# 联邦学习项目——遥感图像分割

## 项目背景

随着人工智能的火热，越来越多基于人工智能的应用开始兴起，如人脸识别，目标识别，遥感图像分割等。但是基于深度学习的人工智能技术需要大量的数据来训练模型，这也是很多人工智能项目落地的第一个难点。

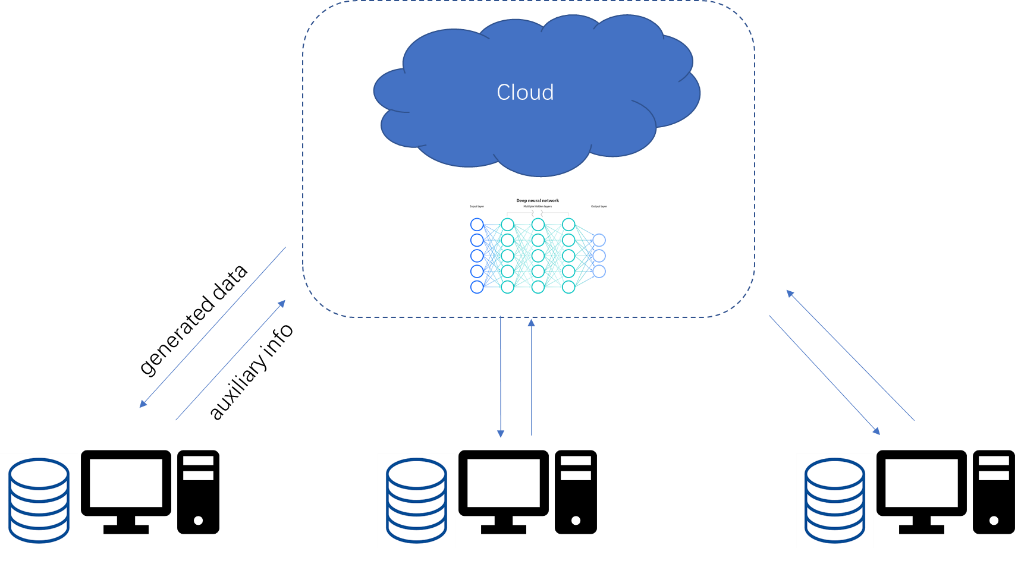
现实生活中，除了少数巨型科技公司可以收集足够的数据，绝大多数数据都是无法共通，形成一个个数据孤岛。同时国内外监管环境也在加强数据保护，陆续出台相关政策，如欧盟的《通用数据保护条例》法案，我国的由国家互联网信息办公室起草的《数据安全管理办法(征求意见稿)》。因此为了保护数据隐私，数据的随意流动会受到大量限制。

为了解决数据不足和数据隐私的人工智能训练问题，联邦学习应运而生。联邦学习本质上是一种分布式机器学习技术，以保证数据隐私安全及合法合规的基础上，实现共同建模，提升AI模型效果。

在遥感领域，语义实例分割可以说智能地图、智慧城市提出的新需求，对某一类的个体区分是建立更精细化模型的前提。尤其是随着高分辨率视频通讯小卫星的兴起，应用途径、场景大大扩宽，这也是一个令人期待的方向。与此同时，遥感数据领域的隐私保护问题尤其重要。

## 项目任务

由于在遥感领域中，真实的遥感图像是敏感数据而遥感图像的语义标注是脱敏数据，在本项目中训练AI遥感图像分割模型的过程中借助联邦学习来完成对遥感图像数据的保护。在本项目中包含多个边缘实体和一个中心实体。边缘实体负责提供遥感图像的标注数据和辅助中心实体的分割图像训练。中心实体负责训练出鲁棒的遥感图像分割模型。



因此本项目的具体任务为：

1. 保护遥感数据隐私。
2. 通用跨平台跨框架的链式梯度传导开发。
3. 多线程数据同步框架开发。
4. 训练遥感图像分割模型。
5. 鲁棒的分割效果（单个实体和多个实体的指标<0.02）。
6. 可视化的结果展示。

## 技术方案

### 模型定义

本项目使用GAN模型负责虚拟遥感数据生成来保护真实遥感数据隐私。分布式GAN网络模型，由一台中央生成器和位于不同实体中的多个分布式鉴别器组成。接下来，我们介绍网络体系结构、目标函数，然后分析分布式异步优化的过程。

分布式GAN架构的概述如图3所示。中央生成器（表示为G）采用特定任务的输入（在我们的实验中为分割掩码），并生成合成图像来欺骗鉴别器。本地鉴别器（从D1到Dn）分别分布在各个实体（各个中心/部门），学习区分本地真实图像和G的合成图像。由于图像的敏感性，可能无法从外部访问每个实体的真实图像。我们的体系结构能够避免这种限制，因为只有同一实体中的特定鉴别器才需要访问真实图像。这样，本地实体中的真实图像将被秘密保存。仅需要在中央生成器和每个实体之间传输合成图像、分割掩码和损失函数值。

生成器将从属于不同实体的不同数据集中学习联合分布。然后，它可以用作图像提供者来训练特定任务，因为我们希望合成图像与真实图像共享相同或相似的分布。在实验中，我们将分布式GAN框架应用于分割任务以说明其有效性。Deeplab v3+被用作分割模型，有关为分割任务设计的G和D的详细信息如下所述。

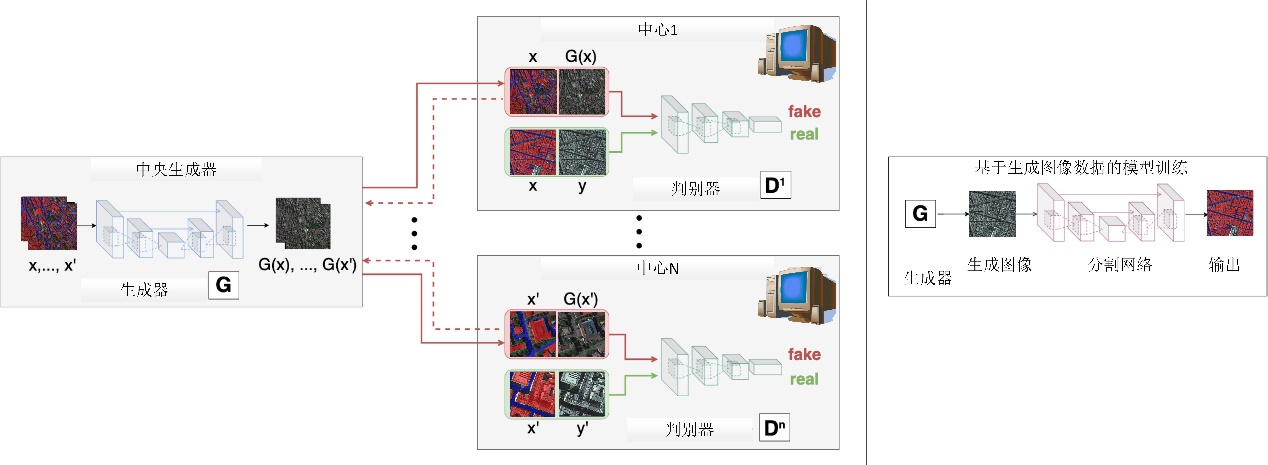


图3 分布式GAN网络架构设计。

#### 中央生成器

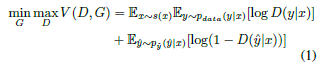
对于分割任务，中央生成器是一个编码器-解码器网络，它由两个卷积神经网络构成，一个卷积神经网络充当编码器，将模型输入的遥感分割标注转化成一个矢量表示。一个卷积神经网络充当解码器，其输入为矢量表示，输出为一张模型生成的遥感图像。

#### 分布式鉴别器

在分布式GAN框架中，鉴别器分布在N个节点（实体）上。每个鉴别器Dj仅可访问存储在第j个节点中的数据，因此以异步方式训练鉴别器。对于分割，每个鉴别器具有与PatchGAN中相同的结构。鉴别器分别判定图像中不同小块的真实性（伪造或真实）。这样的架构假设像素以Markov随机场的方式逐块独立，可以捕获几何结构中的差异（例如背景和前景目标）。

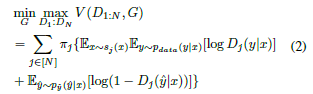
#### 分布式GAN的目标函数

分布式GAN是基于条件GAN。经典条件GAN的目标是：



其中D代表鉴别器，G代表生成器。 G旨在近似条件分布pdata(y|x)，以使D无法判断数据是否为“伪造”。隐藏变量x是控制生成数据模式的辅助变量。实际上，x通常是可以提供有关要生成的数据的信息的类标签或掩码。

在分布式GAN框架中，生成器由N个不同的鉴别器监督，每个鉴别器与数据集的子集相关联，使用辅助变量x上的混合分布来量化这种设置。换句话说，不是给定朴素的s(x)，而是x的分布变为C:\Users\dongdi\AppData\Local\Temp\1618029148(1).png。 对于每个子分布，都有一个对应的鉴别器Dj，它仅接收从sj(x)生成的数据。因此，我们的分布式GAN的损失函数变为：



#### 优化过程

分布式GAN的优化过程为迭代更新过程，在每次迭代中，都会向系统提供一个随机采样的元组(x; y)。在此，x表示由生成器观察到的输入标签，而y是仅实体可访问的真实图像。然后按以下顺序迭代更新网络块：

1）D-更新：计算第j个鉴别器Dj的对抗损失并更新Dj，j=1,2,…,N。

2) G更新：更新所有鉴别器后，使用对抗损失C:\Users\dongdi\AppData\Local\Temp\1618029032(1).png更新G。

其中，对抗损失采用了交叉熵损失。

### 框架搭建

基于分布式GAN网络的遥感图像分割系统主要包含三个部分——UI界面、训练框架、通讯服务。

#### UI界面

UI界面搭建用的主要框架为PySimpleGUI来保证UI框架的敏捷性开发和快速UI元素调整。在UI的设计上，包含模型可视化展示，模型训练流程控制，模型评测三个模块。在UI设计上同时融合了多线程通讯，使得不同中心的UI界面可以互相交换信息。首先程序会初始化UI界面。在UI程序退出之前，程序会循环检测来自其他中心的消息和事件，及时对这些时间进行映射和处理。

#### 训练框架

分布式GAN训练框架包括数据预处理、模型与参数定义、模型训练三个部分，以下分别介绍。

1. 数据预处理

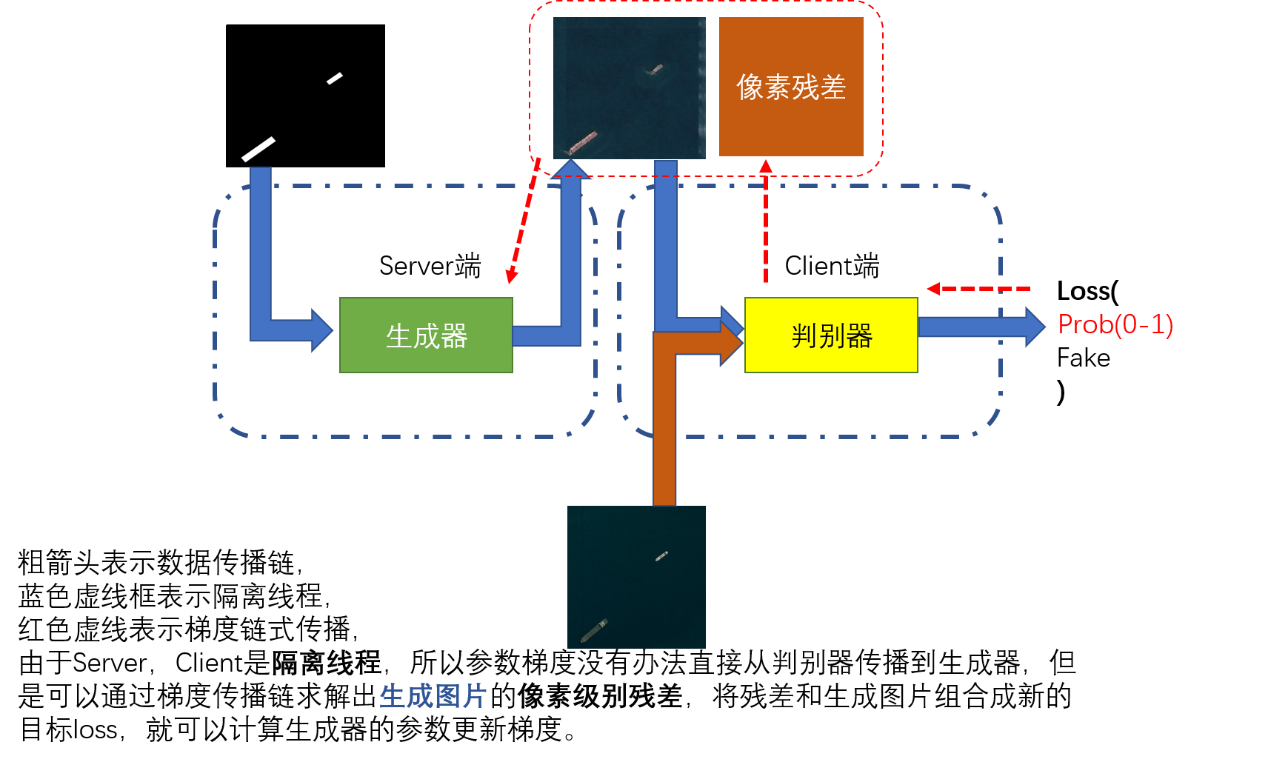
主要完成各节点端数据的预加载，预处理和通信的建立。首先建立各边缘节点对中心节点的多线程通道网络连接。各个边缘节点收集图像语义标注，同步到中心节点。中心节点收集各个边缘节点数据后准备开启训练。

1. 模型与参数定义

主要完成深度学习模型参数的选择和参数的设定。在中央节点端基于PyTorch编写深度神经网络模型，本项目采用的具体模型请参看下一小节的介绍。在建立深度神经网络模型后，初始化模型参数，同时为模型设置分布式训练参数，包括迭代轮数、批处理大小、学习率、损失函数及优化器等。

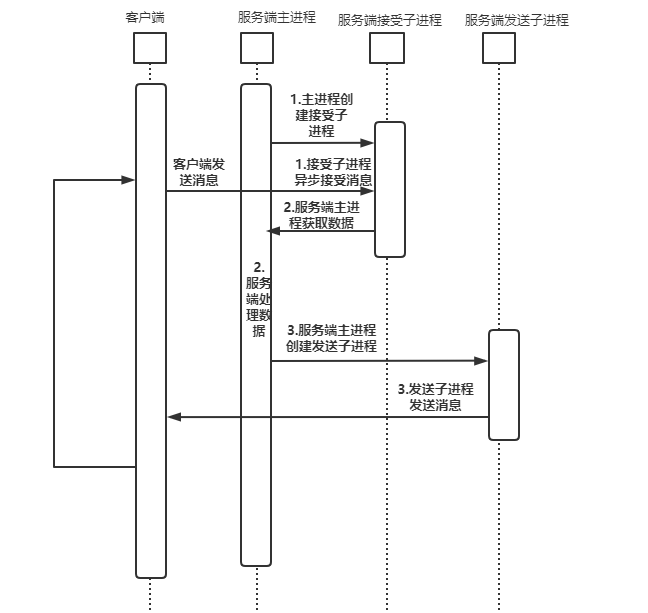
1. 模型训练

主要完成基于多中心通信的分布式训练和最终模型的生成。在WebSocket API中，服务器端和客户端只需要完成一次握手就可以创建持久性的连接，并进行双向参数传输。结合WebSocket通信协议，首先通过WebSocket通信协议建立中央节点端与分布式节点端的通信连接，获取存储训练数据的各节点端对应的id，ip及端口号。在分布式节点端和中心节点端，对已加载的训练数据进行协同训练，中央节点首先生成虚拟图片并传递给边缘节点，边缘节点并通过反向传播计算生成图像残差，随后，将残差通过WebSocket通信协议框架反馈给中央节点端，中央节点拿到残差后构造新的目标函数，进行梯度反传优化生成器。



通过这三个步骤，可以构建一个接口简单、可扩展性强的分布式深度学习架构。

#### 通信方案



采用socket通信和多进程技术实现异步通信。

1)客户端与服务端利用socket技术创建持久性连接 。

2)服务端采用多进程技术，实现异步通信：

①服务端主进程会创建多个接受子进程，接受子进程会重复接受来自客户端的数据,并将这些数据放入为每个客户端所创建的队列里面。

②当服务端接受子进程接受一轮数据后，服务端主进程开始依次从客户端队列里面获取数据，处理数据。

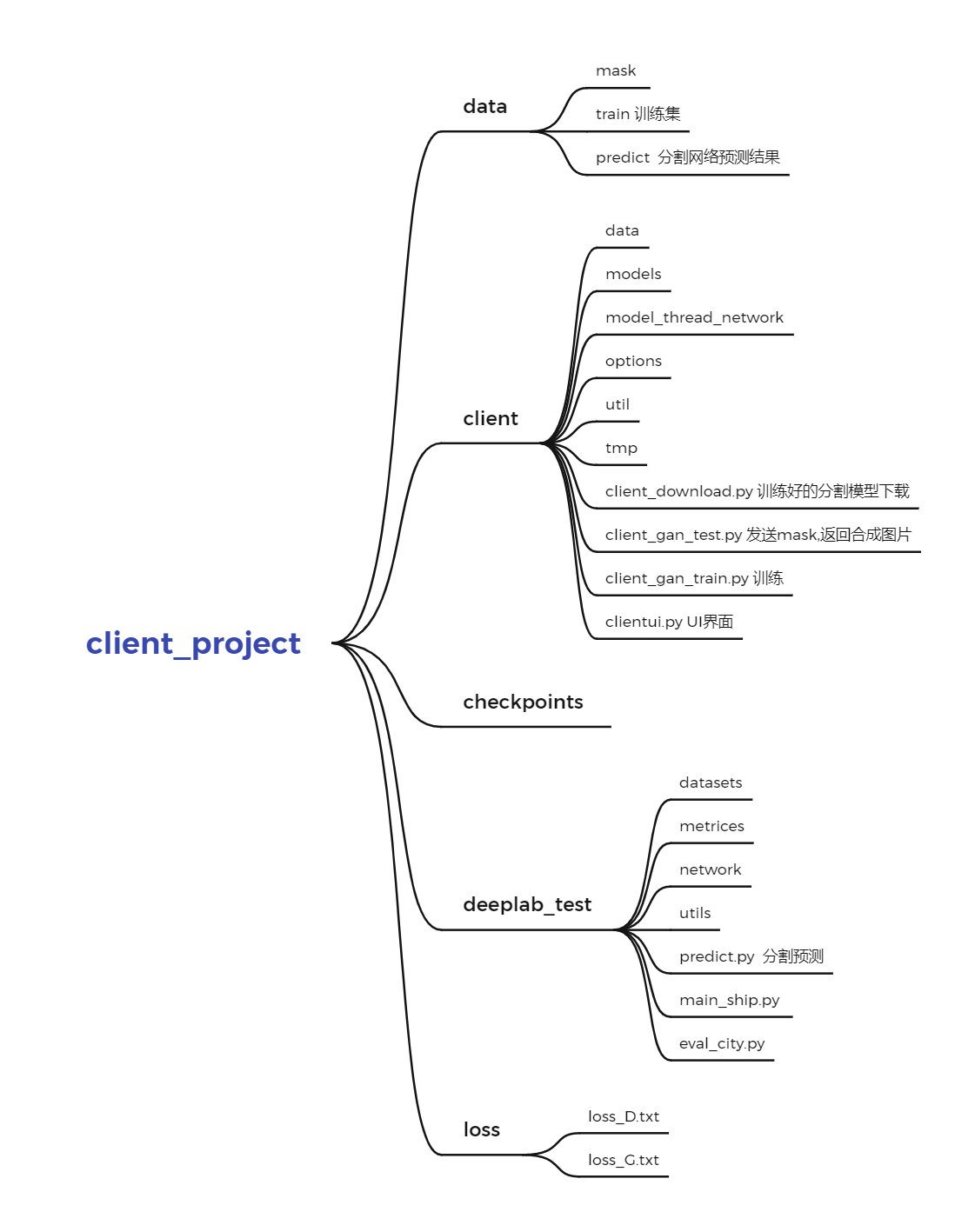
③服务端主进程创建发送子进程，通过发送子进程，将主进程处理的数据回传给客户端。

重复执行上面三个步骤。

## 使用文档

### 边缘节点

#### 项目架构



**\***按如上进行配置后，需要注意data/train文件夹下包括 mask（.png）和image(.jpg)

**data**/mask 中只存放mask

**client**/client\_download.py 模型下载脚本

client\_gan\_test.py asyndgan测试

cclient\_gan\_train.py asyndgan训练

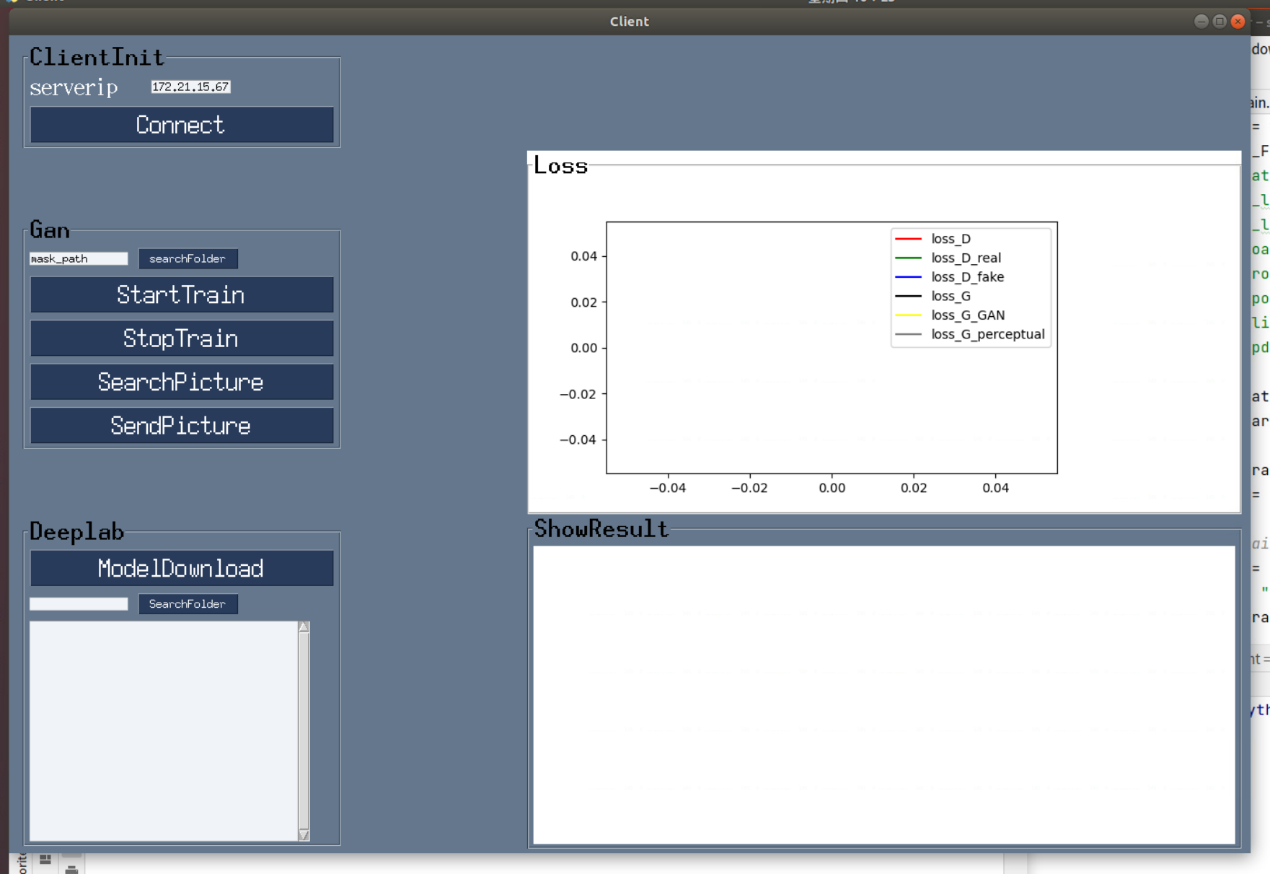
clientui.py UI界面

**deeplab\_test**/main\_ship.py deeplab训练脚本

predict.py deeplab测试脚本

eval\_city.py生成评测指标

#### 界面展示及说明



##### ClientInit

该模块完成边缘节点与中心节点SOCKET连接的建立。

|  |  |  |
| --- | --- | --- |
| ClientInit子模块说明 | | |
| 名称 | 说明 | 备注 |
| serverip | 中心节点IP地址 |  |
| connect | 连接中心节点按钮 |  |

##### Gan

该模块完成Gan网络的训练以及发送mask。

|  |  |  |
| --- | --- | --- |
| Gan子模块说明 | | |
| 名称 | 说明 | 备注 |
| searchfolder | 浏览mask图片所在路径 | 文件目录下为图片 |
| starttrain | 开始训练 |  |
| stoptrain |  |  |
| searchpicture | 浏览要发送的mask | searchpicture->sendpicture |
| sendpicture | 发送 | 点击此按钮发送图片 |

##### Deeplab

该模块完成分割权重下载，分割测试。

|  |  |  |
| --- | --- | --- |
| Deeplab子模块说明 | | |
| 名称 | 说明 | 备注 |
| modeldownload | 下载模型 |  |
| SearchFolder | 分割图片所在文件夹 | 文件夹下所有图片将展示在下面的界面中 |

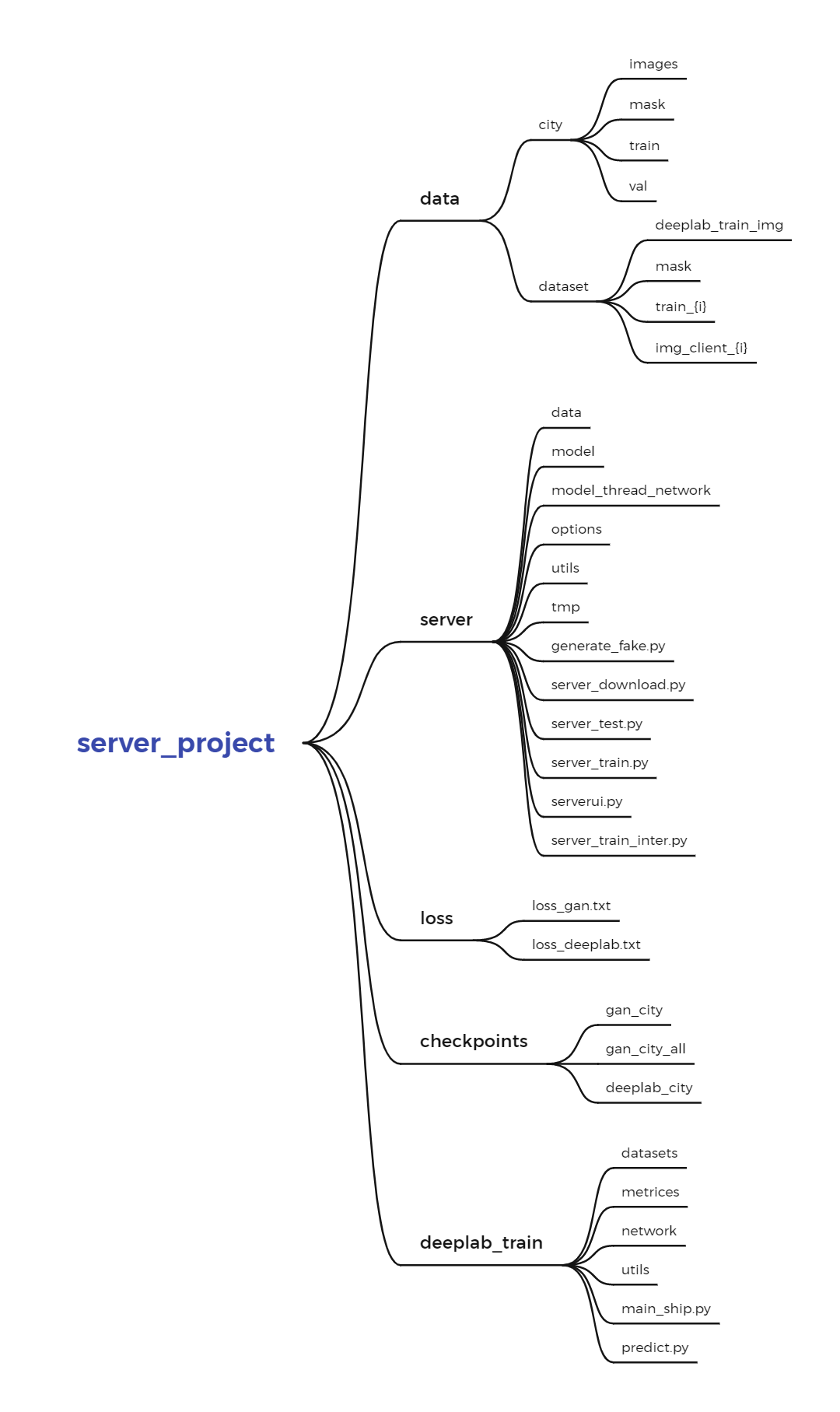
##### LOSS&ShowResult

展示训练过程中的loss；

展示合成图片的结果以及分割结果；

### 中心节点

#### 项目架构



**data/dataset**下 deeplab\_train\_img下为每个客户端上传的mask，当server\_test也就是生成加图片后，加图片也将存放在该目录下。

mask 上传的mask图片

train\_{i}：gan训练过程中每个鉴别器需要的mask

img\_client\_{i}:每个客户端上传的压缩文件

**checkpoints**/gan\_city：存放asydngan的权重

/deeplab\_city:存放使用加图片训练deeplab的权重

**server**/generate\_fake.py 客户端上传待检测mask返回生成加图片

server/server\_download.py 模型下载的脚本，将权重返回给客户端

server/server\_test.py 生成deeplab训练所需加图片的代码

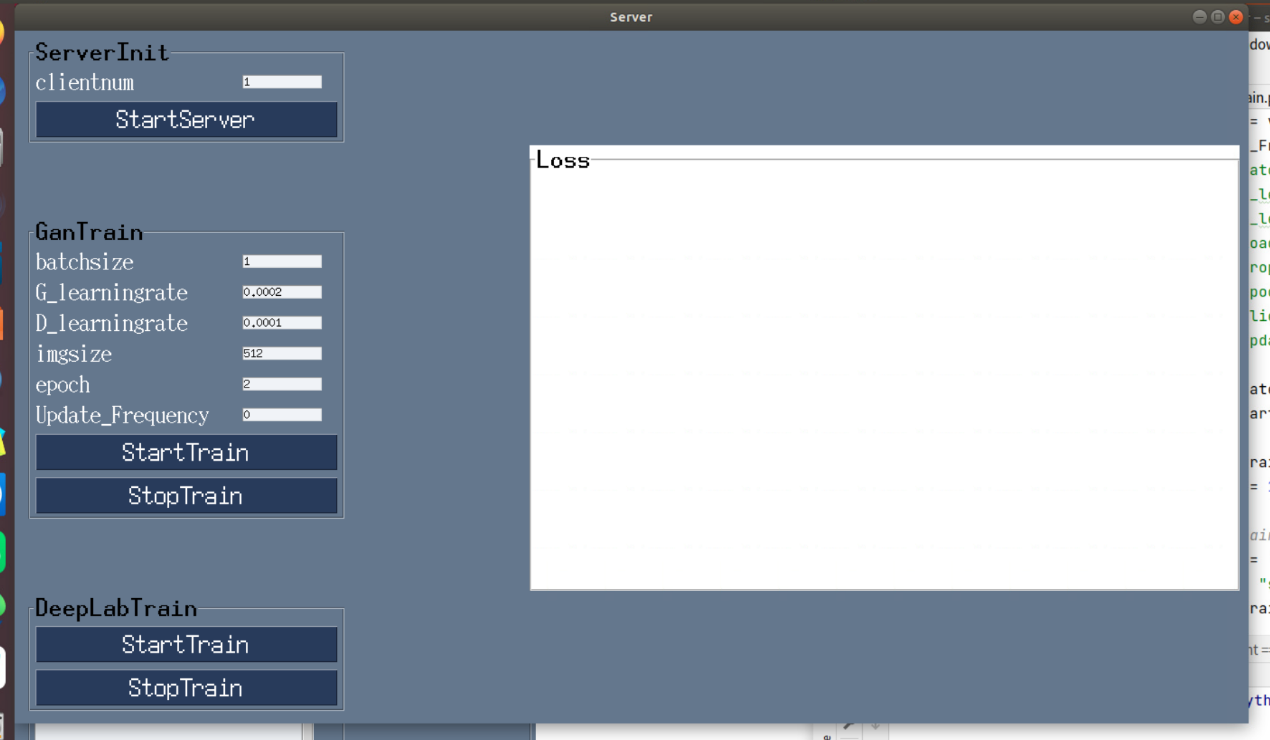
server/server\_train.py asyndgan训练脚本

server/server\_train\_inter.py asyndgan按照epoch更新训练脚本

server/serverui.py UI 界面

**deeplab\_train**/main\_ship.py deeplab训练脚本

#### 界面展示及说明



##### ServerInit

|  |  |  |
| --- | --- | --- |
| ServerInit子模块说明 | | |
| 名称 | 说明 | 备注 |
| clientnum | 边缘节点数量 |  |
| StartServe | 连接中心节点按钮 |  |

##### GanTrain

|  |  |  |
| --- | --- | --- |
| ServerInit子模块说明 | | |
| 名称 | 说明 | 备注 |
| batchsize | 批大小 |  |
| G\_learningrate | 中心节点学习率 |  |
| D\_learningrate | 边缘节点学习率 |  |
| imgsize | 图片大小 |  |
| epoch | 训练轮数 |  |
| Update\_Frequency | 更新方式 | 0 or >0“当客户端数据分布相差较大时，使用epoch为单位的频率更新模型参数” |
| StrainTrain | 训练 | 等所有客户端点击完开始 |
| StopTrain | 停止训练 |  |

##### DeepLabTrain

该模块完成分割网络训练。

|  |  |  |
| --- | --- | --- |
| DeepLabTriin子模块说明 | | |
| 名称 | 说明 | 备注 |
| StrainTrain | 训练 |  |
| StopTrain | 停止训练 |  |

##### Loss

展示Gan网络和分割网络训练的loss。

### 使用方法

#### Asyndgan训练

1. Server: (1)输入客户端数量，点击“startserver”

(2)设置训练的参数，点击“starttrain”

2. Client: (1) 输入服务端ip，点击“connect”

(2)点击“searchfolder”找到data/mask目录

(3)点击“starttrain”

3. Server: Loss窗口展示训练的loss

4. Client:Loss窗口展示loss

#### 中心节点训练Deeplab：

Server（1）点击Deeplab模块中“StartTrain”

(2) loss展示在Loss模块中

#### 边缘节点展示合成图片

Client：（1）点击“searchpicture”找到待生成加图片的mask，点击“sendpicture”发送假图片到服务端

（2）生成的假图片展示在“ShowResult”模块中

#### 边缘节点Deeplab模块

待服务端Deeplab训练完成

1. 点击“modeldownload”下载中心节点的权重
2. 点击“浏览”找到待分割图片的路径，随后图片文件名将展示在下面的空白中
3. 点击图片名称，分割结果将展示在ShowResult模块中

### 注意事项及拓展

#### 1.增加了提示弹窗，提示后自动关闭

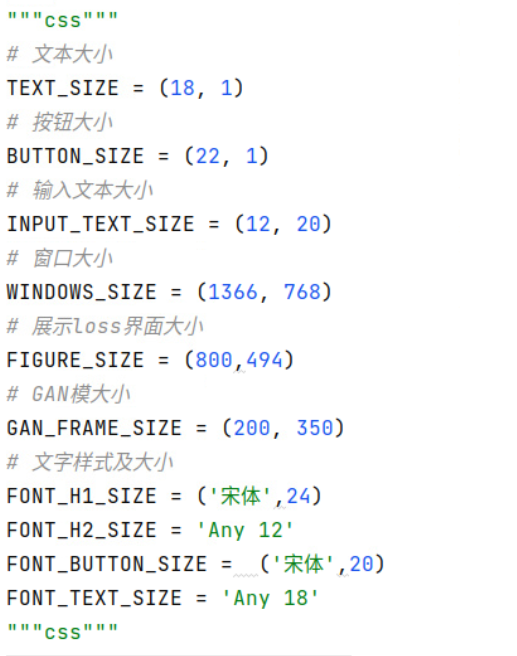
****

**可按照此代码拓展**

*import PySimpleGUI as sg*

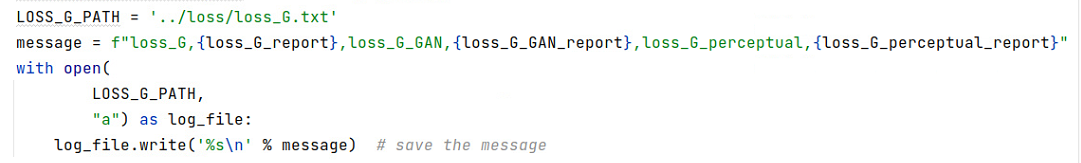
*sg.popup\_auto\_close("****需要提示的内容****", title='提示', button\_type=5, font='Any 18')*

#### 2.设置CSS样式全局变量，后续可直接在此添加，修改样式

****

#### 3.loss保存与可视化

（1）定义保存文件的路径，保存的message信息（“,”分割），写入文件。示例如下。

（2）每次在 client\_ui.py中会先清楚数据，然后在固定文件位置写入。

（3）解析（1）中保存的文件，绘制。



**后续可按照此步骤拓展。**

#### 4.当客户端数据分布相差较大时，使用epoch为单位的频率更新模型参数。

基本框架搭建完成，后续可直接修改，若需要增加提示语，loss展示，可根据1，3完成。

服务端文件位置

**…/Server/server/server\_train\_inter.py**

客户端文件位置

**…/Client/client/client\_gan\_train.py**

**if opt.Update\_Frequency > 0:**

按照epoch更新

**else:**

第一版方式更新