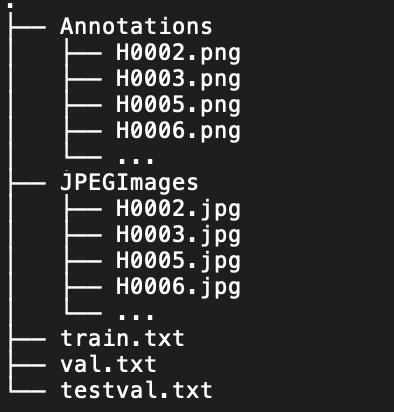
# AutoCV模型使用说明书

### 1.1依赖

本系统前端页面只在Chrome浏览器上进行调试，为保证正常使用，请使用Chrome浏览器。

### 1.2数据格式要求

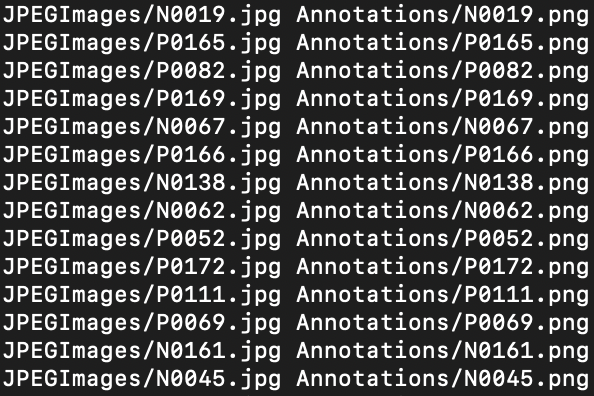
目前模型指定的数据组织格式只有一种，数据集根目录下必须包含以下文件和文件夹:**train.txt、val.txt、testval.txt、原始图片数据保存文件夹**（名字可自由定义，如 JPEGImages），**标签图片数据保存文件夹**（名字可自由定义，如Annotations）。下面以开源数据集optic的例子介绍每个文件或文件夹的作用，下图是该数据集根目录下的文件组成:



原始图片数据保存文件夹: 即上图中的JPEGImages文件夹，该文件夹中保存所有训练、验证和测试的原始图片数据，支持的图片格式总共有4种：jpg、png、tif、jpge；

标签图片数据保存文件夹：即上图中Annotations文件夹，其中每一张标签名字需要与对应的图片名字一致，支持的图片格式同上。标签图片应与对应的原始图片尺寸一致，每一个像素值为原始图片对应像素的类别。

train.txt：保存所有训练图片和标签相对当前数据集根目录的相对路径，一条训练数据一行，图片路径和标签路径用空格隔开。下面是optic中train.txt内容部分截图:



val.txt和testval.txt : 除了train.txt，还需要添加val.txt和testval.txt两个文件，分别保存验证数据和测试数据的相对路径，其内容格式和train.txt一致。**另外，为了使模型训练正常进行，训练数据集中每个类别的样本数至少为20张图片。**

### 训练使用方法

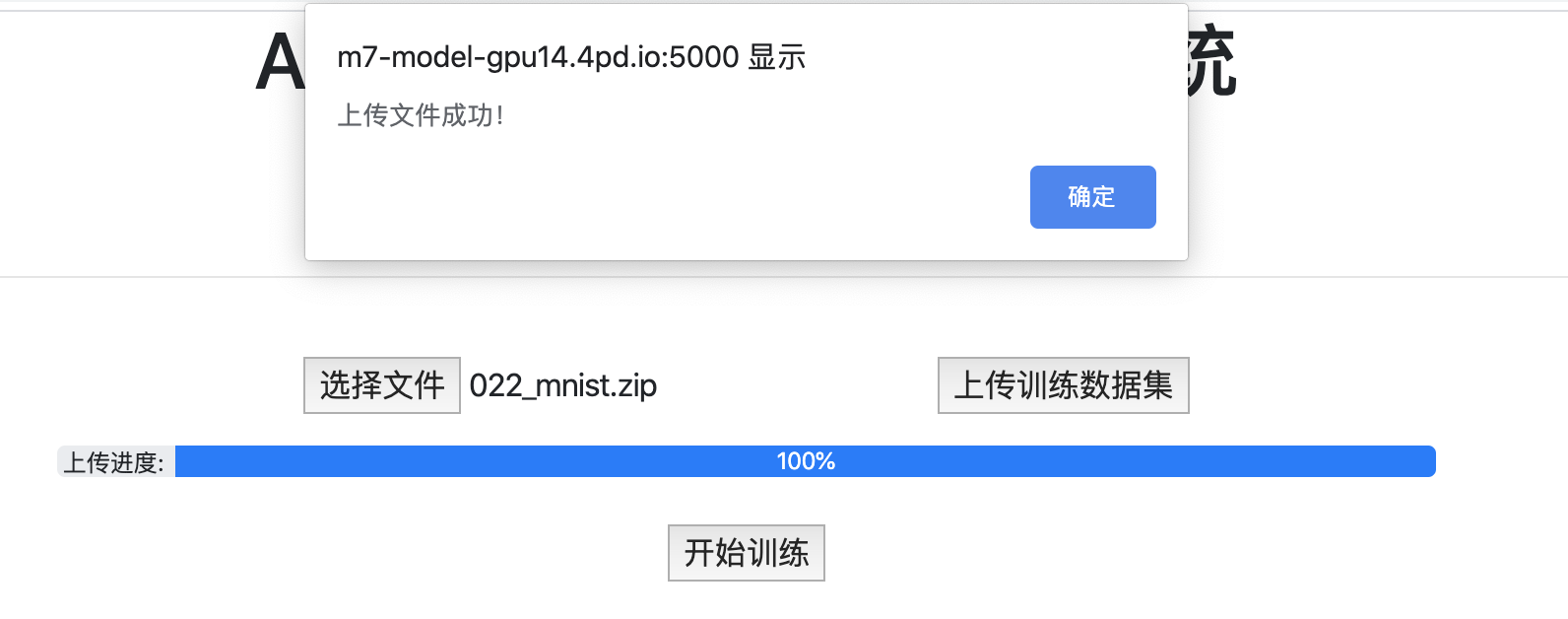
#### 训练

登录系统首页，系统的训练模块截图如下：

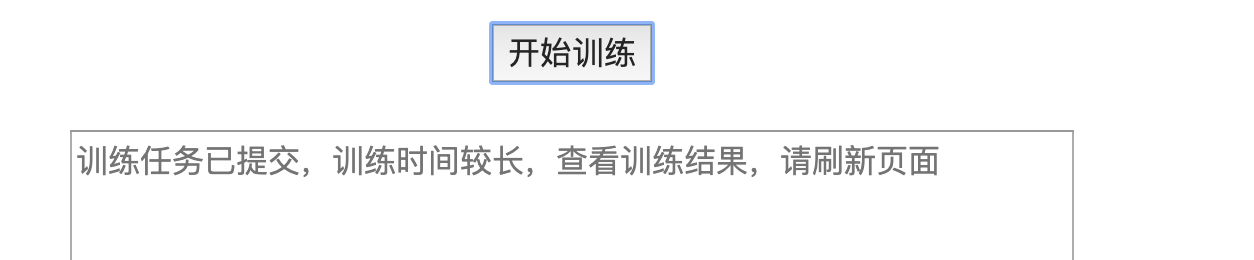


训练步骤如下：

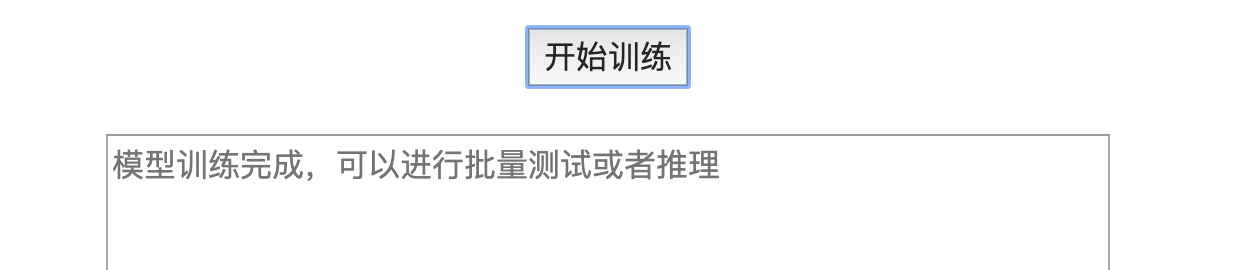
1. 点击“选择文件“，选择要训练的本地训练数据集zip包，该文件包格式必须满足上述的训练数据集格式；
2. 点击“上传训练数据集“开始上传数据，由于训练数据较大，上传需要等待较长时间，下面的上传进度条会指示文件上传进度，如下图所示，上传期间请勿刷新页面，否则上传过程会取消；



1. 文件上传成功后浏览器会弹出文件上传成功弹窗（如上图），点击“开始训练“提交训练任务，任务提交成功后，下方文本框会提示“正在训练”，如下图所示，由于训练时间较长，可以定期刷新页面查看是否训练完成；



1. 训练成功后，文本框会给出如下提示：



注意：系统只支持同一时间保存一个模型，多次提交训练，系统会删除之前的训练模型。另外，本系统同一时间只支持一个用户使用。

#### 批量测试

批量测试的文件上传与开始的操作方法与训练的类似，此处不再赘述，测试提交后需要等待结果测试返回，测试请勿刷新页面，否则浏览器会取消测试请求，从而无法收到测试结果，网页功能如下图所示。



后台会根据模型预测的结果与数据实际的标签计算多个评价指标，用户可以根据自己业务的要求参考响应的指标，具体效果如下图例所示。

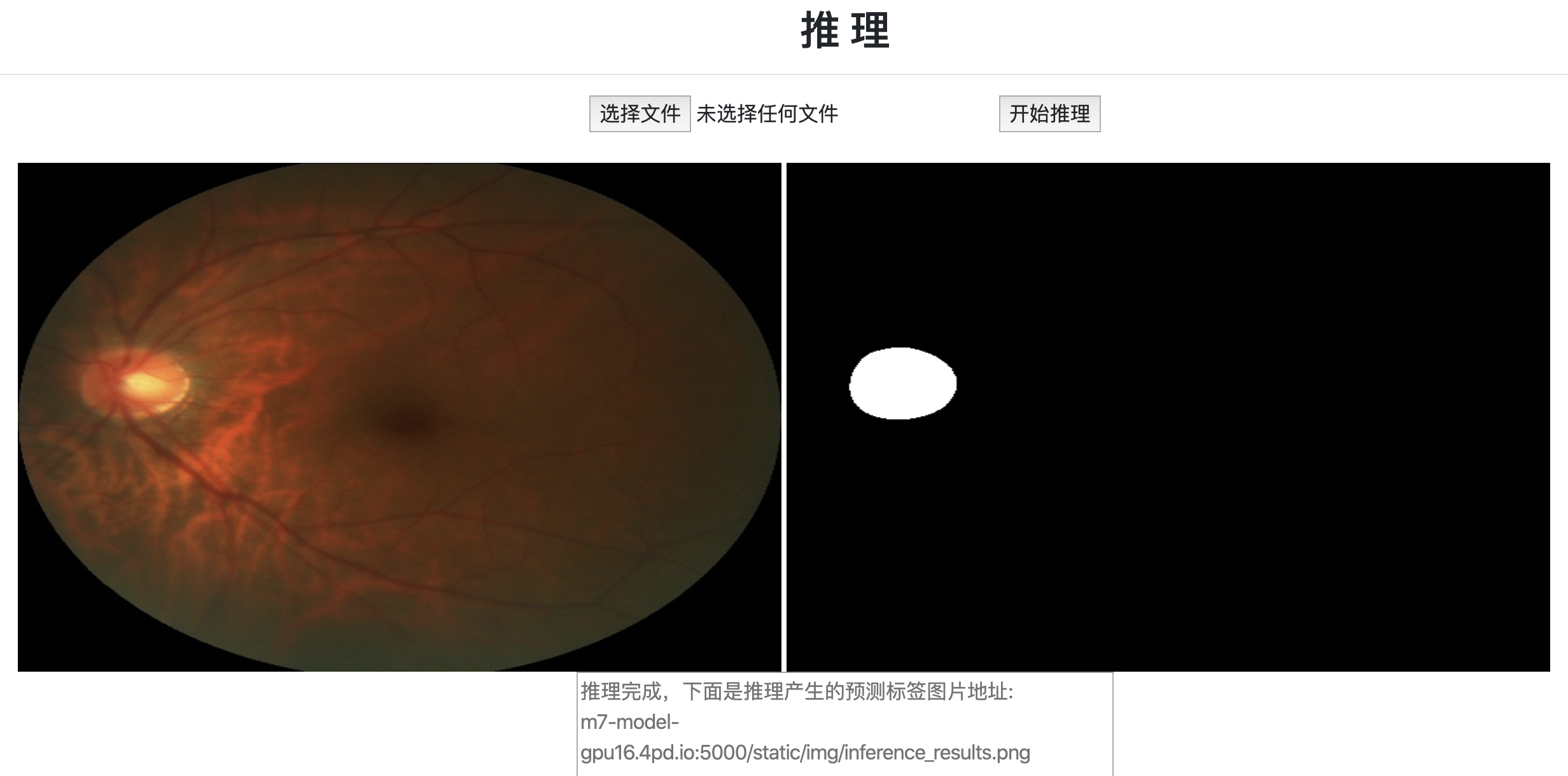


#### 单个图片推理

训练成功后，就可以通过页面中的推理功能上传单个图片文件进行推理预测，如下截图：



图片上传成功后，模型自动开始推理计算，推理成功后后台返回并显示如下信息在下方文本框中，如下图所示：



推理结果说明：

左侧图为原图，右侧为模型预测的结果标签图像，可通过图片下方的链接下载预测结果图片。

#### 视频文件推理

本系统支持上传视频文件进行推理，目前仅支持“AVI”格式视频文件，视频文件上传成功后，点击“开始推理”按钮后，系统开始进行后台推理计算，计算结束后会在下面文本框中显示推理结果视频文件URL链接，将该链接复制到浏览器输入框中便可以下载，如下图所示：



### 1.4 第三方开发接口说明

#### 脚本提交单个文件推理

可以使用如下Python脚本（Python3）所示的方式循环上传单张文件进行测试，通过脚本上传单张文件返回的内容与上述“单个图片推理“小节的内容相同，不过脚本返回的是JSON格式字符串，此处不再赘述。

|  |
| --- |
| import argparse  import requests  def segment(img, ip = '0.0.0.0', port = 5001):  req\_url = 'http://{}:{}/segment/inference'.format(ip, port)  # 发起推理请求  with open(img, 'rb') as f:  r = requests.post(req\_url, files = {'file\_inference' : f})  # 下载推理标签图片文件  results = json.loads(r.text)  download\_url = 'http://{}:{}/{}'.format(ip, port, results['predict'])  r = requests.get(download\_url)  save\_file\_name = img.split('/')[-1].split('.')[0] + '\_predict.png'  with open(save\_file\_name, 'wb') as f:  f.write(r.content)  if \_\_name\_\_ == '\_\_main\_\_':  parser = argparse.ArgumentParser(description="远程访问模型进行图片预测工具")  parser.add\_argument('--image', help="图片的本地地址，必须提供", required=True)  parser.add\_argument('--ip', help="服务IP，默认 0.0.0.0", type = str, default = '0.0.0.0', required = False)  parser.add\_argument('--port', help="服务端口号，默认 5001", type = int, default = 5001, required = False)  args, \_ = parser.parse\_known\_args()  predict = segment (args.image, args.ip, args.port)  print(predict) |

#### 脚本方式提交训练任务

|  |
| --- |
| import argparse  import requests  import json  import time  def train(dataset\_path, ip = '0.0.0.0', port = 5001):  # 先上传数据，如果数据较大上传时间可能会很长  upload\_url = 'http://{}:{}/segment/upload/train'.format(ip, port)  with open(dataset\_path, 'rb') as f:  r = requests.post(upload\_url, files={'file\_train': f})  print(r.text)  format\_ret = json.loads(r.text)  # 如果接口有返回，但是返回内容不是success的话，说明上传失败，失败的字段有dataset invalid、zip file error和error  # 其中dataset invalid和zip file error表示数据格式存在错误，error表示其他错误  if format\_ret['return'] != 'success':  return  print('upload dataset success')  # 发起训练  train\_url = 'http://{}:{}/segment/train'.format(ip, port)  # 一般训练时间很长，这个接口返回可能需要很久，数据量大的话可能需要几天，此处建议设置timeout,  # 超时后，通过轮询请求目前系统的状态  try:  r = requests.post(train\_url, timeout = 30)  format\_ret = json.loads(r.text)  if format\_ret['return'] == 'no data':  print('no data')  return  except requests.exceptions.Timeout:  print('request timeout')  # 超时轮询检查是否训练完成，后台返回的状态类型有training、trained、non-trained、testing和tested，  # training: 表示正在训练中，训练中不能发送批量测试、推理和再次训练  # trained: 表示训练完成，训练完成后才可以发送批量测试、推理  # non-trained: 表示没有训练，后台正在闲置，没有训练不能发送批量测试、推理，只能发送训练请求  # testing: 表示正在批量测试中，批量测试中不能发送再次批量测试、推理和再次训练  # tested: 表示正在批量测试完成，这个状态是瞬间状态，只在调用批量测试接口后返回  status\_url = 'http://{}:{}/segment/get\_status'.format(ip, port)  while True:  r = requests.get(status\_url)  print(r)  format\_ret = json.loads(r.text)  if format\_ret['return'] == 'trained':  print('训练完成')  break  elif format\_ret['return'] == 'non-trained':  print('训练失败')  break  elif format\_ret['return'] == 'training':  print('训练中...')  # 由于训练时间较长，建议不要频繁请求状态  time.sleep(60)  if \_\_name\_\_ == '\_\_main\_\_':  parser = argparse.ArgumentParser(description="远程访问模型进行图片预测工具")  sub\_parser = parser.add\_subparsers(help='选择不同功能')  train\_parser = sub\_parser.add\_parser('train', help='训练')  train\_parser.add\_argument('--subcommand', default='train')  train\_parser.add\_argument('--file', help="图片数据集zip格式压缩包路径，必须提供", required=True)  train\_parser.add\_argument('--ip', help="服务IP，默认 0.0.0.0", type = str, default = '0.0.0.0', required = False)  train\_parser.add\_argument('--port', help="服务端口号，默认 5001", type = int, default = 5001, required = False)  args, \_ = parser.parse\_known\_args()  if args.subcommand == 'train':  train(args.file, args.ip, args.port) |

#### 脚本方式提交批量测试任务

|  |
| --- |
| import argparse  import requests  import json  import time  def test(dataset\_path, ip = '0.0.0.0', port = 5001):  # 先上传数据，如果数据较大上传时间可能会很长  upload\_url = 'http://{}:{}/segment/upload/test'.format(ip, port)  with open(dataset\_path, 'rb') as f:  r = requests.post(upload\_url, files={'file\_test': f})  print(r.text)  format\_ret = json.loads(r.text)  # 如果接口有返回，但是返回内容不是success的话，说明上传失败，失败的字段有dataset invalid、zip file error和error  # 其中dataset invalid和zip file error表示数据格式存在错误，error表示其他错误  if format\_ret['return'] != 'success':  return  print('upload dataset success')  # 发起测试  test\_url = 'http://{}:{}/segment/test'.format(ip, port)  # 一般测试时间略长  r = requests.post(test\_url)  format\_ret = json.loads(r.text)  if format\_ret['return'] == 'no data':  print('no data')  return  # records字段包含测试结果  records = format\_ret['records']  print(records)  if \_\_name\_\_ == '\_\_main\_\_':  parser = argparse.ArgumentParser(description="远程访问模型进行图片预测工具")  sub\_parser = parser.add\_subparsers(help='选择不同功能')  test\_parser = sub\_parser.add\_parser('test', help='测试')  test\_parser.add\_argument('--subcommand', default='test')  test\_parser.add\_argument('--file', help="图片数据集zip格式压缩包路径，必须提供", required=True)  test\_parser.add\_argument('--ip', help="服务IP，默认 0.0.0.0", type = str, default = '0.0.0.0', required = False)  test\_parser.add\_argument('--port', help="服务端口号，默认 5001", type = int, default = 5001, required = False)  args, \_ = parser.parse\_known\_args()  if args.subcommand == 'test':  test(args.file, args.ip, args.port) |