# 网络封装接口说明

## 目录

```
- your codes # 一个网络的所需文件
  |- ...
- model.py # 一个包含Trainer和Infer的文件，下面有详细说明

例如：
- mask_rcnn # mask_rcnn跑通的源文件
  |- backbone
  |- weights
  |- train_utils
  |- train.py
  |- infer.py
  |- ...
- mask_rcnn.py # Trainer·Infer文件
```

## 模型类

```
class Model(nn.Module):
  def __init__(self, ...):

    # 在构造函数中添加下面几行代码，无需改动
    self.train_loss = []
    self.train_metrics = []
    self.val_loss = []
    self.val_metrics = []

  # 在模型类中添加以下四个函数，无需改动
  def get_train_loss(self):
    return self.train_loss

  def get_train_metrics(self):
    return self.train_metrics

  def get_val_loss(self):
    return self.val_loss

  def get_val_metrics(self):
    return self.val_metrics
```

## Dataset类

```
class Model_Dataset:
  def __init__(self,
               data_dir, # 数据集文件夹
               train, # 该项为True时创建训练集dataset，为False时创建验证集Dataset
               ...) # 其他需要的参数，请给定默认值，如有没有默认值需要在训练时赋值的参数请写明注释

  def __getitem__(self, idx):
    # ...

  def __len__(self):
    # ...
```

## model.py文件

包含Trainer类和Infer类，请按照改方式封装。

### Trainer类

```
class Model_Trainer:

  def __init__(self,
               out_path,  # 参数保存路径，包含路径名和文件名，不包含后缀名，请使用.pth.tar作为后缀名
               train_dataloader,  # 训练集的dataloader
               val_dataloader,  # 验证集的dataloader
               model,  # 模型实例，在trainer中无需重新定义模型
               model_params,  # 一些模型参数，后面详细介绍
               q,  # 用于控制线程的，不用管这个
               pretrainModel,  # 预训练模型路径
               param_save_type, # 模型保存的方式，有best和last两种，分别表示保存最优参数和最后一轮参数
               log_path, # 日志文件创建路径
               ...) # 其他需要的参数，请给定默认值

    self.out_path = out_path
    self.train_dataloader = train_dataloader
    self.val_dataloader = val_dataloader
    self.model = model
    self.q = q
    self.pretrainModel = pretrainModel
    self.para_save_type = param_save_type
    self.log_path = log_path
    # epoch的获取，位于model_params中，2为默认值（请修改为实际默认值）
    self.epoch = 2 if 'epoch' not in model_params.keys() or model_params['epoch'] == '' else int(model_params['epoch'])
    # lr的获取，位于model_params中，0.001为默认值（请修改为实际默认值）
    self.lr = 0.001 if 'lr' not in model_params.keys() or model_params['lr'] == '' else float(model_params['lr'])
    # loss的获取，位于model_params中，MSELoss为默认值（请修改为实际默认值），并以注释的形式在此处列出所有可选loss选项，例如：MSELoss,DICELoss
    self.criterion = 'MSELoss' if 'loss' not in model_params.keys() or model_params['loss'] == '' else model_params['loss']
    # optimizer的获取，位于model_params中，Adam为默认值（请修改为实际默认值），并以注释的形式在此处列出所有可选优化器的选项，例如：MSELoss,DICELoss
    self.optimizer = 'Adam' if 'optimizer' not in model_params.keys() or model_params['optimizer'] == '' else model_params['optimizer']
    # batch的获取，位于model_params中，8为默认值（请修改为实际默认值）
    self.batch_size = 8 if "batch" not in model_params.keys() or model_params['batch'] == '' else int(model_params["batch"])

  def train():
    # 训练代码

    # train开头添加如下代码创建日志文件
    now =  time.localtime()
    now_time = time.strftime("%Y-%m-%d %H:%M:%S", now)
    logging.basicConfig(filename=self.log_path+"/log_"+now_time+".txt", level=logging.INFO,
                        format='[%(asctime)s.%(msecs)03d] %(message)s', datefmt='%H:%M:%S')
    logging.getLogger().addHandler(logging.StreamHandler(sys.stdout))

    # 预训练模型导入部分代码参考
    if pretrainModel is not None:
      model.load_state_dict(torch.load(pretrainModel))


    for i in range(self.epoch):
      # 针对每一个epoch添加如下代码
      self.model.train_loss.append(train_loss)
      self.model.train_metrics.append(train_metrics)
      self.model.val_loss.append(val_loss)
      self.model.val_metrics.append(val_metrics)
      # 日志输出参考，可根据不同需要修改输出字段
      logging.info("epoch: {}, loss: {}, metrics: {}".format(i, train_loss, train_metrics))
      self.q.put([self.model.train_loss, self.model.train_metrics, self.model.val_loss, self.model.val_metrics])

    # 参数保存部分代码参考
    if param_save_type == "best" and train_loss < lowest_train_loss:
      lowest_train_loss = train_loss
      _path = out_path + ".pth.tar"
      torch.save(model, _path)
    if param_save_type == "last":
      _path = out_path + ".pth.tar"
      torch.save(model, _path)

    # 请针对构造函数中的参数修改训练代码
```

## Infer类

```
class Model_Infer:
  def __init__(self,
               input_path, # 输入文件的路径，路径包括文件名，可直接使用
               model_path, # 模型参数路径，包含后缀名，可直接使用
               save_path, # 结果保存路径，包含文件名，可直接使用
               ...): # 其他参数，请给出默认值
  # 若有两个图像输入，则改为__init__(self, input_path_1, input_path_2, model_path, save_path)
  # 以此类推，若有其他情况，请跟我讨论

  def infer(self):
    # 请将结果保存到指定路径
```