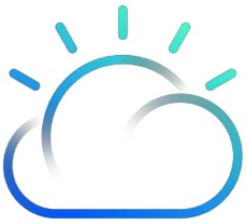


Watson Studio democratizes machine learning and deep learning to accelerate infusion of AI in your business to drive innovation. Watson Studio provides a suite of tools and a collaborative environment for data scientists, developers and domain experts.

(http://cocl.us/pytorch_link_top)

Prebuilt Datasets and Transforms

Table of Contents

In this lab, you will use a prebuilt dataset and then use some prebuilt dataset transforms.

Estimated Time Needed: **10 min**

Preparation

The following are the libraries we are going to use for this lab. The `torch.manual_seed()` is for forcing the random function to give the same number every time we try to recompile it.

In [1]:

```
# These are the libraries will be used for this lab.

import torch
import matplotlib.pylab as plt
import numpy as np
torch.manual_seed(0)
```

Out[1]:

```
<torch._C.Generator at 0x7f2a685d0f30>
```

This is the function for displaying images.

In [2]:

```
# Show data by diagram

def show_data(data_sample, shape = (28, 28)):
    plt.imshow(data_sample[0].numpy().reshape(shape), cmap='gray')
    plt.title('y = ' + str(data_sample[1].item()))
```

Prebuilt Datasets

You will focus on the following libraries:

In [3]:

```
# Run the command below when you do not have torchvision installed
# !conda install -y torchvision

import torchvision.transforms as transforms
import torchvision.datasets as dsets
```

We can import a prebuilt dataset. In this case, use MNIST. You'll work with several of these parameters later by placing a transform object in the argument `transform` .

In [4]:

```
# Import the prebuilt dataset into variable dataset

dataset = dsets.MNIST(
    root = './data',
    train = False,
    download = True,
    transform = transforms.ToTensor()
)
```

```
Downloading http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz
Processing...
Done!
```

Each element of the dataset object contains a tuple. Let us see whether the first element in the dataset is a tuple and what is in it.

In [5]:

```
# Examine whether the elements in dataset MNIST are tuples, and what is in the tupl
e?

print("Type of the first element: ", type(dataset[0]))
print("The length of the tuple: ", len(dataset[0]))
print("The shape of the first element in the tuple: ", dataset[0][0].shape)
print("The type of the first element in the tuple", type(dataset[0][0]))
print("The second element in the tuple: ", dataset[0][1])
print("The type of the second element in the tuple: ", type(dataset[0][1]))
print("As the result, the structure of the first element in the dataset is (tensor
([1, 28, 28]), tensor(7)).")
```

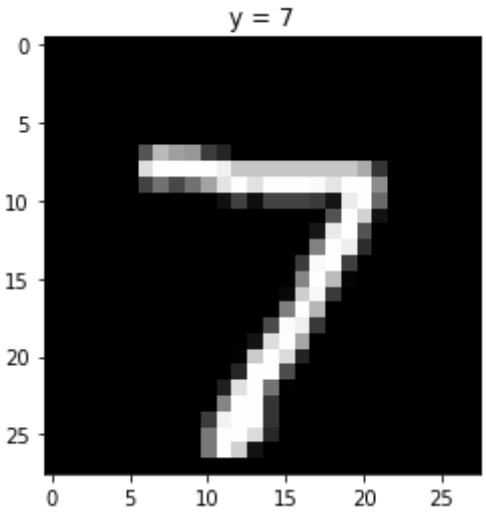
```
Type of the first element:  <class 'tuple'>
The length of the tuple:  2
The shape of the first element in the tuple:  torch.Size([1, 28, 28])
The type of the first element in the tuple <class 'torch.Tensor'>
The second element in the tuple:  tensor(7)
The type of the second element in the tuple:  <class 'torch.Tensor'>
As the result, the structure of the first element in the dataset is (te
nsor([1, 28, 28]), tensor(7)).
```

As shown in the output, the first element in the tuple is a cuboid tensor. As you can see, there is a dimension with only size 1, so basically, it is a rectangular tensor.
The second element in the tuple is a number tensor, which indicate the real number the image shows. As the second element in the tuple is `tensor(7)` , the image should show a hand-written 7.

Let us plot the first element in the dataset:

In [6]:

```
# Plot the first element in the dataset

show_data(dataset[0])
```

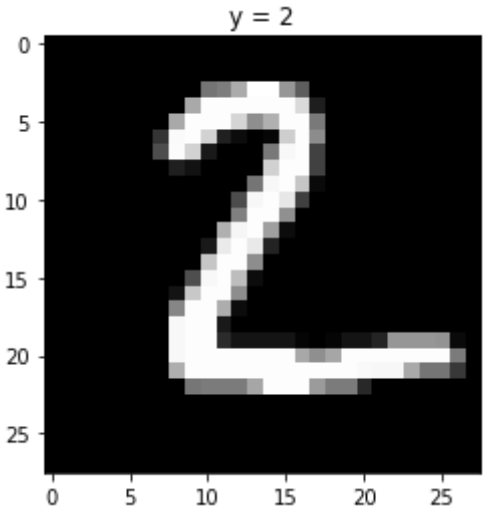


As we can see, it is a 7.

Plot the second sample:

In [7]:

```
# Plot the second element in the dataset

show_data(dataset[1])
```



Torchvision Transforms

We can apply some image transform functions on the MNIST dataset.

As an example, the images in the MNIST dataset can be cropped and converted to a tensor. We can use `transform.Compose` we learned from the previous lab to combine the two transform functions.

In [8]:

```
# Combine two transforms: crop and convert to tensor. Apply the compose to MNIST da
taset

croptensor_data_transform = transforms.Compose([transforms.CenterCrop(20), transfor
ms.ToTensor()])
dataset = dsets.MNIST(root = './data', train = False, download = True, transform =
croptensor_data_transform)
print("The shape of the first element in the first tuple: ", dataset[0][0].shape)
```

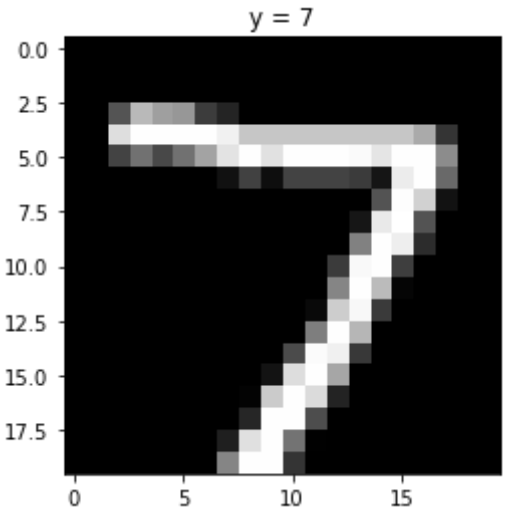
```
The shape of the first element in the first tuple:  torch.Size([1, 20,
20])
```

We can see the image is now 20 x 20 instead of 28 x 28.

Let us plot the first image again. Notice that the black space around the **7** become less apparent.

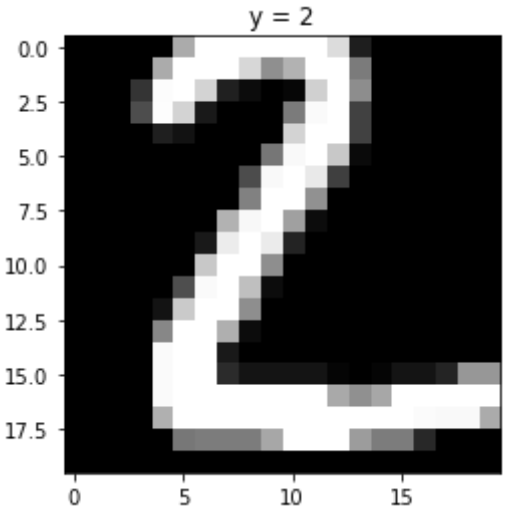
In [9]:

```
# Plot the first element in the dataset

show_data(dataset[0],shape = (20, 20))
```



In [10]:

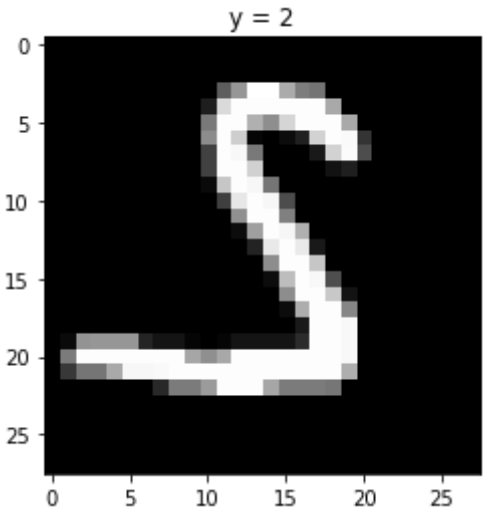
```
# Plot the second element in the dataset

show_data(dataset[1],shape = (20, 20))
```



In the below example, we horizontally flip the image, and then convert it to a tensor. Use `transforms.Compose()` to combine these two transform functions. Plot the flipped image.

In [11]:

```
# Construct the compose. Apply it on MNIST dataset. Plot the image out.

fliptensor_data_transform = transforms.Compose([transforms.RandomHorizontalFlip(p =
1),transforms.ToTensor()])
dataset = dsets.MNIST(root = './data', train = False, download = True, transform =
fliptensor_data_transform)
show_data(dataset[1])
```

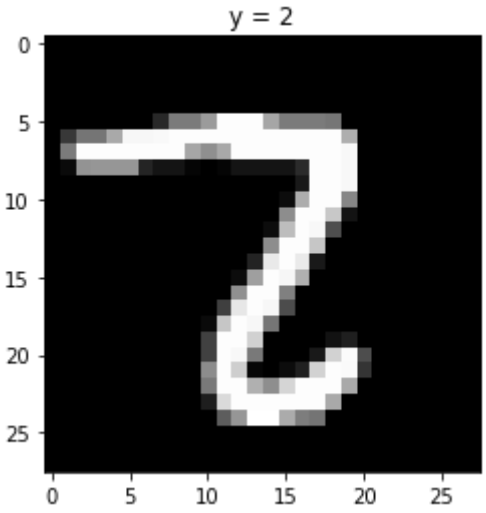


Practice

Try to use the `RandomVerticalFlip` (vertically flip the image) with horizontally flip and convert to tensor as a compose. Apply the compose on image. Use `show_data()` to plot the second image (the image as **2**).

In [13]:

```
# Practice: Combine vertical flip, horizontal flip and convert to tensor as a compo
se. Apply the compose on image. Then plot the image

data_transform_all = transforms.Compose([transforms.RandomVerticalFlip(p = 1),
                                         transforms.RandomHorizontalFlip(p = 1),
                                         transforms.ToTensor()])
dataset = dsets.MNIST(root = './data', train = False, download = True, transform =
data_transform_all)
show_data(dataset[1])
```



Double-click **here** for the solution.

About the Authors:

Joseph Santarcangelo (https://www.linkedin.com/in/joseph-s-50398b136/) has a PhD in Electrical Engineering, his research focused on using machine learning, signal processing, and computer vision to determine how videos impact human cognition. Joseph has been working for IBM since he completed his PhD.

Other contributors: Michelle Carey (https://www.linkedin.com/in/michelleccarey/), Mavis Zhou (www.linkedin.com/in/jiahui-mavis-zhou-a4537814a)