Practical Exercise with MNIST Example

In [1]:
```python
import torch
import torch.nn
```

```
c:\Users\Infornet\anaconda3\envs\kingdu\lib\site-packages\tqdm\auto.py:22: TqdmWarning: IProgress not foun
d. Please update jupyter and ipywidgets. See https://ipywidgets.readthedocs.io/en/stable/user_install.html
  from .autonotebook import tqdm as notebook_tqdm
```

In [2]:
```python
import sys
import numpy as np
import matplotlib.pyplot as plt

from mnist_classification.data_loader import load_mnist

from mnist_classification.models.fc_model import FullyConnectedClassifier
from mnist_classification.models.cnn_model import ConvolutionalClassifier
```

In [3]:
```python
model_fn = "./model3.pth"
```

In [4]:
```python
device = torch.device('cuda') if torch.cuda.is_available() else torch.device('cpu')
```

In [5]:
```python
def load(fn, device):
    d = torch.load(fn, map_location=device)

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

In [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)))
```

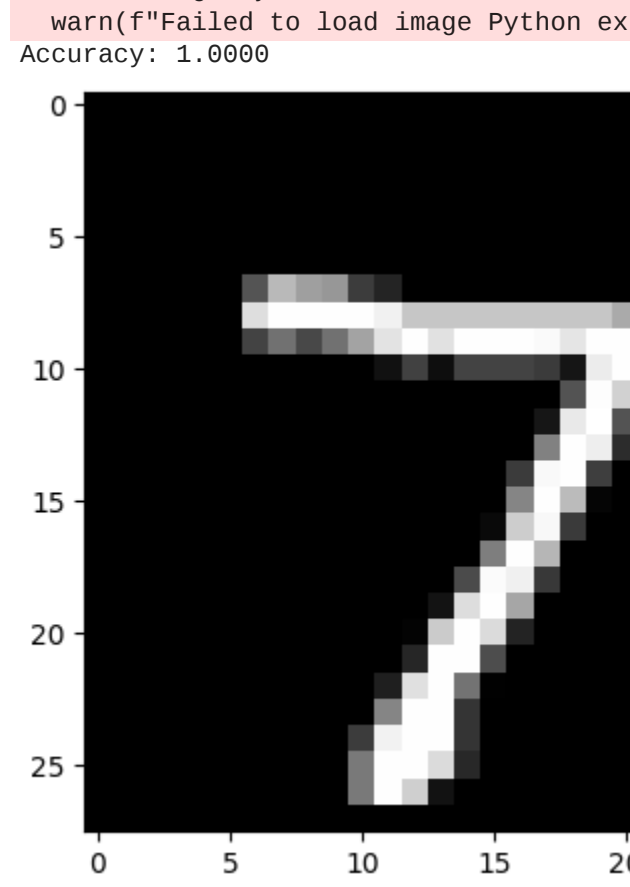
In [7]:
```python
def test(model, x, y, to_be_shown=True):
    model.eval()

    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)
```

In [8]:
```python
from train import get_model

train_config, state_dict = load(model_fn, device)

model = get_model(train_config).to(device)
model.load_state_dict(state_dict)

print(model)
```

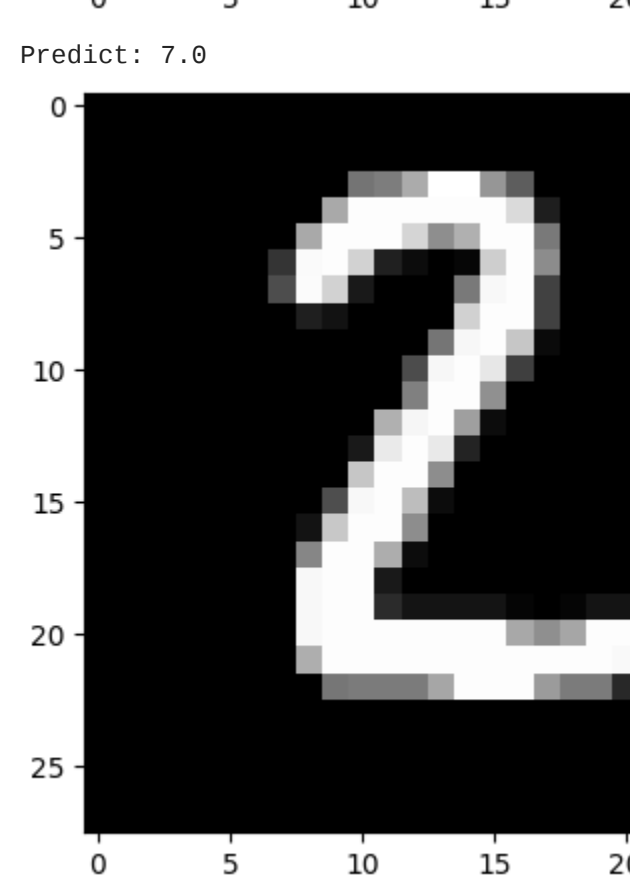
```
ConvolutionalClassifier(
  (blocks): Sequential(
    (0): ConvolutionBlock(
      (layers): Sequential(
        (0): Conv2d(1, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (1): ReLU()
        (2): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (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)
  )
)
```

In [9]:
```python
# Load MNIST test set.
x, y = load_mnist(is_train=False,
                  flatten=True if train_config.model == 'fc' else False)

x, y = x.to(device), y.to(device)

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

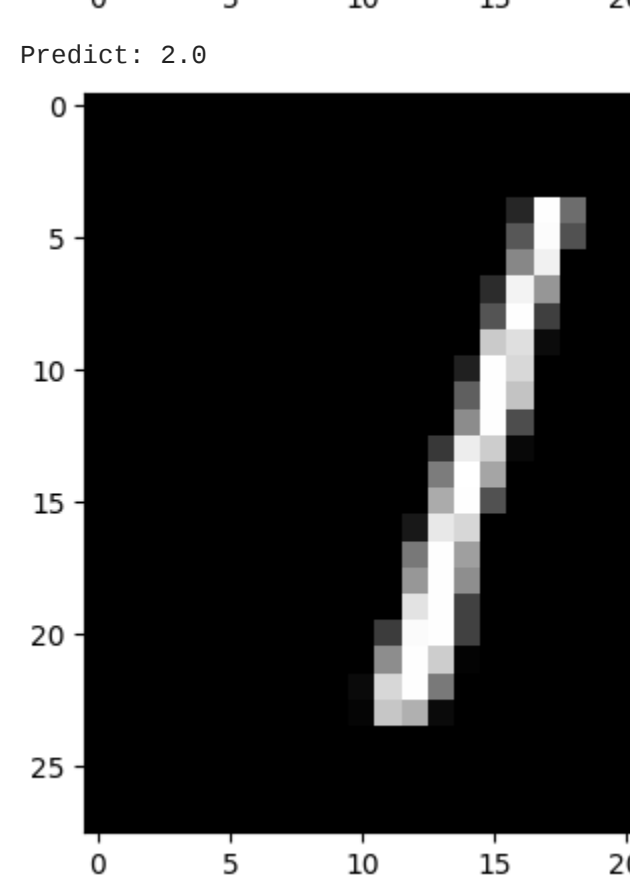
```
c:\Users\Infornet\anaconda3\envs\kingdu\lib\site-packages\torchvision\io\image.py:13: UserWarning: Failed
to load image Python extension:
  warn(f"Failed to load image Python extension: {e}")
Accuracy: 1.0000
```



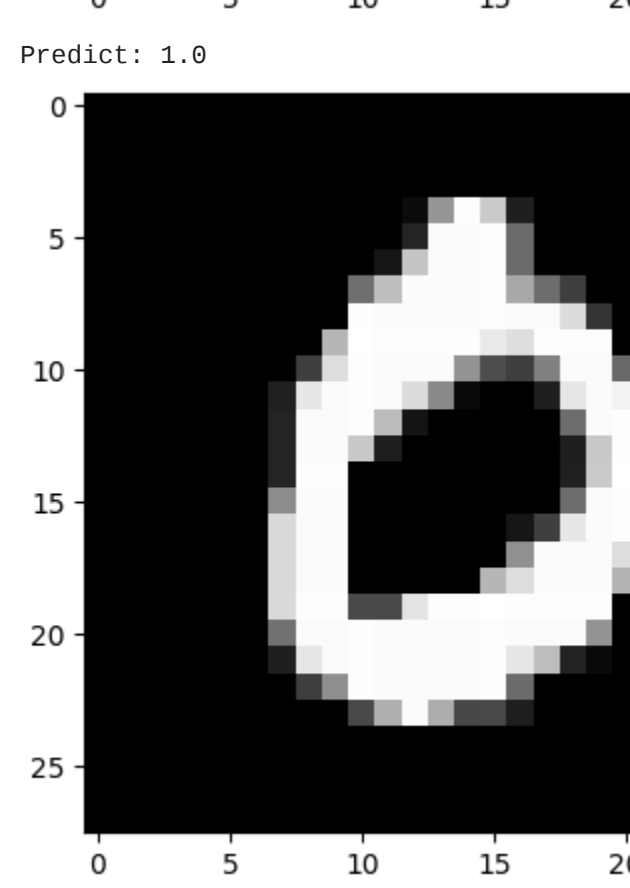
Predict: 7.0



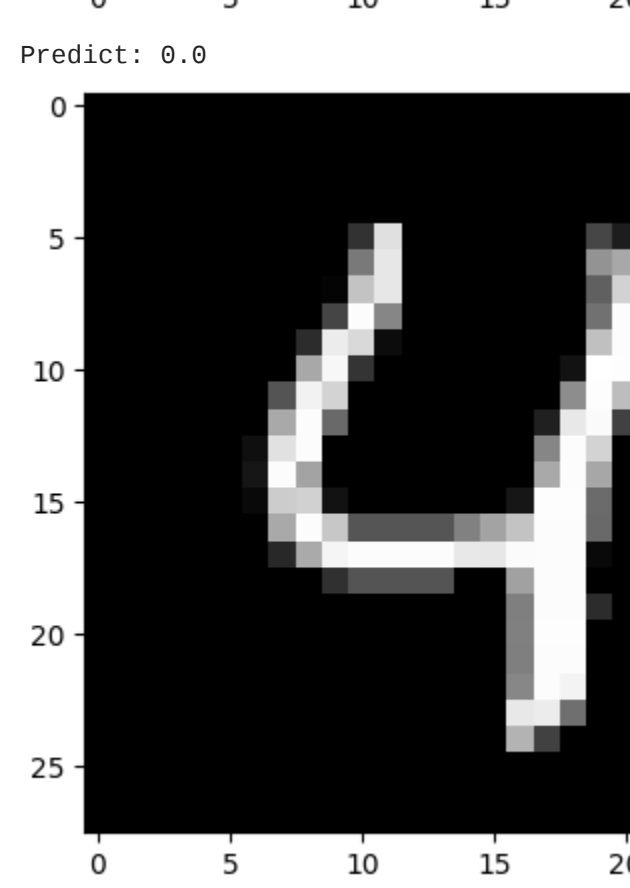
Predict: 2.0



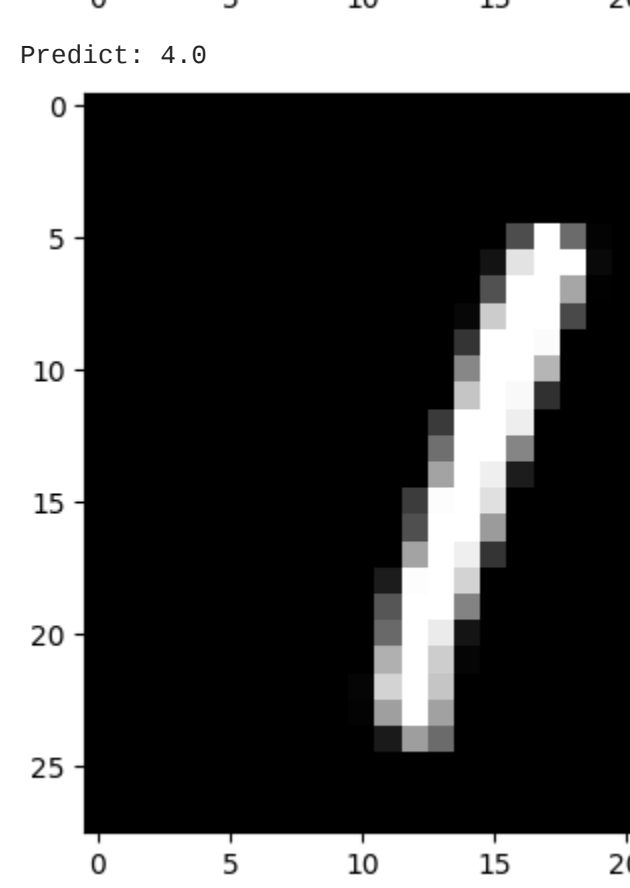
Predict: 1.0



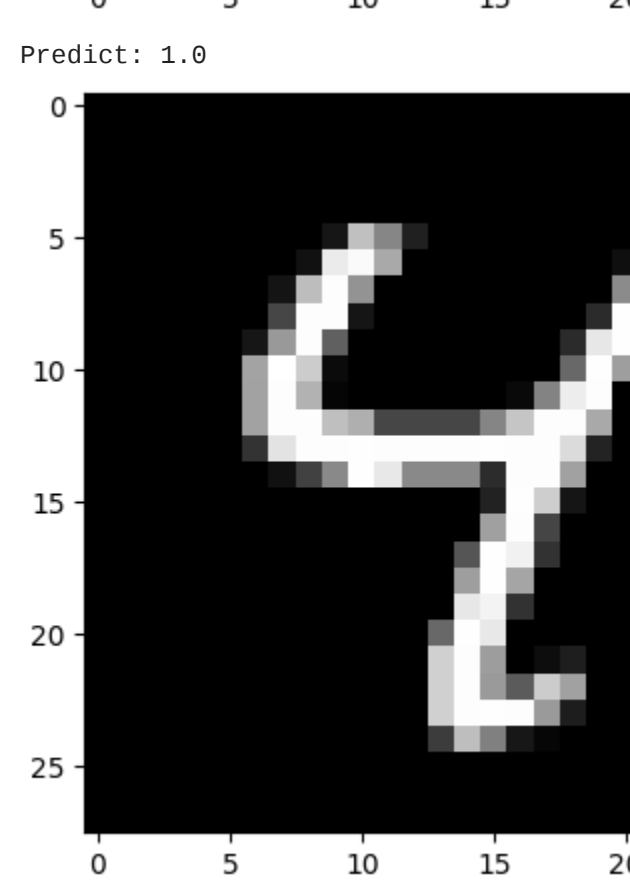
Predict: 0.0



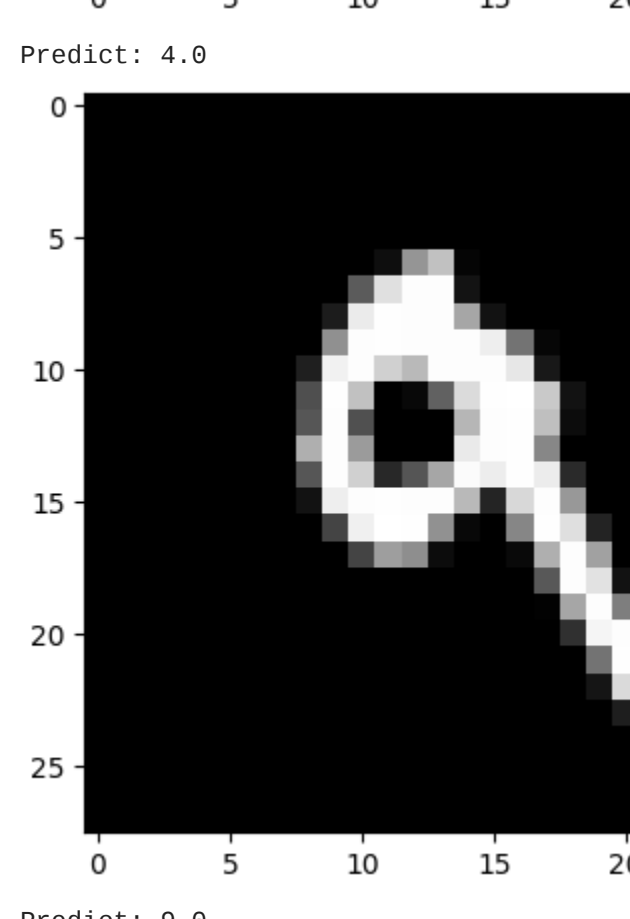
Predict: 4.0



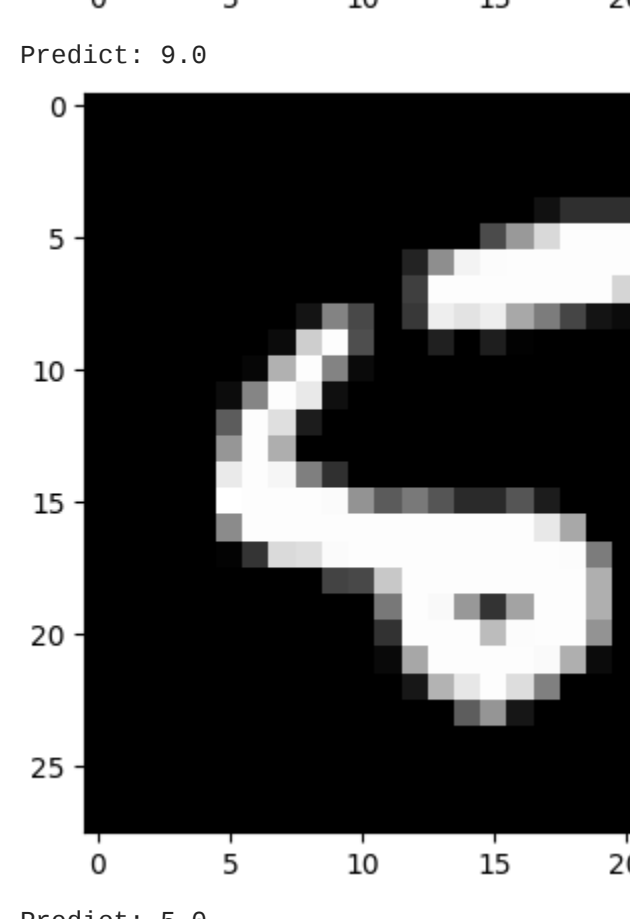
Predict: 1.0



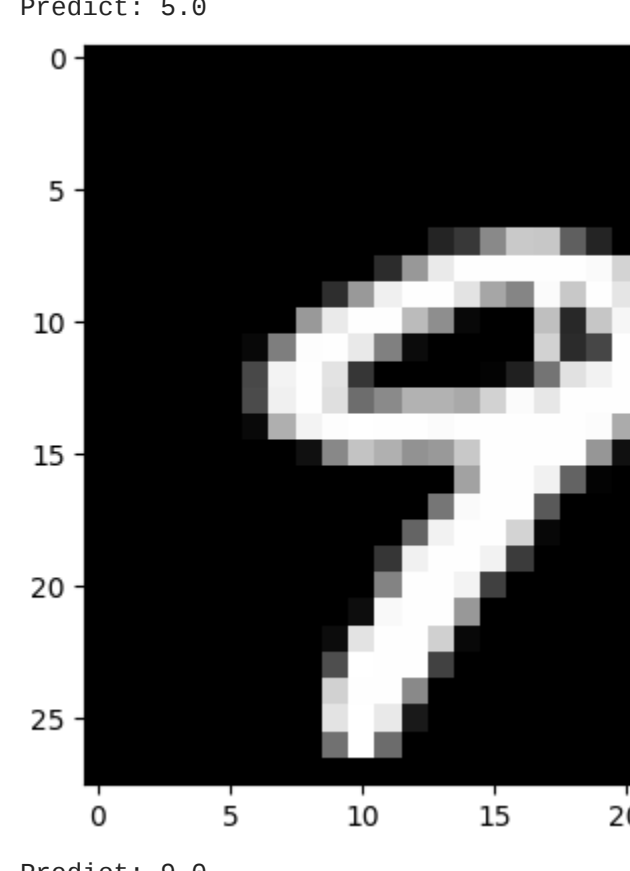
Predict: 4.0



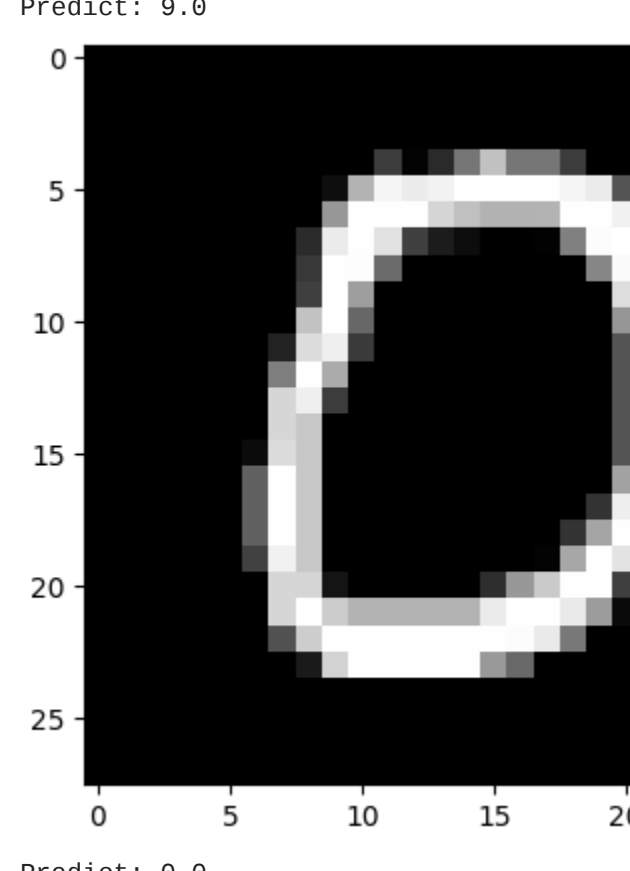
Predict: 9.0



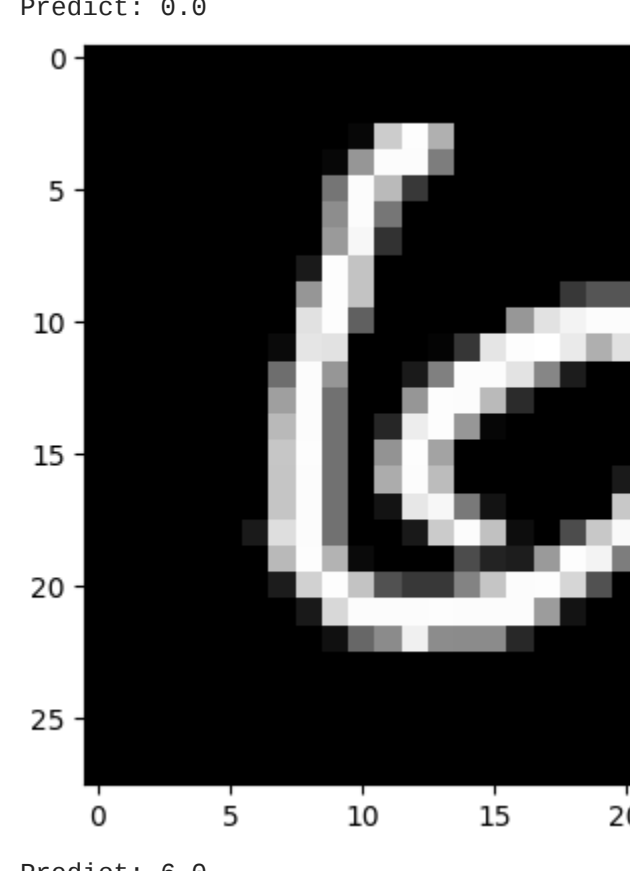
Predict: 5.0



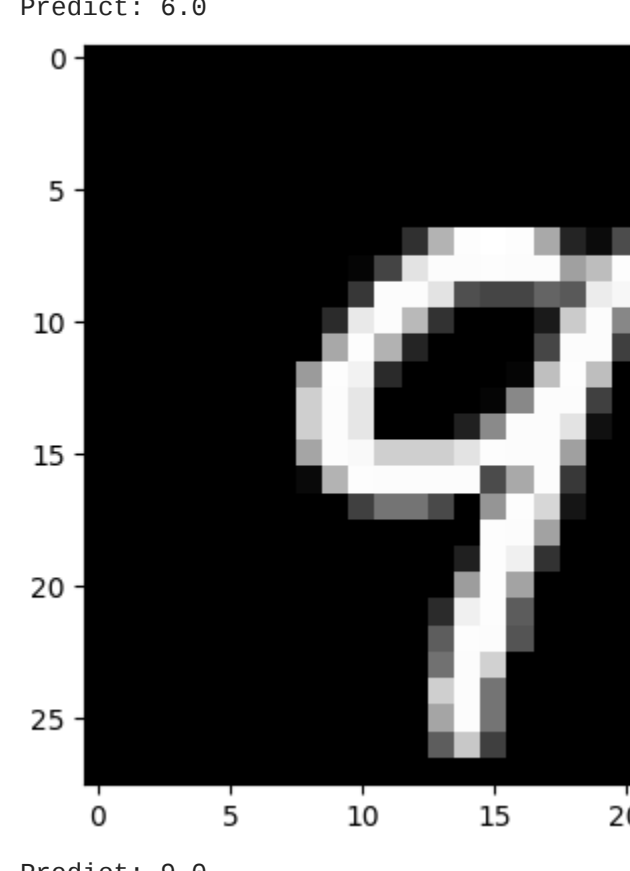
Predict: 9.0



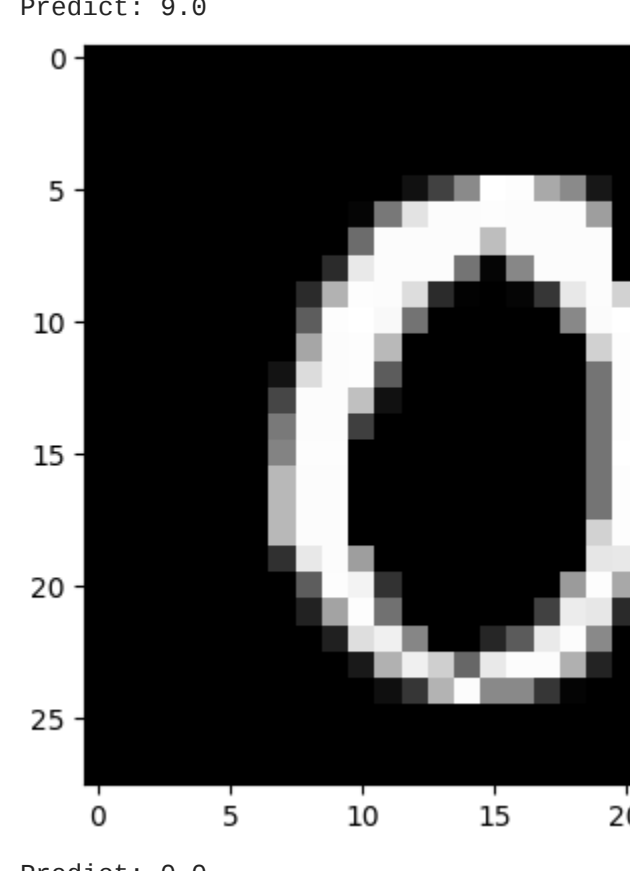
Predict: 0.0



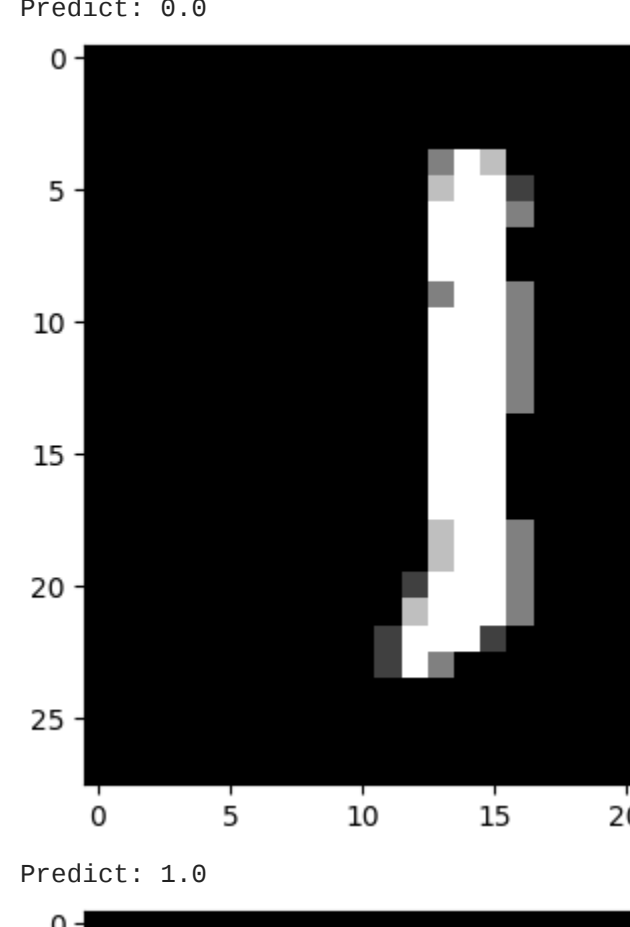
Predict: 6.0



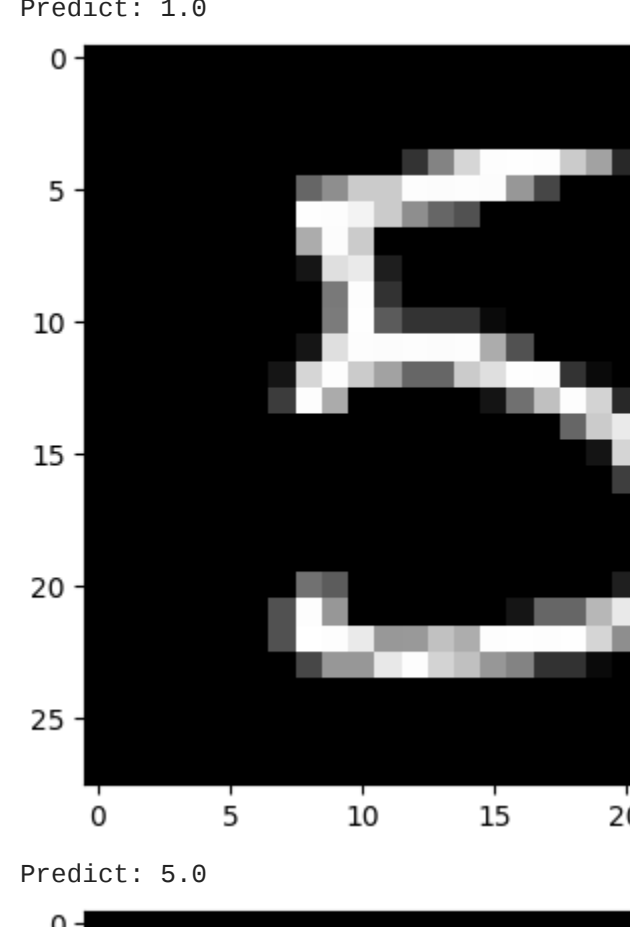
Predict: 9.0



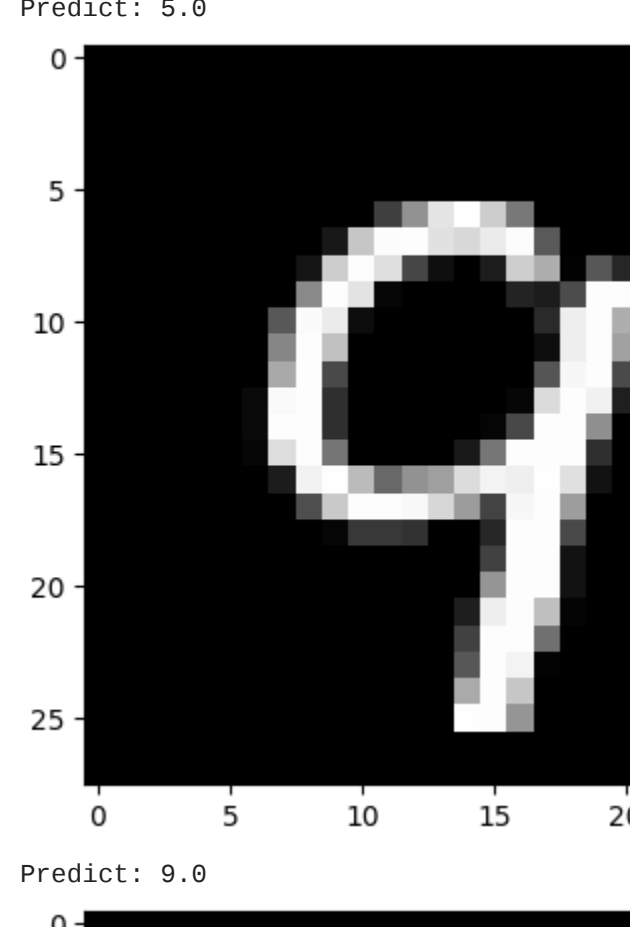
Predict: 0.0



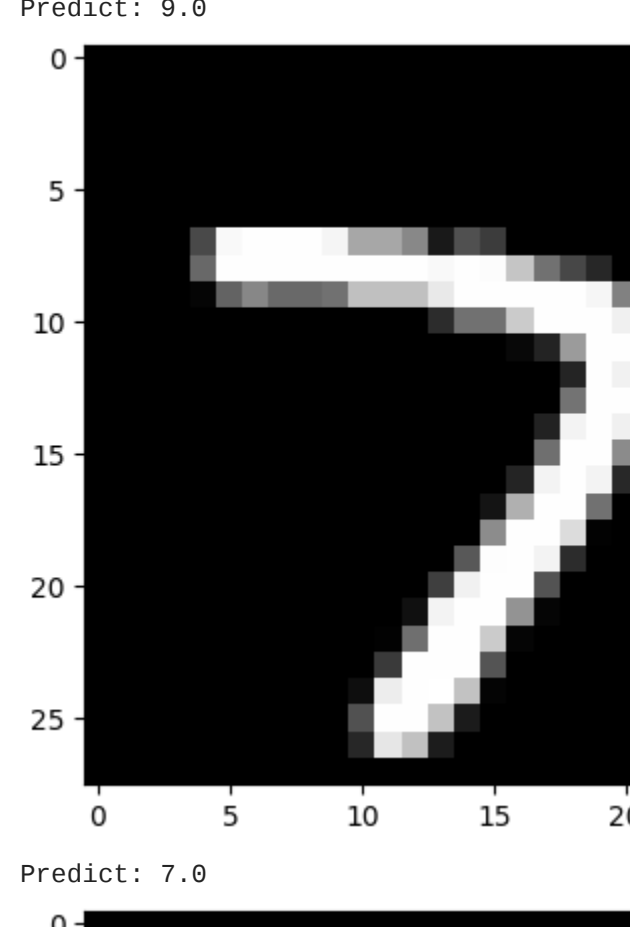
Predict: 1.0



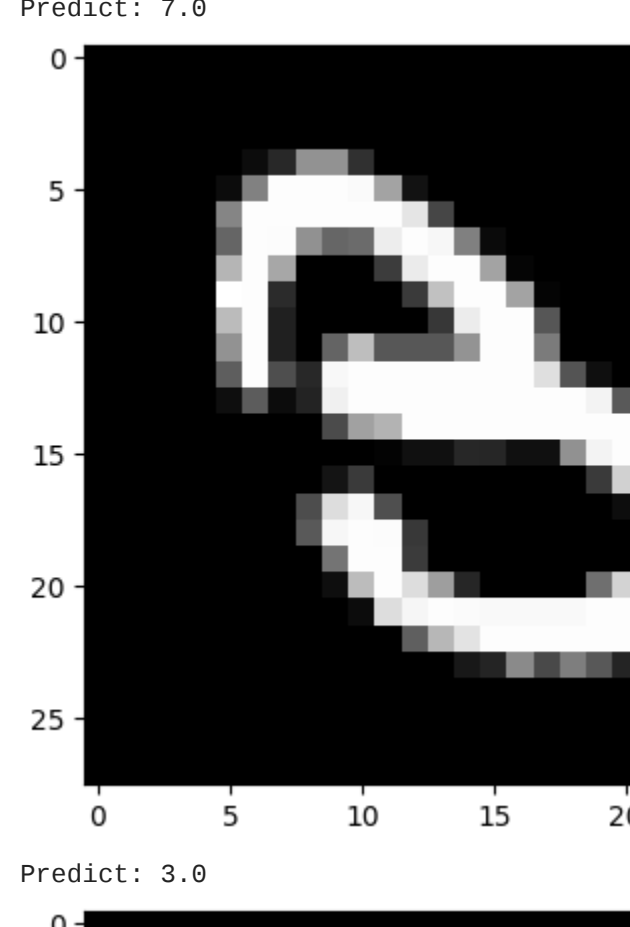
Predict: 5.0



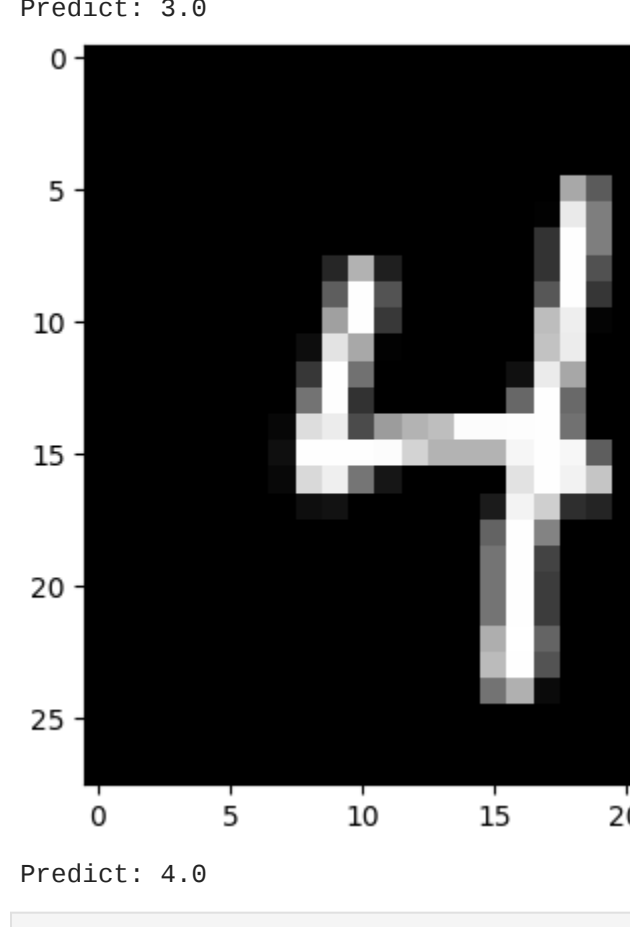
Predict: 9.0



Predict: 7.0



Predict: 3.0



Predict: 4.0

In [ ]: