接口设计文档

前后端交互接口

> 接口状态说明：
>
> 接口未完成/接口测试出错
>
> 接口已完成但未上线测试
>
> 接口已完成且测试通过
>
> 服务器公网IP：139.196.90.131
>
> 服务端口：8000

> 阿里云OSS 获取文件：向后端给定 uri 地址发送不加任何参数的 GET 请求
>
> 阿里云OSS 简单下载文件参考文档：简单下载
>
> 如果前端做缓存处理可在该阿里云接口文档下参考获取 文件hash、ETag等 信息
>
> 其中权限设为全体可读，不需要身份认证直接请求即可

OSS文件路径：

业务码及其含义

```
// 系统相关
```

```go
var (
    Success      = NewError( code: 0,  msg: "成功")
    ServerError  = NewError( code: 1000,  msg: "服务内部错误")
    ParamError   = NewError( code: 1001,  msg: "传入参数错误")
)

// 用户及鉴权相关
var (
    UnauthorizedTokenNull    = NewError( code: 2000,  msg: "认证信息有误")  //鉴权失败，token 为空
    UnauthorizedTokenError   = NewError( code: 2001,  msg: "认证信息有误")  //鉴权失败，token 错误
    UnauthorizedUserNotFound = NewError( code: 2002,  msg: "认证信息有误")  //鉴权失败，用户不存在
    PasswordNull             = NewError( code: 2003,  msg: "密码不能为空")  //传入密码为空
    AccountExist             = NewError( code: 2004,  msg: "帐号已存在")   //账号已存在
    AccountNotFound          = NewError( code: 2005,  msg: "帐号或密码错误") //用户不存在
    PasswordError            = NewError( code: 2006,  msg: "帐号或密码错误") //密码错误
    UploadAvatarError        = NewError( code: 2007,  msg: "上传头像失败")  //上传头像失败
)

// 项目相关
var (
    AccountProjectError = NewError( code: 3000,  msg: "服务器内部错误") //用户和项目不匹配
    UploadFileError     = NewError( code: 3001,  msg: "服务器内部错误") //上传图片发生错误
    GetFolderError      = NewError( code: 3002,  msg: "服务器内部错误") //获取文件目录发生错误
    DeleteFileError     = NewError( code: 3003,  msg: "服务器内部错误") //删除文件发生错误
)
```

用户管理接口

> 前后端使用 token 对用户做鉴权处理
>
> 需要含 token 的接口中：将 token 放在 http header 的 Authorization 中

用户注册

- **HTTP 方法**

  [POST]

- **PATH**

  api/v1/user

- **Request**

JSON

```json
{
    "account": "user1",    //string，注册帐号/名称
    "password": "123456789",    //string，用户密码
}
```

> 前端对用户输入做初步判断（判空、长度限制）
>
> password 加密（之后再说）

- **Response**

JSON

```
{
    "code": 0,    //int，状态码
    "msg": "成功",    //string，返回信息
    "data": {}
}
```

获取用户信息

- **HTTP Method**

  [GET]

- **PATH**

  /api/v1/user

- **Request**

  含 token

- **Response**

JSON

```
{
    "code": 0,    //int，状态码
    "msg": "成功",    //string，返回信息
    "data": {
        "name": "qwer",    //string，用户名
        "avatarURL": "https://remotesensing.oss-cn-beijing.aliyuncs.com/avatar/example.jpg"    //string，用户头像存储路径
    }
}
```

用户登录

- **HTTP Method**

  [POST]

- **PATH**

  /api/v1/session

- **Request**

JSON

```
{
    "account": "123456789@gmail.com",    //string，注册邮箱
    "password": "123456789"    //string，用户密码（加密传输）
}
```

- **Response**

JSON

```
{
    "code": 0,    //int，状态码
    "msg": "注册成功",    //string，返回信息
    "data": {
        "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE2MjAxMTIyMzIsIm9wZW5JRCI6IjEyMzM0NTM0NSJ9.U5bTxP6VJcIYKVolaYKobOm5oEn_-nydr01aHWz72cI",    //string，token
    }
}
```

修改用户头像

- **HTTP Method**

[PUT]

- **PATH**

/api/v1/user/avatar

- **Request**

含 token

发送 form-data 表单，form-data 表单的结构如下：

avatar	图片文件

- **Response**

JSON

```
{
    "code": 0,    //int，状态码
    "msg": "头像设置成功",    //string，返回信息
    "data": {
        "avatarURL": "https://remotesensing.oss-cn-beijing.aliyuncs.com/avatar/example.jpg"    //string，用户头像存储路径
    }
}
```

项目管理接口

创建项目

- **HTTP Method**

  [POST]

- **PATH**

  /api/v1/project

- **Request**

  含 token

JSON

```
{
    "name": "project1"    //string，项目名称
}
```

- **Response**

JSON

```
{
    "code": 0,    //int，状态码
    "msg": "项目创建成功",    //string，返回信息
    "data": {
        "projectID": 1    //int，项目id，后续获取项目信息的凭证
    }
}
```

上传项目封面（低优先级）

TODO

根据搜索关键字获取项目

- **HTTP Method**

  [GET]

- **PATH**

  /api/v1/project?keyword=name

- **Request**

  含 token

- **Response**

JSON

```
{
    "code": 0,    //int，状态码
    "msg": "项目获取成功",    //string，返回信息
    "data": {
        "projects": [
            {
                "id": 22,
                "name": "name1",
                "lastVisit": "2022-05-11 12:00:00",    //string
                "coverURL": "xxx"
            },
            {
                "id": 12,
                "name": "name2",
                "lastVisit": "2022-05-11 12:00:00",    //string
                "coverURL": "xxx"
            },
        ]    //int，项目信息，按照匹配度顺序返回
    }
}
```

> 匹配模糊程度：项目名称完全含有该关键字
>
> 如：keyword: test，则 **test**/**test**1/1**test**/sdgagdf**test**asdfdsga 可以匹配，而 tesat/tes 无法匹配

获取最近项目

- **HTTP Method**

  [GET]

- **PATH**

/api/v1/project

- **Request**

  含 token

- **Response**

JSON

```
{
    "code": 0,    //int，状态码
    "msg": "项目获取成功",    //string，返回信息
    "data": {
        "projects": [
            {
                "id": 22,
                "name": "name1",
                "lastVisit": "2022-05-11 12:00:00",    //string
                "coverURL": "xxx"
            },
            {
                "id": 12,
                "name": "name2",
                "lastVisit": "2022-05-11 12:00:00",    //string
                "coverURL": "xxx"
            },
        ]    //int，项目信息，按照最近访问顺序返回
    }
}
```

在项目中上传图片（待优化）

> 图片名称保证在一个项目中无重名图片
>
> 在上传图片和修改图片名称时，若名称与项目中已有名称冲突，则会在名称后自动加上**"-数字"**，并且返回体中的 code = 3005，3005 仅仅作为警告处理，不作错误处理，即图片 上传/更名 请求成功但系统自动避免重名做加后缀处理。

- **HTTP Method**

  [POST]

- **PATH**

  /api/v1/project/picture

- **Request**

  含 token

发送 form-data 表单，form-data 表单的结构如下：

projectID	项目 id
imgNum	此次上传图片数量
img1	图片文件1
uuid1	图片文件1 uuid
name1	图片文件1 name
img2	图片文件2
uuid2	图片文件2 uuid
name2	图片文件2 name
...	根据 imgNum 确定接收图片数量

> 前端生成图片文件对应的 uuid，此 uuid 为图片文件唯一标识符

- **Response**

JSON

```
{
    "code": 0,    //int, 状态码
    "msg": "图片上传成功",    //string, 返回信息
    "data": {}
}
```

修改已上传图片名称

> 图片名称保证在一个项目中无重名图片
>
> 在上传图片和修改图片名称时，若名称与项目中已有名称冲突，则会在名称后自动加上**"-数字"**，并且返回体中的 code = 3005，3005 仅仅作为警告处理，不作错误处理，即图片 上传/更名 请求成功但系统自动避免重名做加后缀处理。

- **HTTP Method**

[POST]

- **PATH**

/api/v1/project/picture/name

- **Request**

含 token

JSON

```
{
    "projectID": 1,
    "uuid": "12341234",
    "name": "newname"
}
```

- **Response**

JSON

```
{
    "code": 0,
    "msg": "成功",
    "data": {}
}
```

修改组名称

- **HTTP Method**

  [POST]

- **PATH**

  /api/v1/project/group/name

- **Request**

  含 token

JSON

```
{
    "projectID": 1,
    "groupID": 12,
    "name": "newname"
}
```

- **Response**

JSON

```
{
    "code": 0,
    "msg": "成功",
    "data": {}
}
```

删除组

- **HTTP Method**

  [DELETE]

- **PATH**

  /api/v1/project/group

- **Request**

  含 token

JSON

```
{
    "projectID": 1,    //项目id
    "groupID": 28    //组id
}
```

- **Response**

JSON

```
{
    "code": 0,    //int，状态码
    "msg": "图片获取成功",    //string，返回信息
    "data": {}
}
```

获取项目下的所有分组和上传图片（已完成）

- **HTTP Method**

  [GET]

- **PATH**

  /api/v1/project/:id

- **Request**

含 token

- **Response**

> group分组中，第一张图片是**处理结果**，在**变化检测**分组中，第二张图片是**旧图**，第三张图片是**新图**，在**其他分组**中第二张图片是**原图片**。

> 组 type 说明（暂定，后续可能更改对应数字）：
>
> 综合分析台：1
>
> 目标提取：2
>
> 地物分类：3
>
> 目标检测：4
>
> 变化检测：5

> 在综合分析分组中，会除了 mark[] 数组指示的检测结果外，还会在最后追加一张检测原图，若该综合分析结果中存在变化检测分组（分组中含有原图）则不展示该图片，若综合分析结果中不存在变化检测分组则展示该原图。

JSON

```
{
    "code": 0,    //int, 状态码
    "msg": "成功",    //string, 返回信息
    "data": {
        "groups": [
            {
                "groupID": 223,
                "groupName": "综合分析台",
                "groupType": 1,
                "info": {
                    "mark": [
                        1,
                        1,
                        1,
                        1
                    ],
                    "infos": [
                        {},
                        {},
                        {},
                        {}
                    ]
                },
                "pictures": [
```

```
                    {
                        "uuid": "12329qq3232",
                        "name": "未命名",
                        "url": "xxx"
                    },
                    {
                        "uuid": "123qwqqezc122",
                        "name": "未命名",
                        "url": "xxx"
                    },
                    {
                        "uuid": "2qqqq",
                        "name": "test123",
                        "url": "xxx"
                    },
                    {
                        "uuid": "qweew11qq",
                        "name": "未命名",
                        "url": "xxx"
                    },
                    {
                        "uuid": "qqqqqq",
                        "name": "aircrafta",
                        "url": "xxx"
                    },
                    {
                        "uuid": "wwwwww",
                        "name": "qwe",
                        "url": "xxx"
                    }
                ]
            },
            {
                "groupID": 1,
                "groupName": "地物分类组1",
                "groupType": 3,
                "info": {
                    "colors": [0.2222, 0.1111, 0.3333, 0.3334, 0],
                    //长度为5的数组，分别对应五种颜色所占比例，各颜色代表含义见上
                    "nums": [5, 2, 3, 10, 11, 0]
                    //长度为5的数组，分别对应五种颜色块数量，各颜色代表含义见上
                },
                "pictures": [
                    {
                        "uuid": "234223",
                        "name": "img1",
                        "url": "xxx"
                    },
```

```
                    {
                        "uuid": "234224",
                        "name": "img2",
                        "url": "xxx"
                    }
                ]
            },
            {
                "groupID": 1,
                "groupName": "地物分类组2",
                "groupType": 3,
                "info": {
                    "colors": [0.2222, 0.1111, 0.3333, 0.3334, 0],
                    //长度为5的数组，分别对应五种颜色所占比例，各颜色代表含义见上
                    "nums": [5, 2, 3, 10, 11, 0]
                    //长度为5的数组，分别对应五种颜色块数量，各颜色代表含义见上
                },
                "pictures": [
                    {
                        "id": "234225",
                        "name": "img1",
                        "url": "xxx"
                    },
                    {
                        "id": "234226",
                        "name": "img2",
                        "url": "xxx"
                    }
                ]
            }
        ],    //string数组，图片存储 URL
        "pictures": [
            {
                "id": "234225",
                "name": "img1",
                "url": "xxx"
            }
        ]    //所有可以处理的图片
    }
}
```

删除项目中已上传图片

- **HTTP Method**

  [DELETE]

- **PATH**

/api/v1/project/picture

- **Request**

  含 token

JSON

```
{
    "projectID": 2,     //int, 项目id
    "pictures": []    //uuid数组
}
```

- **Response**

JSON

```
{
    "code": 0,    //int, 状态码
    "msg": "成功",    //string, 返回信息
    "data": {}
}
```

将项目移入回收站|软删除

- **HTTP Method**

  [POST]

- **PATH**

  /api/v1/project/:id/delete

- **Request**

  含 token

- **Response**

JSON

```
{
    "code": 0,    //int, 状态码
    "msg": "成功",    //string, 返回信息
    "data": {}
}
```

根据关键字获取回收站中的项目

- **HTTP Method**

[GET]

- **PATH**

/api/v1/project/recycle?keyword=name

- **Request**

含 token

- **Response**

JSON

```
{
    "code": 0,    //int，状态码
    "msg": "成功",    //string，返回信息
    "data": {
        "projects": [
            {
                "id": 22,
                "name": "name1",
                "lastVisit": "2022-05-11 12:00:00",    //string
                "coverURL": "xxx"
            },
            {
                "id": 12,
                "name": "name2",
                "lastVisit": "2022-05-11 12:00:00",    //string
                "coverURL": "xxx"
            },
        ]    //int，项目信息，按照匹配度顺序返回
    }
}
```

> 匹配模糊程度：项目名称完全含有该关键字
>
> 如：keyword: test，则 **test**/**test**1/1**test**/sdgagdf**test**asdfdsga 可以匹配，而 tesat/tes 无法匹配

获取回收站中的项目

- **HTTP Method**

[GET]

- **PATH**

/api/v1/project/recycle

- **Request**

含 token

- **Response**

JSON

```
{
    "code": 0,    //int，状态码
    "msg": "成功",    //string，返回信息
    "data": {
        "projects": [
            {
                "id": 22,
                "name": "name1",
                "lastVisit": "2022-05-11 12:00:00",    //string
                "coverURL": "xxx"
            },
            {
                "id": 12,
                "name": "name2",
                "lastVisit": "2022-05-11 12:00:00",    //string
                "coverURL": "xxx"
            },
        ]    //int，项目信息，按照最近访问顺序返回
    }
}
```

将回收站内的项目恢复

- **HTTP Method**

  [POST]

- **PATH**

  /api/v1/project/:id/recover

- **Request**

  含 token

- **Response**

JSON

```
{
    "code": 0,    //int，状态码
    "msg": "成功",    //string，返回信息
    "data": {}
}
```

将回收站中项目彻底删除|硬删除

- **HTTP Method**

  [DELETE]

- **PATH**

  /api/v1/project/:id

- **Request**

  含 token

- **Response**

JSON

```
{
    "code": 0,     //int，状态码
    "msg": "成功",     //string，返回信息
    "data": {}
}
```

图片处理相关接口

> 若该图片已存在综合分析分组且综合分析分组中不包含对应检测结果（**因变化检测目前允许重复，故变化检测不包含在内**），则返回
>
> {
>
> “code”：3007,
>
> "msg": "xxx",
>
> "data": {
>
> "groupID": 20 // 已存在的综合分析分组 id
>
> }
>
> }
>
> 用户选择重新创建综合分析分组并覆盖原分组则调用**删除分组接口**和**综合分析台接口**

综合分析台（已修改）

> 已存在再做综合分析，则返回：
>
> {
>
> "code": 3004,
>
> "msg": "该图片的综合分析分组已存在",

```
    "data": {}
}
```

其他三种只会存在一个相关分组的检测操作，若重复再做检测会直接返回上次检测结果，也不会有新的分组产生，不需要特殊处理

说明：若该图片未做任何处理分析，改为不生成综合分析分组，返回 code = 3006 错误码。

- **HTTP Method**

  [POST]

- **PATH**

  /api/v1/project/picture/overall

- **Request**

  含 token

JSON

```
{
    "projectID": 1,
    "originUUID": "12345678",    //待处理图片uuid
    "groupName": "综合分析台name"
}
```

- **Response**

说明：分析台中目标提取结果、地物分类结果、目标检测结果均只会有一张，而变化检测分组（结果、新图、旧图）可能有多组

JSON

```
{
    "code": 0,    //int，状态码
    "msg": "图片上传成功",    //string，返回信息
    "data": {
        "oa": {
            "uuid": "1231231",
            "url": "xxx",
            "name": "目标提取结果"
        },    //目标提取结果
        "gs": {},    //地物分类结果（若图片未做地物分类检测，则为空）
        "od": {},    //目标检测结果
        "cd": [
            [
                {
                    "uuid": "123123",
```

```
                    "url": "xxx",
                    "name": "变化检测结果"
                },
                {
                    "uuid": "123123",
                    "url": "xxx",
                    "name": "新图"
                },
                {
                    "uuid": "123123",
                    "url": "xxx",
                    "name": "旧图"
                }
            ],    //一组变化检测的结果，数组长度为3，分别为 结果图/新图/旧图
            [
                {
                    "uuid": "123123",
                    "url": "xxx",
                    "name": "变化检测结果"
                },
                {
                    "uuid": "123123",
                    "url": "xxx",
                    "name": "新图"
                },
                {
                    "uuid": "123123",
                    "url": "xxx",
                    "name": "旧图"
                }
            ]
        ]    //变化检测结果（可能有多组）
    }
}
```

地物分类接口

> 颜色含义对应：
>
> 0：建筑
>
> 1：耕地
>
> 2：林地
>
> 3：其他
>
> 4：不考虑区域
>
> 序号与返回数据数组下标对应

- **HTTP Method**

  [POST]

- **PATH**

  /api/v1/project/picture/gs

- **Request**

  含 token

JSON

```
{
    "projectID": 1,
    "originUUID": "12345678",    //待处理图片uuid
    "targetUUID": "8888888",    //处理结果的uuid
    "targetName": "地物分类结果"    //处理结果name
}
```

- **Response**

JSON

```
{
    "code": 0,    //int，状态码
    "msg": "图片上传成功",    //string，返回信息
    "data": {
        "url": "xxx",
        "name": "地物分类结果",
        "info": {
            "colors": [0.2222, 0.1111, 0.3333, 0.3334, 0],
            //长度为5的数组，分别对应五种颜色所占比例，各颜色代表含义见上
            "nums": [5, 2, 3, 10, 11, 0]
            //长度为5的数组，分别对应五种颜色块数量，各颜色代表含义见上
        }
    }
}
```

变化检测接口

测试图片大小必须为1024×1024

- **HTTP Method**

  [POST]

- **PATH**

  /api/v1/project/picture/cd

- **Request**

  含 token

JSON

```
{
    "projectID": 1,
    "oldUUID": "12345678",    //旧图片uuid
    "newUUID": "12121212",    //新图片uuid
    "targetUUID": "8888888",    //处理结果的uuid
    "targetName": "变化检测结果"    //处理结果name
}
```

- **Response**

JSON

```
{
    "code": 0,    //int，状态码
    "msg": "成功",    //string，返回信息
    "data": {
        "url": "xxx",
        "name": "变化检测结果",
        "info": {
            "colors": [0.2222, 0.7778],
            //长度为2的数组，分别对应2种颜色所占比例，各颜色代表含义见上
            //index0：非房屋，index1：房屋
            "num": 20    //房屋数量
        }
    }
}
```

目标检测（待修复）

TODO：修复结果图片不透明的bug（jpg图片背景不能为透明，只有png可以）

> **说明：每个图片只做一种类型的检测**

飞机目标检测推荐测试图片：飞机目标检测测试图片

其他暂未找到合适测试图片

- **HTTP Method**

  [POST]

- **PATH**

  /api/v1/project/picture/od

- **Request**

  含 token

JSON

```
{
    "projectID": 1,
    "type": "oiltank",//目标检测类型
("aircraft","overpass","oiltank","playground")
    "originUUID": "12345678",    //待处理图片uuid
    "targetUUID": "8888888",    //处理结果的uuid
    "targetName": "目标检测结果"    //处理结果name
}
```

- **Response**

JSON

```
{
    "code": 0,    //int，状态码
    "msg": "成功",    //string，返回信息
    "data": {
        "url": "xxx",
        "name": "目标检测结果",
        "info": {
            "type": "aircraft",
            "w": 100,    //图片总宽
            "h": 100,    //图片总高
            "boxs": [
                [2, 2, 1, 3],    //长度为4的一维数组，元素分别为左上顶点x、左上顶点y、
宽、高
                [4, 2, 9, 1]
            ]    //二维数组，框集合
        }
    }
}
```

目标提取

推荐测试图片：测试图片

- **HTTP Method**

  [POST]

- **PATH**

  /api/v1/project/picture/oa

- **Request**

  含 token

JSON

```json
{
    "projectID": 1,
    "originUUID": "12345678",    //待处理图片uuid
    "targetUUID": "8888888",    //处理结果的uuid
    "targetName": "目标提取结果"    //处理结果name
}
```

- **Response**

JSON

```json
{
    "code": 0,    //int，状态码
    "msg": "成功",    //string，返回信息
    "data": {
        "url": "xxx",
        "name": "目标提取结果",
        "info": {
            "colors": [0.2222, 0.7778],
            //长度为2的数组，分别对应2种颜色所占比例，各颜色代表含义见上
        }
    }
}
```

算法功能接口（python-go交互）

简单三步调用

Python

```python
import paddlers as pdrs
import paddle
# nvidia-smi

# 指定GPU
paddle.device.set_device('gpu:2')
# 将导出模型所在目录传入Predictor的构造方法中
model_dir='/home/uu201915762/workspace/main/test/gs'
predictor = pdrs.deploy.Predictor(model_dir, use_gpu=True)
# img_file参数指定输入图像路径
img_path='/home/uu201915762/workspace/Data/train_and_label/img_train/T000009.jpg'
pred = predictor.predict(img_file=img_path)
```

预测接口调用

- **基本使用**

以下是一个调用PaddleRS Python预测接口的实例。首先构建 `Predictor` 对象，然后调用 `Predictor` 的 `predict()` 方法执行预测。

```
import paddlers as pdrs
# 将导出模型所在目录传入Predictor的构造方法中
predictor = pdrs.deploy.Predictor('./inference_model')
# img_file参数指定输入图像路径
result = predictor.predict(img_file='test.jpg')
```

- **在预测过程中评估模型预测速度**

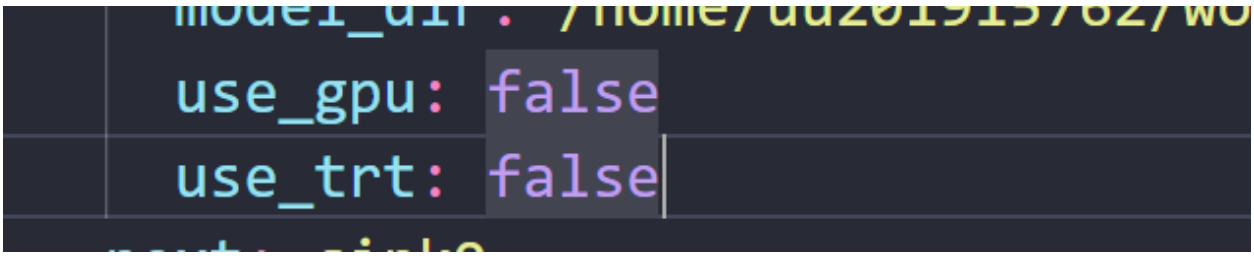
加载模型后，对前几张图片的预测速度会较慢，这是因为程序刚启动时需要进行内存、显存初始化等步骤。通常，在处理20-30张图片后，模型的预测速度能够达到稳定值。基于这一观察，**如果需要评估模型的预测速度，可通过指定预热轮数 `warmup_iters` 对模型进行预热。**此外，**为获得更加精准的预测速度估计值，可指定重复 `repeats` 次预测后计算平均耗时。**

```
import paddlers as pdrs
predictor = pdrs.deploy.Predictor('./inference_model')
result = predictor.predict(img_file='test.jpg',
                           warmup_iters=100,
                           repeats=100)
```

Python

```
model文件目录树
--model_dir
----model.pdiparams
----model.pdiparams.info
----model.pdmodel
----model.yml
----pipline.yml
```

可以修改pipline.yml的配置，决定是否启用gpu和tensort



勘误 要在初始化预测器参数指定才能使用GPU

Python

```
predictor = pdrs.deploy.Predictor(model_dir, use_gpu=True)
```

所有模型文件父目录在/home/uu201915762/workspace/main/test/下

地物分类

- **HTTP 方法**

  [GET]

- **PATH**

  gs/

- **Response**

pred输出格式

JSON

```
{
    "label_map": [w,h]
    //"score_map": [w,h,4]
}
```

> pred['label_map']大小和图片一样，每个点取值为 0，1，2，3 代表不同类别

目标提取

- **HTTP 方法**

  [GET]

- **PATH**

  oa/

- **Response**

JSON

```
{
    "label_map": [w,h]
    //"score_map": [w,h,2]
}
```

> pred['label_map']大小和图片一样，每个点取值为 0，1 0代表像素点不是道路 1代表像素点是道路

目标检测

- **HTTP 方法**

  [GET]

- **PATH**

od/aircraft/

飞机目标检测

od/overpass/

立交桥目标中心检测

od/oiltank/

油井检测

od/playground/

操场检测

以上所有模型（飞机，立交桥，油井，操场）的pred格式是一致的,一个列表，里面n个字典代表一张图片的n个检测框，按score分数从高到低排列，box 和score是要重点关注的 只取score大于0.5的，bbox 检测物体的框 x,y,w,h

结构如下：

JSON

```
[
    {
        'category_id': 0,
        'category': 'aircraft',
        'bbox': [237.32615661621094, 10.862650871276855, 2.6553955078125, 2.4552078247070312],
        'score': 0.010384872555732727
    },
    {
        'category_id': 0,
        'category': 'aircraft',
        'bbox': [237.32615661621094, 10.862650871276855, 2.6553955078125, 2.4552078247070312],
        'score': 0.010384872555732727
    },
    ...
]
```

变化检测

目标提取一样的输出结构

label_map,score_map,只取label_map

模型路径在cd目录下

w,h:1024*1024

但是输入要两张图片

Apache

```
result = predictor.predict(img_file=[('img_path_11','img_path_12'),
('img_path_21','img_path_22'),
.....]
```