Artificial Intelligence Lab Work (4)
レポート解答用紙（Report Answer Sheet）

学生証番号（Student ID): 18521071
名前(Name): ズオン・ミン・ルオン(Duong Minh Luong)

問題 1.

(プログラム)

```
##ライブラリの読み込み
import numpy as np
import matplotlib.pyplot as plt
import torch
import torch.nn.functional as F
import torchvision as tv

#訓練データとテストデータの読み込み (初めて実行するときはデータをネットからダウンロードする)
train_dataset=tv.datasets.MNIST(root="./",train=True,transform=tv.transform
s.ToTensor(),download=True)
test_dataset=tv.datasets.MNIST(root="./",train=False,transform=tv.transform
s.ToTensor(),download=True)

#訓練データとテストデータのミニバッチ処理・ミニバッチサイズ=100・データの順番をシャッフル
train_loader=torch.utils.data.DataLoader(dataset=train_dataset,batch_size=1
00,shuffle=True)
test_loader=torch.utils.data.DataLoader(dataset=test_dataset,batch_size=100
,shuffle=False)

MODELNAME = 'mnist.model'
EPOCH = 10
DEVICE = 'cuda' if torch.cuda.is_available() else 'cpu'
#GPU が利用可能なら DEVICE=”cuda”
#CPU を利用するのなら DEVICE=”cpu

class MNIST(torch.nn.Module):
    def __init__(self):
        super(MNIST,self).__init__()
        self.l1 = torch.nn.Linear(784,300)
        self.l2 = torch.nn.Linear(300,300)
```

```
        self.l3 = torch.nn.Linear(300,10)
    def forward(self,x): #forward 計算のときに forward()が呼ばれる。同時にネットワークを構築
        h = F.relu(self.l1(x))
        h = F.relu(self.l2(h))
        y = self.l3(h)
        return y

def train_MNIST():
    model = MNIST().to(DEVICE)
    optimizer = torch.optim.Adam(model.parameters())
    for epoch in range(EPOCH):
        loss =0
        for images, labels in train_loader: #データをミニバッチサイズに切り出す
            images = images.view(-1,28*28).to(DEVICE)
           #(100X1X28X28)から (100× 784)に変形
            labels = labels.to(DEVICE)
            optimizer.zero_grad()
            y = model(images)
            batchloss =F.cross_entropy(y, labels)
            batchloss.backward()
            optimizer.step()
            loss = loss + batchloss.item()
        print('epoch',epoch,': loss',loss)
    torch.save(model.state_dict(),MODELNAME)

def test_MNIST():
    total = len(test_loader.dataset)
    correct = 0
    model = MNIST().to('cpu')
    model.load_state_dict(torch.load(MODELNAME)) #ファイルに保存したモデルをロード
    model.eval()
    for images, labels in test_loader: #テストデータに対してループ
        images = images.view(-1,28*28).to('cpu')
        y = model(images)
        pred_labels = y.max(dim=1)[1]
        correct = correct + (pred_labels ==labels).sum()
    print('correct:',correct.item())
```

```
    print('total:',total)
    print('accuracy:',(correct.item()/float(total)))

import time
start = time.time()
train_MNIST()
print("Completed:", time.time()-start)

test_MNIST()
```

（実行結果）

```
[8]  import time
     start = time.time()
     train_MNIST()
     print("Completed:", time.time()-start)
```

```
epoch 0 : loss 172.08707903698087
epoch 1 : loss 62.786055171862245
epoch 2 : loss 41.05835762154311
epoch 3 : loss 29.566109999548644
epoch 4 : loss 22.017877524369396
epoch 5 : loss 18.990374520304613
epoch 6 : loss 12.777088044938864
epoch 7 : loss 11.948766264496953
epoch 8 : loss 11.166220424027415
epoch 9 : loss 8.509438167369808
Completed: 83.63480639457703
```

```
[9]  test_MNIST()
```

```
correct: 9767
total: 10000
accuracy: 0.9767
```

問題 2

(プログラム)

```
##ライブラリの読み込み
import numpy as np
import matplotlib.pyplot as plt
import torch
import torch.nn.functional as F
import torchvision as tv
import time
train_dataset =
tv.datasets.CIFAR10(root="./",train=True,transform=tv.transforms.ToTensor(),download=True)
test_dataset = tv.datasets.CIFAR10(root="./",
train=False,transform=tv.transforms.ToTensor(),download=True)
train_loader = torch.utils.data.DataLoader(dataset=train_dataset, batch_size=100, shuffle=True)
test_loader = torch.utils.data.DataLoader(dataset=test_dataset, batch_size=100, shuffle=False)

MODELNAME = 'CIFAR10.model'
EPOCH = 10
DEVICE = 'cuda' if torch.cuda.is_available() else 'cpu'

class CIFAR10(torch.nn.Module):
    def __init__(self):
        super(CIFAR10,self).__init__()
        self.l1 = torch.nn.Linear(3*32*32,300)
        self.l2 = torch.nn.Linear(300,300)
        self.l3 = torch.nn.Linear(300,10)
    def forward(self,x):
        h = F.relu(self.l1(x))
        h = F.relu(self.l2(h))
        y = self.l3(h)
        return y

def train_CIFAR10():
    model = CIFAR10().to(DEVICE)
    optimizer = torch.optim.Adam(model.parameters())
    for epoch in range(EPOCH):
        loss =0
        for images, labels in train_loader:
```

```
            images = images.view(-1,3*32*32).to(DEVICE)
            labels = labels.to(DEVICE)
            optimizer.zero_grad()
            y = model(images)
            batchloss =F.cross_entropy(y, labels)
            batchloss.backward()
            optimizer.step()
            loss = loss + batchloss.item()
        print('epoch',epoch,': loss',loss)
    torch.save(model.state_dict(),MODELNAME)

def test_CIFAR10():
    total = len(test_loader.dataset)
    correct = 0
    model = CIFAR10().to('cpu')
    model.load_state_dict(torch.load(MODELNAME))
    model.eval()
    for images, labels in test_loader:
        images = images.view(-1,3*32*32).to('cpu')
        y = model(images)
        pred_labels = y.max(dim=1)[1]
        correct = correct + (pred_labels ==labels).sum()
    print('correct:',correct.item())
    print('total:',total)
    print('accuracy:',(correct.item()/float(total)))



start = time.time()
train_CIFAR10()
print("Completed:", time.time()-start)

test_CIFAR10()
```

（実行結果）

```
[15]
start = time.time()
train_CIFAR10()
print("Completed:", time.time()-start)
```

```
epoch 0 : loss 919.097843170166
epoch 1 : loss 826.5155259370804
epoch 2 : loss 781.8015896081924
epoch 3 : loss 756.1094368696213
epoch 4 : loss 731.4593398571014
epoch 5 : loss 714.1324229240417
epoch 6 : loss 697.9319487810135
epoch 7 : loss 683.8895984888077
epoch 8 : loss 669.7166402339935
epoch 9 : loss 656.0325155258179
Completed: 84.28762316703796
```

```
[16] test_CIFAR10()
```

```
correct: 5057
total: 10000
accuracy: 0.5057
```

問題 3

(プログラム)

```
##ライブラリの読み込み
import numpy as np
import matplotlib.pyplot as plt
import torch
import torch.nn.functional as F
import torchvision as tv
import time
train_dataset = tv.datasets.CIFAR10(root="./", train=True,transform=tv.transforms.ToTensor(),download=True)
test_dataset = tv.datasets.CIFAR10(root="./", train=False,transform=tv.transforms.ToTensor(),download=True)
train_loader = torch.utils.data.DataLoader(dataset=train_dataset, batch_size=100, shuffle=True)
test_loader = torch.utils.data.DataLoader(dataset=test_dataset, batch_size=100, shuffle=False)

MODELNAME = 'CIFAR10_2D.model'
EPOCH = 10
DEVICE = 'cuda' if torch.cuda.is_available() else 'cpu'

class CIFAR10_2D(torch.nn.Module):
    def __init__(self):
        super(CIFAR10_2D,self).__init__()
        self.l1 = torch.nn.Conv2d(3,16,5)
        self.l2 = torch.nn.Linear(16*28*28,300)
        self.l3 = torch.nn.Linear(300,10)
    def forward(self,x):
        h = F.relu(self.l1(x))
        h = torch.flatten(h,start_dim=1)
        h = F.relu(self.l2(h))
        y = self.l3(h)
        return y

def train_2D():
    model = CIFAR10_2D().to(DEVICE)
    optimizer = torch.optim.Adam(model.parameters())
    for epoch in range(EPOCH):
        loss =0
```

```
            for images, labels in train_loader:
                images = images.view(-1,3,32,32).to(DEVICE)
                labels = labels.to(DEVICE)
                optimizer.zero_grad()
                y = model(images)
                batchloss =F.cross_entropy(y, labels)
                batchloss.backward()
                optimizer.step()
                loss = loss + batchloss.item()
            print('epoch',epoch,': loss',loss)
        torch.save(model.state_dict(),MODELNAME)

def test_2D():
        total = len(test_loader.dataset)
        correct = 0
        model = CIFAR10_2D().to('cpu')
        model.load_state_dict(torch.load(MODELNAME))
        model.eval()
        for images, labels in test_loader:
            images = images.view(-1,3,32,32).to('cpu')
            y = model(images)
            pred_labels = y.max(dim=1)[1]
            correct = correct + (pred_labels ==labels).sum()
        print('correct:',correct.item())
        print('total:',total)
        print('accuracy:',(correct.item()/float(total)))



start = time.time()
train_2D()
print("Completed:", time.time()-start)

test_2D()
```

（実行結果）

```
[29]
start = time.time()
train_2D()
print("Completed:", time.time()-start)
```

```
epoch 0 : loss 750.4988223314285
epoch 1 : loss 613.6148973107338
epoch 2 : loss 545.5721400976181
epoch 3 : loss 478.25372391939163
epoch 4 : loss 418.1068558692932
epoch 5 : loss 354.16109389066696
epoch 6 : loss 292.4664245247841
epoch 7 : loss 238.8138920366764
epoch 8 : loss 187.71668453514576
epoch 9 : loss 141.24120596051216
Completed: 117.6355574131012
```

```
test_2D()
```

```
correct: 5998
total: 10000
accuracy: 0.5998
```