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)

COGNITIVE CLASS

Image Datasets and Transforms

Table of Contents

In this lab, you will build a dataset objects for images; many of the processes can be applied to a larger dataset. Then you will apply pre-build transforms from Torchvision Transforms to that dataset.

Estimated Time Needed: **25 min**

Preparation

Download the dataset and unzip the files in your data directory, **to download faster this dataset has only 100 samples**:

In [1]:

```
! wget https://s3-api.us-geo.objectstorage.softlayer.net/cf-courses-data/CognitiveC
lass/DL0110EN/datasets/img.tar.gz -P /resources/data
```

```
--2020-04-20 22:27:37--  https://s3-api.us-geo.objectstorage.softlayer.
net/cf-courses-data/CognitiveClass/DL0110EN/datasets/img.tar.gz
Resolving s3-api.us-geo.objectstorage.softlayer.net (s3-api.us-geo.obje
ctstorage.softlayer.net)... 67.228.254.196
Connecting to s3-api.us-geo.objectstorage.softlayer.net (s3-api.us-geo.
objectstorage.softlayer.net)|67.228.254.196|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 50460 (49K) [application/octet-stream]
Saving to: '/resources/data/img.tar.gz'

img.tar.gz          100%[===================>]  49.28K  --.-KB/s    in
0.002s

2020-04-20 22:27:37 (21.4 MB/s) - '/resources/data/img.tar.gz' saved [5
0460/50460]
```

In [2]:

```
!tar -xf /resources/data/img.tar.gz
```

In [3]:

```
!wget https://s3-api.us-geo.objectstorage.softlayer.net/cf-courses-data/CognitiveCl
ass/DL0110EN/datasets/index.csv
```

```
--2020-04-20 22:28:00--  https://s3-api.us-geo.objectstorage.softlayer.
net/cf-courses-data/CognitiveClass/DL0110EN/datasets/index.csv
Resolving s3-api.us-geo.objectstorage.softlayer.net (s3-api.us-geo.obje
ctstorage.softlayer.net)... 67.228.254.196
Connecting to s3-api.us-geo.objectstorage.softlayer.net (s3-api.us-geo.
objectstorage.softlayer.net)|67.228.254.196|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1680905 (1.6M) [text/csv]
Saving to: 'index.csv'

index.csv           100%[===================>]   1.60M  3.32MB/s    in
0.5s

2020-04-20 22:28:01 (3.32 MB/s) - 'index.csv' saved [1680905/1680905]
```

We will use this function in the lab:

In [4]:

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

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 [5]:

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

import torch
import matplotlib.pylab as plt
import numpy as np
from torch.utils.data import Dataset, DataLoader
torch.manual_seed(0)
```

Out[5]:

```
<torch._C.Generator at 0x7f0775cfa310>
```

In [6]:

```
from matplotlib.pyplot import imshow
import matplotlib.pylab as plt
from PIL import Image
import pandas as pd
import os
```

Auxiliary Functions

You will use the following function as components of a dataset object, in this section, you will review each of the components independently.

The path to the csv file with the labels for each image.

In [7]:

```
# Read CSV file from the URL and print out the first five samples
directory=""
csv_file ='index.csv'
csv_path=os.path.join(directory,csv_file)
```

You can load the CSV file and convert it into a dataframe , using the Pandas function `read_csv()` . You can view the dataframe using the method head.

In [8]:

```
data_name = pd.read_csv(csv_path)
data_name.head()
```

Out[8]:

	category	image
0	Ankle boot	img/fashion0.png
1	T-shirt	img/fashion1.png
2	T-shirt	img/fashion2.png
3	Dress	img/fashion3.png
4	T-shirt	img/fashion4.png

The first column of the dataframe corresponds to the type of clothing. The second column is the name of the image file corresponding to the clothing. You can obtain the path of the first file by using the method `DATAFRAME.iloc[0, 1]` . The first argument corresponds to the sample number, and the second input corresponds to the column index.

In [9]:

```
# Get the value on location row 0, column 1 (Notice that index starts at 0)
#rember this dataset has only 100 samples to make the download faster
print('File name:', data_name.iloc[0, 1])
```

```
File name: img/fashion0.png
```

As the class of the sample is in the first column, you can also obtain the class value as follows.

In [10]:

```
# Get the value on location row 0, column 0 (Notice that index starts at 0.)

print('y:', data_name.iloc[0, 0])
```

```
y: Ankle boot
```

Similarly, You can obtain the file name of the second image file and class type:

In [11]:

```
# Print out the file name and the class number of the element on row 1 (the second
 row)

print('File name:', data_name.iloc[1, 1])
print('class or y:', data_name.iloc[1, 0])
```

```
File name: img/fashion1.png
class or y: T-shirt
```

The number of samples corresponds to the number of rows in a dataframe. You can obtain the number of rows using the following lines of code. This will correspond the data attribute `len` .

In [12]:

```
# Print out the total number of rows in traing dataset

print('The number of rows: ', data_name.shape[0])
```

```
The number of rows:  60000
```

Load Image

To load the image, you need the directory and the image name. You can concatenate the variable `train_data_dir` with the name of the image stored in a Dataframe. Finally, you will store the result in the variable `image_name`

In [13]:

```
# Combine the directory path with file name

image_name =data_name.iloc[1, 1]
image_name
```

Out[13]:

```
'img/fashion1.png'
```

we can find the image path:

In [14]:

```
image_path=os.path.join(directory,image_name)
image_path
```

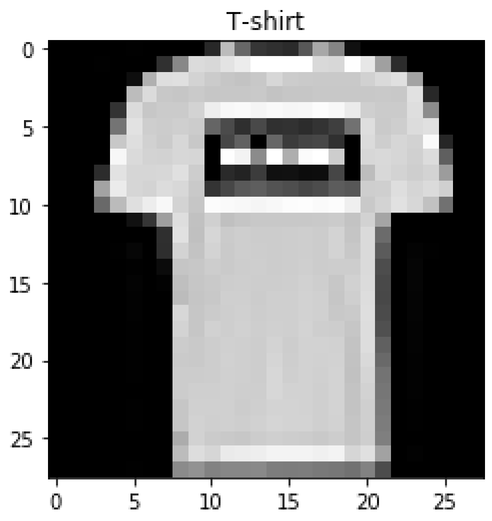
Out[14]:

```
'img/fashion1.png'
```

You can then use the function `Image.open` to store the image to the variable `image` and display the image and class .

```
In [15]:
```

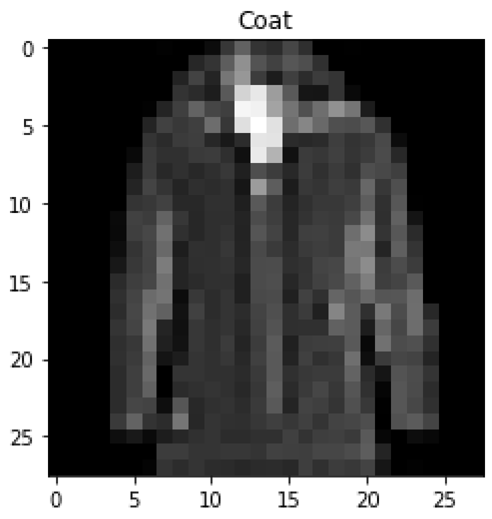
```
# Plot the second training image

image = Image.open(image_path)
plt.imshow(image,cmap='gray', vmin=0, vmax=255)
plt.title(data_name.iloc[1, 0])
plt.show()
```



You can repeat the process for the 20th image.

In [16]:

```
# Plot the 20th image

image_name = data_name.iloc[19, 1]
image_path=os.path.join(directory,image_name)
image = Image.open(image_path)
plt.imshow(image,cmap='gray', vmin=0, vmax=255)
plt.title(data_name.iloc[19, 0])
plt.show()
```



Create the dataset object.

Create a Dataset Class

In this section, we will use the components in the last section to build a dataset class and then create an object.

In [17]:

```
# Create your own dataset object

class Dataset(Dataset):

    # Constructor
    def __init__(self, csv_file, data_dir, transform=None):

        # Image directory
        self.data_dir=data_dir

        # The transform is goint to be used on image
        self.transform = transform
        data_dircsv_file=os.path.join(self.data_dir,csv_file)
        # Load the CSV file contians image info
        self.data_name= pd.read_csv(data_dircsv_file)

        # Number of images in dataset
        self.len=self.data_name.shape[0]

    # Get the length
    def __len__(self):
        return self.len

    # Getter
    def __getitem__(self, idx):

        # Image file path
        img_name=os.path.join(self.data_dir,self.data_name.iloc[idx, 1])
        # Open image file
        image = Image.open(img_name)

        # The class label for the image
        y = self.data_name.iloc[idx, 0]

        # If there is any transform method, apply it onto the image
        if self.transform:
            image = self.transform(image)

        return image, y
```

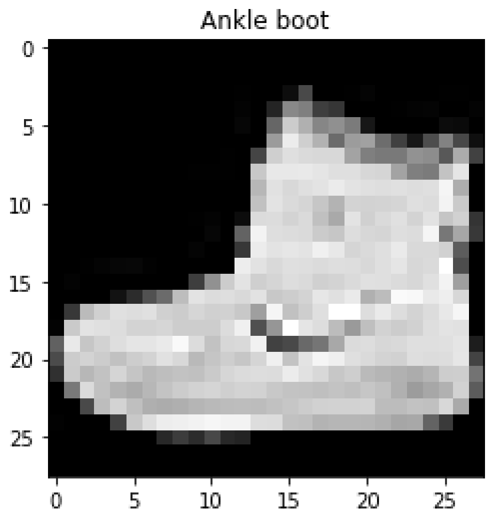
In [18]:

```
# Create the dataset objects

dataset = Dataset(csv_file=csv_file, data_dir=directory)
```

Each sample of the image and the class y is stored in a tuple `dataset[sample]` . The image is the first element in the tuple `dataset[sample][0]` the label or class is the second element in the tuple `dataset[sample][1]` . For example you can plot the first image and class.

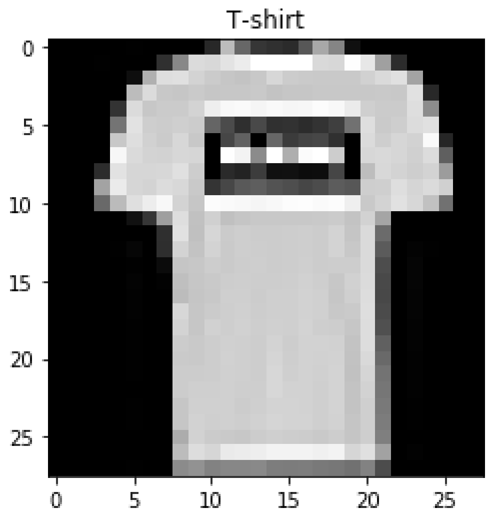
In [19]:

```
image=dataset[0][0]
y=dataset[0][1]

plt.imshow(image,cmap='gray', vmin=0, vmax=255)
plt.title(y)
plt.show()
```



In [30]:

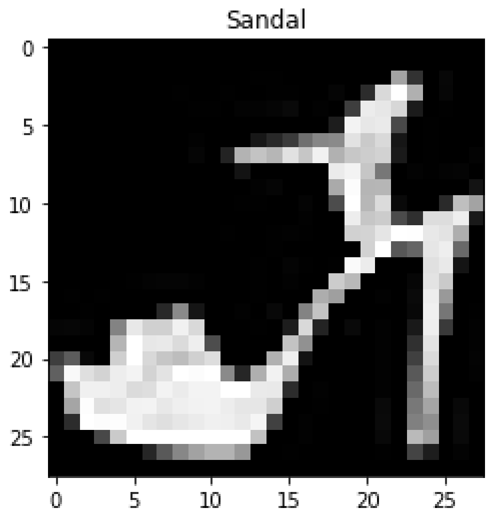
```
image2 = dataset[1][0]
y2 = dataset[1][1]
plt.imshow(image2, cmap='gray', vmin=0, vmax=255)
plt.title(y2)
plt.show()
```



Similarly, you can plot the second image:

In [31]:

```
image=dataset[9][0]
y=dataset[9][1]

plt.imshow(image,cmap='gray', vmin=0, vmax=255)
plt.title(y)
plt.show()
```



Torchvision Transforms

You will focus on the following libraries:

In [32]:

```
import torchvision.transforms as transforms
```

We can apply some image transform functions on the dataset object. The iamge 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 [33]:

```
# 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 = Dataset(csv_file=csv_file , data_dir=directory,transform=croptensor_data_
transform )
print("The shape of the first element tensor: ", dataset[0][0].shape)
```

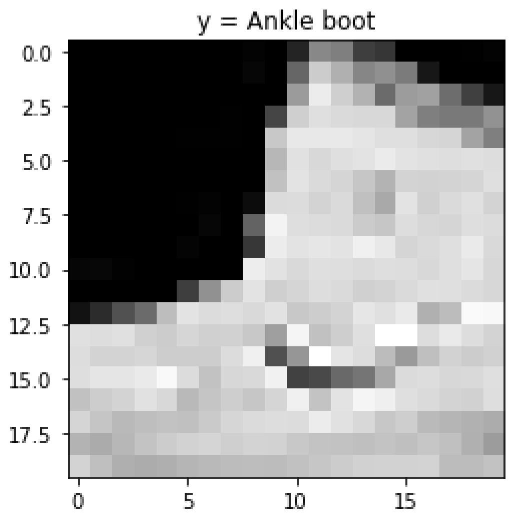
```
The shape of the first element tensor:  torch.Size([1, 20, 20])
```

We can see the image is now 20 x 20

Let us plot the first image again. Notice we see less of the shoe.

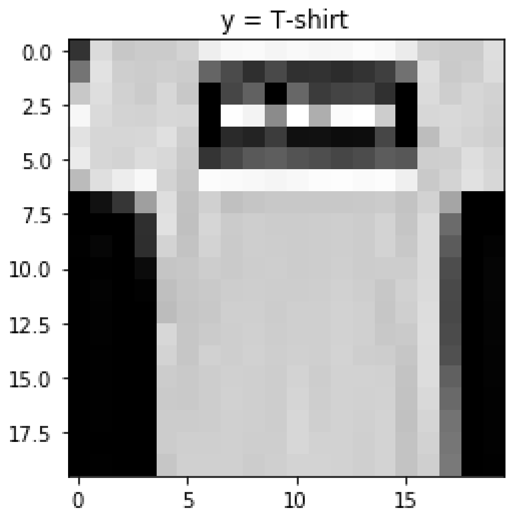
In [34]:

```
# Plot the first element in the dataset

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



In [35]:

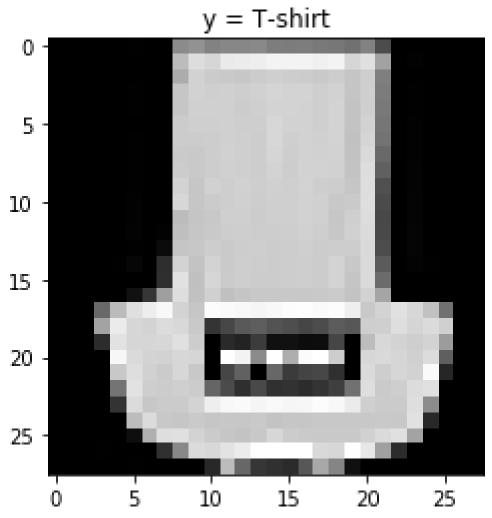
```
# Plot the second element in the dataset

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



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

In [36]:

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

fliptensor_data_transform = transforms.Compose([transforms.RandomVerticalFlip(p=1),
transforms.ToTensor()])
dataset = Dataset(csv_file=csv_file , data_dir=directory,transform=fliptensor_data_
transform )
show_data(dataset[1])
```

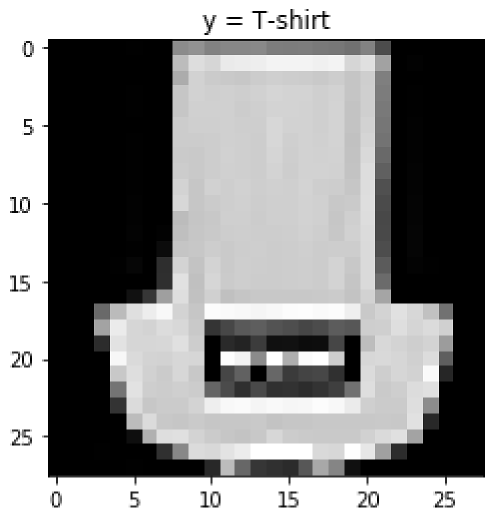


Practice

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 [38]:

```
# 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 = Dataset(csv_file = csv_file, data_dir = directory, transform = fliptensor
_data_transform)
show_data(dataset[1])
```



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

Get IBM Watson Studio free of charge!

Build and train AI & machine learning models, prepare and analyze data – all in a flexible, hybrid cloud environment. Get IBM Watson Studio Lite Plan free of charge.

Learn
Get started or get better with built-in learning.
Create
Use the best of open source tooling with IBM innovation.
Collaborate
Work smarter using community, work faster with your team.
Sign Up For a Free Trial

(http://cocl.us/pytorch_link_bottom)

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)
