```
import torch
torch.manual_seed(17)

import numpy as np
from torchsummary import summary
from tqdm import tqdm
import matplotlib.pyplot as plt

from DatasetLoader import DatasetFetcher
from project_model import *

# if torch.backends.mps.is_available():
#     mps_device = torch.device("mps")
#     x = torch.ones(1, device=mps_device)
#     print (x)
# else:
#     print ("MPS device not found.")

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

```
cuda
```

```
# Fetching Dataset
df = DatasetFetcher(dataset = "CIFAR10", batch_size = 128)
df.addHorizontalFlipping()
#df.addVerticalFlipping()
df.addRandomCrop(size = 32, padding = 4)
#df.addAutoAugmentation()
#df.addHistogramEqualization()
df.addNormalizer()
#df.addGaussianNoise()
trainLoader, testLoader = df.getLoaders()
```

```
Initializing fetching CIFAR10 dataset using torchvision
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
```

```
# Get Model
#model = ResNet(BasicBlock, 32, 4, [4, 4, 4, 2], 10, bias=True)
model = project1_model()
model = model.to(device)
print(summary(model, input_size = (3, 32, 32)))
```

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 32, 32, 32]             896
       BatchNorm2d-2          [-1, 32, 32, 32]              64
            Conv2d-3          [-1, 32, 32, 32]           9,248
```

BatchNorm2d-4	[-1, 32, 32, 32]	64
Conv2d-5	[-1, 32, 32, 32]	9,248
BatchNorm2d-6	[-1, 32, 32, 32]	64
BasicBlock-7	[-1, 32, 32, 32]	0
Conv2d-8	[-1, 32, 32, 32]	9,248
BatchNorm2d-9	[-1, 32, 32, 32]	64
Conv2d-10	[-1, 32, 32, 32]	9,248
BatchNorm2d-11	[-1, 32, 32, 32]	64
BasicBlock-12	[-1, 32, 32, 32]	0
Conv2d-13	[-1, 32, 32, 32]	9,248
BatchNorm2d-14	[-1, 32, 32, 32]	64
Conv2d-15	[-1, 32, 32, 32]	9,248
BatchNorm2d-16	[-1, 32, 32, 32]	64
BasicBlock-17	[-1, 32, 32, 32]	0
Conv2d-18	[-1, 32, 32, 32]	9,248
BatchNorm2d-19	[-1, 32, 32, 32]	64
Conv2d-20	[-1, 32, 32, 32]	9,248
BatchNorm2d-21	[-1, 32, 32, 32]	64
BasicBlock-22	[-1, 32, 32, 32]	0
Conv2d-23	[-1, 64, 16, 16]	18,496
BatchNorm2d-24	[-1, 64, 16, 16]	128
Conv2d-25	[-1, 64, 16, 16]	36,928
BatchNorm2d-26	[-1, 64, 16, 16]	128
Conv2d-27	[-1, 64, 16, 16]	2,112
BatchNorm2d-28	[-1, 64, 16, 16]	128
BasicBlock-29	[-1, 64, 16, 16]	0
Conv2d-30	[-1, 64, 16, 16]	36,928
BatchNorm2d-31	[-1, 64, 16, 16]	128
Conv2d-32	[-1, 64, 16, 16]	36,928
BatchNorm2d-33	[-1, 64, 16, 16]	128
BasicBlock-34	[-1, 64, 16, 16]	0
Conv2d-35	[-1, 64, 16, 16]	36,928
BatchNorm2d-36	[-1, 64, 16, 16]	128
Conv2d-37	[-1, 64, 16, 16]	36,928
BatchNorm2d-38	[-1, 64, 16, 16]	128
BasicBlock-39	[-1, 64, 16, 16]	0
Conv2d-40	[-1, 64, 16, 16]	36,928
BatchNorm2d-41	[-1, 64, 16, 16]	128
Conv2d-42	[-1, 64, 16, 16]	36,928
BatchNorm2d-43	[-1, 64, 16, 16]	128
BasicBlock-44	[-1, 64, 16, 16]	0
Conv2d-45	[-1, 128, 8, 8]	73,856
BatchNorm2d-46	[-1, 128, 8, 8]	256
Conv2d-47	[-1, 128, 8, 8]	147,584
BatchNorm2d-48	[-1, 128, 8, 8]	256
Conv2d-49	[-1, 128, 8, 8]	8,320
BatchNorm2d-50	[-1, 128, 8, 8]	256
BasicBlock-51	[-1, 128, 8, 8]	0
Conv2d-52	[-1, 128, 8, 8]	147,584
BatchNorm2d-53	[-1, 128, 8, 8]	256

```
           Conv2d-54            [-1, 128, 8, 8]         147,584
      BatchNorm2d-55            [-1, 128, 8, 8]             256
       BasicBlock-56            [-1, 128, 8, 8]               0
           Conv2d-57            [-1, 128, 8, 8]         147,584
      BatchNorm2d-58            [-1, 128, 8, 8]             256
           Conv2d-59            [-1, 128, 8, 8]         147,584
      BatchNorm2d-60            [-1, 128, 8, 8]             256
       BasicBlock-61            [-1, 128, 8, 8]               0
           Conv2d-62            [-1, 128, 8, 8]         147,584
      BatchNorm2d-63            [-1, 128, 8, 8]             256
           Conv2d-64            [-1, 128, 8, 8]         147,584
      BatchNorm2d-65            [-1, 128, 8, 8]             256
       BasicBlock-66            [-1, 128, 8, 8]               0
           Conv2d-67            [-1, 256, 4, 4]         295,168
      BatchNorm2d-68            [-1, 256, 4, 4]             512
           Conv2d-69            [-1, 256, 4, 4]         590,080
      BatchNorm2d-70            [-1, 256, 4, 4]             512
           Conv2d-71            [-1, 256, 4, 4]          33,024
      BatchNorm2d-72            [-1, 256, 4, 4]             512
       BasicBlock-73            [-1, 256, 4, 4]               0
           Conv2d-74            [-1, 256, 4, 4]         590,080
      BatchNorm2d-75            [-1, 256, 4, 4]             512
           Conv2d-76            [-1, 256, 4, 4]         590,080
      BatchNorm2d-77            [-1, 256, 4, 4]             512
       BasicBlock-78            [-1, 256, 4, 4]               0
           Linear-79                  [-1, 10]           2,570
================================================================
Total params: 3,576,842
Trainable params: 3,576,842
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.01
Forward/backward pass size (MB): 10.00
Params size (MB): 13.64
Estimated Total Size (MB): 23.66
----------------------------------------------------------------
None
```

```
EPOCHS = 100
globalBestAccuracy = 0.0
trainingLoss = []
testingLoss = []
trainingAccuracy = []
testingAccuracy = []

# Defining Loss Function, Learning Rate, Weight Decay, Optimizer)
lossFunction = torch.nn.CrossEntropyLoss(reduction = 'sum')
learningRate = 0.1
weightDecay = 0.0001
# optimizer = torch.optim.Adam(model.parameters(), lr=learningRate,
weight_decay=weightDecay)
```

```
# optimizer = torch.optim.Adagrad(model.parameters(), lr=learningRate,
weight_decay=weightDecay)
optimizer = torch.optim.Adadelta(model.parameters(), lr =
learningRate, weight_decay = weightDecay)
scheduler = torch.optim.lr_scheduler.CosineAnnealingLR(optimizer,
EPOCHS, eta_min = learningRate/10.0)
print(model.eval())
trainable_parameters = sum(p.numel() for p in model.parameters() if
p.requires_grad)
print("Total Trainable Parameters : %s"%(trainable_parameters))
if trainable_parameters > 5 * (10 ** 6):
    raise Exception("Model not under budget!")
```

```
ResNet(
  (conv1): Conv2d(3, 32, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
  (bn1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (layer1): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
      (bn1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv2): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
      (bn2): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (shortcut): Sequential()
    )
    (1): BasicBlock(
      (conv1): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
      (bn1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv2): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
      (bn2): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (shortcut): Sequential()
    )
    (2): BasicBlock(
      (conv1): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
      (bn1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv2): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
      (bn2): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
```

```
      (shortcut): Sequential()
    )
    (3): BasicBlock(
      (conv1): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
      (bn1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv2): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
      (bn2): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (shortcut): Sequential()
    )
  )
  (layer2): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(32, 64, kernel_size=(3, 3), stride=(2, 2),
padding=(1, 1))
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (shortcut): Sequential(
        (0): Conv2d(32, 64, kernel_size=(1, 1), stride=(2, 2))
        (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (shortcut): Sequential()
    )
    (2): BasicBlock(
      (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
```

```
      (shortcut): Sequential()
    )
    (3): BasicBlock(
      (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (shortcut): Sequential()
    )
  )
  (layer3): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(64, 128, kernel_size=(3, 3), stride=(2, 2),
padding=(1, 1))
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (shortcut): Sequential(
        (0): Conv2d(64, 128, kernel_size=(1, 1), stride=(2, 2))
        (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (shortcut): Sequential()
    )
    (2): BasicBlock(
      (conv1): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
```

```
      (shortcut): Sequential()
    )
    (3): BasicBlock(
      (conv1): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (shortcut): Sequential()
    )
  )
  (layer4): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(128, 256, kernel_size=(3, 3), stride=(2, 2),
padding=(1, 1))
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (shortcut): Sequential(
        (0): Conv2d(128, 256, kernel_size=(1, 1), stride=(2, 2))
        (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (shortcut): Sequential()
    )
  )
  (linear): Linear(in_features=256, out_features=10, bias=True)
)
Total Trainable Parameters : 3576842
```

```
# Training
for i in tqdm(range(EPOCHS)):
    for phase in ['train', 'test']:
        if phase == "train":
```

```
            loader = trainLoader
            model.train()
            optimizer.zero_grad()
        else:
            loader = testLoader
            model.eval()
        runningLoss = 0.0
        runningCorrects = 0
        for images, labels in loader:
            images = images.to(device)
            labels = labels.to(device)
            output = model(images)
            loss = lossFunction(output, labels)
            predicted_labels = torch.argmax(output, dim=1)
            #runningLoss += loss.item()*images.size(0)
            runningLoss += loss.item()
            runningCorrects += torch.sum(predicted_labels ==
labels).float().item()
            if phase == "train":
                loss.backward()
                optimizer.step()
        epochLoss = runningLoss/len(loader.dataset)
        epochAccuracy = runningCorrects/len(loader.dataset)
        if phase == "train":
            scheduler.step()
            trainingLoss.append(epochLoss)
            trainingAccuracy.append(epochAccuracy)
        else:
            testingLoss.append(epochLoss)
            testingAccuracy.append(epochAccuracy)
            if epochAccuracy > globalBestAccuracy:
                globalBestAccuracy = epochAccuracy
                model.saveToDisk()
    print("Training Loss : %s, Testing Loss : %s, Training Accuracy :
%s, Testing Accuracy : %s"\
          %(trainingLoss[-1], testingLoss[-1], trainingAccuracy[-1],
testingAccuracy[-1]))
```

```
  1%|          | 1/100 [00:10<18:08, 10.99s/it]

Training Loss : 1.750534892578125, Testing Loss : 1.7342327341079713,
Training Accuracy : 0.34224, Testing Accuracy : 0.3896

   2%|          | 2/100 [00:21<17:32, 10.74s/it]

Training Loss : 1.5082956085205077, Testing Loss : 1.4530678709030151,
Training Accuracy : 0.44076, Testing Accuracy : 0.4681

   3%|          | 3/100 [00:32<17:10, 10.63s/it]
```

```
Training Loss : 1.306699574584961, Testing Loss : 1.224788269996643,
Training Accuracy : 0.52454, Testing Accuracy : 0.5583

   4%|▌         | 4/100 [00:42<17:04, 10.67s/it]

Training Loss : 1.1761730513000488, Testing Loss : 1.131978190612793,
Training Accuracy : 0.57418, Testing Accuracy : 0.5967

   5%|▌         | 5/100 [00:53<16:43, 10.56s/it]

Training Loss : 1.045868921508789, Testing Loss : 0.9739497625350952,
Training Accuracy : 0.62728, Testing Accuracy : 0.654

   6%|▌         | 6/100 [01:03<16:23, 10.47s/it]

Training Loss : 0.9365599842834472, Testing Loss : 0.9753861132621765,
Training Accuracy : 0.66576, Testing Accuracy : 0.6545

   7%|▋         | 7/100 [01:14<16:19, 10.53s/it]

Training Loss : 0.8710366773986816, Testing Loss : 0.9070924603462219,
Training Accuracy : 0.69146, Testing Accuracy : 0.6908

   8%|▊         | 8/100 [01:24<16:03, 10.47s/it]

Training Loss : 0.7868001916503906, Testing Loss : 0.8118785608291625,
Training Accuracy : 0.72428, Testing Accuracy : 0.72

   9%|▊         | 9/100 [01:34<15:53, 10.48s/it]

Training Loss : 0.7216525913238525, Testing Loss : 0.838003656578064,
Training Accuracy : 0.74744, Testing Accuracy : 0.7098

  10%|▉         | 10/100 [01:45<15:47, 10.53s/it]

Training Loss : 0.6740130103302002, Testing Loss : 0.7190186059951782,
Training Accuracy : 0.76368, Testing Accuracy : 0.7558

  11%|▉         | 11/100 [01:56<15:37, 10.53s/it]

Training Loss : 0.616966068649292, Testing Loss : 0.6758499299526215,
Training Accuracy : 0.78718, Testing Accuracy : 0.7718

  12%|▉         | 12/100 [02:06<15:27, 10.54s/it]

Training Loss : 0.5796961762237549, Testing Loss : 0.6749031675338745,
Training Accuracy : 0.79728, Testing Accuracy : 0.7725

  13%|▉         | 13/100 [02:17<15:16, 10.54s/it]

Training Loss : 0.5621770449066162, Testing Loss : 0.677848752117157,
Training Accuracy : 0.80552, Testing Accuracy : 0.7785

  14%|▉         | 14/100 [02:27<14:59, 10.46s/it]
```

Training Loss : 0.521658950881958, Testing Loss : 0.5816180925369263,
Training Accuracy : 0.82026, Testing Accuracy : 0.8048

15%|█ | 15/100 [02:37<14:45, 10.41s/it]

Training Loss : 0.5030528082275391, Testing Loss : 0.5683472782611847,
Training Accuracy : 0.8271, Testing Accuracy : 0.8063

16%|█ | 16/100 [02:48<14:36, 10.44s/it]

Training Loss : 0.4880191324996948, Testing Loss : 0.5641240973472595,
Training Accuracy : 0.83208, Testing Accuracy : 0.8082

17%|█ | 17/100 [02:58<14:26, 10.45s/it]

Training Loss : 0.46874917739868166, Testing Loss :
0.5573918341636658, Training Accuracy : 0.83866, Testing Accuracy :
0.8151

18%|█ | 18/100 [03:09<14:12, 10.40s/it]

Training Loss : 0.43556508716583253, Testing Loss :
0.5247749044418335, Training Accuracy : 0.84942, Testing Accuracy :
0.8232

19%|█ | 19/100 [03:19<14:03, 10.41s/it]

Training Loss : 0.4199140442276001, Testing Loss : 0.5646599299430847,
Training Accuracy : 0.85444, Testing Accuracy : 0.8075

20%|█ | 20/100 [03:29<13:46, 10.33s/it]

Training Loss : 0.41516722038269044, Testing Loss :
0.5037532205581665, Training Accuracy : 0.8576, Testing Accuracy :
0.8308

21%|█ | 21/100 [03:40<13:40, 10.39s/it]

Training Loss : 0.3896746640777588, Testing Loss : 0.5259988674163818,
Training Accuracy : 0.86682, Testing Accuracy : 0.8253

22%|█ | 22/100 [03:50<13:33, 10.43s/it]

Training Loss : 0.3784850244140625, Testing Loss : 0.4869466375350952,
Training Accuracy : 0.86954, Testing Accuracy : 0.8357

23%|█ | 23/100 [04:01<13:25, 10.47s/it]

Training Loss : 0.3628396271133423, Testing Loss : 0.4769480683326721,
Training Accuracy : 0.87546, Testing Accuracy : 0.8399

24%|█ | 24/100 [04:11<13:08, 10.38s/it]

Training Loss : 0.34634707462310793, Testing Loss : 0.49430470991134645, Training Accuracy : 0.88002, Testing Accuracy : 0.8376

  25%|██ | 25/100 [04:21<13:02, 10.44s/it]

Training Loss : 0.3391695451354981, Testing Loss : 0.493526510810852, Training Accuracy : 0.88278, Testing Accuracy : 0.8397

  26%|██ | 26/100 [04:32<12:52, 10.44s/it]

Training Loss : 0.32523911186218263, Testing Loss : 0.43323381824493407, Training Accuracy : 0.88692, Testing Accuracy : 0.8571

  27%|██ | 27/100 [04:42<12:41, 10.44s/it]

Training Loss : 0.3070455528640747, Testing Loss : 0.4389434522628784, Training Accuracy : 0.89318, Testing Accuracy : 0.8532

  28%|██ | 28/100 [04:53<12:34, 10.48s/it]

Training Loss : 0.296951802444458, Testing Loss : 0.48018632173538206, Training Accuracy : 0.89616, Testing Accuracy : 0.8493

  29%|██ | 29/100 [05:03<12:22, 10.46s/it]

Training Loss : 0.2894485390853882, Testing Loss : 0.45177939550876617, Training Accuracy : 0.89862, Testing Accuracy : 0.8567

  30%|███ | 30/100 [05:14<12:14, 10.49s/it]

Training Loss : 0.2819375555610657, Testing Loss : 0.4342577423095703, Training Accuracy : 0.90124, Testing Accuracy : 0.8612

  31%|███ | 31/100 [05:24<11:57, 10.40s/it]

Training Loss : 0.2773608359146118, Testing Loss : 0.45364204959869386, Training Accuracy : 0.9037, Testing Accuracy : 0.8555

  32%|███ | 32/100 [05:35<11:54, 10.51s/it]

Training Loss : 0.2649358284568787, Testing Loss : 0.41601660060882567, Training Accuracy : 0.908, Testing Accuracy : 0.866

  33%|███ | 33/100 [05:45<11:32, 10.34s/it]

Training Loss : 0.25196005603790284, Testing Loss : 0.43029733076095583, Training Accuracy : 0.91294, Testing Accuracy : 0.8628

34%|███ | 34/100 [05:55<11:21, 10.32s/it]

Training Loss : 0.24615072959899903, Testing Loss :
0.4192706132411957, Training Accuracy : 0.91334, Testing Accuracy :
0.8685

35%|███ | 35/100 [06:06<11:16, 10.41s/it]

Training Loss : 0.22745964323043824, Testing Loss :
0.41750600509643554, Training Accuracy : 0.91998, Testing Accuracy :
0.8717

36%|███ | 36/100 [06:17<11:15, 10.56s/it]

Training Loss : 0.23327418605804442, Testing Loss :
0.43516239695549014, Training Accuracy : 0.91852, Testing Accuracy :
0.864

37%|███ | 37/100 [06:28<11:15, 10.72s/it]

Training Loss : 0.22252747444152832, Testing Loss :
0.40060316247940064, Training Accuracy : 0.92252, Testing Accuracy :
0.8794

38%|███ | 38/100 [06:38<10:58, 10.63s/it]

Training Loss : 0.19907417359352111, Testing Loss :
0.41780849165916445, Training Accuracy : 0.93024, Testing Accuracy :
0.8778

39%|███ | 39/100 [06:49<10:46, 10.59s/it]

Training Loss : 0.20170173211097717, Testing Loss :
0.43313117980957033, Training Accuracy : 0.9291, Testing Accuracy :
0.8717

40%|████ | 40/100 [06:59<10:34, 10.57s/it]

Training Loss : 0.19560000583648682, Testing Loss :
0.4325757998466492, Training Accuracy : 0.93202, Testing Accuracy :
0.8761

41%|████ | 41/100 [07:09<10:15, 10.43s/it]

Training Loss : 0.19260304227828978, Testing Loss :
0.4230586800336838, Training Accuracy : 0.93324, Testing Accuracy :
0.8795

42%|████ | 42/100 [07:20<10:04, 10.43s/it]

Training Loss : 0.1884927225112915, Testing Loss :
0.44363087105751037, Training Accuracy : 0.934, Testing Accuracy :
0.8737

43%|████ | 43/100 [07:30<09:56, 10.47s/it]

Training Loss : 0.18156796796798705, Testing Loss :
0.42046577625274656, Training Accuracy : 0.93464, Testing Accuracy :
0.8784

44%|████ | 44/100 [07:41<09:47, 10.50s/it]

Training Loss : 0.1728495922756195, Testing Loss : 0.4485450621128082,
Training Accuracy : 0.9383, Testing Accuracy : 0.877

45%|████ | 45/100 [07:51<09:39, 10.53s/it]

Training Loss : 0.16596370498657226, Testing Loss :
0.43388833293914797, Training Accuracy : 0.94092, Testing Accuracy :
0.8772

46%|████ | 46/100 [08:02<09:31, 10.58s/it]

Training Loss : 0.16174998712539673, Testing Loss :
0.39738049416542054, Training Accuracy : 0.94316, Testing Accuracy :
0.887

47%|████ | 47/100 [08:13<09:19, 10.56s/it]

Training Loss : 0.14697849188804626, Testing Loss :
0.41343463668823244, Training Accuracy : 0.9477, Testing Accuracy :
0.8905

48%|████ | 48/100 [08:23<09:06, 10.50s/it]

Training Loss : 0.13773472970962525, Testing Loss :
0.4124880380630493, Training Accuracy : 0.95168, Testing Accuracy :
0.8842

49%|████ | 49/100 [08:33<08:49, 10.38s/it]

Training Loss : 0.1393139978981018, Testing Loss :
0.45448302216529846, Training Accuracy : 0.95046, Testing Accuracy :
0.884

50%|████ | 50/100 [08:44<08:41, 10.43s/it]

Training Loss : 0.13646688494682313, Testing Loss :
0.4219742176771164, Training Accuracy : 0.95198, Testing Accuracy :
0.8878

51%|████ | 51/100 [08:54<08:31, 10.44s/it]

Training Loss : 0.1224304889678955, Testing Loss : 0.4543603331565857,
Training Accuracy : 0.9567, Testing Accuracy : 0.8894

52%|████ | 52/100 [09:05<08:22, 10.47s/it]

Training Loss : 0.11998906311511993, Testing Loss :
0.40435436005592346, Training Accuracy : 0.95704, Testing Accuracy :
0.8936

53%|█████ | 53/100 [09:15<08:12, 10.48s/it]

Training Loss : 0.11606632172584534, Testing Loss :
0.4392728645801544, Training Accuracy : 0.9597, Testing Accuracy :
0.8916

54%|█████ | 54/100 [09:26<08:01, 10.46s/it]

Training Loss : 0.10802373628616332, Testing Loss :
0.4287527338027954, Training Accuracy : 0.96098, Testing Accuracy :
0.89

55%|█████ | 55/100 [09:36<07:50, 10.45s/it]

Training Loss : 0.11218075738430024, Testing Loss :
0.4454464340686798, Training Accuracy : 0.96176, Testing Accuracy :
0.8899

56%|█████ | 56/100 [09:46<07:40, 10.47s/it]

Training Loss : 0.10551222920417785, Testing Loss :
0.40828085255622865, Training Accuracy : 0.9629, Testing Accuracy :
0.8933

57%|█████ | 57/100 [09:57<07:33, 10.54s/it]

Training Loss : 0.09989498874664307, Testing Loss :
0.4223347550392151, Training Accuracy : 0.964, Testing Accuracy :
0.8968

58%|█████ | 58/100 [10:08<07:23, 10.56s/it]

Training Loss : 0.09043127286434173, Testing Loss :
0.4521582398176193, Training Accuracy : 0.96802, Testing Accuracy :
0.8915

59%|█████ | 59/100 [10:18<07:10, 10.51s/it]

Training Loss : 0.08866517826795578, Testing Loss :
0.42655759048461916, Training Accuracy : 0.96712, Testing Accuracy :
0.8966

60%|██████ | 60/100 [10:29<07:00, 10.52s/it]

Training Loss : 0.08230563286304474, Testing Loss :
0.4526253000259399, Training Accuracy : 0.97016, Testing Accuracy :
0.8947

61%|██████ | 61/100 [10:39<06:50, 10.52s/it]

Training Loss : 0.08154015812158584, Testing Loss : 0.4504463098526001, Training Accuracy : 0.97072, Testing Accuracy : 0.8958

62%|██████ | 62/100 [10:49<06:35, 10.40s/it]

Training Loss : 0.07949949363470077, Testing Loss : 0.4565501708030701, Training Accuracy : 0.97202, Testing Accuracy : 0.8959

63%|██████ | 63/100 [11:00<06:25, 10.41s/it]

Training Loss : 0.0747082015299797, Testing Loss : 0.4420340425491333, Training Accuracy : 0.97306, Testing Accuracy : 0.8995

64%|██████ | 64/100 [11:11<06:19, 10.55s/it]

Training Loss : 0.07220506791591644, Testing Loss : 0.48928732318878176, Training Accuracy : 0.97422, Testing Accuracy : 0.8956

65%|██████ | 65/100 [11:21<06:05, 10.45s/it]

Training Loss : 0.07177760385513306, Testing Loss : 0.42840618667602537, Training Accuracy : 0.97424, Testing Accuracy : 0.8998

66%|██████ | 66/100 [11:31<05:52, 10.36s/it]

Training Loss : 0.06686747444868088, Testing Loss : 0.45723098306655885, Training Accuracy : 0.97728, Testing Accuracy : 0.9035

67%|██████ | 67/100 [11:41<05:42, 10.39s/it]

Training Loss : 0.06354128739356994, Testing Loss : 0.472225291967392, Training Accuracy : 0.97752, Testing Accuracy : 0.9008

68%|██████ | 68/100 [11:52<05:34, 10.46s/it]

Training Loss : 0.06781714969158173, Testing Loss : 0.42682201583385465, Training Accuracy : 0.97536, Testing Accuracy : 0.9036

69%|██████ | 69/100 [12:03<05:23, 10.45s/it]

Training Loss : 0.05613621583223343, Testing Loss : 0.47618203630447387, Training Accuracy : 0.9805, Testing Accuracy : 0.904

70%|██████ | 70/100 [12:13<05:13, 10.46s/it]

Training Loss : 0.04830798606395721, Testing Loss :
0.42769108983278276, Training Accuracy : 0.98292, Testing Accuracy :
0.9111

71%|███████ | 71/100 [12:23<04:59, 10.33s/it]

Training Loss : 0.04202377602219581, Testing Loss :
0.4650554542541504, Training Accuracy : 0.98532, Testing Accuracy :
0.9086

72%|███████ | 72/100 [12:33<04:47, 10.25s/it]

Training Loss : 0.044704957706928256, Testing Loss :
0.4745908398151398, Training Accuracy : 0.9841, Testing Accuracy :
0.906

73%|███████ | 73/100 [12:43<04:37, 10.28s/it]

Training Loss : 0.04317195358723402, Testing Loss :
0.47118928098678586, Training Accuracy : 0.98452, Testing Accuracy :
0.9054

74%|███████ | 74/100 [12:54<04:30, 10.39s/it]

Training Loss : 0.038022401463091375, Testing Loss :
0.4802045246601105, Training Accuracy : 0.98612, Testing Accuracy :
0.907

75%|███████ | 75/100 [13:05<04:21, 10.45s/it]

Training Loss : 0.038868962845504285, Testing Loss :
0.4588340826034546, Training Accuracy : 0.98592, Testing Accuracy :
0.9075

76%|███████ | 76/100 [13:15<04:11, 10.47s/it]

Training Loss : 0.03363791984736919, Testing Loss :
0.4912823760032654, Training Accuracy : 0.98772, Testing Accuracy :
0.9074

77%|███████ | 77/100 [13:26<04:02, 10.54s/it]

Training Loss : 0.03306935468971729, Testing Loss :
0.4800869453430176, Training Accuracy : 0.98842, Testing Accuracy :
0.9119

78%|███████ | 78/100 [13:36<03:51, 10.54s/it]

Training Loss : 0.030943616832494737, Testing Loss :
0.5103559723854065, Training Accuracy : 0.98878, Testing Accuracy :
0.9073

79%|███████ | 79/100 [13:47<03:41, 10.53s/it]

Training Loss : 0.02811643488228321, Testing Loss :
0.47872517070770265, Training Accuracy : 0.99016, Testing Accuracy :
0.9105

  80%|████████ | 80/100 [13:57<03:30, 10.51s/it]

Training Loss : 0.027139724824428557, Testing Loss :
0.5180482689857483, Training Accuracy : 0.99054, Testing Accuracy :
0.9116

  81%|████████ | 81/100 [14:08<03:18, 10.45s/it]

Training Loss : 0.026257721359729768, Testing Loss :
0.4880937006950378, Training Accuracy : 0.991, Testing Accuracy :
0.9117

  82%|████████ | 82/100 [14:18<03:06, 10.38s/it]

Training Loss : 0.022030702589005233, Testing Loss :
0.5105392691135406, Training Accuracy : 0.9921, Testing Accuracy :
0.9131

  83%|████████ | 83/100 [14:28<02:56, 10.40s/it]

Training Loss : 0.024398157700002192, Testing Loss :
0.5084094598770141, Training Accuracy : 0.99134, Testing Accuracy :
0.9082

  84%|████████ | 84/100 [14:39<02:47, 10.47s/it]

Training Loss : 0.024022529678046703, Testing Loss :
0.5260673250198364, Training Accuracy : 0.99164, Testing Accuracy :
0.9117

  85%|████████ | 85/100 [14:49<02:36, 10.44s/it]

Training Loss : 0.022933623305782677, Testing Loss :
0.5059761106491089, Training Accuracy : 0.99226, Testing Accuracy :
0.9112

  86%|████████ | 86/100 [15:00<02:26, 10.49s/it]

Training Loss : 0.021772636709790676, Testing Loss :
0.5110366032600403, Training Accuracy : 0.99248, Testing Accuracy :
0.9108

  87%|████████ | 87/100 [15:10<02:15, 10.45s/it]

Training Loss : 0.020182699877321 72, Testing Loss :
0.5267858083724976, Training Accuracy : 0.99248, Testing Accuracy :
0.9106

  88%|████████ | 88/100 [15:21<02:05, 10.45s/it]

Training Loss : 0.020168822491690518, Testing Loss :
0.49368031721115113, Training Accuracy : 0.99354, Testing Accuracy :
0.918

89%|████████▉ | 89/100 [15:31<01:55, 10.49s/it]

Training Loss : 0.013290383730456233, Testing Loss :
0.514165759563446, Training Accuracy : 0.9955, Testing Accuracy :
0.9159

90%|█████████ | 90/100 [15:42<01:45, 10.51s/it]

Training Loss : 0.01507324075654149, Testing Loss :
0.5301901911735535, Training Accuracy : 0.99432, Testing Accuracy :
0.9166

91%|█████████ | 91/100 [15:52<01:34, 10.52s/it]

Training Loss : 0.01492599542953074, Testing Loss :
0.5241970549583435, Training Accuracy : 0.995, Testing Accuracy :
0.9126

92%|█████████▏ | 92/100 [16:03<01:24, 10.54s/it]

Training Loss : 0.013875124659389258, Testing Loss :
0.516765375328064, Training Accuracy : 0.9952, Testing Accuracy :
0.918

93%|█████████▎ | 93/100 [16:13<01:13, 10.50s/it]

Training Loss : 0.01340749524973333, Testing Loss :
0.5306621287345886, Training Accuracy : 0.9955, Testing Accuracy :
0.9133

94%|█████████▍ | 94/100 [16:24<01:03, 10.55s/it]

Training Loss : 0.011001844610236585, Testing Loss :
0.5336188425540924, Training Accuracy : 0.9963, Testing Accuracy :
0.9159

95%|█████████▌ | 95/100 [16:34<00:52, 10.47s/it]

Training Loss : 0.013007677709460258, Testing Loss :
0.5420924195289611, Training Accuracy : 0.99538, Testing Accuracy :
0.9141

96%|█████████▌ | 96/100 [16:45<00:41, 10.45s/it]

Training Loss : 0.011571285721063614, Testing Loss :
0.5435201092720032, Training Accuracy : 0.99602, Testing Accuracy :
0.917

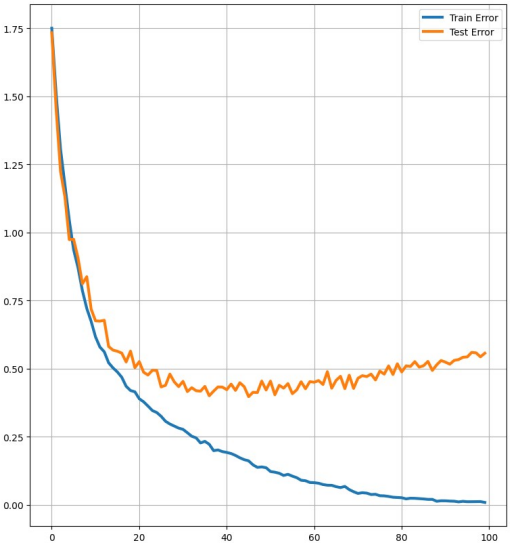
97%|█████████▋ | 97/100 [16:55<00:31, 10.44s/it]

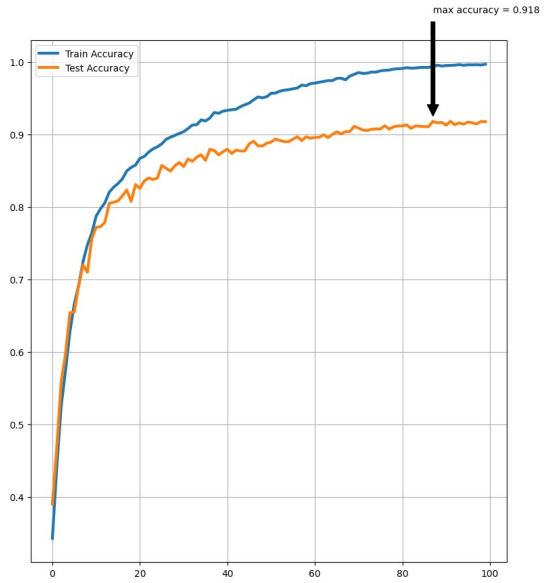
```
Training Loss : 0.011715136720538139, Testing Loss :
0.5601223219871521, Training Accuracy : 0.99596, Testing Accuracy :
0.916

  98%|██████████| 98/100 [17:06<00:20, 10.47s/it]

Training Loss : 0.011892944584768266, Testing Loss :
0.5583502286911011, Training Accuracy : 0.99608, Testing Accuracy :
0.9144

  99%|██████████| 99/100 [17:16<00:10, 10.52s/it]

Training Loss : 0.012115279949195683, Testing Loss :
0.5437278823375702, Training Accuracy : 0.99578, Testing Accuracy :
0.9179

100%|██████████| 100/100 [17:27<00:00, 10.48s/it]

Training Loss : 0.008914166655614971, Testing Loss :
0.5569703786849975, Training Accuracy : 0.99684, Testing Accuracy :
0.9176
```

```
print("Maximum Testing Accuracy Achieved: %s"%(max(testingAccuracy)))
xmax = np.argmax(testingAccuracy)
ymax = max(testingAccuracy)
```

```
Maximum Testing Accuracy Achieved: 0.918
```

```
f, (ax1, ax2) = plt.subplots(1, 2, figsize = (20, 10))
n = len(trainingLoss)
ax1.plot(range(n), trainingLoss, '-', linewidth = '3', label = 'Train
Error')
ax1.plot(range(n), testingLoss, '-', linewidth = '3', label = 'Test
Error')
ax2.plot(range(n), trainingAccuracy, '-', linewidth = '3', label =
'Train Accuracy')
ax2.plot(range(n), testingAccuracy, '-', linewidth  = '3', label =
'Test Accuracy')
ax2.annotate('max accuracy = %s'%(ymax), xy = (xmax, ymax), xytext =
(xmax, ymax+0.15), arrowprops = dict(facecolor = 'black', shrink =
0.05))
ax1.grid(True)
ax2.grid(True)
ax1.legend()
ax2.legend()
f.savefig("./trainTestCurve.png")
```


Train Error
Test Error
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
20
40
60
80
100


max accuracy = 0.918
Train Accuracy
Test Accuracy
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0
20
40
60
80
100