Neural Network

Import Libraries and Dataset

In [ ]:

```
# Import Libraries

import numpy as np
import pandas as pd
import torch
import torchvision
from torchvision import datasets
from torchvision.transforms import ToTensor
from torch.utils.data import DataLoader
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
from torch.nn import CrossEntropyLoss
import matplotlib.pyplot as plt
```

In [ ]:

```
# Import dataset

train_dir = pd.read_csv("E:/Term 7/Intro to Machine Learning/digit-recognizer/train.csv")
test_dir = pd.read_csv("E:/Term 7/Intro to Machine Learning/digit-recognizer/test.csv")
```

In [ ]:

```
train_dir = datasets.MNIST(
    root='archive',
    train=True,
    transform= ToTensor(),
    download=True
)
test_dir = datasets.MNIST(
    root='archive',
    train=False,
    transform= ToTensor(),
    download=True
)
```

In [ ]:

```
train_dir
```

Out[ ]:

```
Dataset MNIST
    Number of datapoints: 60000
    Root location: archive
    Split: Train
    StandardTransform
Transform: ToTensor()
```

In [ ]:

```
test_dir
```

Out[ ]:

```
Dataset MNIST
    Number of datapoints: 10000
    Root location: archive
    Split: Test
    StandardTransform
Transform: ToTensor()
```

Train Test Split

In [ ]:

```
from sklearn.model_selection import train_test_split
X_train,X_val= train_test_split(train_dir,test_size=0.2,random_state=0)
```

In [ ]:

```
loaders = {
    'train': DataLoader(X_train,
                        batch_size=50,
                        shuffle=True,
                        num_workers=1),

    'test': DataLoader(X_val,
                        batch_size=10,
                        shuffle=True,
                        num_workers=1),
}
```

In [ ]:

```
loaders
```

Out[ ]:

```
{'train': <torch.utils.data.dataloader.DataLoader at 0x20fbec38690>,
 'test': <torch.utils.data.dataloader.DataLoader at 0x21013c36650>}
```

Building The Neural Network

In [ ]:

```
class CNN(nn.Module):
    def __init__(self):
        super(CNN,self).__init__()
        self.conv1 = nn.Conv2d(1,10,kernel_size=5)
        self.conv2 = nn.Conv2d(10,20,kernel_size=5)
        self.conv2_drop = nn.Dropout2d()
        self.fc1 = nn.Linear(320,50)
        self.fc2 = nn.Linear(50,10)

    def forward(self,x):
        x = F.relu(F.max_pool2d(self.conv1(x),2))
        x = F.relu(F.max_pool2d(self.conv2_drop(self.conv2(x)),2))
        x = F.normalize(x)
        x = x.view(-1,320)
        x= F.relu(self.fc1(x))
        x = F.dropout(x,training=self.training)
        x = self.fc2(x)

        return F.softmax(x)
```

In [ ]:

```
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

model = CNN().to(device)
optimizer = optim.SGD(model.parameters(),lr=0.01,momentum=0.9)
loss_fn = CrossEntropyLoss()

def train(epoch):
    model.train()
    for batch_index, (data,target) in enumerate(loaders['train']):
        data, target = data.to(device), target.to(device)
        optimizer.zero_grad()
        output = model(data)
        loss = loss_fn(output,target)
        loss.backward()
        optimizer.step()
        if batch_index % 1000 == 0:
            print(f"Training Epoch: {epoch} [{batch_index*len(data)} /{len(loaders['train'].dataset)} ({100.*batch_index/len(
```

In [ ]:

```
loss =[]
acc = []
def test():
    model.eval()
    test_loss= 0
    correct = 0

    with torch.no_grad():
        for data, target in loaders['test']:
            data,target = data.to(device),target.to(device)
            output = model(data)
            test_loss += loss_fn(output,target).item()
            pred = output.argmax(dim=1,keepdim=True)
            correct += pred.eq(target.view_as(pred)).sum().item()
            Accuracy = correct / len(loaders['test'].dataset)


    test_loss /= len(loaders['test'].dataset)
    print(f"\nTest set: Average loss: {test_loss:.4f}, Accuracy {Accuracy} ({100. * correct/len(loaders['test'].dataset):.0f}
    loss.append(test_loss)
    acc.append(Accuracy)

    return (loss,acc)
```

Training The Model

In [ ]:

```
for epoch in range(1,11):
    train(epoch)
    test()
```

```
C:\Users\ADMIN\AppData\Local\Temp\ipykernel_4972\1132881025.py:20: UserWarning: Implicit dimension choice for softmax has been
deprecated. Change the call to include dim=X as an argument.
  return F.softmax(x)
```

```
Training Epoch: 1 [0 /48000 (0%)]       2.302388

Test set: Average loss: 0.2020, Accuracy 0.558 (56%)

Training Epoch: 2 [0 /48000 (0%)]       2.122140

Test set: Average loss: 0.1773, Accuracy 0.6851666666666667 (69%)

Training Epoch: 3 [0 /48000 (0%)]       1.828355

Test set: Average loss: 0.1684, Accuracy 0.7769166666666667 (78%)

Training Epoch: 4 [0 /48000 (0%)]       1.734128

Test set: Average loss: 0.1606, Accuracy 0.8611666666666666 (86%)

Training Epoch: 5 [0 /48000 (0%)]       1.712794

Test set: Average loss: 0.1587, Accuracy 0.8750833333333333 (88%)

Training Epoch: 6 [0 /48000 (0%)]       1.679074

Test set: Average loss: 0.1583, Accuracy 0.8776666666666667 (88%)

Training Epoch: 7 [0 /48000 (0%)]       1.678664

Test set: Average loss: 0.1580, Accuracy 0.8811666666666667 (88%)

Training Epoch: 8 [0 /48000 (0%)]       1.625008

Test set: Average loss: 0.1577, Accuracy 0.8833333333333333 (88%)

Training Epoch: 9 [0 /48000 (0%)]       1.677914

Test set: Average loss: 0.1574, Accuracy 0.8854166666666666 (89%)

Training Epoch: 10 [0 /48000 (0%)]      1.598310

Test set: Average loss: 0.1505, Accuracy 0.9606666666666667 (96%)
```

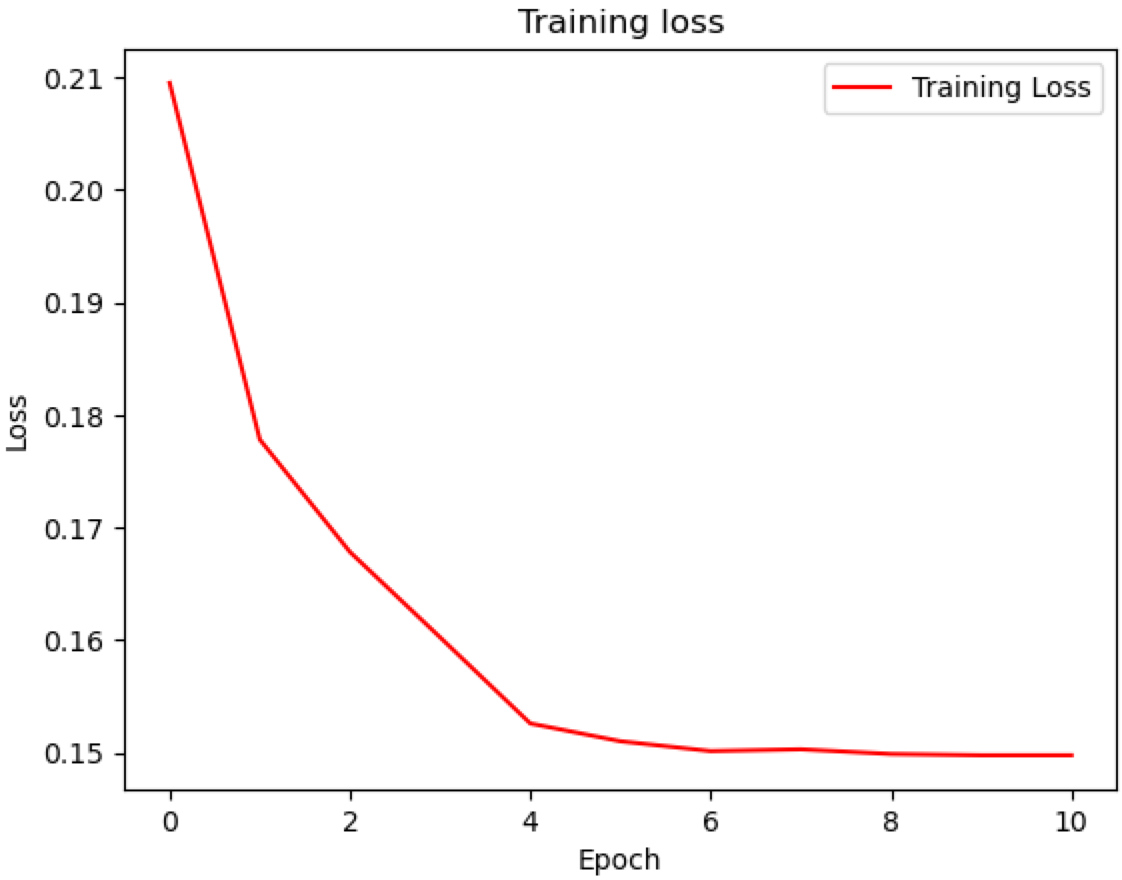
Plot Loss And Accuracy Graph

In [ ]:

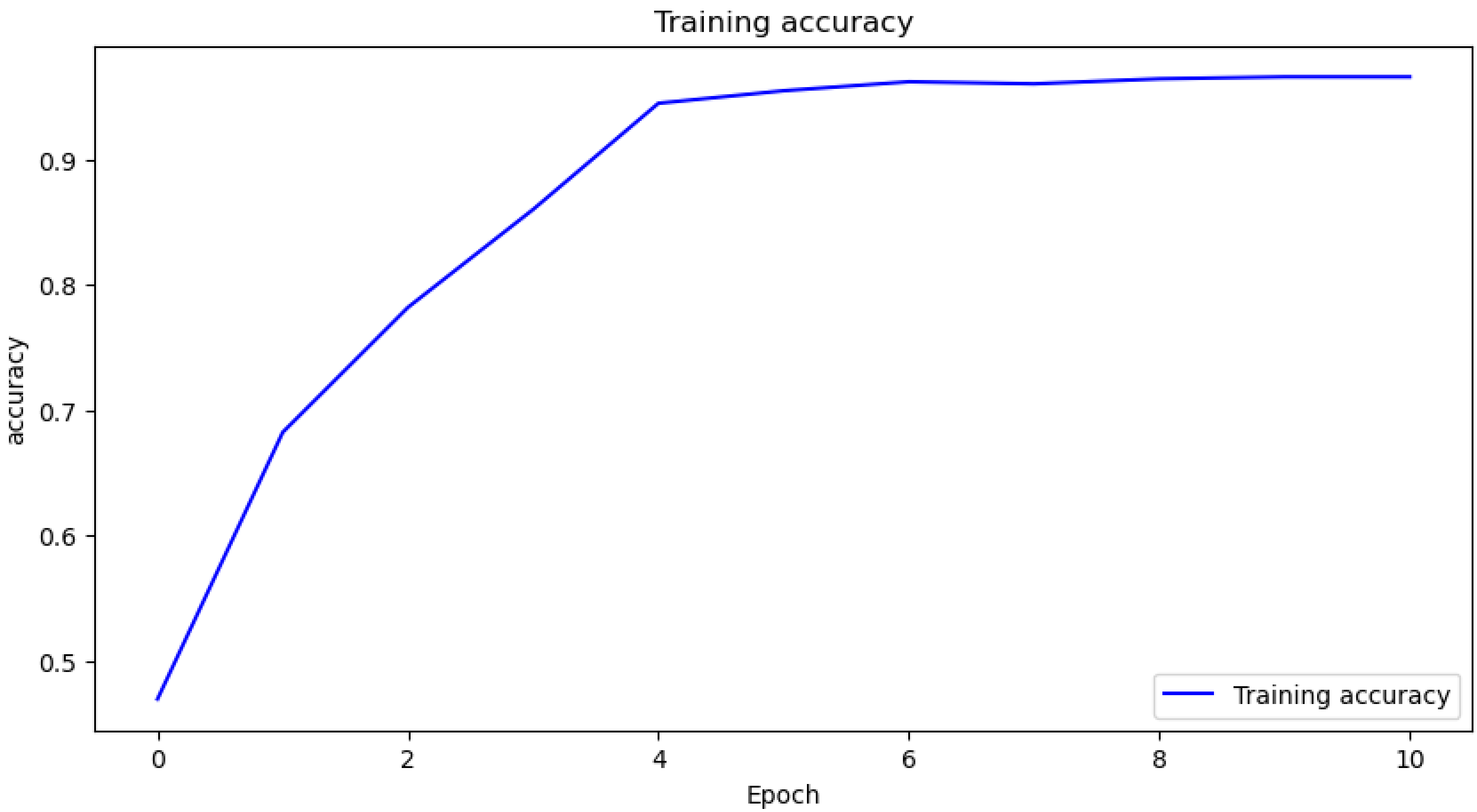
```
(loss,acc)= test()
epochs=range(len(loss))
```

```
plt.plot(epochs, loss, 'r', label="Training Loss")
plt.legend(loc='upper right')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title('Training loss')

plt.show()
```

```
C:\Users\ADMIN\AppData\Local\Temp\ipykernel_4972\1132881025.py:20: UserWarning: Implicit dimension choice for softmax has been
deprecated. Change the call to include dim=X as an argument.
  return F.softmax(x)
Test set: Average loss: 0.1498, Accuracy 0.966 (97%)
```



```
In [ ]: epochs=range(len(acc))
        fig = plt.figure(figsize=(10,5))
        plt.plot(epochs, acc, 'b', label="Training accuracy")
        plt.xlabel('Epoch')
        plt.ylabel('accuracy')
        plt.title('Training accuracy')
        plt.legend(loc='lower right')
        plt.show()
```



Building The Neural Network Without Dropout And Normalization Layer

In [ ]:

```
class CNN(nn.Module):
    def __init__(self):
        super(CNN,self).__init__()

        self.conv1 = nn.Conv2d(1,10,kernel_size=5)
        self.conv2 = nn.Conv2d(10,20,kernel_size=5)
        self.fc1 = nn.Linear(320,50)
        self.fc2 = nn.Linear(50,10)

    def forward(self,x):
        x = F.relu(F.max_pool2d(self.conv1(x),2))
        x = F.relu(F.max_pool2d(self.conv2(x),2))
        x = x.view(-1,320)
        x= F.relu(self.fc1(x))
        x = F.dropout(x,training=self.training)
        x = self.fc2(x)

        return F.softmax(x)
```

In [ ]:

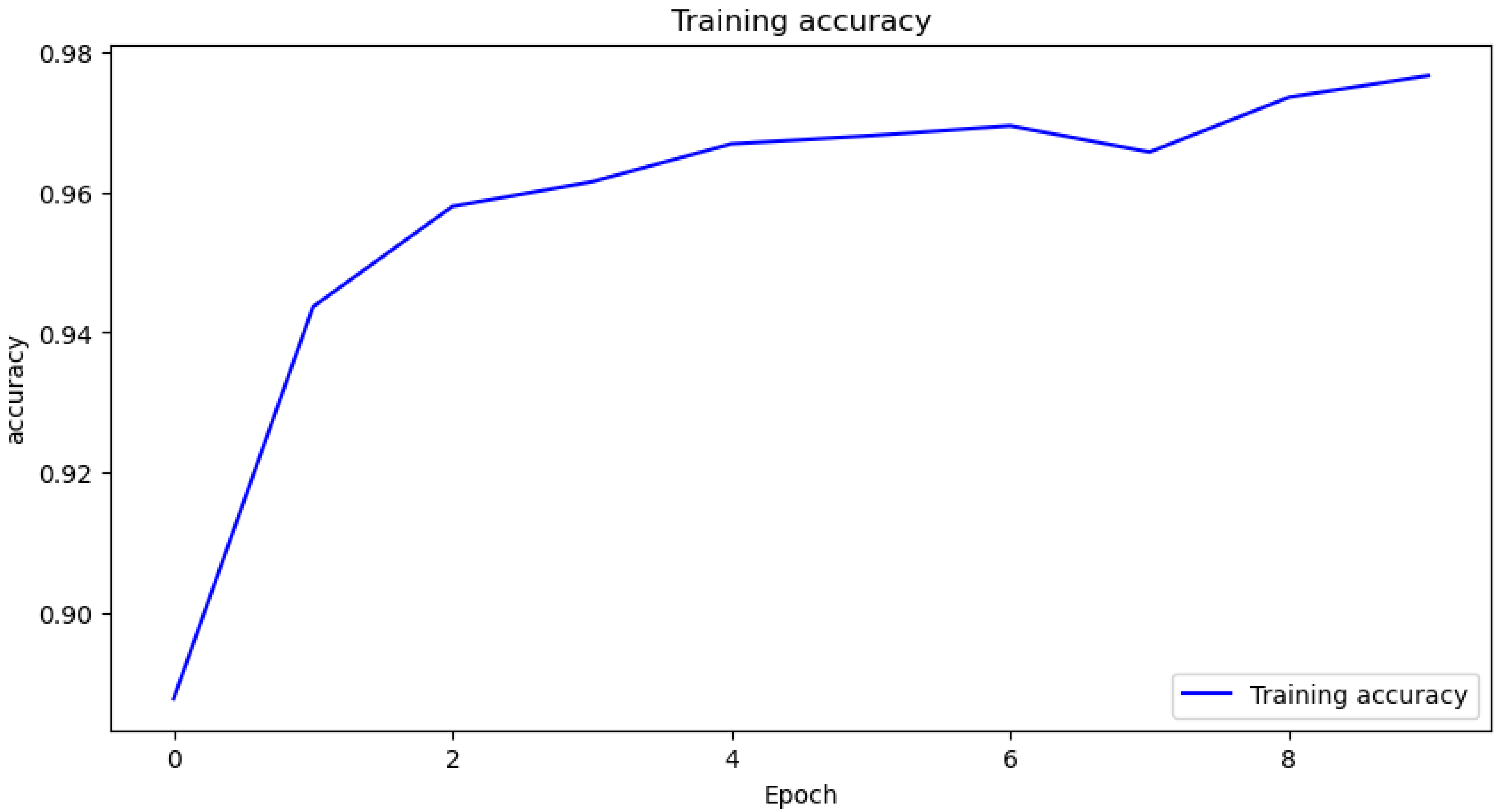
```
for epoch in range(1,11):
    train(epoch)
    test()
```

```
C:\Users\ADMIN\AppData\Local\Temp\ipykernel_4972\2097859788.py:18: UserWarning: Implicit dimension choice for softmax has been
deprecated. Change the call to include dim=X as an argument.
  return F.softmax(x)
```

```
Training Epoch: 1 [0 /48000 (0%)]       2.302093

Test set: Average loss: 0.1585, Accuracy 0.8876666666666667 (89%)

Training Epoch: 2 [0 /48000 (0%)]       1.752460

Test set: Average loss: 0.1521, Accuracy 0.9436666666666667 (94%)

Training Epoch: 3 [0 /48000 (0%)]       1.654783

Test set: Average loss: 0.1505, Accuracy 0.958 (96%)

Training Epoch: 4 [0 /48000 (0%)]       1.549115

Test set: Average loss: 0.1499, Accuracy 0.9615 (96%)

Training Epoch: 5 [0 /48000 (0%)]       1.502136

Test set: Average loss: 0.1495, Accuracy 0.9669166666666666 (97%)

Training Epoch: 6 [0 /48000 (0%)]       1.530154

Test set: Average loss: 0.1493, Accuracy 0.9680833333333333 (97%)

Training Epoch: 7 [0 /48000 (0%)]       1.570755

Test set: Average loss: 0.1493, Accuracy 0.9695 (97%)

Training Epoch: 8 [0 /48000 (0%)]       1.542116

Test set: Average loss: 0.1495, Accuracy 0.96575 (97%)

Training Epoch: 9 [0 /48000 (0%)]       1.522718

Test set: Average loss: 0.1488, Accuracy 0.9735833333333334 (97%)

Training Epoch: 10 [0 /48000 (0%)]      1.478524

Test set: Average loss: 0.1485, Accuracy 0.9766666666666667 (98%)
```

In [ ]:

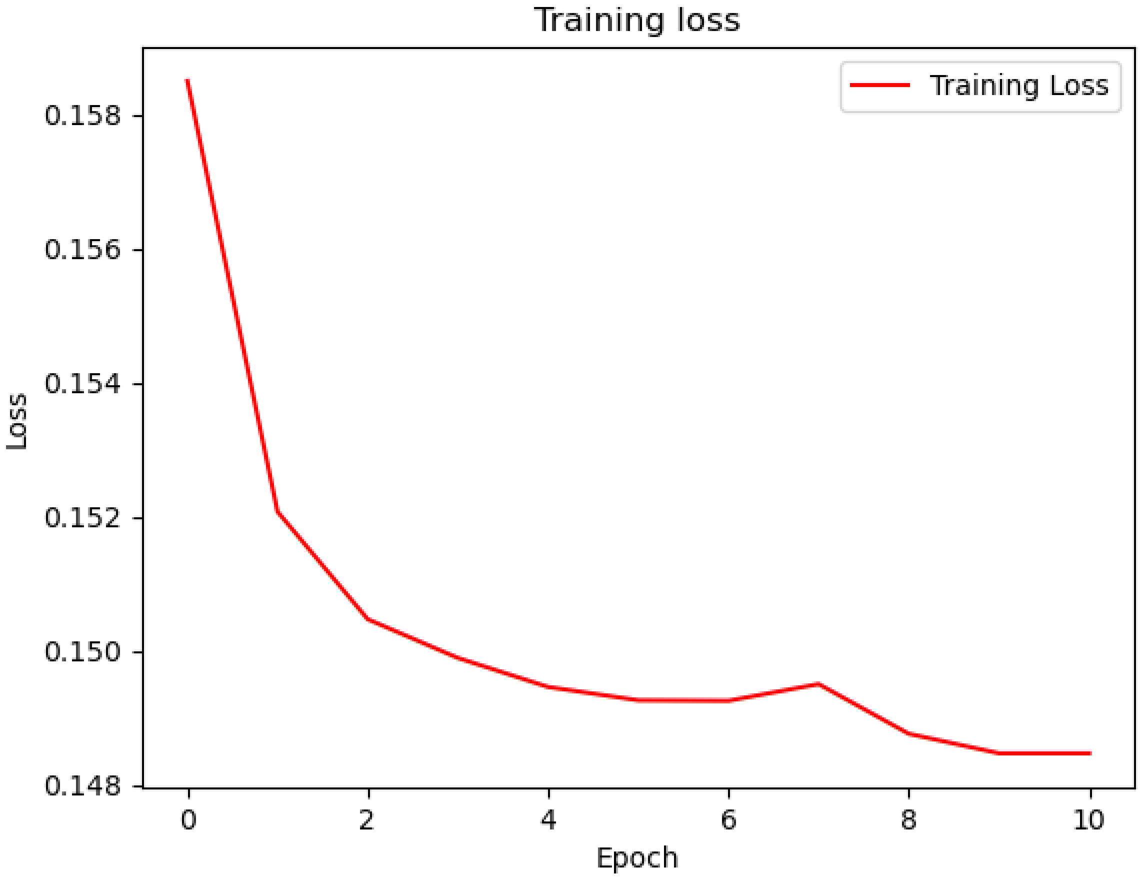
```python
epochs=range(len(acc))
fig = plt.figure(figsize=(10,5))
plt.plot(epochs, acc, 'b', label="Training accuracy")
plt.xlabel('Epoch')
plt.ylabel('accuracy')
plt.title('Training accuracy')
plt.legend(loc='lower right')
plt.show()
```



In [ ]:

```
(loss,acc)= test()
epochs=range(len(loss))
plt.plot(epochs, loss, 'r', label="Training Loss")
plt.legend(loc='upper right')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title('Training loss')

plt.show()
```

```
C:\Users\ADMIN\AppData\Local\Temp\ipykernel_4972\2097859788.py:18: UserWarning: Implicit dimension choice for softmax has been
deprecated. Change the call to include dim=X as an argument.
  return F.softmax(x)
Test set: Average loss: 0.1485, Accuracy 0.9766666666666667 (98%)
```



Applying different batch sizes

In [ ]:

```
loaders = {
    'train': DataLoader(X_train,
                        batch_size=100,
                        shuffle=True,
                        num_workers=1),

    'test': DataLoader(X_val,
                       batch_size=100,
                       shuffle=True,
                        num_workers=1),
}
```

In [ ]:

```
for epoch in range(1,11):
    train(epoch)
    test()
```

```
C:\Users\ADMIN\AppData\Local\Temp\ipykernel_4972\1132881025.py:20: UserWarning: Implicit dimension choice for softmax has been
deprecated. Change the call to include dim=X as an argument.
  return F.softmax(x)
```

```
Training Epoch: 1 [0 /48000 (0%)]       2.301568

Test set: Average loss: 0.0223, Accuracy 0.24691666666666667 (25%)

Training Epoch: 2 [0 /48000 (0%)]       2.262717

Test set: Average loss: 0.0210, Accuracy 0.36425 (36%)

Training Epoch: 3 [0 /48000 (0%)]       2.147787

Test set: Average loss: 0.0183, Accuracy 0.6715833333333333 (67%)

Training Epoch: 4 [0 /48000 (0%)]       2.031319

Test set: Average loss: 0.0177, Accuracy 0.6841666666666667 (68%)

Training Epoch: 5 [0 /48000 (0%)]       1.845222

Test set: Average loss: 0.0174, Accuracy 0.7159166666666666 (72%)

Training Epoch: 6 [0 /48000 (0%)]       1.826994

Test set: Average loss: 0.0168, Accuracy 0.7796666666666666 (78%)

Training Epoch: 7 [0 /48000 (0%)]       1.753381

Test set: Average loss: 0.0168, Accuracy 0.7831666666666667 (78%)

Training Epoch: 8 [0 /48000 (0%)]       1.705883

Test set: Average loss: 0.0167, Accuracy 0.78575 (79%)

Training Epoch: 9 [0 /48000 (0%)]       1.723947

Test set: Average loss: 0.0167, Accuracy 0.7883333333333333 (79%)

Training Epoch: 10 [0 /48000 (0%)]      1.804975

Test set: Average loss: 0.0167, Accuracy 0.78875 (79%)
```

In [ ]:

```
loaders = {
    'train': DataLoader(X_train,
                        batch_size=500,
                        shuffle=True,
                        num_workers=1),

    'test': DataLoader(X_val,
                        batch_size=100,
                        shuffle=True,
                        num_workers=1),
}
```

In [ ]:

```
for epoch in range(1,11):
    train(epoch)
    test()
```

```
C:\Users\ADMIN\AppData\Local\Temp\ipykernel_4972\1132881025.py:20: UserWarning: Implicit dimension choice for softmax has been
deprecated. Change the call to include dim=X as an argument.
  return F.softmax(x)
```

```
Training Epoch: 1 [0 /48000 (0%)]       2.302694

Test set: Average loss: 0.0230, Accuracy 0.11675 (12%)

Training Epoch: 2 [0 /48000 (0%)]       2.302188

Test set: Average loss: 0.0230, Accuracy 0.16166666666666665 (16%)

Training Epoch: 3 [0 /48000 (0%)]       2.300249

Test set: Average loss: 0.0230, Accuracy 0.30633333333333335 (31%)

Training Epoch: 4 [0 /48000 (0%)]       2.297926

Test set: Average loss: 0.0229, Accuracy 0.427 (43%)

Training Epoch: 5 [0 /48000 (0%)]       2.295523

Test set: Average loss: 0.0228, Accuracy 0.5090833333333333 (51%)

Training Epoch: 6 [0 /48000 (0%)]       2.289192

Test set: Average loss: 0.0227, Accuracy 0.5304166666666666 (53%)

Training Epoch: 7 [0 /48000 (0%)]       2.280296

Test set: Average loss: 0.0224, Accuracy 0.4300833333333333 (43%)

Training Epoch: 8 [0 /48000 (0%)]       2.262898

Test set: Average loss: 0.0219, Accuracy 0.35183333333333333 (35%)

Training Epoch: 9 [0 /48000 (0%)]       2.236345

Test set: Average loss: 0.0214, Accuracy 0.35425 (35%)

Training Epoch: 10 [0 /48000 (0%)]      2.187921

Test set: Average loss: 0.0209, Accuracy 0.38708333333333333 (39%)
```

In [ ]:

```
loaders = {
    'train': DataLoader(X_train,
                        batch_size=50,
                        shuffle=True,
                        num_workers=1),

    'test': DataLoader(X_val,
                        batch_size=50,
                        shuffle=True,
                        num_workers=1),
}
```

In [ ]:

```
for epoch in range(1,11):
    train(epoch)
    test()
```

```
C:\Users\ADMIN\AppData\Local\Temp\ipykernel_4972\1132881025.py:20: UserWarning: Implicit dimension choice for softmax has been
deprecated. Change the call to include dim=X as an argument.
  return F.softmax(x)
```

```
Training Epoch: 1 [0 /48000 (0%)]       2.304991

Test set: Average loss: 0.0405, Accuracy 0.4701666666666667 (47%)

Training Epoch: 2 [0 /48000 (0%)]       2.081878

Test set: Average loss: 0.0341, Accuracy 0.7659166666666667 (77%)

Training Epoch: 3 [0 /48000 (0%)]       1.854149

Test set: Average loss: 0.0337, Accuracy 0.7734166666666666 (77%)

Training Epoch: 4 [0 /48000 (0%)]       1.837266

Test set: Average loss: 0.0335, Accuracy 0.7835 (78%)

Training Epoch: 5 [0 /48000 (0%)]       1.781923

Test set: Average loss: 0.0321, Accuracy 0.86325 (86%)

Training Epoch: 6 [0 /48000 (0%)]       1.703944

Test set: Average loss: 0.0318, Accuracy 0.8745833333333334 (87%)

Training Epoch: 7 [0 /48000 (0%)]       1.594163

Test set: Average loss: 0.0316, Accuracy 0.8823333333333333 (88%)

Training Epoch: 8 [0 /48000 (0%)]       1.710352

Test set: Average loss: 0.0317, Accuracy 0.8781666666666667 (88%)

Training Epoch: 9 [0 /48000 (0%)]       1.743715

Test set: Average loss: 0.0315, Accuracy 0.886 (89%)

Training Epoch: 10 [0 /48000 (0%)]      1.696156

Test set: Average loss: 0.0315, Accuracy 0.888 (89%)
```

In [ ]:

```
loaders = {
    'train': DataLoader(X_train,
                        batch_size=20,
                        shuffle=True,
                        num_workers=1),

    'test': DataLoader(X_val,
                        batch_size=20,
                        shuffle=True,
                        num_workers=1),
}
```

In [ ]:

```
for epoch in range(1,11):
    train(epoch)
    test()
```

```
C:\Users\ADMIN\AppData\Local\Temp\ipykernel_4972\1132881025.py:20: UserWarning: Implicit dimension choice for softmax has been
deprecated. Change the call to include dim=X as an argument.
  return F.softmax(x)
```

```
Training Epoch: 1 [0 /48000 (0%)]       2.303238
Training Epoch: 1 [20000 /48000 (42%)]  2.247491
Training Epoch: 1 [40000 /48000 (83%)]  1.889436

Test set: Average loss: 0.0874, Accuracy 0.7485 (75%)

Training Epoch: 2 [0 /48000 (0%)]       1.922498
Training Epoch: 2 [20000 /48000 (42%)]  1.727946
Training Epoch: 2 [40000 /48000 (83%)]  1.680221

Test set: Average loss: 0.0839, Accuracy 0.7811666666666667 (78%)

Training Epoch: 3 [0 /48000 (0%)]       1.693199
Training Epoch: 3 [20000 /48000 (42%)]  1.700500
Training Epoch: 3 [40000 /48000 (83%)]  1.633141

Test set: Average loss: 0.0794, Accuracy 0.8739166666666667 (87%)

Training Epoch: 4 [0 /48000 (0%)]       1.620224
Training Epoch: 4 [20000 /48000 (42%)]  1.624571
Training Epoch: 4 [40000 /48000 (83%)]  1.537749

Test set: Average loss: 0.0758, Accuracy 0.9486666666666667 (95%)

Training Epoch: 5 [0 /48000 (0%)]       1.662729
Training Epoch: 5 [20000 /48000 (42%)]  1.559987
Training Epoch: 5 [40000 /48000 (83%)]  1.574395

Test set: Average loss: 0.0749, Accuracy 0.965 (96%)

Training Epoch: 6 [0 /48000 (0%)]       1.541774
Training Epoch: 6 [20000 /48000 (42%)]  1.586421
Training Epoch: 6 [40000 /48000 (83%)]  1.551128

Test set: Average loss: 0.0750, Accuracy 0.9623333333333334 (96%)

Training Epoch: 7 [0 /48000 (0%)]       1.615554
Training Epoch: 7 [20000 /48000 (42%)]  1.495618
Training Epoch: 7 [40000 /48000 (83%)]  1.520184

Test set: Average loss: 0.0746, Accuracy 0.9703333333333334 (97%)

Training Epoch: 8 [0 /48000 (0%)]       1.633924
Training Epoch: 8 [20000 /48000 (42%)]  1.483155
Training Epoch: 8 [40000 /48000 (83%)]  1.476593
```

```
Test set: Average loss: 0.0745, Accuracy 0.9714166666666667 (97%)

Training Epoch: 9 [0 /48000 (0%)]       1.630529
Training Epoch: 9 [20000 /48000 (42%)]  1.645484
Training Epoch: 9 [40000 /48000 (83%)]  1.511569

Test set: Average loss: 0.0746, Accuracy 0.9695833333333334 (97%)

Training Epoch: 10 [0 /48000 (0%)]      1.526337
Training Epoch: 10 [20000 /48000 (42%)] 1.556197
Training Epoch: 10 [40000 /48000 (83%)] 1.565505

Test set: Average loss: 0.0743, Accuracy 0.9745833333333334 (97%)
```

Applying Different Learning Rates

In [ ]:

```python
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

model = CNN().to(device)
optimizer = optim.SGD(model.parameters(),lr=0.1,momentum=0.9)
loss_fn = CrossEntropyLoss()

def train(epoch):
    model.train()
    for batch_index, (data,target) in enumerate(loaders['train']):
        data, target = data.to(device), target.to(device)
        optimizer.zero_grad()
        output = model(data)
        loss = loss_fn(output,target)
        loss.backward()
        optimizer.step()
        if batch_index % 1000 == 0:
            print(f"Training Epoch: {epoch} [{batch_index*len(data)} /{len(loaders['train'].dataset)} ({100.*batch_index/len(
```

In [ ]:

```python
for epoch in range(1,11):
    train(epoch)
    test()
```

```
C:\Users\ADMIN\AppData\Local\Temp\ipykernel_4972\1132881025.py:20: UserWarning: Implicit dimension choice for softmax has been
deprecated. Change the call to include dim=X as an argument.
  return F.softmax(x)
```

```
Training Epoch: 1 [0 /48000 (0%)]       1.591946

Test set: Average loss: 0.1501, Accuracy 0.959 (96%)

Training Epoch: 2 [0 /48000 (0%)]       1.562823

Test set: Average loss: 0.1494, Accuracy 0.9666666666666667 (97%)

Training Epoch: 3 [0 /48000 (0%)]       1.510599

Test set: Average loss: 0.1492, Accuracy 0.969 (97%)

Training Epoch: 4 [0 /48000 (0%)]       1.535202

Test set: Average loss: 0.1494, Accuracy 0.96775 (97%)

Training Epoch: 5 [0 /48000 (0%)]       1.478298

Test set: Average loss: 0.1488, Accuracy 0.9731666666666666 (97%)

Training Epoch: 6 [0 /48000 (0%)]       1.511799

Test set: Average loss: 0.1487, Accuracy 0.9735833333333334 (97%)

Training Epoch: 7 [0 /48000 (0%)]       1.489348

Test set: Average loss: 0.1487, Accuracy 0.9740833333333333 (97%)

Training Epoch: 8 [0 /48000 (0%)]       1.479044

Test set: Average loss: 0.1486, Accuracy 0.9751666666666666 (98%)

Training Epoch: 9 [0 /48000 (0%)]       1.538928

Test set: Average loss: 0.1486, Accuracy 0.975 (98%)

Training Epoch: 10 [0 /48000 (0%)]      1.525104

Test set: Average loss: 0.1485, Accuracy 0.9759166666666667 (98%)
```

In [ ]:

```
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

model = CNN().to(device)
optimizer = optim.SGD(model.parameters(),lr=0.001,momentum=0.9)
```

```
loss_fn = CrossEntropyLoss()

def train(epoch):
    model.train()
    for batch_index, (data,target) in enumerate(loaders['train']):
        data, target = data.to(device), target.to(device)
        optimizer.zero_grad()
        output = model(data)
        loss = loss_fn(output,target)
        loss.backward()
        optimizer.step()
        if batch_index % 1000 == 0:
            print(f"Training Epoch: {epoch} [{batch_index*len(data)} /{len(loaders['train'].dataset)} ({100.*batch_index/len(
```

In [ ]:

```
for epoch in range(1,11):
    train(epoch)
    test()
```

```
C:\Users\ADMIN\AppData\Local\Temp\ipykernel_4972\1132881025.py:20: UserWarning: Implicit dimension choice for softmax has been
deprecated. Change the call to include dim=X as an argument.
  return F.softmax(x)
```

```
Training Epoch: 1 [0 /48000 (0%)]       2.300941

Test set: Average loss: 0.2301, Accuracy 0.09608333333333334 (10%)

Training Epoch: 2 [0 /48000 (0%)]       2.302563

Test set: Average loss: 0.2298, Accuracy 0.11125 (11%)

Training Epoch: 3 [0 /48000 (0%)]       2.300130

Test set: Average loss: 0.2294, Accuracy 0.143 (14%)

Training Epoch: 4 [0 /48000 (0%)]       2.297866

Test set: Average loss: 0.2288, Accuracy 0.27491666666666664 (27%)

Training Epoch: 5 [0 /48000 (0%)]       2.294431

Test set: Average loss: 0.2278, Accuracy 0.35383333333333333 (35%)

Training Epoch: 6 [0 /48000 (0%)]       2.298170

Test set: Average loss: 0.2261, Accuracy 0.3606666666666667 (36%)

Training Epoch: 7 [0 /48000 (0%)]       2.278619

Test set: Average loss: 0.2220, Accuracy 0.3433333333333333 (34%)

Training Epoch: 8 [0 /48000 (0%)]       2.283365

Test set: Average loss: 0.2161, Accuracy 0.35041666666666665 (35%)

Training Epoch: 9 [0 /48000 (0%)]       2.252784

Test set: Average loss: 0.2109, Accuracy 0.3785 (38%)

Training Epoch: 10 [0 /48000 (0%)]      2.161472

Test set: Average loss: 0.2050, Accuracy 0.40158333333333335 (40%)
```

In [ ]:

```
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

model = CNN().to(device)
optimizer = optim.SGD(model.parameters(),lr=0.0001,momentum=0.9)
```

```
loss_fn = CrossEntropyLoss()

def train(epoch):
    model.train()
    for batch_index, (data,target) in enumerate(loaders['train']):
        data, target = data.to(device), target.to(device)
        optimizer.zero_grad()
        output = model(data)
        loss = loss_fn(output,target)
        loss.backward()
        optimizer.step()
        if batch_index % 1000 == 0:
            print(f"Training Epoch: {epoch} [{batch_index*len(data)} /{len(loaders['train'].dataset)} ({100.*batch_index/len(
```

In [ ]:

```
for epoch in range(1,11):
    train(epoch)
    test()
```

```
C:\Users\ADMIN\AppData\Local\Temp\ipykernel_4972\1132881025.py:20: UserWarning: Implicit dimension choice for softmax has been
deprecated. Change the call to include dim=X as an argument.
  return F.softmax(x)
```

```
Training Epoch: 1 [0 /48000 (0%)]       2.302290

Test set: Average loss: 0.2302, Accuracy 0.1245 (12%)

Training Epoch: 2 [0 /48000 (0%)]       2.301475

Test set: Average loss: 0.2302, Accuracy 0.12941666666666668 (13%)

Training Epoch: 3 [0 /48000 (0%)]       2.302164

Test set: Average loss: 0.2301, Accuracy 0.13416666666666666 (13%)

Training Epoch: 4 [0 /48000 (0%)]       2.301218

Test set: Average loss: 0.2301, Accuracy 0.138 (14%)

Training Epoch: 5 [0 /48000 (0%)]       2.300221

Test set: Average loss: 0.2301, Accuracy 0.1425 (14%)

Training Epoch: 6 [0 /48000 (0%)]       2.303961

Test set: Average loss: 0.2301, Accuracy 0.14691666666666667 (15%)

Training Epoch: 7 [0 /48000 (0%)]       2.303224

Test set: Average loss: 0.2301, Accuracy 0.15116666666666667 (15%)

Training Epoch: 8 [0 /48000 (0%)]       2.301388

Test set: Average loss: 0.2300, Accuracy 0.156 (16%)

Training Epoch: 9 [0 /48000 (0%)]       2.299639

Test set: Average loss: 0.2300, Accuracy 0.16525 (17%)

Training Epoch: 10 [0 /48000 (0%)]      2.300627

Test set: Average loss: 0.2300, Accuracy 0.18283333333333332 (18%)
```

In [ ]:

```
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

model = CNN().to(device)
optimizer = optim.SGD(model.parameters(),lr=0.00001,momentum=0.9)
```

```
loss_fn = CrossEntropyLoss()

def train(epoch):
    model.train()
    for batch_index, (data,target) in enumerate(loaders['train']):
        data, target = data.to(device), target.to(device)
        optimizer.zero_grad()
        output = model(data)
        loss = loss_fn(output,target)
        loss.backward()
        optimizer.step()
        if batch_index % 1000 == 0:
            print(f"Training Epoch: {epoch} [{batch_index*len(data)} /{len(loaders['train'].dataset)} ({100.*batch_index/len(
```

In [ ]:

```
for epoch in range(1,11):
    train(epoch)
    test()
```

```
C:\Users\ADMIN\AppData\Local\Temp\ipykernel_4972\1132881025.py:20: UserWarning: Implicit dimension choice for softmax has been
deprecated. Change the call to include dim=X as an argument.
  return F.softmax(x)
```

```
Training Epoch: 1 [0 /48000 (0%)]       2.302121

Test set: Average loss: 0.2303, Accuracy 0.09866666666666667 (10%)

Training Epoch: 2 [0 /48000 (0%)]       2.302371

Test set: Average loss: 0.2303, Accuracy 0.09866666666666667 (10%)

Training Epoch: 3 [0 /48000 (0%)]       2.302096

Test set: Average loss: 0.2303, Accuracy 0.09866666666666667 (10%)

Training Epoch: 4 [0 /48000 (0%)]       2.304729

Test set: Average loss: 0.2303, Accuracy 0.09866666666666667 (10%)

Training Epoch: 5 [0 /48000 (0%)]       2.302164

Test set: Average loss: 0.2303, Accuracy 0.0985 (10%)

Training Epoch: 6 [0 /48000 (0%)]       2.301810

Test set: Average loss: 0.2303, Accuracy 0.0985 (10%)

Training Epoch: 7 [0 /48000 (0%)]       2.300794

Test set: Average loss: 0.2303, Accuracy 0.09841666666666667 (10%)

Training Epoch: 8 [0 /48000 (0%)]       2.302886

Test set: Average loss: 0.2303, Accuracy 0.09841666666666667 (10%)

Training Epoch: 9 [0 /48000 (0%)]       2.301160

Test set: Average loss: 0.2303, Accuracy 0.09841666666666667 (10%)

Training Epoch: 10 [0 /48000 (0%)]      2.305634

Test set: Average loss: 0.2303, Accuracy 0.09833333333333333 (10%)
```

```
In [ ]:
```