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.

# Fashion-MNIST Project

## Table of Contents

In this project, you will classify Fashion-MNIST dataset using convolutional neural networks.



Estimated Time Needed: **30 min**

---

## Preparation

Download the datasets you needed for this lab.

The following are the PyTorch modules you are going to need

In [1]:
```
!pip install torch
!pip install torchvision
```

```
Requirement already satisfied: torch in /opt/conda/envs/Python-RT23.1/lib/python3.10/si
te-packages (2.0.1)
Requirement already satisfied: filelock in /opt/conda/envs/Python-RT23.1/lib/python3.1
0/site-packages (from torch) (3.9.0)
Requirement already satisfied: typing-extensions in /opt/conda/envs/Python-RT23.1/lib/p
ython3.10/site-packages (from torch) (4.4.0)
Requirement already satisfied: sympy in /opt/conda/envs/Python-RT23.1/lib/python3.10/si
te-packages (from torch) (1.11.1)
Requirement already satisfied: networkx in /opt/conda/envs/Python-RT23.1/lib/python3.1
0/site-packages (from torch) (2.8.4)
Requirement already satisfied: jinja2 in /opt/conda/envs/Python-RT23.1/lib/python3.10/s
ite-packages (from torch) (3.1.2)
Requirement already satisfied: MarkupSafe>=2.0 in /opt/conda/envs/Python-RT23.1/lib/pyt
hon3.10/site-packages (from jinja2->torch) (2.1.1)
Requirement already satisfied: mpmath>=0.19 in /opt/conda/envs/Python-RT23.1/lib/python
3.10/site-packages (from sympy->torch) (1.3.0)
Requirement already satisfied: torchvision in /opt/conda/envs/Python-RT23.1/lib/python
3.10/site-packages (0.15.2)
Requirement already satisfied: numpy in /opt/conda/envs/Python-RT23.1/lib/python3.10/si
te-packages (from torchvision) (1.23.5)
Requirement already satisfied: requests in /opt/conda/envs/Python-RT23.1/lib/python3.1
0/site-packages (from torchvision) (2.31.0)
Requirement already satisfied: torch in /opt/conda/envs/Python-RT23.1/lib/python3.10/si
te-packages (from torchvision) (2.0.1)
Requirement already satisfied: pillow!=8.3.*,>=5.3.0 in /opt/conda/envs/Python-RT23.1/l
ib/python3.10/site-packages (from torchvision) (10.0.1)
Requirement already satisfied: charset-normalizer<4,>=2 in /opt/conda/envs/Python-RT23.
1/lib/python3.10/site-packages (from requests->torchvision) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/envs/Python-RT23.1/lib/python
3.10/site-packages (from requests->torchvision) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in /opt/conda/envs/Python-RT23.1/lib/
python3.10/site-packages (from requests->torchvision) (1.26.18)
Requirement already satisfied: certifi>=2017.4.17 in /opt/conda/envs/Python-RT23.1/lib/
python3.10/site-packages (from requests->torchvision) (2023.7.22)
Requirement already satisfied: filelock in /opt/conda/envs/Python-RT23.1/lib/python3.1
0/site-packages (from torch->torchvision) (3.9.0)
Requirement already satisfied: typing-extensions in /opt/conda/envs/Python-RT23.1/lib/p
ython3.10/site-packages (from torch->torchvision) (4.4.0)
Requirement already satisfied: sympy in /opt/conda/envs/Python-RT23.1/lib/python3.10/si
te-packages (from torch->torchvision) (1.11.1)
Requirement already satisfied: networkx in /opt/conda/envs/Python-RT23.1/lib/python3.1
0/site-packages (from torch->torchvision) (2.8.4)
Requirement already satisfied: jinja2 in /opt/conda/envs/Python-RT23.1/lib/python3.10/s
ite-packages (from torch->torchvision) (3.1.2)
Requirement already satisfied: MarkupSafe>=2.0 in /opt/conda/envs/Python-RT23.1/lib/pyt
hon3.10/site-packages (from jinja2->torch->torchvision) (2.1.1)
Requirement already satisfied: mpmath>=0.19 in /opt/conda/envs/Python-RT23.1/lib/python
3.10/site-packages (from sympy->torch->torchvision) (1.3.0)
```

In [2]:
```python
# PyTorch Modules you need for this lab

from torch.utils.data import Dataset, DataLoader

from torchvision import transforms
import torch
import torch.nn as nn
import torchvision.transforms as transforms
import torchvision.datasets as dsets
torch.manual_seed(0)
```

Out[2]:
```
<torch._C.Generator at 0x7f9b1114b190>
```

Import Non-PyTorch Modules

In [3]:
```
# Other non-PyTorch Modules

from matplotlib.pyplot import imshow
import matplotlib.pylab as plt

from PIL import Image
```

In [4]:
```
def show_data(data_sample):
    plt.imshow(data_sample[0].numpy().reshape(IMAGE_SIZE, IMAGE_SIZE), cmap='gray')
    plt.title('y = '+ str(data_sample[1]))
```

---

---

# Questions 1: Create a Dataset Class

In this section, you will load a Dataset object, but first you must transform the dataset. Use the `Compose` function to perform the following transforms:.

1. use the transforms object to `Resize` to resize the image.
2. use the transforms object to `ToTensor` to convert the image to a tensor.

You will then take a screen shot of your validation data.

Use the compose function ot compse the

In [5]:
```
#Hint:

IMAGE_SIZE = 16

transforms.Resize((IMAGE_SIZE, IMAGE_SIZE)),
transforms.ToTensor()#
composed = transforms.Compose([transforms.Resize((IMAGE_SIZE, IMAGE_SIZE)), transforms.
```

---

Create two dataset objects for the Fashion MNIST dataset. One for training data called `dataset_train` and one for validation data `dataset_val`. You will be asked to take a screenshot of several samples.

**Hint:** `dsets.FashionMNIST(root= '.fashion/data', train=???, transform=composed, download=True)`

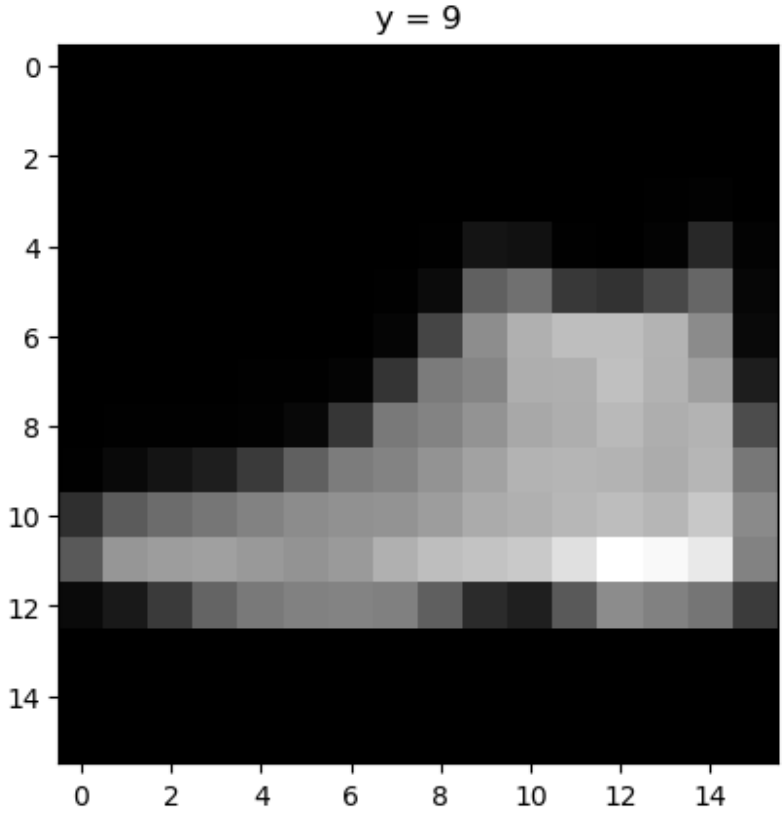
In [6]:
```
dataset_train = dsets.FashionMNIST(root= '.fashion/data', train=True, transform=compose
dataset_val = dsets.FashionMNIST(root= '.fashion/data', train=False, transform=composed
```

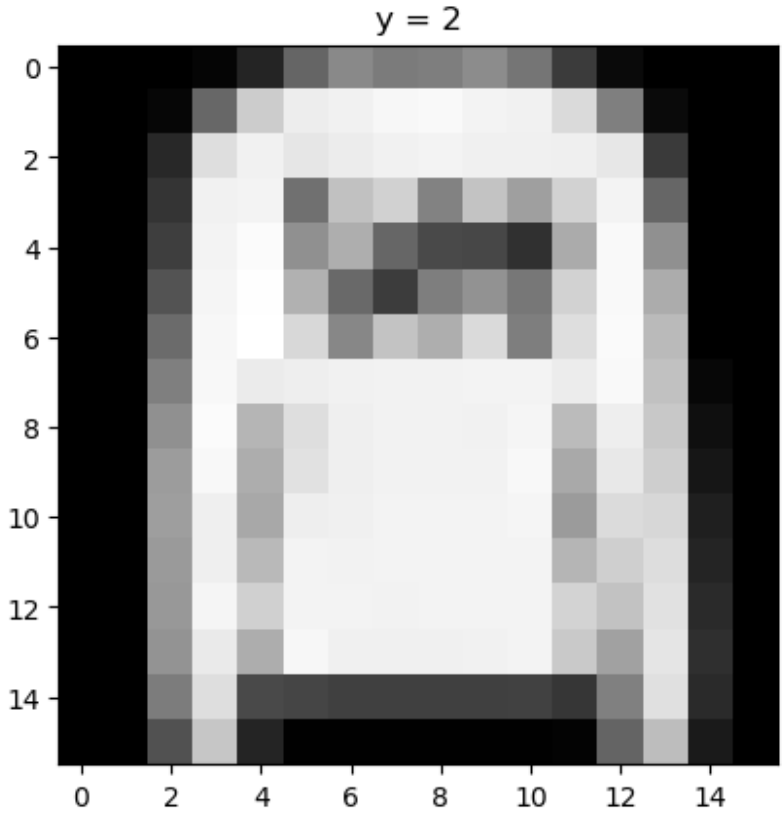
```
Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/train-images-idx
3-ubyte.gz
Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/train-images-idx
3-ubyte.gz to .fashion/data/FashionMNIST/raw/train-images-idx3-ubyte.gz
100%|██████████| 26421880/26421880 [00:00<00:00, 80876713.08it/s]
```

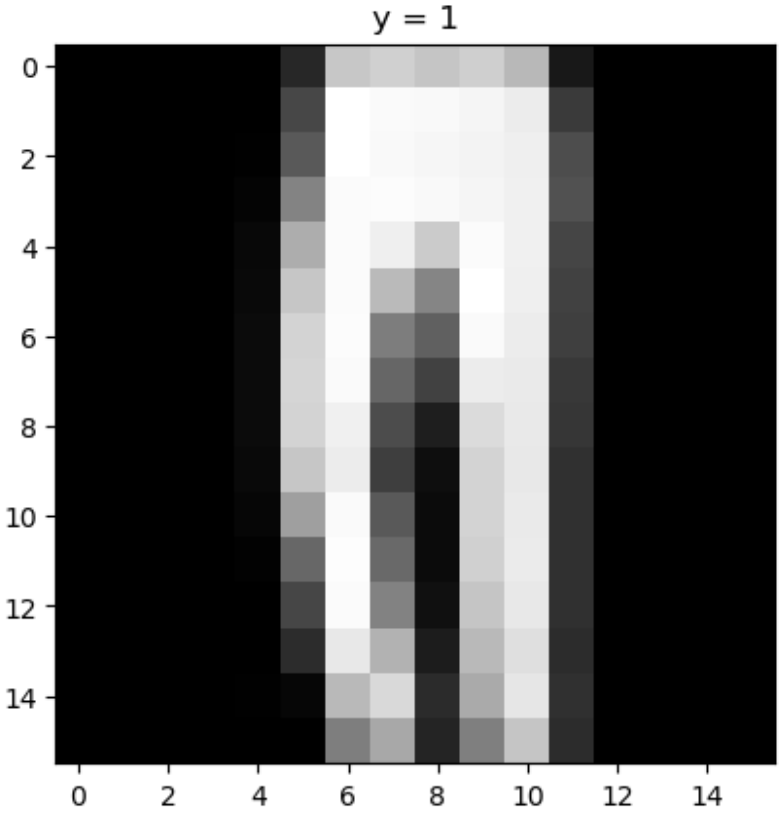
```
Extracting .fashion/data/FashionMNIST/raw/train-images-idx3-ubyte.gz to .fashion/data/F
ashionMNIST/raw

Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/train-labels-idx
1-ubyte.gz
Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/train-labels-idx
1-ubyte.gz to .fashion/data/FashionMNIST/raw/train-labels-idx1-ubyte.gz
100%|██████████| 29515/29515 [00:00<00:00, 3342645.67it/s]
Extracting .fashion/data/FashionMNIST/raw/train-labels-idx1-ubyte.gz to .fashion/data/F
ashionMNIST/raw

Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/t10k-images-idx3
-ubyte.gz

Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/t10k-images-idx3
-ubyte.gz to .fashion/data/FashionMNIST/raw/t10k-images-idx3-ubyte.gz
100%|██████████| 4422102/4422102 [00:00<00:00, 31627461.43it/s]
Extracting .fashion/data/FashionMNIST/raw/t10k-images-idx3-ubyte.gz to .fashion/data/Fa
shionMNIST/raw

Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/t10k-labels-idx1
-ubyte.gz
Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/t10k-labels-idx1
-ubyte.gz to .fashion/data/FashionMNIST/raw/t10k-labels-idx1-ubyte.gz
100%|██████████| 5148/5148 [00:00<00:00, 6121995.18it/s]
Extracting .fashion/data/FashionMNIST/raw/t10k-labels-idx1-ubyte.gz to .fashion/data/Fa
shionMNIST/raw
```

In [7]:

```
for n,data_sample in enumerate(dataset_val):

    show_data(data_sample)
    plt.show()
    if n==2:
        break
```


y = 9
0
2
4
6
8
10
12
14
0
2
4
6
8
10
12
14


y = 2
0
2
4
6
8
10
12
14
0
2
4
6
8
10
12
14




## Questions 2

Create a Convolutional Neural Network class using ONE of the following constructors. Train the network using the provided code then provide a screenshot of your training cost and accuracy with your validation data.

Constructor using Batch Norm

In [8]:
```
class CNN_batch(nn.Module):

    # Contructor
    def __init__(self, out_1=16, out_2=32,number_of_classes=10):
        super(CNN_batch, self).__init__()
        self.cnn1 = nn.Conv2d(in_channels=1, out_channels=out_1, kernel_size=5, padding
        self.conv1_bn = nn.BatchNorm2d(out_1)

        self.maxpool1=nn.MaxPool2d(kernel_size=2)

        self.cnn2 = nn.Conv2d(in_channels=out_1, out_channels=out_2, kernel_size=5, str
        self.conv2_bn = nn.BatchNorm2d(out_2)

        self.maxpool2=nn.MaxPool2d(kernel_size=2)
        self.fc1 = nn.Linear(out_2 * 4 * 4, number_of_classes)
        self.bn_fc1 = nn.BatchNorm1d(10)

    # Prediction
    def forward(self, x):
        x = self.cnn1(x)
        x=self.conv1_bn(x)
        x = torch.relu(x)
        x = self.maxpool1(x)
        x = self.cnn2(x)
        x=self.conv2_bn(x)
```

```
        x = torch.relu(x)
        x = self.maxpool2(x)
        x = x.view(x.size(0), -1)
        x = self.fc1(x)
        x=self.bn_fc1(x)
        return x
```

Constructor for regular Convolutional Neural Network

In [9]:
```
class CNN(nn.Module):

    # Contructor
    def __init__(self, out_1=16, out_2=32,number_of_classes=10):
        super(CNN, self).__init__()
        self.cnn1 = nn.Conv2d(in_channels=1, out_channels=out_1, kernel_size=5, padding
        self.maxpool1=nn.MaxPool2d(kernel_size=2)

        self.cnn2 = nn.Conv2d(in_channels=out_1, out_channels=out_2, kernel_size=5, str
        self.maxpool2=nn.MaxPool2d(kernel_size=2)
        self.fc1 = nn.Linear(out_2 * 4 * 4, number_of_classes)

    # Prediction
    def forward(self, x):
        x = self.cnn1(x)
        x = torch.relu(x)
        x = self.maxpool1(x)
        x = self.cnn2(x)
        x = torch.relu(x)
        x = self.maxpool2(x)
        x = x.view(x.size(0), -1)
        x = self.fc1(x)
        return x
```

train loader and validation loader

In [10]:
```
train_loader = torch.utils.data.DataLoader(dataset=dataset_train, batch_size=100 )
test_loader = torch.utils.data.DataLoader(dataset=dataset_val, batch_size=100 )
```

Convolutional Neural Network object

In [11]:
```
#model = CNN(out_1=16, out_2=32,number_of_classes=10)
model =CNN_batch(out_1=16, out_2=32,number_of_classes=10)
```

Create the objects for the criterion and the optimizer named `criterion` and `optimizer`. Make the optimizer use SGD with a learning rate of 0.1 and the optimizer use Cross Entropy Loss

In [13]:
```
criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.SGD(model.parameters(), lr = 0.1)
```

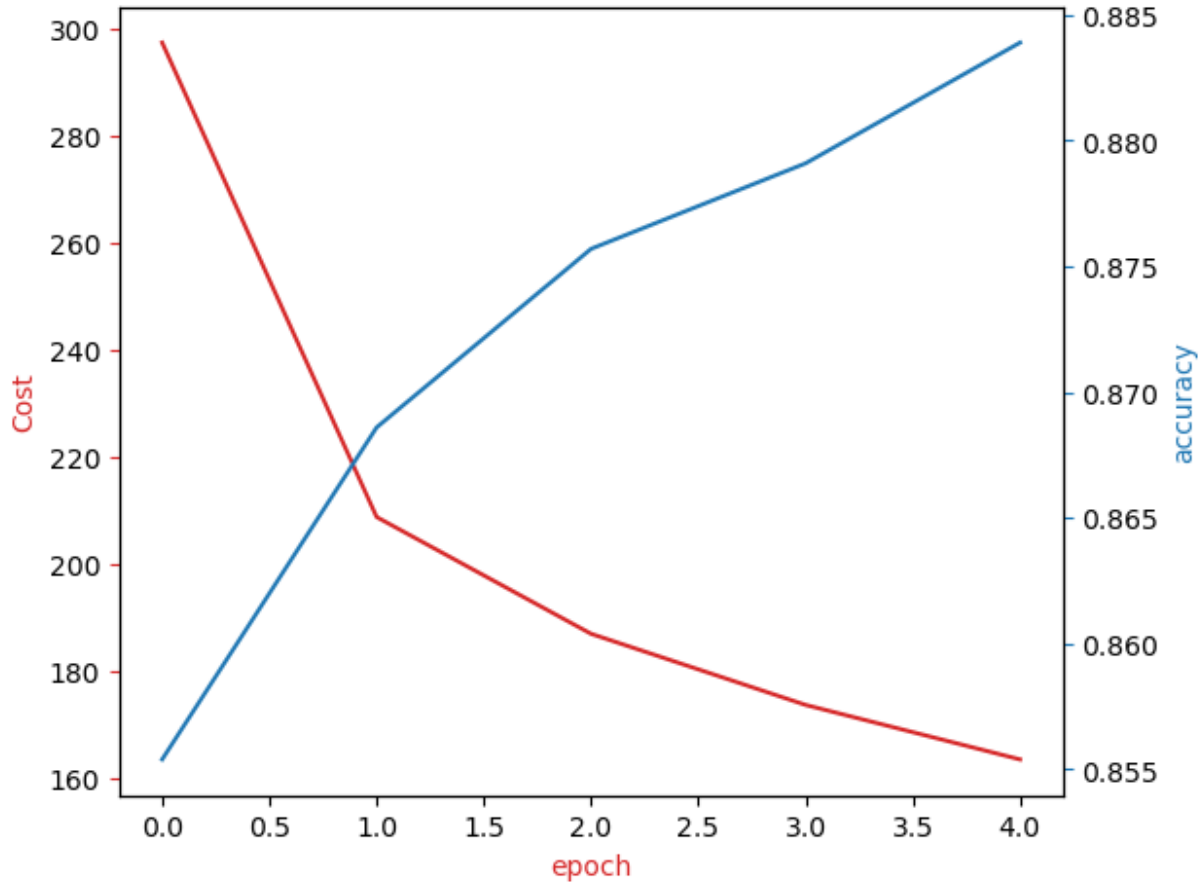
Code used to train the model

In [14]:
```
import time
start_time = time.time()

cost_list=[]
accuracy_list=[]
N_test=len(dataset_val)
n_epochs=5
```

```
for epoch in range(n_epochs):
    cost=0
    model.train()
    for x, y in train_loader:
        optimizer.zero_grad()
        z = model(x)
        loss = criterion(z, y)
        loss.backward()
        optimizer.step()
        cost+=loss.item()
    correct=0
    #perform a prediction on the validation  data
    model.eval()
    for x_test, y_test in test_loader:
        z = model(x_test)
        _, yhat = torch.max(z.data, 1)
        correct += (yhat == y_test).sum().item()
    accuracy = correct / N_test
    accuracy_list.append(accuracy)
    cost_list.append(cost)
```

You will use the following to plot the Cost and accuracy for each epoch for the training and testing data, respectively.

In [15]:
```
fig, ax1 = plt.subplots()
color = 'tab:red'
ax1.plot(cost_list, color=color)
ax1.set_xlabel('epoch', color=color)
ax1.set_ylabel('Cost', color=color)
ax1.tick_params(axis='y', color=color)

ax2 = ax1.twinx()
color = 'tab:blue'
ax2.set_ylabel('accuracy', color=color)
ax2.set_xlabel('epoch', color=color)
ax2.plot( accuracy_list, color=color)
ax2.tick_params(axis='y', color=color)
fig.tight_layout()
```




dataset: https://github.com/zalandoresearch/fashion-mnist

## About the Authors:

Joseph Santarcangelo 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, Mavis Zhou

---