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)
```

\[w_1 = \text{torch.randn}(784, 256)\]
\[b_1 = \text{torch.randn}(256)\]
\[w_2 = \text{torch.randn}(256, 10)\]
\[b_2 = \text{torch.randn}(10)\]

\[h = \text{sigmoid}(\text{torch.mm}(\text{inputs}, w_1) + b_1)\]
\[\text{out} = \text{torch.mm}(h, w_2) + b_2\]

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

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

train_loader = 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 images
1 color channel

dataiter = iter(train_loader)
images, labels = dataiter.next()