Assignment 2
Machine Learning
BCS 7A

Students Group		
Selected Project Title:		
S.No.	Name	Registration No.
1.	SYED MUHAMMAD SHAHEER ALI SHAH	FA20-BCS-079
2.	FAZEELA REHMAN	FA20-BCS-090
3.	Shumaila Rafique	FA20-BCS-027
4.	Tooba Haider	FA20-BCS-018

Experiments	Learning rate	Optimization Algo	Batch Normalization Layer	Training Loss	Training accuracy
1	1	SGD	No	84.9	0.12
2	0.1	SGD	Yes, AFTER CONVOLUTION LAYER 1 AND 3	28.9	0.77
3	0.01	SGD	No	78.3	0.22
4	1	ADAM	No	2440183.8	0.103
5	0.1	ADAM	Yes, AFTER CONVOLUTION LAYER 1 AND 3	834.7	0.11
6	0.01	ADAM	No	84.9	0.13

Group 5:

Tree Nuts -Image Classification, for reference
https://www.kaggle.com/datasets/gpiosenka/tree-nuts-image-classification
The dataset contains images from 10 classes.
Required Input dimensions are: 200×200

Code:

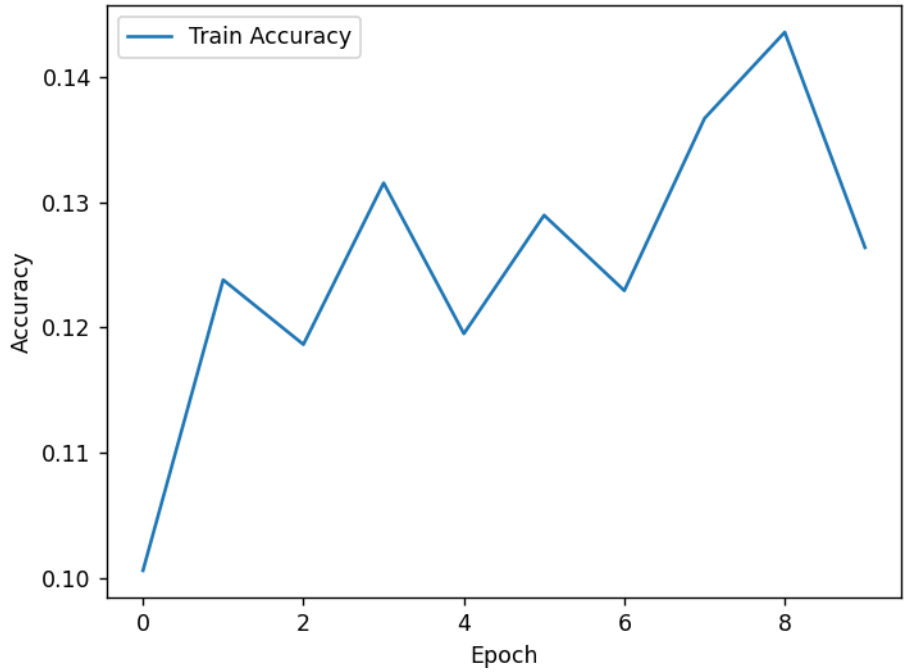
```
import os
import torch
import glob
import torch.nn as nn
from torchvision.transforms import transforms
from torch.utils.data import DataLoader
from torch.optim import Adam
from torch.optim import SGD
import torchvision
import matplotlib.pyplot as plt

def main():
    # Transforms
    transformer = transforms.Compose([
        transforms.Resize((200, 200)),
        transforms.ToTensor(),  # 0-255 to 0-1, numpy to tensors
        transforms.Normalize([0.5, 0.5, 0.5],  # 0-1 to [-1,1] , formula
(x-mean)/std
                             [0.5, 0.5, 0.5])
    ])

    # Path for training and testing directory
    train_path = 'E:/comsat/Comsats/Semester7/machineLearning/Data/train'

    train_loader = DataLoader(
        torchvision.datasets.ImageFolder(train_path,
transform=transformer),
        batch_size=32, shuffle=True
    )
    train_count = len(glob.glob(train_path + '/**/*.jpg'))

    learning_rate_arr = [1, 0.1, 0.01, 1, 0.1, 0.01]
    optimizer_arr = ['SGD', 'SGD', 'SGD', 'Adam', 'Adam', 'Adam']
    batching_arr = [False, True, False, False, True, False]

    for i in range(len(learning_rate_arr)):
        model = ConvNet(num_classes=10, batch=batching_arr[i])
        if optimizer_arr[i] == 'Adam':
            optimizer = Adam(model.parameters(), lr=learning_rate_arr[i])
        else:
            optimizer = SGD(model.parameters(), lr=learning_rate_arr[i])

        optimizer.zero_grad()
        loss_function = nn.CrossEntropyLoss()
        num_epochs = 10
        epoch_arr = [i for i in range(num_epochs)]
        train_acc_arr = []
        loss_arr = []

        for epoch in range(num_epochs):
            train_accuracy = 0.0
            train_loss = 0.0
            model.train()

            for images, labels in train_loader:
                optimizer.zero_grad()
                outputs = model(images)
                loss = loss_function(outputs, labels)
```

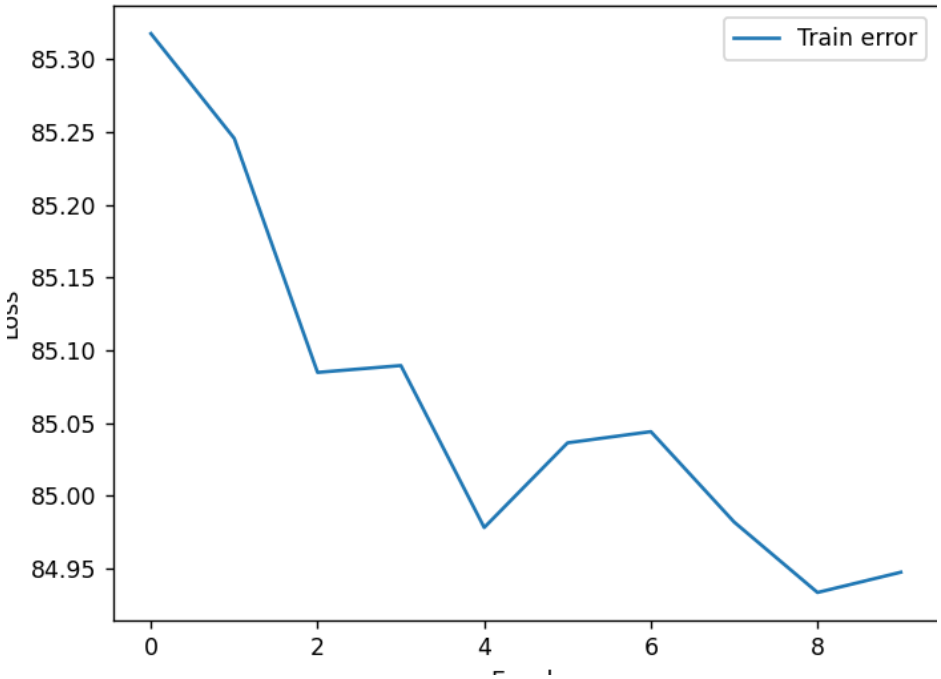
```
                loss.backward()
                optimizer.step()

                train_loss += loss.item()

                per_error, prediction = torch.max(outputs.data, 1)
                train_accuracy += int(torch.sum(prediction == labels.data))

            train_accuracy = train_accuracy / train_count
            train_acc_arr.append(train_accuracy)

            loss_arr.append(train_loss)

            print('Epoch: ' + str(epoch) + ' Train Loss: ' +
str(train_loss) + ' Train Accuracy: ' + str(
                train_accuracy))

        plt.plot(epoch_arr, train_acc_arr, label='Train Accuracy')
        plt.xlabel('Epoch')
        plt.ylabel('Accuracy')
        plt.legend()
        plt.show()

        plt.plot(epoch_arr, loss_arr, label='Train error')
        plt.xlabel('Epoch')
        plt.ylabel('Loss')
        plt.legend()
        plt.show()

class ConvNet(nn.Module):
    def __init__(self, num_classes=10, batch=False):
        super(ConvNet, self).__init__()
        self.isBatch = batch
        # Input shape= (32,3,200,200)
        self.conv1 = nn.Conv2d(in_channels=3, out_channels=12,
kernel_size=3, stride=1, padding=1)
        # Shape= (32,12,200,200)
        self.bn1 = nn.BatchNorm2d(num_features=12)
        # Shape= (32,12,200,200)
        self.relu1 = nn.ReLU()
        # Shape= (32,12,200,200)

        self.conv2 = nn.Conv2d(in_channels=12, out_channels=23,
kernel_size=3, stride=1, padding=1)
        # Shape= (32,23,200,200)
        self.relu2 = nn.ReLU()
        # Shape= (32,23,200,200)
        self.pool1 = nn.MaxPool2d(kernel_size=2)
        # Shape= (32,23,100,100)

        self.conv3 = nn.Conv2d(in_channels=23, out_channels=32,
kernel_size=3, stride=1, padding=1)
        # Shape= (32,32,100,100)
        self.bn2 = nn.BatchNorm2d(num_features=32)
        # Shape= (32,32,100,100)
        self.relu3 = nn.ReLU()
        # Shape= (32,32,100,100)

        self.conv4 = nn.Conv2d(in_channels=32, out_channels=20,
kernel_size=3, stride=1, padding=1)
        # Shape= (32,20,100,100)
```

```
        self.bn3 = nn.BatchNorm2d(num_features=20)
        # Shape= (32,20,100,100)
        self.relu4 = nn.ReLU()
        # Shape= (32,20,100,100)
        self.pool2 = nn.MaxPool2d(kernel_size=2)
        # Shape= (32,20,50,50)

        self.fl = nn.Flatten()
        self.fc1 = nn.Linear(in_features=50 * 50 * 20, out_features=500)
        # self.dropout1 = nn.Dropout(0.5)
        self.fc2 = nn.Linear(in_features=500, out_features=100)
        # self.dropout2 = nn.Dropout(0.5)
        self.fc3 = nn.Linear(in_features=100, out_features=num_classes)

        # Feed forwad function
    def forward(self, input):
        output = self.conv1(input)
        if self.isBatch:
            output = self.bn1(output)
        output = self.relu1(output)

        output = self.conv2(output)
        output = self.relu2(output)

        output = self.pool1(output)

        output = self.conv3(output)
        if self.isBatch:
            output = self.bn2(output)
        output = self.relu3(output)

        output = self.conv4(output)
        output = self.relu4(output)

        output = self.pool2(output)

        output = self.fl(output)
        output = self.fc1(output)
        # output = self.dropout1(output)
        output = self.fc2(output)
        # output = self.dropout2(output)
        output = self.fc3(output)
        return output

main()
```

Experiment 1:

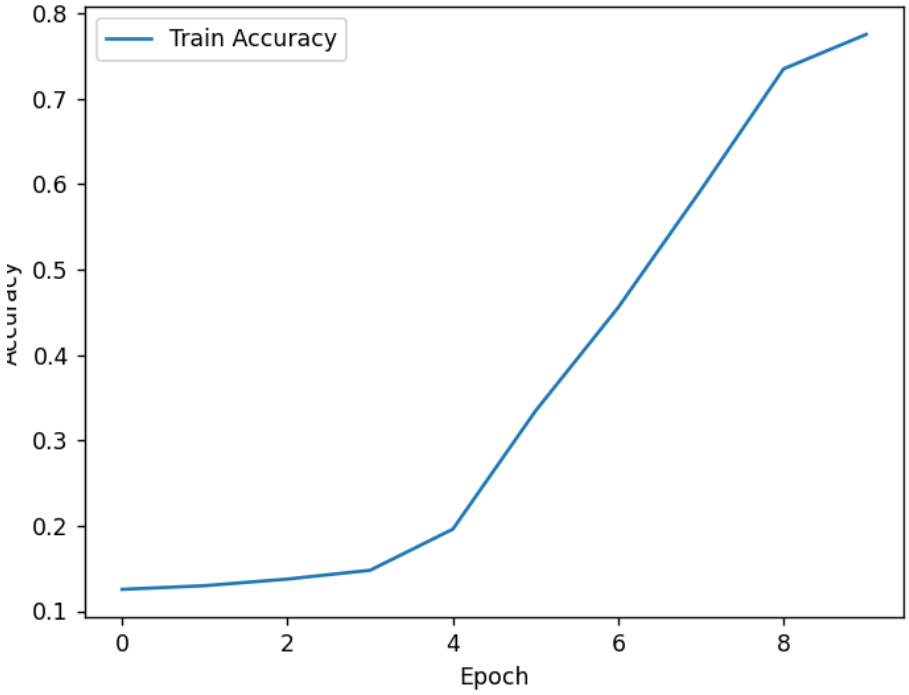
```
Epoch: 0 Train Loss: 85.31753206253052 Train Accuracy: 0.10060189165950129
Epoch: 1 Train Loss: 85.24559211730957 Train Accuracy: 0.12381771281169389
Epoch: 2 Train Loss: 85.08484315872192 Train Accuracy: 0.11865864144453998
Epoch: 3 Train Loss: 85.0895745754242 Train Accuracy: 0.13155631986242478
Epoch: 4 Train Loss: 84.97832012176514 Train Accuracy: 0.11951848667239896
Epoch: 5 Train Loss: 85.03647589683533 Train Accuracy: 0.1289767841788478
Epoch: 6 Train Loss: 85.04420137405396 Train Accuracy: 0.12295786758383491
Epoch: 7 Train Loss: 84.98206901550293 Train Accuracy: 0.13671539122957868
Epoch: 8 Train Loss: 84.93370747566223 Train Accuracy: 0.14359415305245055
Epoch: 9 Train Loss: 84.9476363658905 Train Accuracy: 0.12639724849527084
```

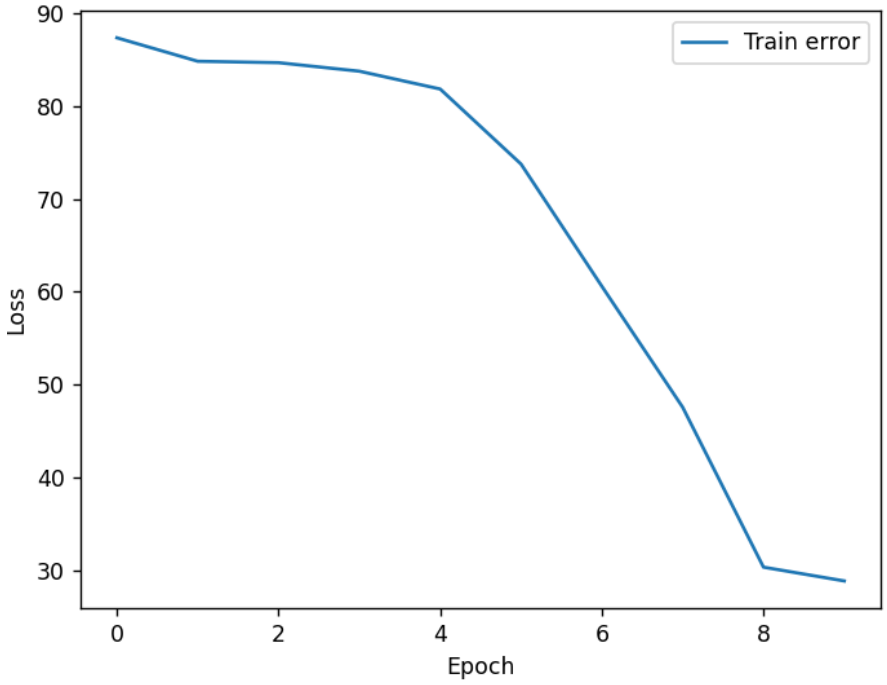




Experiment 2:

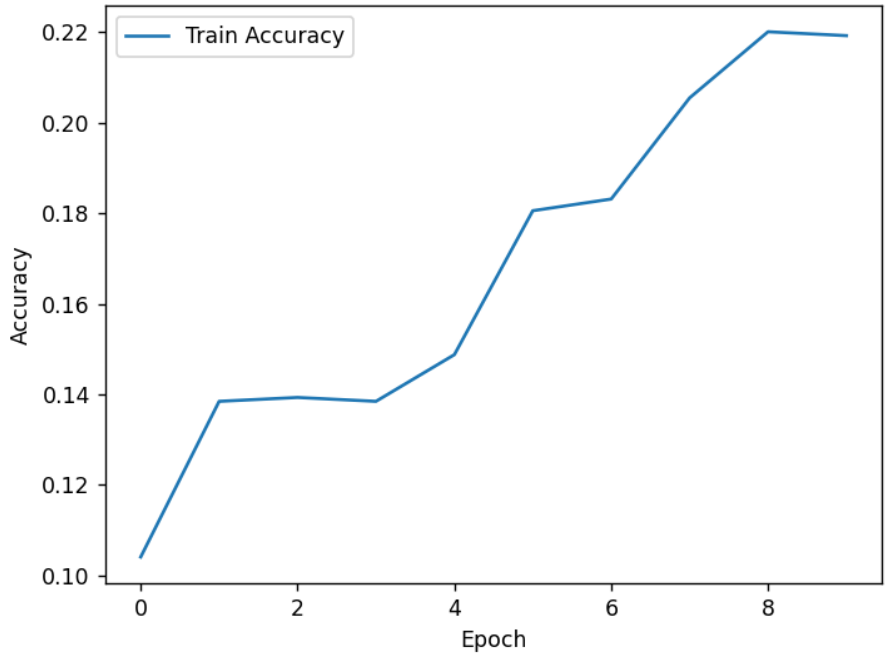
```
Epoch: 3 Train Loss: 83.71506762504578 Train Accuracy: 0.14789337919174547
Epoch: 4 Train Loss: 81.79925334453583 Train Accuracy: 0.19604471195184867
Epoch: 5 Train Loss: 73.7357827425003 Train Accuracy: 0.3344797936371453
Epoch: 6 Train Loss: 60.60324418544769 Train Accuracy: 0.45571797076526227
Epoch: 7 Train Loss: 47.60061639547348 Train Accuracy: 0.5932932072227
Epoch: 8 Train Loss: 30.411908984184265 Train Accuracy: 0.7351676698194325
Epoch: 9 Train Loss: 28.930124059319496 Train Accuracy: 0.7755803955288049
```

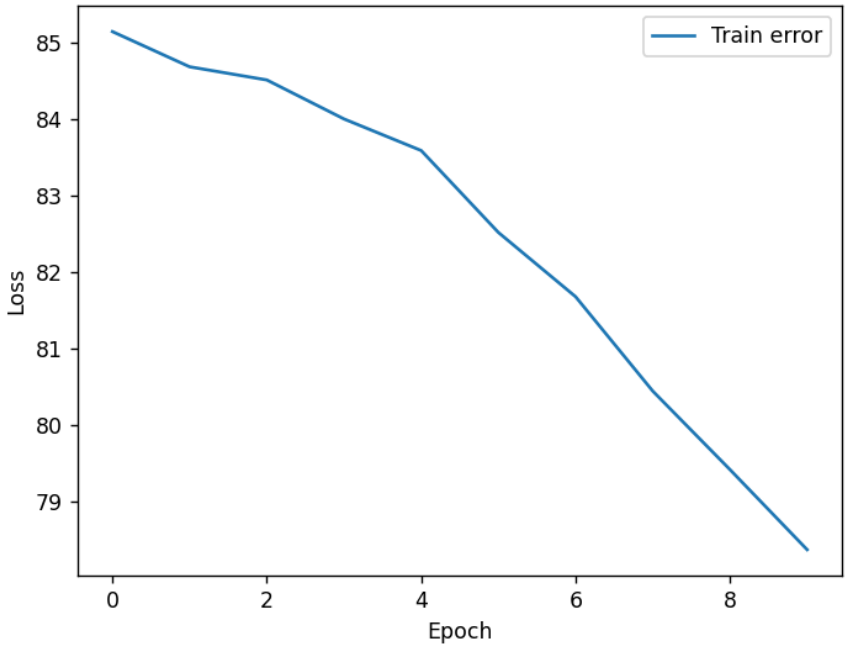




Experiment 3:

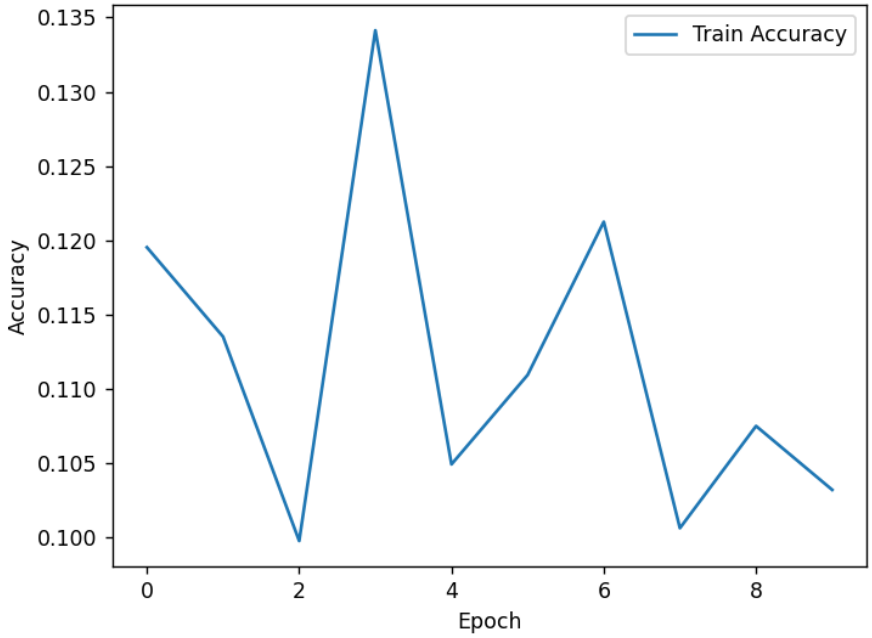
```
Epoch: 0 Train Loss: 85.15265369415283 Train Accuracy: 0.10404127257093723
Epoch: 1 Train Loss: 84.69225239753723 Train Accuracy: 0.13843508168529664
Epoch: 2 Train Loss: 84.52020406723022 Train Accuracy: 0.13929492691315562
Epoch: 3 Train Loss: 84.0100359916687 Train Accuracy: 0.13843508168529664
Epoch: 4 Train Loss: 83.59698247909546 Train Accuracy: 0.14875322441960448
Epoch: 5 Train Loss: 82.52565670013428 Train Accuracy: 0.18056749785038692
Epoch: 6 Train Loss: 81.6855616569519 Train Accuracy: 0.1831470335339639
Epoch: 7 Train Loss: 80.45085096359253 Train Accuracy: 0.2055030094582975
Epoch: 8 Train Loss: 79.4245092868805 Train Accuracy: 0.22012037833190026
Epoch: 9 Train Loss: 78.37690782546997 Train Accuracy: 0.21926053310404128
```

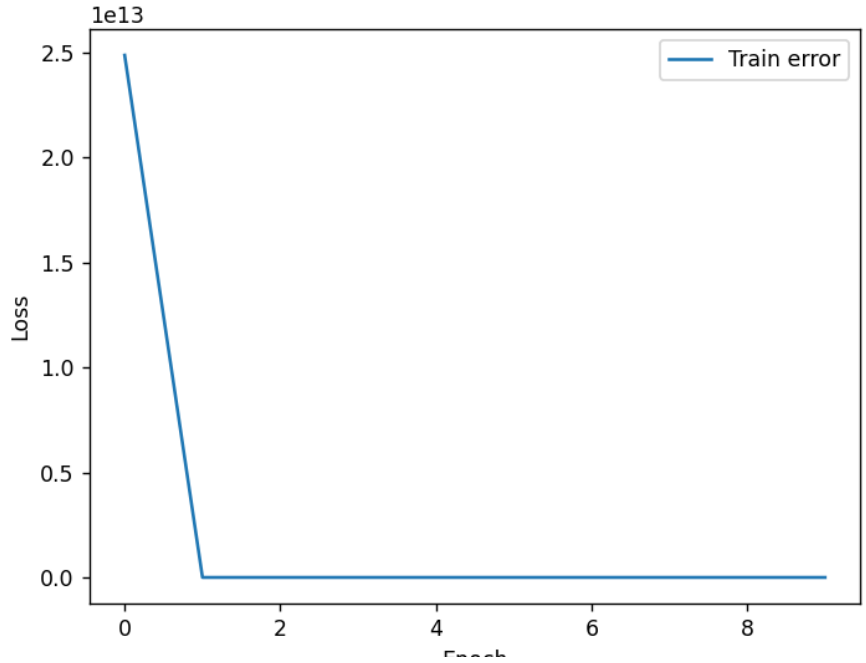




Experiment 4:

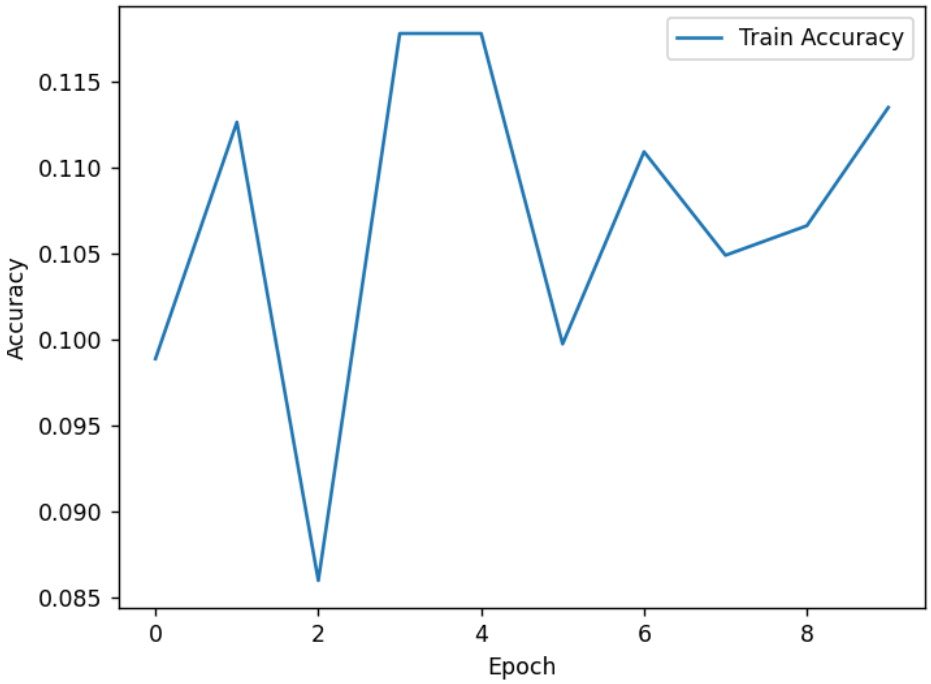
```
Epoch: 0 Train Loss: 24874975498511.74 Train Accuracy: 0.11951848667239896
Epoch: 1 Train Loss: 3309465.521484375 Train Accuracy: 0.11349957007738606
Epoch: 2 Train Loss: 3366840.458984375 Train Accuracy: 0.0997420464316423
Epoch: 3 Train Loss: 3714874.5234375 Train Accuracy: 0.13413585554600171
Epoch: 4 Train Loss: 4398447.318359375 Train Accuracy: 0.10490111779879621
Epoch: 5 Train Loss: 9832006.453125 Train Accuracy: 0.11092003439380911
Epoch: 6 Train Loss: 6374401.5625 Train Accuracy: 0.12123817712811694
Epoch: 7 Train Loss: 2684783.853515625 Train Accuracy: 0.10060189165950129
Epoch: 8 Train Loss: 2219293.66796875 Train Accuracy: 0.10748065348237318
Epoch: 9 Train Loss: 2440183.05078125 Train Accuracy: 0.10318142734307825
```

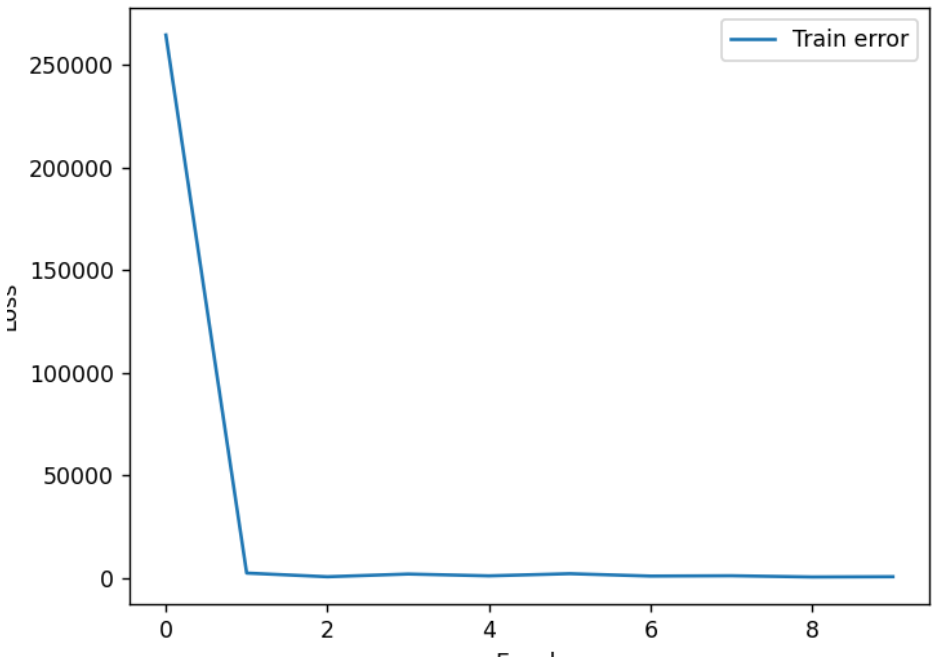




Experiment 5:

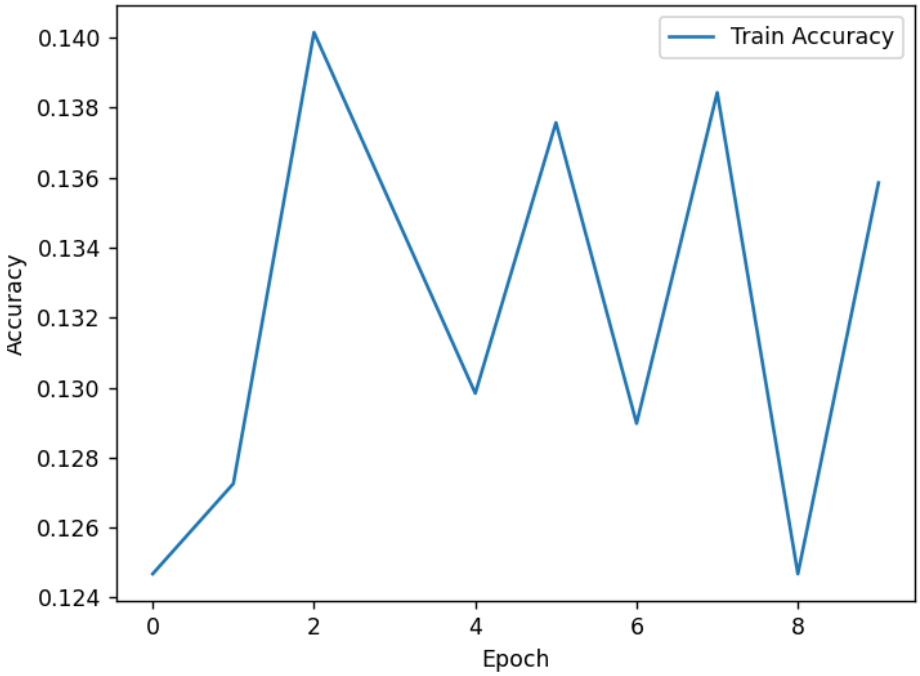
```
Epoch: 0 Train Loss: 264395.80838871 Train Accuracy: 0.09888220120378331
Epoch: 1 Train Loss: 2596.9485778808594 Train Accuracy: 0.11263972484952708
Epoch: 2 Train Loss: 805.3549251556396 Train Accuracy: 0.08598452278589853
Epoch: 3 Train Loss: 2138.601625919342 Train Accuracy: 0.117798796216681
Epoch: 4 Train Loss: 1240.1029124259949 Train Accuracy: 0.117798796216681
Epoch: 5 Train Loss: 2305.8961877822876 Train Accuracy: 0.0997420464316423
Epoch: 6 Train Loss: 1132.258466720581 Train Accuracy: 0.11092003439380911
Epoch: 7 Train Loss: 1293.3165192604065 Train Accuracy: 0.10490111779879621
Epoch: 8 Train Loss: 693.9559087753296 Train Accuracy: 0.10662080825451418
Epoch: 9 Train Loss: 834.7074480056763 Train Accuracy: 0.11349957007738606
```

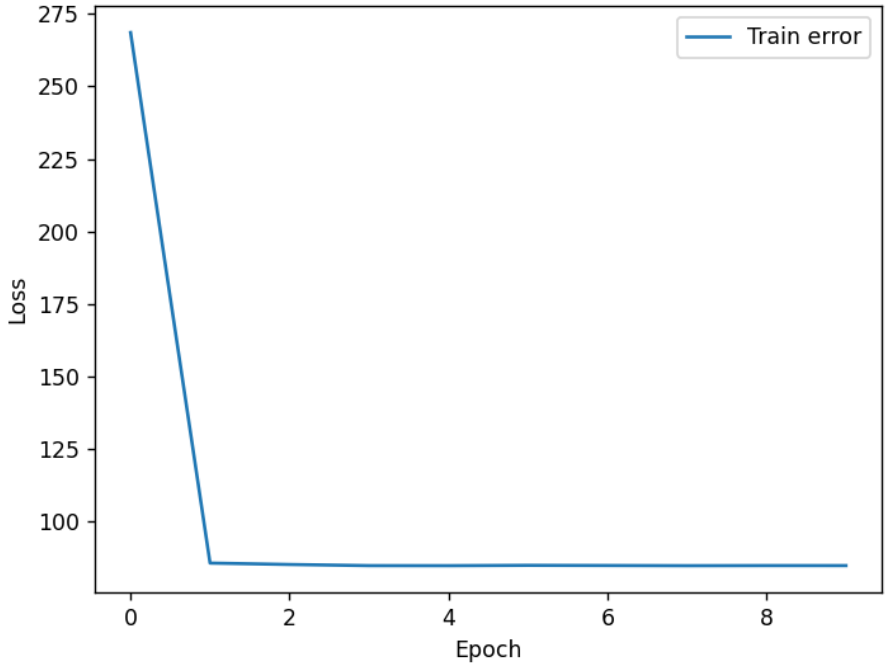




Experiment 6:

```
Epoch: 0 Train Loss: 268.5272362232208 Train Accuracy: 0.12467755803955288
Epoch: 1 Train Loss: 85.7963981628418 Train Accuracy: 0.12725709372312985
Epoch: 2 Train Loss: 85.31355023384094 Train Accuracy: 0.14015477214101463
Epoch: 3 Train Loss: 84.91426038742065 Train Accuracy: 0.1349957007738607
Epoch: 4 Train Loss: 84.89566707611084 Train Accuracy: 0.1298366294067068
Epoch: 5 Train Loss: 85.01969194412231 Train Accuracy: 0.13757523645743766
Epoch: 6 Train Loss: 84.95802640914917 Train Accuracy: 0.1289767841788478
Epoch: 7 Train Loss: 84.88926529884338 Train Accuracy: 0.13843508168529664
Epoch: 8 Train Loss: 84.93296933174133 Train Accuracy: 0.12467755803955288
Epoch: 9 Train Loss: 84.92508602142334 Train Accuracy: 0.1358555460017197
```





Observations and Changes Made to Improve Model Accuracy:

As accuracy of my model is very low in all the experiments but if we decrease the learning rate and incorporate batch normalization, I observed an improvement in the accuracy of my model. I made several parameter changes to enhance the accuracy rate, and one notable observation is that setting the learning rate to 0.0001 and applying batch normalization yields better results.

```
Epoch: 0 Train Loss: 88.1453001499176 Train Accuracy: 0.15219260533104043
Epoch: 1 Train Loss: 63.97628104686737 Train Accuracy: 0.4660361134995701
Epoch: 2 Train Loss: 45.28483074903488 Train Accuracy: 0.6362854686156492
Epoch: 3 Train Loss: 22.70558586716652 Train Accuracy: 0.883061049011178
Epoch: 4 Train Loss: 9.55778743326664 Train Accuracy: 0.9587274290627688
Epoch: 5 Train Loss: 3.5811375733464956 Train Accuracy: 0.9948409286328461
Epoch: 6 Train Loss: 1.9744951911270618 Train Accuracy: 0.9965606190885641
Epoch: 7 Train Loss: 1.3644322315230966 Train Accuracy: 0.9974204643164231
Epoch: 8 Train Loss: 1.4282833747565746 Train Accuracy: 0.9965606190885641
Epoch: 9 Train Loss: 4.755986938253045 Train Accuracy: 0.9707652622527945
```