PA2_EE21S063

September 25, 2022

0.1 MANSI KAKKAR (EE21S063)

1 PROGRAMMING ASSIGNMENT 2 OF DEEP LEARNING FOR IMAGING (EE5179)

1.0.1 Comprises of three sections :

- *MNIST classification using CNN*
- With the architecture as:
  - A convolution layer with 32 3x3 filters with stride 1 and padding 1
  - A 2x2 maxpool layer with stride 2
  - A convolution layer with 32 3x3 filters with stride 1 and padding 1
  - A 2x2 maxpool layer with stride 2
  - A fully connected layer with 500 outputs
  - A fully connected layer with 10 outputs
- Training the network for 8 epochs with learning rate = 0.01
- *Visualising the Convolutional layers*
- Visualising both the convolutional layers along with one fully connected layer as well
- Visualising the output of layers after each convolutional layer
- Visualising Occlusion effects on test images
- *Adversarial Examples*
- Non - Targetted Attack
- Targetted Attack
- Adding Noise

1.1 Importing Libraries

[ ]:
```
import sys
import numpy as np
import os
import matplotlib.pyplot as plt
import torch
from torchvision import datasets
import torchvision.transforms as transforms
from torch.utils.data import Dataset, DataLoader, random_split
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
from torchvision.utils import make_grid

torch.manual_seed(2111)
```

[ ]:
```
<torch._C.Generator at 0x7fe65dc0f030>
```

[ ]:
```
%matplotlib inline
plt.rcParams['figure.figsize'] = (10.0, 10.0) # set default size of plots
```

[ ]:
```
#Setting the device to GPU
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
print(device)
```

```
cpu
```

1.2 Intialisation

[ ]:
```
epochs = 8
learning_rate = 0.01
batch_size = 100
momentum = 0.9
indices = [10, 2, 1, 63, 65, 15, 66, 60, 61, 62]
non_targetted_n = 15000
targetted_n = 5000
non_targetted_step_size = 0.01
beta = 0.001
alpha = 0.1
```

1.3 Loading Dataset

[ ]:
```
transform = transforms.Compose([transforms.ToTensor(), transforms.Normalize((0.
↪1307,), (0.3081,))])
train_dataset = datasets.MNIST(root = "data/", train = True, transform =␣
↪transform, download = True)
test_dataset = datasets.MNIST(root = "data/", train = False, transform =␣
↪transform, download = True)

print(f"number of train samples: {len(train_dataset)}")
print(f"number of test samples: {len(test_dataset)}")

train_loader = DataLoader(train_dataset, batch_size = batch_size, shuffle =␣
↪True)
test_loader = DataLoader(test_dataset, batch_size = batch_size, shuffle = 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
  0%|          | 0/9912422 [00:00<?, ?it/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
  0%|          | 0/28881 [00:00<?, ?it/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
  0%|          | 0/1648877 [00:00<?, ?it/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
  0%|          | 0/4542 [00:00<?, ?it/s]

Extracting data/MNIST/raw/t10k-labels-idx1-ubyte.gz to data/MNIST/raw
```

```
number of train samples: 60000
number of test samples: 10000
```

[ ]:
```
print(f"size of train dataloader is :{len(train_loader)}")
print(f"size of test dataloader is :{len(test_loader)}")
data = next(iter(train_loader))
img, target = data
target
print(f"image shape:{img.shape}")
print(f"Targets shape:{target.shape}")
```

```
size of train dataloader is :600
size of test dataloader is :100
image shape:torch.Size([100, 1, 28, 28])
Targets shape:torch.Size([100])
```

2 MNIST CLASSIFICATION USING CNN

3 CNN Model

- $log(softmax)$ is used here instead of $softmax$
- Forward function returns output with the condition, whether the output needs softmax output or not.

[ ]:
```
class CNN(nn.Module):
  def __init__(self):
    super(CNN, self).__init__()

    self.activ = nn.ReLU()

    self.conv1 = nn.Conv2d(1, 32, kernel_size = 3, stride = 1, padding = 1)
    self.layer1 = nn.Sequential(self.conv1, self.activ, nn.
↪MaxPool2d(kernel_size = 2, stride = 2))

    self.conv2 = nn.Conv2d(32, 32, kernel_size = 3, stride = 1, padding = 1)
    self.layer2 = nn.Sequential(self.conv2, self.activ, nn.
↪MaxPool2d(kernel_size = 2, stride = 2))

    self.fc1 = nn.Linear(32*7*7, 500)
    self.layer3 = nn.Sequential(self.fc1, self.activ)

    self.layer4 = nn.Linear(500, 10)

  # def pool(self, x, kernel = 2, stride = 2):
    # out = F.max_pool2d(x, kernel_size = kernel, stride = stride)
```

```
    # return out

  def forward(self, x, softmax = True):
    out = self.layer1(x.float())
    out = self.layer2(out)

    out = out.view(out.size(0),-1)
    out = self.layer3(out)
    out = self.layer4(out)
    if softmax:
      return F.log_softmax(out, dim = 1)
    else:
      return out
```

4 Training

- Used nll loss here
- Returns training loss

[ ]:
```
def train_model(model, data_loader, optimizer, epoch, device, out_dir = "out/"):
  model.train()
  train_loss = []
  print('For Train Dataset:\n')
  for index, (image, target) in enumerate(data_loader):
    image, target = image.to(device), target.to(device)
    optimizer.zero_grad()
    out = model(image)
    loss = F.nll_loss(out, target)
    loss.backward()
    optimizer.step()
  train_loss.append(loss.item())
  print('Epoch: {}[{}/{} ({}%)], Train Loss:{:.06f}\n'.format(epoch,␣
↪index*len(image), len(data_loader.dataset), 100.0*index/len(data_loader),␣
↪loss.item()))

  return train_loss
```

5 Testing

- Returns predicted values, test/validation loss and accuracy

[ ]:
```
def test_model(model, data_loader, epoch, device, out_dir="out/"):
  #Testing
    model.eval()
```

```
    correct = 0
    loss = 0
    prediction = []
    print('For Test Dataset:\n')
    with torch.no_grad():
      for index, (image, target) in enumerate(data_loader):
        image = image.to(device)
        target = target.to(device)
        out = model(image)
        loss+= F.nll_loss(out, target, reduction = "sum").item()
        predicted = out.argmax(dim=1, keepdim = True)
        correct += predicted.eq(target.view_as(predicted)).sum().item()
        prediction.append(predicted)
    loss/= len(test_loader.dataset)
    acc = correct/(len(test_loader.dataset))
    print('Epoch:{} [{}/{} ({}%)] , Test Loss:{:.4f}, Test Accuracy:{}%\n'.
↪format(epoch,index*len(image), len(data_loader.dataset), 100.0*index/
↪len(data_loader), loss, 100.0*acc))


    return prediction, loss, acc
```

6 Main Function

- Stochastic Gradient Descent is used as the optimiser
- Validation losses, Accuracies are stored
- Graph for Train and Validation loss, and Accuracies has been plotted in the later stages

[ ]:
```
train_losses = []
validation_losses = []
validation_accuracies = []
predicted = []
model = CNN().to(device)
optimizer = torch.optim.SGD(model.parameters(), lr = learning_rate)
for epoch in range(epochs):
  #calling Train Model
  train_loss = train_model(model, train_loader, optimizer, epoch, device)
  #calling Test Model
  predicted, validation_loss, validation_acc = test_model(model, test_loader,␣
↪epoch, device)

  train_losses.append(train_loss)
  validation_losses.append(validation_loss)
  validation_accuracies.append(validation_acc)
```

```
For Train Dataset:
```

```
Epoch: 0[59900/60000 (99.83333333333333%)], Train Loss:0.253941

For Test Dataset:

Epoch:0 [9900/10000 (99.0%)] , Test Loss:0.2527, Test Accuracy:92.74%

For Train Dataset:

Epoch: 1[59900/60000 (99.83333333333333%)], Train Loss:0.105856

For Test Dataset:

Epoch:1 [9900/10000 (99.0%)] , Test Loss:0.1518, Test Accuracy:95.57%

For Train Dataset:

Epoch: 2[59900/60000 (99.83333333333333%)], Train Loss:0.089902

For Test Dataset:

Epoch:2 [9900/10000 (99.0%)] , Test Loss:0.1136, Test Accuracy:96.75%

For Train Dataset:

Epoch: 3[59900/60000 (99.83333333333333%)], Train Loss:0.071108

For Test Dataset:

Epoch:3 [9900/10000 (99.0%)] , Test Loss:0.0818, Test Accuracy:97.56%

For Train Dataset:

Epoch: 4[59900/60000 (99.83333333333333%)], Train Loss:0.070827

For Test Dataset:

Epoch:4 [9900/10000 (99.0%)] , Test Loss:0.0738, Test
Accuracy:97.74000000000001%

For Train Dataset:

Epoch: 5[59900/60000 (99.83333333333333%)], Train Loss:0.066107

For Test Dataset:

Epoch:5 [9900/10000 (99.0%)] , Test Loss:0.0670, Test Accuracy:97.84%
```

```
For Train Dataset:

Epoch: 6[59900/60000 (99.83333333333333%)], Train Loss:0.042361

For Test Dataset:

Epoch:6 [9900/10000 (99.0%)] , Test Loss:0.0554, Test Accuracy:98.13%

For Train Dataset:

Epoch: 7[59900/60000 (99.83333333333333%)], Train Loss:0.059322

For Test Dataset:

Epoch:7 [9900/10000 (99.0%)] , Test Loss:0.0485, Test
Accuracy:98.35000000000001%
```

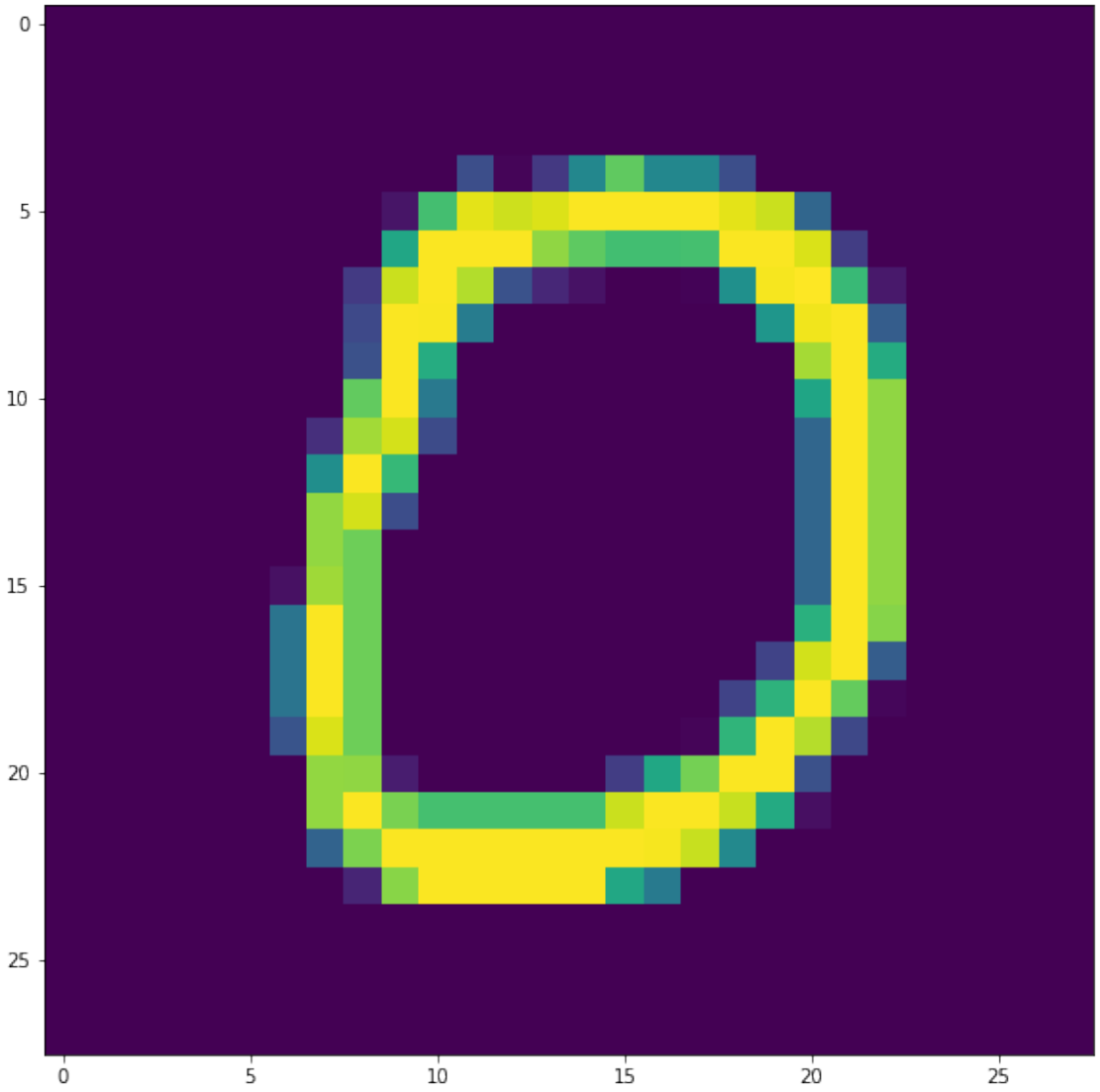
[ ]:
```
image, correct = next(iter(test_loader))
```

7 Visualising the outputs

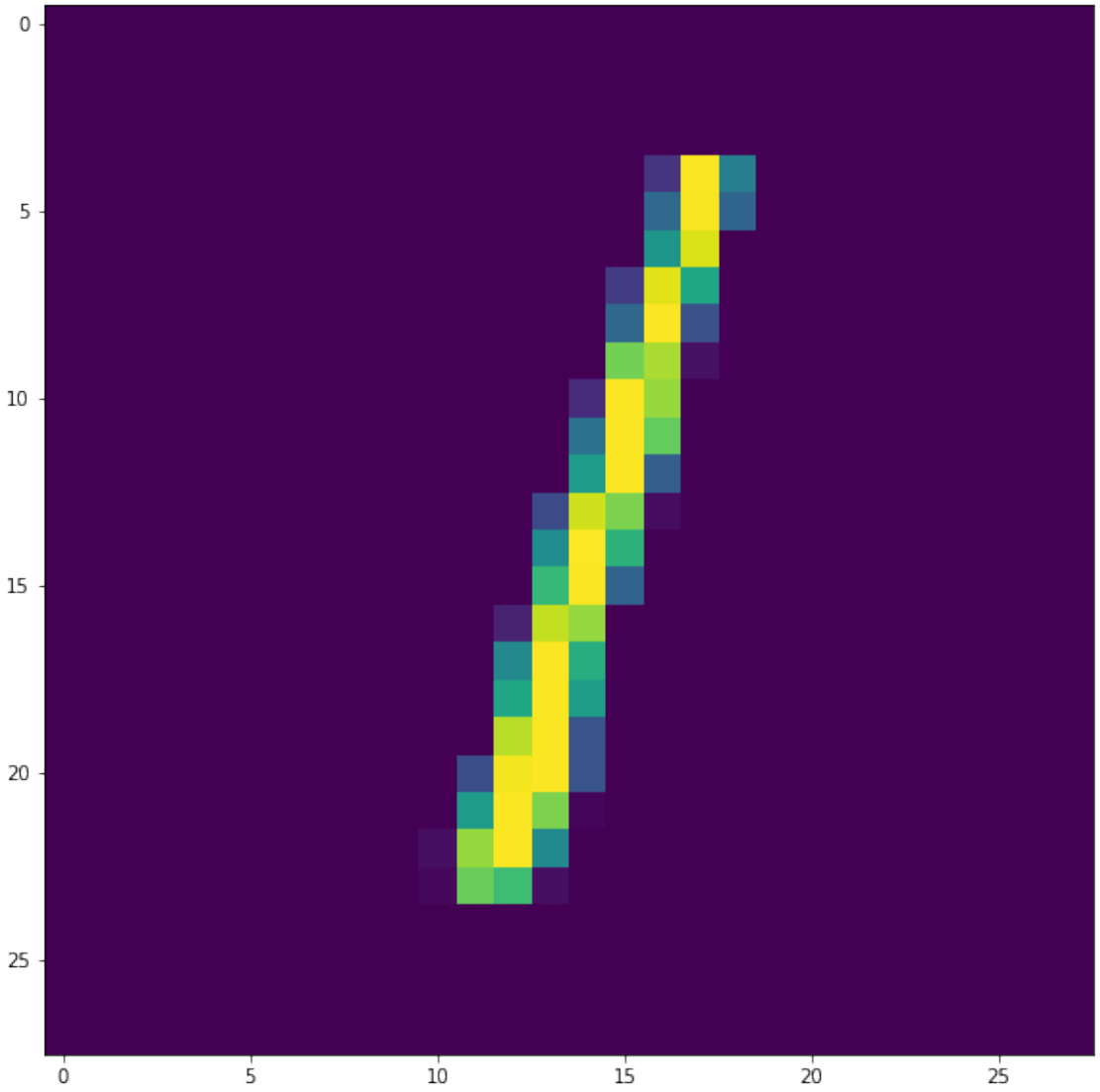
Correct class labels along with the predicted class labels corresponding to the images of all the numbers from 0 to 9 have been plotted to show the results

[ ]:
```
print(image.shape)
print(correct.shape)
print(len(predicted))
for i in range(10):
  print("Index is: {} \t Correct: {} \t Predicted: {}\n".format(i,␣
 ↪correct[indices[i]], predicted[0][indices[i]]))
  plt.imshow(image[indices[i]].reshape(28,28).cpu())
  plt.show()
```

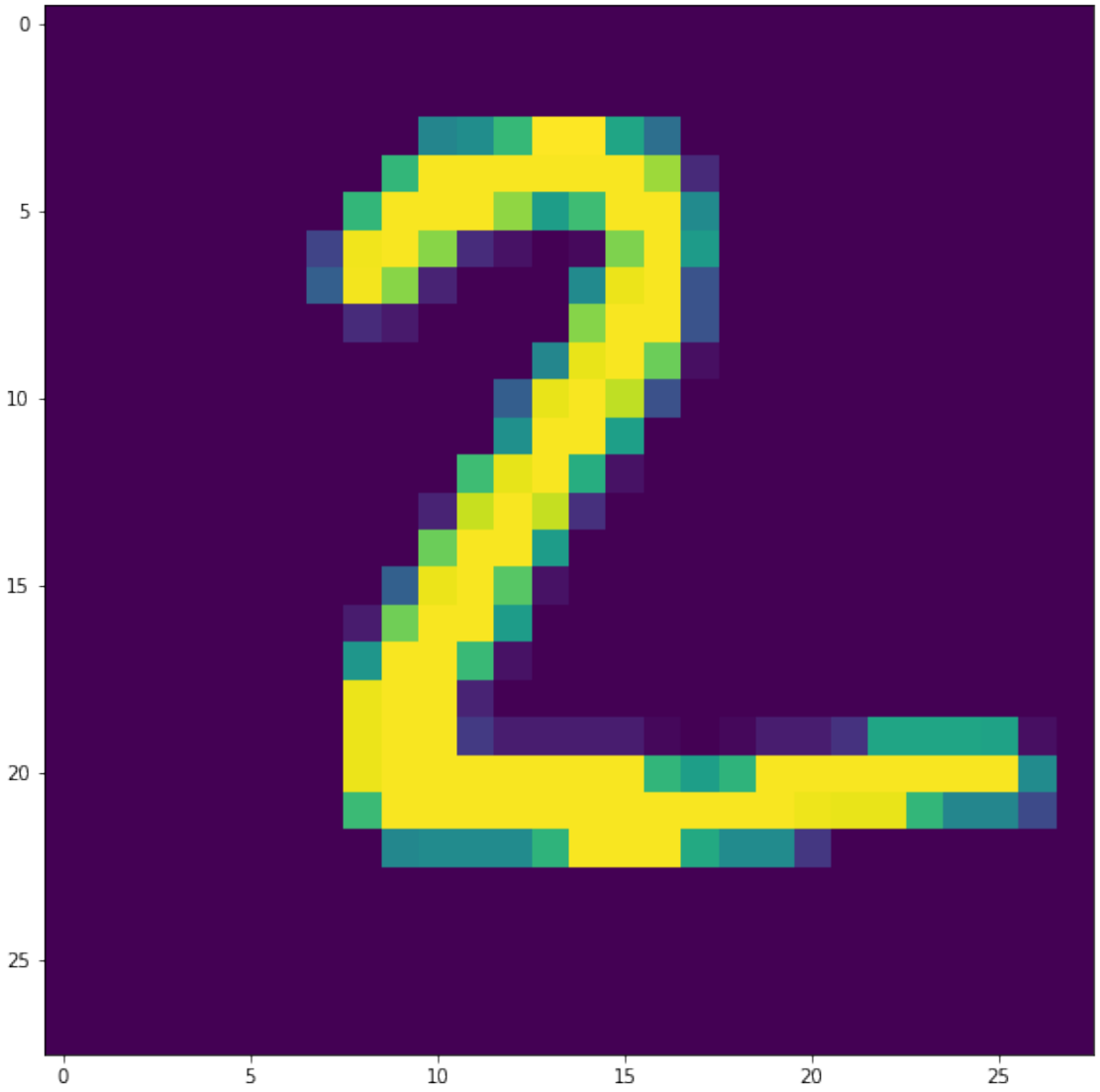
```
torch.Size([100, 1, 28, 28])
torch.Size([100])
100
Index is: 0      Correct: 0      Predicted: tensor([0])
```



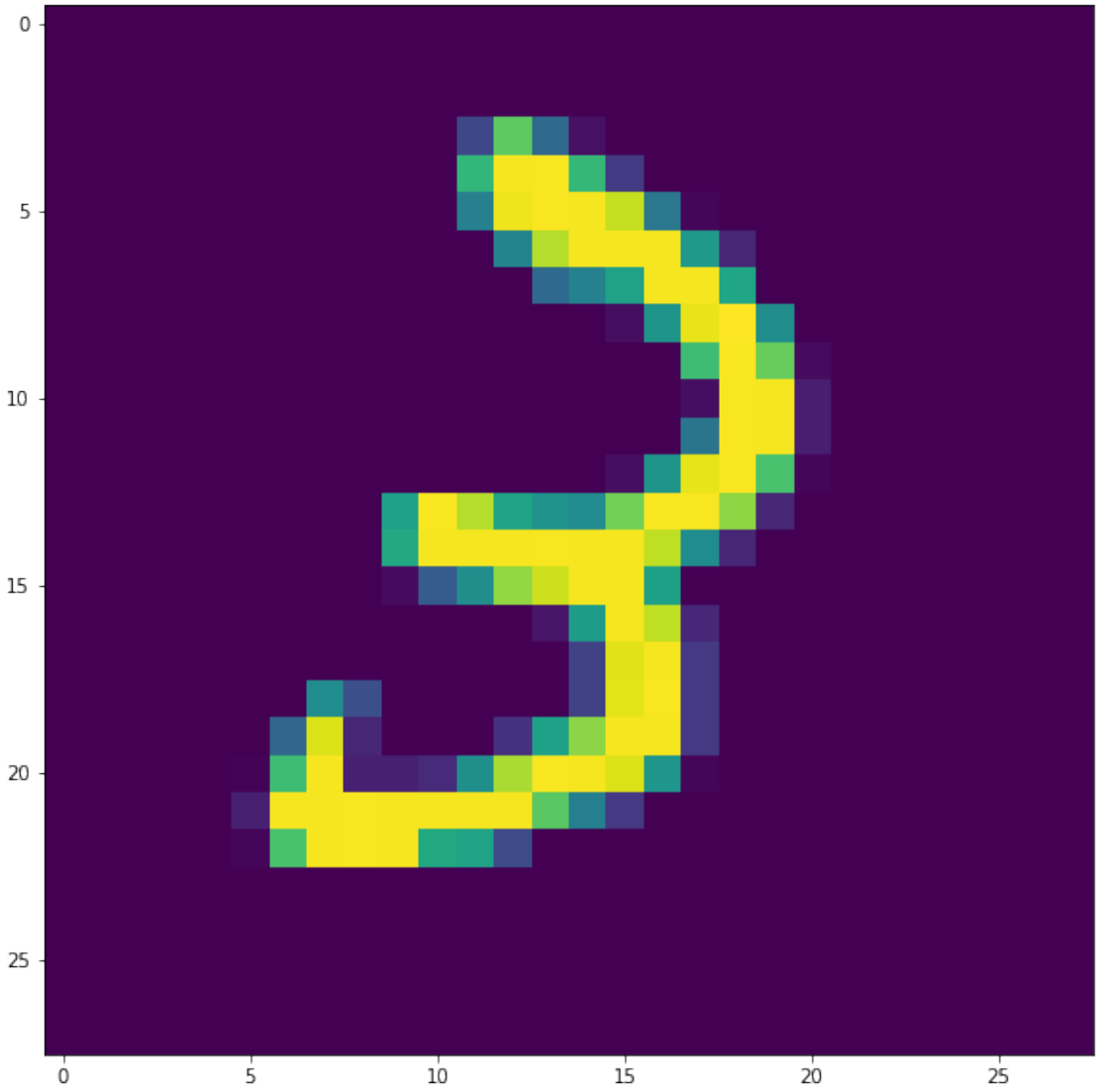

```
Index is: 1      Correct: 1      Predicted: tensor([1])
```



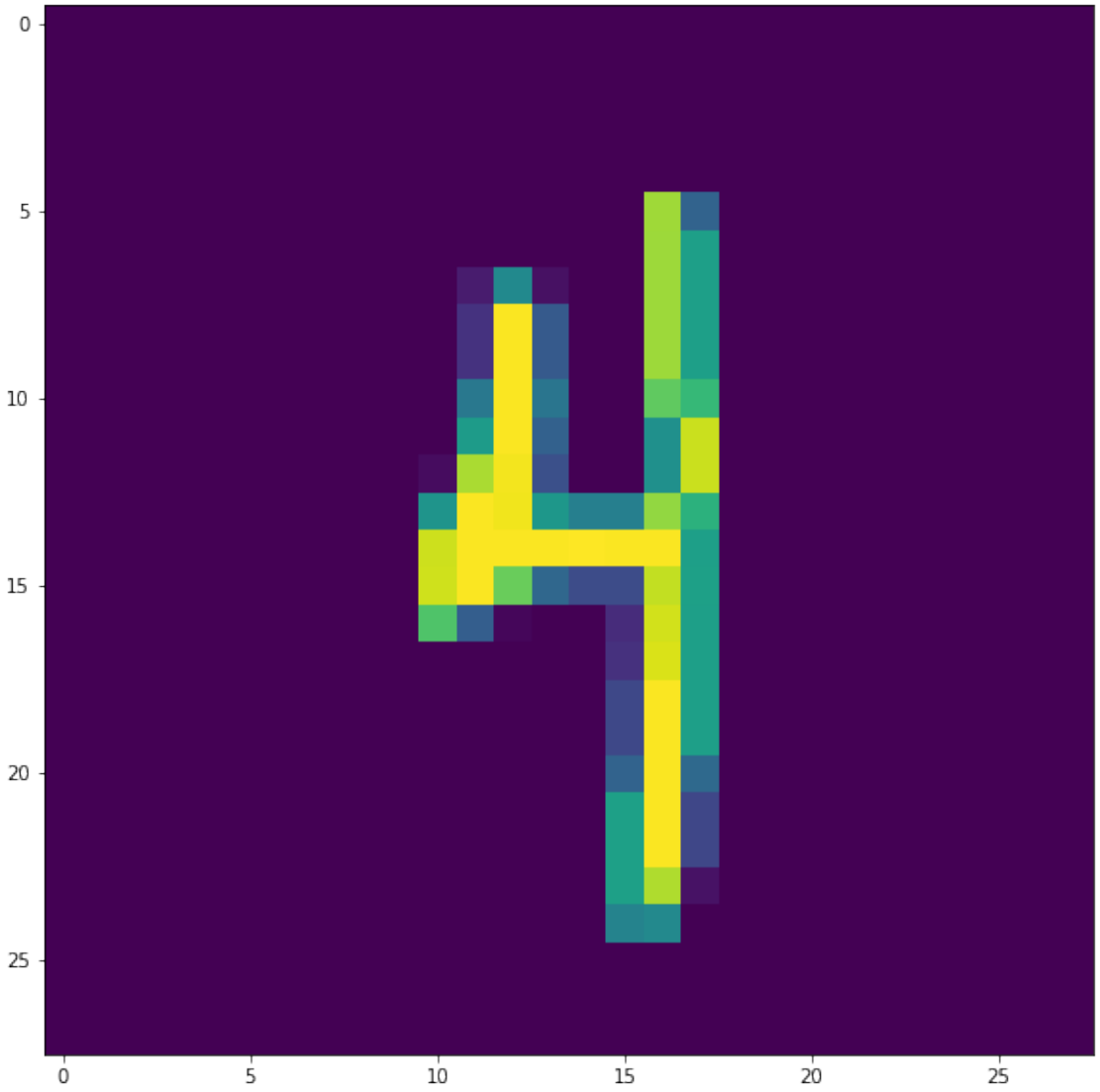

```
Index is: 2      Correct: 2      Predicted: tensor([2])
```



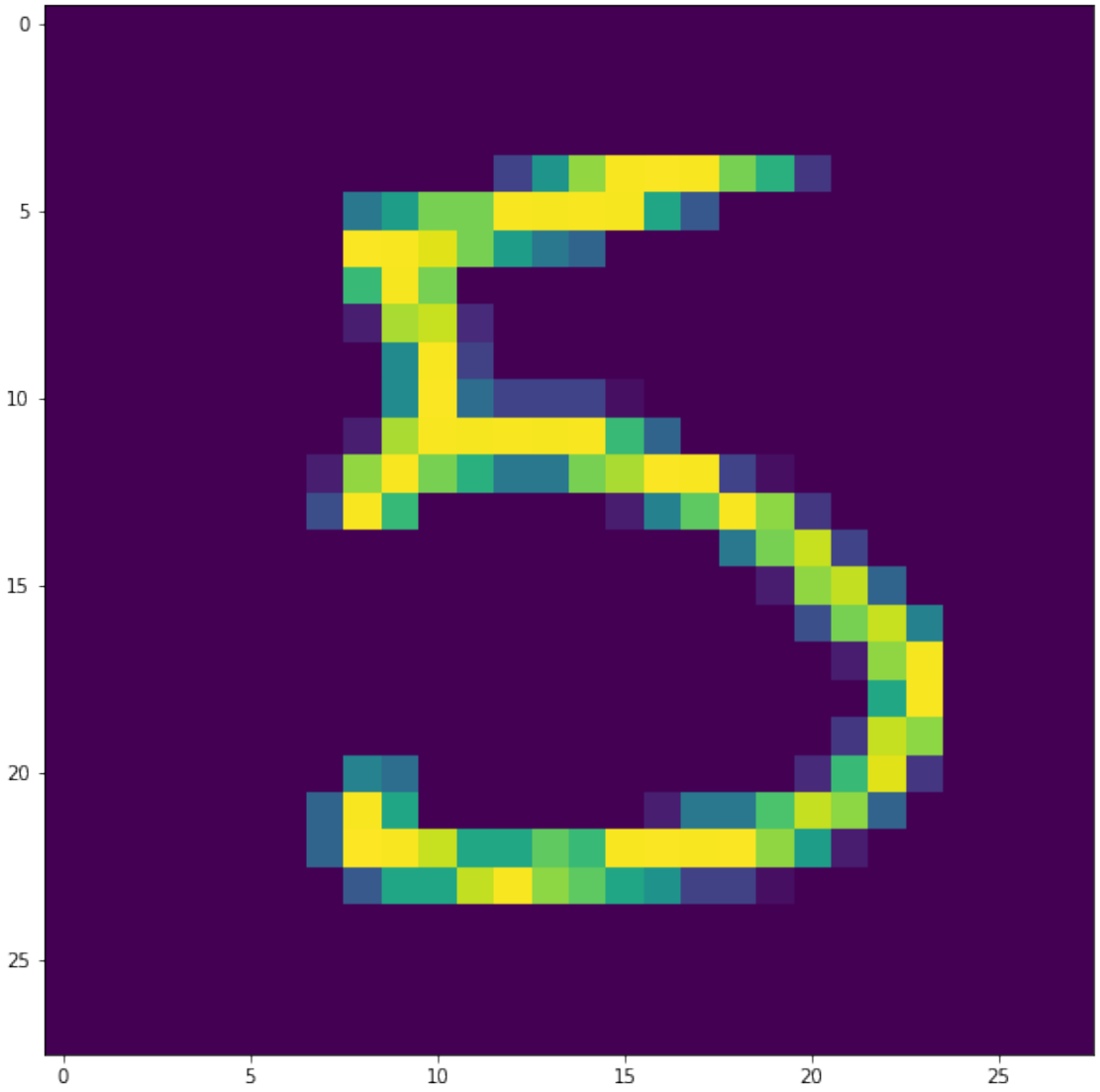

```
Index is: 3      Correct: 3      Predicted: tensor([3])
```



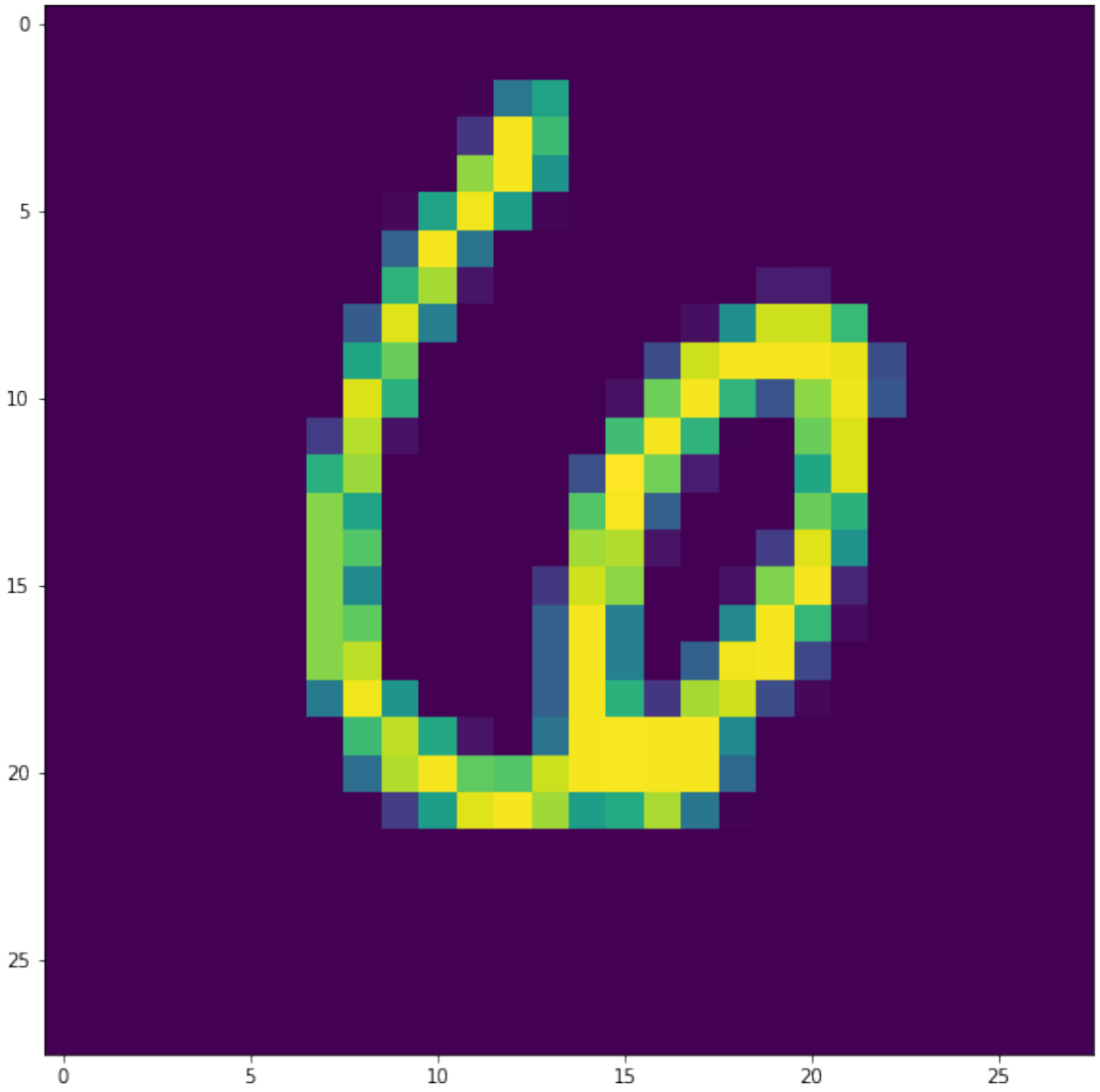

```
Index is: 4      Correct: 4      Predicted: tensor([4])
```



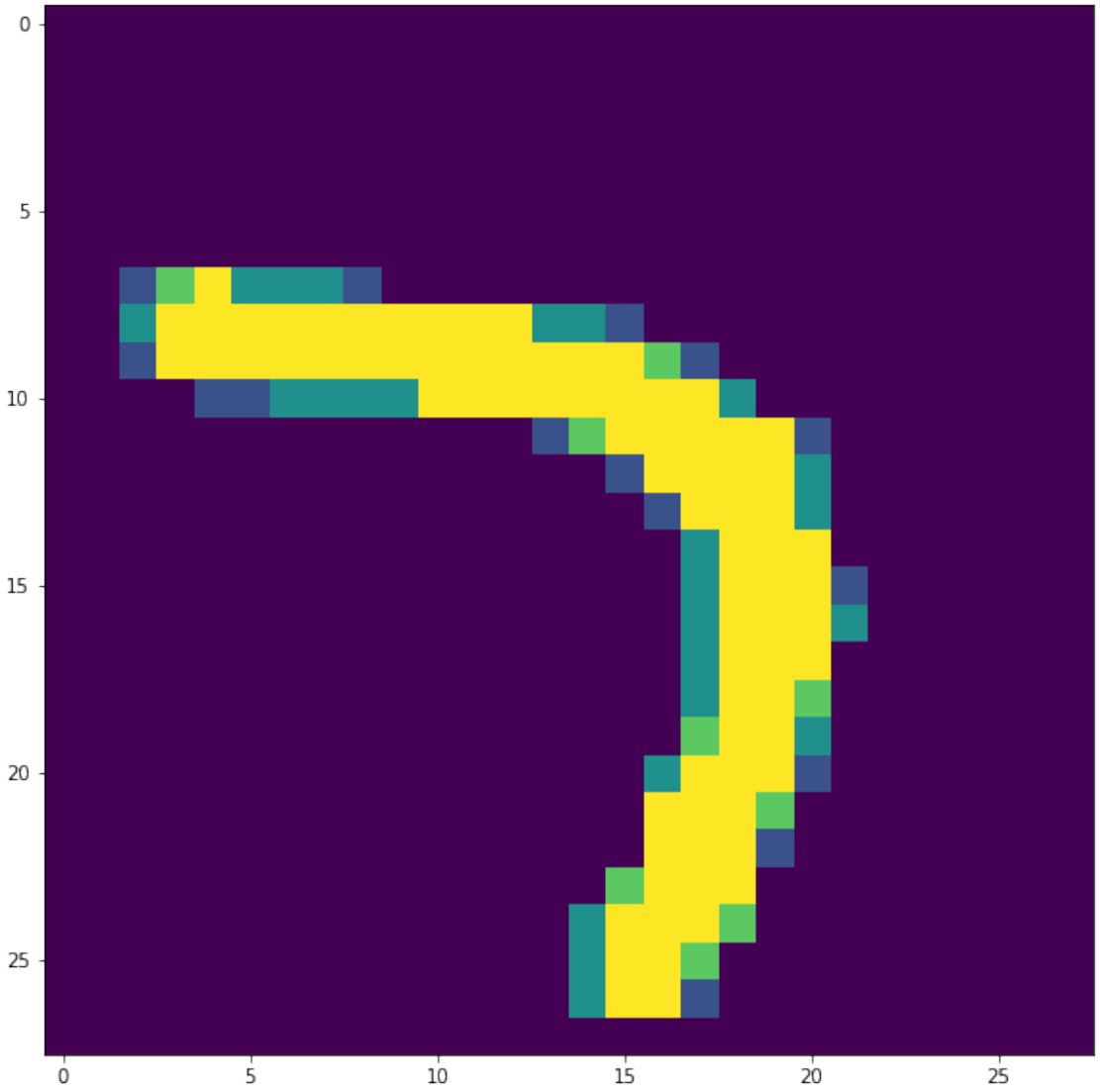

```
Index is: 5      Correct: 5      Predicted: tensor([5])
```



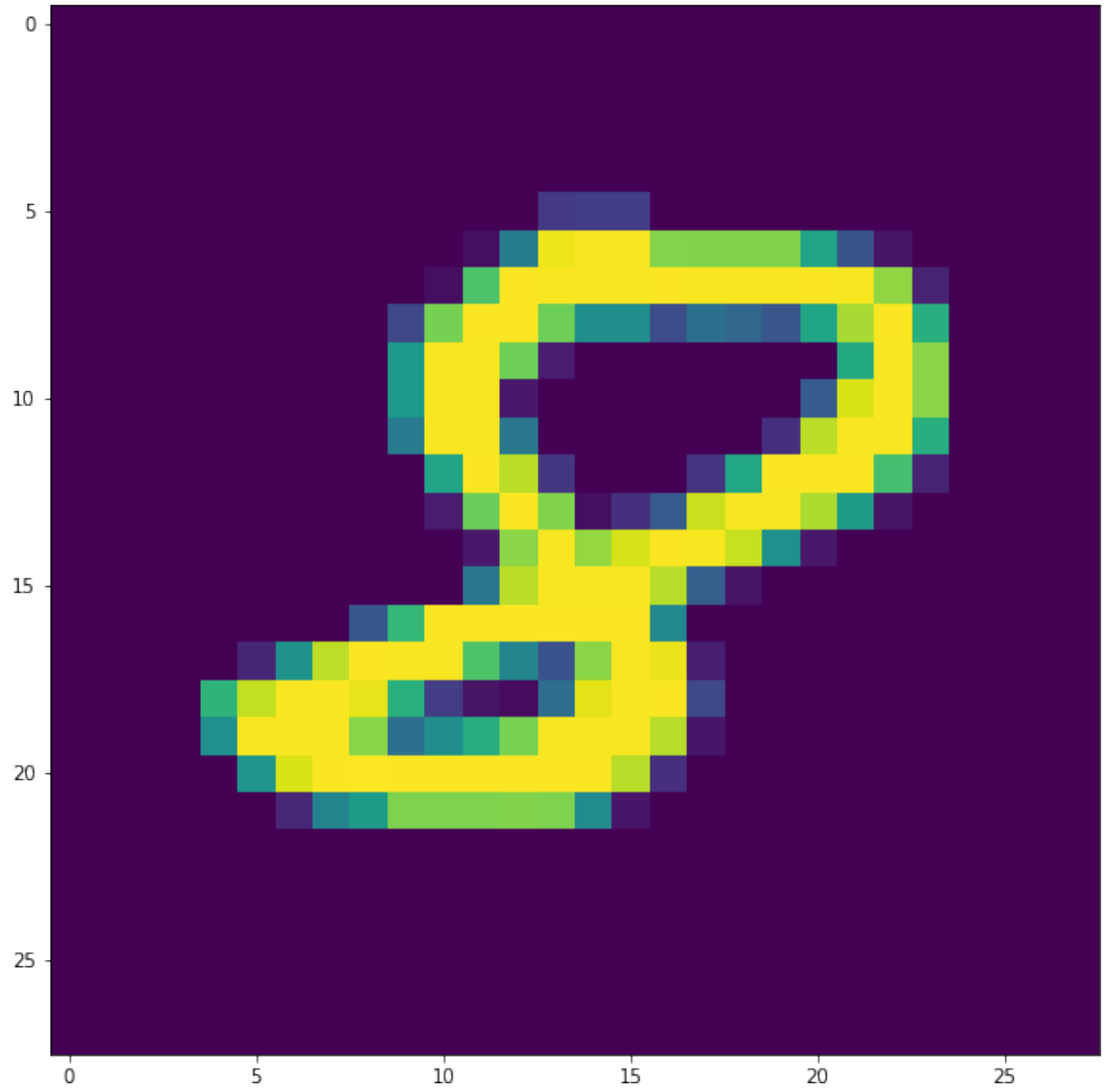

```
Index is: 6      Correct: 6      Predicted: tensor([6])
```



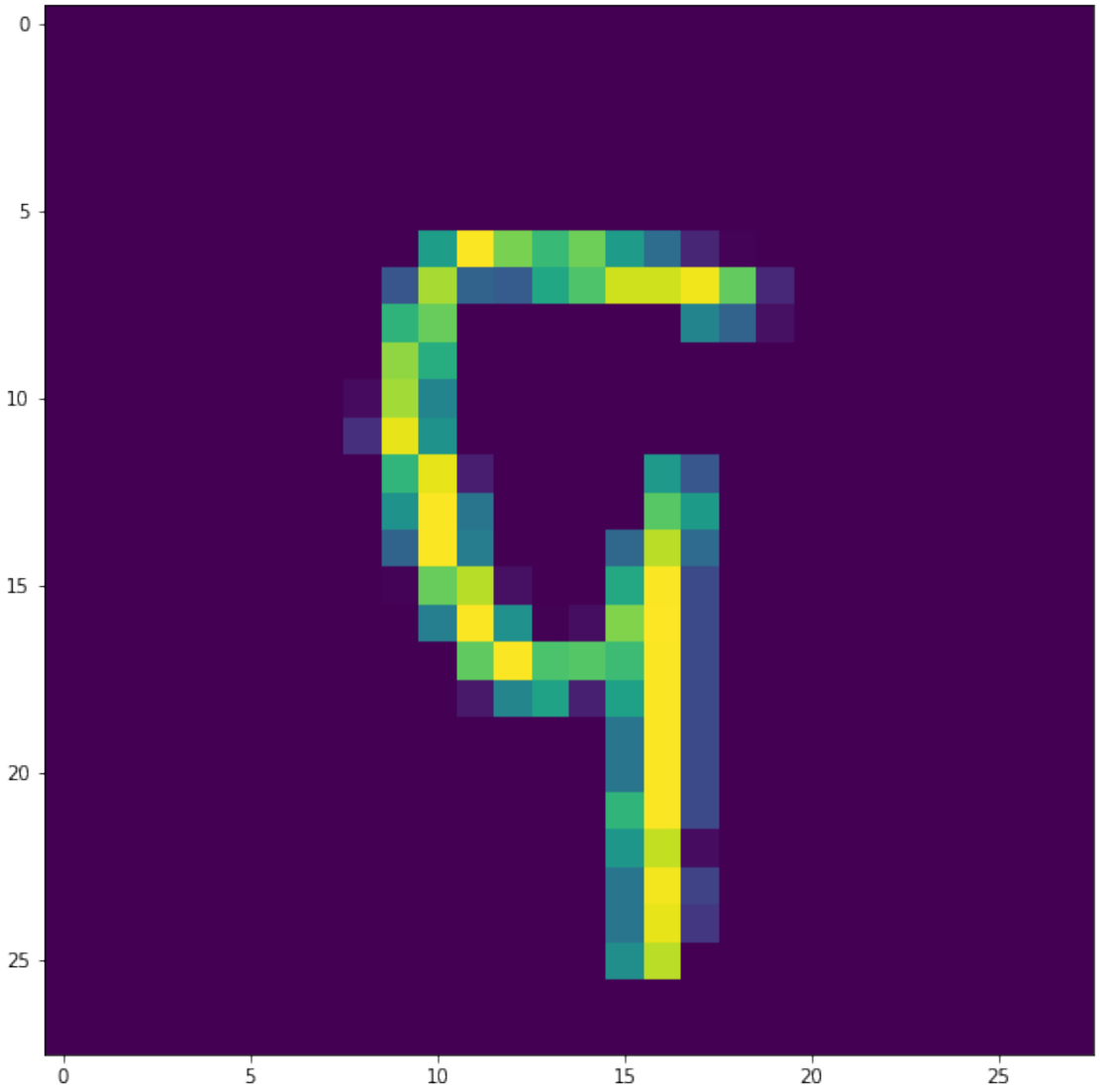

```
Index is: 7      Correct: 7      Predicted: tensor([7])
```




```
Index is: 8      Correct: 8      Predicted: tensor([8])
```



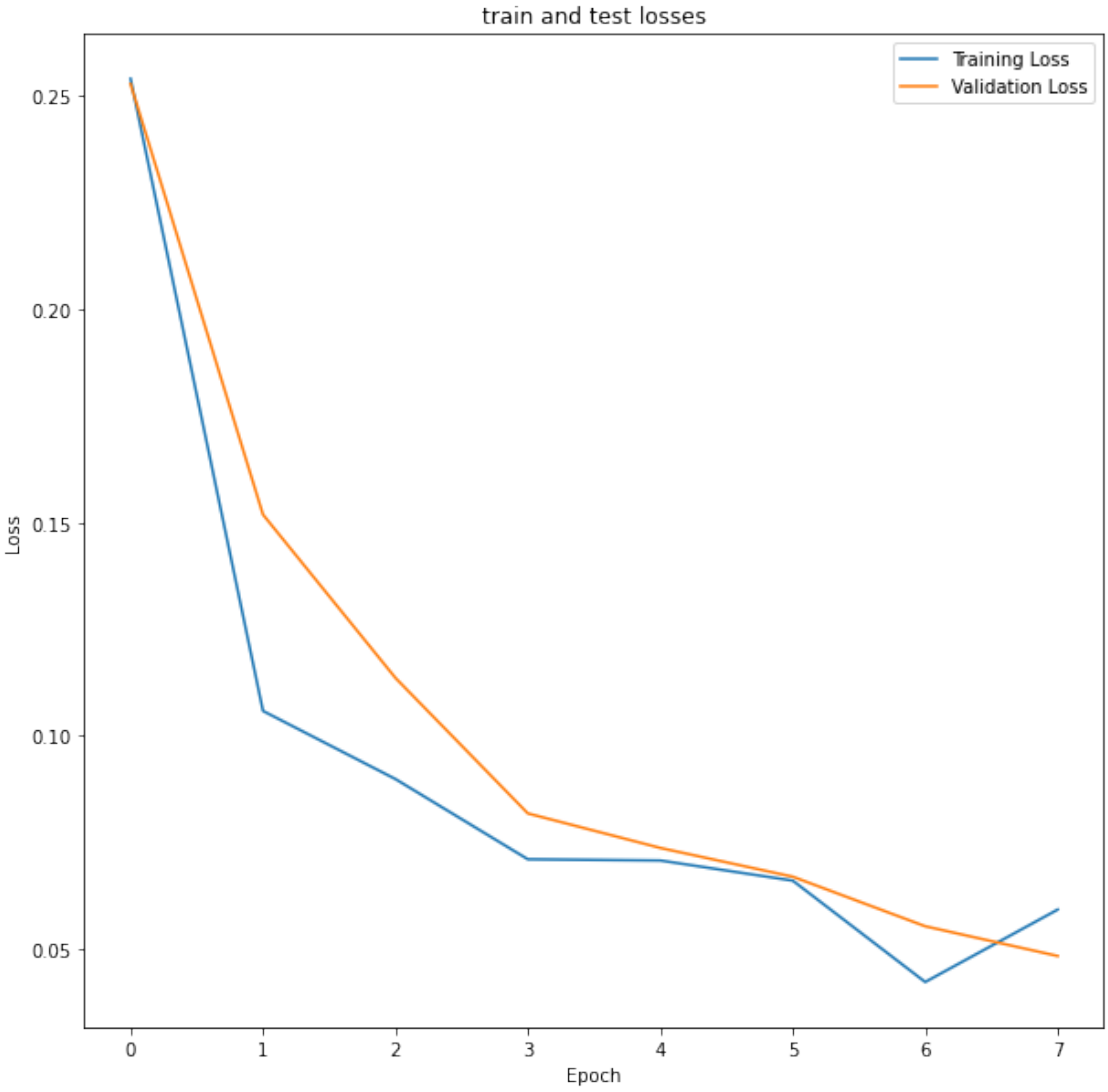

```
Index is: 9      Correct: 9      Predicted: tensor([9])
```

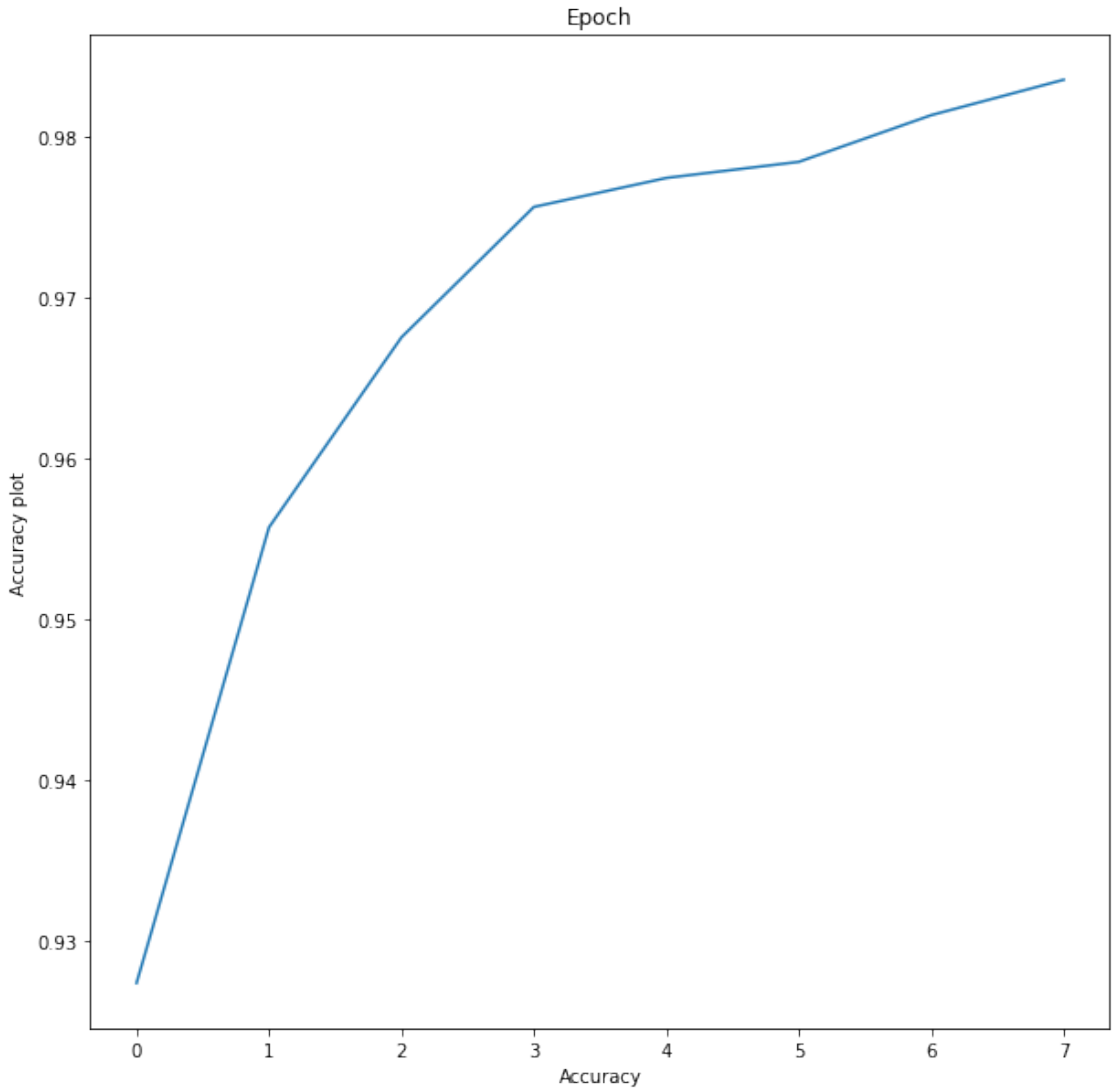


8 Plot Loss Graphs and Accuracy

[ ]:
```
#plot
plt.plot(np.asfarray(train_losses)[::int(len(train_losses)/epochs)])
plt.plot(np.asfarray(validation_losses))
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title('train and test losses')
plt.legend(['Training Loss', 'Validation Loss'])
plt.show()
```

```
#plot Accuracies
plt.plot(np.asfarray(validation_accuracies))
plt.xlabel('Accuracy')
plt.ylabel('Accuracy plot')
plt.title('Epoch')
plt.show()
```





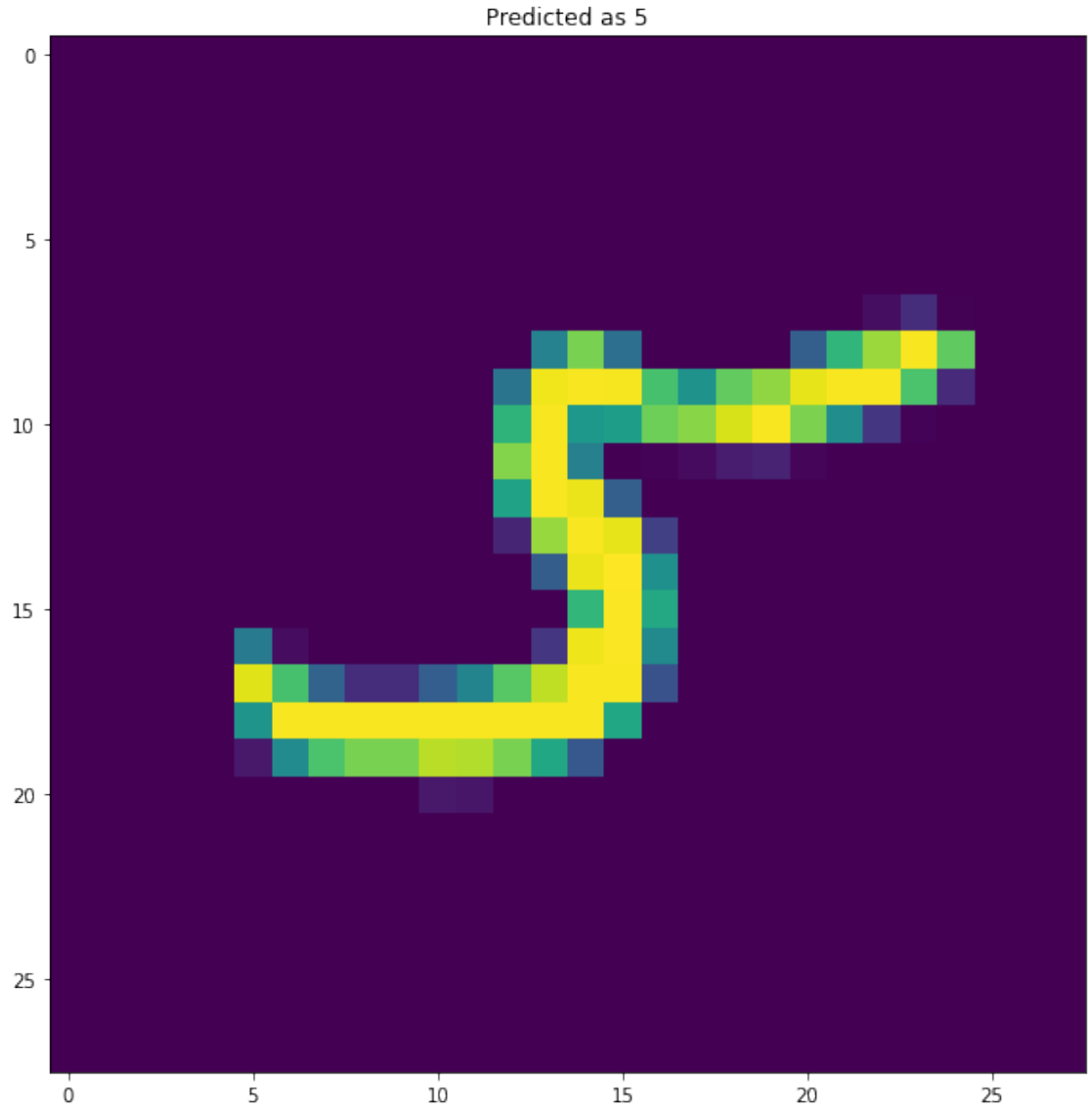
9 Average Prediction Accuracy

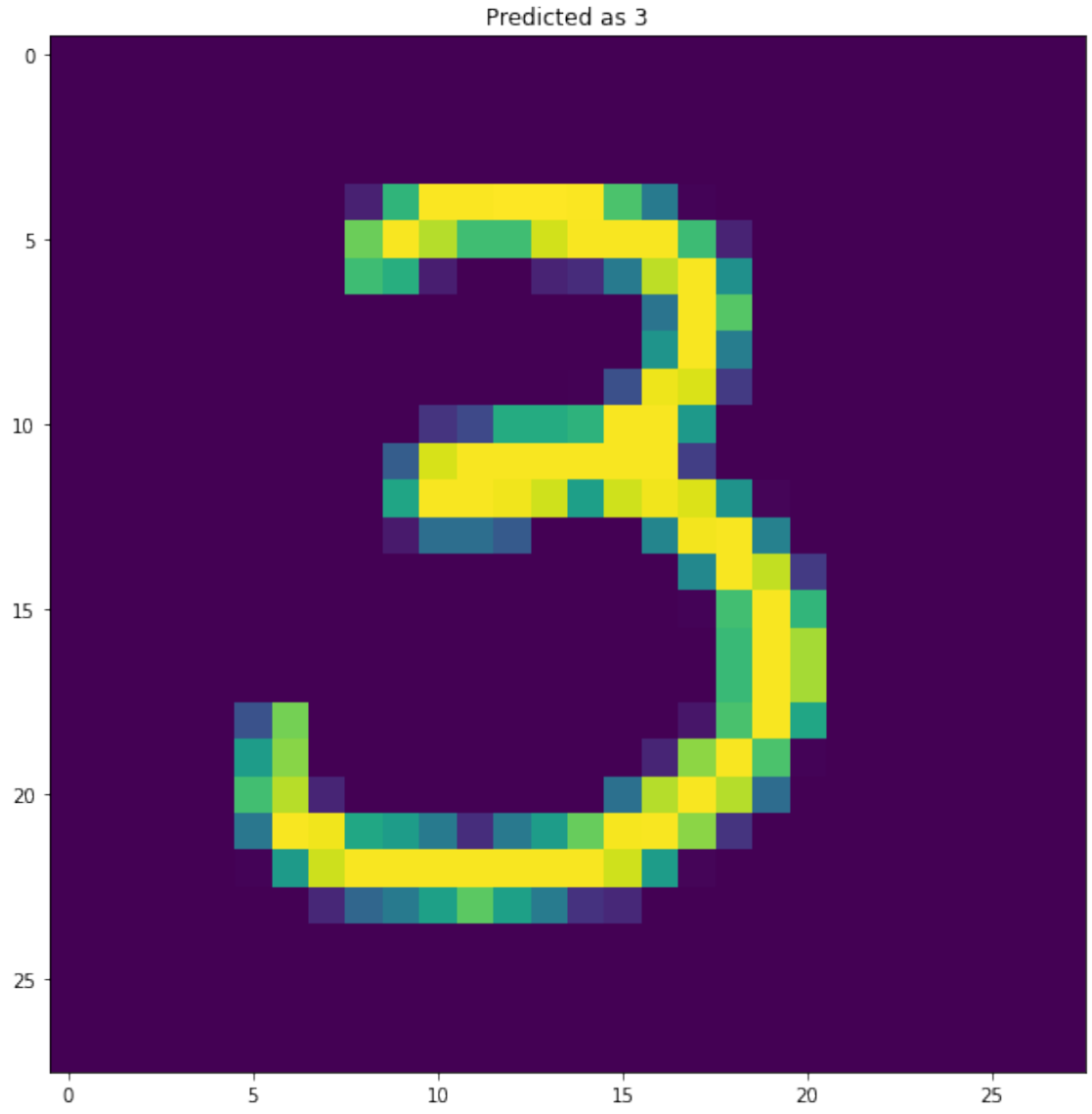
[ ]:
```
#Average prediction accuracy
acc = np.mean(np.asarray(validation_accuracies))
print(f'Average prediction Accuracy is:{acc*100}%')
```

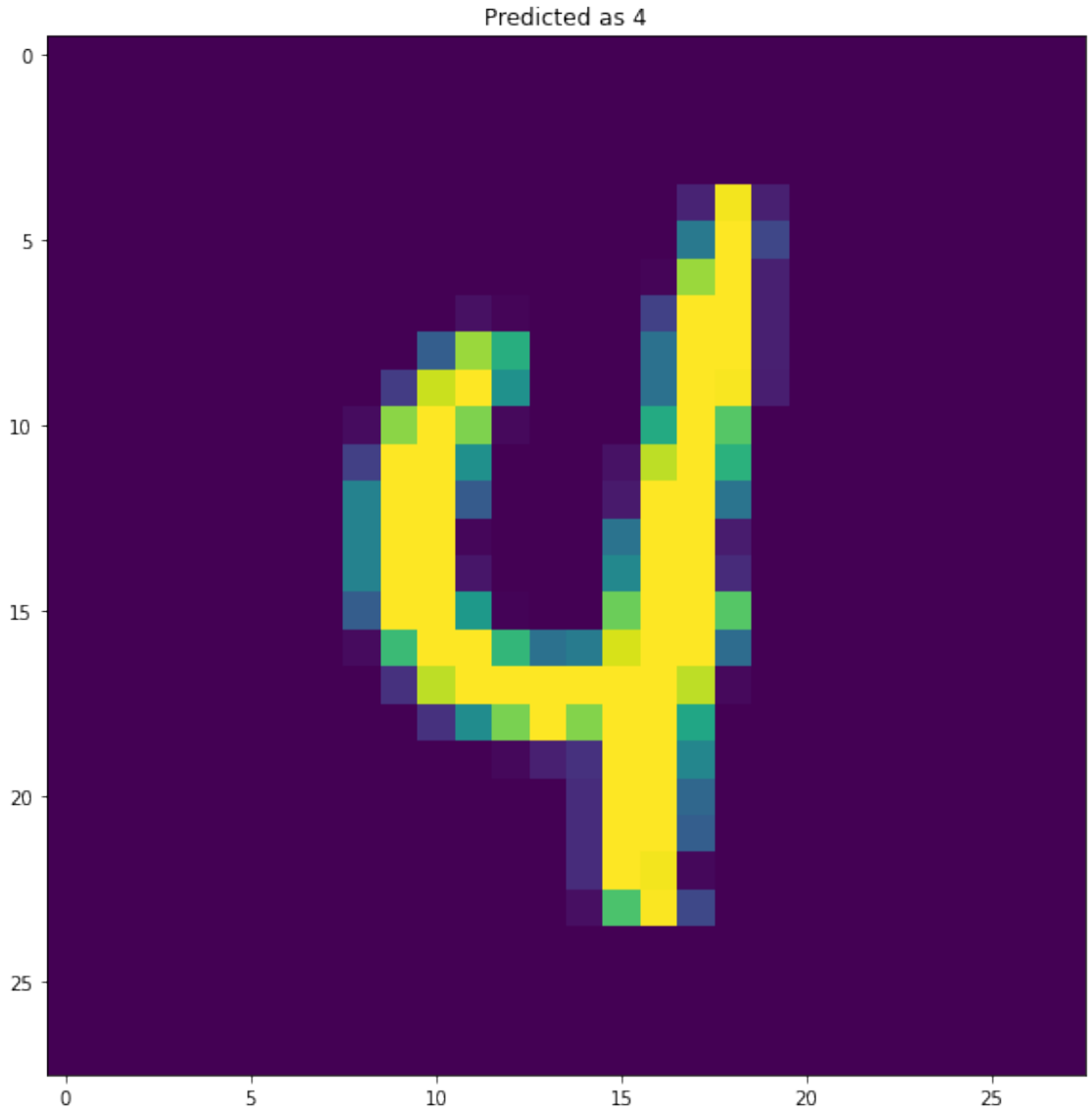
```
Average prediction Accuracy is:96.83500000000001%
```

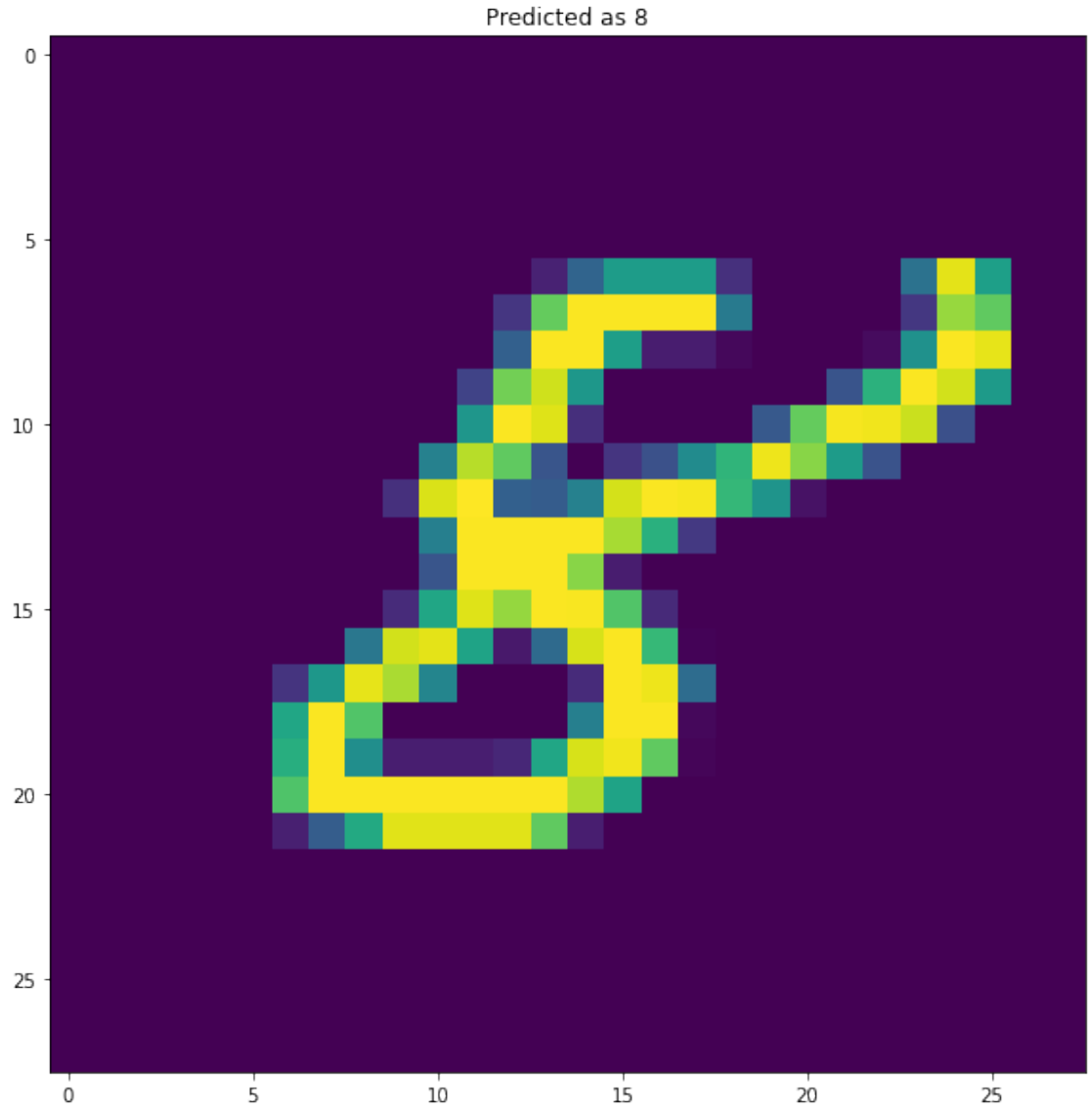
10 Taking Random Images and plotting them

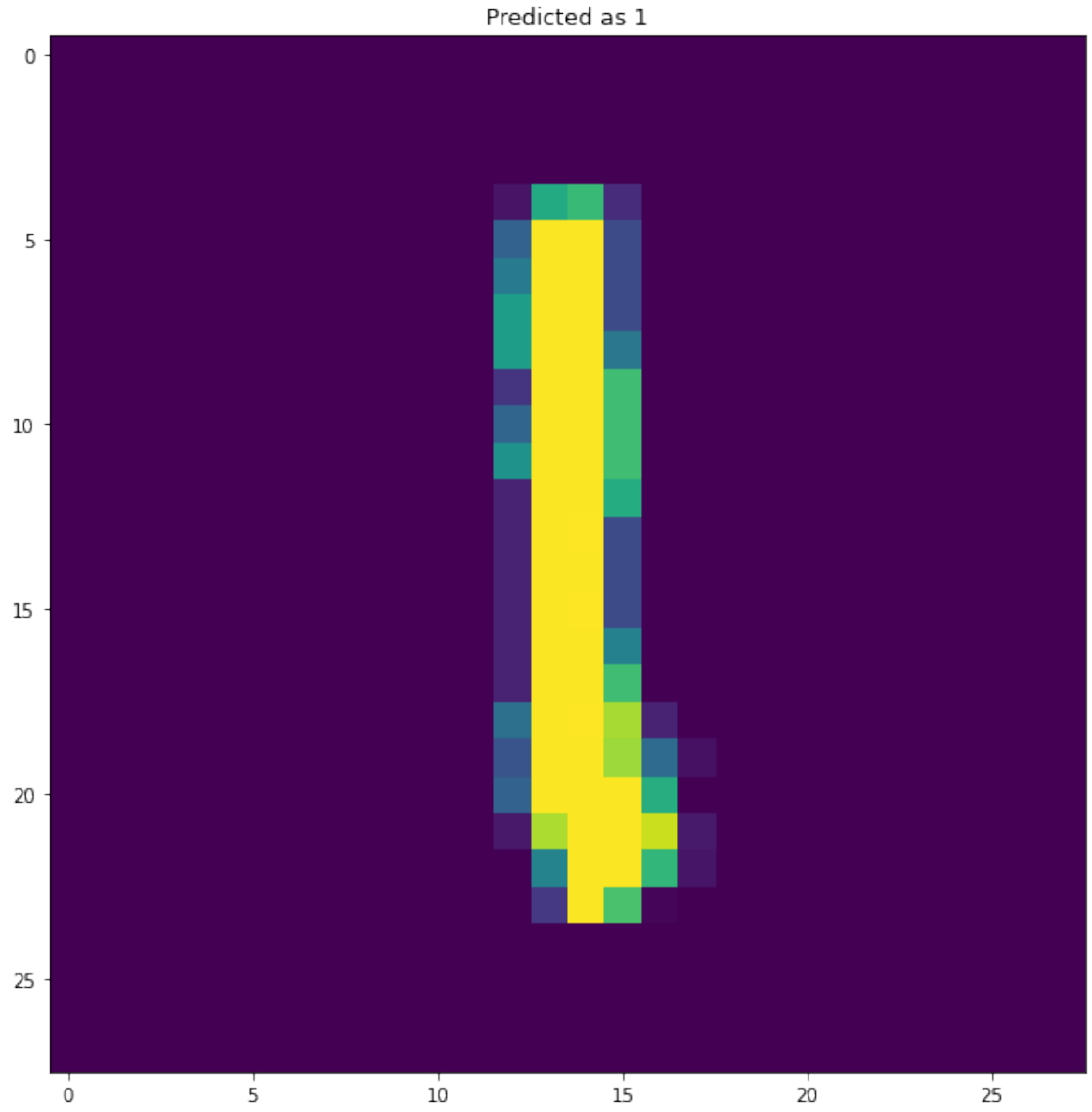
[ ]:
```
rand_index = np.random.randint(low = 0, high = 9999, size = 5)
for idx in rand_index:
  img = test_loader.dataset.data[idx, :, :].clone()
  with torch.no_grad():
    if device==torch.device("cuda"):
      img = img.reshape(1,1,28,28).cuda().float()
    else:
      img = img.reshape(1,1,28,28).float()
    out = model.forward(img).detach().cpu().numpy()
  pred = np.argmax(out)
  plt.imshow(img.detach().cpu().numpy().reshape(28,28))
  plt.title(f'Predicted as {pred}')
  plt.show()
```


Predicted as 5
0
5
10
15
20
25
0
5
10
15
20
25


Predicted as 3
0
5
10
15
20
25
0
5
10
15
20
25


Predicted as 4
0
5
10
15
20
25
0
5
10
15
20
25


Predicted as 8
0
5
10
15
20
25
0
5
10
15
20
25



11 Dimensions of Input and Output of each layer

- Layer 1

  After Convolution

  - Input - 28 X 28
  - Output - 28 X 28 X 32

  After Maxpool

  - Input - 28 X 28 X 32
  - Ouput - 14 X 14 X 32

- Layer 2

  After Convolution

  - Input - 14 X 14 X 32
  - Output - 14 X 14 X 32

  After Maxpool

  - Input - 14 X 14 X 32
  - Ouput - 7 X 7 X 32

- Layer 3

  Fully Connected Layer

  - Input - 7 X 7 X 32
  - Ouput - 500

- Layer 4

  Fully Connected Layer

  - Input - 500
  - Ouput - 10

12 Parameters

- Layer 1

  Convolutional layer 1

  - Weights = 3 X 3 X 28 = 288
  - Biases = 32
  - Total = 320

  Weights for Pooling Layer is 0

- Layer 2

  Convolutional layer 2

  - Weights = 32 X 3 X 3 X 32 = 9216
  - Biases = 32
  - Total = 9248

  Weights for Pooling Layer is 0

- Layer 3

  Fully Connected Layer 1

  - Weights = 7 X 7 X 32 X 500 = 784000
  - Biases = 500
  - Total = 784500

- Layer 4

  Fully Connected Layer 2

  - Weights = 500 X 10 = 5000
  - Biases = 10
  - Total = 5010

So, Total number of parameters = 799078

13 Number of Neurons

- Layer 1
- 1 X 3 X 3 X 32 = 288
- Layer 2
- 3 X 3 X 3 X 32 = 9216
- Layer 3
- 500
- Layer 4
- 10

So, total number of neurons are = 9564

14 Batch Normalisation

- Batch normalisation shows almost same effects as the model without batch normalisation, not much change in accuracy

---

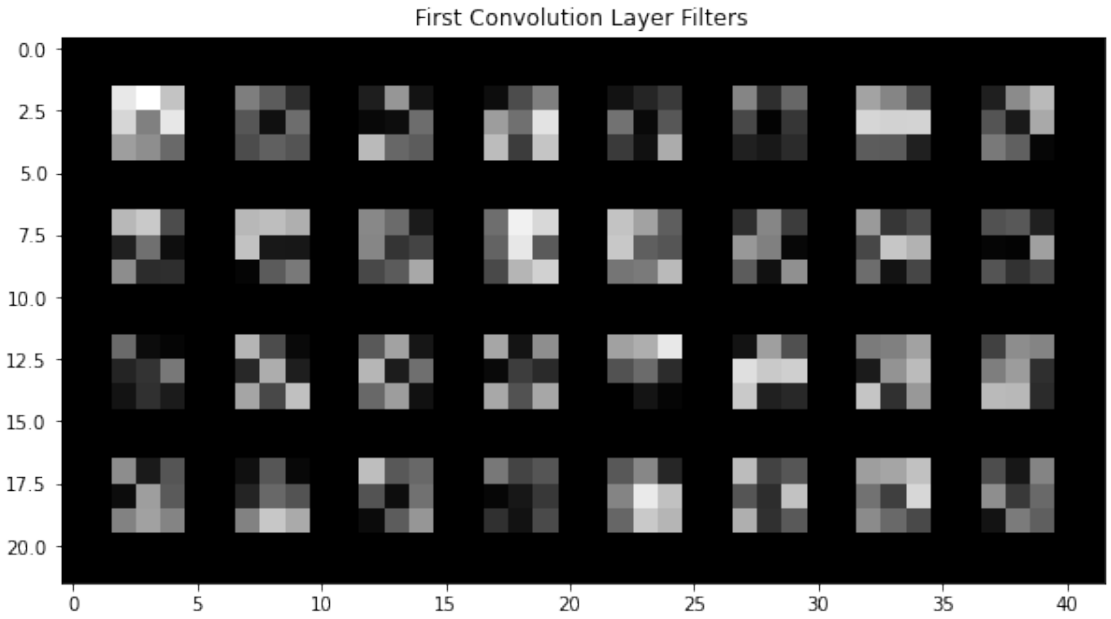
15 VISUALISING THE MODEL

16 Plot Kernels

Visualising kernel 1

- Convolutional layer 1 _____

[ ]:
```
kernel1 = model.conv1.weight.detach().clone()
kernel1 = kernel1.cpu()
kernel1 = kernel1-kernel1.min()
kernel1 = kernel1/kernel1.max()
img1 = make_grid(kernel1)
plt.imshow(img1.permute(1,2,0))
plt.title("First Convolution Layer Filters")
plt.show()
```

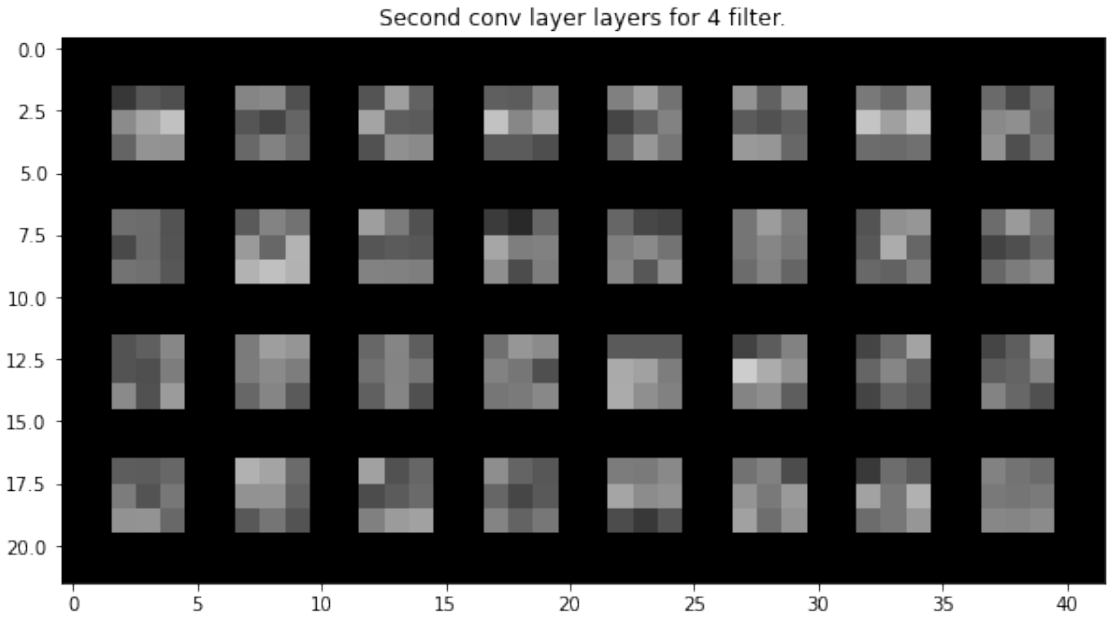


Visualising Kernel 2

- Convolutional Layer 2 _____

[ ]:
```
kernel2 = model.conv2.weight.detach().clone()
kernel2 = kernel2.cpu()
kernel2 = kernel2 - kernel2.min()
kernel2 = kernel2/kernel2.max()
filter_number = int(input("\nChoose the filter number between 0 and 31 which␣
 ↪you want to visualize.\n"))
temp_kernel = kernel2[filter_number, :, :, :].reshape(32, 1, 3, 3)
img = make_grid(temp_kernel)
print(f'shape of the image is {img.shape}')
plt.imshow(img.permute(1, 2, 0))
to_show = int(filter_number)+1
plt.title(f"Second conv layer layers for {to_show} filter.")
plt.show()
```

```
Choose the filter number between 0 and 31 which you want to visualize.
3
shape of the image is torch.Size([3, 22, 42])
```



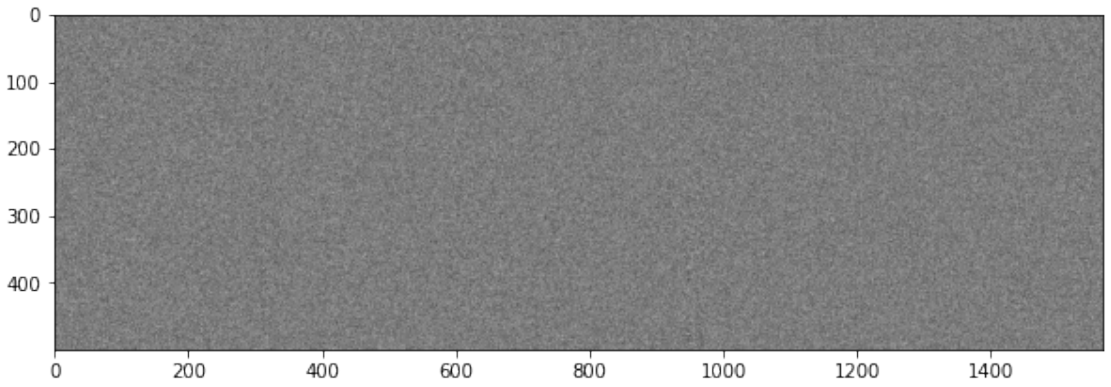
Visualising kernel 3

- Fully Connected layer 1 ____

[ ]:
```
kernel3 = model.fc1.weight.detach().clone()
kernel3 = kernel3.cpu()
kernel3 = kernel3 - kernel3.min()
kernel3 = kernel3/kernel3.max()
img = make_grid(kernel3)
print(f'shape of the image is {img.shape}')
plt.imshow(img.permute(1, 2, 0))
```

```
shape of the image is torch.Size([3, 500, 1568])
```

[ ]:
```
<matplotlib.image.AxesImage at 0x7fe659ea6ed0>
```



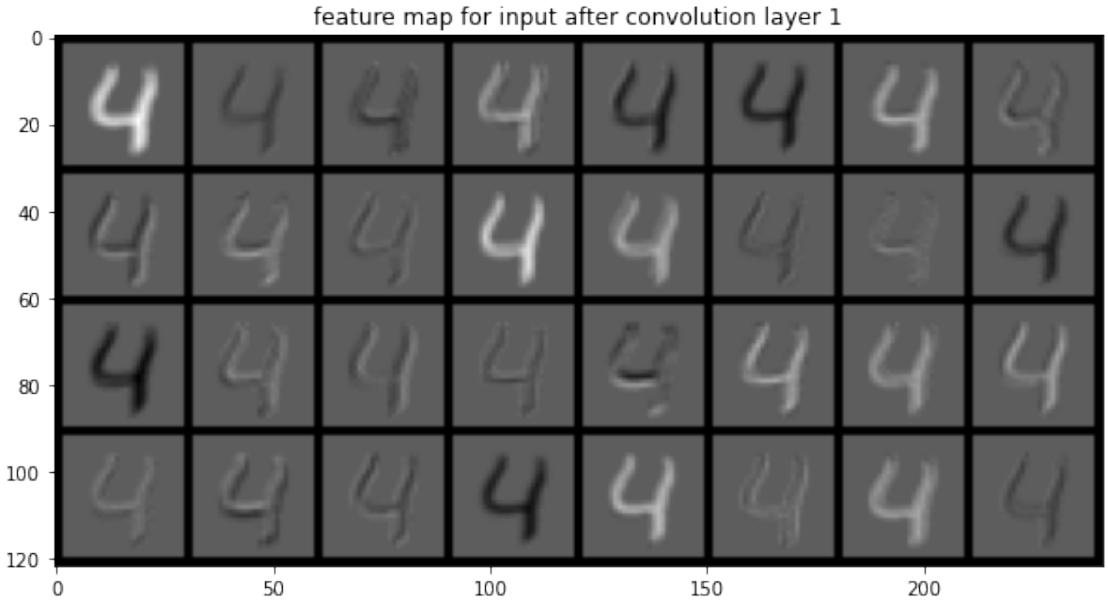
16.1 *Observations:*

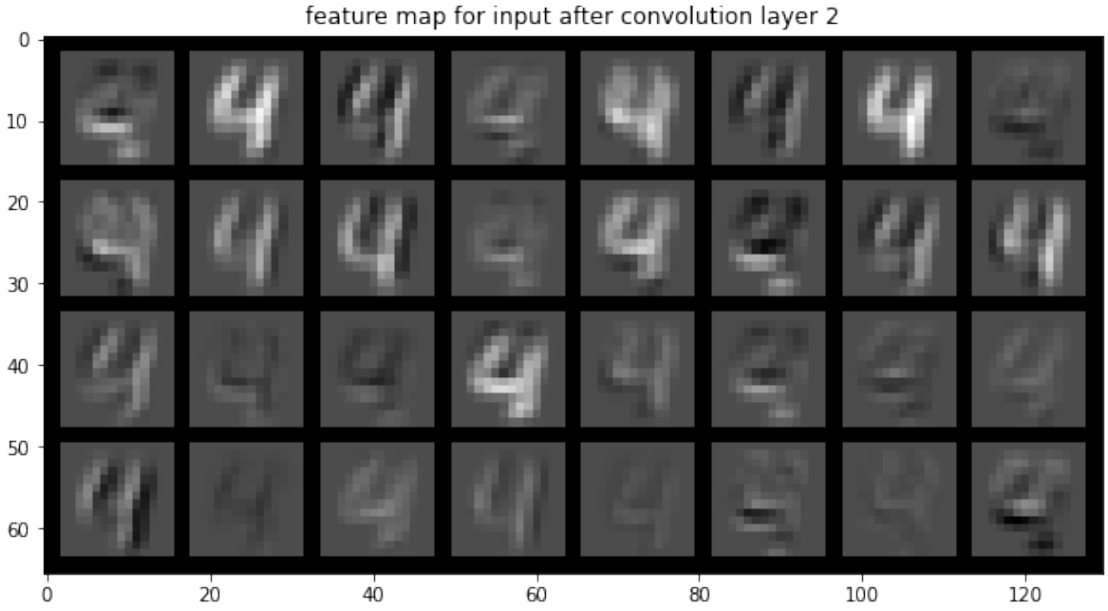
- Convolutional kernel 1 is more sharp whereas Convolutional kernel 2 is more smoothened.
- In convolutional kernel we can white and black patches in grids whereas in Convolutional kernel 2 we can observe grey patches in all grids

17 Visualising Features

[ ]:
```
index = int(input("\nChoose the number for which you want to visualise feature␣
 ↪map and occlusion effects\t"))
test_img = test_loader.dataset.data[[index], :, :].clone()

if (device == torch.device("cuda")):
  test_image = test_img.reshape(1,1,28,28).clone().cuda().float()
else:
  test_image = test_img.reshape(1,1,28,28).clone().float()
with torch.no_grad():
  out_1 = model.layer1[0].forward(test_image).reshape(32,1,28,28)
  out_1 = out_1.cpu()
  out_1 = out_1 - out_1.min()
  out_1 = out_1/out_1.max()
  img = make_grid(out_1)
  plt.imshow(img.permute(1,2,0))
  plt.title("feature map for input after convolution layer 1")
  plt.show()

out_2 = model.layer1.forward(test_image)
out_2 = model.layer2[0].forward(out_2)
out_2 = out_2.cpu()
out_2 = out_2 - out_2.min()
out_2 = out_2/out_2.max()
out_2 = out_2.reshape(32,1,14,14)
img = make_grid(out_2)
plt.imshow(img.permute(1,2,0))
plt.title("feature map for input after convolution layer 2")
plt.show()
```

```
Choose the number for which you want to visualise feature map and occlusion
effects     4
```



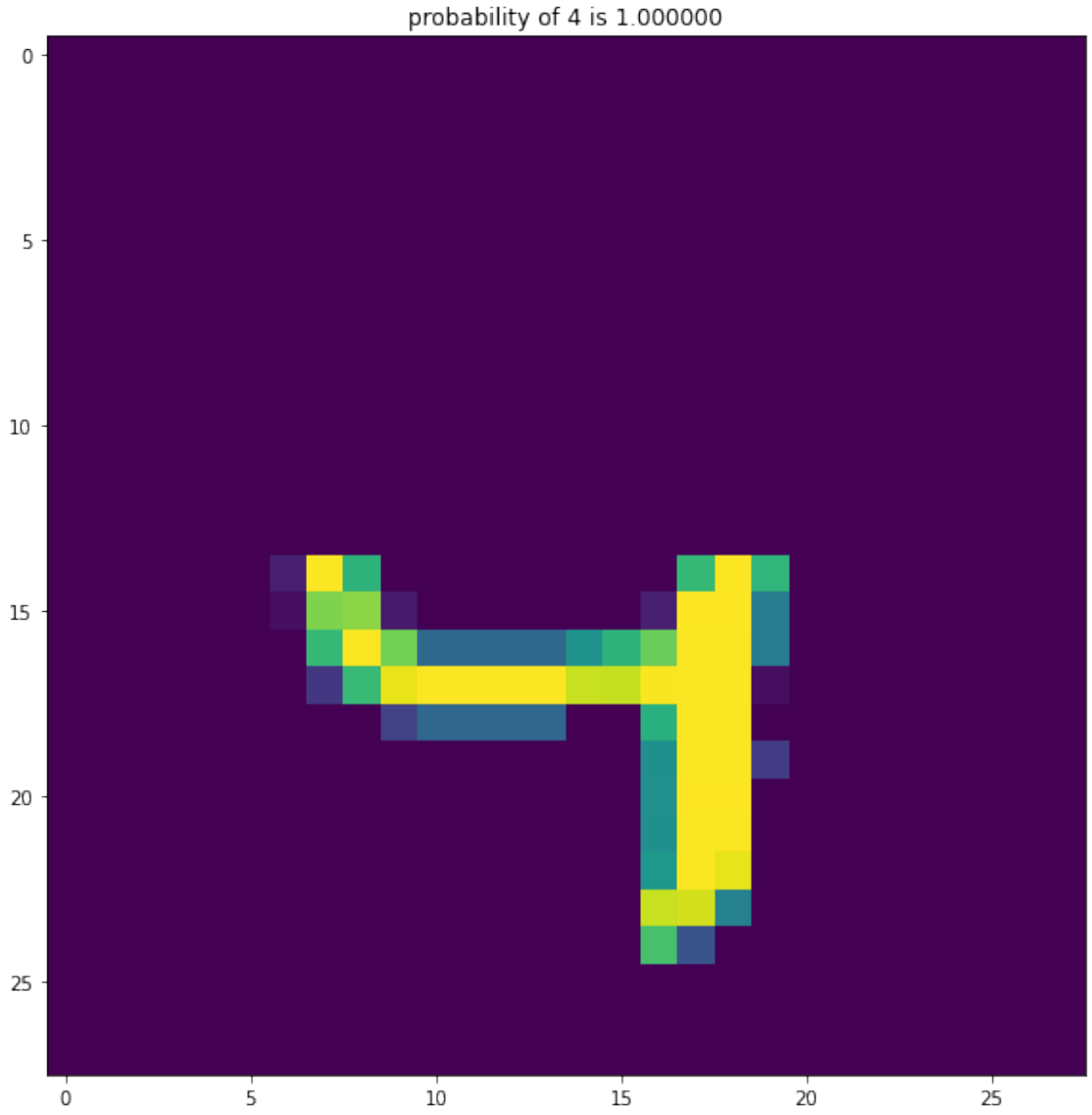


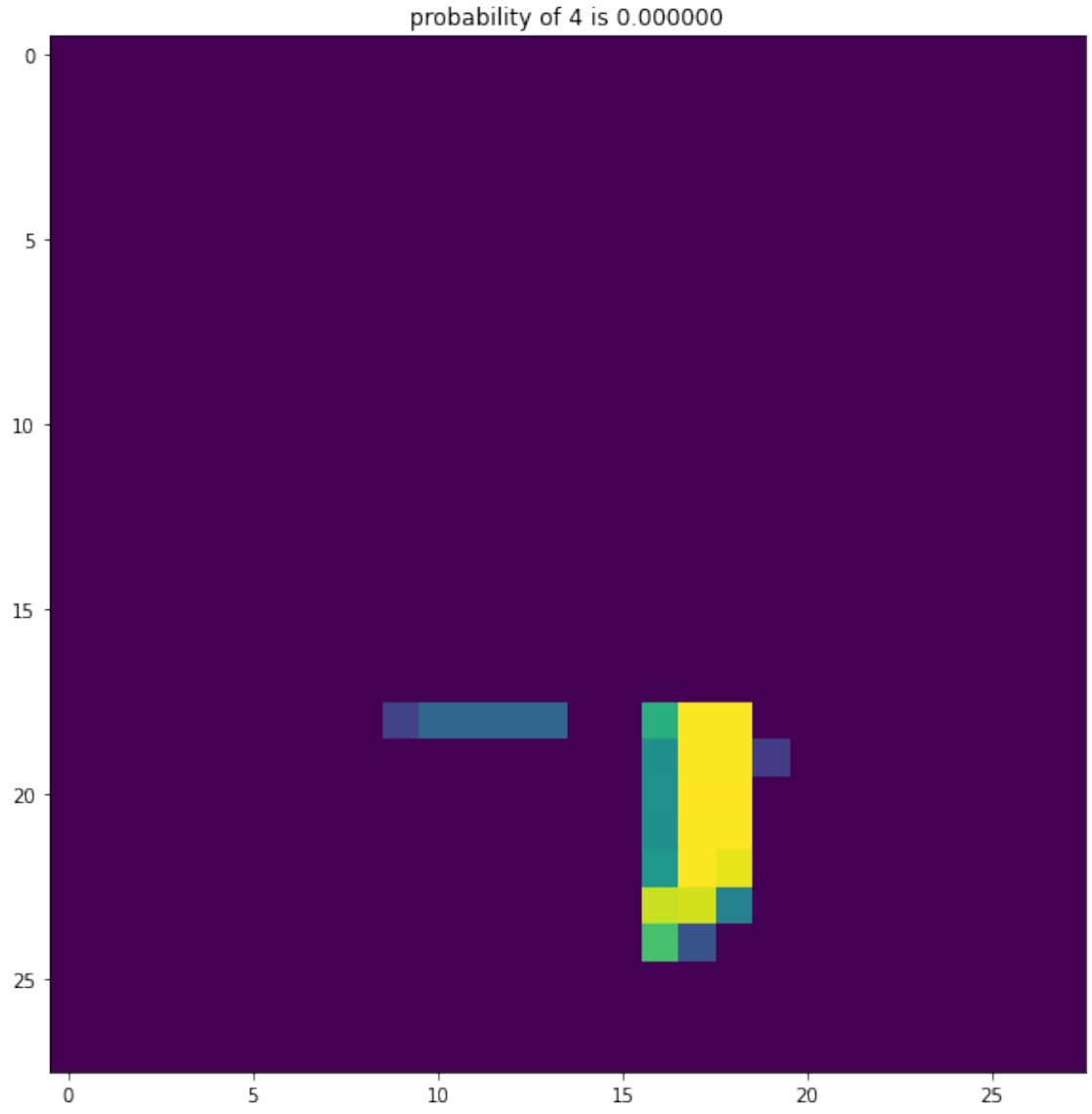
17.1 *Observations:*

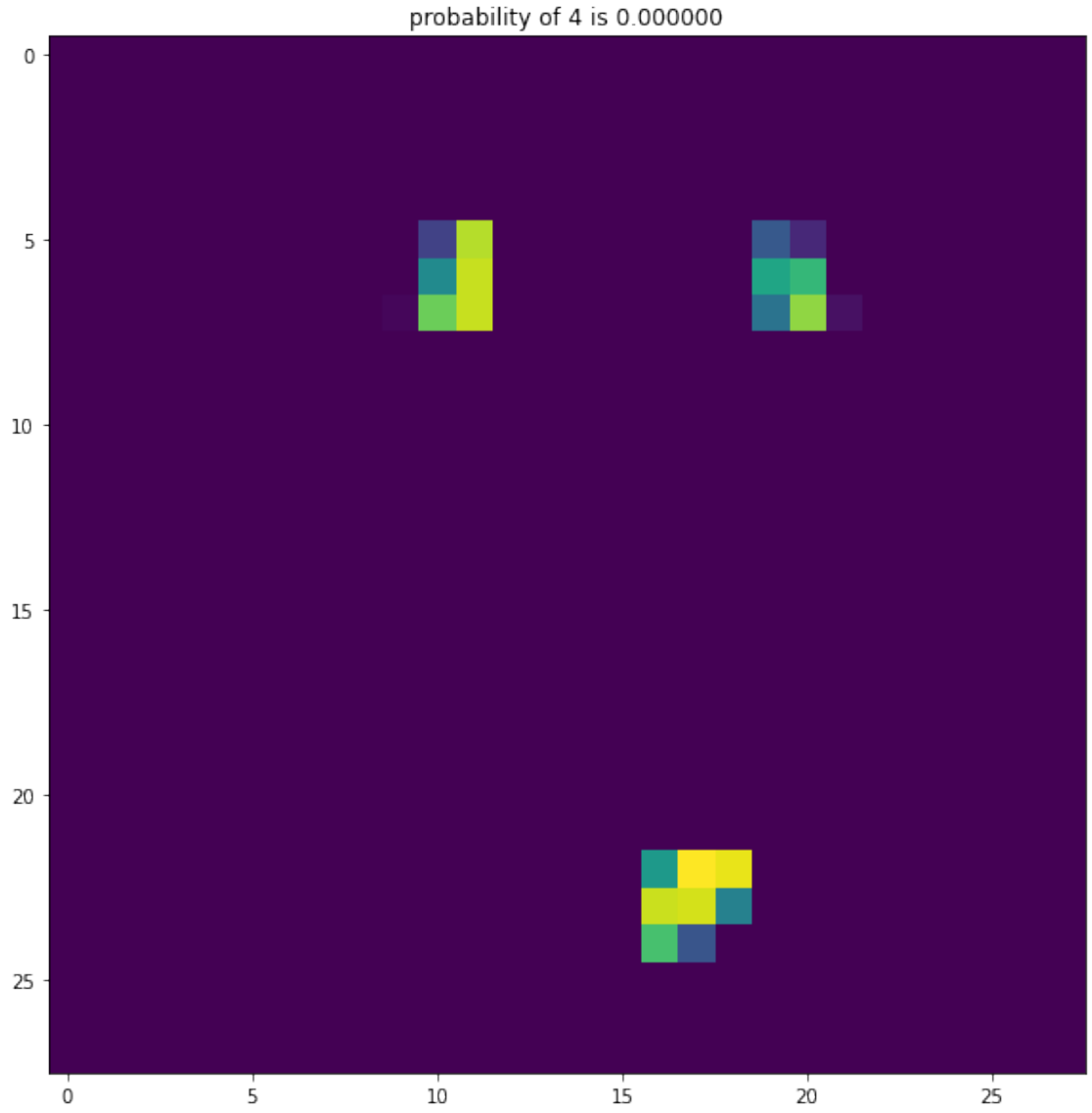
Ouput of Convolutional layer 1 shows proper feature extraction

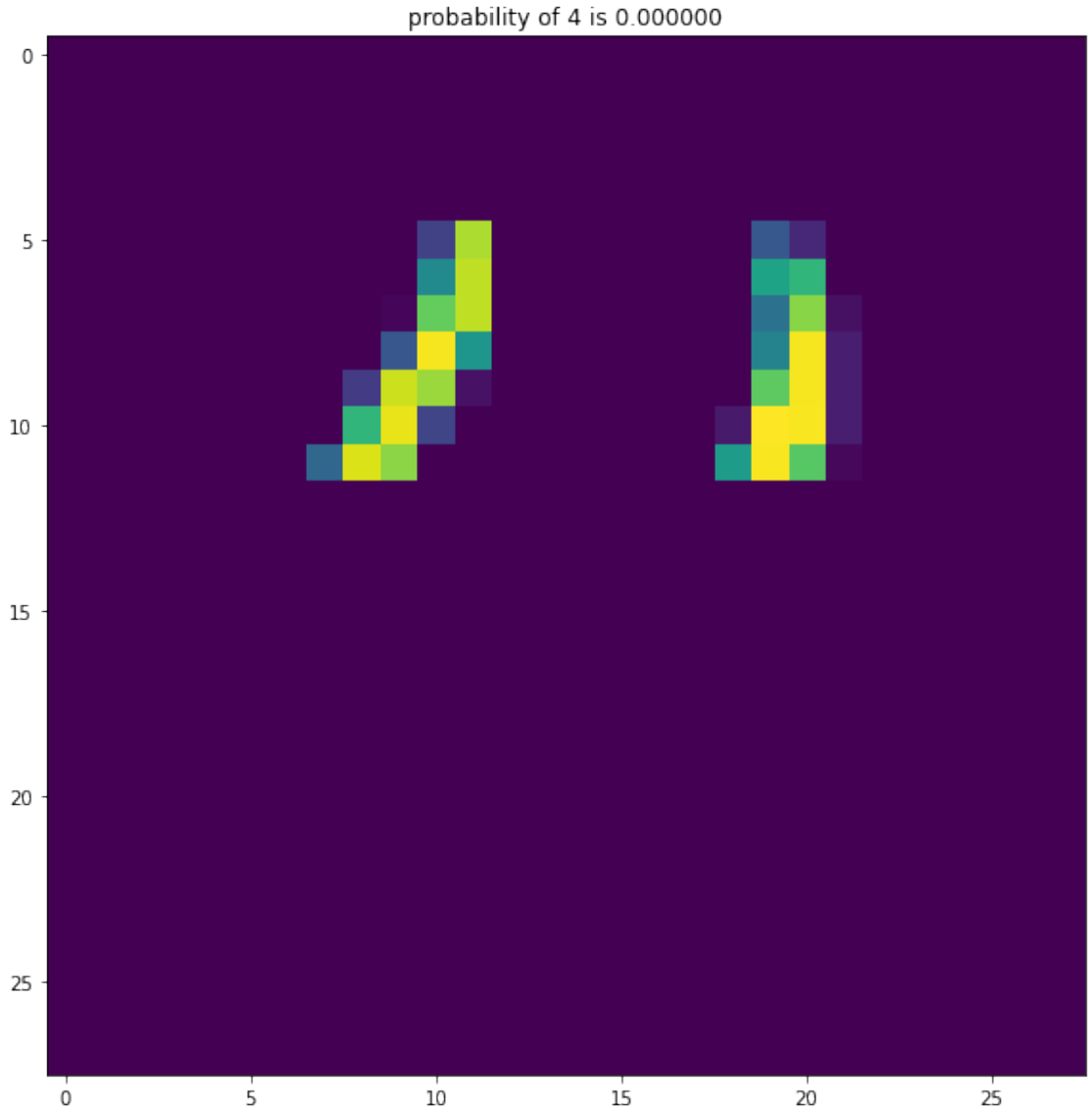
17.2 Visualising Occluding parts of the Image

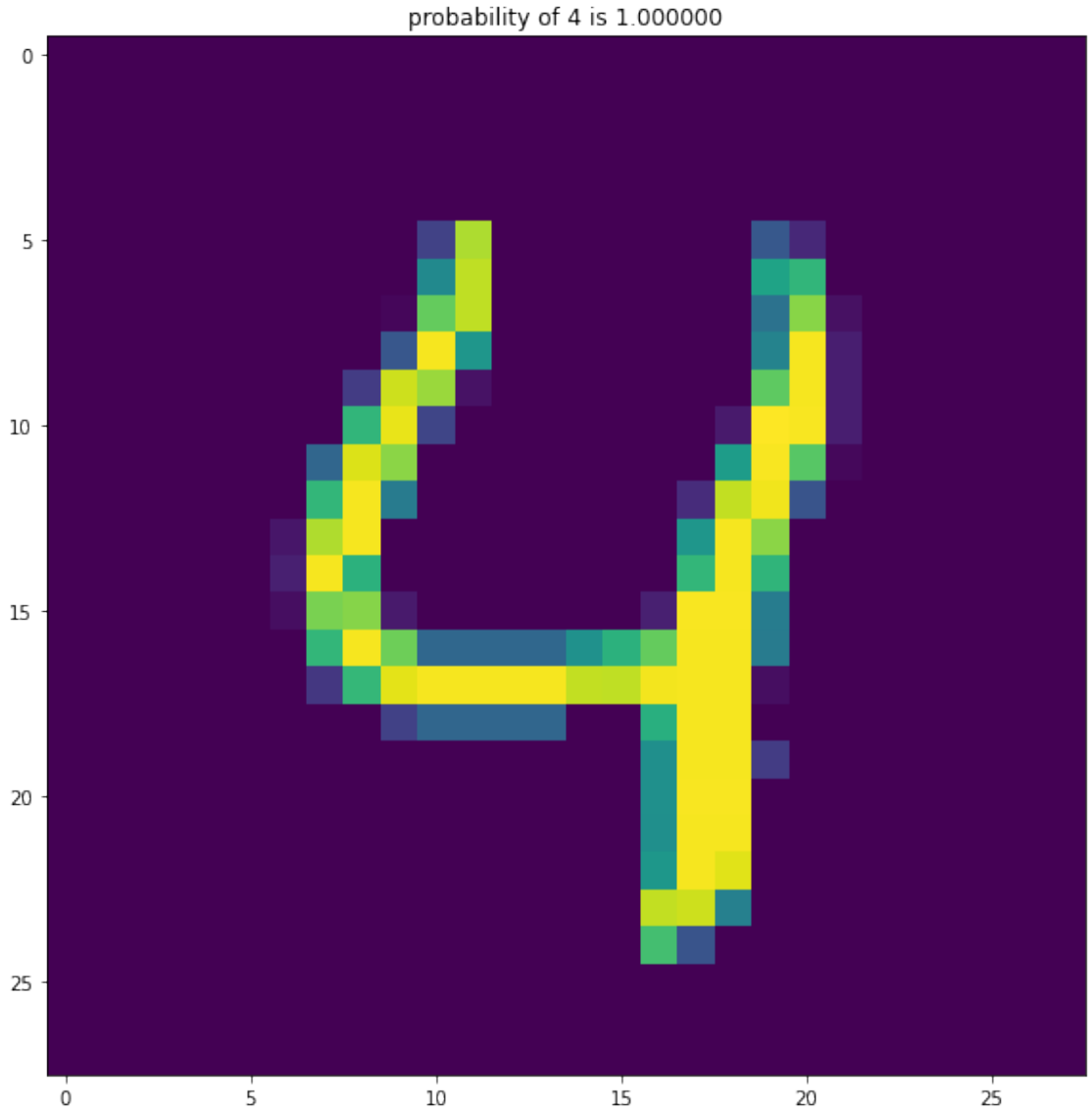
[ ]:
```
dimension = len(range(0,14,2))
prob_map = np.zeros((dimension, dimension), dtype = float)
max_prob_class_map = np.zeros_like(prob_map)
for y in range(0,14,2):
  for x in range(0,14,2):
    temp = test_img.clone()
    temp[y:y+14, x:x+14] = 0
    with torch.no_grad():
      if(device==torch.device("cuda")):
        temp = temp.clone().reshape(1,1,28,28).cuda().float()
      else:
        temp = temp.clone().reshape(1,1,28,28).float()
      out = model.forward(temp, softmax = False)
      probab = F.softmax(out, dim=1).cpu().detach().numpy()
      prediction = np.argmax(probab)
      prob = probab[:, index]
      max_prob_class = prediction
      prob_map[int(y/2),int(x/2)] = prob[0]
      max_prob_class_map[int(y/2), int(x/2)] = max_prob_class
      if ((x%4 == 0)&(y%4==0)):
        plt.imshow(temp.cpu().numpy().reshape(28,28))
        plt.title("probability of {} is {:.6f}".format(index, prob[0]))
        plt.show()
print("\nProbability of {} as patch is move is :".format(index))
print(prob_map)
print("\nMaximum probable class as the patch is moved is :")
print(max_prob_class_map)
```

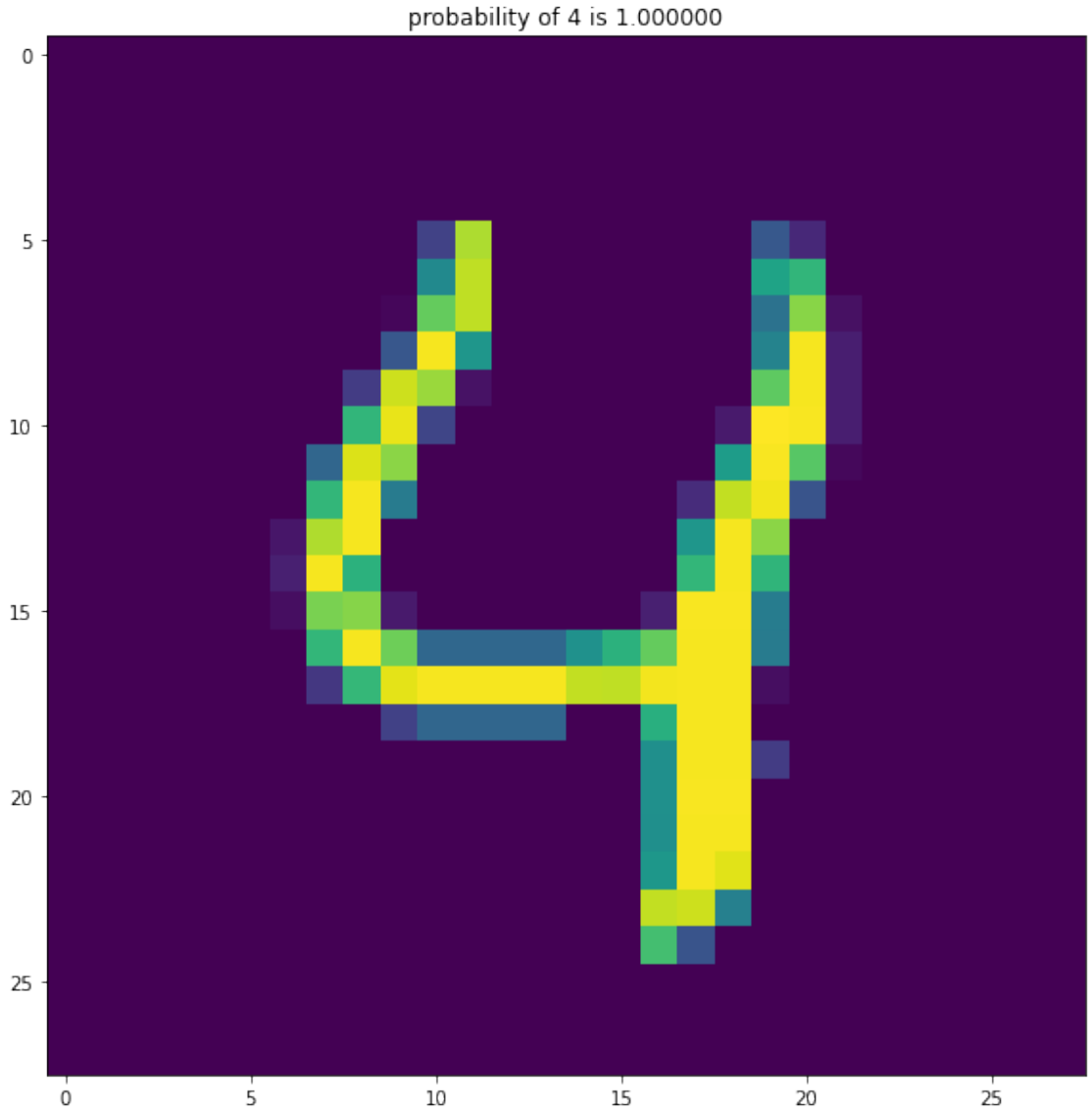

probability of 4 is 1.000000
0
5
10
15
20
25
0
5
10
15
20
25

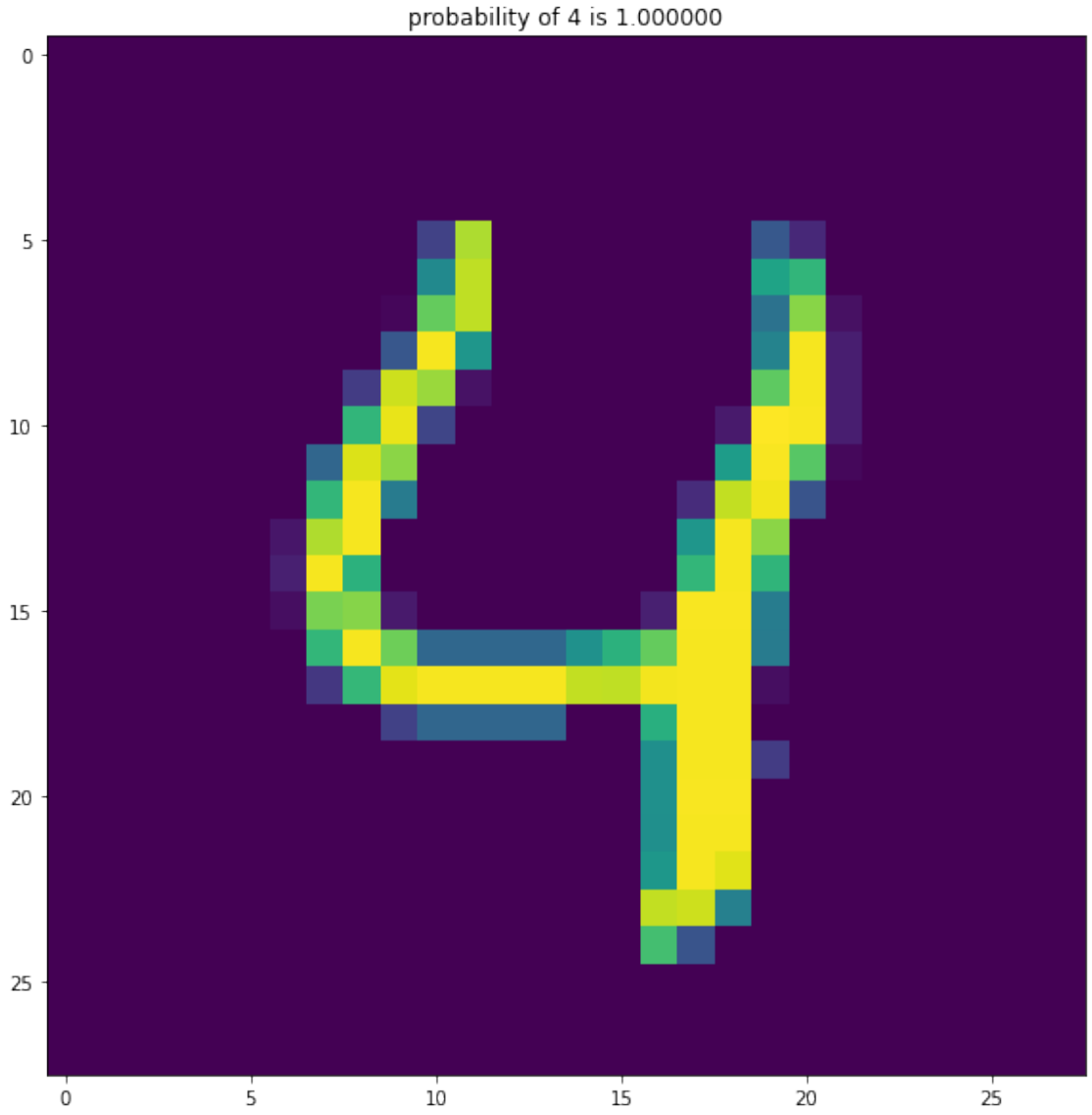

probability of 4 is 0.000000
0
5
10
15
20
25
0
5
10
15
20
25

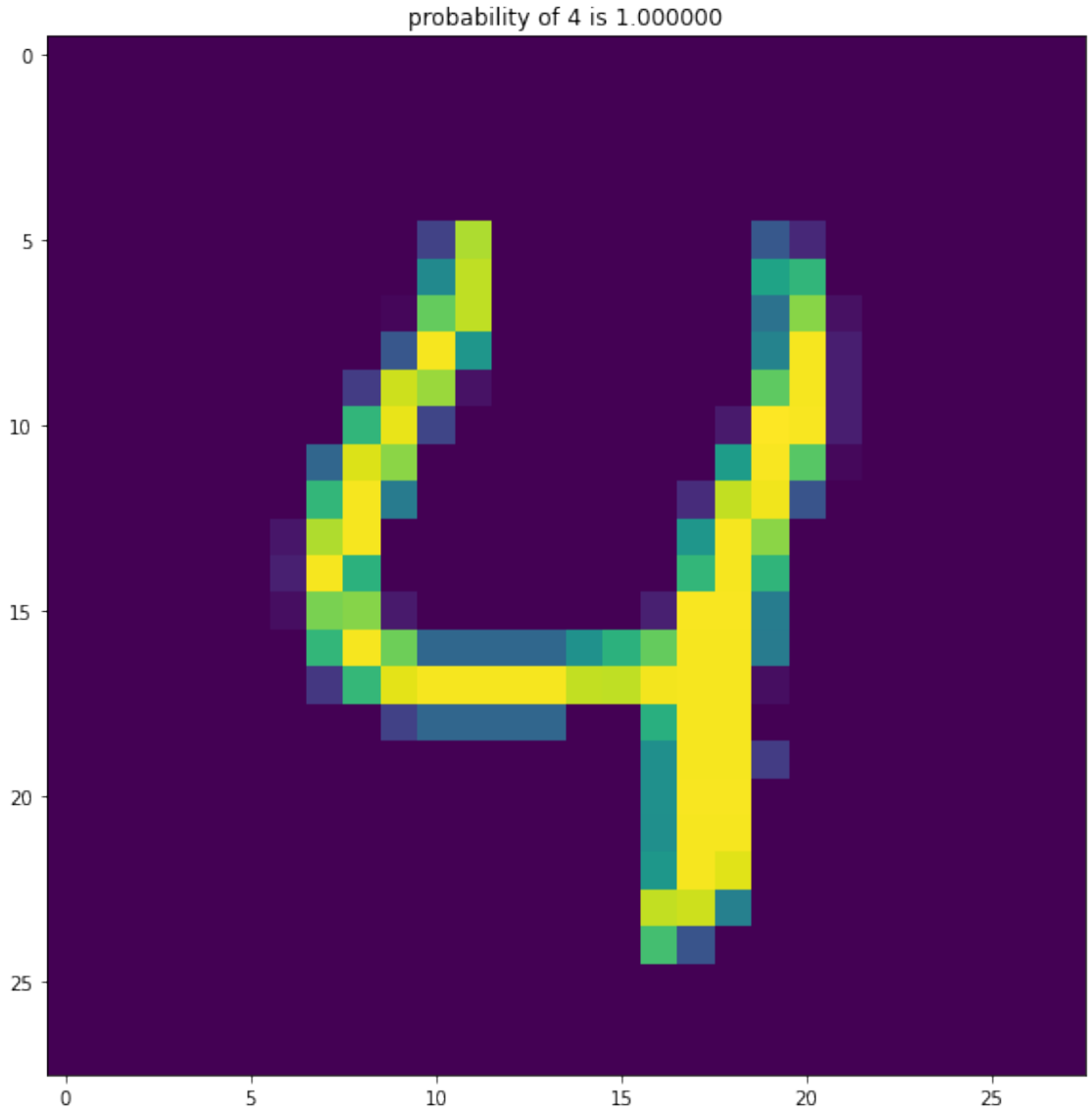

probability of 4 is 0.000000
0
5
10
15
20
25
0
5
10
15
20
25

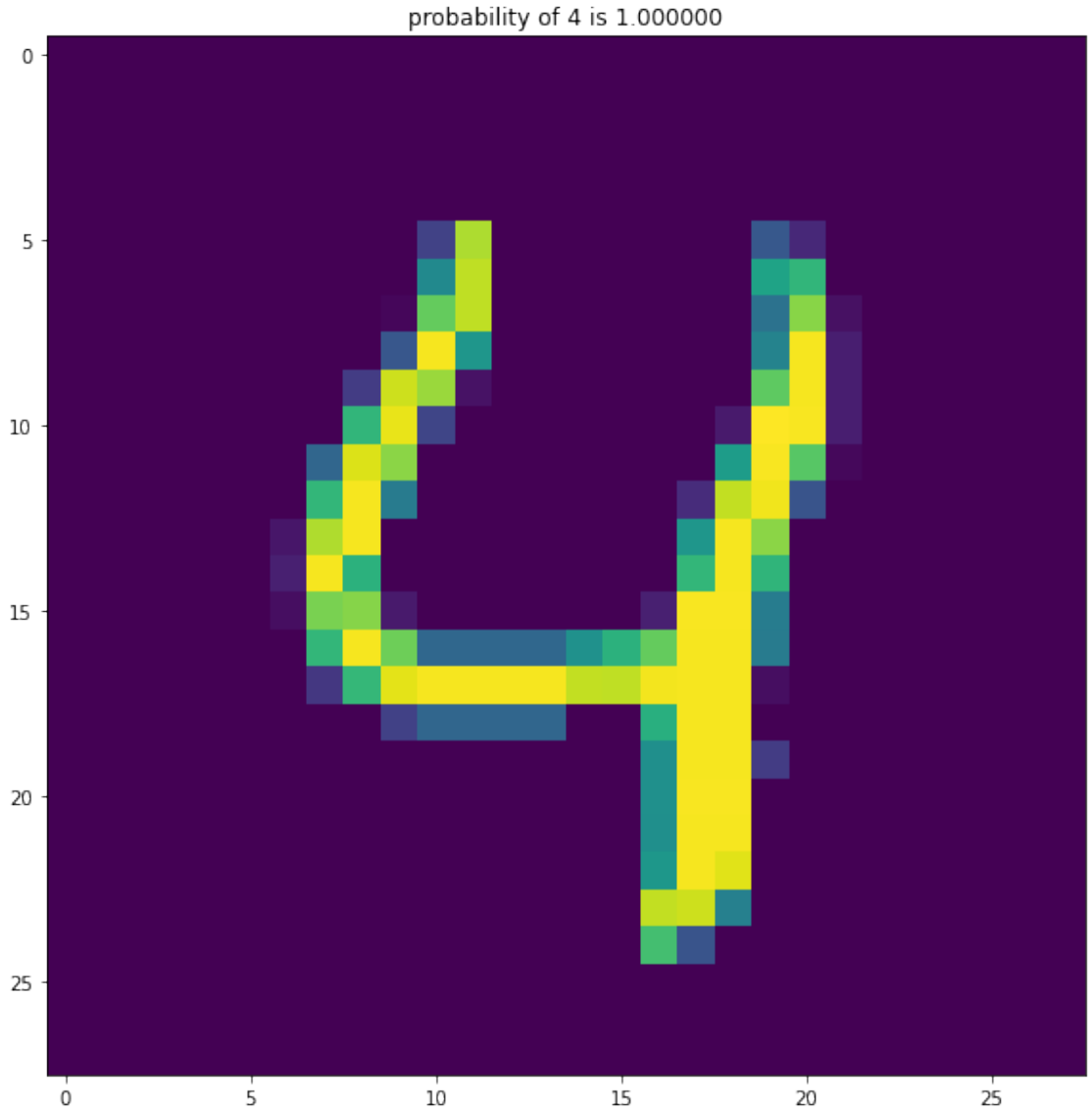

probability of 4 is 0.000000
0
5
10
15
20
25
0
5
10
15
20
25

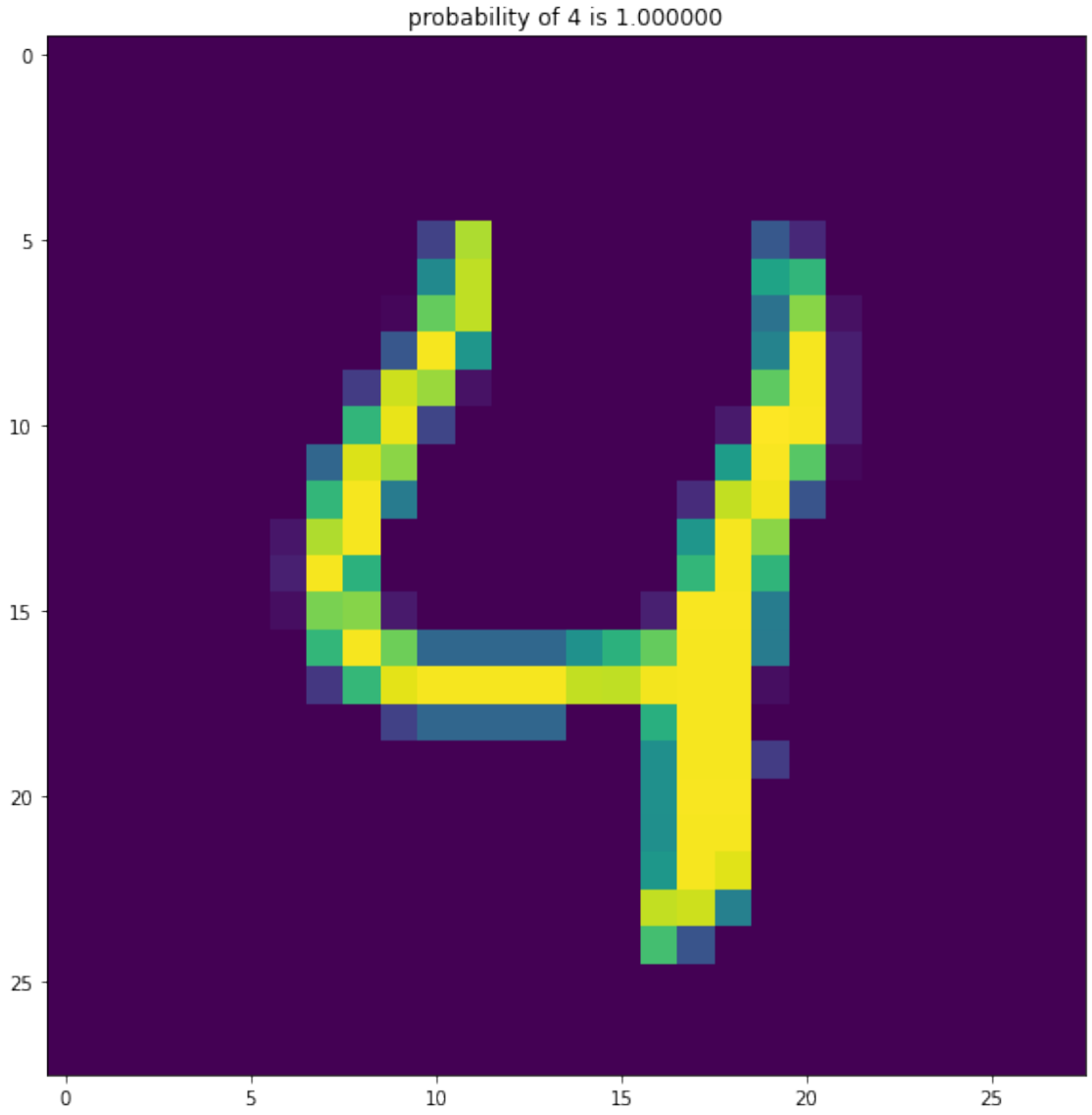

probability of 4 is 1.000000
0
5
10
15
20
25
0
5
10
15
20
25

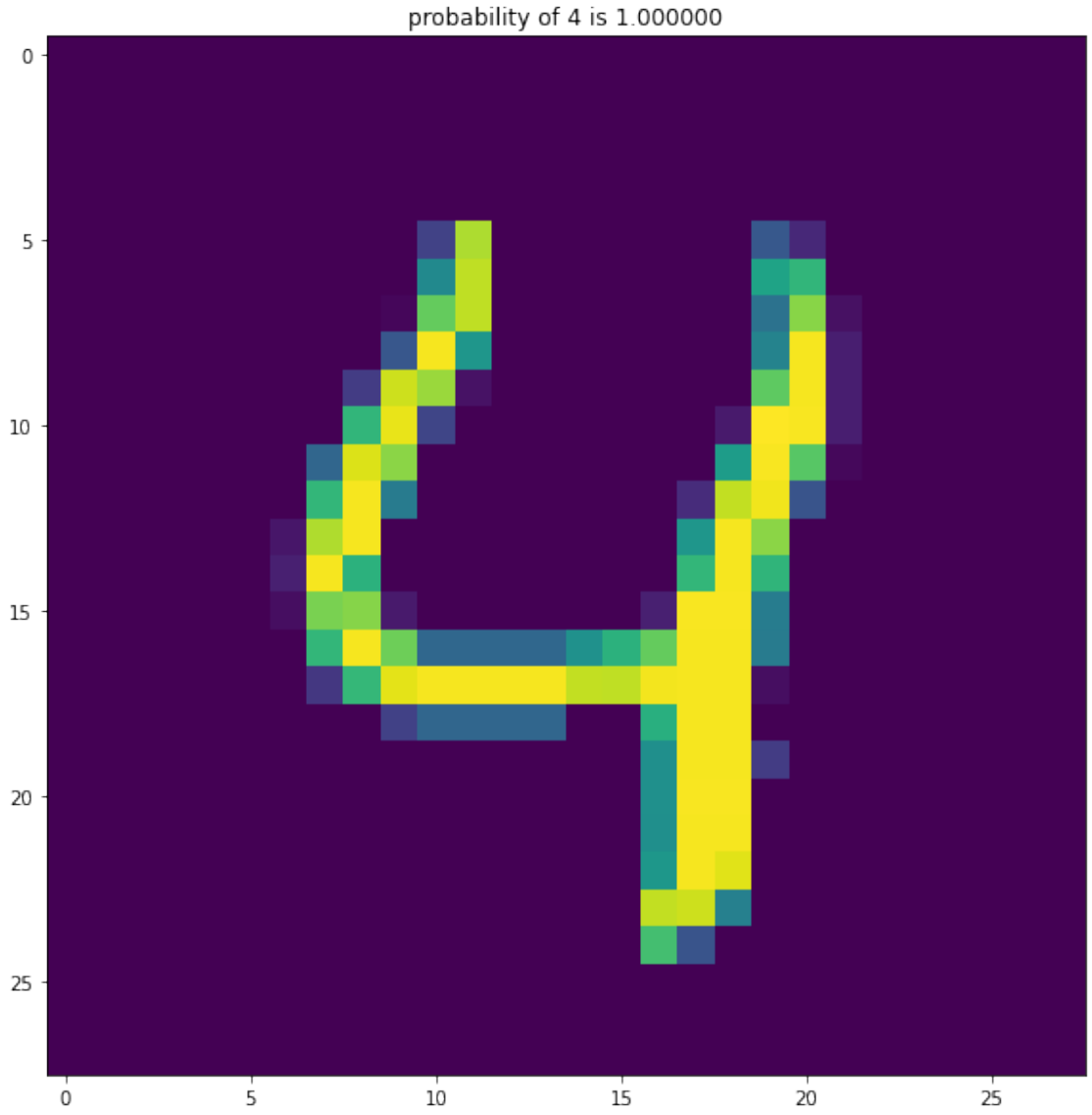

probability of 4 is 1.000000
0
5
10
15
20
25
0
5
10
15
20
25

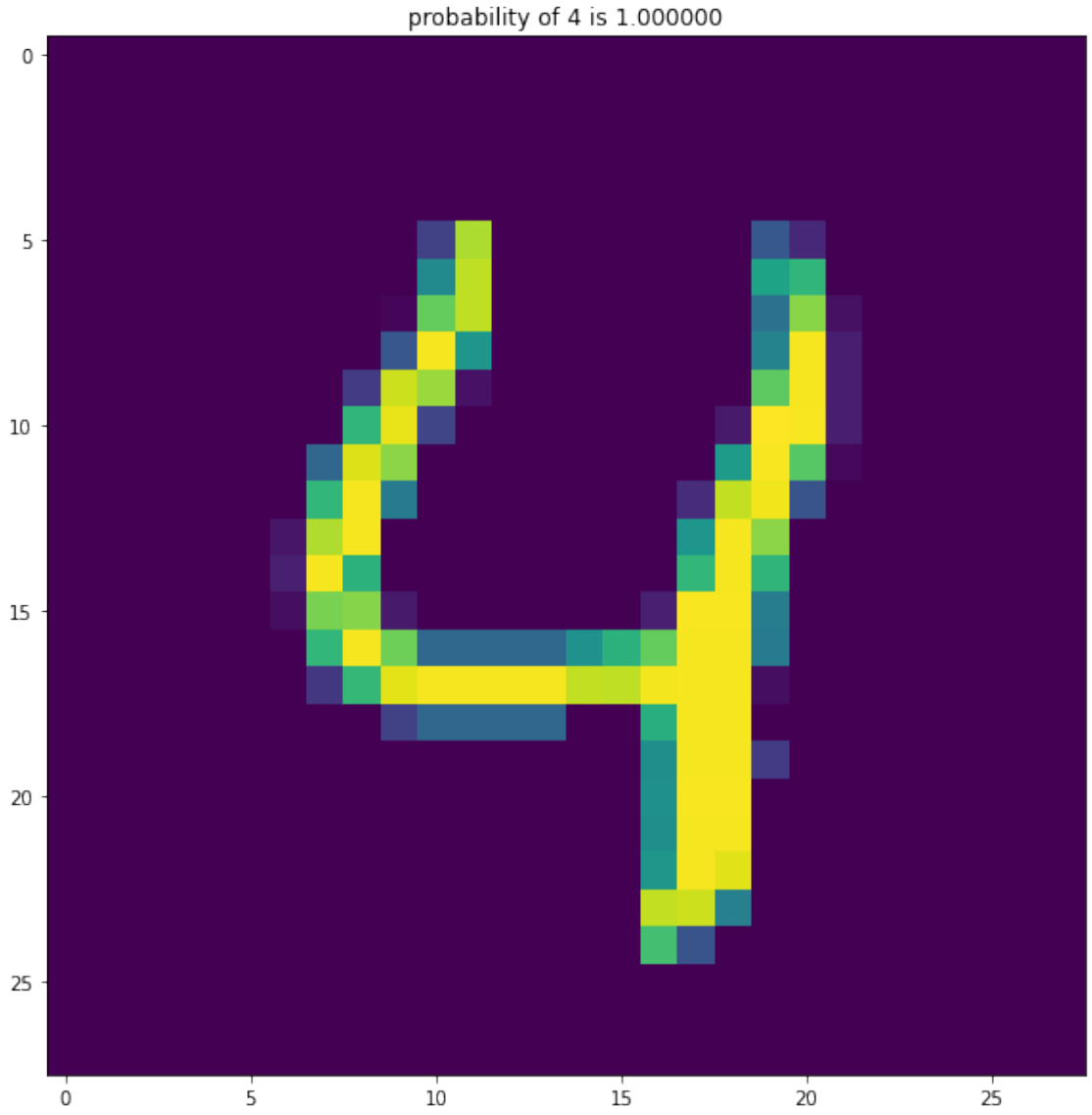

probability of 4 is 1.000000
0
5
10
15
20
25
0
5
10
15
20
25

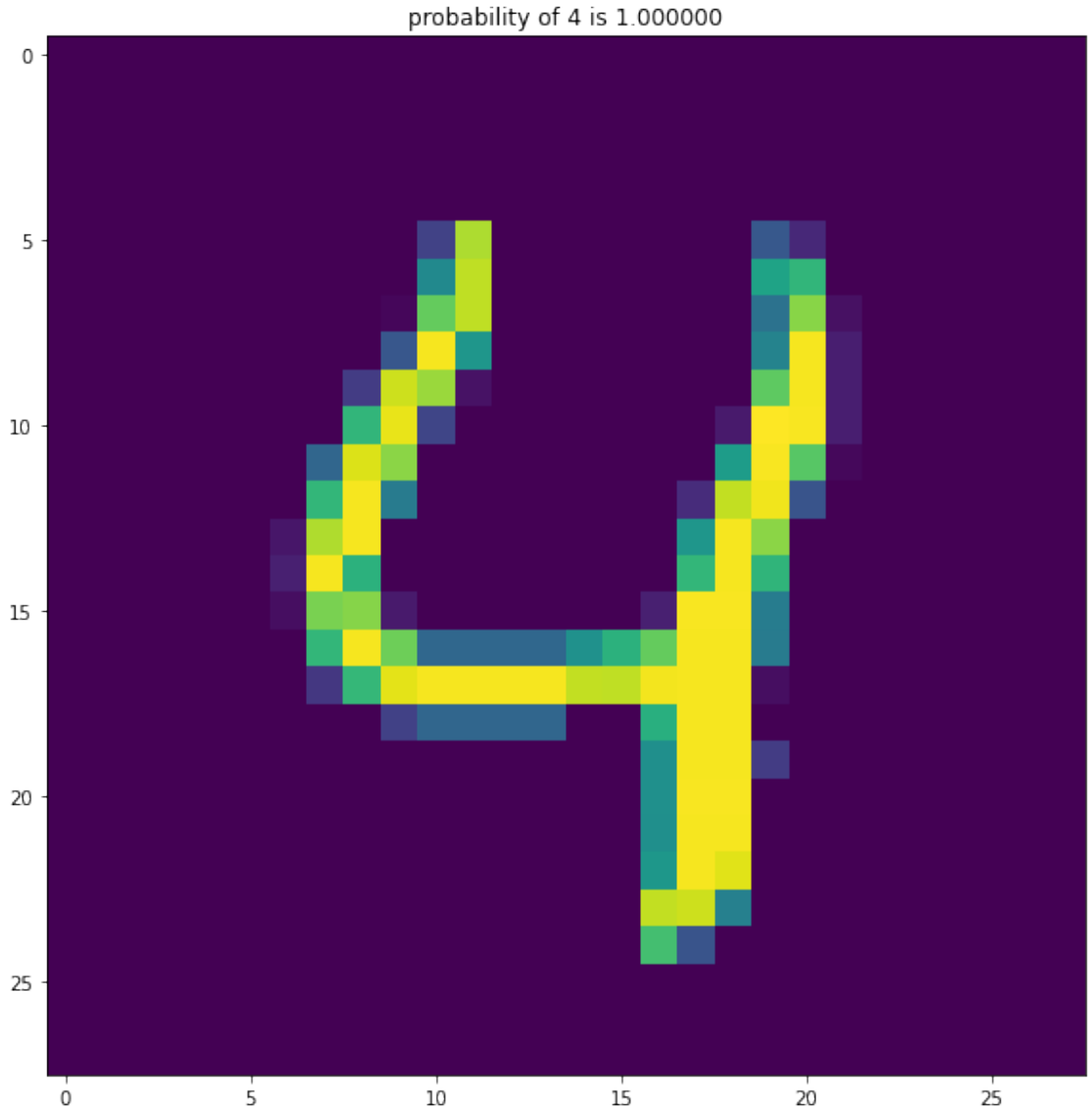

probability of 4 is 1.000000
0
5
10
15
20
25
0
5
10
15
20
25

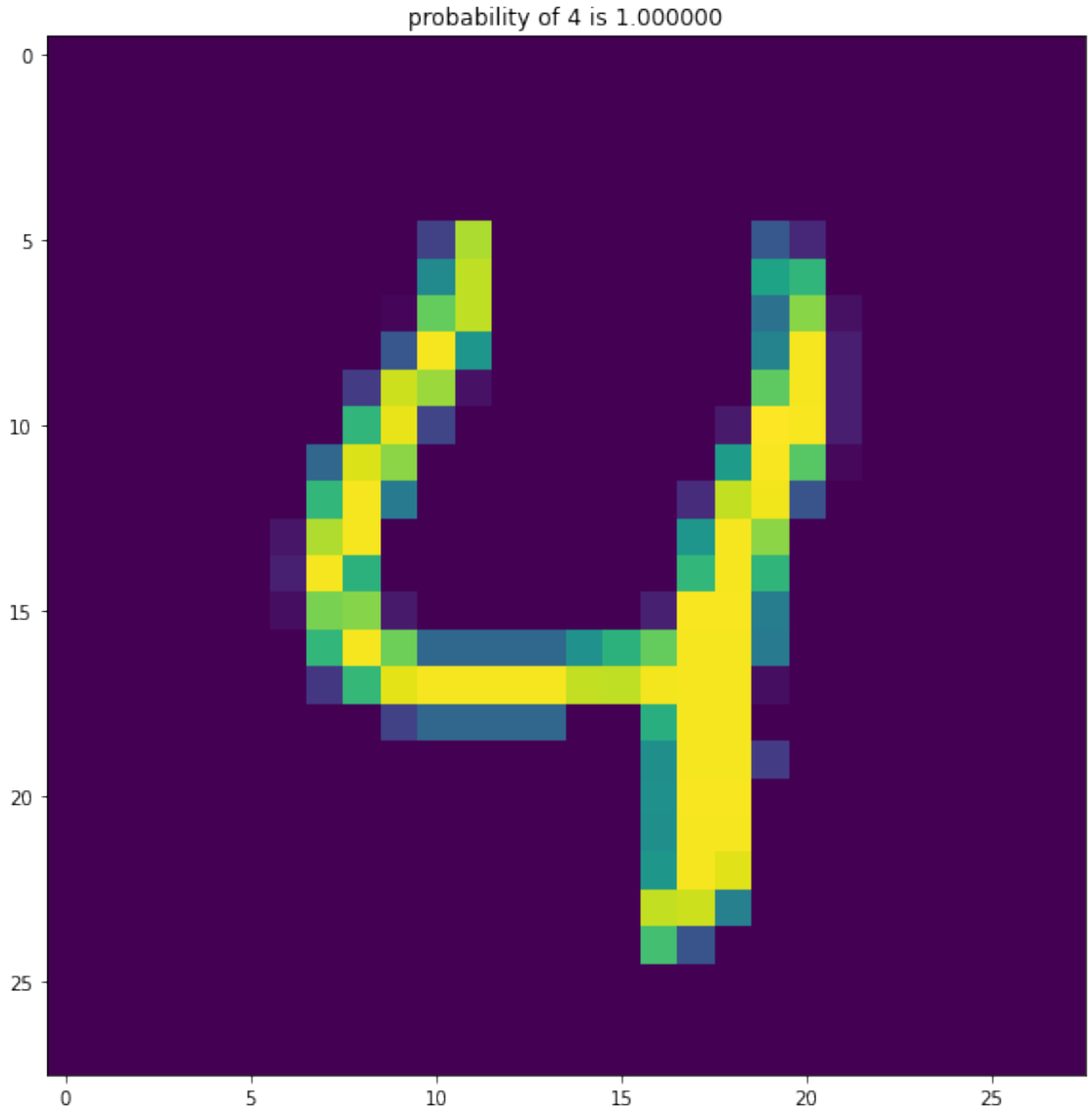

probability of 4 is 1.000000
0
5
10
15
20
25
0
5
10
15
20
25

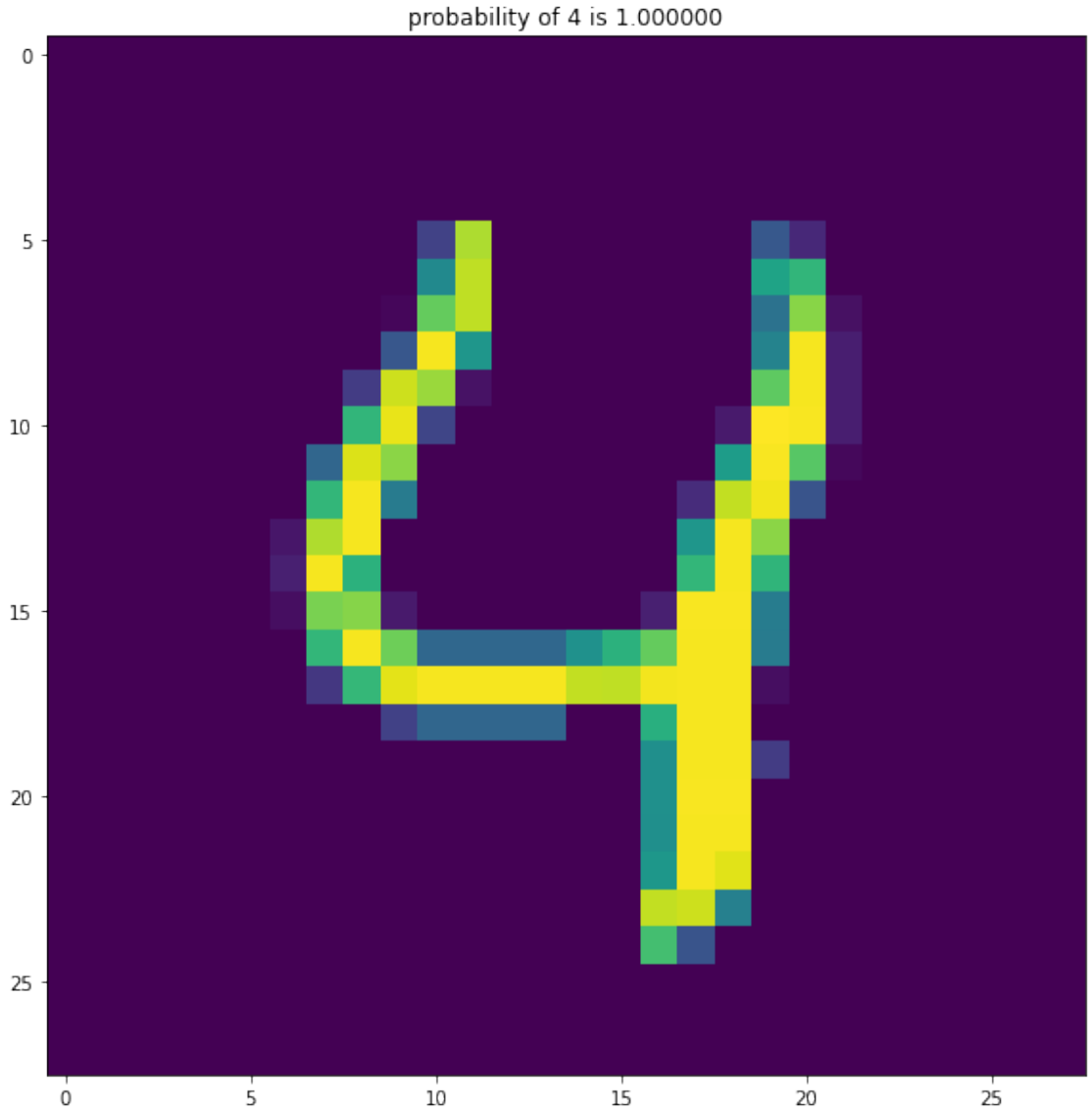

probability of 4 is 1.000000
0
5
10
15
20
25
0
5
10
15
20
25

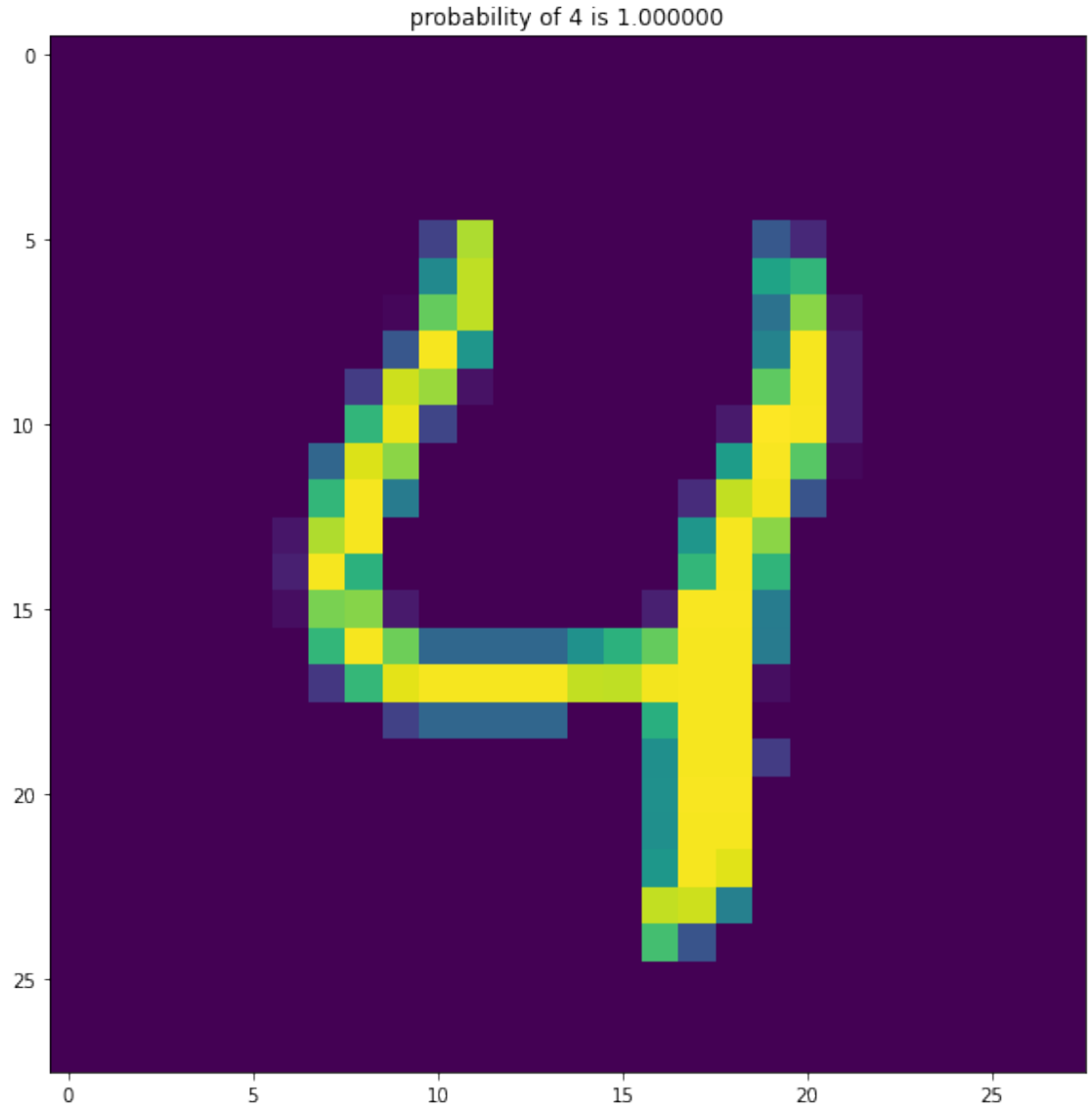

probability of 4 is 1.000000
0
5
10
15
20
25
0
5
10
15
20
25


probability of 4 is 1.000000
0
5
10
15
20
25
0
5
10
15
20
25


probability of 4 is 1.000000
0
5
10
15
20
25
0
5
10
15
20
25


probability of 4 is 1.000000
0
5
10
15
20
25
0
5
10
15
20
25


probability of 4 is 1.000000
0
5
10
15
20
25
0
5
10
15
20
25




```
Probability of 4 as patch is move is :
[[1. 1. 0. 0. 0. 0. 0.]
 [1. 1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1.]]

Maximum probable class as the patch is moved is :
[[4. 4. 5. 1. 5. 7. 7.]
 [4. 4. 4. 4. 4. 4. 4.]
```

```
[4. 4. 4. 4. 4. 4. 4.]
[4. 4. 4. 4. 4. 4. 4.]
[4. 4. 4. 4. 4. 4. 4.]
[4. 4. 4. 4. 4. 4. 4.]
[4. 4. 4. 4. 4. 4. 4.]]
```

17.3 *Observations:*

- For the patches at the edges the model is able to make right predictions whereas if the patch lies around the center of the image model gives wrong predicted class

---

18 Adversarial Examples

19 Non - Targetted Attack

[ ]:
```
index = int(input("\n Input the number for which you want to generate the non␣
 ↪targetted adversarial image\t"))
gaussian_noise = np.random.normal(loc = 128, scale = 10, size = (28,28))
if(device==torch.device("cuda")):
  gaussian_noise_ = torch.from_numpy(gaussian_noise).reshape(1,1,28,28).cuda().
 ↪float()
else:
  gaussian_noise_ = torch.from_numpy(gaussian_noise).reshape(1,1,28,28).float()
logit_values = []
for i in range(non_targetted_n):
  gaussian_noise_ = torch.autograd.Variable(gaussian_noise_, requires_grad =␣
 ↪True)
  out = model.forward(gaussian_noise_, softmax = False)
  loss = out[:, index]
  loss_index = loss.cpu().detach().numpy()
  logit_values.append(loss_index)
  if (i%100==0):
    print("Adversarial Image of number: {}\t  : For Iteration: {}\tLogit value:␣
 ↪{}".format(index, i, loss_index))
  loss.backward(retain_graph = True)
  d = torch.sign(gaussian_noise_.grad.data)
  gaussian_noise_ = gaussian_noise_ + non_targetted_step_size*d
plt.plot(np.asfarray(logit_values))
plt.title(f"Cost function for {index}")
plt.show()

print("Adversarial Image of number: {}\t  : For Iteration: {}\tLogit value: {}".
 ↪format(index, i, loss_index))
```

```
noise_kernel = gaussian_noise_.cpu().reshape(28,28).detach().numpy()
noise_kernel = noise_kernel - noise_kernel.min()
noise_kernel = noise_kernel/noise_kernel.max()
plt.imshow(noise_kernel)
plt.colorbar()
plt.title(f"Non-Targetted Adversarial Image gernerated for {i}")
plt.show()
```

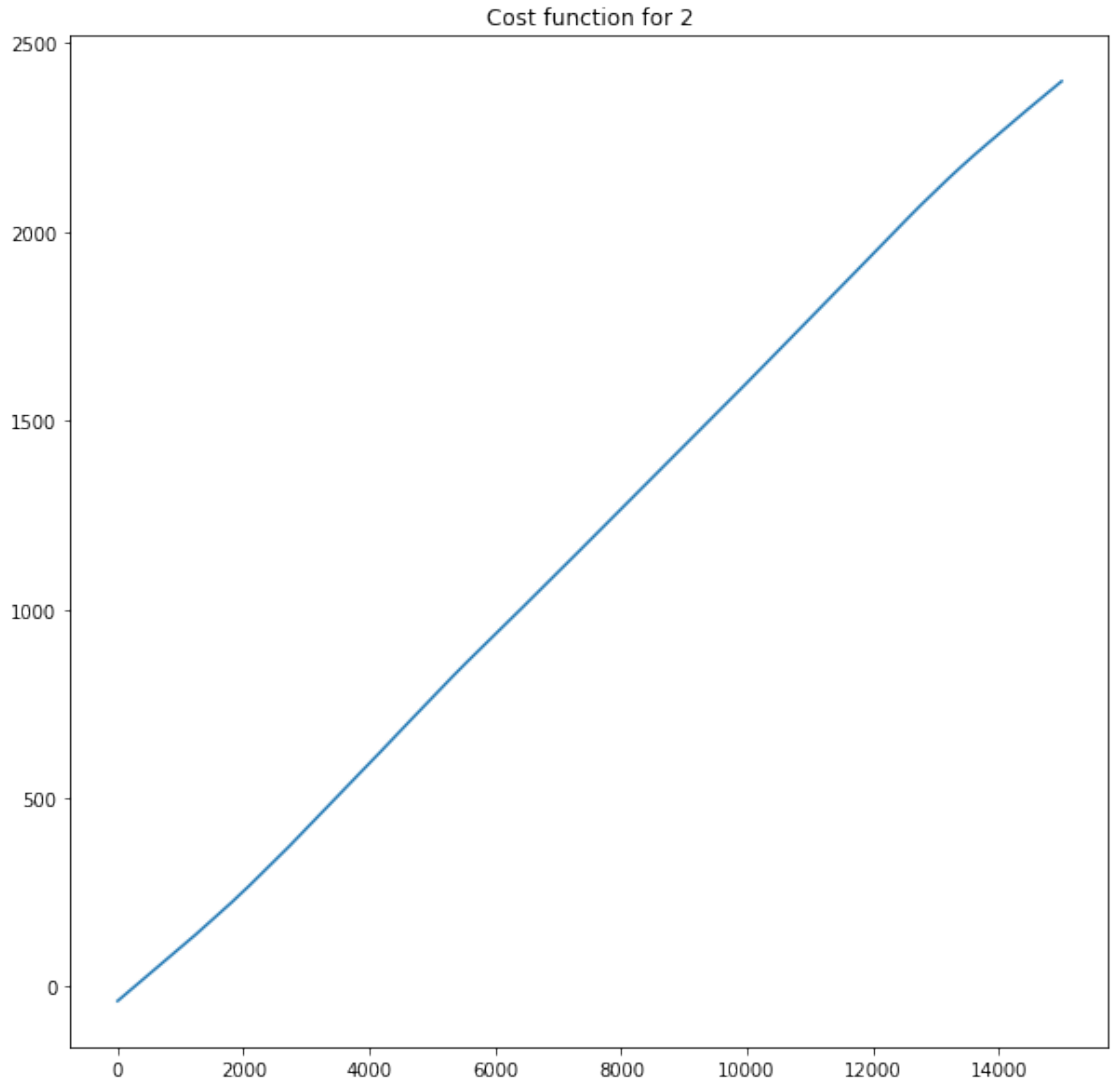
```
 Input the number for which you want to generate the non targetted adversarial
image    2
Adversarial Image of number: 2    : For Iteration: 0    Logit value: [-38.10905]
Adversarial Image of number: 2    : For Iteration: 100  Logit value:
[-23.902775]
Adversarial Image of number: 2    : For Iteration: 200  Logit value: [-9.716741]
Adversarial Image of number: 2    : For Iteration: 300  Logit value: [4.215282]
Adversarial Image of number: 2    : For Iteration: 400  Logit value: [17.900095]
Adversarial Image of number: 2    : For Iteration: 500  Logit value: [32.006042]
Adversarial Image of number: 2    : For Iteration: 600  Logit value: [46.26215]
Adversarial Image of number: 2    : For Iteration: 700  Logit value: [60.40389]
Adversarial Image of number: 2    : For Iteration: 800  Logit value: [74.6654]
Adversarial Image of number: 2    : For Iteration: 900  Logit value: [88.92104]
Adversarial Image of number: 2    : For Iteration: 1000 Logit value: [103.25814]
Adversarial Image of number: 2    : For Iteration: 1100 Logit value: [117.65619]
Adversarial Image of number: 2    : For Iteration: 1200 Logit value: [131.97504]
Adversarial Image of number: 2    : For Iteration: 1300 Logit value: [146.67616]
Adversarial Image of number: 2    : For Iteration: 1400 Logit value: [161.44267]
Adversarial Image of number: 2    : For Iteration: 1500 Logit value: [176.2153]
Adversarial Image of number: 2    : For Iteration: 1600 Logit value: [191.1542]
Adversarial Image of number: 2    : For Iteration: 1700 Logit value: [206.23053]
Adversarial Image of number: 2    : For Iteration: 1800 Logit value: [221.48038]
Adversarial Image of number: 2    : For Iteration: 1900 Logit value: [236.97792]
Adversarial Image of number: 2    : For Iteration: 2000 Logit value: [252.66455]
Adversarial Image of number: 2    : For Iteration: 2100 Logit value: [268.63437]
Adversarial Image of number: 2    : For Iteration: 2200 Logit value: [284.8304]
Adversarial Image of number: 2    : For Iteration: 2300 Logit value: [301.20212]
Adversarial Image of number: 2    : For Iteration: 2400 Logit value: [317.70517]
Adversarial Image of number: 2    : For Iteration: 2500 Logit value: [333.96118]
Adversarial Image of number: 2    : For Iteration: 2600 Logit value: [350.2793]
Adversarial Image of number: 2    : For Iteration: 2700 Logit value: [367.25452]
Adversarial Image of number: 2    : For Iteration: 2800 Logit value: [384.31528]
Adversarial Image of number: 2    : For Iteration: 2900 Logit value: [401.50894]
Adversarial Image of number: 2    : For Iteration: 3000 Logit value: [418.64432]
Adversarial Image of number: 2    : For Iteration: 3100 Logit value: [435.8132]
Adversarial Image of number: 2    : For Iteration: 3200 Logit value: [453.25522]
Adversarial Image of number: 2    : For Iteration: 3300 Logit value: [470.81445]
Adversarial Image of number: 2    : For Iteration: 3400 Logit value: [488.04984]
```

```
Adversarial Image of number: 2    : For Iteration: 3500 Logit value: [505.14438]
Adversarial Image of number: 2    : For Iteration: 3600 Logit value: [522.17834]
Adversarial Image of number: 2    : For Iteration: 3700 Logit value: [539.2304]
Adversarial Image of number: 2    : For Iteration: 3800 Logit value: [556.4044]
Adversarial Image of number: 2    : For Iteration: 3900 Logit value: [573.7238]
Adversarial Image of number: 2    : For Iteration: 4000 Logit value: [591.1583]
Adversarial Image of number: 2    : For Iteration: 4100 Logit value: [608.6943]
Adversarial Image of number: 2    : For Iteration: 4200 Logit value: [626.3247]
Adversarial Image of number: 2    : For Iteration: 4300 Logit value: [643.8217]
Adversarial Image of number: 2    : For Iteration: 4400 Logit value: [661.35675]
Adversarial Image of number: 2    : For Iteration: 4500 Logit value: [678.84576]
Adversarial Image of number: 2    : For Iteration: 4600 Logit value: [696.38837]
Adversarial Image of number: 2    : For Iteration: 4700 Logit value: [713.8523]
Adversarial Image of number: 2    : For Iteration: 4800 Logit value: [731.31213]
Adversarial Image of number: 2    : For Iteration: 4900 Logit value: [748.7936]
Adversarial Image of number: 2    : For Iteration: 5000 Logit value: [766.17584]
Adversarial Image of number: 2    : For Iteration: 5100 Logit value: [783.5324]
Adversarial Image of number: 2    : For Iteration: 5200 Logit value: [800.7908]
Adversarial Image of number: 2    : For Iteration: 5300 Logit value: [817.8989]
Adversarial Image of number: 2    : For Iteration: 5400 Logit value: [834.90753]
Adversarial Image of number: 2    : For Iteration: 5500 Logit value: [851.664]
Adversarial Image of number: 2    : For Iteration: 5600 Logit value: [868.3288]
Adversarial Image of number: 2    : For Iteration: 5700 Logit value: [884.7343]
Adversarial Image of number: 2    : For Iteration: 5800 Logit value: [900.97833]
Adversarial Image of number: 2    : For Iteration: 5900 Logit value: [917.2064]
Adversarial Image of number: 2    : For Iteration: 6000 Logit value: [933.4102]
Adversarial Image of number: 2    : For Iteration: 6100 Logit value: [949.7841]
Adversarial Image of number: 2    : For Iteration: 6200 Logit value: [966.2858]
Adversarial Image of number: 2    : For Iteration: 6300 Logit value: [982.8173]
Adversarial Image of number: 2    : For Iteration: 6400 Logit value: [999.3483]
Adversarial Image of number: 2    : For Iteration: 6500 Logit value:
[1015.89667]
Adversarial Image of number: 2    : For Iteration: 6600 Logit value: [1032.4503]
Adversarial Image of number: 2    : For Iteration: 6700 Logit value: [1048.8296]
Adversarial Image of number: 2    : For Iteration: 6800 Logit value: [1065.4583]
Adversarial Image of number: 2    : For Iteration: 6900 Logit value: [1082.1415]
Adversarial Image of number: 2    : For Iteration: 7000 Logit value: [1098.8428]
Adversarial Image of number: 2    : For Iteration: 7100 Logit value: [1115.4437]
Adversarial Image of number: 2    : For Iteration: 7200 Logit value: [1132.0876]
Adversarial Image of number: 2    : For Iteration: 7300 Logit value: [1148.744]
Adversarial Image of number: 2    : For Iteration: 7400 Logit value: [1165.369]
Adversarial Image of number: 2    : For Iteration: 7500 Logit value: [1182.0908]
Adversarial Image of number: 2    : For Iteration: 7600 Logit value: [1198.8019]
Adversarial Image of number: 2    : For Iteration: 7700 Logit value: [1215.5239]
Adversarial Image of number: 2    : For Iteration: 7800 Logit value: [1232.2323]
Adversarial Image of number: 2    : For Iteration: 7900 Logit value: [1248.9297]
Adversarial Image of number: 2    : For Iteration: 8000 Logit value: [1265.6167]
Adversarial Image of number: 2    : For Iteration: 8100 Logit value: [1282.3741]
```

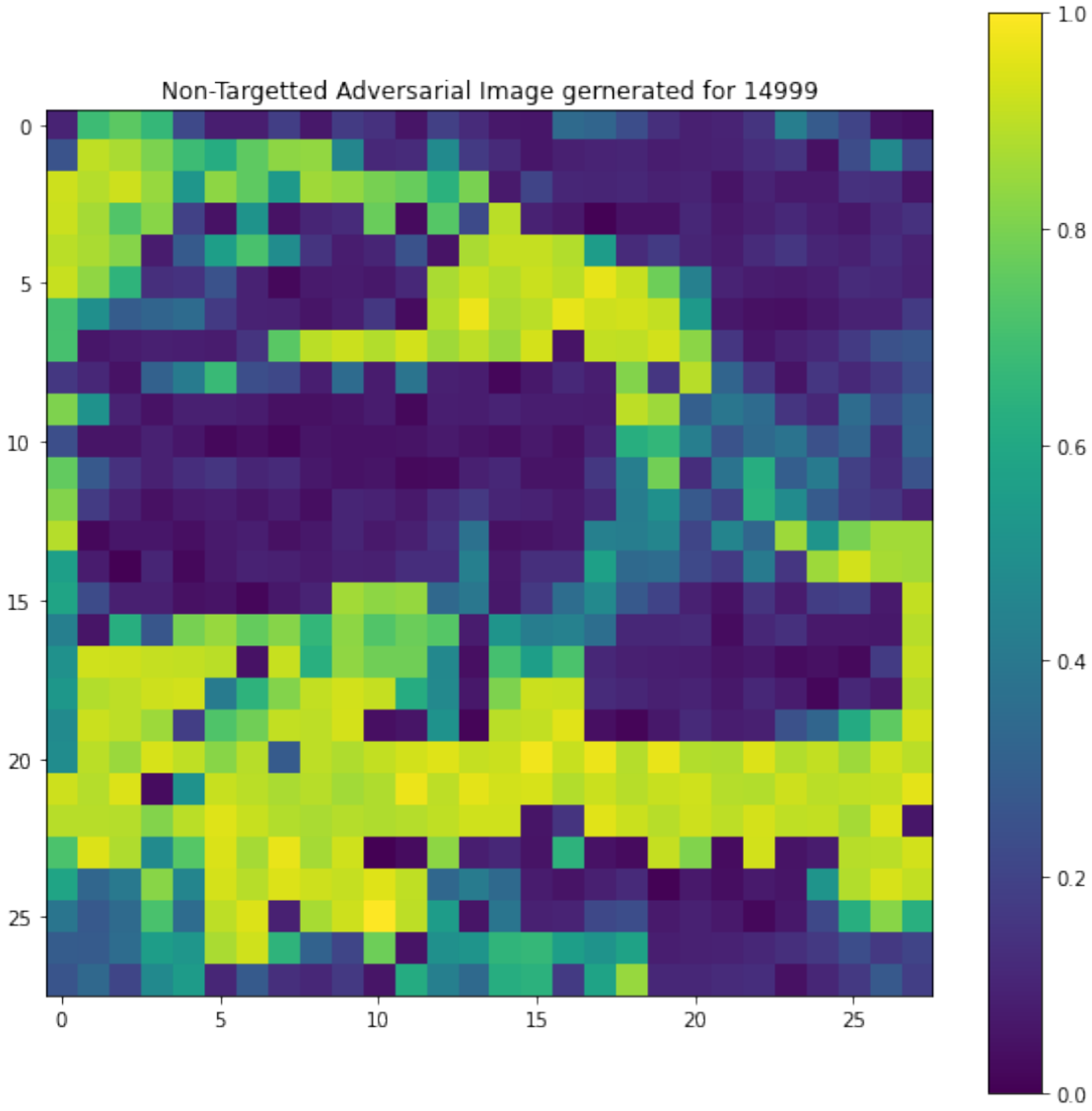
```
Adversarial Image of number: 2    : For Iteration: 8200 Logit value: [1299.1537]
Adversarial Image of number: 2    : For Iteration: 8300 Logit value: [1315.8593]
Adversarial Image of number: 2    : For Iteration: 8400 Logit value: [1332.4738]
Adversarial Image of number: 2    : For Iteration: 8500 Logit value: [1349.1443]
Adversarial Image of number: 2    : For Iteration: 8600 Logit value: [1365.819]
Adversarial Image of number: 2    : For Iteration: 8700 Logit value: [1382.4961]
Adversarial Image of number: 2    : For Iteration: 8800 Logit value: [1399.188]
Adversarial Image of number: 2    : For Iteration: 8900 Logit value: [1415.8092]
Adversarial Image of number: 2    : For Iteration: 9000 Logit value: [1432.4487]
Adversarial Image of number: 2    : For Iteration: 9100 Logit value: [1449.1019]
Adversarial Image of number: 2    : For Iteration: 9200 Logit value: [1465.7426]
Adversarial Image of number: 2    : For Iteration: 9300 Logit value: [1482.3683]
Adversarial Image of number: 2    : For Iteration: 9400 Logit value: [1499.0144]
Adversarial Image of number: 2    : For Iteration: 9500 Logit value: [1515.7329]
Adversarial Image of number: 2    : For Iteration: 9600 Logit value: [1532.6]
Adversarial Image of number: 2    : For Iteration: 9700 Logit value: [1549.5641]
Adversarial Image of number: 2    : For Iteration: 9800 Logit value: [1566.52]
Adversarial Image of number: 2    : For Iteration: 9900 Logit value: [1583.4691]
Adversarial Image of number: 2    : For Iteration: 10000        Logit value:
[1600.4056]
Adversarial Image of number: 2    : For Iteration: 10100        Logit value:
[1617.3842]
Adversarial Image of number: 2    : For Iteration: 10200        Logit value:
[1634.3678]
Adversarial Image of number: 2    : For Iteration: 10300        Logit value:
[1651.3531]
Adversarial Image of number: 2    : For Iteration: 10400        Logit value:
[1668.4207]
Adversarial Image of number: 2    : For Iteration: 10500        Logit value:
[1685.463]
Adversarial Image of number: 2    : For Iteration: 10600        Logit value:
[1702.5176]
Adversarial Image of number: 2    : For Iteration: 10700        Logit value:
[1719.5729]
Adversarial Image of number: 2    : For Iteration: 10800        Logit value:
[1736.6105]
Adversarial Image of number: 2    : For Iteration: 10900        Logit value:
[1753.6605]
Adversarial Image of number: 2    : For Iteration: 11000        Logit value:
[1770.7472]
Adversarial Image of number: 2    : For Iteration: 11100        Logit value:
[1787.7692]
Adversarial Image of number: 2    : For Iteration: 11200        Logit value:
[1804.8456]
Adversarial Image of number: 2    : For Iteration: 11300        Logit value:
[1821.9634]
Adversarial Image of number: 2    : For Iteration: 11400        Logit value:
[1838.98]
```

```
Adversarial Image of number: 2    : For Iteration: 11500        Logit value:
[1855.9265]
Adversarial Image of number: 2    : For Iteration: 11600        Logit value:
[1872.9681]
Adversarial Image of number: 2    : For Iteration: 11700        Logit value:
[1890.0127]
Adversarial Image of number: 2    : For Iteration: 11800        Logit value:
[1907.1073]
Adversarial Image of number: 2    : For Iteration: 11900        Logit value:
[1924.1901]
Adversarial Image of number: 2    : For Iteration: 12000        Logit value:
[1941.2793]
Adversarial Image of number: 2    : For Iteration: 12100        Logit value:
[1958.3485]
Adversarial Image of number: 2    : For Iteration: 12200        Logit value:
[1975.4581]
Adversarial Image of number: 2    : For Iteration: 12300        Logit value:
[1992.6053]
Adversarial Image of number: 2    : For Iteration: 12400        Logit value:
[2009.7645]
Adversarial Image of number: 2    : For Iteration: 12500        Logit value:
[2026.9603]
Adversarial Image of number: 2    : For Iteration: 12600        Logit value:
[2043.9894]
Adversarial Image of number: 2    : For Iteration: 12700        Logit value:
[2060.759]
Adversarial Image of number: 2    : For Iteration: 12800        Logit value:
[2077.3518]
Adversarial Image of number: 2    : For Iteration: 12900        Logit value:
[2093.7607]
Adversarial Image of number: 2    : For Iteration: 13000        Logit value:
[2109.8857]
Adversarial Image of number: 2    : For Iteration: 13100        Logit value:
[2125.804]
Adversarial Image of number: 2    : For Iteration: 13200        Logit value:
[2141.5461]
Adversarial Image of number: 2    : For Iteration: 13300        Logit value:
[2157.0789]
Adversarial Image of number: 2    : For Iteration: 13400        Logit value:
[2172.361]
Adversarial Image of number: 2    : For Iteration: 13500        Logit value:
[2187.468]
Adversarial Image of number: 2    : For Iteration: 13600        Logit value:
[2202.2654]
Adversarial Image of number: 2    : For Iteration: 13700        Logit value:
[2216.7478]
Adversarial Image of number: 2    : For Iteration: 13800        Logit value:
[2231.1494]
```

```
Adversarial Image of number: 2    : For Iteration: 13900        Logit value:
[2245.4944]
Adversarial Image of number: 2    : For Iteration: 14000        Logit value:
[2259.7761]
Adversarial Image of number: 2    : For Iteration: 14100        Logit value:
[2274.0222]
Adversarial Image of number: 2    : For Iteration: 14200        Logit value:
[2288.1853]
Adversarial Image of number: 2    : For Iteration: 14300        Logit value:
[2302.2708]
Adversarial Image of number: 2    : For Iteration: 14400        Logit value:
[2316.3354]
Adversarial Image of number: 2    : For Iteration: 14500        Logit value:
[2330.3577]
Adversarial Image of number: 2    : For Iteration: 14600        Logit value:
[2344.2625]
Adversarial Image of number: 2    : For Iteration: 14700        Logit value:
[2358.0461]
Adversarial Image of number: 2    : For Iteration: 14800        Logit value:
[2371.8577]
Adversarial Image of number: 2    : For Iteration: 14900        Logit value:
[2385.6401]
```




```
Adversarial Image of number: 2    : For Iteration: 14999        Logit value:
[2399.2395]
```



19.1 *Observations:*

- The cost function is always increasing
- The Adversarial Image doesnot compllletely look like the original image (2) but still somehow it is a bit realisable.
- For other images also this snippet was run, and it provided similar observations

19.2 Targetted Attack

[ ]:
```
look_like = int(input("Input the number you want to generate image to look␣
 ↪like\t"))
classify_as = int(input("Input the number you want the original number to be␣
 ↪classified as\t "))
if(device==torch.device("cuda")):
  image = test_loader.dataset.data[indices[look_like], :, :].clone().
 ↪reshape(1,1,28,28).cuda().float()
else:
  image = test_loader.dataset.data[indices[look_like], :, :].clone().
 ↪reshape(1,1,28,28).float()
gaussian_noise = np.random.normal(loc = 128, scale = 10, size = (28,28))
if(device==torch.device("cuda")):
  gaussian_noise_ = torch.from_numpy(gaussian_noise).reshape(1,1,28,28).cuda().
 ↪float()
else:
  gaussian_noise_ = torch.from_numpy(gaussian_noise).reshape(1,1,28,28).float()
for i in range(targetted_n):
  gaussian_noise_ = torch.autograd.Variable(gaussian_noise_, requires_grad =␣
 ↪True)
  out = model.forward(gaussian_noise_, softmax = False)
  probab = F.softmax(out, dim = 1)
  to_be_predicted_prob = probab[:,classify_as].cpu().detach().numpy()
  logit_value = out[:, classify_as]
  mse_error = F.mse_loss(gaussian_noise_, image)
  mse_error_ = mse_error.cpu().detach().numpy()
  loss = logit_value - beta*mse_error
  if (i%20==0):
    print("Iteration: {}\t Number: {}\t Classified with probability: {}\t MSE:␣
 ↪{}\n".format(i, look_like, to_be_predicted_prob, mse_error_))
  loss.backward(retain_graph = True)
  d = torch.sign(gaussian_noise_.grad.data)
  gaussian_noise_ = gaussian_noise_ + non_targetted_step_size*d
print("Iteration: {}\t Number: {}\t Classified with probability: {}\t MSE:␣
 ↪{}\n".format(i, look_like, to_be_predicted_prob, mse_error_))
classified_img = gaussian_noise_.cpu().reshape(28,28).detach().numpy()
classified_img = classified_img - classified_img.min()
classified_img = classified_img/classified_img.max()
f, ax = plt.subplots(1,2)
ax[0].imshow(image.cpu().reshape(28,28).numpy())
ax[0].set_title(f"Target Image of {look_like}")
ax[1].imshow(classified_img)
ax[1].set_title(f"Generated image of {look_like} classified as {classify_as}")
plt.show()
```

```
Input the number you want to generate image to look like        3
```

```
Input the number you want the original number to be classified as        6
Iteration: 0     Number: 3       Classified with probability: [0.]       MSE:
15350.0126953125

Iteration: 20    Number: 3       Classified with probability: [0.]       MSE:
15345.30078125

Iteration: 40    Number: 3       Classified with probability: [0.]       MSE:
15340.4697265625

Iteration: 60    Number: 3       Classified with probability: [0.]       MSE:
15336.3056640625

Iteration: 80    Number: 3       Classified with probability: [0.]       MSE:
15332.5966796875

Iteration: 100   Number: 3       Classified with probability: [0.]       MSE:
15328.9609375

Iteration: 120   Number: 3       Classified with probability: [0.]       MSE:
15324.677734375

Iteration: 140   Number: 3       Classified with probability: [0.]       MSE:
15320.29296875

Iteration: 160   Number: 3       Classified with probability: [0.]       MSE:
15316.3662109375

Iteration: 180   Number: 3       Classified with probability: [0.]       MSE:
15311.423828125

Iteration: 200   Number: 3       Classified with probability: [0.]       MSE:
15306.5341796875

Iteration: 220   Number: 3       Classified with probability: [0.]       MSE:
15301.1328125

Iteration: 240   Number: 3       Classified with probability: [0.]       MSE:
15296.1083984375

Iteration: 260   Number: 3       Classified with probability: [0.]       MSE:
15290.966796875

Iteration: 280   Number: 3       Classified with probability: [0.]       MSE:
15284.8369140625

Iteration: 300   Number: 3       Classified with probability: [0.]       MSE:
15278.8818359375
```

```
Iteration: 320   Number: 3       Classified with probability: [0.]       MSE:
15273.216796875

Iteration: 340   Number: 3       Classified with probability: [0.]       MSE:
15268.662109375

Iteration: 360   Number: 3       Classified with probability: [0.]       MSE:
15264.642578125

Iteration: 380   Number: 3       Classified with probability: [0.]       MSE:
15260.208984375

Iteration: 400   Number: 3       Classified with probability: [0.]       MSE:
15255.1240234375

Iteration: 420   Number: 3       Classified with probability: [0.]       MSE:
15249.7861328125

Iteration: 440   Number: 3       Classified with probability: [0.]       MSE:
15244.2216796875

Iteration: 460   Number: 3       Classified with probability: [0.]       MSE:
15239.095703125

Iteration: 480   Number: 3       Classified with probability: [0.]       MSE:
15233.3203125

Iteration: 500   Number: 3       Classified with probability: [0.]       MSE:
15227.287109375

Iteration: 520   Number: 3       Classified with probability: [0.]       MSE:
15220.9375

Iteration: 540   Number: 3       Classified with probability: [0.]       MSE:
15214.5380859375

Iteration: 560   Number: 3       Classified with probability: [0.]       MSE:
15208.6640625

Iteration: 580   Number: 3       Classified with probability: [0.]       MSE:
15202.515625

Iteration: 600   Number: 3       Classified with probability: [0.]       MSE:
15196.484375

Iteration: 620   Number: 3       Classified with probability: [0.]       MSE:
15190.099609375
```

```
Iteration: 640   Number: 3       Classified with probability: [0.]       MSE:
15183.4287109375

Iteration: 660   Number: 3       Classified with probability: [0.]       MSE:
15177.16796875

Iteration: 680   Number: 3       Classified with probability: [0.]       MSE:
15171.1875

Iteration: 700   Number: 3       Classified with probability: [0.]       MSE:
15165.4443359375

Iteration: 720   Number: 3       Classified with probability: [0.]       MSE:
15159.607421875

Iteration: 740   Number: 3       Classified with probability: [0.]       MSE:
15153.7880859375

Iteration: 760   Number: 3       Classified with probability: [0.]       MSE:
15148.345703125

Iteration: 780   Number: 3       Classified with probability: [0.]       MSE:
15142.5654296875

Iteration: 800   Number: 3       Classified with probability: [0.]       MSE:
15136.041015625

Iteration: 820   Number: 3       Classified with probability: [0.]       MSE:
15129.14453125

Iteration: 840   Number: 3       Classified with probability: [0.]       MSE:
15122.724609375

Iteration: 860   Number: 3       Classified with probability: [0.]       MSE:
15116.6630859375

Iteration: 880   Number: 3       Classified with probability: [0.]       MSE:
15110.82421875

Iteration: 900   Number: 3       Classified with probability: [0.]       MSE:
15104.724609375

Iteration: 920   Number: 3       Classified with probability: [0.]       MSE:
15098.357421875

Iteration: 940   Number: 3       Classified with probability: [0.]       MSE:
15091.7451171875
```

```
Iteration: 960   Number: 3       Classified with probability: [0.]       MSE:
15085.2705078125

Iteration: 980   Number: 3       Classified with probability: [0.]       MSE:
15079.072265625

Iteration: 1000  Number: 3       Classified with probability: [0.]       MSE:
15072.7744140625

Iteration: 1020  Number: 3       Classified with probability: [0.]       MSE:
15066.724609375

Iteration: 1040  Number: 3       Classified with probability: [0.]       MSE:
15060.9443359375

Iteration: 1060  Number: 3       Classified with probability: [0.]       MSE:
15055.201171875

Iteration: 1080  Number: 3       Classified with probability: [0.]       MSE:
15049.1884765625

Iteration: 1100  Number: 3       Classified with probability: [0.]       MSE:
15043.05859375

Iteration: 1120  Number: 3       Classified with probability: [0.]       MSE:
15036.9072265625

Iteration: 1140  Number: 3       Classified with probability: [0.]       MSE:
15030.388671875

Iteration: 1160  Number: 3       Classified with probability: [0.]       MSE:
15023.9208984375

Iteration: 1180  Number: 3       Classified with probability: [0.]       MSE:
15017.3759765625

Iteration: 1200  Number: 3       Classified with probability: [0.]       MSE:
15010.94140625

Iteration: 1220  Number: 3       Classified with probability: [0.]       MSE:
15004.6552734375

Iteration: 1240  Number: 3       Classified with probability: [0.]       MSE:
14998.3056640625

Iteration: 1260  Number: 3       Classified with probability: [0.]       MSE:
14992.1123046875
```

```
Iteration: 1280  Number: 3       Classified with probability: [0.]       MSE:
14985.853515625

Iteration: 1300  Number: 3       Classified with probability: [0.]       MSE:
14979.6953125

Iteration: 1320  Number: 3       Classified with probability: [0.]       MSE:
14973.4921875

Iteration: 1340  Number: 3       Classified with probability: [0.]       MSE:
14967.4208984375

Iteration: 1360  Number: 3       Classified with probability: [0.]       MSE:
14961.7744140625

Iteration: 1380  Number: 3       Classified with probability: [0.]       MSE:
14956.099609375

Iteration: 1400  Number: 3       Classified with probability: [0.]       MSE:
14949.9248046875

Iteration: 1420  Number: 3       Classified with probability: [0.]       MSE:
14943.5458984375

Iteration: 1440  Number: 3       Classified with probability: [0.]       MSE:
14936.8720703125

Iteration: 1460  Number: 3       Classified with probability: [0.]       MSE:
14930.015625

Iteration: 1480  Number: 3       Classified with probability: [6.e-45]   MSE:
14923.41796875

Iteration: 1500  Number: 3       Classified with probability: [6.4e-44]  MSE:
14917.083984375

Iteration: 1520  Number: 3       Classified with probability: [7.82e-43]
MSE: 14911.0205078125

Iteration: 1540  Number: 3       Classified with probability: [9.792e-42]
MSE: 14905.205078125

Iteration: 1560  Number: 3       Classified with probability: [1.29145e-40]
MSE: 14899.5322265625

Iteration: 1580  Number: 3       Classified with probability: [1.732979e-39]
MSE: 14893.703125
```

```
Iteration: 1600  Number: 3       Classified with probability: [2.299814e-38]
MSE: 14888.052734375

Iteration: 1620  Number: 3       Classified with probability: [2.9768519e-37]
MSE: 14882.521484375

Iteration: 1640  Number: 3       Classified with probability: [3.760498e-36]
MSE: 14877.052734375

Iteration: 1660  Number: 3       Classified with probability: [4.7243408e-35]
MSE: 14871.0048828125

Iteration: 1680  Number: 3       Classified with probability: [5.847755e-34]
MSE: 14865.025390625

Iteration: 1700  Number: 3       Classified with probability: [7.54102e-33]
MSE: 14859.462890625

Iteration: 1720  Number: 3       Classified with probability: [1.0550139e-31]
MSE: 14854.81640625

Iteration: 1740  Number: 3       Classified with probability: [1.4597837e-30]
MSE: 14850.1015625

Iteration: 1760  Number: 3       Classified with probability: [2.0532223e-29]
MSE: 14845.2666015625

Iteration: 1780  Number: 3       Classified with probability: [2.8642108e-28]
MSE: 14840.259765625

Iteration: 1800  Number: 3       Classified with probability: [4.090465e-27]
MSE: 14835.263671875

Iteration: 1820  Number: 3       Classified with probability: [6.0899205e-26]
MSE: 14830.5380859375

Iteration: 1840  Number: 3       Classified with probability: [9.194063e-25]
MSE: 14825.8408203125

Iteration: 1860  Number: 3       Classified with probability: [1.3915011e-23]
MSE: 14821.234375

Iteration: 1880  Number: 3       Classified with probability: [2.1142486e-22]
MSE: 14816.2294921875

Iteration: 1900  Number: 3       Classified with probability: [3.2386247e-21]
MSE: 14811.1953125
```

```
Iteration: 1920  Number: 3       Classified with probability: [4.9481776e-20]
MSE: 14806.2041015625

Iteration: 1940  Number: 3       Classified with probability: [7.5776966e-19]
MSE: 14801.099609375

Iteration: 1960  Number: 3       Classified with probability: [1.1680073e-17]
MSE: 14796.1611328125

Iteration: 1980  Number: 3       Classified with probability: [1.7698313e-16]
MSE: 14791.5126953125

Iteration: 2000  Number: 3       Classified with probability: [2.5987644e-15]
MSE: 14786.673828125

Iteration: 2020  Number: 3       Classified with probability: [3.8174582e-14]
MSE: 14781.798828125

Iteration: 2040  Number: 3       Classified with probability: [5.8171805e-13]
MSE: 14776.9873046875

Iteration: 2060  Number: 3       Classified with probability: [8.755142e-12]
MSE: 14772.4462890625

Iteration: 2080  Number: 3       Classified with probability: [1.1767816e-10]
MSE: 14767.9765625

Iteration: 2100  Number: 3       Classified with probability: [1.5703416e-09]
MSE: 14763.6923828125

Iteration: 2120  Number: 3       Classified with probability: [1.9813388e-08]
MSE: 14759.6181640625

Iteration: 2140  Number: 3       Classified with probability: [2.4227307e-07]
MSE: 14755.35546875

Iteration: 2160  Number: 3       Classified with probability: [2.9555833e-06]
MSE: 14751.3623046875

Iteration: 2180  Number: 3       Classified with probability: [3.5680332e-05]
MSE: 14747.7529296875

Iteration: 2200  Number: 3       Classified with probability: [0.00042684]
MSE: 14744.1591796875

Iteration: 2220  Number: 3       Classified with probability: [0.00492496]
MSE: 14740.7431640625
```

```
Iteration: 2240  Number: 3       Classified with probability: [0.05046939]
MSE: 14737.974609375

Iteration: 2260  Number: 3       Classified with probability: [0.34564346]
MSE: 14735.4501953125

Iteration: 2280  Number: 3       Classified with probability: [0.84679353]
MSE: 14732.958984375

Iteration: 2300  Number: 3       Classified with probability: [0.9841053]
MSE: 14730.533203125

Iteration: 2320  Number: 3       Classified with probability: [0.9985868]
MSE: 14728.115234375

Iteration: 2340  Number: 3       Classified with probability: [0.9998797]
MSE: 14725.6220703125

Iteration: 2360  Number: 3       Classified with probability: [0.99998975]
MSE: 14723.2998046875

Iteration: 2380  Number: 3       Classified with probability: [0.99999917]
MSE: 14721.025390625

Iteration: 2400  Number: 3       Classified with probability: [0.9999999]
MSE: 14718.60546875

Iteration: 2420  Number: 3       Classified with probability: [1.]       MSE:
14716.486328125

Iteration: 2440  Number: 3       Classified with probability: [1.]       MSE:
14714.2861328125

Iteration: 2460  Number: 3       Classified with probability: [1.]       MSE:
14712.177734375

Iteration: 2480  Number: 3       Classified with probability: [1.]       MSE:
14709.8623046875

Iteration: 2500  Number: 3       Classified with probability: [1.]       MSE:
14707.6220703125

Iteration: 2520  Number: 3       Classified with probability: [1.]       MSE:
14705.66796875

Iteration: 2540  Number: 3       Classified with probability: [1.]       MSE:
14703.94140625
```

```
Iteration: 2560  Number: 3       Classified with probability: [1.]       MSE:
14702.1943359375

Iteration: 2580  Number: 3       Classified with probability: [1.]       MSE:
14700.486328125

Iteration: 2600  Number: 3       Classified with probability: [1.]       MSE:
14699.0380859375

Iteration: 2620  Number: 3       Classified with probability: [1.]       MSE:
14697.587890625

Iteration: 2640  Number: 3       Classified with probability: [1.]       MSE:
14696.1923828125

Iteration: 2660  Number: 3       Classified with probability: [1.]       MSE:
14694.8544921875

Iteration: 2680  Number: 3       Classified with probability: [1.]       MSE:
14693.736328125

Iteration: 2700  Number: 3       Classified with probability: [1.]       MSE:
14692.521484375

Iteration: 2720  Number: 3       Classified with probability: [1.]       MSE:
14691.69140625

Iteration: 2740  Number: 3       Classified with probability: [1.]       MSE:
14690.9833984375

Iteration: 2760  Number: 3       Classified with probability: [1.]       MSE:
14690.328125

Iteration: 2780  Number: 3       Classified with probability: [1.]       MSE:
14689.67578125

Iteration: 2800  Number: 3       Classified with probability: [1.]       MSE:
14688.923828125

Iteration: 2820  Number: 3       Classified with probability: [1.]       MSE:
14688.17578125

Iteration: 2840  Number: 3       Classified with probability: [1.]       MSE:
14687.595703125

Iteration: 2860  Number: 3       Classified with probability: [1.]       MSE:
14687.5302734375
```

```
Iteration: 2880  Number: 3       Classified with probability: [1.]       MSE:
14687.701171875

Iteration: 2900  Number: 3       Classified with probability: [1.]       MSE:
14687.7470703125

Iteration: 2920  Number: 3       Classified with probability: [1.]       MSE:
14688.103515625

Iteration: 2940  Number: 3       Classified with probability: [1.]       MSE:
14688.67578125

Iteration: 2960  Number: 3       Classified with probability: [1.]       MSE:
14689.1708984375

Iteration: 2980  Number: 3       Classified with probability: [1.]       MSE:
14689.9501953125

Iteration: 3000  Number: 3       Classified with probability: [1.]       MSE:
14691.0

Iteration: 3020  Number: 3       Classified with probability: [1.]       MSE:
14691.8955078125

Iteration: 3040  Number: 3       Classified with probability: [1.]       MSE:
14692.5390625

Iteration: 3060  Number: 3       Classified with probability: [1.]       MSE:
14693.26171875

Iteration: 3080  Number: 3       Classified with probability: [1.]       MSE:
14694.0927734375

Iteration: 3100  Number: 3       Classified with probability: [1.]       MSE:
14694.99609375

Iteration: 3120  Number: 3       Classified with probability: [1.]       MSE:
14695.6044921875

Iteration: 3140  Number: 3       Classified with probability: [1.]       MSE:
14696.2734375

Iteration: 3160  Number: 3       Classified with probability: [1.]       MSE:
14697.125

Iteration: 3180  Number: 3       Classified with probability: [1.]       MSE:
14697.845703125
```

```
Iteration: 3200  Number: 3       Classified with probability: [1.]       MSE:
14698.388671875

Iteration: 3220  Number: 3       Classified with probability: [1.]       MSE:
14698.783203125

Iteration: 3240  Number: 3       Classified with probability: [1.]       MSE:
14699.6953125

Iteration: 3260  Number: 3       Classified with probability: [1.]       MSE:
14700.8056640625

Iteration: 3280  Number: 3       Classified with probability: [1.]       MSE:
14701.9716796875

Iteration: 3300  Number: 3       Classified with probability: [1.]       MSE:
14703.5234375

Iteration: 3320  Number: 3       Classified with probability: [1.]       MSE:
14705.291015625

Iteration: 3340  Number: 3       Classified with probability: [1.]       MSE:
14707.2265625

Iteration: 3360  Number: 3       Classified with probability: [1.]       MSE:
14709.24609375

Iteration: 3380  Number: 3       Classified with probability: [1.]       MSE:
14711.1328125

Iteration: 3400  Number: 3       Classified with probability: [1.]       MSE:
14713.07421875

Iteration: 3420  Number: 3       Classified with probability: [1.]       MSE:
14715.111328125

Iteration: 3440  Number: 3       Classified with probability: [1.]       MSE:
14717.0498046875

Iteration: 3460  Number: 3       Classified with probability: [1.]       MSE:
14719.1455078125

Iteration: 3480  Number: 3       Classified with probability: [1.]       MSE:
14720.8720703125

Iteration: 3500  Number: 3       Classified with probability: [1.]       MSE:
14722.826171875
```

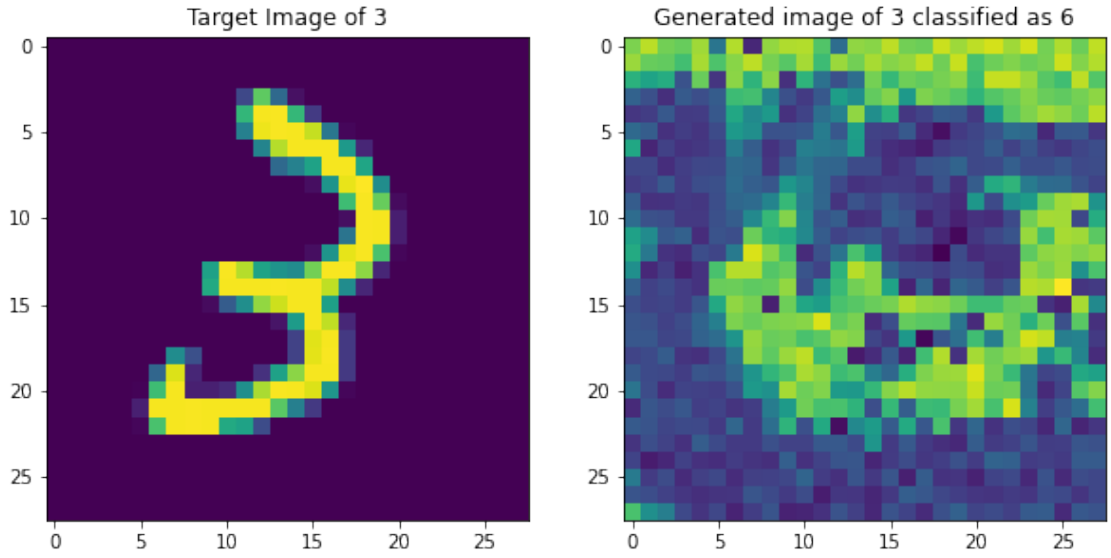
```
Iteration: 3520  Number: 3       Classified with probability: [1.]       MSE:
14724.7734375

Iteration: 3540  Number: 3       Classified with probability: [1.]       MSE:
14726.5048828125

Iteration: 3560  Number: 3       Classified with probability: [1.]       MSE:
14728.1806640625

Iteration: 3580  Number: 3       Classified with probability: [1.]       MSE:
14730.01953125

Iteration: 3600  Number: 3       Classified with probability: [1.]       MSE:
14731.7802734375

Iteration: 3620  Number: 3       Classified with probability: [1.]       MSE:
14733.7138671875

Iteration: 3640  Number: 3       Classified with probability: [1.]       MSE:
14735.759765625

Iteration: 3660  Number: 3       Classified with probability: [1.]       MSE:
14738.041015625

Iteration: 3680  Number: 3       Classified with probability: [1.]       MSE:
14740.8369140625

Iteration: 3700  Number: 3       Classified with probability: [1.]       MSE:
14743.423828125

Iteration: 3720  Number: 3       Classified with probability: [1.]       MSE:
14746.5927734375

Iteration: 3740  Number: 3       Classified with probability: [1.]       MSE:
14749.470703125

Iteration: 3760  Number: 3       Classified with probability: [1.]       MSE:
14752.3369140625

Iteration: 3780  Number: 3       Classified with probability: [1.]       MSE:
14755.658203125

Iteration: 3800  Number: 3       Classified with probability: [1.]       MSE:
14758.6845703125

Iteration: 3820  Number: 3       Classified with probability: [1.]       MSE:
14761.541015625
```

```
Iteration: 3840  Number: 3       Classified with probability: [1.]       MSE:
14764.615234375

Iteration: 3860  Number: 3       Classified with probability: [1.]       MSE:
14767.6181640625

Iteration: 3880  Number: 3       Classified with probability: [1.]       MSE:
14770.654296875

Iteration: 3900  Number: 3       Classified with probability: [1.]       MSE:
14773.7919921875

Iteration: 3920  Number: 3       Classified with probability: [1.]       MSE:
14777.1484375

Iteration: 3940  Number: 3       Classified with probability: [1.]       MSE:
14780.29296875

Iteration: 3960  Number: 3       Classified with probability: [1.]       MSE:
14783.3662109375

Iteration: 3980  Number: 3       Classified with probability: [1.]       MSE:
14786.4208984375

Iteration: 4000  Number: 3       Classified with probability: [1.]       MSE:
14789.677734375

Iteration: 4020  Number: 3       Classified with probability: [1.]       MSE:
14793.1376953125

Iteration: 4040  Number: 3       Classified with probability: [1.]       MSE:
14796.5166015625

Iteration: 4060  Number: 3       Classified with probability: [1.]       MSE:
14799.7587890625

Iteration: 4080  Number: 3       Classified with probability: [1.]       MSE:
14802.9208984375

Iteration: 4100  Number: 3       Classified with probability: [1.]       MSE:
14806.392578125

Iteration: 4120  Number: 3       Classified with probability: [1.]       MSE:
14810.1572265625

Iteration: 4140  Number: 3       Classified with probability: [1.]       MSE:
14814.095703125
```

```
Iteration: 4160  Number: 3       Classified with probability: [1.]       MSE:
14818.4638671875

Iteration: 4180  Number: 3       Classified with probability: [1.]       MSE:
14822.8056640625

Iteration: 4200  Number: 3       Classified with probability: [1.]       MSE:
14827.1328125

Iteration: 4220  Number: 3       Classified with probability: [1.]       MSE:
14831.5966796875

Iteration: 4240  Number: 3       Classified with probability: [1.]       MSE:
14835.890625

Iteration: 4260  Number: 3       Classified with probability: [1.]       MSE:
14840.3251953125

Iteration: 4280  Number: 3       Classified with probability: [1.]       MSE:
14844.62890625

Iteration: 4300  Number: 3       Classified with probability: [1.]       MSE:
14848.85546875

Iteration: 4320  Number: 3       Classified with probability: [1.]       MSE:
14853.25

Iteration: 4340  Number: 3       Classified with probability: [1.]       MSE:
14857.5029296875

Iteration: 4360  Number: 3       Classified with probability: [1.]       MSE:
14861.7490234375

Iteration: 4380  Number: 3       Classified with probability: [1.]       MSE:
14866.046875

Iteration: 4400  Number: 3       Classified with probability: [1.]       MSE:
14870.412109375

Iteration: 4420  Number: 3       Classified with probability: [1.]       MSE:
14874.765625

Iteration: 4440  Number: 3       Classified with probability: [1.]       MSE:
14878.80078125

Iteration: 4460  Number: 3       Classified with probability: [1.]       MSE:
14882.9326171875
```

```
Iteration: 4480  Number: 3       Classified with probability: [1.]       MSE:
14887.154296875

Iteration: 4500  Number: 3       Classified with probability: [1.]       MSE:
14891.5537109375

Iteration: 4520  Number: 3       Classified with probability: [1.]       MSE:
14896.1796875

Iteration: 4540  Number: 3       Classified with probability: [1.]       MSE:
14900.8408203125

Iteration: 4560  Number: 3       Classified with probability: [1.]       MSE:
14905.408203125

Iteration: 4580  Number: 3       Classified with probability: [1.]       MSE:
14909.9580078125

Iteration: 4600  Number: 3       Classified with probability: [1.]       MSE:
14914.4345703125

Iteration: 4620  Number: 3       Classified with probability: [1.]       MSE:
14918.60546875

Iteration: 4640  Number: 3       Classified with probability: [1.]       MSE:
14922.55078125

Iteration: 4660  Number: 3       Classified with probability: [1.]       MSE:
14926.7490234375

Iteration: 4680  Number: 3       Classified with probability: [1.]       MSE:
14931.1962890625

Iteration: 4700  Number: 3       Classified with probability: [1.]       MSE:
14935.44921875

Iteration: 4720  Number: 3       Classified with probability: [1.]       MSE:
14939.2587890625

Iteration: 4740  Number: 3       Classified with probability: [1.]       MSE:
14943.0947265625

Iteration: 4760  Number: 3       Classified with probability: [1.]       MSE:
14947.0166015625

Iteration: 4780  Number: 3       Classified with probability: [1.]       MSE:
14951.2412109375
```

```
Iteration: 4800  Number: 3       Classified with probability: [1.]       MSE:
14955.3671875

Iteration: 4820  Number: 3       Classified with probability: [1.]       MSE:
14959.64453125

Iteration: 4840  Number: 3       Classified with probability: [1.]       MSE:
14963.8759765625

Iteration: 4860  Number: 3       Classified with probability: [1.]       MSE:
14968.162109375

Iteration: 4880  Number: 3       Classified with probability: [1.]       MSE:
14972.6484375

Iteration: 4900  Number: 3       Classified with probability: [1.]       MSE:
14977.0947265625

Iteration: 4920  Number: 3       Classified with probability: [1.]       MSE:
14981.7080078125

Iteration: 4940  Number: 3       Classified with probability: [1.]       MSE:
14986.283203125

Iteration: 4960  Number: 3       Classified with probability: [1.]       MSE:
14990.775390625

Iteration: 4980  Number: 3       Classified with probability: [1.]       MSE:
14995.3701171875

Iteration: 4999  Number: 3       Classified with probability: [1.]       MSE:
14999.6884765625
```



19.3 *Observations:*

- Although, as per classified probability the generated image of 3 is closely classified to 6 in about 2500 iterations but still the output was not good enough and no where was close to 6, so iterations have been increased to 5000 to make a clear observation

19.4 Adding Noise

[ ]:
```
orig_input = int(input("Input the original image class\t"))
target_input = int(input("Input the target image class\t"))
if(device==torch.device("cuda")):
  image = test_loader.dataset.data[indices[orig_input], :, :].clone().
 ↪reshape(1,1,28,28).cuda().float()
else:
  image = test_loader.dataset.data[indices[orig_input], :, :].clone().
 ↪reshape(1,1,28,28).float()
gaussian_noise = np.random.normal(loc = 128, scale = 10, size = (28,28))
if(device==torch.device("cuda")):
  gaussian_noise_ = torch.from_numpy(gaussian_noise).reshape(1,1,28,28).cuda().
 ↪float()
else:
  gaussian_noise_ = torch.from_numpy(gaussian_noise).reshape(1,1,28,28).float()
prob = 0
highest_prob = orig_input
iter = 0
while(highest_prob!=target_input):
```

```
  gaussian_noise_ = torch.autograd.Variable(gaussian_noise_, requires_grad =␣
 ↪True)
  out = model.forward(gaussian_noise_, softmax = False)
  probab = F.softmax(out, dim = 1).cpu().detach().numpy()
  highest_prob = int(np.argmax(probab))
  prob = probab[:, target_input]
  loss = out[:, target_input]
  loss_op = loss.cpu().detach().numpy()
  if (iter%5==0):
    print("Iteration: {}\t target: {}\t probability: {}\t Logit Value: {}".
 ↪format(iter, target_input, prob, loss_op))
  loss.backward(retain_graph = True)
  d = torch.sign(gaussian_noise_.grad.data)
  d = d - d.min()
  d = d/d.max()
  gaussian_noise_ = gaussian_noise_ + alpha*d
  iter = iter+1

print("Iteration: {}\t target: {}\t probability: {}\t Logit Value: {}".
 ↪format(iter, target_input, prob, loss_op))
noisy_image = (gaussian_noise_ + image).cpu().reshape(28,28).detach().numpy()
noisy_image = noisy_image - noisy_image.min()
noisy_image = noisy_image/noisy_image.max()
f, ax = plt.subplots(1,2)
ax[0].imshow(image.cpu().reshape(28,28).numpy())
ax[0].set_title(f"Original Image of {orig_input}")
ax[1].imshow(noisy_image)
ax[1].set_title(f"noisy image of {orig_input} classified as {target_input}")
plt.show()

new_input = int(input("\nInput the number to which you want to add noise\t"))
if(device==torch.device("cuda")):
  image = test_loader.dataset.data[indices[new_input], :, :].clone().
 ↪reshape(1,1,28,28).cuda().float()
else:
  new_image = test_loader.dataset.data[indices[new_input], :, :].clone().
 ↪reshape(1,1,28,28).float()
out = model.forward(gaussian_noise_ + new_image, softmax = False)
probab = F.softmax(out, dim = 1).cpu().detach().numpy()
highest_prob = int(np.argmax(probab))
prob = probab[:, highest_prob]

noisy_image = (gaussian_noise_ + new_image).cpu().reshape(28,28).detach().
 ↪numpy()
noisy_image = noisy_image - noisy_image.min()
noisy_image = noisy_image/noisy_image.max()
```

```
f, ax = plt.subplots(1,2)
ax[0].imshow(new_image.cpu().reshape(28,28).numpy())
ax[0].set_title(f"Original Image of {new_input}")
ax[1].imshow(noisy_image)
ax[1].set_title(f"noisy image of {new_input} classified as {highest_prob}")
plt.show()

#Displaying the Gaussian Noise
gaussian = (gaussian_noise_).cpu().reshape(28,28).detach().numpy()
gaussian = gaussian - np.min(gaussian)
gaussian = gaussian/np.max(gaussian)
plt.imshow(gaussian)
plt.title(f"Gaussian Noise")
plt.show()
```

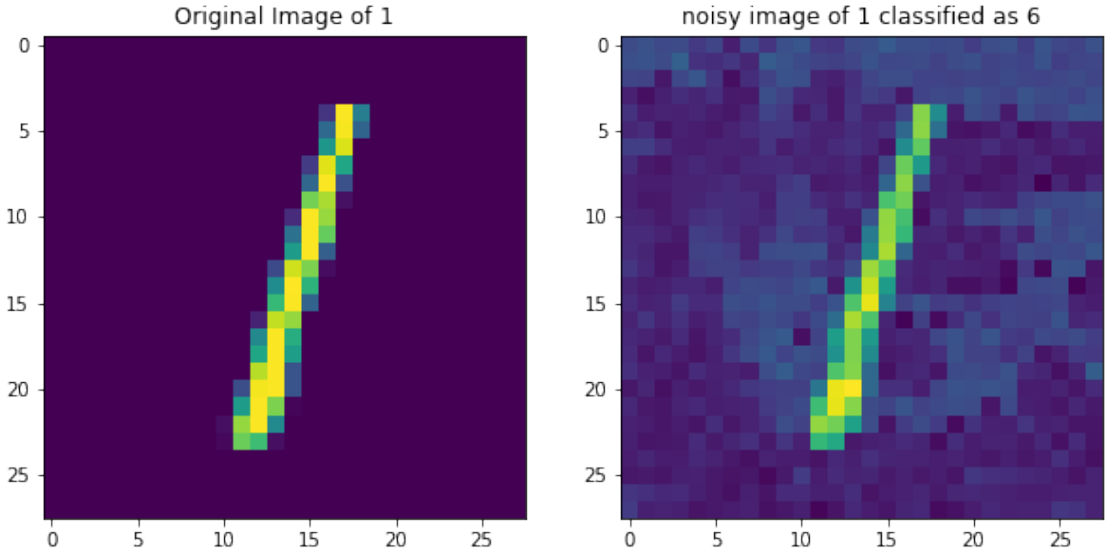
```
Input the original image class  1
Input the target image class    6
Iteration: 0     target: 6       probability: [0.]       Logit Value:
[-120.84282]
Iteration: 5     target: 6       probability: [0.]       Logit Value:
[-116.45839]
Iteration: 10    target: 6       probability: [0.]       Logit Value:
[-112.15775]
Iteration: 15    target: 6       probability: [0.]       Logit Value:
[-107.888084]
Iteration: 20    target: 6       probability: [0.]       Logit Value:
[-103.65156]
Iteration: 25    target: 6       probability: [0.]       Logit Value:
[-99.45339]
Iteration: 30    target: 6       probability: [0.]       Logit Value:
[-95.23397]
Iteration: 35    target: 6       probability: [0.]       Logit Value:
[-90.99499]
Iteration: 40    target: 6       probability: [0.]       Logit Value:
[-86.78435]
Iteration: 45    target: 6       probability: [0.]       Logit Value: [-82.5534]
Iteration: 50    target: 6       probability: [0.]       Logit Value:
[-78.32528]
Iteration: 55    target: 6       probability: [0.]       Logit Value:
[-74.09195]
Iteration: 60    target: 6       probability: [0.]       Logit Value:
[-69.906425]
Iteration: 65    target: 6       probability: [0.]       Logit Value:
[-65.785255]
Iteration: 70    target: 6       probability: [0.]       Logit Value:
[-61.70714]
Iteration: 75    target: 6       probability: [0.]       Logit Value:
```

```
[-57.652782]
Iteration: 80    target: 6       probability: [0.]       Logit Value:
[-53.597675]
Iteration: 85    target: 6       probability: [0.]       Logit Value:
[-49.57004]
Iteration: 90    target: 6       probability: [0.]       Logit Value:
[-45.556816]
Iteration: 95    target: 6       probability: [0.]       Logit Value:
[-41.46698]
Iteration: 100   target: 6       probability: [0.]       Logit Value:
[-37.281227]
Iteration: 105   target: 6       probability: [0.]       Logit Value:
[-33.078987]
Iteration: 110   target: 6       probability: [0.]       Logit Value:
[-28.798023]
Iteration: 115   target: 6       probability: [0.]       Logit Value:
[-24.466846]
Iteration: 120   target: 6       probability: [0.]       Logit Value:
[-20.15675]
Iteration: 125   target: 6       probability: [0.]       Logit Value:
[-15.857097]
Iteration: 130   target: 6       probability: [0.]       Logit Value:
[-11.495364]
Iteration: 135   target: 6       probability: [0.]       Logit Value:
[-7.110835]
Iteration: 140   target: 6       probability: [0.]       Logit Value:
[-2.7245054]
Iteration: 145   target: 6       probability: [0.]       Logit Value:
[1.6141633]
Iteration: 150   target: 6       probability: [0.]       Logit Value:
[5.9246187]
Iteration: 155   target: 6       probability: [0.]       Logit Value:
[10.305806]
Iteration: 160   target: 6       probability: [0.]       Logit Value:
[14.685559]
Iteration: 165   target: 6       probability: [0.]       Logit Value: [19.06596]
Iteration: 170   target: 6       probability: [0.]       Logit Value:
[23.400944]
Iteration: 175   target: 6       probability: [0.]       Logit Value:
[27.757862]
Iteration: 180   target: 6       probability: [0.]       Logit Value: [32.11764]
Iteration: 185   target: 6       probability: [0.]       Logit Value: [36.47514]
Iteration: 190   target: 6       probability: [0.]       Logit Value:
[40.838604]
Iteration: 195   target: 6       probability: [0.]       Logit Value:
[45.189182]
Iteration: 200   target: 6       probability: [0.]       Logit Value:
[49.510468]
```

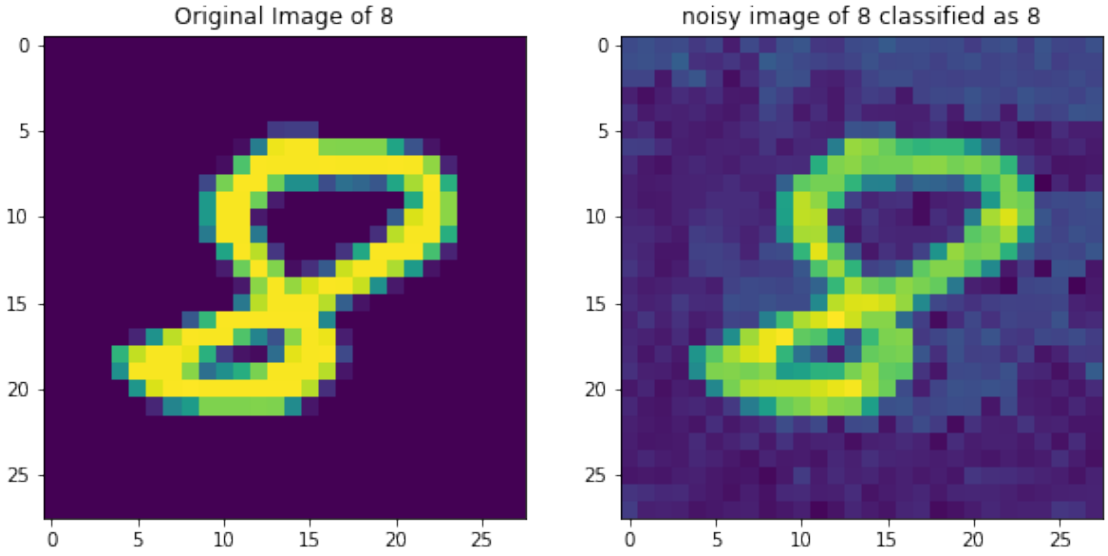
```
Iteration: 205   target: 6      probability: [0.]       Logit Value: [53.83653]
Iteration: 210   target: 6      probability: [0.]       Logit Value:
[58.032223]
Iteration: 215   target: 6      probability: [0.]       Logit Value:
[62.198997]
Iteration: 220   target: 6      probability: [0.]       Logit Value: [66.39063]
Iteration: 225   target: 6      probability: [0.]       Logit Value:
[70.589935]
Iteration: 230   target: 6      probability: [0.]       Logit Value:
[74.783585]
Iteration: 235   target: 6      probability: [0.]       Logit Value:
[78.958145]
Iteration: 240   target: 6      probability: [0.]       Logit Value: [83.14925]
Iteration: 245   target: 6      probability: [0.]       Logit Value: [87.34281]
Iteration: 250   target: 6      probability: [0.]       Logit Value:
[91.513275]
Iteration: 255   target: 6      probability: [0.]       Logit Value: [95.69894]
Iteration: 260   target: 6      probability: [0.]       Logit Value: [99.90609]
Iteration: 265   target: 6      probability: [0.]       Logit Value:
[104.18825]
Iteration: 270   target: 6      probability: [0.]       Logit Value:
[108.46722]
Iteration: 275   target: 6      probability: [0.]       Logit Value:
[112.75619]
Iteration: 280   target: 6      probability: [0.]       Logit Value:
[117.05495]
Iteration: 285   target: 6      probability: [0.]       Logit Value:
[121.37104]
Iteration: 290   target: 6      probability: [0.]       Logit Value:
[125.827156]
Iteration: 295   target: 6      probability: [0.]       Logit Value:
[130.29489]
Iteration: 300   target: 6      probability: [0.]       Logit Value:
[134.76794]
Iteration: 305   target: 6      probability: [0.]       Logit Value: [139.2342]
Iteration: 310   target: 6      probability: [0.]       Logit Value:
[143.71367]
Iteration: 315   target: 6      probability: [8.e-45]   Logit Value:
[148.20105]
Iteration: 320   target: 6      probability: [1.98e-43]         Logit Value:
[152.70844]
Iteration: 325   target: 6      probability: [4.975e-42]        Logit Value:
[157.215]
Iteration: 330   target: 6      probability: [1.28997e-40]      Logit Value:
[161.70735]
Iteration: 335   target: 6      probability: [3.497672e-39]     Logit Value:
[166.17903]
Iteration: 340   target: 6      probability: [9.587751e-38]     Logit Value:
```

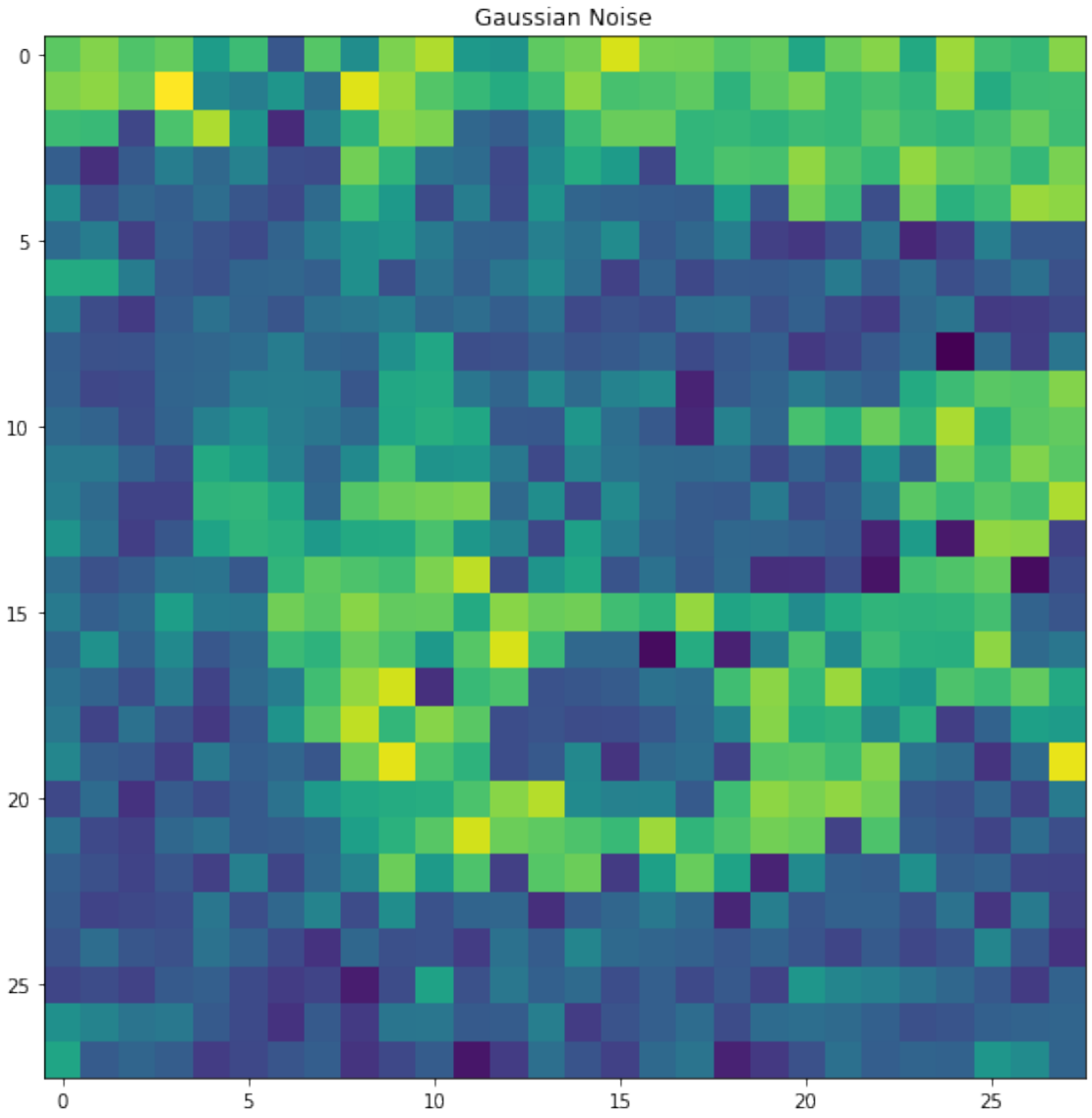
```
[170.66594]
Iteration: 345   target: 6       probability: [2.4672863e-36]    Logit Value:
[175.14253]
Iteration: 350   target: 6       probability: [5.9502956e-35]    Logit Value:
[179.6344]
Iteration: 355   target: 6       probability: [1.5106252e-33]    Logit Value:
[184.17712]
Iteration: 360   target: 6       probability: [3.7528815e-32]    Logit Value:
[188.72229]
Iteration: 365   target: 6       probability: [9.766636e-31]     Logit Value:
[193.25668]
Iteration: 370   target: 6       probability: [2.55953e-29]      Logit Value:
[197.79614]
Iteration: 375   target: 6       probability: [6.92879e-28]      Logit Value:
[202.3387]
Iteration: 380   target: 6       probability: [1.8786983e-26]    Logit Value:
[206.89755]
Iteration: 385   target: 6       probability: [4.8970325e-25]    Logit Value:
[211.43645]
Iteration: 390   target: 6       probability: [1.3008667e-23]    Logit Value:
[215.98582]
Iteration: 395   target: 6       probability: [3.3422565e-22]    Logit Value:
[220.53339]
Iteration: 400   target: 6       probability: [8.489005e-21]     Logit Value:
[225.0705]
Iteration: 405   target: 6       probability: [2.1372244e-19]    Logit Value:
[229.59233]
Iteration: 410   target: 6       probability: [5.2181547e-18]    Logit Value:
[234.11568]
Iteration: 415   target: 6       probability: [1.2206997e-16]    Logit Value:
[238.64331]
Iteration: 420   target: 6       probability: [2.8986005e-15]    Logit Value:
[243.16391]
Iteration: 425   target: 6       probability: [6.608872e-14]     Logit Value:
[247.69434]
Iteration: 430   target: 6       probability: [1.4965944e-12]    Logit Value:
[252.22542]
Iteration: 435   target: 6       probability: [3.885591e-11]     Logit Value:
[256.78198]
Iteration: 440   target: 6       probability: [9.897071e-10]     Logit Value:
[261.34277]
Iteration: 445   target: 6       probability: [2.5952172e-08]    Logit Value:
[265.89835]
Iteration: 450   target: 6       probability: [6.9506467e-07]    Logit Value:
[270.45212]
Iteration: 455   target: 6       probability: [1.7995802e-05]    Logit Value:
[275.01865]
Iteration: 460   target: 6       probability: [0.0004588]        Logit Value:
```

```
[279.57602]
Iteration: 465   target: 6       probability: [0.01211608]       Logit Value:
[284.06186]
Iteration: 470   target: 6       probability: [0.24875082]       Logit Value:
[288.4764]
Iteration: 473   target: 6       probability: [0.5525548]        Logit Value:
[290.24255]
```




```
Input the number to which you want to add noise 8
```





19.5 *Observations:*

- The iterations run till the target class gets highest probability
- First Plot is of the original image with noise added to it to classify it as another number
- Second plot shows behaviour of a new class image on noise being added to it
- Third plot is the noise kernel