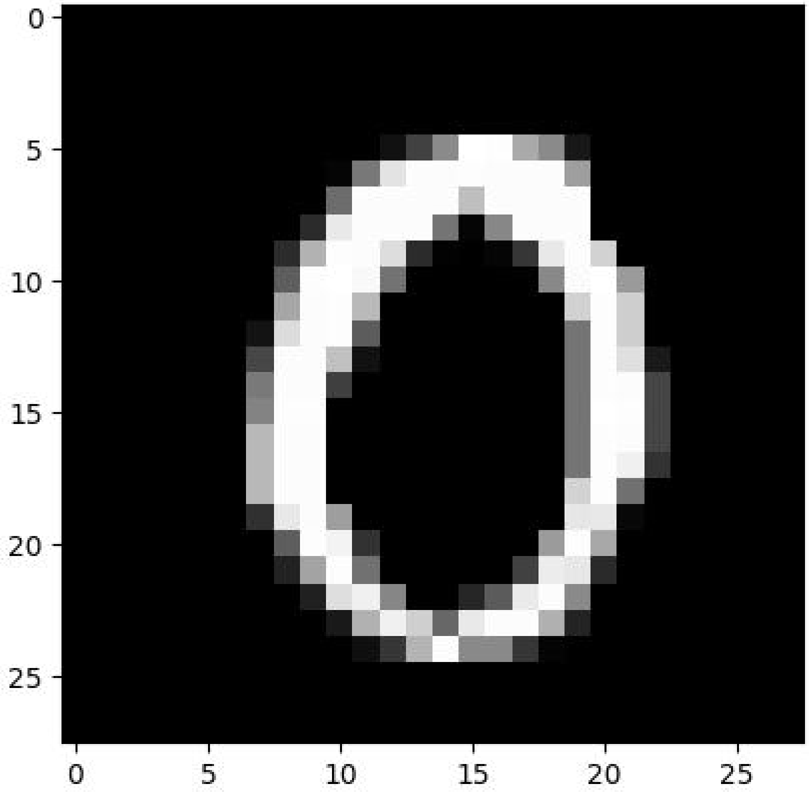
Predict: 0.0



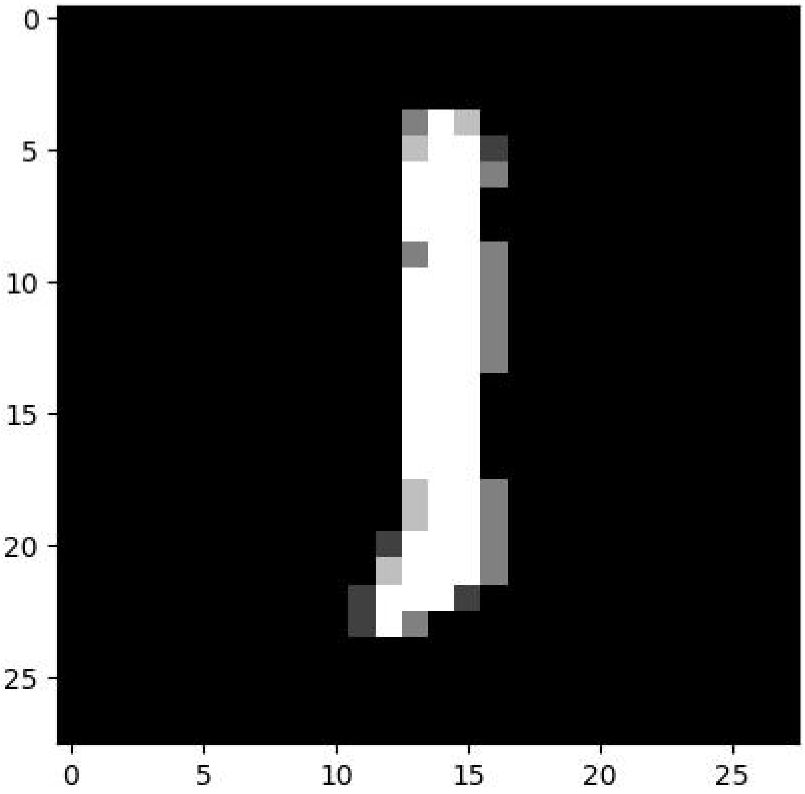
```
Predict: 6.0
```



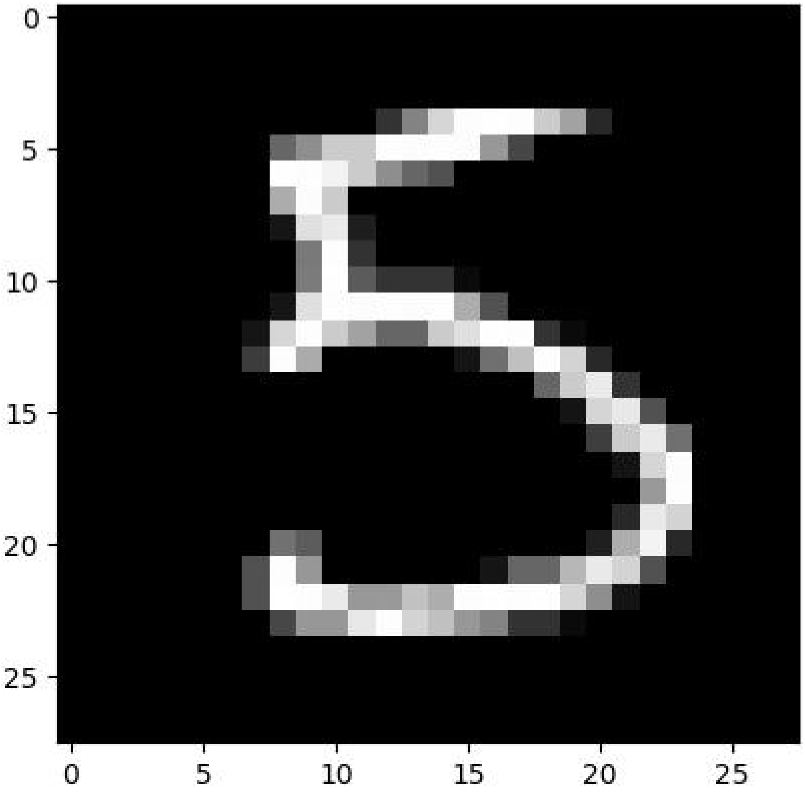
```
Predict: 9.0
```



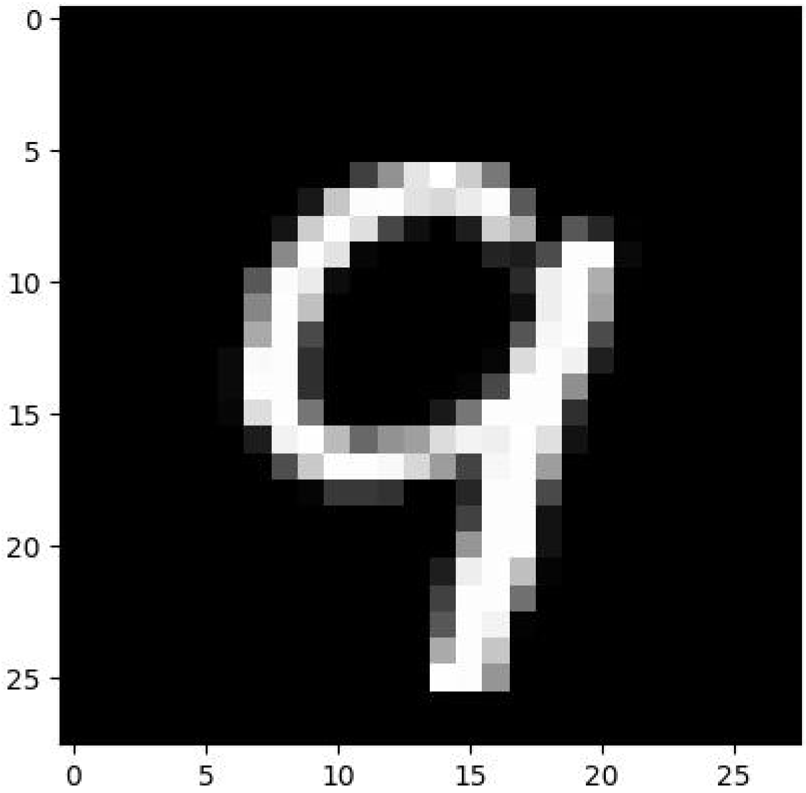
Predict: 0.0



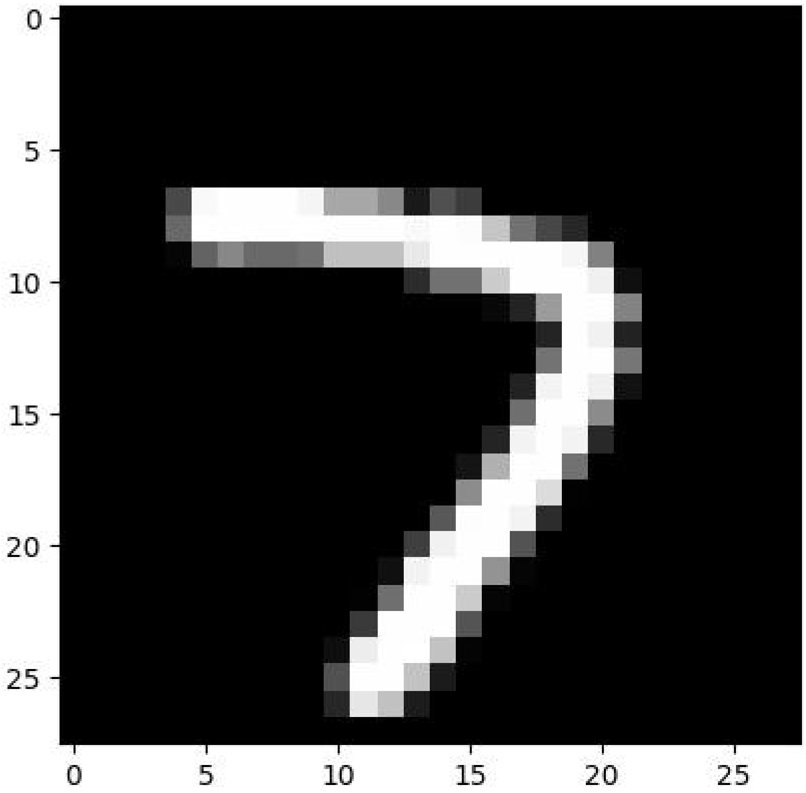
Predict: 1.0



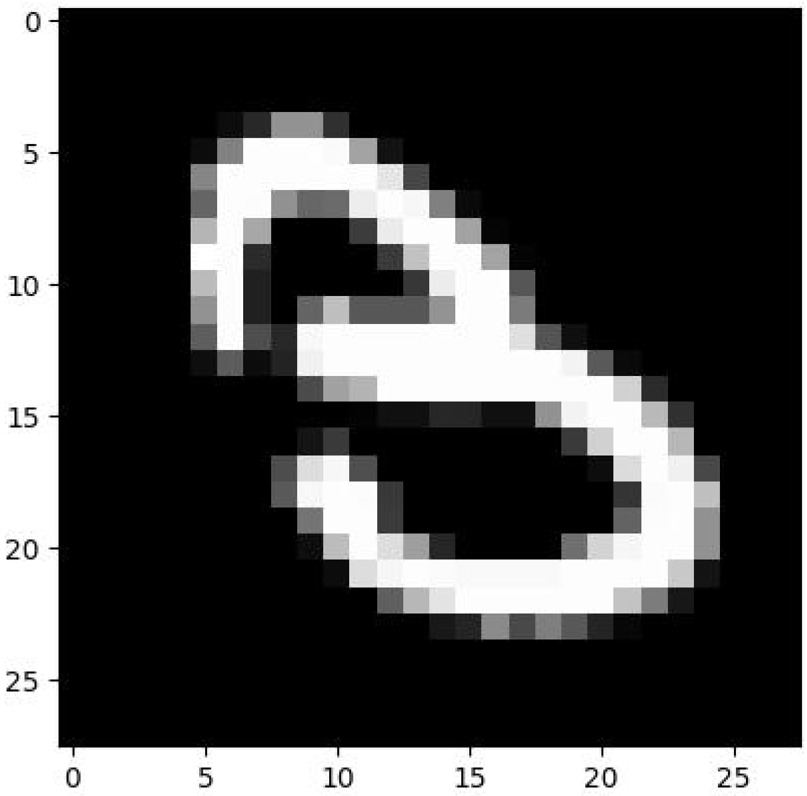
Predict: 5.0



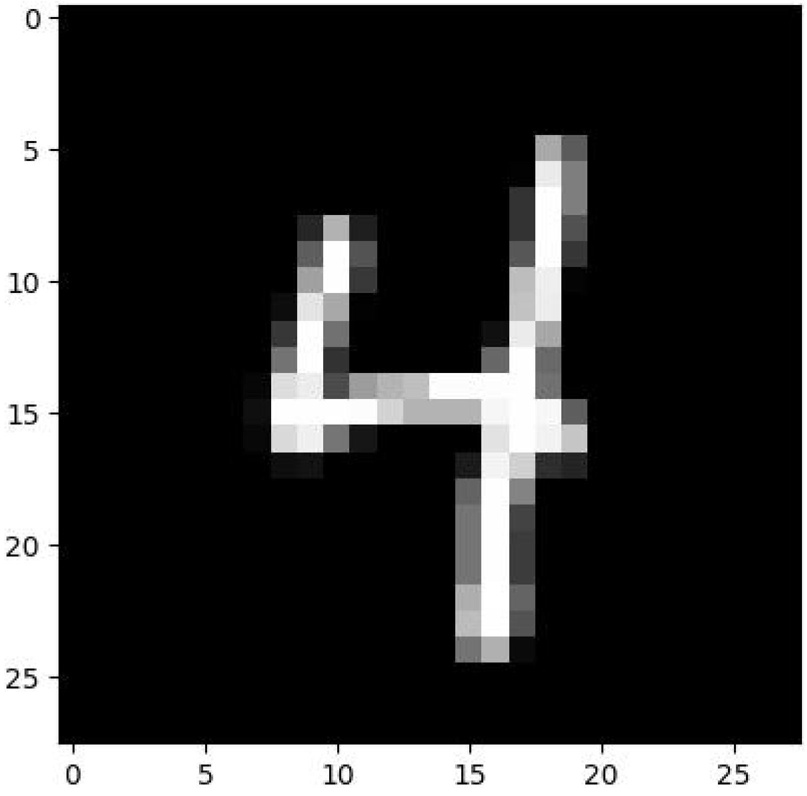
Predict: 9.0



```
Predict: 7.0
```

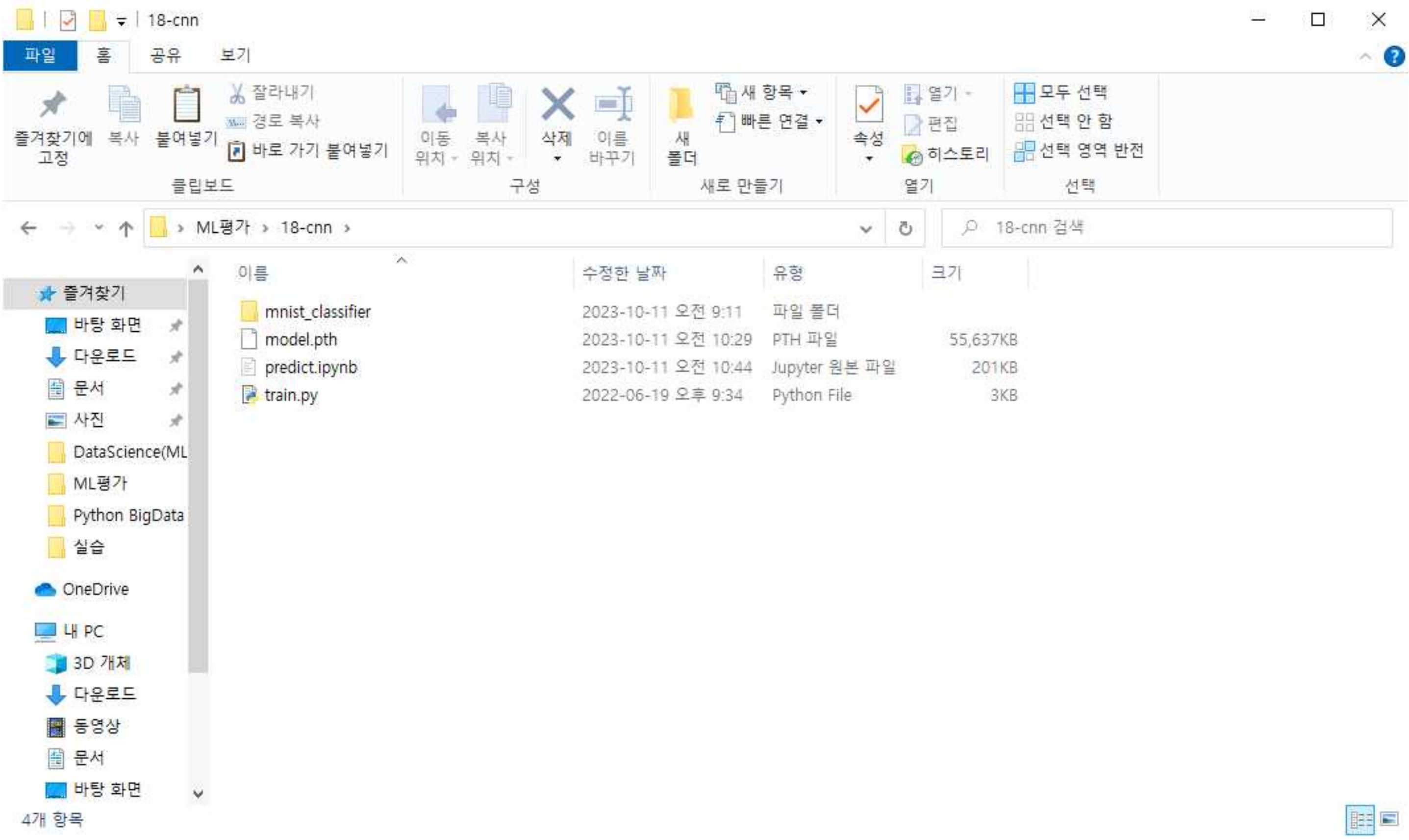


```
Predict: 3.0
```



```
Predict: 4.0
```

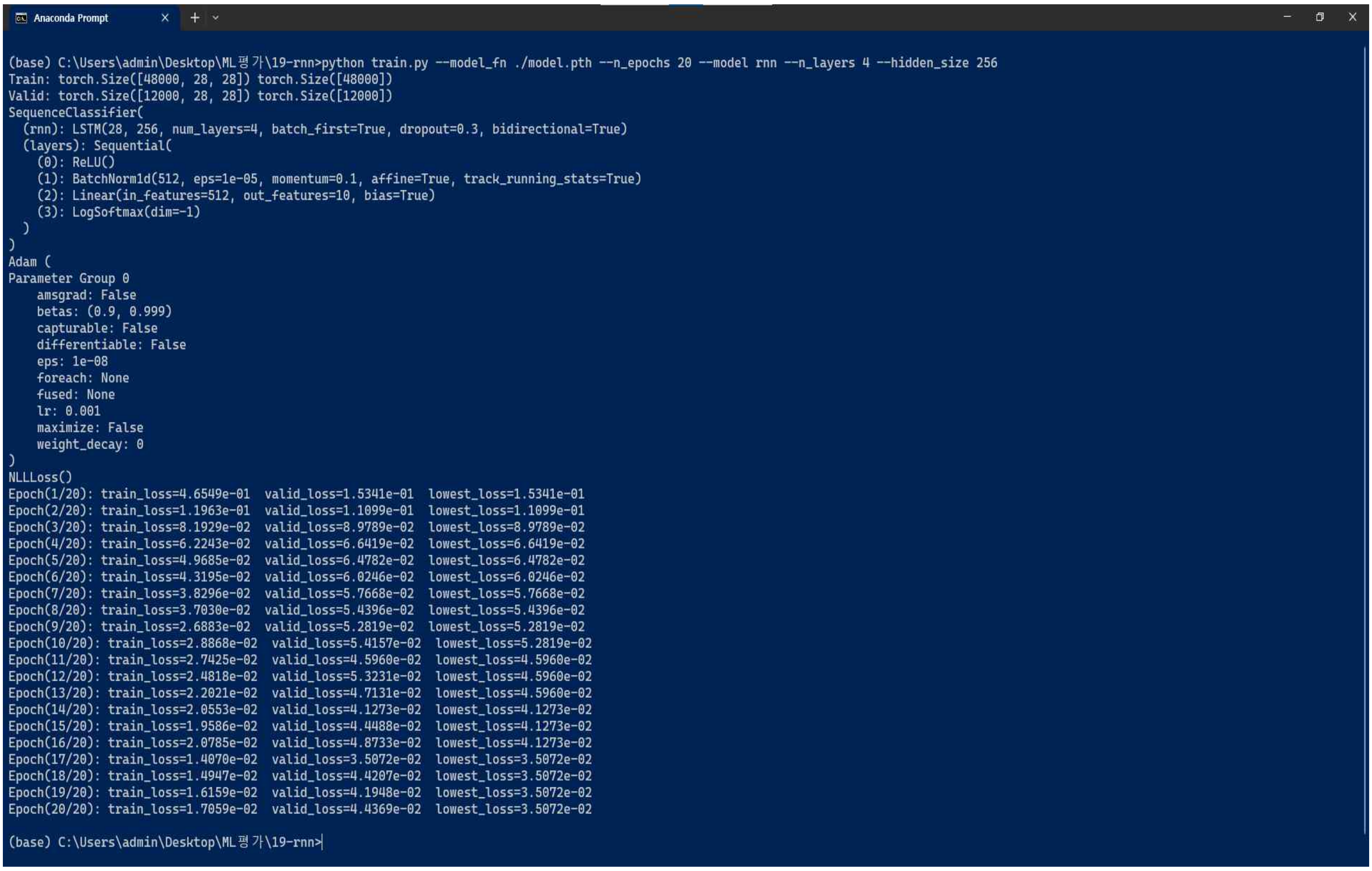
<작업폴더 화면 캡처>



이름	수정한 날짜	유형	크기
mnist_classifier	2023-10-11 오전 9:11	파일 폴더	
model.pth	2023-10-11 오전 10:29	PTH 파일	55,637KB
predict.ipynb	2023-10-11 오전 10:44	Jupyter 원본 파일	201KB
train.py	2022-06-19 오후 9:34	Python File	3KB

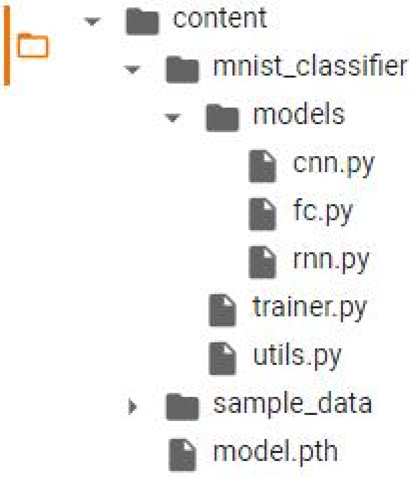
19장

<모델 학습 - model.pth>

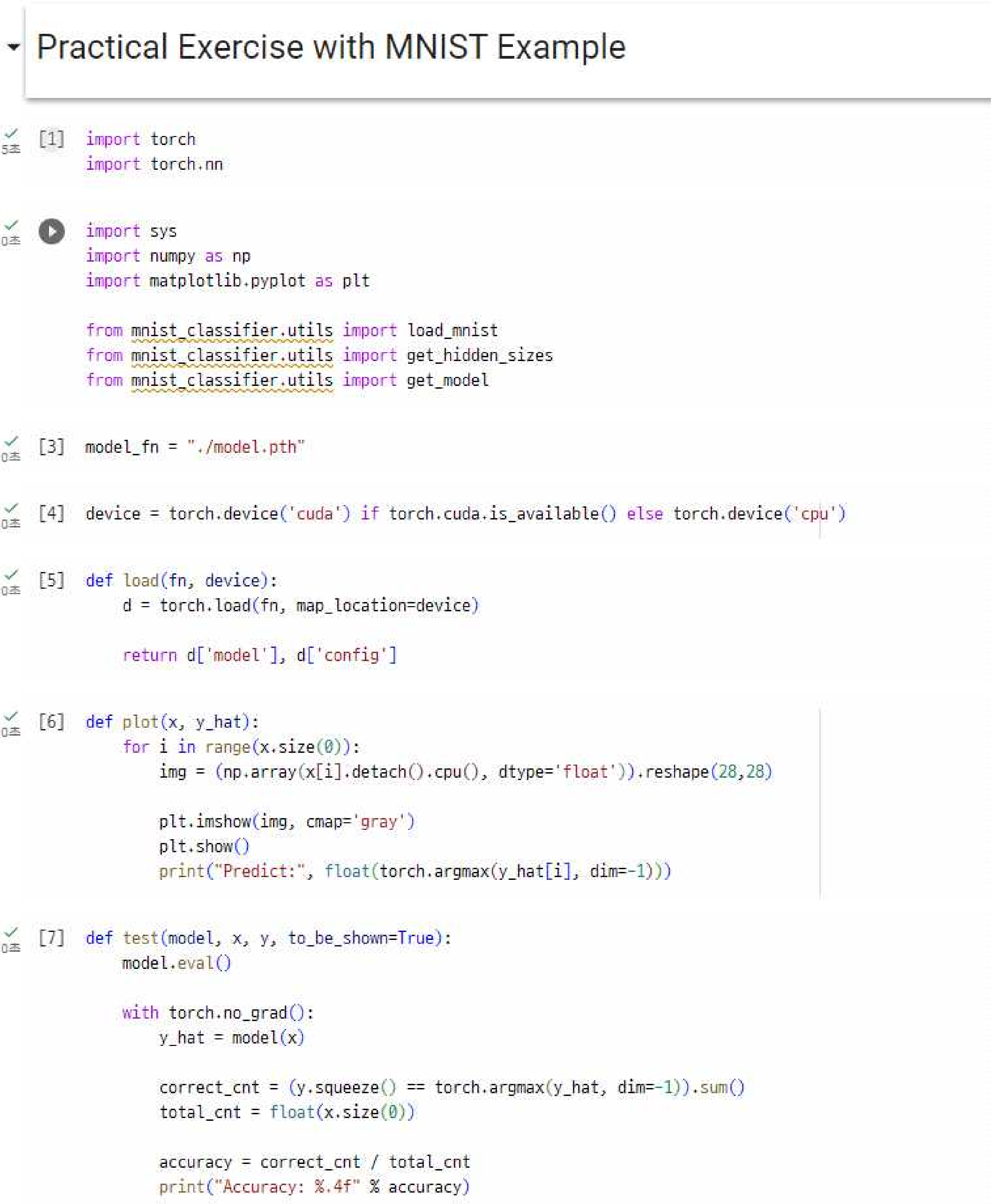


```
(base) C:\Users\admin\Desktop\ML평가\19-rnn>python train.py --model_fn ./model.pth --n_epochs 20 --model rnn --n_layers 4 --hidden_size 256
Train: torch.Size([48000, 28, 28]) torch.Size([48000])
Valid: torch.Size([12000, 28, 28]) torch.Size([12000])
SequenceClassifier(
  (rnn): LSTM(28, 256, num_layers=4, batch_first=True, dropout=0.3, bidirectional=True)
  (layers): Sequential(
    (0): ReLU()
    (1): BatchNorm1d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): Linear(in_features=512, out_features=10, bias=True)
    (3): LogSoftmax(dim=-1)
  )
)
Adam (
Parameter Group 0
    amsgrad: False
    betas: (0.9, 0.999)
    capturable: False
    differentiable: False
    eps: 1e-08
    foreach: None
    fused: None
    lr: 0.001
    maximize: False
    weight_decay: 0
)
NLLLoss()
Epoch(1/20): train_loss=4.6549e-01  valid_loss=1.5341e-01  lowest_loss=1.5341e-01
Epoch(2/20): train_loss=1.1963e-01  valid_loss=1.1099e-01  lowest_loss=1.1099e-01
Epoch(3/20): train_loss=8.1929e-02  valid_loss=8.9789e-02  lowest_loss=8.9789e-02
Epoch(4/20): train_loss=6.2243e-02  valid_loss=6.6419e-02  lowest_loss=6.6419e-02
Epoch(5/20): train_loss=4.9685e-02  valid_loss=6.4782e-02  lowest_loss=6.4782e-02
Epoch(6/20): train_loss=4.3195e-02  valid_loss=6.0246e-02  lowest_loss=6.0246e-02
Epoch(7/20): train_loss=3.8296e-02  valid_loss=5.7668e-02  lowest_loss=5.7668e-02
Epoch(8/20): train_loss=3.7030e-02  valid_loss=5.4396e-02  lowest_loss=5.4396e-02
Epoch(9/20): train_loss=2.6883e-02  valid_loss=5.2819e-02  lowest_loss=5.2819e-02
Epoch(10/20): train_loss=2.8868e-02  valid_loss=5.4157e-02  lowest_loss=5.2819e-02
Epoch(11/20): train_loss=2.7425e-02  valid_loss=4.5960e-02  lowest_loss=4.5960e-02
Epoch(12/20): train_loss=2.4818e-02  valid_loss=5.3231e-02  lowest_loss=4.5960e-02
Epoch(13/20): train_loss=2.2021e-02  valid_loss=4.7131e-02  lowest_loss=4.5960e-02
Epoch(14/20): train_loss=2.0553e-02  valid_loss=4.1273e-02  lowest_loss=4.1273e-02
Epoch(15/20): train_loss=1.9586e-02  valid_loss=4.4488e-02  lowest_loss=4.1273e-02
Epoch(16/20): train_loss=2.0785e-02  valid_loss=4.8733e-02  lowest_loss=4.1273e-02
Epoch(17/20): train_loss=1.4070e-02  valid_loss=3.5072e-02  lowest_loss=3.5072e-02
Epoch(18/20): train_loss=1.4947e-02  valid_loss=4.4207e-02  lowest_loss=3.5072e-02
Epoch(19/20): train_loss=1.6159e-02  valid_loss=4.1948e-02  lowest_loss=3.5072e-02
Epoch(20/20): train_loss=1.7059e-02  valid_loss=4.4369e-02  lowest_loss=3.5072e-02

(base) C:\Users\admin\Desktop\ML평가\19-rnn>
```

<content 파일 삽입 및 폴더 생성 - mnist_classifier 폴더, models 폴더>



<predict.ipynb 구동>



Practical Exercise with MNIST Example

```
import torch
import torch.nn
```

```
import sys
import numpy as np
import matplotlib.pyplot as plt

from mnist_classifier.utils import load_mnist
from mnist_classifier.utils import get_hidden_sizes
from mnist_classifier.utils import get_model
```

```
model_fn = "./model.pth"
```

```
device = torch.device('cuda') if torch.cuda.is_available() else torch.device('cpu')
```

```
def load(fn, device):
    d = torch.load(fn, map_location=device)

    return d['model'], d['config']
```

```
def plot(x, y_hat):
    for i in range(x.size(0)):
        img = (np.array(x[i].detach().cpu(), dtype='float')).reshape(28,28)

        plt.imshow(img, cmap='gray')
        plt.show()
        print("Predict:", float(torch.argmax(y_hat[i], dim=-1)))
```

```
def test(model, x, y, to_be_shown=True):
    model.eval()

    with torch.no_grad():
        y_hat = model(x)

        correct_cnt = (y.squeeze() == torch.argmax(y_hat, dim=-1)).sum()
        total_cnt = float(x.size(0))

        accuracy = correct_cnt / total_cnt
        print("Accuracy: %.4f" % accuracy)
```

✓ 0초 [7]

```
        if to_be_shown:
            plot(x, y_hat)
```

✓ 0초 [8]

```
model_dict, train_config = load(model_fn, device)

print(train_config)
```

```
Namespace(model_fn='./model.pth', gpu_id=-1, train_ratio=0.8, batch_size=256, n_epochs=20, model='rnn', n_layers=4, use_dropout=False, dropout_p=0.3, hidden_size=256, verbose=1)
```

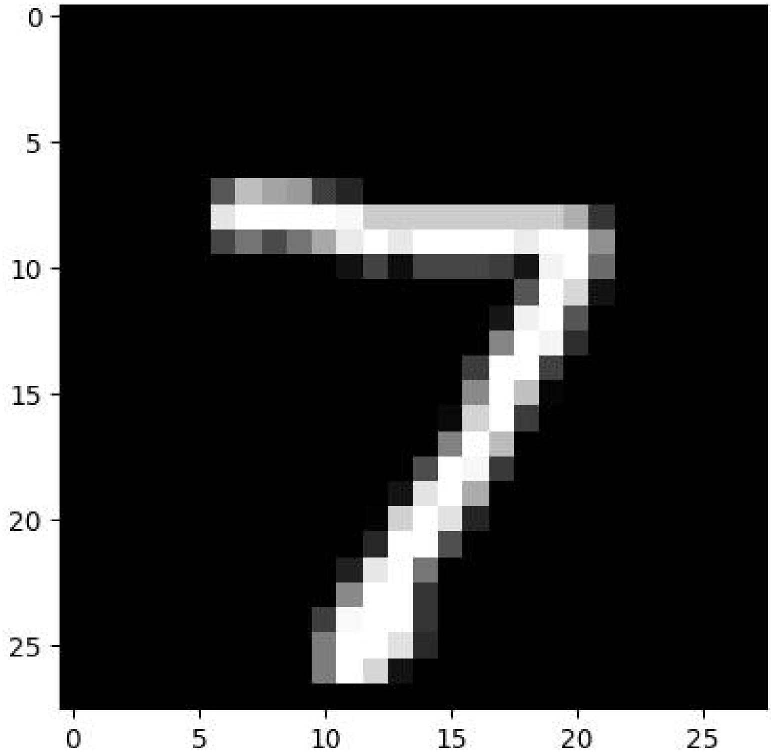
✓ 1분 [9]

```
# Load MNIST test set.
x, y = load_mnist(is_train=False, flatten=(train_config.model == "fc"))
x, y = x.to(device), y.to(device)

print(x.shape, y.shape)

input_size = int(x.shape[-1])
output_size = int(max(y)) + 1

model = get_model(
    input_size,
    output_size,
    train_config,
    device,
)

model.load_state_dict(model_dict)

test(model, x, y, to_be_shown=False)
```

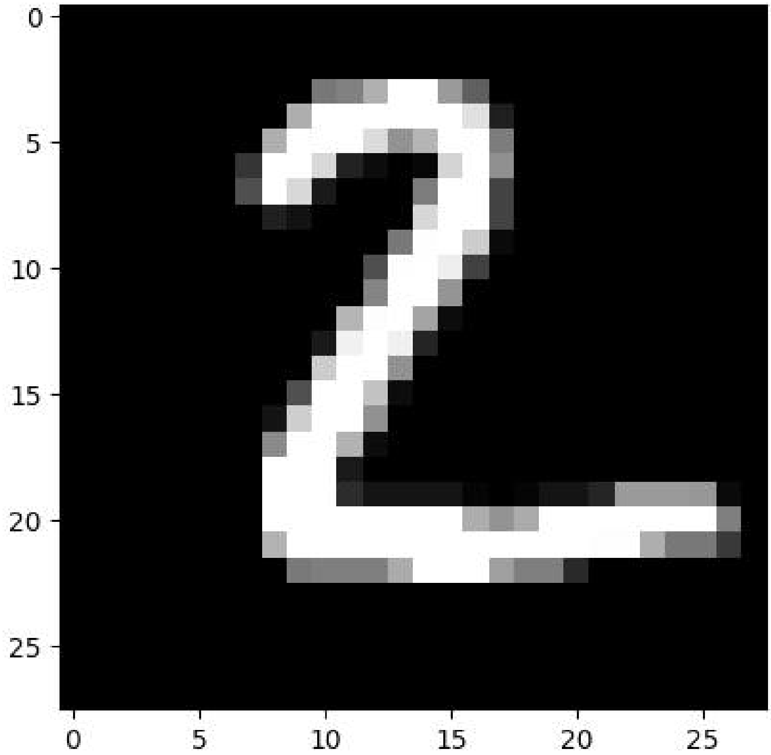
```
Downloading http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz to ../data/MNIST/raw/train-images-idx3-ubyte.gz
100%|██████████| 9912422/9912422 [00:00<00:00, 116121594.60it/s]
Extracting ../data/MNIST/raw/train-images-idx3-ubyte.gz to ../data/MNIST/raw

Downloading http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz to ../data/MNIST/raw/train-labels-idx1-ubyte.gz
100%|██████████| 28881/28881 [00:00<00:00, 44898329.81it/s]
Extracting ../data/MNIST/raw/train-labels-idx1-ubyte.gz to ../data/MNIST/raw

Downloading http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz to ../data/MNIST/raw/t10k-images-idx3-ubyte.gz
100%|██████████| 1648877/1648877 [00:00<00:00, 25443844.58it/s]
Extracting ../data/MNIST/raw/t10k-images-idx3-ubyte.gz to ../data/MNIST/raw

Downloading http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz to ../data/MNIST/raw/t10k-labels-idx1-ubyte.gz
100%|██████████| 4542/4542 [00:00<00:00, 12742828.61it/s]
Extracting ../data/MNIST/raw/t10k-labels-idx1-ubyte.gz to ../data/MNIST/raw

torch.Size([10000, 28, 28]) torch.Size([10000])
Accuracy: 0.9887
```

```
[10] n_test = 20
     test(model, x[:n_test], y[:n_test], to_be_shown=True)
```

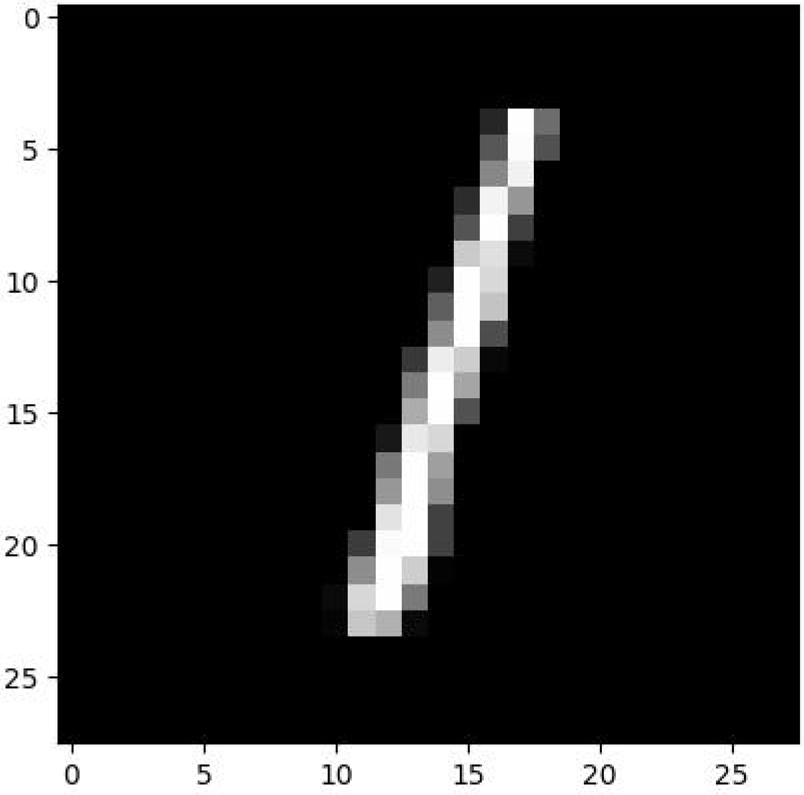
```
Accuracy: 0.9500
```



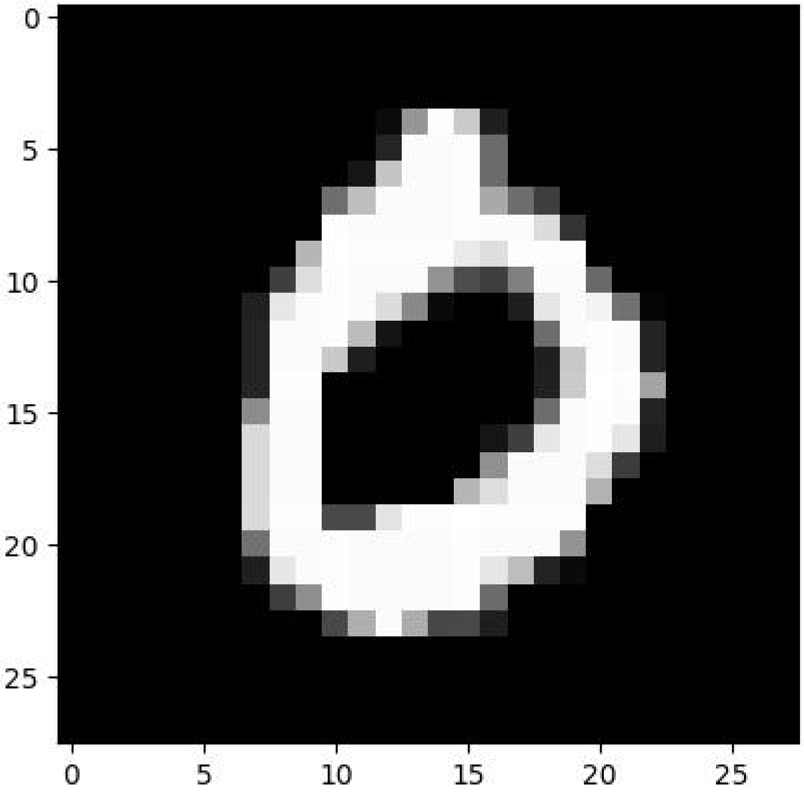
```
Predict: 7.0
```



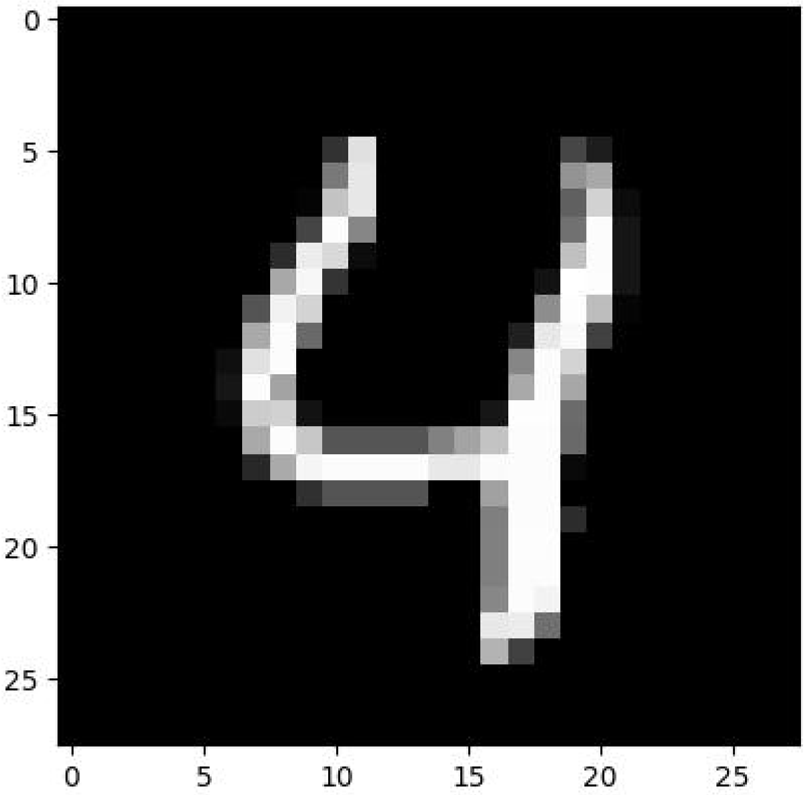
Predict: 2.0



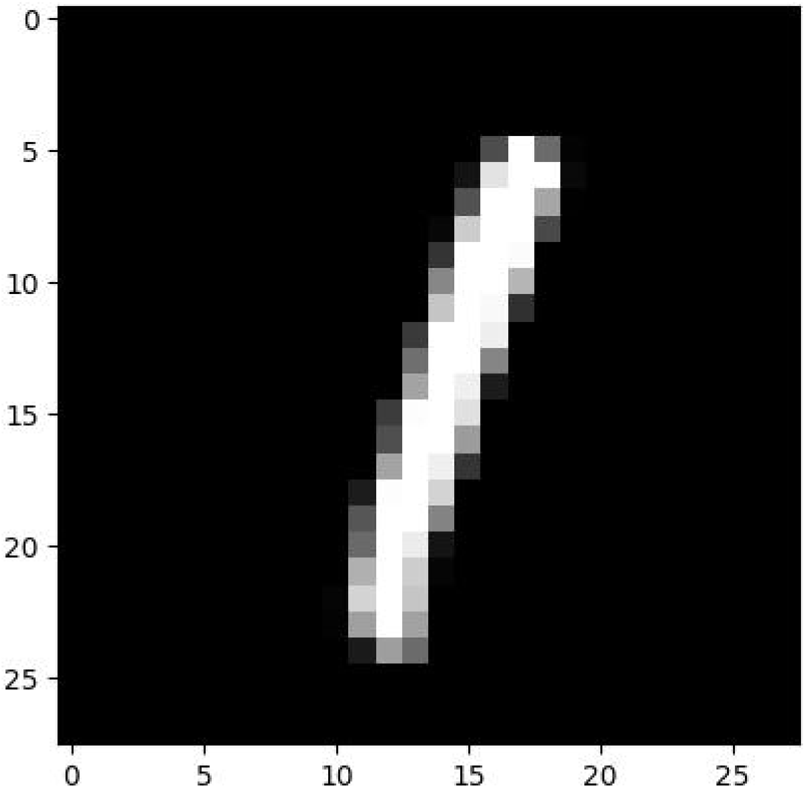
Predict: 1.0



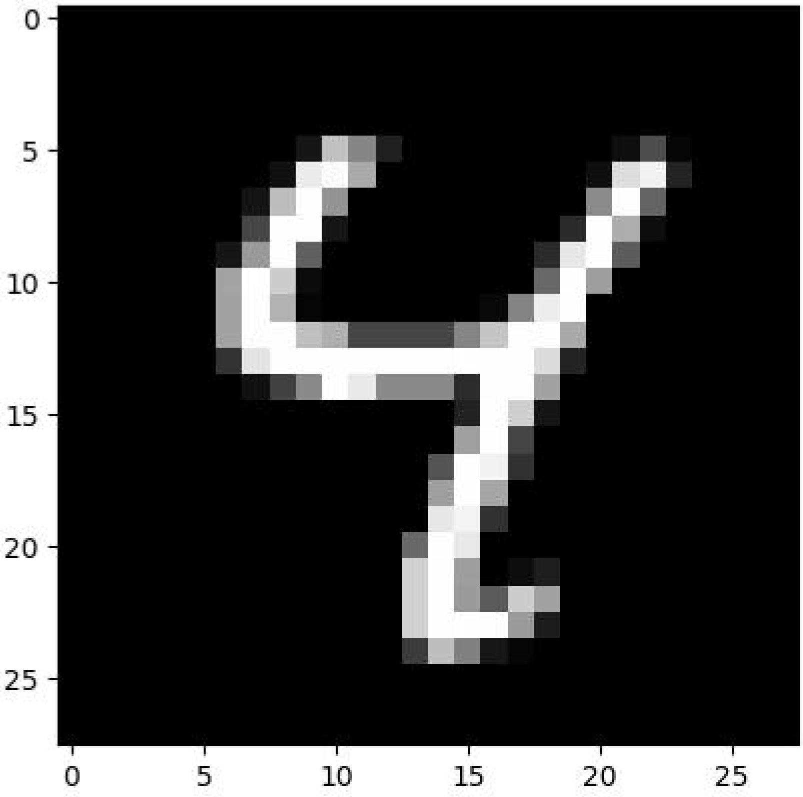
```
Predict: 0.0
```



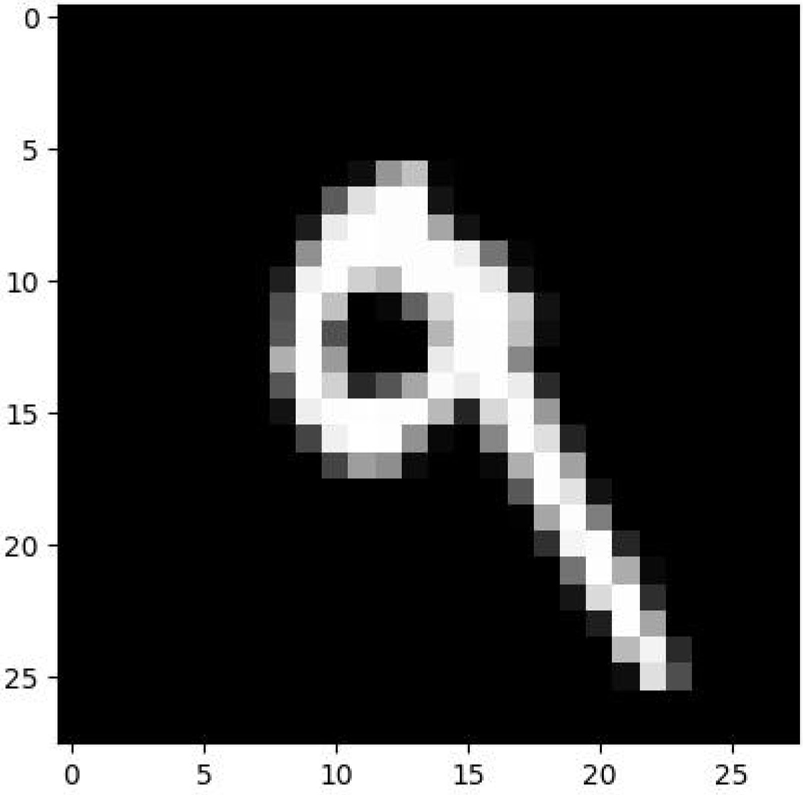
```
Predict: 4.0
```



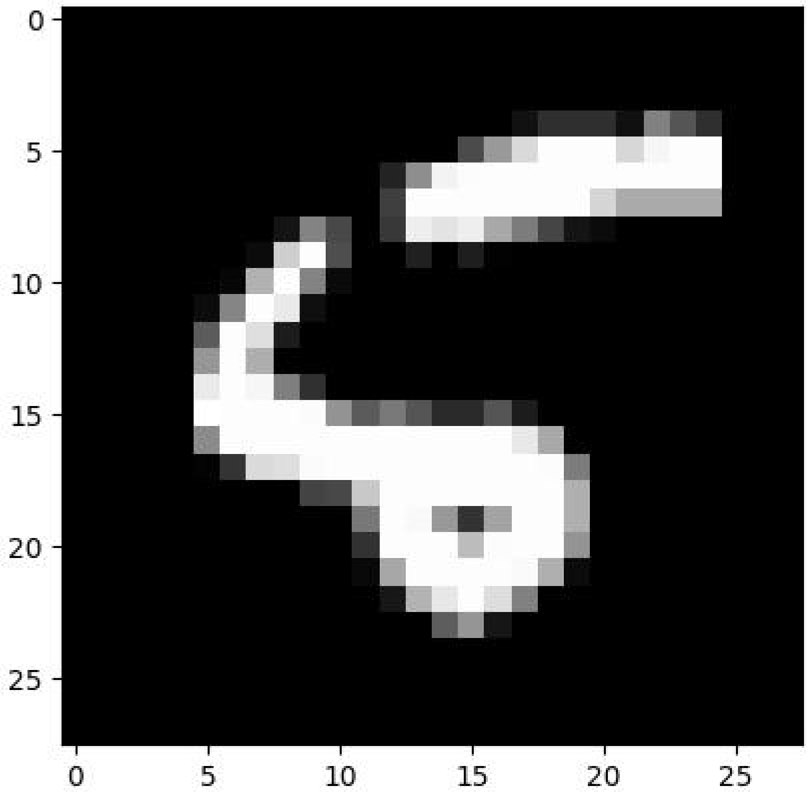
Predict: 1.0



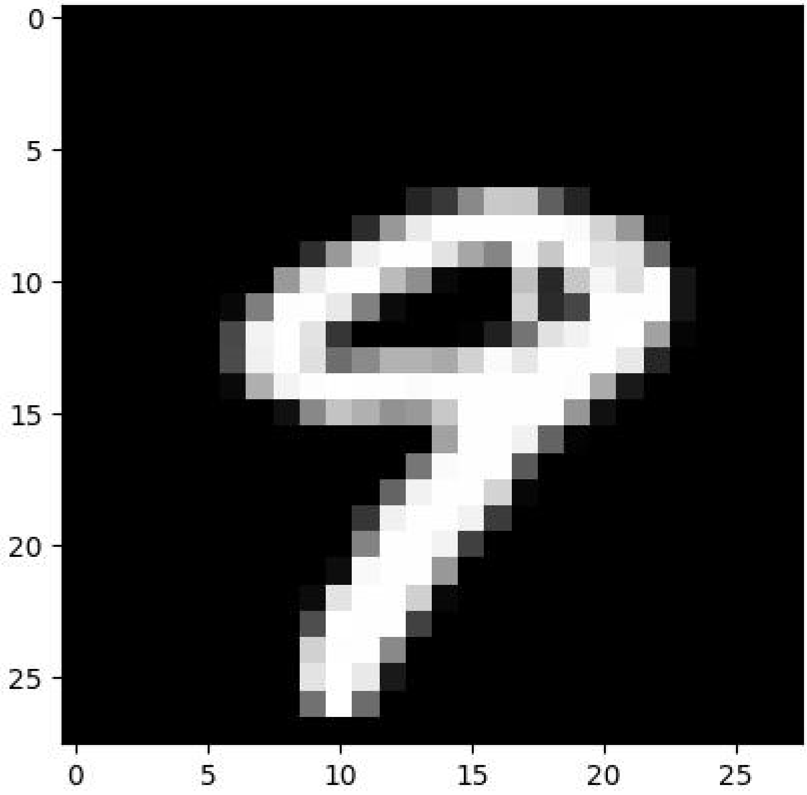
Predict: 4.0



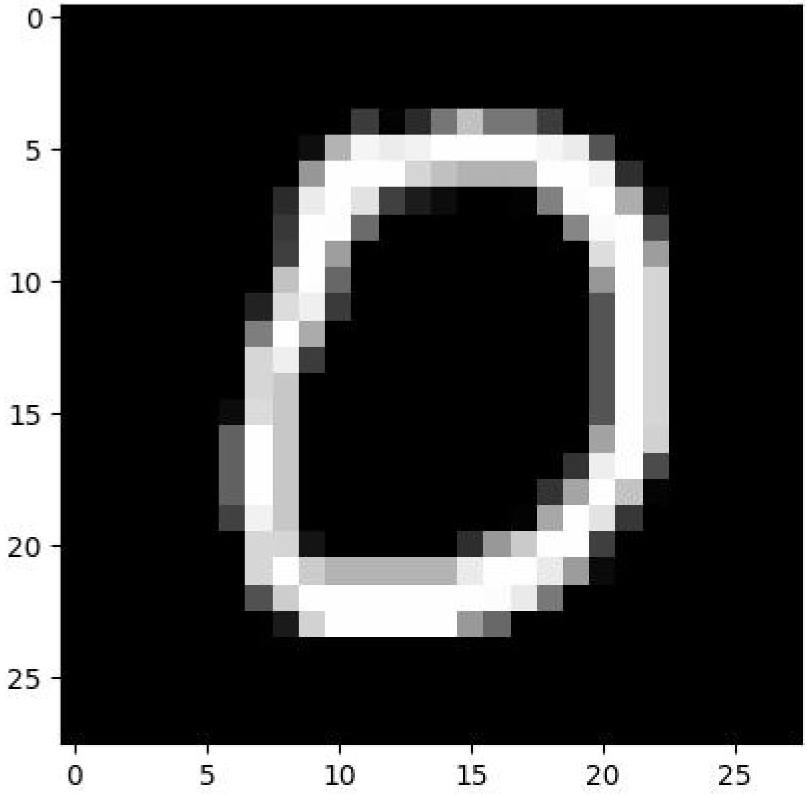
```
Predict: 9.0
```



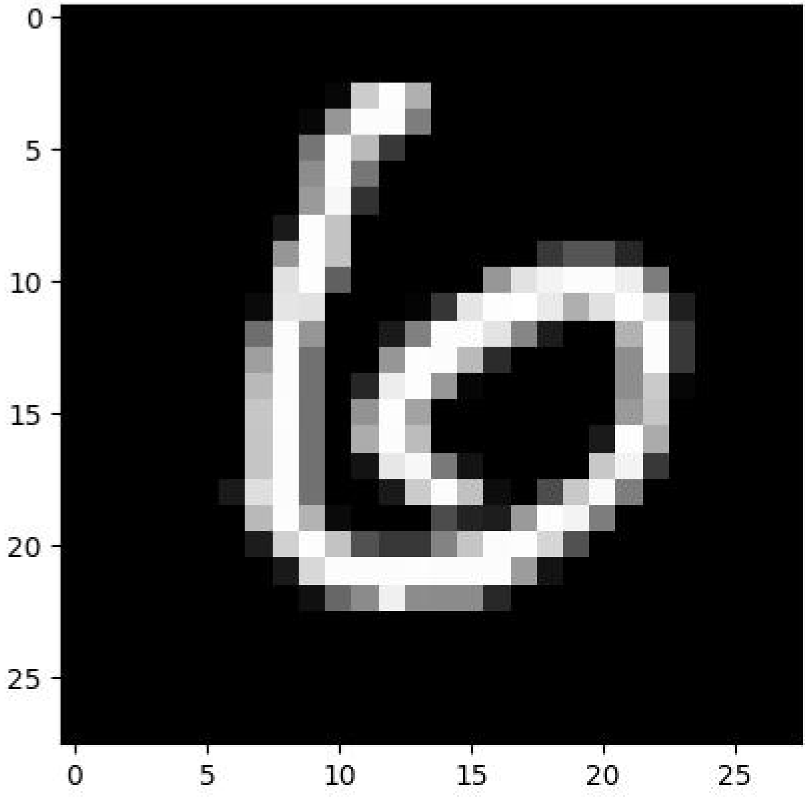
```
Predict: 8.0
```



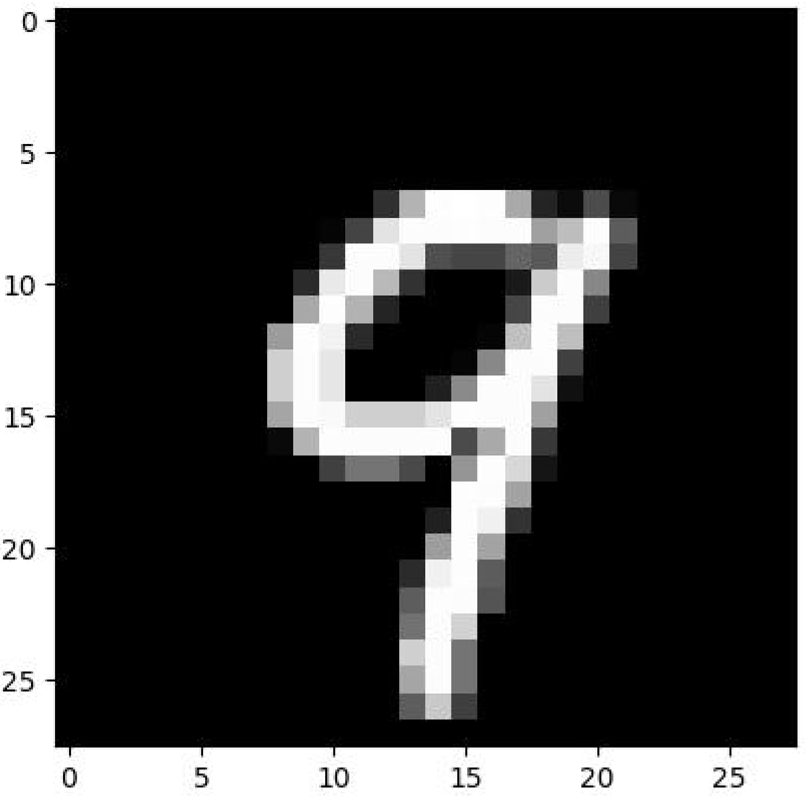
Predict: 9.0



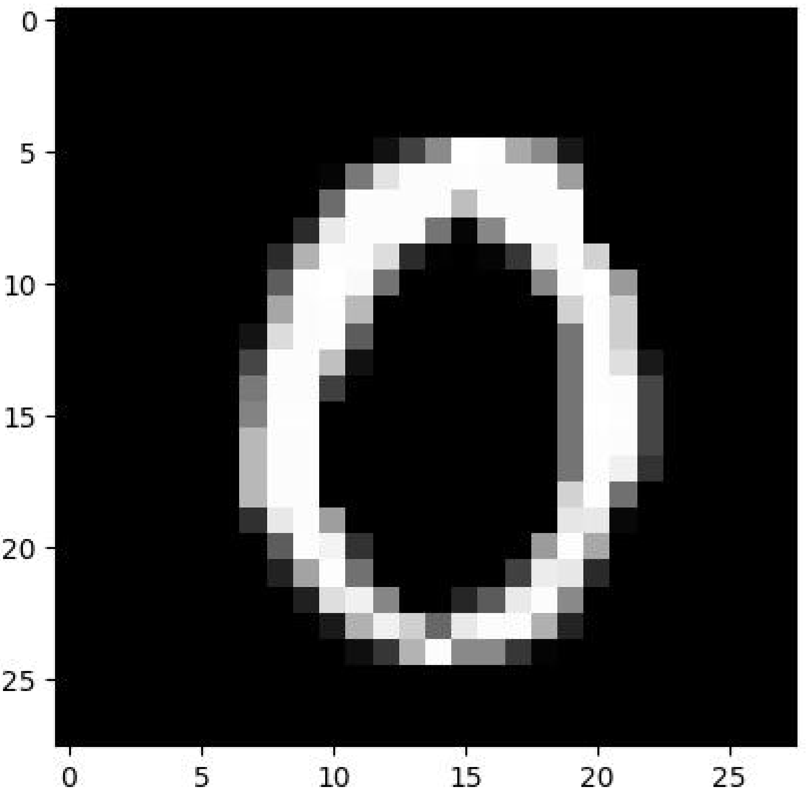
Predict: 0.0



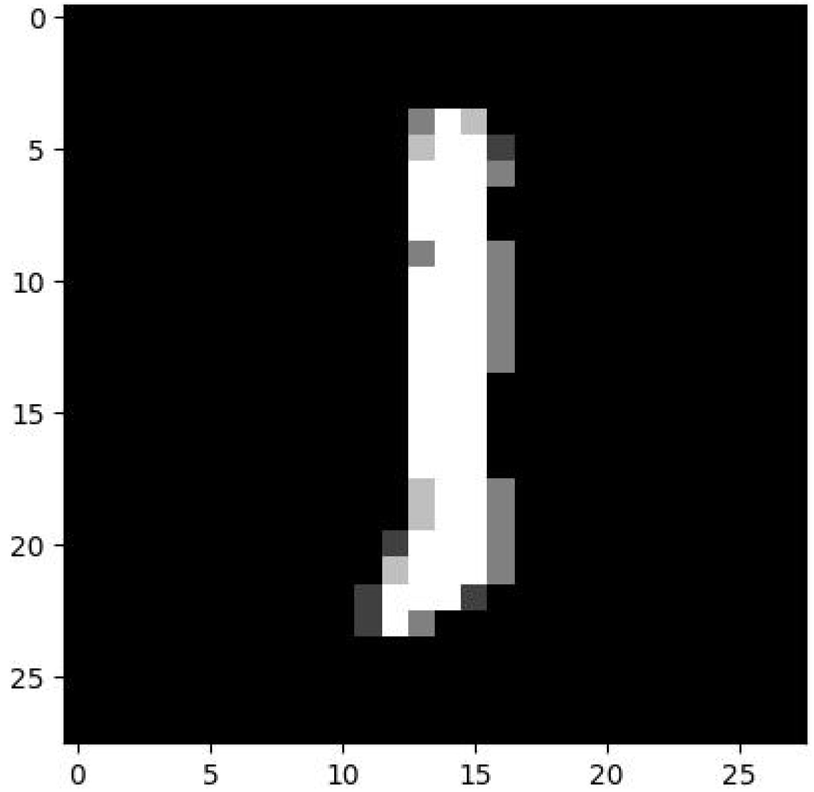
Predict: 6.0



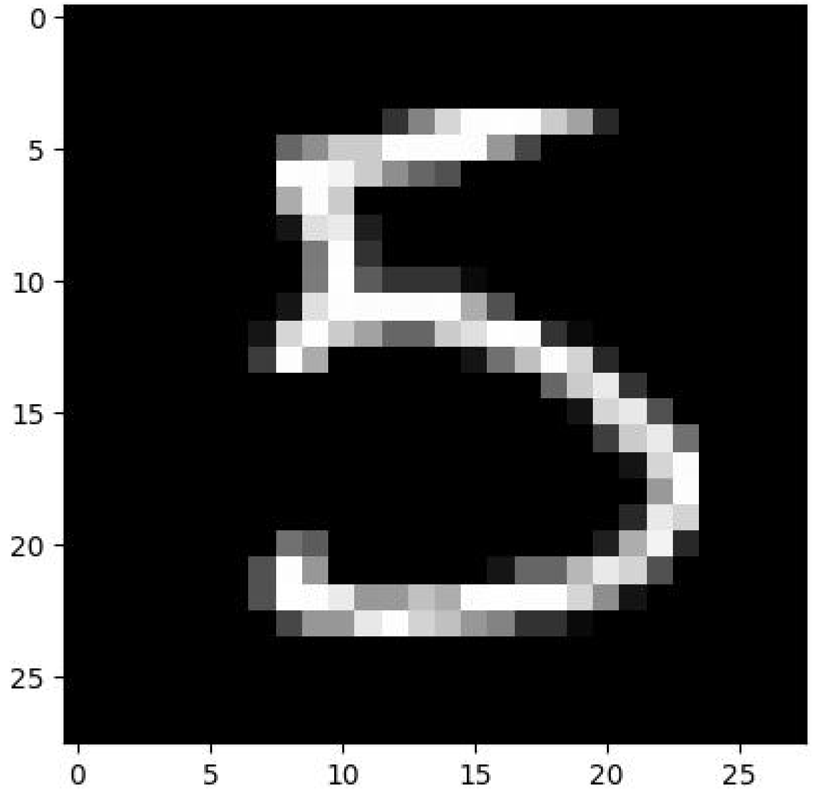
Predict: 9.0



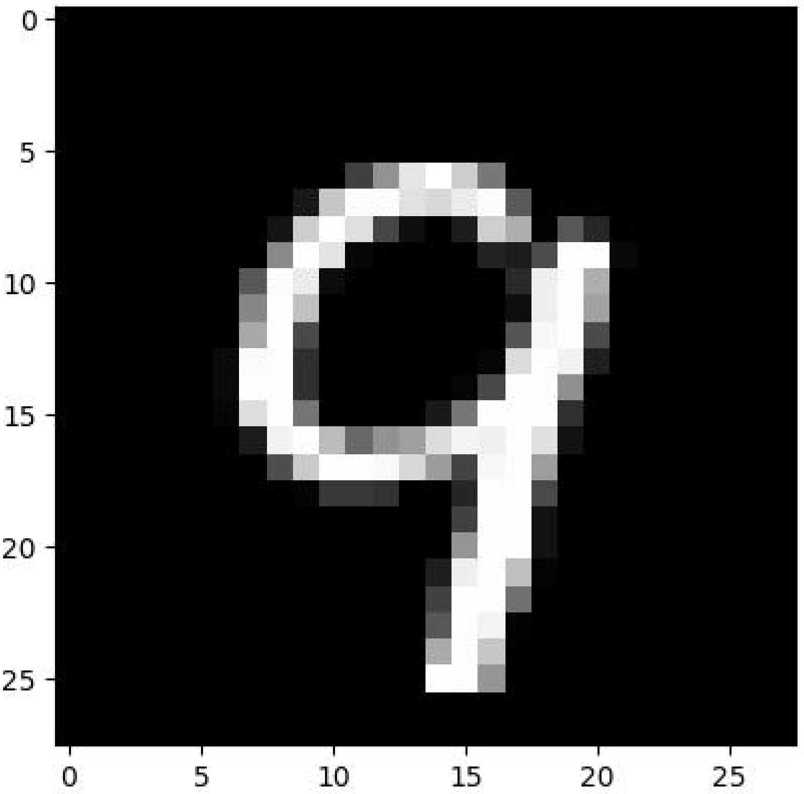
```
Predict: 0.0
```



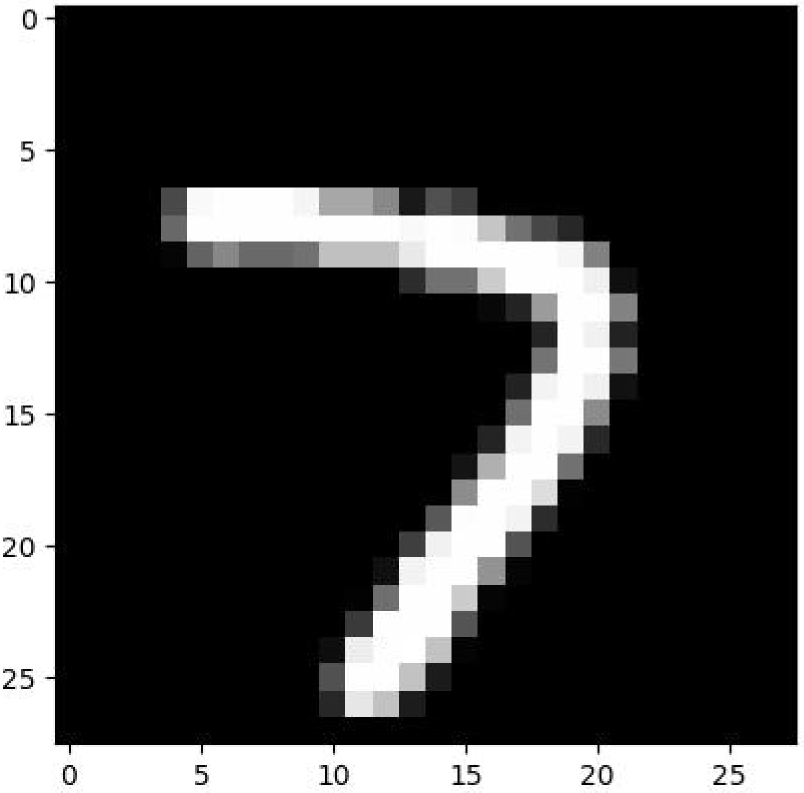
```
Predict: 1.0
```



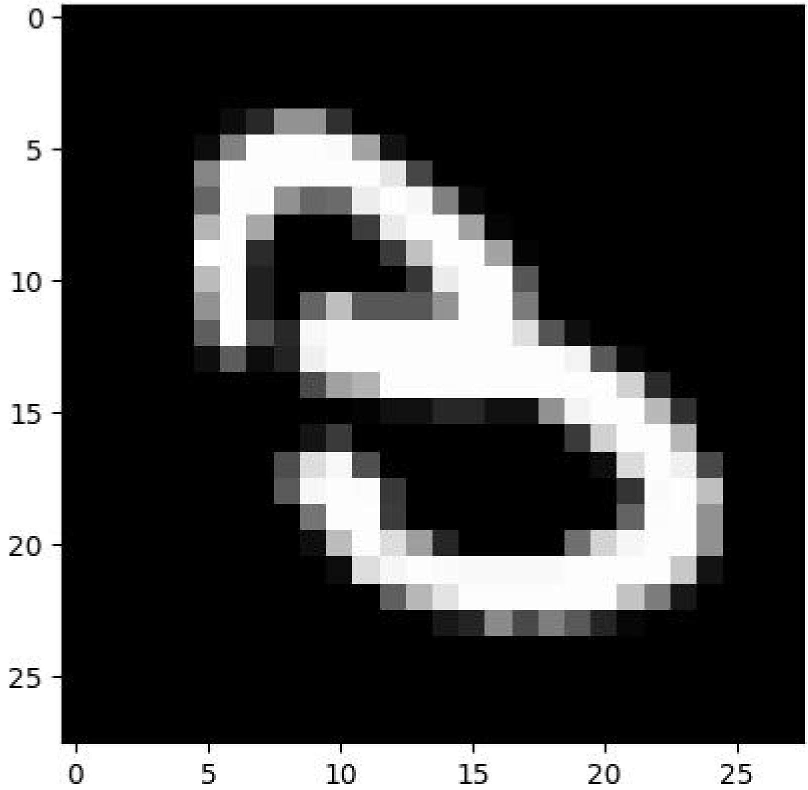
Predict: 5.0



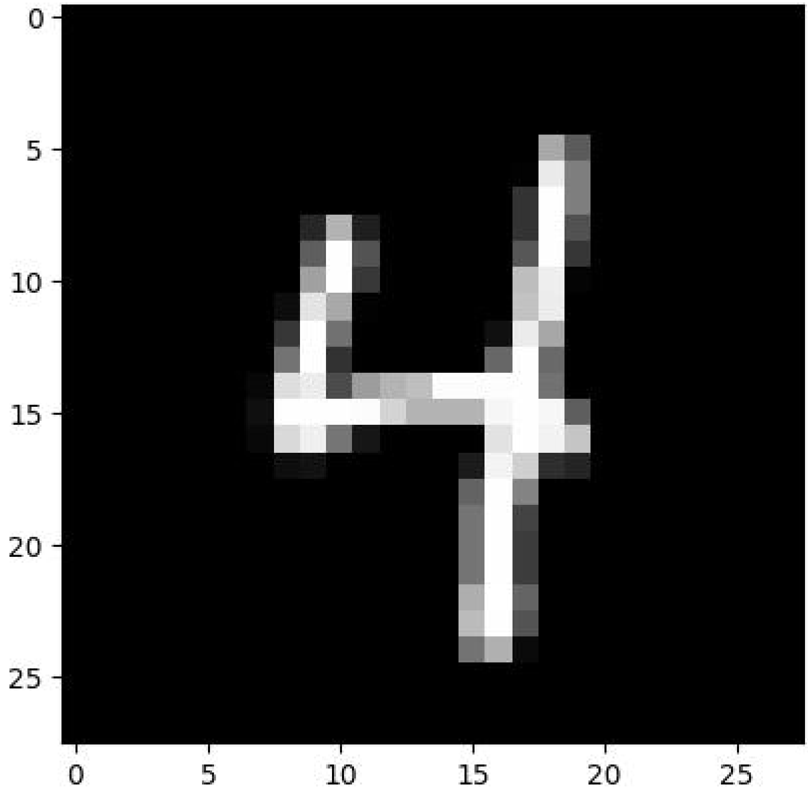
Predict: 9.0



```
Predict: 7.0
```

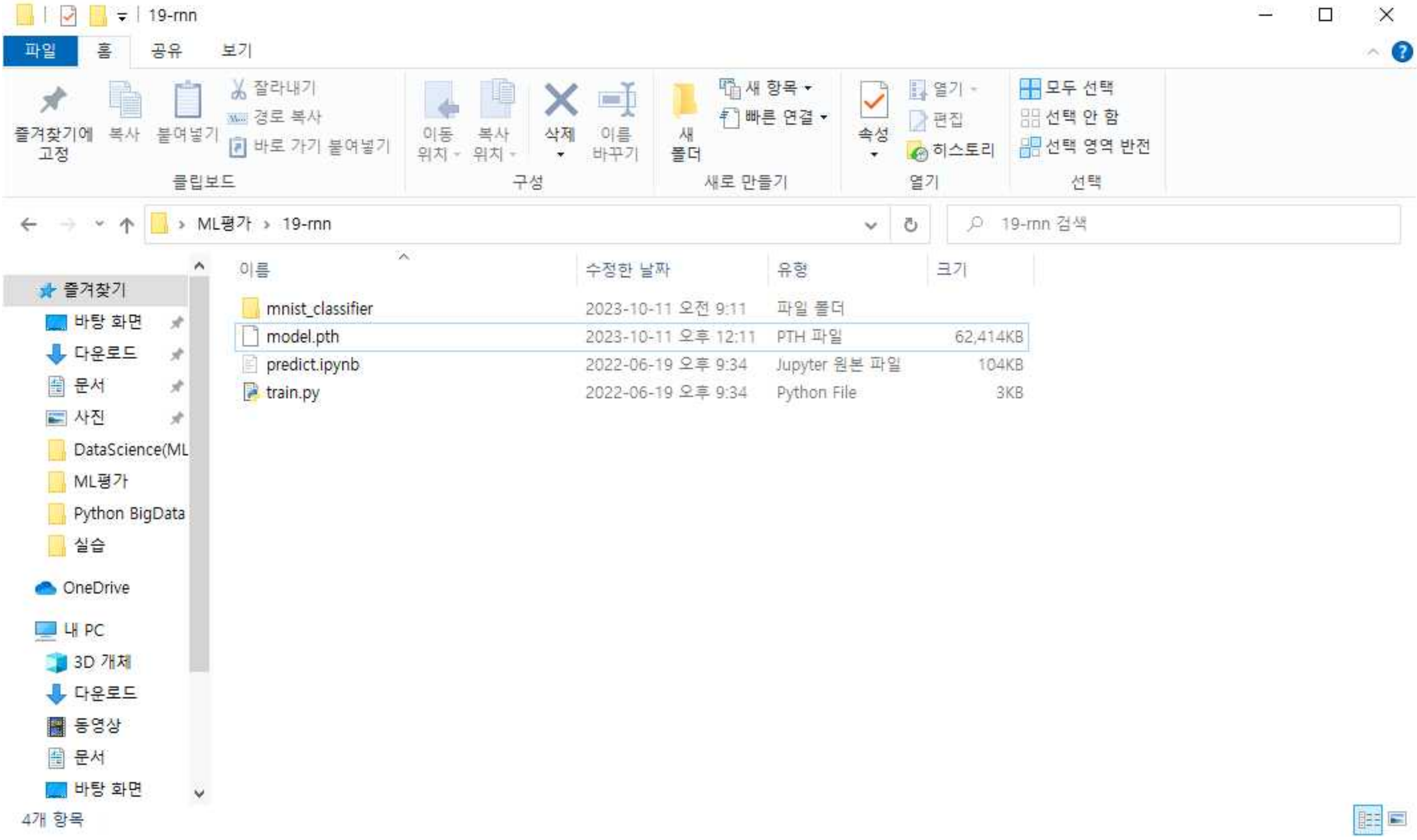


```
Predict: 3.0
```



```
Predict: 4.0
```

<작업폴더 화면 캡처>



이름	수정한 날짜	유형	크기
mnist_classifier	2023-10-11 오전 9:11	파일 폴더	
model.pth	2023-10-11 오후 12:11	PTH 파일	62,414KB
predict.ipynb	2022-06-19 오후 9:34	Jupyter 원본 파일	104KB
train.py	2022-06-19 오후 9:34	Python File	3KB

4개 항목