Artificial Intelligence Lab Work (3)

レポート解答用紙（Report Answer Sheet）

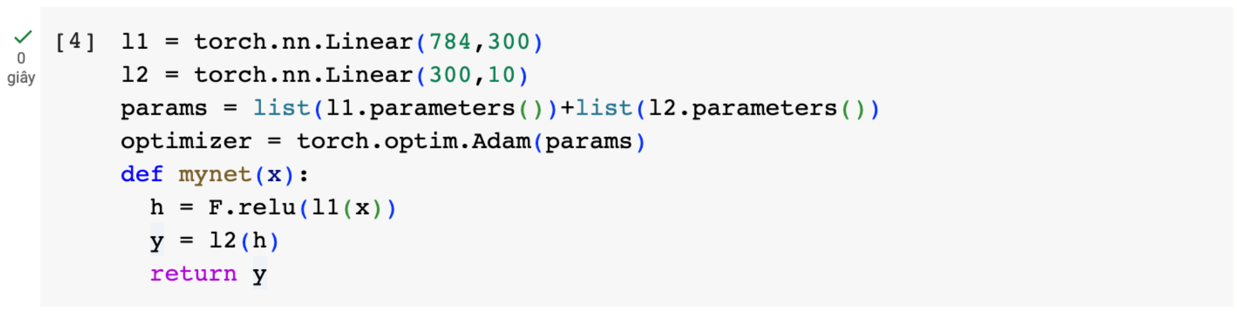
学生証番号（Student ID): 20521150

名前(Name): Pham Quoc Cuong

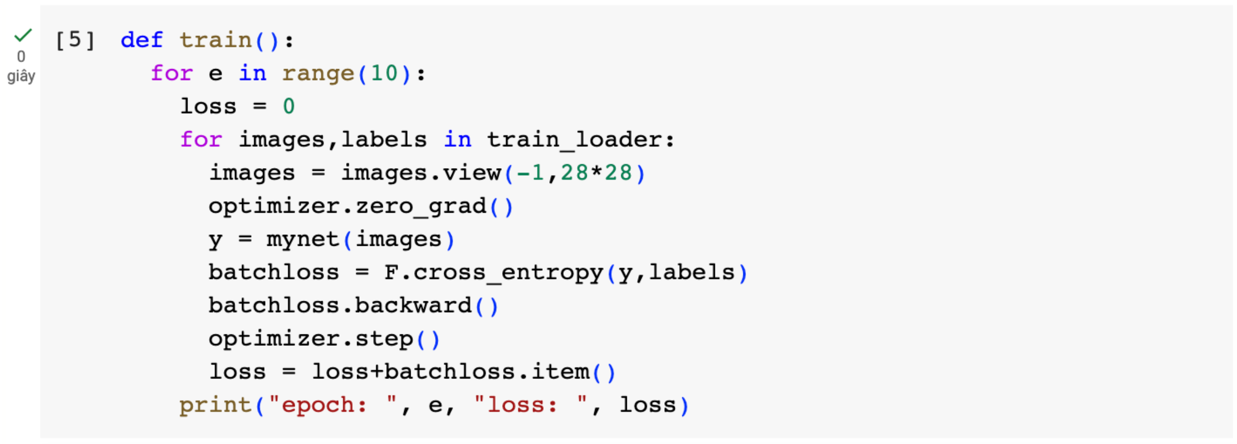
問題 1.

（プログラム）

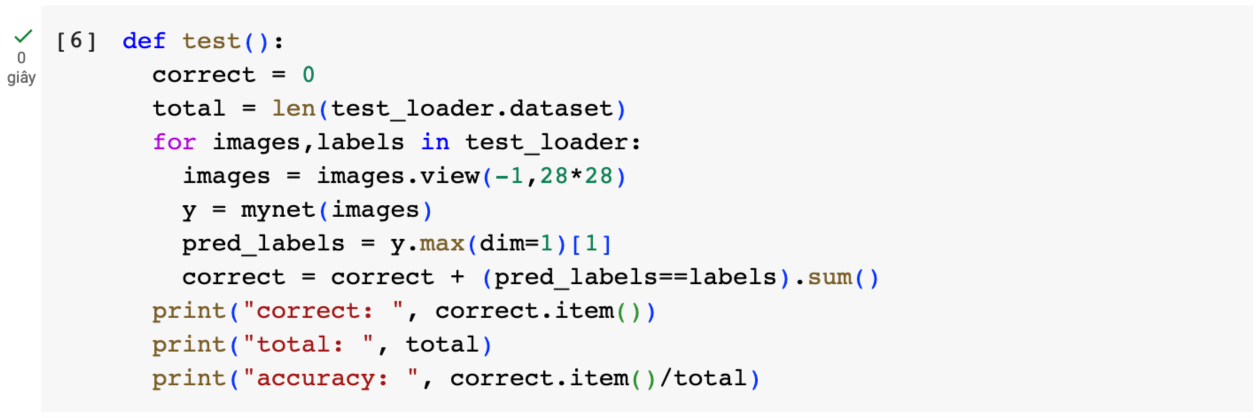
第 9 回の講義資料に基づいて MNIST の学習・推論プログラムを実装し .



```
[4]  l1 = torch.nn.Linear(784,300)
     l2 = torch.nn.Linear(300,10)
     params = list(l1.parameters())+list(l2.parameters())
     optimizer = torch.optim.Adam(params)
     def mynet(x):
       h = F.relu(l1(x))
       y = l2(h)
       return y
```

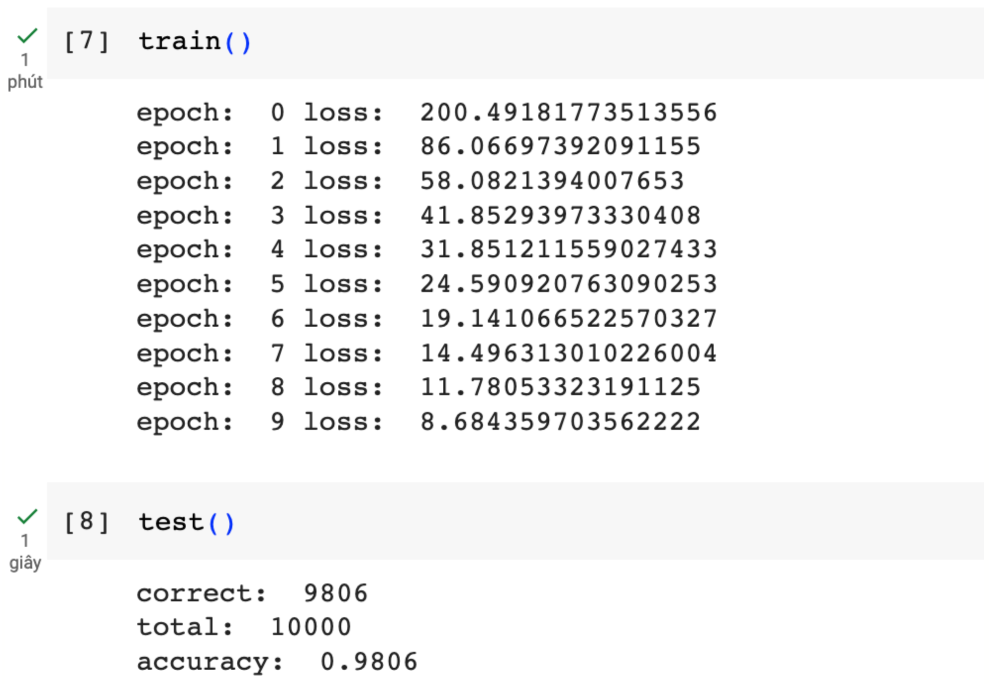


```
[5]  def train():
       for e in range(10):
         loss = 0
         for images,labels in train_loader:
           images = images.view(-1,28*28)
           optimizer.zero_grad()
           y = mynet(images)
           batchloss = F.cross_entropy(y,labels)
           batchloss.backward()
           optimizer.step()
           loss = loss+batchloss.item()
         print("epoch: ", e, "loss: ", loss)
```



```
[6]  def test():
       correct = 0
       total = len(test_loader.dataset)
       for images,labels in test_loader:
         images = images.view(-1,28*28)
         y = mynet(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()/total)
```

（実行結果）



```
[7]  train()

     epoch:  0 loss:  200.49181773513556
     epoch:  1 loss:  86.06697392091155
     epoch:  2 loss:  58.0821394007653
     epoch:  3 loss:  41.85293973330408
     epoch:  4 loss:  31.851211559027433
     epoch:  5 loss:  24.590920763090253
     epoch:  6 loss:  19.141066522570327
     epoch:  7 loss:  14.496313010226004
     epoch:  8 loss:  11.78053323191125
     epoch:  9 loss:  8.684359703562222
```

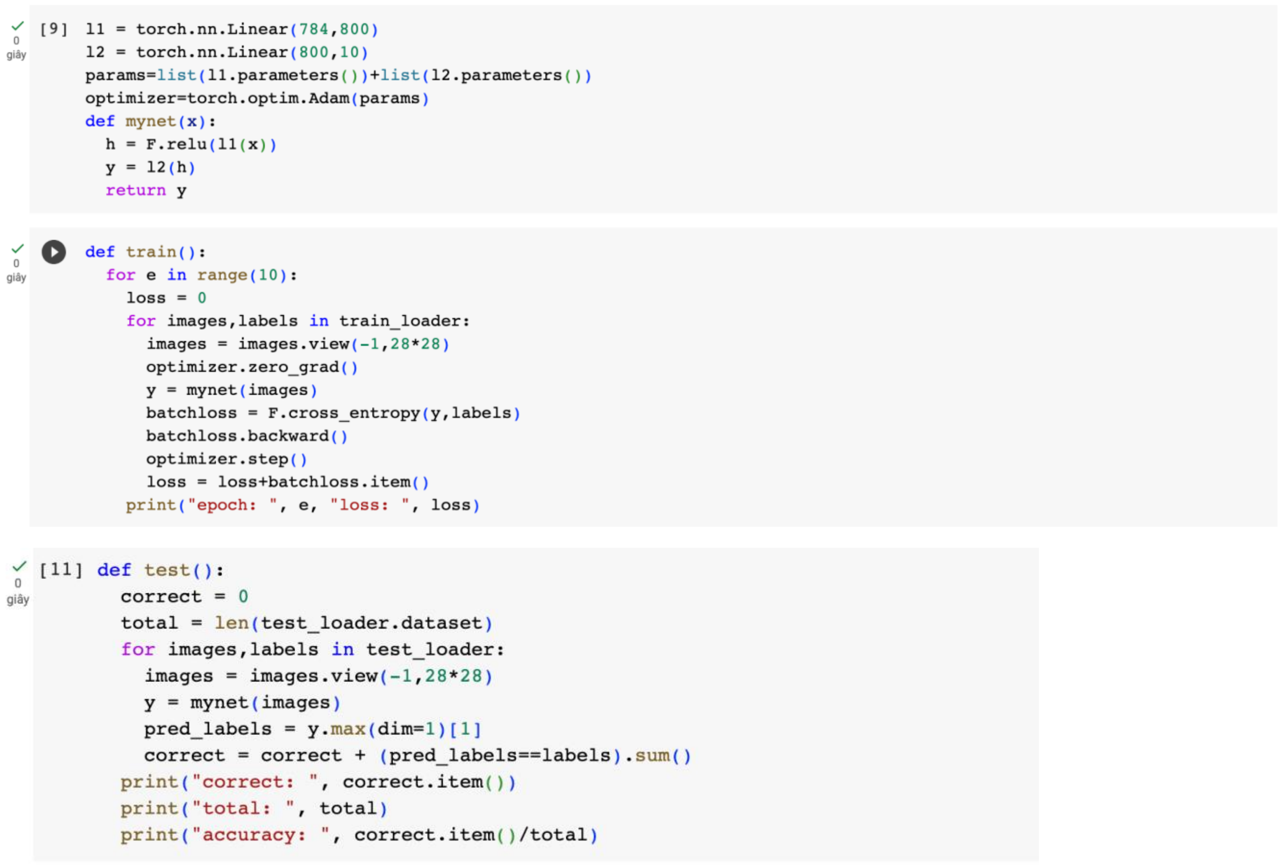
```
[8]  test()

     correct:  9806
     total:  10000
     accuracy:  0.9806
```

問題 2

（プログラム）

第 9 回の講義資料での MNIST の学習・推論プログラムは中間層が 300 次元であった。中間層 を 800 次元に変更し、そのプログラム (.py)とコンソールに表示される実行結果を word ファイルに貼り 付けて提出せよ。 (300 次元の場合に比べて誤差が大きく減る。精度は若干良くなるがだいたい同じ。)

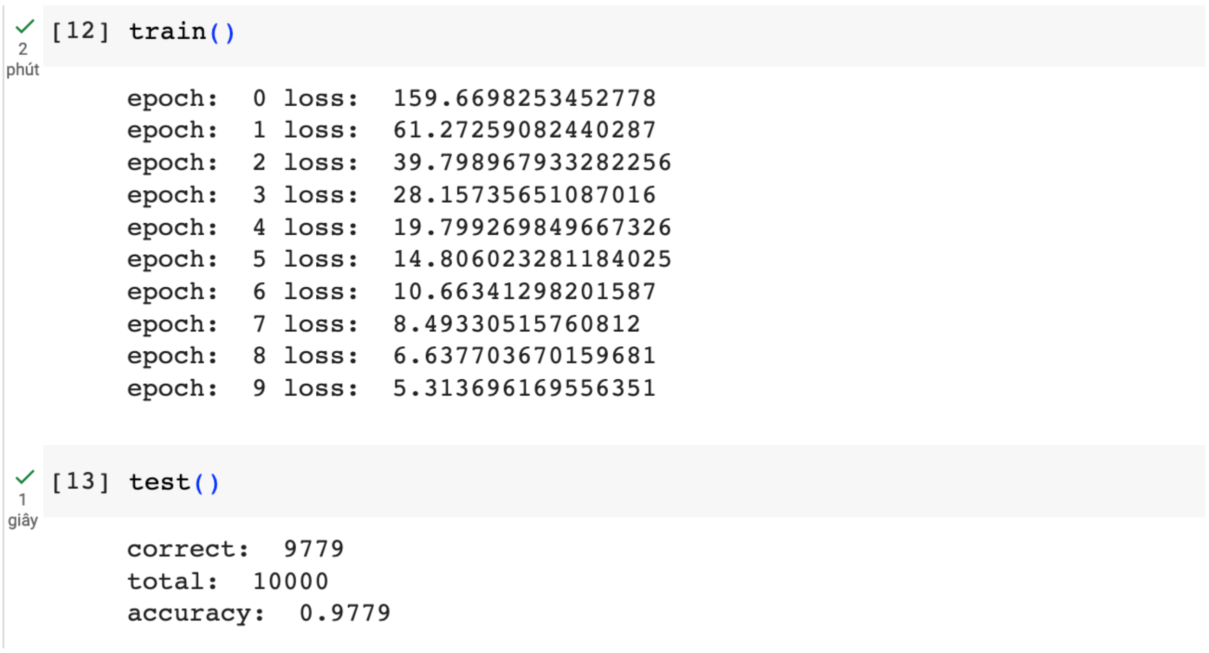


```
l1 = torch.nn.Linear(784,800)
l2 = torch.nn.Linear(800,10)
params=list(l1.parameters())+list(l2.parameters())
optimizer=torch.optim.Adam(params)
def mynet(x):
  h = F.relu(l1(x))
  y = l2(h)
  return y
```

```
def train():
  for e in range(10):
    loss = 0
    for images,labels in train_loader:
      images = images.view(-1,28*28)
      optimizer.zero_grad()
      y = mynet(images)
      batchloss = F.cross_entropy(y,labels)
      batchloss.backward()
      optimizer.step()
      loss = loss+batchloss.item()
    print("epoch: ", e, "loss: ", loss)
```

```
def test():
  correct = 0
  total = len(test_loader.dataset)
  for images,labels in test_loader:
    images = images.view(-1,28*28)
    y = mynet(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()/total)
```

（実行結果）



```
[12] train()

     epoch:  0 loss:  159.6698253452778
     epoch:  1 loss:  61.27259082440287
     epoch:  2 loss:  39.798967933282256
     epoch:  3 loss:  28.15735651087016
     epoch:  4 loss:  19.799269849667326
     epoch:  5 loss:  14.806023281184025
     epoch:  6 loss:  10.66341298201587
     epoch:  7 loss:  8.49330515760812
     epoch:  8 loss:  6.637703670159681
     epoch:  9 loss:  5.313696169556351
```

```
[13] test()

     correct:  9779
     total:  10000
     accuracy:  0.9779
```