we usually flatten the input:

from (64, 1, 28, 28) to (64, 784)

Build a network for identifying images:

```
inputs = images.view(images.shape[0], -1)
```

choose the appropriate size

```
w1 = torch.randn(784, 256)
b1 = torch.randn(256)
w2 = torch.randn(256, 10)
b2 = torch.randn(10)

h = sigmoid(torch.mm(inputs, w1) + b1)
out = torch.mm(h, w2) + b2
```

input units

hidden units

one output digit for each digit.

one hidden layer

```
def softmax(x):
    return torch.exp(x)/torch.sum(torch.exp(x), dim=1)

probs = softmax(out)  # shape = (64, 10)
```

- view(-1, 1)

Neural Networks with Pytorch

```
from torchvision import datasets, transforms
```

for normalizing data

```
transform = transforms.Compose([transforms.ToTensor(),
                                transforms.Normalize((0.5,0.5,0.5), (0.5,0.5,0.5)),
                               ])
```

download mnist dataset

```
trainset = datasets.MNIST('MNIST_data/',
                          download=True,
                          train=True,
                          transform=transform)
```

define a loader

```
trainloader = torch.utils.data.DataLoader(trainset,
                                          batch_size=64,
                                          shuffle=True)
```

each time we get a batch, it's shape is (64, 1, 28, 28)

- 64 → 64 images
- 1 → 1 color channel
- 28, 28 → image size

```
dataiter = iter(trainloader)
images, labels = dataiter.next()
```