# Data Science(Machine Learning, Deep Learning)
- 소스 깃 참고

<RNN - 순환신경망>
Recurrent Neutral Networks
RNN은 시퀀스 데이터(어떤 항목 또는 이벤트가 순서대로 발생하는 데이터)를 처리하는 데 사용되는 인공 신경망의 한 유형이다.
기존의 신경망은 테이블 데이터나 이미지 데이터를 다룰 수 있음, RNN은 자연어 처리(NLP)와 같이 순서정보가 담긴 데이터나 시계열 데이터를 다루는데 적합한 것이 순환신경망 RNN이다.

<RNN 활용사례>
RNN이 가장 많이 활용되는 분야가 자연어 처리이다.
대화형 AI 등(Chat GPT, Bard 등)에서 활용이 많이 이용이 됨.

<RNN의 형태>
N:1, N:N, 1:N

<LSTM>
Long Short Term Memory
장단기 기억을 수행할 수 있는 구조모델
Gradient 소실 문제를 해결한 데이터 모델의 형태

<GRU>
Gated Recurrent Unit
LSTM의 단점을 보완한 데이터 모델형태

<Gradient Cliping>
RNN = BPTT Algorithm에 의해 학습
BPTT Algorithm =  순환 신경망(Recurrent Neural Networks, RNNs)에서 학습을 위해 사용되는 역전파 알고리즘의 한 형태이다.

Gradient 노름의 최댓값을 정하여 학습이 산으로 가지 않게 최댓값을 정하고 최댓값이 넘어갈 경우 Gradient를 강제로 줄이는 방법이 Gradient Cliping
즉, 벡터의 크기는 줄이되 방향을 유지하도록 클리핑 작업을 수행하는 것을 의미한다.

<LSTM으로 MNIST 분류 구현>
앞선 CNN에서 했던 프로젝트 코드를 LSTM으로 MNIST를 구현
N:1의 형태이기에 양방향 RNN을 사용
RNN의 학습
nn.LSTM 클래스 활용

CNN, FC, RNN 3가지가 다 추가 되어있음

cd C:\Users\admin\Desktop\BigData DeepLearning\DeepLearning 소스코드\실습\19-rnn

<rnn.py>
```
self.rnn = nn.LSTM(
            input_size=input_size,
            hidden_size=hidden_size,
            num_layers=n_layers,
```

```
            batch_first=True,
            dropout=dropout_p,
            bidirectional=True,
        )
# 이것을 가지고 아래의 self.layer를 만듬

        self.layers = nn.Sequential(
            nn.ReLU(),
            nn.BatchNorm1d(hidden_size * 2),
            nn.Linear(hidden_size * 2, output_size),
            nn.LogSoftmax(dim=-1),
        )
```

<train.py, utils.py 수정>
[train.py]
p.add_argument("--model", default="fc", choices=["fc", "cnn", "rnn"]) - 인자값 추가(모델 추가)
p.add_argument("--hidden_size", type=int, default=128) 추가함. - LSTM의 은닉 상태 크기 설정

[utils.py]
```
def get_model(input_size, output_size, config, device):
    if config.model == "fc":
        model = ImageClassifier(
            input_size=input_size,
            output_size=output_size,
            hidden_sizes=get_hidden_sizes(
                input_size,
                output_size,
                config.n_layers
            ),
            use_batch_norm=not config.use_dropout,
            dropout_p=config.dropout_p,
        )
    elif config.model == "cnn":
        model = ConvolutionalClassifier(output_size)
    elif config.model == "rnn":
        model = SequenceClassifier(
            input_size=input_size,
            hidden_size=config.hidden_size,
            output_size=output_size,
            n_layers=config.n_layers,
            dropout_p=config.dropout_p,
        )
    else:
        raise NotImplementedError
    return model
```

수정 끝 RNN기반의 MNIST 학습

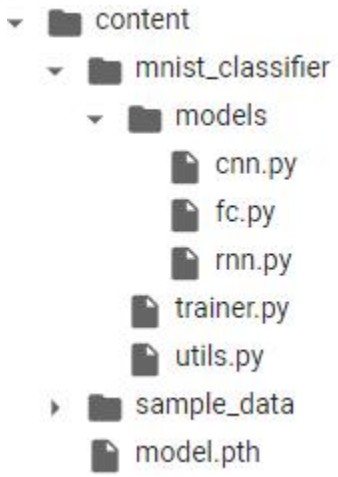
(base) C:\Users\admin\Desktop\BigData DeepLearning\DeepLearning 소스코드\실습\19-rnn>python train.py --model_fn ./model.pth --n_epochs 20 --model rnn

```
--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=4.5190e-01  valid_loss=1.2544e-01  lowest_loss=1.2544e-01
Epoch(2/20): train_loss=1.2404e-01  valid_loss=1.0763e-01  lowest_loss=1.0763e-01
Epoch(3/20): train_loss=8.1964e-02  valid_loss=7.3950e-02  lowest_loss=7.3950e-02
Epoch(4/20): train_loss=6.7142e-02  valid_loss=6.4672e-02  lowest_loss=6.4672e-02
Epoch(5/20): train_loss=5.2731e-02  valid_loss=5.9712e-02  lowest_loss=5.9712e-02
Epoch(6/20): train_loss=4.4674e-02  valid_loss=8.1173e-02  lowest_loss=5.9712e-02
Epoch(7/20): train_loss=4.2149e-02  valid_loss=5.1743e-02  lowest_loss=5.1743e-02
Epoch(8/20): train_loss=3.4009e-02  valid_loss=5.2396e-02  lowest_loss=5.1743e-02
Epoch(9/20): train_loss=3.1898e-02  valid_loss=4.9468e-02  lowest_loss=4.9468e-02
Epoch(10/20): train_loss=2.7877e-02  valid_loss=4.2063e-02  lowest_loss=4.2063e-02
Epoch(11/20): train_loss=2.8139e-02  valid_loss=4.5317e-02  lowest_loss=4.2063e-02
Epoch(12/20): train_loss=2.4396e-02  valid_loss=4.4507e-02  lowest_loss=4.2063e-02
Epoch(13/20): train_loss=2.1337e-02  valid_loss=4.1087e-02  lowest_loss=4.1087e-02
Epoch(14/20): train_loss=2.3014e-02  valid_loss=4.2287e-02  lowest_loss=4.1087e-02
Epoch(15/20): train_loss=1.9760e-02  valid_loss=4.5443e-02  lowest_loss=4.1087e-02
Epoch(16/20): train_loss=1.9895e-02  valid_loss=4.6152e-02  lowest_loss=4.1087e-02
Epoch(17/20): train_loss=1.7195e-02  valid_loss=4.4830e-02  lowest_loss=4.1087e-02
Epoch(18/20): train_loss=1.6456e-02  valid_loss=4.6463e-02  lowest_loss=4.1087e-02
Epoch(19/20): train_loss=1.7726e-02  valid_loss=5.2325e-02  lowest_loss=4.1087e-02
Epoch(20/20): train_loss=1.4901e-02  valid_loss=5.0110e-02  lowest_loss=4.1087e-02
```

이와 같이 나타난다.

<predict.ipynb>
이 파일은 수정 없이 실행하면 됩니다.

Colab에서 train.py로 만든 model.pth content로 이동 시키기
mnist_classifier 폴더 생성 trainer.py utils.py 이동
하위 폴더로 models 폴더 생성 rnn.py cnn.py fc.py 이동



이와 같이 하면 됨.

<MNIST 결과>

```
# 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, 166165924.26it/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, 36530667.62it/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

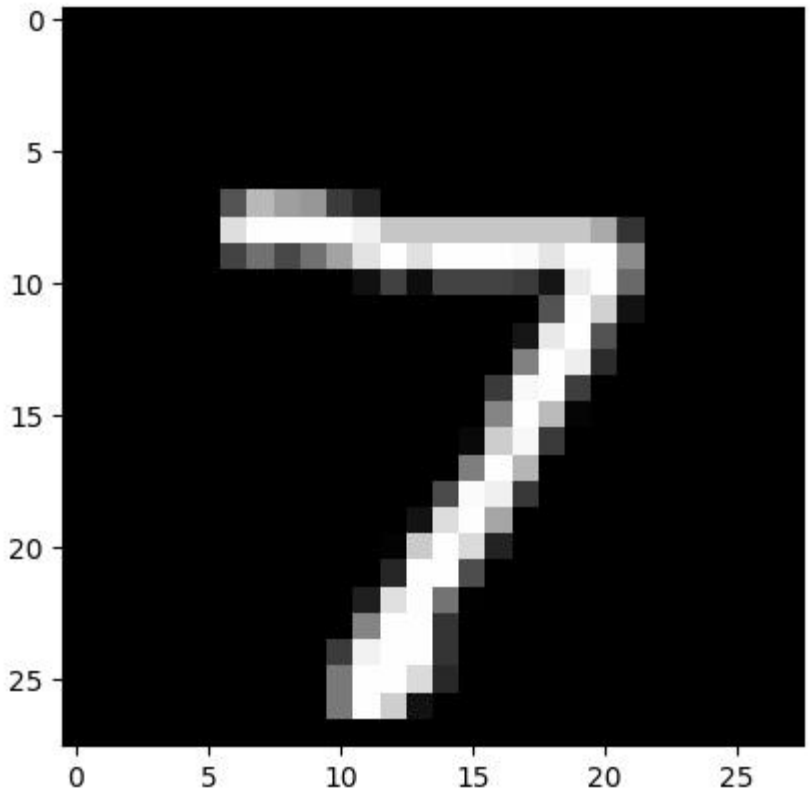
100%|████████████████| 1648877/1648877 [00:00<00:00, 34956285.74it/s]
Extracting ../data/MNIST/raw/t10k-images-idx3-ubyte.gz to ../data/MNIST/raw

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, 5892523.59it/s]
Extracting ../data/MNIST/raw/t10k-labels-idx1-ubyte.gz to ../data/MNIST/raw

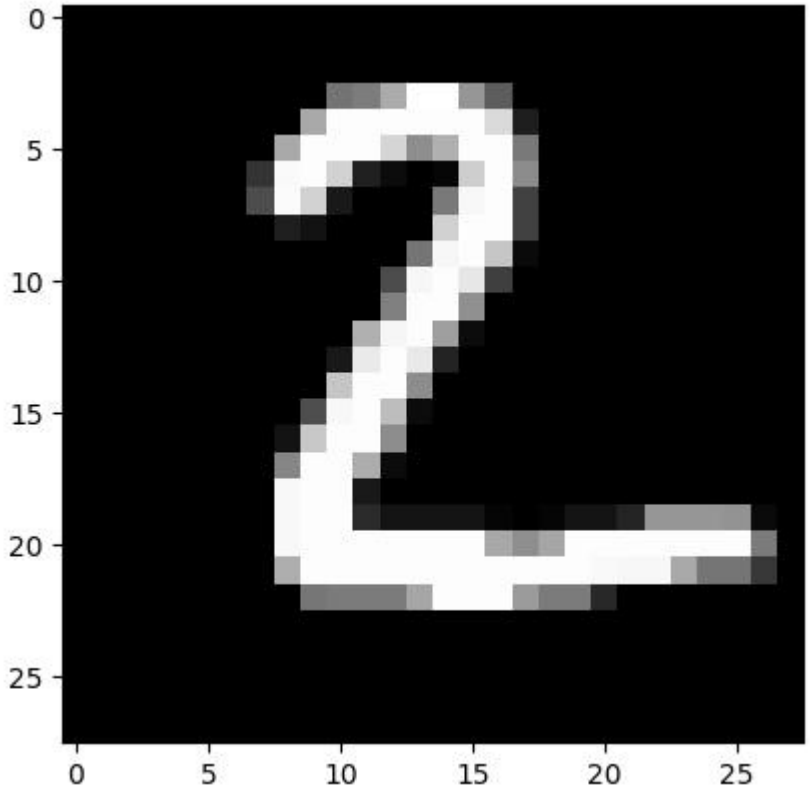
torch.Size([10000, 28, 28]) torch.Size([10000])
Accuracy: 0.9875

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

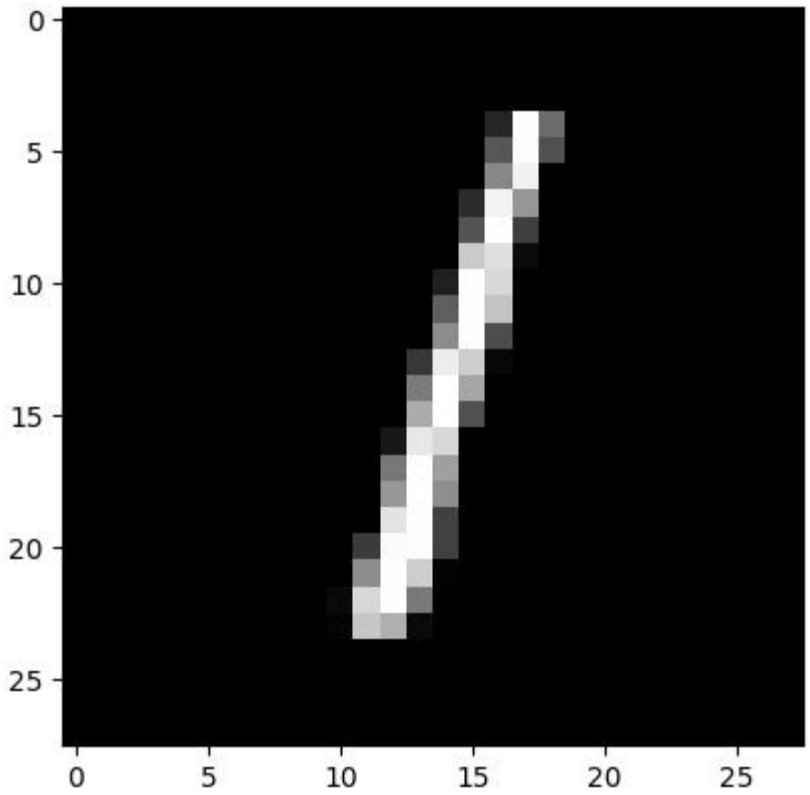
Accuracy: 1.0000



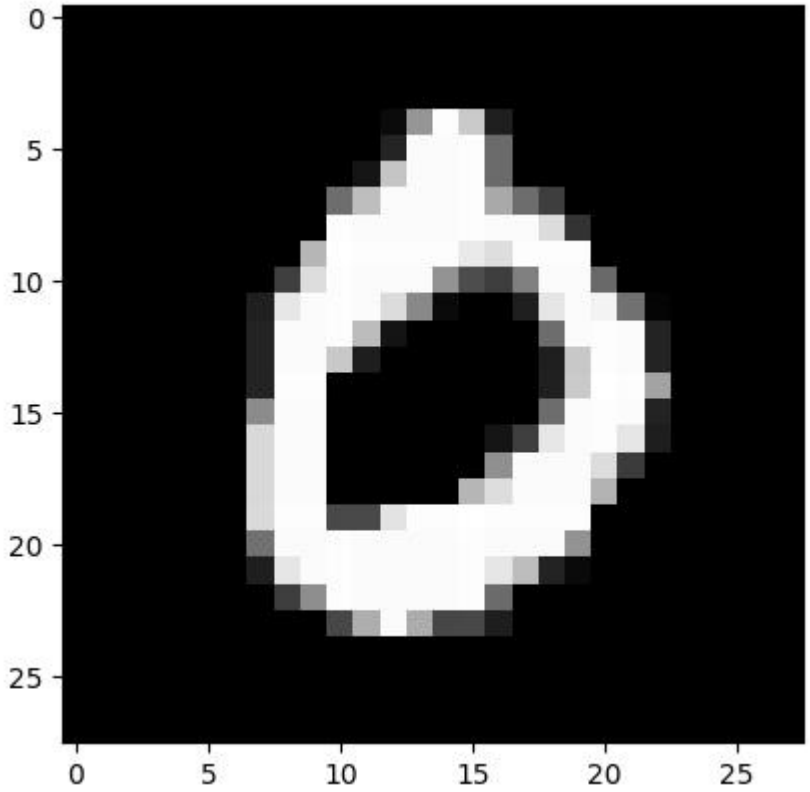
Predict: 7.0



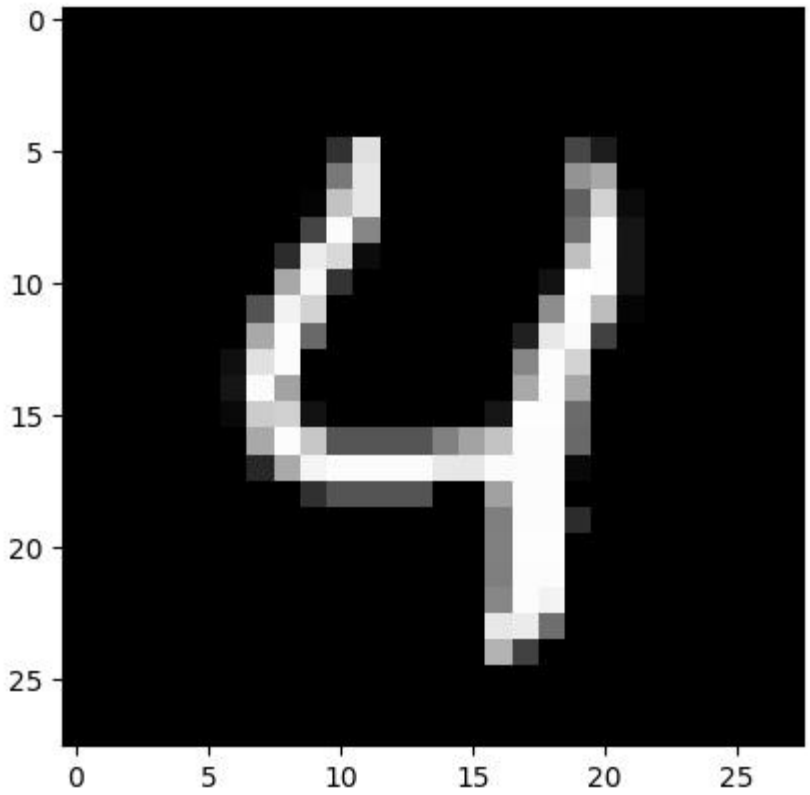

Predict: 2.0



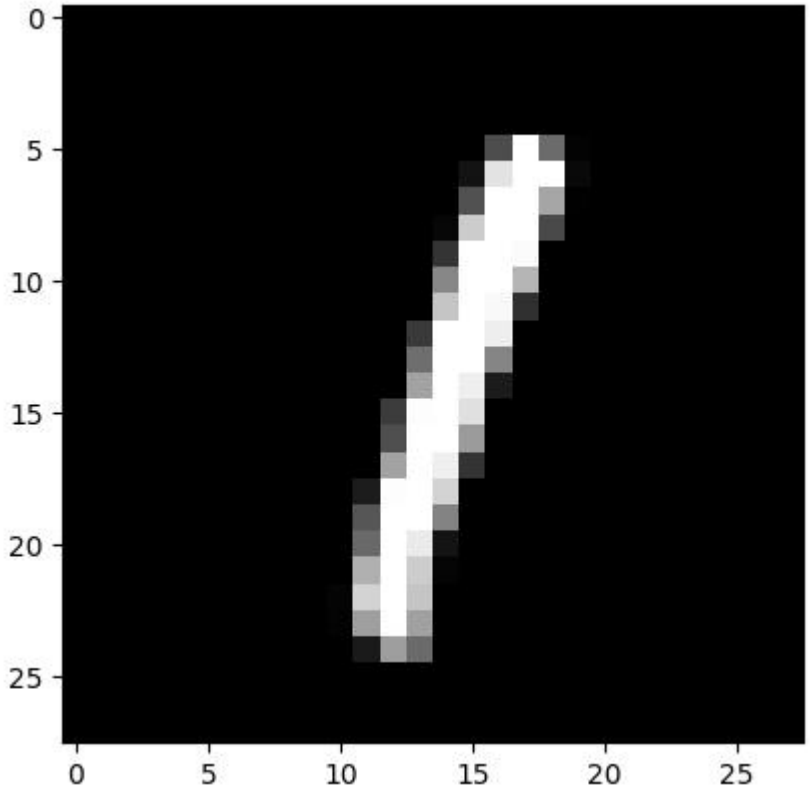
Predict: 1.0



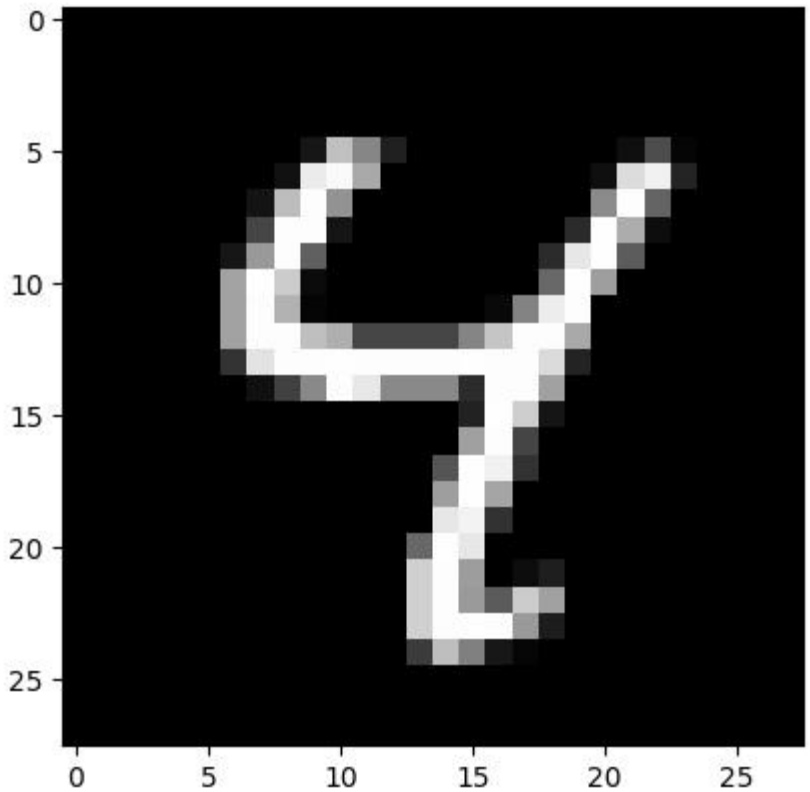

Predict: 0.0



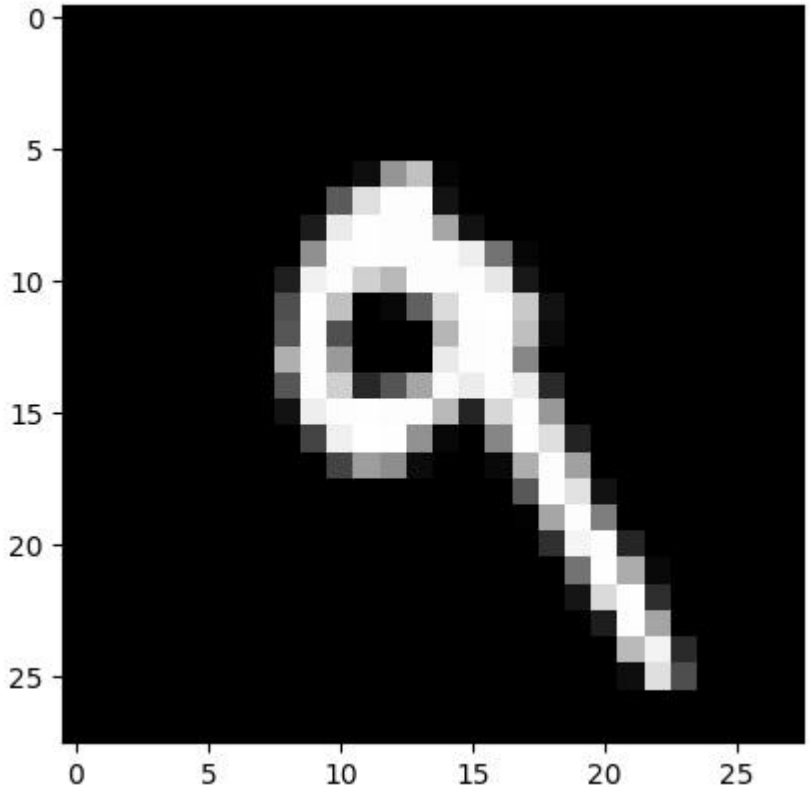
Predict: 4.0



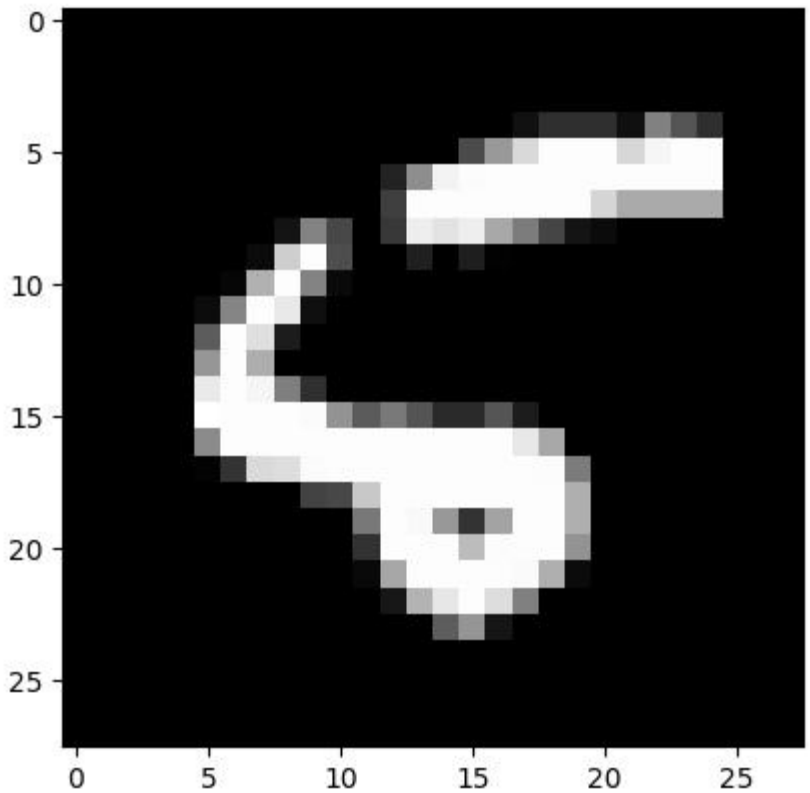

Predict: 1.0



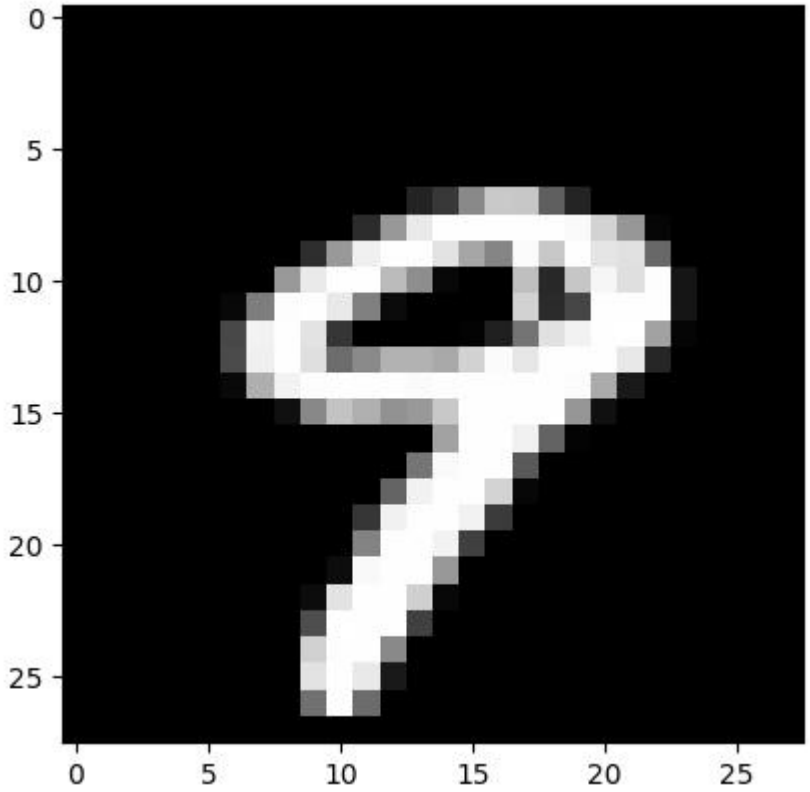
Predict: 4.0



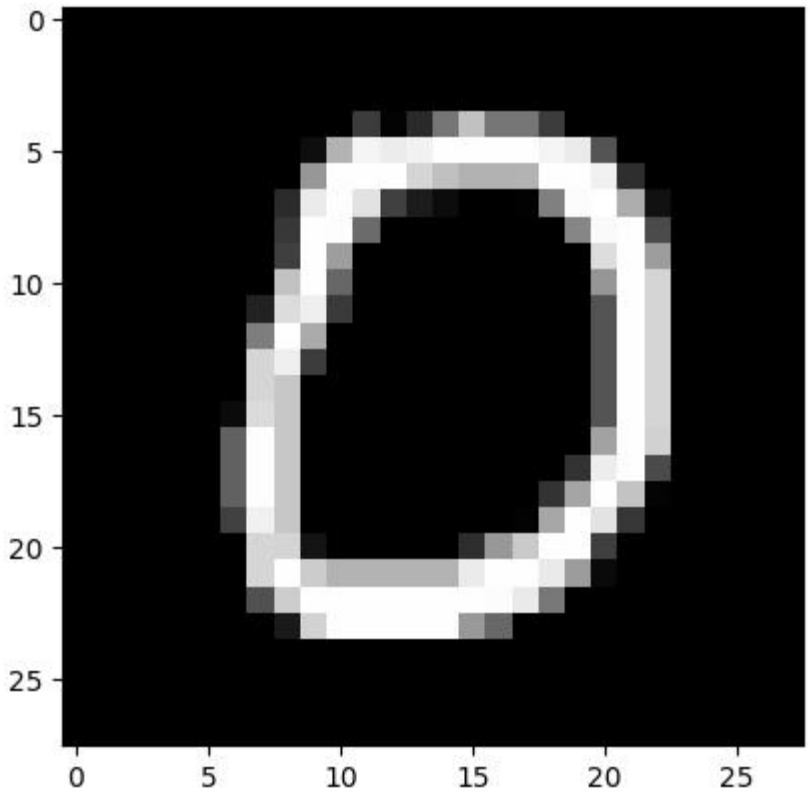

Predict: 9.0



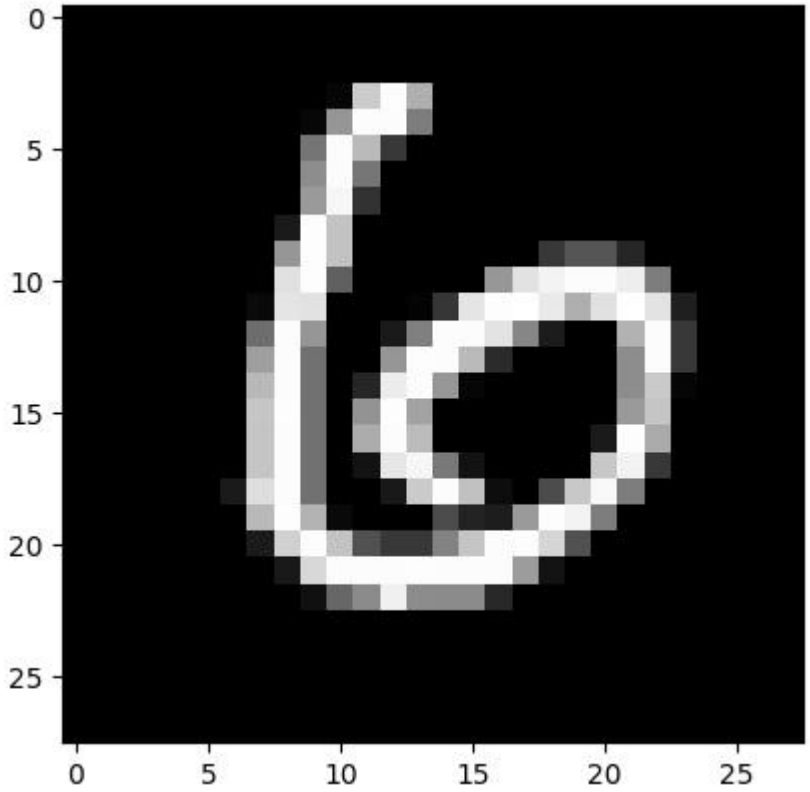
Predict: 5.0



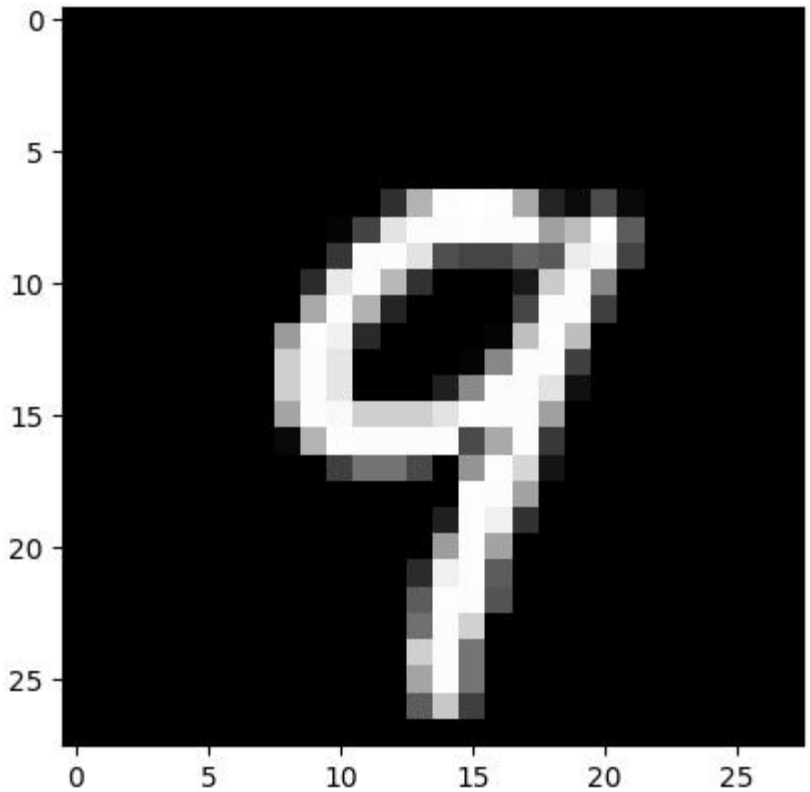

Predict: 9.0



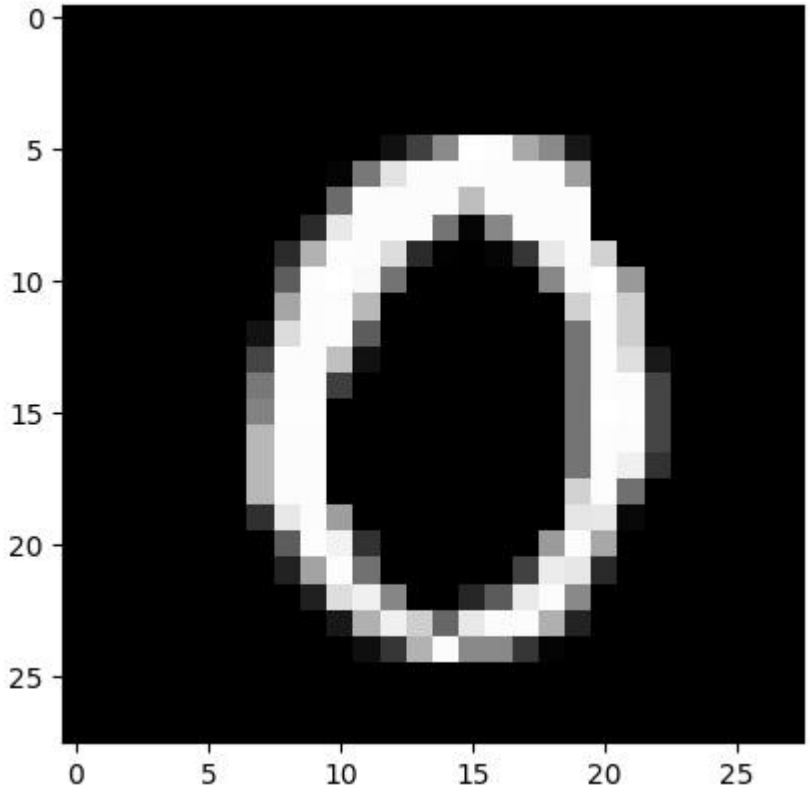
Predict: 0.0



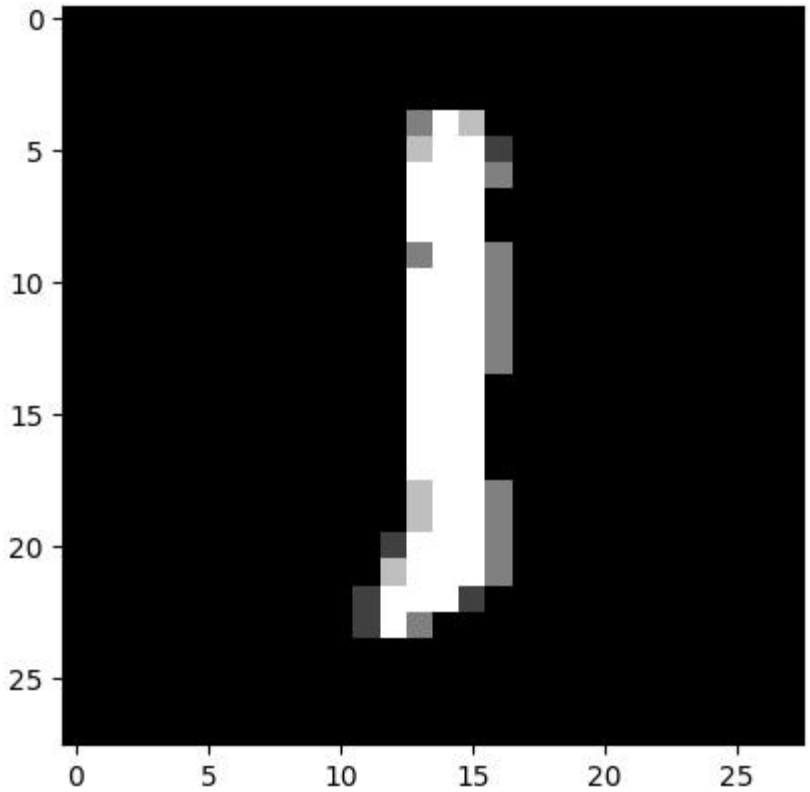

Predict: 6.0



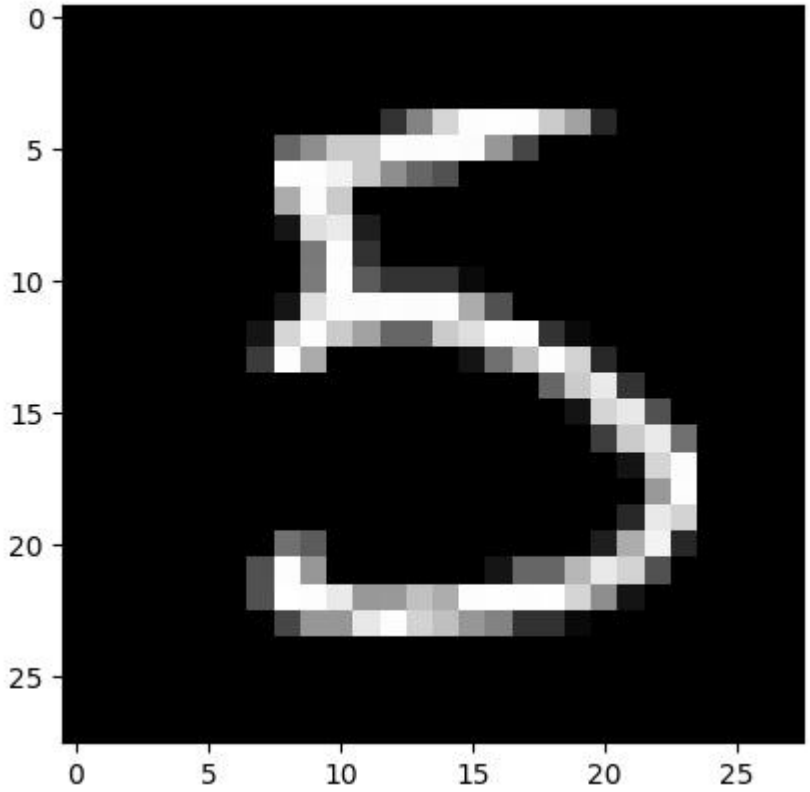
Predict: 9.0



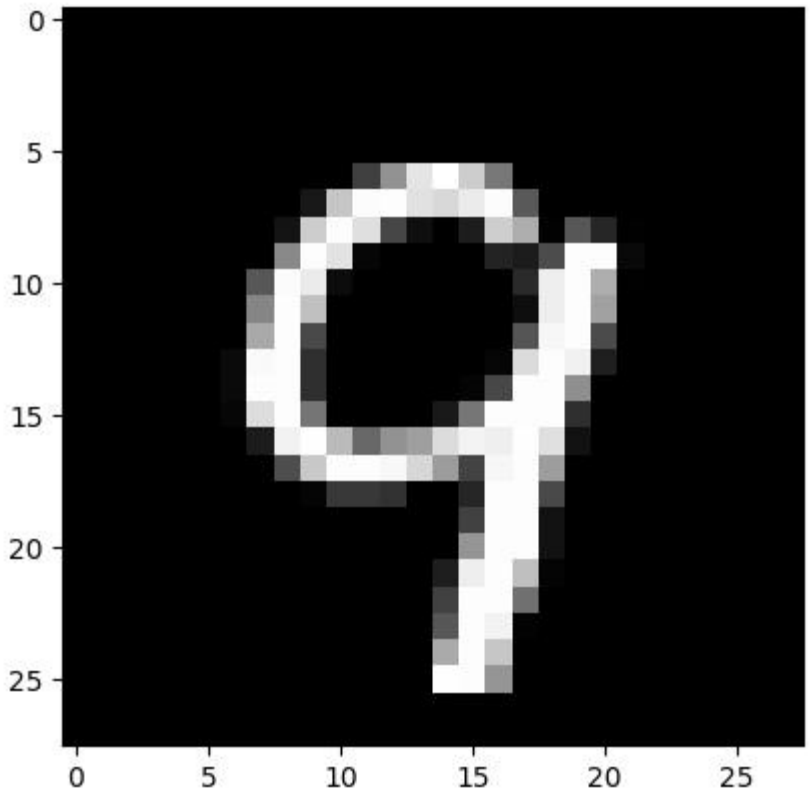

Predict: 0.0



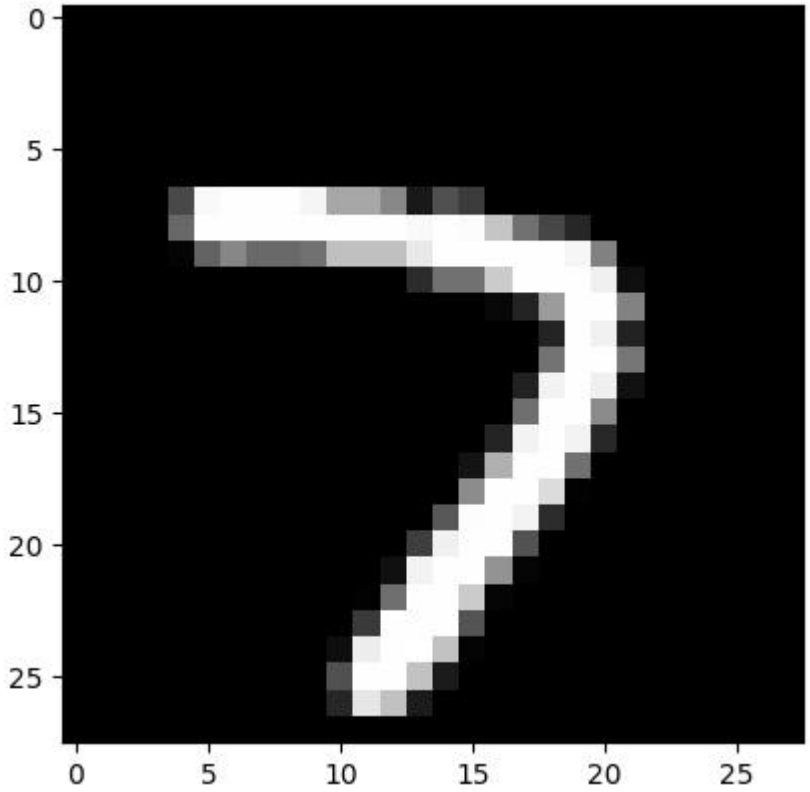
Predict: 1.0



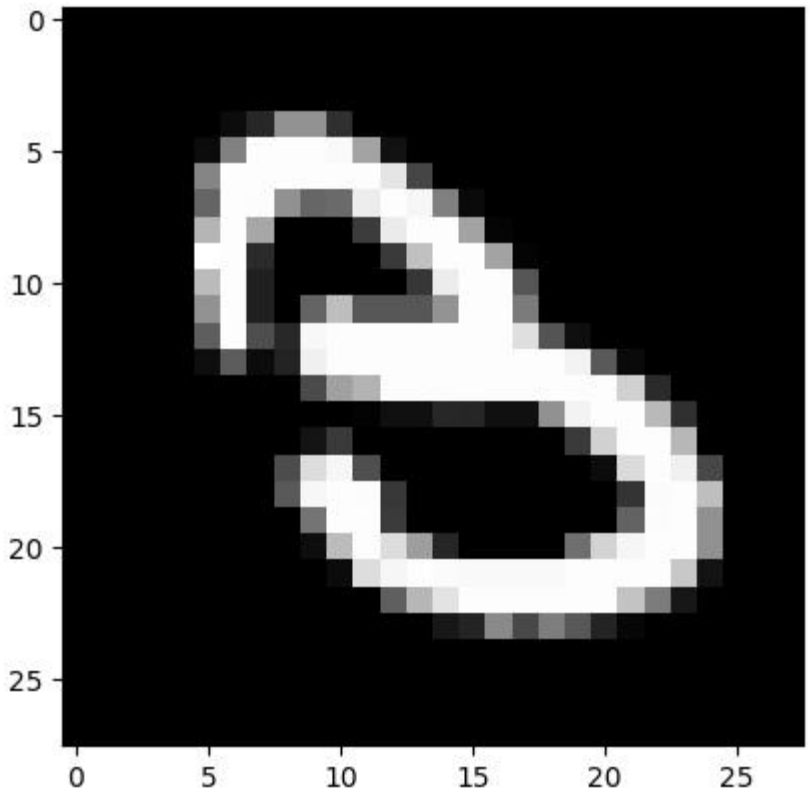

Predict: 5.0



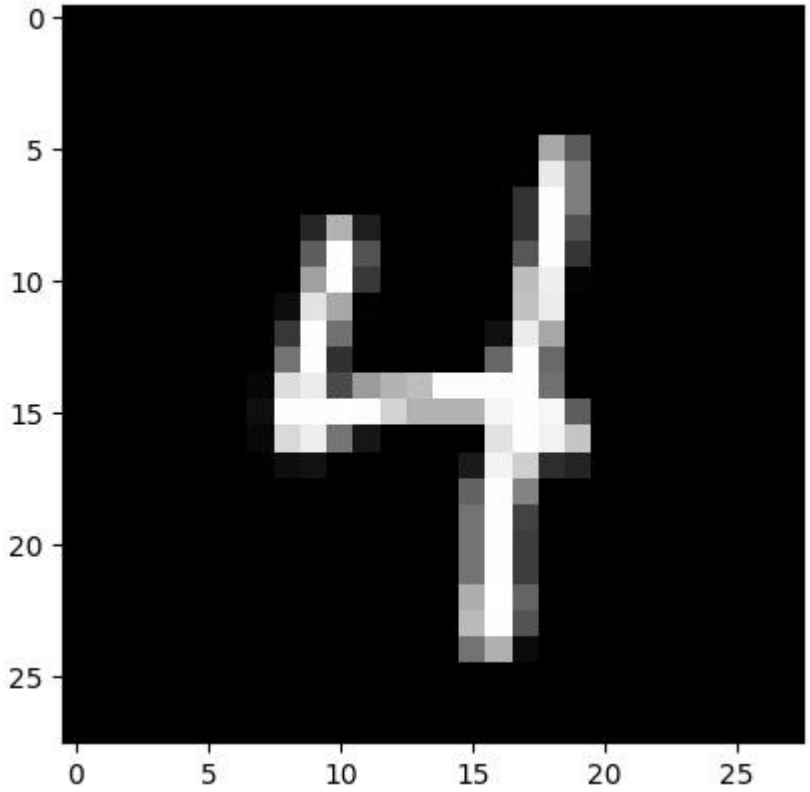
Predict: 9.0




Predict: 7.0



Predict: 3.0




Predict: 4.0

이와 같은 형태로 나타난다.