Deep-learning-Hand written Digit Recognition

```
!pip install jovian --upgrade --quiet
```

```
import jovian
```

```
# Execute this to save new versions of the notebook
jovian.commit(project="deep-learning-basic")
```

```
[jovian] Detected Colab notebook...
[jovian] Uploading colab notebook to Jovian...
Committed successfully! https://jovian.ai/btech60309-19/deep-learning-basic
```

```
'https://jovian.ai/btech60309-19/deep-learning-basic'
```

```
!pip install matplotlib
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/dist-packages (3.2.2)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib) (1.4.4)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib) (3.0.9)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages (from matplotlib) (0.11.0)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib) (2.8.2)
Requirement already satisfied: numpy>=1.11 in /usr/local/lib/python3.7/dist-packages (from matplotlib) (1.21.6)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.7/dist-packages (from kiwisolver>=1.0.1->matplotlib) (4.1.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from python-dateutil>=2.1->matplotlib) (1.15.0)
```

```
import torch
import torchvision
import jovian
import torch.nn as nn
import matplotlib.pyplot as plt
import torch.nn.functional as F
import torchvision.transforms as transforms
from torchvision.datasets import MNIST
from torch.utils.data import random_split
```

```
from torch.utils.data import DataLoader
%matplotlib inline
```

```
batch_size=128
learning_rate=0.001
input_size=28*28
num_classes=10
```

```
jovian.commit()
```

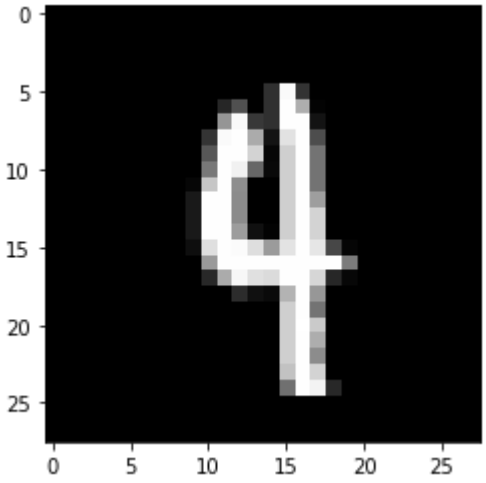
```
[jovian] Detected Colab notebook...
[jovian] Uploading colab notebook to Jovian...
Committed successfully! https://jovian.ai/btech60309-19/deep-learning-basic
```

```
'https://jovian.ai/btech60309-19/deep-learning-basic'
```

```
dataset=MNIST(root='data/',train=True,transform=transforms.ToTensor(),download=True)
train_ds,val_ds=random_split(dataset,[50000,10000])
test_ds=MNIST(root='data/',train=False,transform=transforms.ToTensor())
train_loader=DataLoader(train_ds,batch_size,shuffle=True)
val_loader=DataLoader(val_ds,batch_size*2)
test_loader=DataLoader(test_ds,batch_size*2)
```

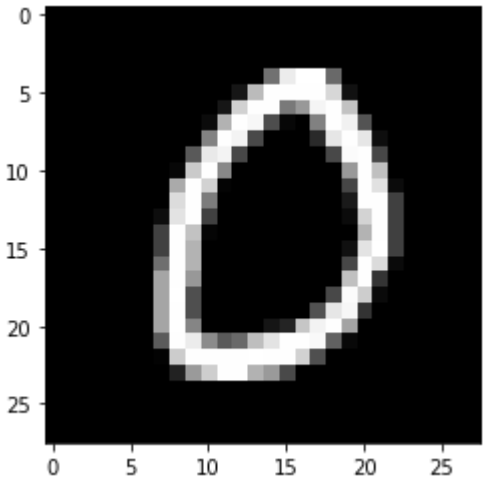
```
image,label=train_ds[0]
plt.imshow(image[0],cmap='gray')
print('Label : ',label)
```

```
Label :  4
```



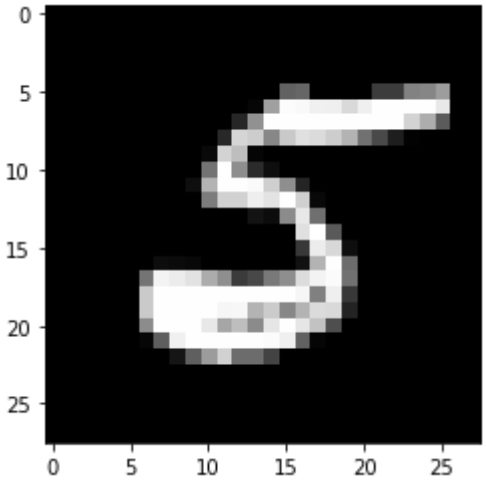

```
image,label=train_ds[100]
plt.imshow(image[0],cmap='gray')
print('Label : ',label)
```

```
Label :  0
```



```
image,label=train_ds[500]
plt.imshow(image[0],cmap='gray')
print('Label : ',label)
```

```
Label :  5
```



#Model

```
class MnistModel(nn.Module):
  def __init__(self):
    super().__init__()
    self.linear=nn.Linear(input_size,num_classes)

  def forward(self,xb):
    xb=xb.reshape(-1,784)
    out=self.linear(xb)
    return out

  def training_step(self,batch):
    images,labels=batch
    out=self(images)
    loss=F.cross_entropy(out,labels)
    return loss

  def validation_step(self,batch):
    images, labels = batch
```

```
    out = self(images)
    loss = F.cross_entropy(out, labels)
    acc = accuracy(out, labels)
    return {'val_loss': loss.detach(), 'val_acc': acc.detach()}

  def validation_epoch_end(self, outputs):
    batch_losses = [x['val_loss'] for x in outputs]
    epoch_loss = torch.stack(batch_losses).mean()   # Combine losses
    batch_accs = [x['val_acc'] for x in outputs]
    epoch_acc = torch.stack(batch_accs).mean()      # Combine accuracies
    return {'val_loss': epoch_loss.item(), 'val_acc': epoch_acc.item()}

  def epoch_end(self, epoch, result):
    print("Epoch [{}], val_loss: {:.4f}, val_acc: {:.4f}".format(epoch, result['val_los


model=MnistModel()
```

```
def accuracy(outputs,labels):
  _,preds=torch.max(outputs,dim=1)
  return torch.tensor(torch.sum(preds==labels).item()/len(preds))
```

```
def evaluate(model, val_loader):
    outputs = [model.validation_step(batch) for batch in val_loader]
    return model.validation_epoch_end(outputs)

def fit(epochs, lr, model, train_loader, val_loader, opt_func=torch.optim.SGD):
    history = []
    optimizer = opt_func(model.parameters(), lr)
    for epoch in range(epochs):
        # Training Phase
        for batch in train_loader:
            loss = model.training_step(batch)
            loss.backward()
            optimizer.step()
            optimizer.zero_grad()
        # Validation phase
        result = evaluate(model, val_loader)
        model.epoch_end(epoch, result)
        history.append(result)
    return history
```

```
evaluate(model, val_loader)
```

```
{'val_acc': 0.10185547173023224, 'val_loss': 2.297032594680786}
```

```
history = fit(5, 0.001, model, train_loader, val_loader)
```

```
Epoch [0], val_loss: 1.9385, val_acc: 0.6468
Epoch [1], val_loss: 1.6735, val_acc: 0.7355
```

```
Epoch [2], val_loss: 1.4753, val_acc: 0.7644
Epoch [3], val_loss: 1.3257, val_acc: 0.7813
Epoch [4], val_loss: 1.2106, val_acc: 0.7926
```

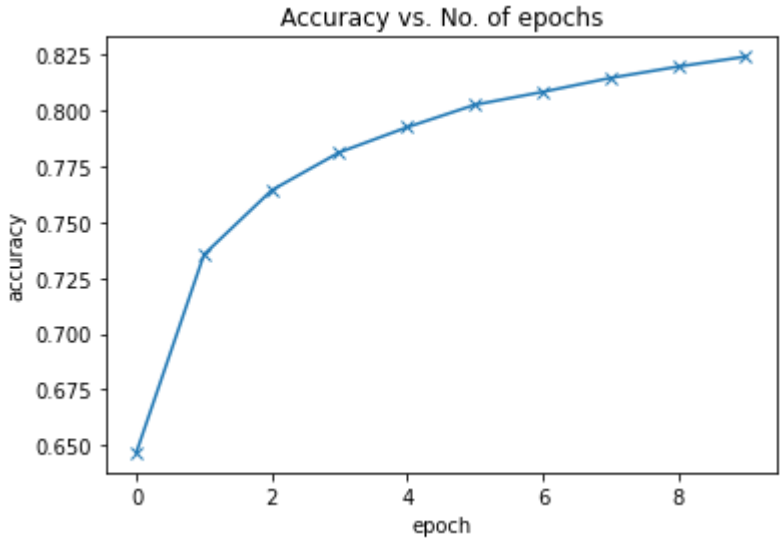
```
history1 = fit(5, 0.001, model, train_loader, val_loader)
```

```
Epoch [0], val_loss: 1.1201, val_acc: 0.8026
Epoch [1], val_loss: 1.0475, val_acc: 0.8084
Epoch [2], val_loss: 0.9881, val_acc: 0.8146
Epoch [3], val_loss: 0.9387, val_acc: 0.8197
Epoch [4], val_loss: 0.8969, val_acc: 0.8242
```

```
history3 = fit(10, 0.001, model, train_loader, val_loader)
```

```
Epoch [0], val_loss: 0.8610, val_acc: 0.8280
Epoch [1], val_loss: 0.8300, val_acc: 0.8311
Epoch [2], val_loss: 0.8027, val_acc: 0.8357
Epoch [3], val_loss: 0.7787, val_acc: 0.8380
Epoch [4], val_loss: 0.7572, val_acc: 0.8394
Epoch [5], val_loss: 0.7380, val_acc: 0.8418
Epoch [6], val_loss: 0.7207, val_acc: 0.8455
Epoch [7], val_loss: 0.7049, val_acc: 0.8470
Epoch [8], val_loss: 0.6905, val_acc: 0.8482
Epoch [9], val_loss: 0.6773, val_acc: 0.8497
```

```
accuracies = [r['val_acc'] for r in history+history1]
plt.plot(accuracies, '-x')
plt.xlabel('epoch')
plt.ylabel('accuracy')
plt.title('Accuracy vs. No. of epochs');
```




```
result = evaluate(model, test_loader)
result
```

```
{'val_acc': 0.856738269329071, 'val_loss': 0.6411406993865967}
```

```
jovian.commit()
```

```
[jovian] Detected Colab notebook...
[jovian] Uploading colab notebook to Jovian...
Committed successfully! https://jovian.ai/btech60309-19/deep-learning-basic
```

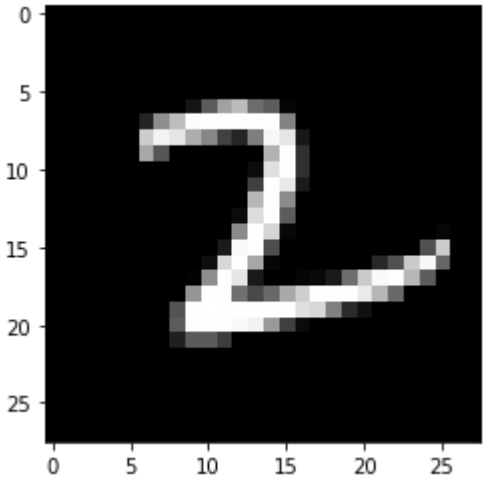
```
'https://jovian.ai/btech60309-19/deep-learning-basic'
```

#Prediction

```
def predict_image(img, model):
    xb = img.unsqueeze(0)
    yb = model(xb)
    _, preds  = torch.max(yb, dim=1)
    return preds[0].item()
```

```
img, label = test_ds[919]
plt.imshow(img[0], cmap='gray')
print('Label:', label, ', Predicted:', predict_image(img, model))
```

```
Label: 2 , Predicted: 6
```




```
torch.save(model.state_dict(), 'mnist-logistic.pth')
```

```
jovian.commit(project='mnist-logistic-minimal', environment=None, outputs=['mnist-logis
jovian.commit(project='mnist-logistic-minimal', environment=None, outputs=['mnist-logis
```

```
[jovian] Detected Colab notebook...
[jovian] Uploading colab notebook to Jovian...
[jovian] Uploading additional outputs...
Committed successfully! https://jovian.ai/btech60309-19/mnist-logistic-minimal
[jovian] Detected Colab notebook...
[jovian] Uploading colab notebook to Jovian...
```

```
[jovian] Uploading additional outputs...
Committed successfully! https://jovian.ai/btech60309-19/mnist-logistic-minimal
```

'https://jovian.ai/btech60309-19/mnist-logistic-minimal'

```
jovian.commit()
```

```
[jovian] Detected Colab notebook...
[jovian] Uploading colab notebook to Jovian...
Committed successfully! https://jovian.ai/btech60309-19/deep-learning-hand-written-digit-recognition
```

'https://jovian.ai/btech60309-19/deep-learning-hand-written-digit-recognition'

```

```