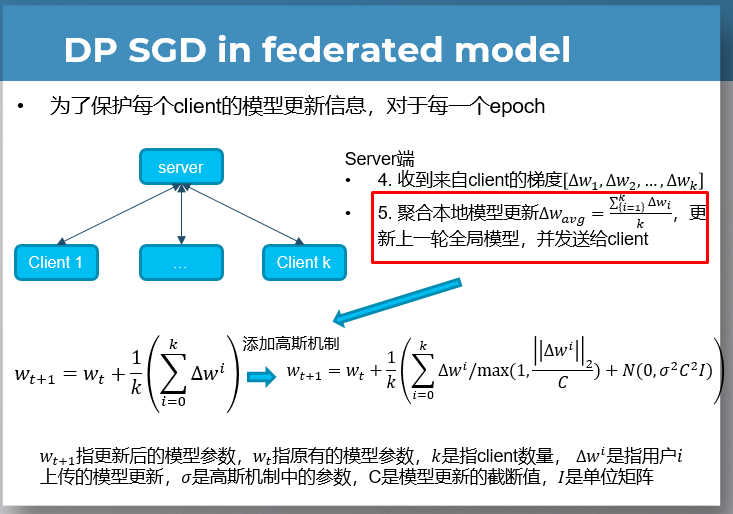
# 数据隐私Lab2实验报告

**PB18111699**

**魏钊**

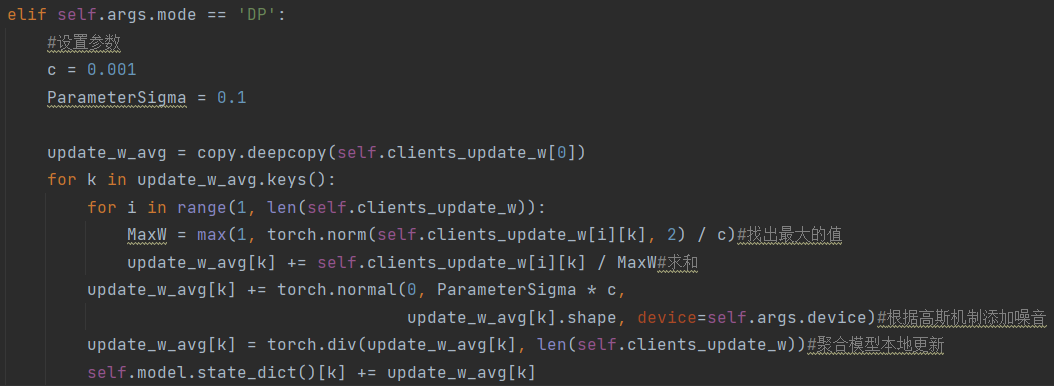
Part1

根据PPT



Server:

在原来代码的基础上通过加噪实现DP：



Client:

和Plain处理一样。

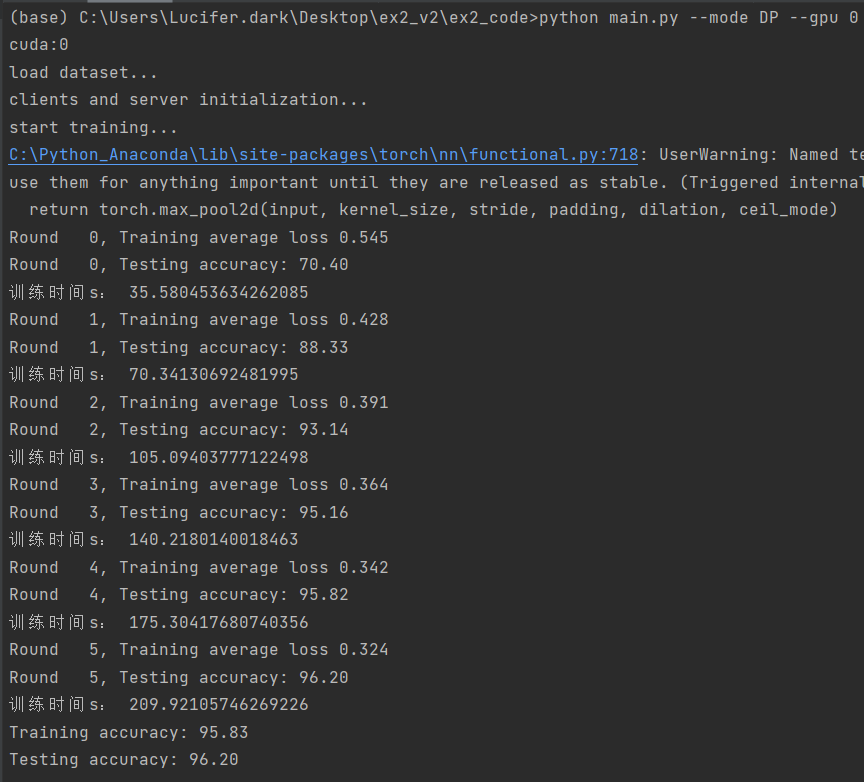
Train：



update:



运行结果：（这里是用GPU进行训练）

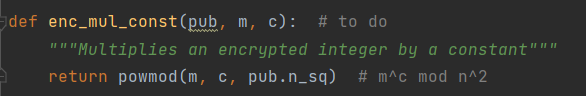
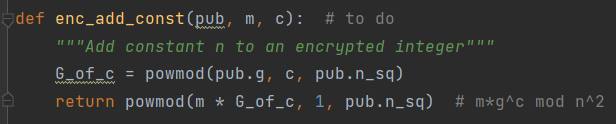
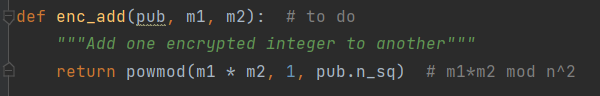
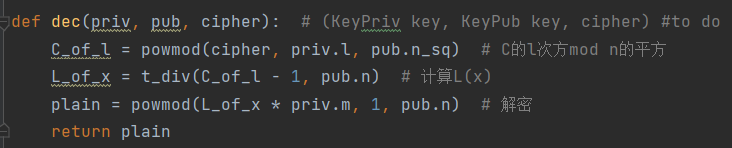
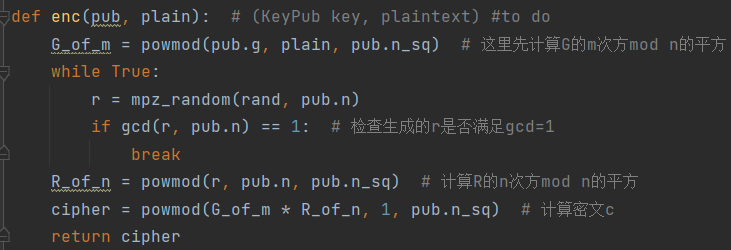
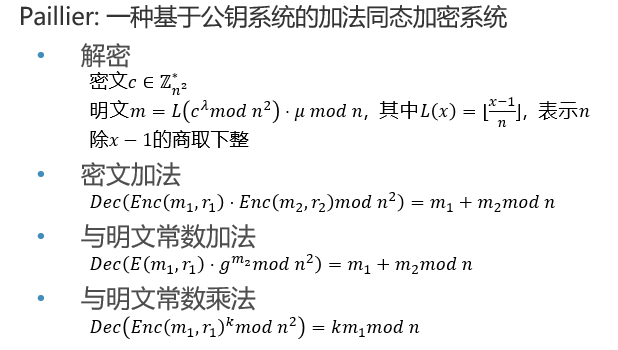
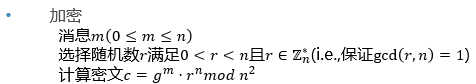


可以看出这里已经训练到收敛。

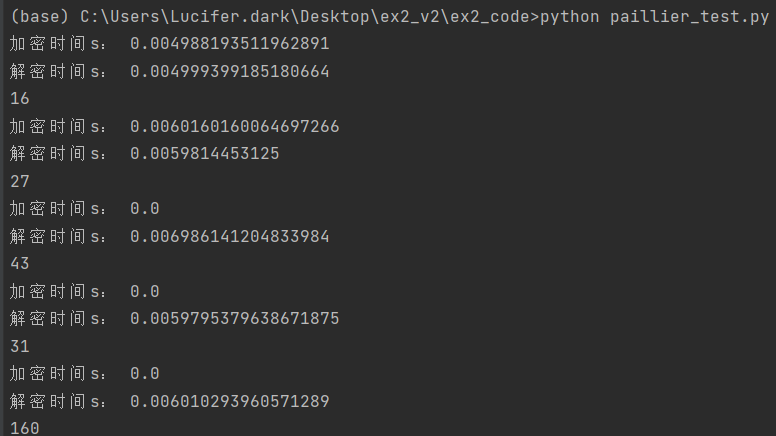
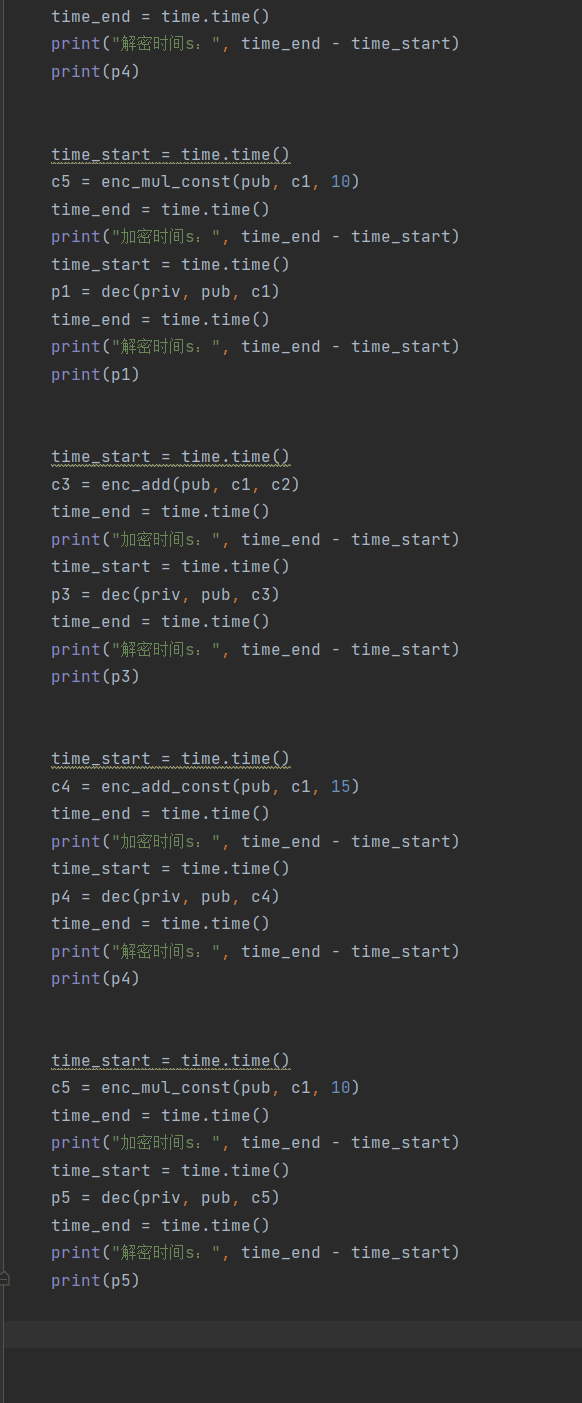
Part2

任务1：

根据PPT：



测试：

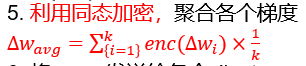


结果正确。

任务2：

Server:

根据PPT：



elif self.args.mode == 'Paillier':  
 update\_w\_avg = copy.deepcopy(self.clients\_update\_w[0])  
 for k in update\_w\_avg.keys():  
 for i in range(1, len(self.clients\_update\_w)):  
 for n in range(len(update\_w\_avg[k])):  
 update\_w\_avg[k][n] += self.clients\_update\_w[i][k][n] # 求和  
 for n in range(len(update\_w\_avg[k])):  
 update\_w\_avg[k][n] /= len(self.clients\_update\_w) # 同态加密聚合梯度  
 return copy.deepcopy(update\_w\_avg), sum(self.clients\_loss) / len(self.clients\_loss)

Client:

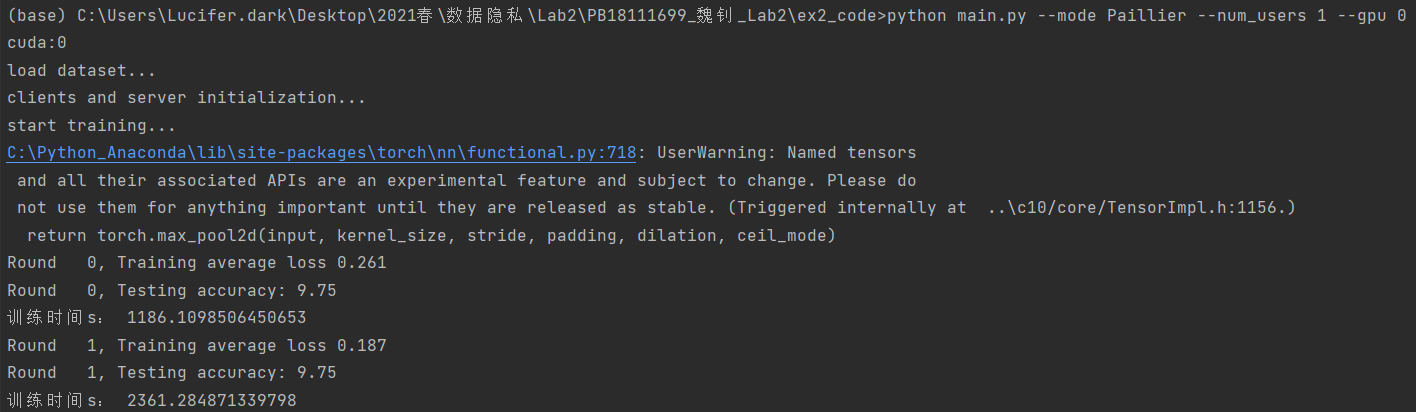
Train:

elif self.args.mode == 'Paillier':  
 for k in w\_new.keys():  
 update\_w[k] = (w\_new[k] - w\_old[k]).flatten(start\_dim=0).tolist() # 用w\_new-w\_old代替  
 for j in range(len(update\_w[k])):  
 update\_w[k][j] = KeyPub.encrypt(update\_w[k][j]) # 用公共密钥进行加密

update:

elif self.args.mode == 'Paillier':  
 for k in w\_glob.keys():  
 for i in range(len(w\_glob[k])):  
 w\_glob[k][i] = KeyPriv.decrypt(w\_glob[k][i]) # 用私钥进行解密  
 Final\_tensor = torch.Tensor(w\_glob[k]).to(self.args.device)  
 Final\_tensor = Final\_tensor.reshape(self.model.state\_dict()[k].shape) # 类型转换回来  
 self.model.state\_dict()[k] += Final\_tensor

训练结果：



一轮大概需要20分钟。