**游戏视频理解AI能力定制项目**

**详细设计方案**

咪咕互动娱乐有限公司

2017.10

**修订记录**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **版本** | **修订日期** | **修订内容** | **修订人** | **核准人** |
| V1.0 | 2017/10/10 | 创建文档 | 侯逸文 |  |
| V1.1 | 2017/10/14 | 编写文档主体内容 | 侯逸文  管筱  张运伟 |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**目 录**

[1 概述 5](#_Toc495864407)

[1.1 建设背景 5](#_Toc495864408)

[1.2 参考文档 5](#_Toc495864409)

[1.3 建设目标 5](#_Toc495864410)

[2 总体设计 7](#_Toc495864411)

[2.1 总体架构 7](#_Toc495864412)

[2.2 功能架构 8](#_Toc495864413)

[2.3 系统流程 9](#_Toc495864414)

[2.4 数据拓扑 11](#_Toc495864415)

[3 功能设计 12](#_Toc495864416)

[3.1 视频模型训练 12](#_Toc495864417)

[3.1.1 视频模型管理 12](#_Toc495864418)

[3.1.2 视频标签同步 17](#_Toc495864419)

[3.1.3 训练视频采集 18](#_Toc495864420)

[3.1.4 样本数据标注 18](#_Toc495864421)

[3.1.5 数据上传云平台 25](#_Toc495864422)

[3.1.6 视频模型训练 27](#_Toc495864423)

[3.1.7 训练结果发布 35](#_Toc495864424)

[3.2 视频模型使用 36](#_Toc495864425)

[3.2.1 源视频同步 36](#_Toc495864426)

[3.2.2 视频模型使用 37](#_Toc495864427)

[3.2.3 模型结果输出 47](#_Toc495864428)

[3.2.4 模型反馈修正 49](#_Toc495864429)

[3.3 咪咕游戏配套 50](#_Toc495864430)

[3.3.1 数据表 50](#_Toc495864431)

[3.3.2 视频标签管理 53](#_Toc495864432)

[3.3.3 AI视频合成 56](#_Toc495864433)

[3.3.4 AI视频发布 62](#_Toc495864434)

[3.4 系统接口 64](#_Toc495864435)

[3.4.1 数据上传接口 64](#_Toc495864436)

[3.4.2 模型发布接口 64](#_Toc495864437)

[3.4.3 模型使用接口 64](#_Toc495864438)

[3.4.4 标签同步接口 65](#_Toc495864439)

[3.4.5 视频同步接口 65](#_Toc495864440)

[3.4.6 结果输出接口 65](#_Toc495864441)

[3.4.7 模型反馈接口 65](#_Toc495864442)

[4 系统响应码 66](#_Toc495864443)

# 概述

## 建设背景

咪咕游戏作为咪咕旗下重点产品，在不断寻求产品突破，整合产品体系架构，丰富产品内容，运营了包含游戏视频、电竞赛事、热门游戏等游戏内容版块。随着自然语言处理、计算机视觉、机器学习等技术的发展，AI技术已经用范围越来越广，众多顶级企业IBM、谷歌、百度、爱奇艺等不断扩大AI领域布局，通过AI技术提高产品体验，实现产品内容智能化。

游戏视频理解AI能力定制项目（以下简称“项目”）是为应对当前的AI发展趋势，配合总部建设AI统一云平台，通过构建统一的、跨媒介的、多形态交互能力的AI云服务，为咪咕游戏提供统一的AI服务与能力。通过AI服务于咪咕游戏，形成差异化能力，提升客户体验，打造杀手级应用；通过AI提升业务全流程效率，节省研发/运营成本；通过AI整合并挖掘内部数据和能力，实现公司数据价值最大化；通过AI构建新的业务或商业模式。

## 参考文档

* 《灵犀云三期扩容工程项目（游戏视频理解AI能力定制）技术服务规范书》
* 《灵犀云三期扩容工程项目（AI统一云平台与通用能力）技术服务规范书》
* 《咪咕公司AI能力数据规范》
* 《咪咕公司AI数据标注规范》
* 《游戏视频理解AI能力定制项目需求分析方案》

## 建设目标

本项目计划建设咪咕游戏产品的AI化定制能力，AI能力由引擎软件和下层原子模型组成，在特定深度学习框架下采用适合的算法构建神经网络，通过基于业务场景的数据集训练，形成原子模型。多个原子模型组合并按照一定的业务逻辑构建出满足业务场景的AI能力，多个AI能力的组合形成产品AI化解决方案。AI化能力接入AI统一云平台的前置服务子平台，使用平台的服务编排能力对外提供HTTP API接口。

咪咕游戏AI能力包括游戏特定主体检测能力、游戏关键场景提取能力、游戏技能招式识别能力、游戏精彩度理解能力、游戏精彩视频剪辑能力共五部分，五部分的先后关系为：



项目主要实现的业务场景如下：

1. 游戏特定主体检测

AI对游戏视频资源进行处理，扫描视频中的每一帧图像，检测出特定的游戏人物、道具等主体形象。

1. 游戏关键场景提取

AI对游戏视频资源进行处理，扫描视频中的每一帧图像，提取出指定的游戏关键场景。

1. 游戏技能招式识别

AI对游戏视频资源进行处理，扫描视频中的每一帧图像，识别出指定的游戏人物技能招式。

1. 游戏精彩度理解

AI识别出视频每一帧图像中指定的主体形象、关键场景、人物招式、视频标题、视频字幕等关键内容，并根据关键内容的语义关系对视频内容进行理解，基于理解结果按照规则对视频片段进行精彩度打分。

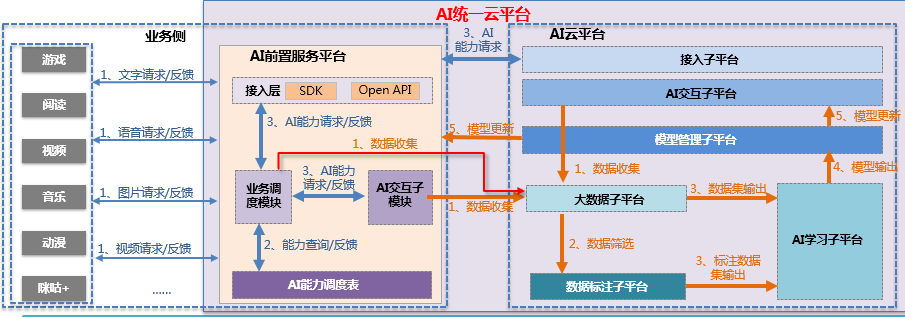
1. 游戏精彩视频剪辑

AI基于对整体视频的理解和视频片段精彩度得分，按播放时间顺序自动选取打分高于设置阀值的视频片段，结合镜头和时长的要求，将长视频智能剪辑为精彩短视频，并输出最具代表性的优质封面图。

# 总体设计

## 总体架构

咪咕AI云平台的总体架构如图1所示，整个平台由咪咕游戏、AI前置服务平台、AI云平台三大部分组成。



**图1 AI云平台架构示意**

AI云平台中各个部件的主要功能如下：

1. **咪咕游戏，产品侧-南京互娱**

* 负责游戏源视频和训练视频采集、格式转换，并向AI云平台上传样本视频
* 负责游戏主体、场景和招式的标签定义
* 负责精彩短视频预览、合成和发布
* 负责咪咕游戏前端的内容呈现

1. **AI前置服务平台，能力侧-南京互娱**

* 负责下载模型文件，并载入机器学习框架运行
* 负责提供视频理解的AI能力，实现标签识别、视频剪辑、封面输出
* 负责接入AI云平台提供的其他AI能力服务
* 负责各个AI能力服务的流程编排、定时调度和对外输出

1. **AI云平台，平台侧-成都大数据**

* 负责接收、清洗、存储源视频和训练视频
* 负责对训练视频进行数据标注，形成训练样本
* 负责载入训练样本，对视频模型进行训练，输出模型文件
* 负责对各类模型进行统一管理，包括注册、发布、评测、版本管理等

## 功能架构

根据系统的总体功能和系统架构，将各部分功能细化，形成整体功能架构，如图2所示。



**图2 系统功能架构示意图**

整个系统由游戏客户端、游戏服务端、互娱大数据、咪咕云、AI前置平台、AI云平台共六个部件构成，各部件的细化功能如下：

1. **游戏客户端**

* 与游戏服务端交互数据，向用户呈现精彩集锦、英雄教学、赛事精华等内容。
* 播放存储在咪咕云上的游戏视频。

1. **咪咕云**

* 保存游戏视频源文件，支持视频播放。

1. **游戏服务端**

* 与游戏客户端端交互数据，向用户呈现游戏视频内容。
* 向运营人员提供标签管理界面，定义各个游戏标签，并将标签同步给互娱大数据平台。
* 调用互娱大数据平台的游戏视频模型使用接口，获取长视频的游戏标签、精彩短视频的标签和播放地址。

1. **互娱大数据**

* 在接收到游戏服务端的模型后，调用AI前置平台的模型服务接口，获取长视频的游戏标签、精彩短视频的标签和播放地址。
* 2017年向算法人员提供模型管理功能，包括接收并查询游戏服务端同步来的标签数据、查询模型日志、查询和载入模型版本等功能。
* 2017年向标注人员提供数据标注功能，可完成源视频的样本标注。
* 按照AI数据规范要求将源视频、标注结果导出成规定格式，生成数据文件、字段文件、校验文件，并传输到FTP前置机上。

1. **AI前置平台**

* 使用模型部署工具在Tensflow框架中部署最新版的游戏视频理解模型文件。
* 对外提供HTTP形式的游戏视频理解模型服务。

1. **AI云平台**

* 存储模型的训练样本等数据。
* 向算法人员提供Tensflow框架进行游戏视频理解模型训练，完成训练的模型可导出成模型文件，下载到AI前置平台部署。
* 2018年向算法人员提供模型管理功能，包括接收并查询善跑服务端同步来的各类数据、查询模型日志、查询载入模型版本等功能。
* 2018年向标注人员提供数据标注功能，可完成源视频的样本标注。

## 系统流程

1. **AI模型训练流程**



1. **AI模型使用流程**



## 数据拓扑



# 功能设计

## 视频模型训练

### 视频模型管理

#### 数据表

游戏模型表（game\_model）、游戏标签表（game\_label）、游戏样本上传表（game\_sample\_upload）、游戏模型日志表（game\_model\_log）、游戏模型版本表(game\_model\_version)、游戏在线使用日志表（game\_api\_log）

游戏模型表（目前存放有关键场景模型、英雄人物模型，game\_id和model\_id组成联合主键）

|  |  |  |  |
| --- | --- | --- | --- |
| game\_model 游戏模型表 | | | |
| 字段 | 类型 | 是否必填 | 说明 |
| game\_id | Number(11) | Y | game id，自增id。比如王者荣耀为1 |
| model\_id | Number(11) | Y | model id，自增id。比如王者荣耀中的英雄人物识别模型为1，关键场景模型为2。 |
| model\_name | Number(11) | Y | 模型的名字。 |
| description | varchar2(2048) | Y | （game\_id + model\_id）对应着的模型的说明，包括训练和使用情况。 |
| create\_time | date | N | （game\_id + service\_id）对应的记录的创建时间 |
| update\_time | date | N | 记录的最新一次修改时间 |
| status | Varchar2(128) | N | 模型的状态 |

游戏标签表（存放要标注的英雄人物和场景的标签）

|  |  |  |  |
| --- | --- | --- | --- |
| game\_label 游戏标签表 | | | |
| 字段 | 类型 | 是否必填 | 说明 |
| Id | Number(11) | Y | 标注系统中的标签id，自增id |
| category | varchar2(20) | Y | 该标签所述的分类，比如王者荣耀的英雄人物 |
| label\_name | varchar2(20) | Y | 标注系统中标签的名字 |
| label\_color | varchar2(20) | Y | 标签在网页上显示的颜色 |
| label\_order | number(11) | Y | 标签在网页上显示的顺序 |
| label\_description | varchar2(200) | Y | 标签说明 |
| label\_create\_time | date | Y | 标签创建时间 |
| label\_update\_time | date | Y | 标签最新一次修改时间 |
| Status | varchar2(20) | Y | 标签的状态（正常，废弃，失效…） |

游戏模型日志表（定时任务从接口机上扫描目录发现新的训练模型和日志文件）

|  |  |  |  |
| --- | --- | --- | --- |
| game\_model\_log游戏模型日志表 | | | |
| 字段 | 类型 | 是否必填 | 说明 |
| model\_version\_id | number(11) | Y | 模型版本ID，自增 |
| model\_version\_name | varchar2(128) | Y | 模型版本号，由主版本、次版本、patch号、时间 组成 |
| train\_user | varchar2(70) | Y | 模型训练人员 |
| model\_version | varchar2(70) | Y | 模型版本，由主版本、次版本、patch号、时间 组成 |
| model\_name | varchar2(70) | Y | 模型名称 |
| model\_description | varchar2(200) | Y | 模型介绍 |
| game\_id | number(11) | Y | 游戏ID |
| model\_id | number(11) | Y | 模型服役的业务id |
| model\_name | varchar2(255) | Y | 模型服役的业务名称 |
| dataset\_id | number(11) | Y | 训练此模型所用数据集id |
| testset\_id | number(11) | Y | 验证此模型所用测试集id |
| model\_file\_name | varchar2(128) | Y | 模型文件压缩包的名称 |
| model\_file\_url | varchar2(100) | Y | 模型文件压缩包的地址 |
| model\_train\_status | Bool | Y | 模型训练操作是否正常完成 |
| model\_file\_size | varchar2(70) | Y | 模型文件压缩包大小 |
| model\_file\_md5 | varchar2(70) | Y | 上传的模型文件的md5校验 |
| model\_train\_log | text | Y | 模型训练过程中产生的log |
| Starttime | date | Y | 训练开始时间 |
| Endtime | date | Y | 训练结束时间 |
| train\_last\_time | number(11) | Y | 模型训练耗时 |
| batch\_size | number(11) | Y | 模型训练的batch size 大小 |
| Epoch | number(11) | Y | 模型训练的epoch大小 |
| test\_score | number(11) | Y | 模型在测试集上的成绩 |

游戏模型版本表（记录模型的版本及是否正在使用）

|  |  |  |  |
| --- | --- | --- | --- |
| game\_model\_version游戏模型版本表 | | | |
| 字段 | 类型 | 是否必填 | 说明 |
| model\_id | number(11) | Y | 模型服役的业务id |
| model\_name | varchar2(255) | Y | 模型服役的业务名称 |
| model\_version | Varchar2(128) | Y | 模型版本，由主版本、次版本、patch号、时间 组成 |
| model\_version\_id | Number(11) | Y | 模型版本id |
| model\_file\_path | Varchar2(2048) | N | 模型文件夹路径 |
| is\_used | Bool | N | 已使用/未使用 |
| create\_time | Date | N | 创建时间 |
| update\_time | Date | N | 状态修改时间 |
| Status | Varchar2(128) | N | 状态 |

游戏模型在线使用日志表（模型运行时每次REST调用会产生一条调用日志记录）

|  |  |  |  |
| --- | --- | --- | --- |
| game\_api\_log游戏模型API调用日志表 | | | |
| 字段 | 类型 | 是否必填 | 说明 |
| api\_id | number(11) | Y | REST API调用的记录id，自增 |
| game\_id | number(11) | Y | 哪个游戏的REST API |
| model\_id | number(11) | Y | 某个AI服务对应的id |
| model\_version\_id | number(11) | Y | 所使用的model的版本id |
| start\_time | Date | Y | API调用开始的时间 |
| end\_time | Date | Y | API调用结束的时间 |
| last\_time | number(11) | Y | API调用耗时 |
| Code | varchar2(10) | Y | API调用的返回码 |
| response | varchar2(1000) | Y | API调用的返回值 |
| Log | Text | Y | API调用期间产生的log |

#### 主要功能

视频模型管理为BS架构，前端由html+css+javascript负责提供界面的渲染，后端由Nginx+uwsgi+python web framework + sqlite/mysql提供REST API、逻辑处理以及数据的关系型存储。

1. **模型管理**

****

界面功能说明：

1. 此界面可以查看已经定义的模型，每一行是一个模型，以及这个模型属于哪一个游戏的哪一个业务（比如是人物识别还是场景识别）。
2. 点击“标签详情”按钮可以查看模型所对应的标签，链接将跳转至标注系统中的标签页面，在那里可以查看每个标签记录的创建时间和状态更新时间，参见第3.1.4节。
3. 点击“模型详情”按钮可以查看模型版本情况和模型训练日志。
4. 点击“日志详情”按钮可以查看模型的样本上传日志、训练日志、发布日志、使用日志等。
5. 点击“筛选”按钮，可以按文本框中的关键词筛选列表中的结果。
6. **模型详情**

在“模型管理”界面中点击某条模型记录后的“模型详情”按钮时，会进入此界面。



界面功能说明：

1. 此界面可以查看模型版本情况和模型训练日志。
2. 点击“使用”按钮，会将对应版本的模型文件信息写入到tensorflow框架初始化配置文件中，当tensorflow重新初始化时，就会按照配置文件中模型版本载入相应的模型文件，重新初始化网络参数。点击使用的版本状态标记为“已使用”，其他版本状态标记为“未使用”。配置文件model.conf格式如下：

[

{

"game\_id": 1,

"service\_id": 1,

"model\_name": "英雄人物",

"model\_version": "1.0.0",

"file\_name": "mghy\_mgAI\_Origin\_model\_3\_20171003\_00000001\_01",

"file\_size": "500M",

"file\_path":"file:///data/mghy\_mgAI\_Origin\_model\_3\_20171003\_00000001\_01",

"file\_time": "2017-10-03 10:00:00"

},

......

]

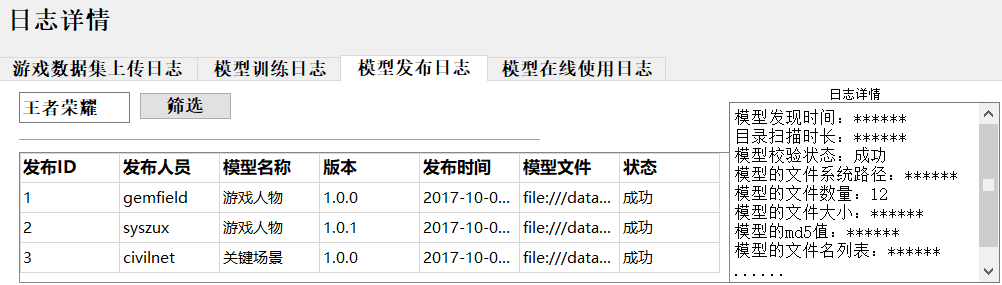
1. 点击“发现新版本”按钮，服务会到指定的文件夹下读取新的模型文件：如果有新的模型文件，会读取并分析其的日志记录，解析出其中的版本号，同时将模型信息添加到版本列表中，同时生成一条模型发布日志。如果没有新的模型文件，则不做处理。支持手动点击按钮触发，也支持后台的定时任务去扫描。
2. **数据上传日志查询**

在“模型管理”界面中点击某条模型记录后的“日志详情”按钮时，会进入此界面。查询数据集从标注系统导出到接口机相应目录的日志，可以按照时间、文件名、操作人员、关键字等查询条件进行查询。



1. 模型发布日志查询

在“模型管理”界面中点击某条模型记录后的“日志详情”按钮时，会进入此界面。查询模型发布的日志，可以按照时间段、发布人员、模型名字、版本号、关键字等筛选条件进行查询。



1. **模型使用日志查询**

在“模型管理”界面中点击某条模型记录后的“日志详情”按钮时，会进入此界面。查询模型在线使用日志，也就是模型对外提供的API调用情况，可以按照时间段、api接口名字、客户端ip、服务端ip、4位状态码等筛选条件进行查询。



### 视频标签同步

#### 数据表

游戏标签表（game\_label）

#### 主要功能

游戏服务端将运营人员配置的游戏角色标签、游戏场景标签通过互娱内部的kafka消息队列同步给互娱大数据平台，互娱大数据平台接收后保存入标签表供样本数据标注时使用，标签数据为JSON格式，实时增量同步，主要字段如下：

|  |  |
| --- | --- |
| 功能 | 咪咕游戏向大数据平台同步游戏角色标签 |
| Topic | t\_mggame\_to\_bigdata\_person\_label |
| 数据表 | 游戏标签表（game\_label） |
| content\_id | 游戏内容编码 |
| label \_id | 标签编码 |
| label \_name | 标签名称 |
| role\_type | 角色类型，坦克、战士、刺客、辅助、法师、射手 |
| sex | 性别, 0其他，1男，2女 |
| role\_screenshot | 角色截图地址 |
| oper\_type | 操作类型，1-新增，2-修改 |

|  |  |
| --- | --- |
| 功能 | 咪咕游戏向大数据平台同步游戏场景标签 |
| Topic | t\_mggame\_to\_bigdata\_scene\_label |
| 数据表 | 游戏标签表（game\_label） |
| content\_id | 游戏内容编码 |
| label \_id | 标签编码 |
| label \_name | 标签名称 |
| scence\_score | 场景得分 |
| scence\_describe | 场景描述 |
| scence\_screenshot | 场景截图地址 |
| oper\_type | 操作类型，1-新增，2-修改 |

### 训练视频采集

#### 数据表

游戏视频表（game\_video）、游戏样本表（game\_sample）

#### 主要功能

游戏运营人员离线方式统计好需要标注的游戏视频列表，互娱大数据平台根据视频地址从咪咕云上直接下载视频源文件，下载成功后将视频的元数据保存入游戏视频表（game\_video）、游戏样本表（game\_sample），各新增一条记录，视频下载方式为HTTP。

### 样本数据标注

#### 数据表

游戏样本表（game\_sample）、游戏标签表（game\_label）、游戏样本结果标注表（game\_sample\_label）、游戏样本上传表（game\_sample\_upload）

游戏视频标签对应表(game\_video\_label)

游戏样本表（存放待标注的视频）

|  |  |  |  |
| --- | --- | --- | --- |
| game\_sample 游戏样本表 | | | |
| 字段 | 类型 | 是否必填 | 说明 |
| id | Number(11) | Y | 标注系统中的视频id，自增id |
| category | varchar2(20) | Y | 该视频所属的分类，比如王者荣耀 |
| labels | varchar2(100) | Y | 该视频上将会使用的标签 |
| host | varchar2(100) | Y | 待标注视频所在服务器，可为本地 |
| filename | varchar2(100) | Y | 待标注视频在文件系统上的路径 |
| video\_description | varchar2(200) | Y | 待标注视频说明 |
| video\_source | varchar2(20) | Y | 待标注视频来自于原始视频，还是AI交互子平台上预测出错的视频 |

游戏样本结果标注表（存放标注结果）

|  |  |  |  |
| --- | --- | --- | --- |
| game\_sample\_label游戏样本结果标注表 | | | |
| 字段 | 类型 | 是否必填 | 说明 |
| video\_id | Number(11) | Y | 标注系统中的视频id |
| annotation | varchar2(20) | Y | Json格式的标注结果 |
| annotator | Number(11) | Y | 标注人员的用户id |
| annotation\_start\_time | date | Y | 此视频标注开始时间 |
| annotation\_update\_time | date | Y | 此视频标注最新一次的submit时间 |
| rejected | bool | Y | 标注成果是否被拒绝 |
| verified | Bool | Y | 标注成果是否被批准 |

游戏视频标签对应表（模型运行时每次REST调用返回值中标签对应部分会录入此表）

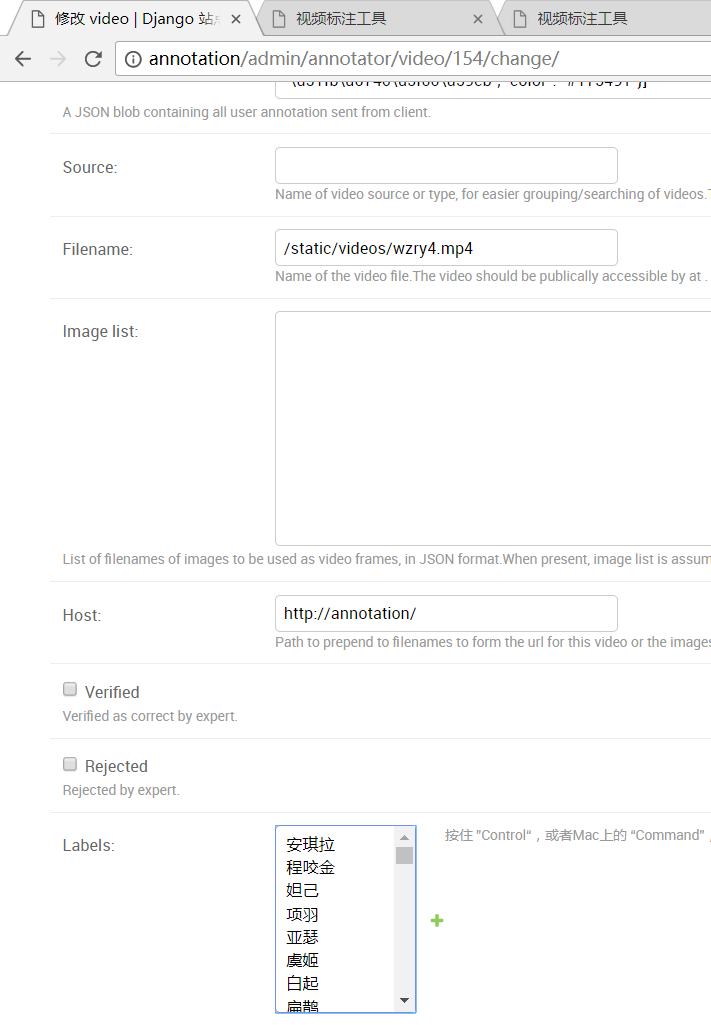
|  |  |  |  |
| --- | --- | --- | --- |
| game\_video\_label游戏视频标签对应表 | | | |
| 字段 | 类型 | 是否必填 | 说明 |
| video\_id | number(11) | Y | REST API调用的输入视频id |
| api\_id | number(11) | Y | 对应的API调用id |
| Labels | varchar2(10) | Y | 识别出来的label |

#### 主要功能

标注人员使用视频标注工具来进行游戏视频的标注。视频标注工具主要分为后台管理、视频标注、标注验证三大部分。后台管理的内容主要有：

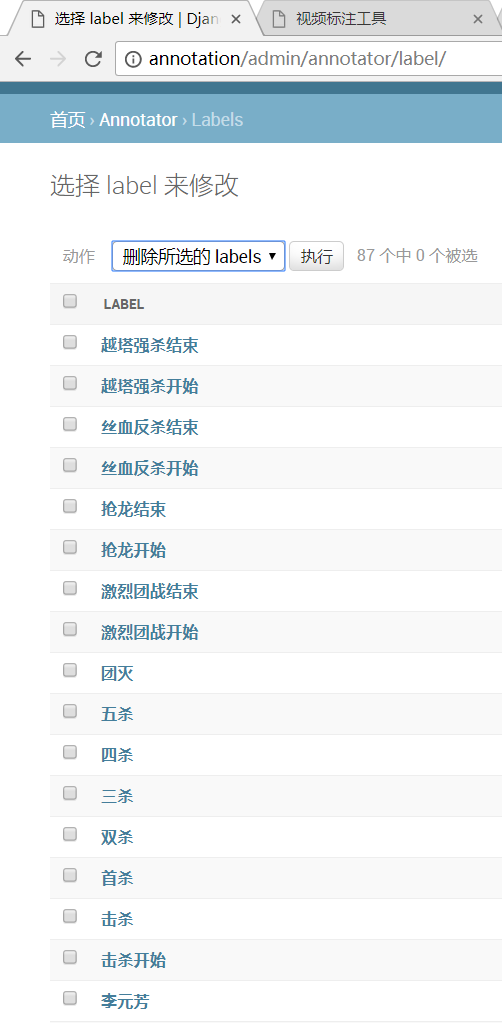
1. **后台管理-添加视频**

要标注的视频需要符合数据规范要求的格式，然后导入到标注系统web框架根目录下的/static/videos目录下。之后使用管理员账号登陆到后台，在添加视频的对话框中，填入视频所在的主机和路径，如下图所示：



1. **后台管理-添加标签**

视频标注工具的标签管理有2种方式，一种是通过REST API的方式进行增删改查，另一种是从后台管理界面进行增删改查，如下图所示：



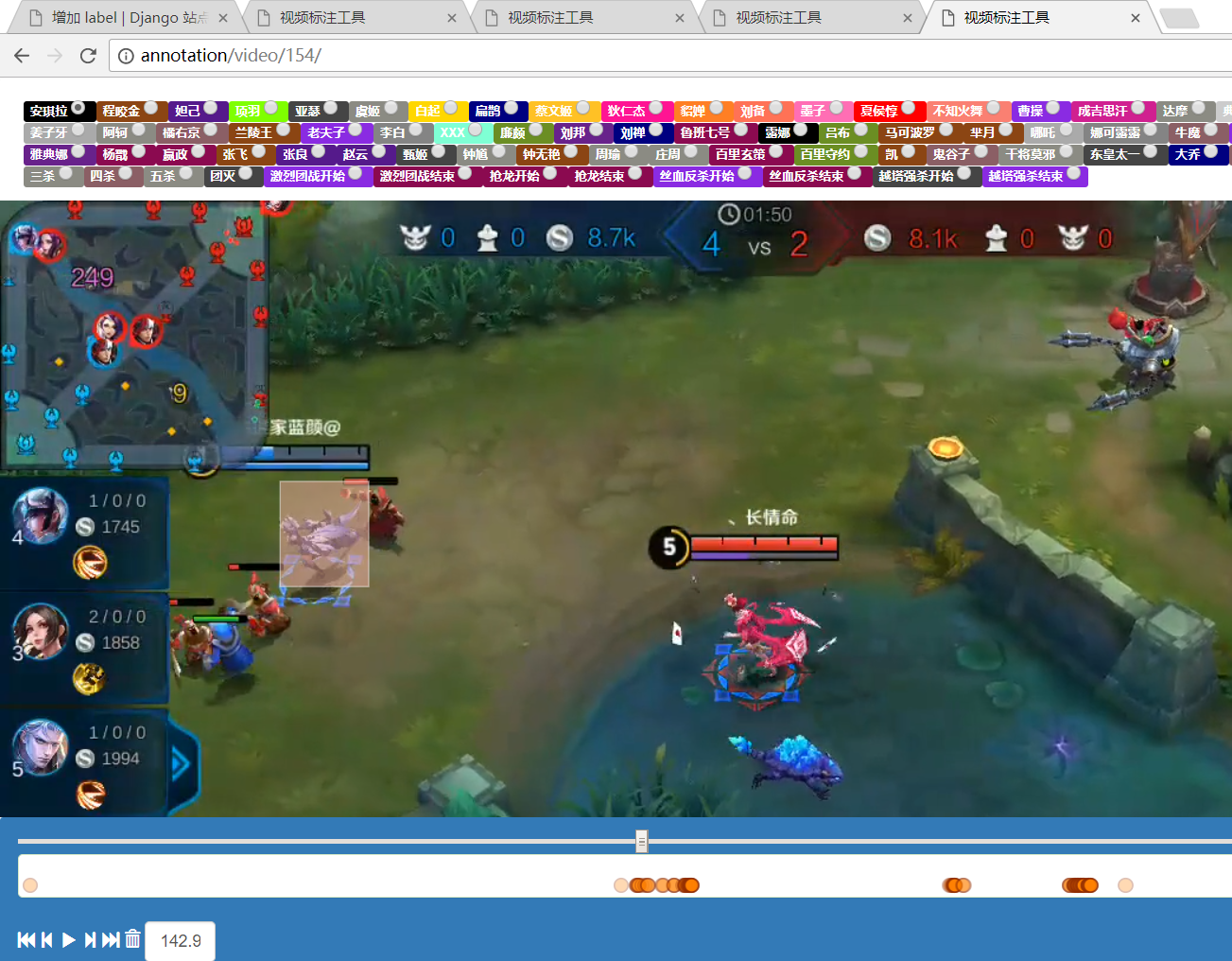


1. **视频标注**

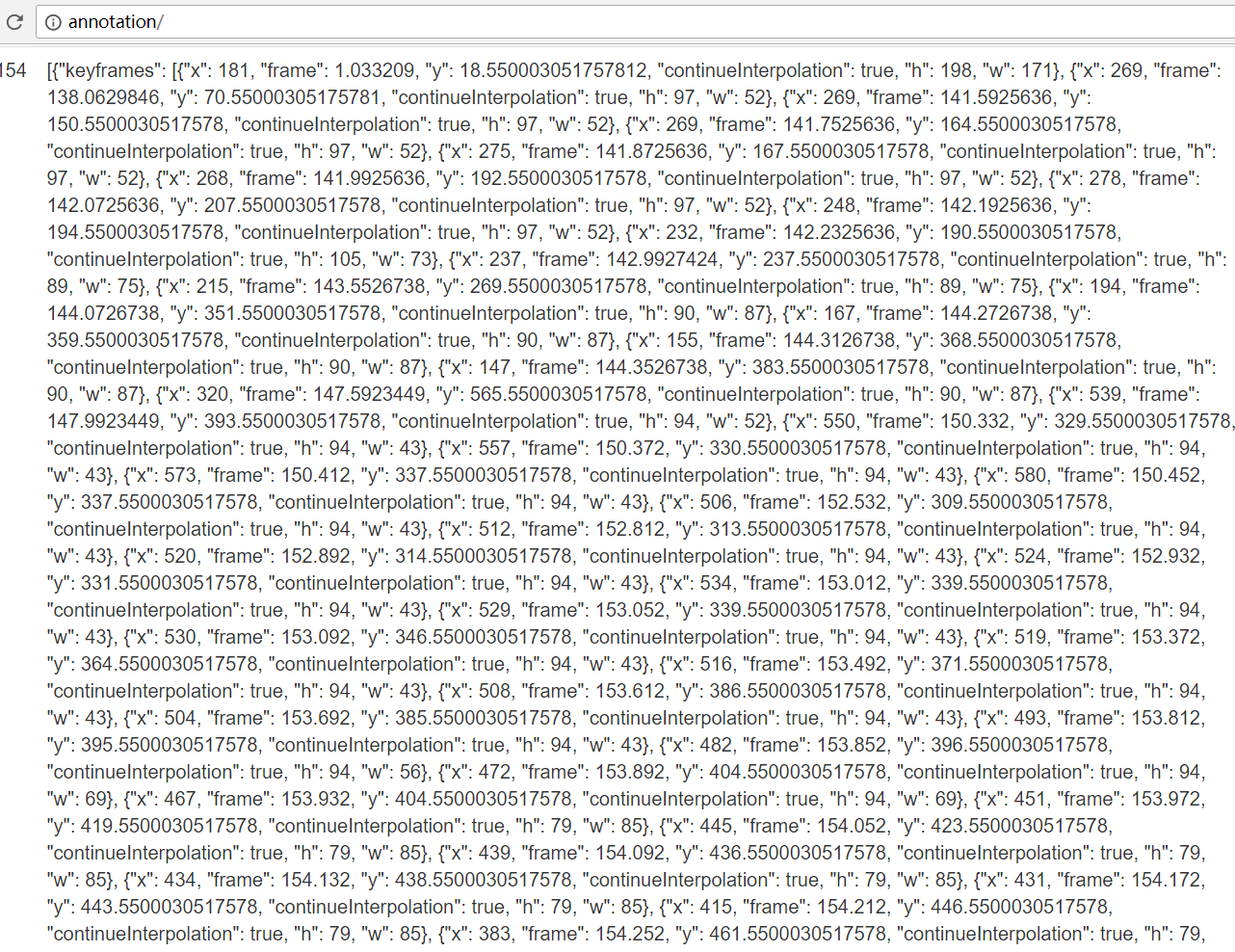
在标注页面的首页，可以看到后台添加的待标注视频的列表。如下图所示：



在每一条视频记录上，可以点击右边的view链接，进入单个视频的标注页面。通常，一个视频中有多个标签要进行标注，比如对于王者荣耀的电竞视频，单条视频中通常需要标注10人英雄人物。因此，我们的标注流程设计如下，先选中1个人物进行标注，从视频开始部分标注到视频结束部分，中间拖动标注框追踪被标注人物，必要时候需要缩放矩形框以适应人物的大小变化。从视频开始至结尾完成一个英雄人物的标注后，视频再返回至开始处进行下一个标签（或者标注对象）的标注。如下图所示：



标注完成的视频可以点击submit按钮进行结果的提交，这一数据会被保存在数据库中，同时可以通过web网页的方式显示，如下图所示：



1. **标注结果验证和导出**

在视频标注系统的verify子页面，审核员可以使用人工的方式直接在视频上进行审核，也可以使用程序根据标注的JSON结果生成截图，再进一步做验证。

验证过的视频可以点击verify按钮进行标志位的设置，标注结果可以导出成XML格式文件，具体如下：



标注完成的视频以及标注产生的Json/Xml文件将会被上传至学习平台进行神经网络模型的训练。

每一个游戏视频文件对应一个标注文件，标注文件包含了多条标注信息，可以理解为是一个标注信息的数组，数组中的每个元素描述了对应视频帧的标注信息，该数组中的每个元素包含了以下字段：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段** | **类型** | **描述** | | |
| Frame | Int | 原视频中的第几帧 | | |
| File\_name | string | 该帧视频转化成的图像文件的名字 | | |
| Height | Int | 图像的高度 | | |
| Width | Int | 图像的宽度 | | |
| Bbox（边界框） | Array（数组，数组中每个元素的字段参考右侧列表描述） | **字段** | **类型** | **描述** |
| x | Int | 边界框左上点的x坐标 |
| y | Int | 边界框左上点的y坐标 |
| Width | Int | 边界框的宽度 |
| Height | Int | 边界框的高度 |
| Label\_id | string | 标签ID |
| category | string | 所属分类 |

### 数据上传云平台

#### 数据表

样本上传表（game\_sample\_upload）

游戏样本上传表（将标注数据按照数据规范要求的格式从数据库导出到本地）

|  |  |  |  |
| --- | --- | --- | --- |
| game\_sample\_upload 游戏样本上传表 | | | |
| 字段 | 类型 | 是否必填 | 说明 |
| id | Number(11) | Y | 操作id，自增id |
| status | varchar2(20) | Y | 导出操作的状态 |
| username | varchar2(20) | Y | 操作人员用户名 |
| dataset\_id | Number(11) | Y | 此次导出数据的唯一编号 |
| rich\_text | Varchar2(2048) | Y | 内容数据文件名 |
| fields | Varchar2(2048) | Y | 信息记录文件名 |
| field\_description | Varchar2(2048) | Y | 字段说明文件名 |
| verf | Varchar2(2048) | Y | 数据校验文件名 |
| base\_dir | Varchar2(2048) | Y | 指定导出目录的根目录 |
| time | date | Y | 数据集导出操作的时间 |

#### 主要功能

参照《咪咕AI能力数据规范》，定义了数据上传云平台的文件名命名规则，以下分别对内容数据文件、信息纪录文件、字段说明文件、标注结果文件、数据校验文件、校验日志和样本数据上传表进行了详细定义。在模型样本管理界面上，点击某个样本视频记录后的“导出”按钮，依据数据规范导出数据到本地电脑，最后由运维人员手工拷贝到互娱的FTP前置服务器上。

1. 内容数据文件

该文件为咪咕游戏向AI云平台提供的符合数据规范要求的内容数据，包括游戏的训练视频、标注结果等数据。该内容数据需要提供给AI云平台中的学习子平台进行模型能力训练，数据全部为预处理后数据，源数据与预处理数据为1：1对应，即1个源数据会对应1个预处理后的数据。此时预处理后的数据命名规则如下，其中[项目名]中不能包含下划线“\_”，[重传序号]默认为两位，为数字型，第一次传输是采用“00”，每次重传该值递增1。

mg[hy\_gameAI\_Origin\_Video\_20170103\_00000001\_Preprocessing\_00.mp4](mailto:yy_mgAI_Origin_Image@1504529951_00000001_Preprocessing_00000001.jpg)

[子公司名]\_[项目名]\_ [Origin] \_[源文件类型]\_[源日期]\_[源自增标号（8位）]\_ [Preprocessing]\_[重传序号].文件类型

1. 信息记录文件
2. 当接口中有多个内容数据文件时，信息记录文件只有一个。
3. 信息记录文件记录内容数据中的各信息字段的取值，例如文件id、所属项目、生成日期等。
4. 信息记录文件统一采用Fields\_[重传序号].json或Fields\_[重传序号].csv命名。标识该接口的内容数据文件信息。
5. 字段说明文件

字段说明文件用以说明信息记录文件中各个字段的含义。信息字段说明文件统一命名为FieldDescription\_[重传序号].txt，为UTF-8编码的文本格式。

1. 标注结果文件

善跑的内容数据文件与标注结果文件为一一对应关系，标注结果文件命名规则如下，其中[重传序号]默认为两位，为数字型，第一次传输是采用“00”，每次重传该值递增1。

[mghy\_gameAI\_Origin\_Video\_20170103\_00000001\_](mailto:yy_mgAI_Origin_Image@1504529951_00000001_Preprocessing_00000001.jpg)Preprocessing\_00\_label\_00\_00.xml

[被标注的文件名称]\_ [标注日期]\_[label]\_[标注序号]\_ [重传序号].文件类型

1. 数据校验文件

当接口中有多个文件时，校验文件只有一个。标识该接口内所有数据校验信息，包含文件大小、生成日期、校验码等。该文件用于数据提供方向数据接收方提供的接口数据校验的动态校验规则。

校验文件命名规则如下，校验文件命名规则中的[子公司名]取值为mghy， [分类]字段取值为Origin或Preprocessing，根据校验文件所在目录属于预处理前（Origin）或预处理后（Preprocessing）决定；[日期]字段为该校验文件生成的时间，格式为YYYYMMDD。[重传序号]默认为两位，为数字型，第一次传输是采用“00”，每次重传该值递增1。重传校验文件中只记录重传数据的校验信息。

mg[hy\_gameAI\_Origin\_20170712\_00.verf](mailto:yy_mgAI_Origin_Image@1504529951_00000001.jpg)

[子公司名]\_[项目名]\_[分类] \_[日期] \_ [重传序号].文件类型

### 视频模型训练

算法人员远程登陆AI云平台的学习子平台手工进行视频模型训练。学习子平台支持基于Tensorflow和Caffe平台所搭建的深度学习网络，通过组件化的方式实现各种网络架构的灵活调用，易于实现不同规模样本集合下训练复杂度与模型精确度之间的折衷。平台也能够引入及验证新的网络架构。

学习平台通过运行python或c++的训练程序完成模型的训练及验证。以tensorflow平台上网络模型训练为例，其训练脚本主要应包含训练/验证集样本划分、样本读取及预处理、定义网络及超参数、模型训练与验证、模型保存与落日志等步骤。

模型的训练环境要求如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 运行环境 | 要求 | 备注 | 运行环境 |
| gcc版本 | 4.8及以上 |  | gcc版本 |
| Python版本 | 2.7.13以上，3.0以下 | python3.0及以上版本与2.x系列版本不兼容 | Python版本 |
| Cuda/cublas版本 | 推荐6.5及以上 |  | Cuda/cublas版本 |
| Cudnn版本 | 推荐5.0及以上 |  | Cudnn版本 |
| GPU驱动 | NV驱动 381.04 |  | GPU驱动 |
| Linux | 操作系统内核Centos7.2 |  |  |
| TensorFlow | 推荐1.2.0及以上 |  |  |

#### 数据集上传、样本预处理

视频标注工具产生的结果和原始长视频需要按照AI能力数据规范要求的格式存放到接口机的相应目录下，以方便大数据平台的数据拉取。

标注系统提供给训练平台的是原始的视频文件，在对应的标注文件中记录的是视频文件中角色及场景出现的时间及坐标。首先训练平台需要将其转换为模型在输入层期望的图像帧或帧组，及相应的以图像为基本单位的标注文件。

标签转化程序功能流程为：

1. 读取原始视频文件，及对应的原始视频标注文件；
2. 遍历原始视频标注文件Json串中的每一个元素。取出元素中的"frame"字段，它代表出现标注对象的帧在视频中对应的时戳。利用opencv等视频处理软件，从原始视频中抽取对应的帧，若这一帧之前并未被抽取过，则将该帧存为一幅jpg图像。其命名方式可为

[源视频ID]\_[帧号].jpg

1. 取出Json元素中的其它标注信息字段，包括角色ID("type")及位置坐标("x", "y", "w", "h")生成一条新的Json记录，写成一个文件。文件名应与相应的图片帧保持一致，即

[源视频ID]\_[帧号].json

1. 若通过frame字段的比对，发现对应的帧之前处理中已抽取过，则直接将该frame的标注信息(角色与坐标)，追加写入对应的.json文件中。

在样本被模型读入前，还应先进行缩放及有效性判断，即将源图片按照模型需要的图片宽高(对于AlexNet,模型输入层接收的是224\*224的图片)进行缩放，并且判断图片是RGB三通道的彩色图片(而不能是单通道的灰度图)。随后，还需进行样本图片的其它预处理，主要是对像素的RGB值进行云中心化及归一化，这将有助于后续训练的收敛速度。

#### 训练/验证集划分

训练集用来进行模型拟合；验证集为拟合后的模型提供无偏估计，可用来对训练得到模型的可泛化性进行判别，从而确认模型的有效性。在训练/验证样本的划分上，一种做法是K折交叉验证法(k-folds法),就是将样本随机乱序，然后均匀分成k份，轮流选择其中（k-1）份训练，剩余的一份做验证，计算预测误差平方和，最后把k次的预测误差平方和再做平均作为选择最优模型结构的依据。

#### 定义模型结构及参数

Tensorflow在脚本中用一个专门的函数来定义神经网络，英雄人物识别所用网络模型的前面部分，为五组抽取图像特征所用的卷积与池化层，其网络定义代码为

def inference(images):

"""Build the AlexNet model.

Args:

images: Images Tensor

Returns:

pool5: the last Tensor in the convolutional component of AlexNet.

parameters: a list of Tensors corresponding to the weights and biases of the

AlexNet model.

"""

parameters = []

# conv1

with tf.name\_scope('conv1') as scope:

kernel = tf.Variable(tf.truncated\_normal([11, 11, 3, 64], dtype=tf.float32,

stddev=1e-1), name='weights')

conv = tf.nn.conv2d(images, kernel, [1, 4, 4, 1], padding='SAME')

biases = tf.Variable(tf.constant(0.0, shape=[64], dtype=tf.float32),

trainable=True, name='biases')

bias = tf.nn.bias\_add(conv, biases)

conv1 = tf.nn.relu(bias, name=scope)

print\_activations(conv1)

parameters += [kernel, biases]

# lrn1

with tf.name\_scope('lrn1') as scope:

lrn1 = tf.nn.local\_response\_normalization(conv1,

alpha=1e-4,

beta=0.75,

depth\_radius=2,

bias=2.0)

# pool1

pool1 = tf.nn.max\_pool(lrn1,

ksize=[1, 3, 3, 1],

strides=[1, 2, 2, 1],

padding='VALID',

name='pool1')

print\_activations(pool1)

# conv2

with tf.name\_scope('conv2') as scope:

kernel = tf.Variable(tf.truncated\_normal([5, 5, 64, 192], dtype=tf.float32,

stddev=1e-1), name='weights')

conv = tf.nn.conv2d(pool1, kernel, [1, 1, 1, 1], padding='SAME')

biases = tf.Variable(tf.constant(0.0, shape=[192], dtype=tf.float32),

trainable=True, name='biases')

bias = tf.nn.bias\_add(conv, biases)

conv2 = tf.nn.relu(bias, name=scope)

parameters += [kernel, biases]

print\_activations(conv2)

# lrn2

with tf.name\_scope('lrn2') as scope:

lrn2 = tf.nn.local\_response\_normalization(conv2,

alpha=1e-4,

beta=0.75,

depth\_radius=2,

bias=2.0)

# pool2

pool2 = tf.nn.max\_pool(lrn2,

ksize=[1, 3, 3, 1],

strides=[1, 2, 2, 1],

padding='VALID',

name='pool2')

print\_activations(pool2)

# conv3

with tf.name\_scope('conv3') as scope:

kernel = tf.Variable(tf.truncated\_normal([3, 3, 192, 384],

dtype=tf.float32,

stddev=1e-1), name='weights')

conv = tf.nn.conv2d(pool2, kernel, [1, 1, 1, 1], padding='SAME')

biases = tf.Variable(tf.constant(0.0, shape=[384], dtype=tf.float32),

trainable=True, name='biases')

bias = tf.nn.bias\_add(conv, biases)

conv3 = tf.nn.relu(bias, name=scope)

parameters += [kernel, biases]

print\_activations(conv3)

# conv4

with tf.name\_scope('conv4') as scope:

kernel = tf.Variable(tf.truncated\_normal([3, 3, 384, 256],

dtype=tf.float32,

stddev=1e-1), name='weights')

conv = tf.nn.conv2d(conv3, kernel, [1, 1, 1, 1], padding='SAME')

biases = tf.Variable(tf.constant(0.0, shape=[256], dtype=tf.float32),

trainable=True, name='biases')

bias = tf.nn.bias\_add(conv, biases)

conv4 = tf.nn.relu(bias, name=scope)

parameters += [kernel, biases]

print\_activations(conv4)

# conv5

with tf.name\_scope('conv5') as scope:

kernel = tf.Variable(tf.truncated\_normal([3, 3, 256, 256],

dtype=tf.float32,

stddev=1e-1), name='weights')

conv = tf.nn.conv2d(conv4, kernel, [1, 1, 1, 1], padding='SAME')

biases = tf.Variable(tf.constant(0.0, shape=[256], dtype=tf.float32),

trainable=True, name='biases')

bias = tf.nn.bias\_add(conv, biases)

conv5 = tf.nn.relu(bias, name=scope)

parameters += [kernel, biases]

print\_activations(conv5)

# pool5

pool5 = tf.nn.max\_pool(conv5,

ksize=[1, 3, 3, 1],

strides=[1, 2, 2, 1],

padding='VALID',

name='pool5')

print\_activations(pool5)

return pool5, parameters

在最后的全连接层节点数目上，考虑到我们的模型应用需用来鉴别70个角色，需要全连接层提供较丰富的节点，以提供更精确的图像特征汇总。因此采用与cifar-100图像鉴别相同的全连接层参数，即两层全连接层，分别有4096和1024个节点。最后，输出层进行分类判别，输出一个数组结果，对应着训练图片是每一个角色的可能程度。

def full\_connect\_layers(conv\_layer\_in):  
 # Store layers weight & bias  
 weights = {  
 'wd1': tf.Variable(  
 tf.random\_normal([batch\_size \* 6 \* 6 \* 256, 4096], dtype=tf.float32, stddev=3e-1, name="dense1\_weights")),  
 'wd2': tf.Variable(tf.random\_normal([4096, 1024], dtype=tf.float32, stddev=3e-1, name="dense2\_weights")),  
 # 1024 inputs, 70 outputs (class prediction)  
 'out': tf.Variable(tf.random\_normal([1024, 70]))  
 }  
  
 biases = {  
 'bd1': tf.Variable(tf.random\_normal([4096])),  
 'bd2': tf.Variable(tf.random\_normal([1024])),  
 'out': tf.Variable(tf.random\_normal([70]))  
 }  
   
 # Fully connected layer  
 # Reshape conv output to fit fully connected layer input  
 fc1 = tf.reshape(conv\_layer\_in, [-1, weights['wd1'].get\_shape().as\_list()[0]])  
 fc1 = tf.add(tf.matmul(fc1, weights['wd1']), biases['bd1'])  
 fc1 = tf.nn.relu(fc1)  
 # Apply Dropout  
 fc1 = tf.nn.dropout(fc1, dropout)  
  
 fc2 = tf.add(tf.matmul(fc1, weights['wd2']), biases['bd2'])  
 fc2 = tf.nn.relu(fc2)  
 fc2 = tf.nn.dropout(fc2, dropout)  
  
 # Output, figure prediction  
 out = tf.add(tf.matmul(fc2, weights['out']), biases['out'])  
 return out

最终的输出进行softmax转换，将70种角色的可能程度值，归一为概率值，再与图像的标签值的独热编码向量求交叉熵损失，确定损失值，并定义反向传播时的优化函数。

# Define loss and optimizer

cost = tf.reduce\_mean(tf.nn.softmax\_cross\_entropy\_with\_logits(logits=pred, labels=Y))

optimizer = tf.train.AdamOptimizer(learning\_rate=learning\_rate).minimize(cost)

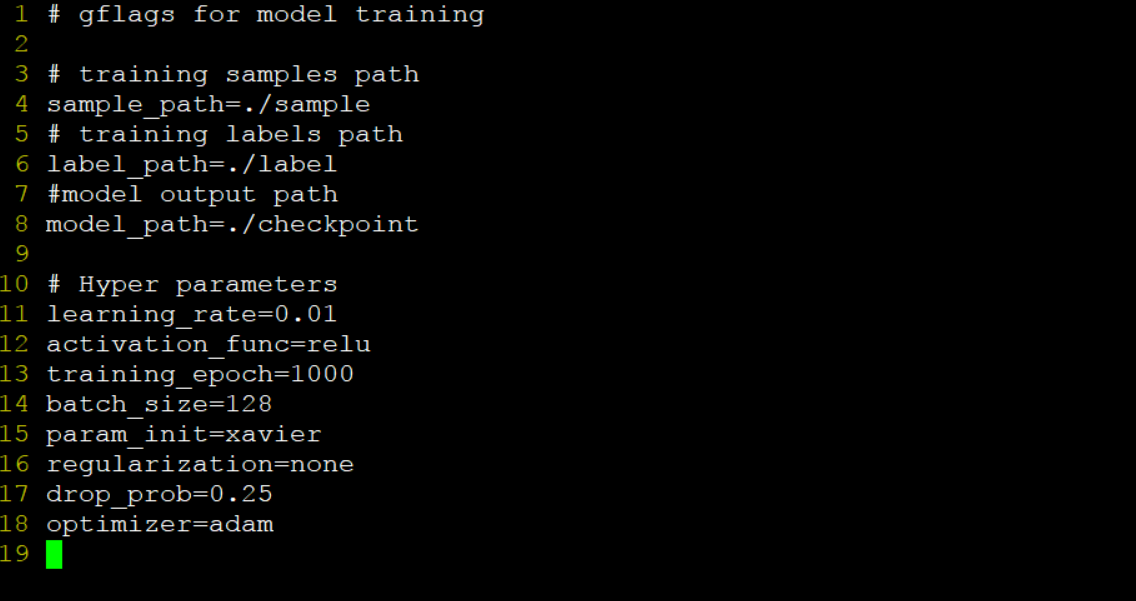
定义好网络的所有流程后，在调用脚本中，通过定义一个tensorflow session，调用session的run()函数来进行训练脚本的执行。

训练所需的参数（指的是训练样本及标签文件的读取路径，模型输出文件路径，训练的超参数配置等）使用google开源的gflag库完成配置。

学习平台所可配置的超参数包括而不限于以下：

|  |  |  |
| --- | --- | --- |
| **参数名称** | **可取值** | **默认值** |
| 学习率 | (0, 1) | 0.01，并随训练周期衰减 |
| 激活函数 | Sigmoid, Tanh, ReLU, Leaky ReLU, PReLU, SeLU | ReLU |
| 训练轮数(epoch) | 实验中调优 |  |
| 每批次样本数(batch size) | 实验中调优 |  |
| 参数初始化方法 | 高斯、有偏高斯、哈维尔(Xavier) | 哈维尔初始化 |
| 正则化 | L1、L2、不启用 | 不启用 |
| dropout | 实验中调优 |  |
| 优化算法 | SGD、Momentum、RMSProp、Adam | Adam |

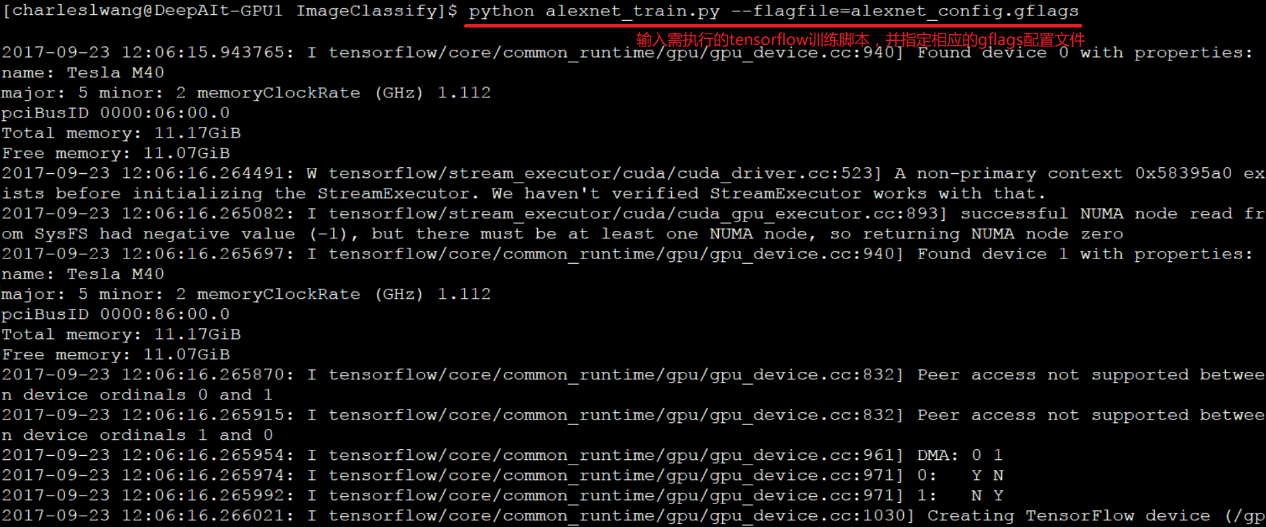
若无特殊训练需求，一般可以使用系统预存的默认配置gflags文件。管理员也可在登录训练系统主机后通过vim等编辑器进行自定义配置gflags文件的创建与修改。一个gflags文件的编辑界面如下图所示。



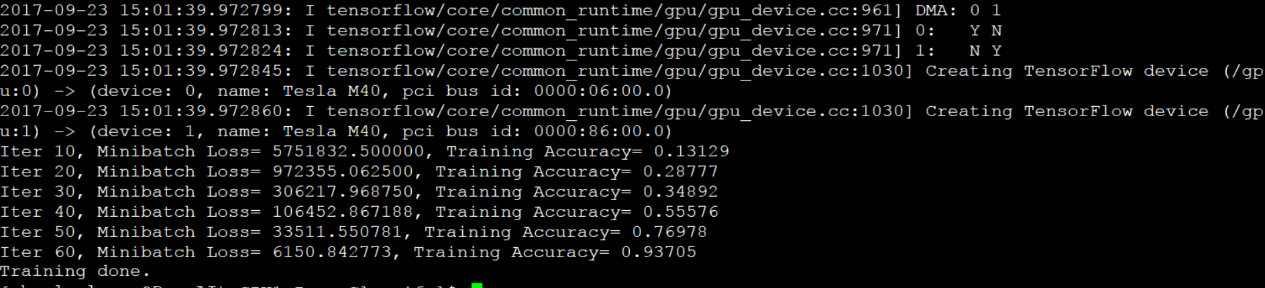
#### 模型训练过程

训练过程的执行通过命令行键入训练脚本运行命令，并在命令行参数中指定加载的gflags配置文件。一个基于python语言的tensorflow模型训练脚本的执行示例如下。

python alexnet\_train.py --flagfile=alexnet\_config.gflags



在进行模型训练和验证时，能够将相关数据显示在控制台界面上，以便用户直观地监测训练进度及效果。网络训练阶段，反馈数据主要包括当前迭代轮数、训练图片批次读取进度、当前训练批次的总损失值、当前训练批次上的准确率等（如下图所示）。通过观测训练集的损失值及准确率，模型训练操作者可以判断训练算法是否有效，是否已收敛。



在模型验证方面，如前所述，训练系统利用k-folds方法实现交叉验证，充分检验模型的有效性，可在模型训练轮次完成后进行验证，输出每一次验证集图片的准确率，以便监测模型是否出现过拟合现象。

### 训练结果发布

#### 数据表

游戏模型版本表(game\_model\_version)

#### 主要功能

模型的训练与验证过程结束后，最终可生成可供使用的tensorflow checkpoint模型文件和此次训练的日志文件。其中模型文件包括：

|  |  |
| --- | --- |
| **文件名** | **描述** |
| checkpoint文件 | 当前最新模型文件信息 |
| .meta文件 | 保存了当前图结构 |
| .index文件 | 保存了当前参数名 |
| .data文件 | 保存了当前参数值 |

在模型训练完成生成模型文件后，训练脚本会将校验合格的模型文件按照AI能力数据规范要求放置到接口机的相应目录下，同时训练脚本也会产生日志文件。模型发布包含两种情况：新模型发布和新版本发布。新模型发布，系统会自动分配模型id，版本号默认为1；新版本发布，需要明确指定模型id，版本号会自动加1。模型文件存储路径格式规范“<接口机相应目录>/模型id/模型版本号/模型相关文件”，算法人员手工将模型文件、训练日志从AI云平台下载到AI前置服务平台上。

训练产生的日志文件为文本格式，内容为模型运行过程中脚本代码的日志语句所写入，字段包括

|  |  |
| --- | --- |
| **字段含义** | **说明** |
| 日期时间 | 写入日志的时间点，精确至毫秒 |
| 日志内容等级 | 取值范围包括错误(E)、告警(W)、提示(I) |
| 落日志的语句 | 写本条日志的语句位置，精确到某个源文件的某一行 |
| 日志内容 | 写入的具体日志内容 |
| Epoch | 目前进行了多少个epoch的训练（1个epoch等于使用训练集中的全部样本训练一次） |
| iteration | 目前进行了多少个iteration的训练（1个iteration等于使用batchsize个样本训练一次） |
| batchsize | SGD训练时每一批次数据的大小 |
| Score | 每一批次的数据训练之后在验证集上测试模型得到的成绩 |

## 视频模型使用

### 源视频同步

#### 数据表

游戏视频表（game\_video）

#### 主要功能

游戏服务端每小时将增量的游戏视频数据通过互娱内部的kafka消息队列同步给互娱大数据平台，大数据平台接收后保存入游戏视频表，再通过视频地址下载视频源文件到本地服务器库作为视频模型输入，数据为JSON格式，主要字段如下：

|  |  |
| --- | --- |
| 功能 | 咪咕游戏向大数据平台同步游戏视频记录 |
| Topic | t\_mggame\_to\_bigdata\_video |
| 数据表 | 游戏视频表（game\_video） |
| content\_id | 游戏编码 |
| video\_id | 视频编码 |
| link\_url | 下载地址 |

### 视频模型使用

#### 数据表

游戏模型表（game\_model）、游戏模型版本表(game\_model\_version)、游戏在线使用日志表（game\_api\_log）、游戏短视频表(game\_short\_video)、游戏视频标签对应表(game\_video\_label)

#### AI模型初始化

前置服务平台上的AI交互子平台上会运行游戏AI模型，对外提供REST API服务。咪咕游戏服务端通过互娱大数据平台调用AI前置服务平台的业务调度模块来使用REST API接口发起业务处理请求。API接口分为游戏主体检测、关键场景识别、精彩短视频生成共3个。

AI模型接收到业务处理请求后对视频文件进行预处理，将原始视频转化为模型的输入，再接下来用模型进行分析，得到初始的分析结果，然后将分析结果进行综合与归纳，得到符合返回需求的处理结果，最后将处理结果进行序列化，再根据业务处理请求中携带的callback\_url，将序列化的处理结果返回至AI能力客户端。每次API调用都会产生相应的log，存入模型API调用日志表。

模型运行的环境要求如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 运行环境 | 要求 | 备注 | 运行环境 |
| gcc版本 | 4.8及以上 |  | gcc版本 |
| Python版本 | 2.7.13以上，3.0以下 | python3.0及以上版本与2.x系列版本不兼容 | Python版本 |
| Cuda/cublas版本 | 推荐6.5及以上 |  | Cuda/cublas版本 |
| Cudnn版本 | 推荐5.0及以上 |  | Cudnn版本 |
| GPU驱动 | NV驱动 381.04 |  | GPU驱动 |
| Linux | 操作系统内核Centos7.2 |  |  |
| TensorFlow | 推荐1.2.0及以上 |  |  |

模型服务在完成初始化工作后，就开始进入事件的监听模式。侦听来自客户端的http请求。而在初始化阶段，最主要的工作就是使用tensorflow框架按照配置文件中模型版本载入相应的模型文件，初始化好网络参数。配置文件如下所示：

[

{

"game\_id": 1,

"service\_id": 1,

"model\_name": "英雄人物",

"model\_version": "1.0.0",

"file\_name": "mghy\_mgAI\_Origin\_model\_3\_20171003\_00000001\_01",

"file\_size": "500M",

"file\_path":"file:///data/mghy\_mgAI\_Origin\_model\_3\_20171003\_00000001\_01",

"file\_time": "2017-10-03 10:00:00"

},

......

]

配置文件的另外一层意义是，在模型管理系统上，当点击模型发布按钮去发布一个新模型时，会将对应版本的模型文件信息写入到此配置文件中。Tensorflow程序被重启从而重新载入配置文件，当tensorflow重新初始化时，就会按照配置文件中模型版本载入相应的模型文件。

#### 特定主体检测

1. **输入请求**

对于游戏特定主体检测，REST的URL为http://migu\_url//game/kingofglory/video/hero\_detect/v1/，使用POST方法调用REST API时的输入参数设计如下：

|  |  |  |
| --- | --- | --- |
| REST API调用所传递参数 | 类型 | 描述 |
| video\_url | string | 游戏视频的存储路径 |
| task\_id | string | 此次调用任务的唯一编号 |
| callback\_url | string | 游戏AI引擎回调咪咕游戏的REST URL |

REST服务的客户端，向

<http://migu_url//game/kingofglory/video/hero_detect/v1>发起一个POST请求，携带的参数是string类型的video\_url、task\_id、callback\_url，其中，video\_url是该视频的存储路径，task\_id和 callback\_id可使AI模型服务在通过模型完成视频分析之后，能够将结果以异步回调的方式返回给调用端。

1. **视频文件预处理**

通常情况下，一段流畅播放的游戏视频文件，其帧率在25帧/秒以上。这意味着邻近两帧之间的图像仅会有很微小的变化，从信息论的角度看，邻近两帧或数帧的图像信息是极度冗余的。因此，我们需要对原始视频文件，作合适的采样等操作，以使模型分析准确性不下降的基础上，更好地降低模型处理的复杂度，从而降低耗时。

对于游戏特定主体检测，可在待检视频中每秒抽取1~2帧。对于抽取好的帧图片文件，需要统一其文件名格式，如

[源视频ID]\_[帧号].jpg

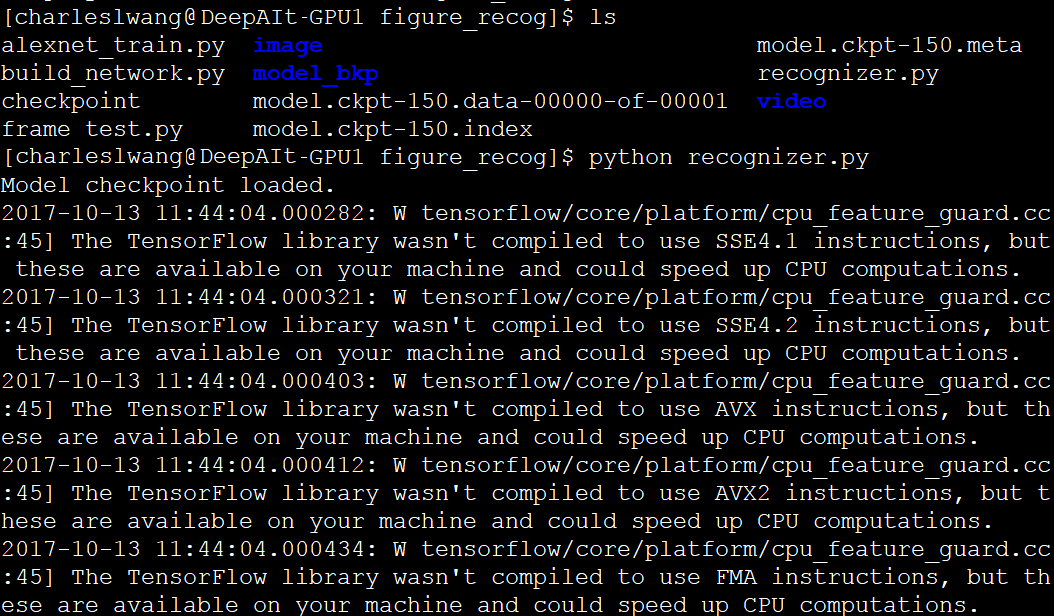
以便在模型检测后对原始结果数据进行后续处理。

另外，帧图片文件在送入模型之前，应对其作进一步的预处理。应对图片文件进行缩放，以适配模型输入端对输入图像数据的要求，另外，需对原始像素信息进行去中心化与归一化，即将每个像素[0, 255]的原始RGB值，线性转换为[-0.5, 0.5]或[-1, 1]的值(根据模型的输入要求选择转换方式)。

1. **模型进行分析**

在这个阶段，游戏特定主体检测模型文件被运行的分析程序所读取，读取通过tensorflow的Saver对象来完成。

def load\_model(sess):  
 saver = tf.train.Saver()  
 record = tf.train.get\_checkpoint\_state('.')  
 if record and record.model\_checkpoint\_path:  
 saver.restore(sess, record.model\_checkpoint\_path)  
 print('Model checkpoint loaded.')  
 return True  
 else  
 print('Model not found.')  
 return False



程序成功读取模型后，用模型顺次接收预处理后的视频图像各帧数据，经过深度学习二维卷积神经网络的前向推导过程，得到每帧中检测出了哪些游戏主体的初始检测结果数据。数据的组织方式为

帧文件名1 游戏主体ID1,游戏主体ID2,…

帧文件名2 游戏主体ID3,游戏主体ID4,…

…

1. **分析结果后处理**

模型分析视频图像，给出的初始检测结果数据是以帧为索引的，即结果数据组织方式为每个截取的帧中，包含了哪些特定主体。还需将结果转化为接口需要的数据组织方式，即以每个特定主体为索引，记录每个特定主体，都出现在哪些帧中。对于在连续帧的出现，记录其出现的开始和结束帧。

1. **结果封装**

结果封装的工作是将处理好的分析结果进行序列化。这里我们采用目前主机间进行RESTful交互时主流的Json编码方式。Json与XML类似，都是用结构化方法来标记数据，而Json在可读性、可扩展性不输XML的情况下，由于取消了严格的标签定义，能够达到更高的有效数据率，网络传输时能有更高的效率。

AI模型服务把返回结果的数据描述定义为hero\_items，则其描述为：

|  |  |  |  |
| --- | --- | --- | --- |
| **hero\_item字段** | | **类型** | **字段描述** |
| hero\_id | | String | 人物编号 |
| task\_id | | String | 任务编号 |
| emergences | start\_frame | int | 起始帧 |
| end\_frame | int | 结束帧 |
| start\_time | int | 起始时间 |
| end\_time | int | 结束时间 |

一个转换了的Json结构示例为：

{

"task\_id"：10001,

"hero\_items": [{

"hero\_id": "001",

"emergences": [{

"start\_frame": 1000,

"end\_frame": 2000,

"start\_time": 40,

"end\_time": 80

}],

"hero\_id": "002",

"emergences": [{

"start\_frame": 3000,

"end\_frame": 4000,

"start\_time": 120,

"end\_time": 160

}]

}]

}

1. **异步AI分析结果返回**

如前文所述，callback\_url和task\_id是使游戏AI引擎在处理完视频后能够进行回调用的。因为游戏AI引擎处理一个视频（比如30分钟时长的视频）需要较长的时间，所以针对AI模型服务的REST调用是一个异步操作，当视频真正被处理完成之后，AI模型服务会发起一个REST调用至callback\_url，需要的内容就来自callback\_url和task\_id。我们可以把这个回调理解为之前REST调用的返回值。使用<http://callback_url/> 进行post方法的回调时，输入参数定义如下：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **REST API调用所传递参数** | **类型** | **描述** | | |
| hero\_items | Json Array  （数组中的每个元素定义参考右侧列表） | **hero\_item字段** | **类型** | **字段描述** |
| hero\_id | string | 人物编号 |
| start\_frame | int | 起始帧 |
| end\_frame | int | 结束帧 |
| start\_time | int | 起始时间 |
| end\_time | int | 结束时间 |
| task\_id | string | 此次调用任务的唯一编号 | | |

AI模型服务回调咪咕游戏地址，内容包含了4位响应码和一组英雄人物的Json数组（上述定义中的hero\_items），这个数组是由游戏AI引擎分析完整个视频得到的，数组中的每个元素是一个包含5个字段的结构体（上述定义中的hero\_item）：人物的名字、所处的起始帧、结束帧、起始时间、结束时间。

响应码设计如下：

|  |  |  |
| --- | --- | --- |
| 响应码 | 类型 | 描述 |
| code | String | 0000英雄人物识别成功；  2011 输入视频无法下载  2012 视频无法打开进行分析  2013 AI能力内部逻辑异常  2014 没有发现待检测对象  2015 AI能力分析超时 |

#### 关键场景识别

1. **输入请求**

对于游戏关键场景提取，REST的URL为http://migu\_url//game/kingofglory/video/ specified\_scene /v1/，使用POST方法来调用REST API时的输入参数设计如下：

|  |  |  |
| --- | --- | --- |
| REST API调用所传递参数 | 类型 | 描述 |
| video\_url | string | 游戏视频的存储路径 |
| task\_id | string | 此次调用任务的唯一编号 |
| callback\_url | string | 游戏AI引擎回调咪咕游戏的REST URL |

REST服务的客户端，向

<http://migu_url//game/kingofglory/video/specified_scene/v1>发起一个POST请求，携带的参数是string类型的video\_url、task\_id、callback\_url，其中，video\_url是该视频的存储路径，task\_id和 callback\_id可使AI模型服务在通过模型完成视频分析之后，能够将结果以异步回调的方式返回给调用端。

1. **视频文件预处理**

对于游戏关键场景提取，需要衔接性强的，较连续的视频图像帧，因此图像帧抽取频率需较大，至少每秒应抽取5帧以上。对于抽取好的帧图片文件，需要统一其文件名格式，如

[源视频ID]\_[帧号].jpg

以便在模型检测后对原始结果数据进行后续处理。

另外，帧图片文件在送入模型之前，应对其作进一步的预处理。应对图片文件进行缩放，以适配模型输入端对输入图像数据的要求，另外，需对原始像素信息进行去中心化与归一化，即将每个像素[0, 255]的原始RGB值，线性转换为[-0.5, 0.5]或[-1, 1]的值(根据模型的输入要求选择转换方式)。

1. **模型进行分析**

在这个阶段，游戏关键场景提取模型顺次接收预处理后的视频图像多帧数据组，经过深度学习三维卷积神经网络的前向推导过程，得到每帧中检测出了哪些游戏场景的初始检测结果数据。数据的组织方式为

起始帧文件名1\_终止帧文件名1 关键场景ID1

起始帧文件名2\_终止帧文件名2 关键场景ID2

…

1. **分析结果封装返回**

模型分析视频图像，给出的初始检测结果数据是以帧为索引的，即结果数据组织方式为每个截取的帧中，包含了哪些关键场景。还需将结果转化为接口需要的数据组织方式，即以每个场景为索引，记录每个场景都出现在哪些片段(图像帧组)中，记录其出现的开始和结束帧。

结果封装同样采用Json编码方式。我们把返回结果的数据描述定义为specified\_scene\_items，则其描述为：

|  |  |  |  |
| --- | --- | --- | --- |
| **specified\_scene\_item字段** | | **类型** | **字段描述** |
| specified\_scene\_id | | string | 场景编号 |
| task\_id | | String | 任务编号 |
| emergences | start\_frame | int | 起始帧 |
| end\_frame | int | 结束帧 |
| start\_time | int | 起始时间 |
| end\_time | int | 结束时间 |

Json结构示例为：

{

"task\_id"：10001,

"specified\_scene\_items": [{

"specified\_scene\_id": "001",

"emergences": [{

"start\_frame": 1600,

"end\_frame": 1675,

"start\_time": 64,

"end\_time": 67

}],

"specified\_scene\_id": "002",

"emergences": [{

"start\_frame": 2000,

"end\_frame": 2050,

"start\_time": 80,

"end\_time": 82

}]

}]

}

1. **异步AI分析结果返回**

当视频处理完成之后，游戏AI模型服务会发起一个REST调用至callback\_url，需要的内容就来自callback\_url和task\_id。使用<http://callback_url/> 进行post方法的回调时，输入参数定义如下：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **REST API调用所传递参数** | **类型** | **描述** | | |
| specified\_scene\_items | Json Array  （数组中的每个元素定义参考右侧列表） | **specified\_scene\_item字段** | **类型** | **字段描述** |
| specified\_scene\_id | String | 场景编号 |
| start\_frame | int | 起始帧 |
| end\_frame | int | 结束帧 |
| start\_time | int | 起始时间 |
| end\_time | int | 结束时间 |
| task\_id | string | 此次调用任务的唯一编号 | | |
| Code | String | 响应码 | | |

AI模型服务回调咪咕游戏地址，内容包含了一个4位数的响应码和一组关键场景的Json数组（上述定义中的specified\_scene\_items），这个数组是由游戏AI引擎分析完整个视频得到的，数组中的每个元素是一个包含5个字段的结构体（上述定义中的specified\_scene\_item）：场景编号、所处的起始帧、结束帧、起始时间、结束时间。

响应码设计如下：

|  |  |  |
| --- | --- | --- |
| 响应码 | 类型 | 描述 |
| code | String | 0000英雄人物识别成功；  2011 输入视频无法下载  2012 视频无法打开进行分析  2013 AI能力内部逻辑异常  2014 没有发现待检测对象  2015 AI能力分析超时 |

#### 精彩短视频生成

针对精彩短视频生成的AI能力调用，是建立在对关键场景的识别基础上的。因此，AI能力调用逻辑和返回值与3.2.2.4节的步骤和逻辑都是一样的，有一点不同的就是，精彩视频生成后需要由互娱大数据平台将生成的精彩短视频上传至咪咕云并获取vid、播放地址，再把vid、播放地址返回至咪咕游戏服务端。

结果封装同样采用Json编码方式。我们把返回结果的数据描述定义为specified\_scene\_items，则其描述为：

|  |  |  |  |
| --- | --- | --- | --- |
| **specified\_scene\_item字段** | | **类型** | **字段描述** |
| specified\_scene\_id | | string | 场景编号 |
| short\_video\_id | | string | 短视频编号 |
| task\_id | | String | 任务编号 |
| emergences | start\_frame | int | 起始帧 |
| end\_frame | int | 结束帧 |
| start\_time | int | 起始时间 |
| end\_time | int | 结束时间 |
| highlight\_score | double | 场景得分 |
| cover | String | 封面截图 |
| Short\_video\_url | | list | 精彩短视频在FTP服务器上的路径，子目录以task\_id命名 |

Json结构示例为：

{

"task\_id"：10001,

"short\_video\_url"："/data/10001/",

"specified\_scene\_items": [{

"specified\_scene\_id": "001",

"short\_video\_id":"01.MP4"

"emergences": [{

"start\_frame": 1600,

"end\_frame": 1675,

"start\_time": 64,

"end\_time": 67,

"highlight\_score":3,

"cover":"01.jpg"

}],

"specified\_scene\_id": "002",

"short\_video\_id":"02.MP4"

"emergences": [{

"start\_frame": 2000,

"end\_frame": 2050,

"start\_time": 80,

"end\_time": 82

"highlight\_score":3,

"cover":"01.jpg"

}]

}]

}

当视频处理完成之后，游戏AI模型服务会发起一个REST调用至callback\_url，需要的内容就来自callback\_url和task\_id。使用<http://callback_url/> 进行post方法的回调时，输入参数定义如下：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **REST API调用所传递参数** | **类型** | **描述** | | |
| highlight\_scene\_items | Array  （数组中的每个元素定义参考右侧列表） | **highlight\_scene\_item字段** | **类型** | **字段描述** |
| specified\_scene\_id | String | 精彩场景编号 |
| short\_video\_id | string | 短视频名称 |
| cover | string | 游戏封面照片 |
| highlight\_score | int | 精彩度得分 |
| start\_frame | int | 在原视频中的起始帧 |
| end\_frame | int | 在原视频中的结束帧 |
| start\_time | String | 在原视频中的起始时间 |
| end\_time | String | 在原视频中的结束时间 |
| task\_id | string | 此次调用任务的唯一编号 | | |
| short\_video\_url | string | 精彩短视频在FTP服务器上的路径，子目录以task\_id命名 | | |

AI模型服务回调咪咕游戏地址，内容包含了4位数的响应码和一组关键场景的Json数组（上述定义中的specified\_scene\_items），这个数组是由游戏AI引擎分析完整个视频得到的，数组中的每个元素是一个包含5个字段的结构体（上述定义中的specified\_scene\_item）：场景编号、所处的起始帧、结束帧、起始时间、结束时间、咪咕云视频vid、location、play\_url、该场景的封面照片。

响应码设计如下：

|  |  |  |
| --- | --- | --- |
| 响应码 | 类型 | 描述 |
| code | String | 0000英雄人物识别成功；  2011 输入视频无法下载  2012 视频无法打开进行分析  2013 AI能力内部逻辑异常  2014 没有发现待检测对象  2015 AI能力分析超时  2020 创建上传咪咕云的task出错  2021 上传视频出错  2022 产生短视频至本地文件系统时出错  2023 上传视频超时 |

### 模型结果输出

#### 数据表

游戏视频表（game\_video）、游戏短视频表（game\_short\_video）、游戏视频标签对应表（game\_video\_label）

#### 主要功能

互娱大数据平台调用AI前置平台上的模型服务成功后，获取源视频标签，并将标签通过互娱内部的Kafka消息队列实时同步给咪咕游戏服务端，数据为JSON格式，主要字段如下：

|  |  |
| --- | --- |
| 功能 | 大数据平台向咪咕游戏同步长视频标签 |
| Topic | t\_bigdata\_to\_mggame\_video\_label |
| 数据表 | 游戏视频表（game\_video）、游戏视频标签对应表（game\_video\_label） |
| video\_id | 长视频编码 |
| game\_person\_label | 视频角色标签ID列表，英文逗号分割 |
| game\_scence\_label | 视频场景标签ID列表，英文逗号分割 |

互娱大数据平台调用AI前置平台上的模型服务成功后，将源视频拆分成若干个短视频并获取视频标签；互娱大数据平台将短视频上传咪咕云后，再将短视频的标签、存储地址通过互娱内部的Kafka消息队列实时同步给咪咕游戏服务端，数据为JSON格式，主要字段如下：

|  |  |
| --- | --- |
| 功能 | 大数据平台向咪咕游戏同步短视频和标签 |
| Topic | t\_bigdata\_to\_mggame\_short\_video |
| 数据表 | 游戏短视频表（game\_short\_video）、游戏视频标签对应表（game\_video\_label） |
| video\_id | 长视频编码 |
| vid | 咪咕云视频ID |
| duration | 短视频时长（秒），可为空 |
| video\_resolution | 视频分辨率，宽\*高，可为空 |
| link\_url | 下载地址 |
| person\_labels | 视频角色标签ID列表，英文逗号分割 |
| scence\_labels | 视频场景标签ID列表，英文逗号分割 |

咪咕云的视频上传、下载等接口由咪咕视讯提供，具体如下：



### 模型反馈修正

#### 数据表

游戏视频表（game\_video）、游戏短视频表（game\_short\_video）、游戏视频标签对应表（game\_video\_label）、游戏视频报错表（game\_video\_error）

#### 主要功能

咪咕游戏运营人员在审核AI模型处理的视频时，如发现视频与标签对应关系错误，可以为视频选择正确标签并向互娱大数据平台报错，错误视频和标签由咪咕游戏服务端通过互娱内部的Kafka消息队列实时同步给互娱大数据平台，错误视频可作为新的样本进行重新标注，并上传到AI云平台上对模型进行重新训练。数据为JSON格式，主要字段如下：

|  |  |
| --- | --- |
| 功能 | 咪咕游戏向大数据平台同步短视频出错记录 |
| Topic | t\_mggame\_to\_bigdata\_short\_video\_error |
| 数据表 | 游戏视频报错表（game\_video\_error） |
| vid | 短视频编码 |
| video\_id | 视频编码 |
| person\_labels | 视频角色标签ID列表，英文逗号分割 |
| scence\_labels | 视频场景标签ID列表，英文逗号分割 |
| error\_describe | 错误描述 |

|  |  |
| --- | --- |
| 功能 | 咪咕游戏向大数据平台同步长视频出错记录 |
| Topic | t\_mggame\_to\_bigdata\_video\_error |
| 数据表 | 游戏视频报错表（game\_video\_error） |
| video\_id | 长视频编码 |
| person\_labels | 视频角色标签ID列表，英文逗号分割 |
| scence\_labels | 视频场景标签ID列表，英文逗号分割 |
| error\_describe | 错误描述 |

## 咪咕游戏配套

### 数据表

1. **视频角色标签表game\_video\_role\_label**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 字段类型 | 表字典描述 | 是否必填 |
| ID | NUMBER(12) | 主键 | 是 |
| content\_id | VARCHAR2(20) | 游戏对应内容ID | 是 |
| label \_name | Varchar(128) | 标签名称 | 是 |
| role\_type | Varchar(128) | 角色类型 | 否 |
| sex | NUMBER (1) | 性别: 0其他，1男，2女 | 是 |
| role\_screenshot\_id | NUMBER(11) | 角色截图 | 否 |
| Create\_TIME | DATE | 创建时间 | 否 |
| create\_USER | VARCHAR2(100) | 创建人 | 否 |
| UPDATE\_TIME | DATE | 更新时间 | 否 |
| UPDATE\_USER | VARCHAR2(100) | 更新人 | 否 |

1. **游戏场景标签表game\_game\_** **scene\_label**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 字段类型 | 表字典描述 | 是否必填 |
| ID | NUMBER(12) | 主键 | 是 |
| content\_id | VARCHAR2(20) | 游戏对应内容ID | 是 |
| game\_scence | Varchar(128) | 游戏场景 | 是 |
| scence\_score | NUMBER(2) | 场景得分 | 否 |
| scence\_describe | Varchar(128) | 场景描述 | 否 |
| scence\_ screenshot\_id | NUMBER(11) | 场景截图 | 否 |
| create\_TIME | DATE | 创建时间 | 否 |
| create\_USER | VARCHAR2(100) | 创建人 | 否 |
| UPDATE\_TIME | DATE | 更新时间 | 否 |
| UPDATE\_USER | VARCHAR2(100) | 更新人 | 否 |

1. **短视频报错表game\_short\_video\_error**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 字段类型 | 表字典描述 | 是否必填 |
| ID | NUMBER(12) | 主键 | 是 |
| VID | VARCHAR2(255) | 咪咕云视频ID | 是 |
| VIDEO\_ID | NUMBER(12) | 视频ID | 是 |
| error\_describe | Varchar(1024) | 报错描述 | 是 |
| game\_scence\_label | Varchar(1024) | 报错内容，角色标签id | 否 |
| game\_role\_label | Varchar(1024) | 报错内容，场景标签id | 否 |
| create\_TIME | DATE | 创建时间 | 否 |
| create\_USER | VARCHAR2(100) | 创建人 | 否 |
| UPDATE\_TIME | DATE | 更新时间 | 否 |
| UPDATE\_USER | VARCHAR2(100) | 更新人 | 否 |

1. **短视频表game\_short\_video**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 字段类型 | 表字典描述 | 是否必填 |
| ID | NUMBER(12) | 主键 | 是 |
| VID | VARCHAR2(255) | 咪咕云视频ID | 是 |
| VIDEO\_ID | NUMBER(12) | 原视频ID | 是 |
| VIDEO\_RESOLUTION | VARCHAR2(20) | 视频分辨率，宽\*高 | 否 |
| game\_scence\_label | VARCHAR2(120) | 游戏场景标签id,多个标签用id分割 | 否 |
| game\_role\_label | VARCHAR2(120) | 游戏角色标签id, 多个标签用id分割 | 否 |
| DURATION | NUMBER | 时长 | 否 |
| LINK\_URL | VARCHAR2(1024) | 咪咕云视频播放地址 | 是 |
| Merge\_VIDEO\_ID | NUMBER(12) | 合并后对应的视频id | 否 |
| create\_TIME | DATE | 创建时间 | 否 |
| create\_USER | VARCHAR2(100) | 创建人 | 否 |
| UPDATE\_TIME | DATE | 更新时间 | 否 |
| UPDATE\_USER | VARCHAR2(100) | 更新人 | 否 |

1. **视频同步记录表game\_video\_** **synchro\_record**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 字段类型 | 表字典描述 | 是否必填 |
| ID | NUMBER(12) | 主键 | 是 |
| VIDEO\_ID | NUMBER(12) | 视频ID，与 | 是 |
| synchro\_result\_type | NUMBER(1) | 1:成功,2:失败 | 是 |
| synchro\_describe | clob | 同步结果描述 | 是 |
| create\_TIME | DATE | 创建时间 | 否 |

1. **标签同步错误表game\_label\_ synchro\_error**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 字段类型 | 表字典描述 | 是否必填 |
| ID | NUMBER(12) | 主键 | 是 |
| label\_ID | NUMBER(12) | 视频ID | 是 |
| Label\_type | NUMBER(1) | 1:角色标签,2：角色标签 | 是 |
| error\_describe | clob | 报错描述 | 是 |
| create\_TIME | DATE | 创建时间 | 否 |

1. **视频表game\_video\_info**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 字段类型 | 表字典描述 | 是否必填 |
| ID | NUMBER(11) | 主键ID | N |
| VIDEO\_NAME | VARCHAR2(70) | 名称 | N |
| LINK\_URL | VARCHAR2(1024) | 视频链接地址 | N |
| USER\_ID | NUMBER(11) | 用户ID | N |
| VIDEO\_STATUS | VARCHAR2(2) | 视频状态（1：在线、2：下线、3：上传中、4：用户删除） | N |
| VIDEO\_TYPE | VARCHAR2(2) | 视频类型（1:攻略、2:赛事、3:明星战队、4:游戏风云、5:大神专辑） | N |
| VIDEO\_SOURCE | VARCHAR2(2) | 采编状态（1：非采编（手机端上传）、2：采编（后台上传视频）、3：三方渠道引入、4：直播回放） | N |
| DURATION | NUMBER | 时长 | N |
| RECORD\_COMPLETED\_TIME | DATE | 视频完成时间，格式：yyyy-mm-ddhh:mm:ss | Y |
| VIDEO\_COVER | VARCHAR2(200) | 视频封面地址 | N |
| SERVICE\_ID | VARCHAR2(12) | 业务ID | Y |
| SERVICE\_NAME | VARCHAR2(255) | 业务名称 | Y |
| MATCH\_ID | NUMBER(11) | 赛事ID | Y |
| MATCH\_NAME | VARCHAR2(50) | 赛事名称 | Y |
| CREATE\_USER | VARCHAR2(20) | 创建者 | Y |
| CREATE\_TIME | DATE | 创建时间 | Y |
| UPDATE\_USER | VARCHAR2(20) | 最新一次修改者 | Y |
| UPDATE\_TIME | DATE | 最新一次修改时间 | Y |
| FILE\_SIZE | NUMBER(11) | 视频大小 | Y |
| VIDEO\_DEFINITION | VARCHAR2(10) | 视频清晰度 1:流畅 2：标清 3：高清 4：超清 | Y |
| VIDEO\_RESOLUTION | VARCHAR2(20) | 视频分辨率，宽\*高 | Y |
| MAX\_ONLINE | NUMBER(11) | 最高在线人数，针对采编状态为直播回放 | Y |
| VIDEO\_URL\_TYPE | VARCHAR2(2) | 视频文件路径类型 0:相对路径，1：全路径 | N |
| PLAY\_COUNT | NUMBER(11) | 播放次数 | N |
| DESCRIPTION | VARCHAR2(200) |  | Y |
| TEAM\_ID | NUMBER(11) | 战队ID | Y |
| TEAM\_NAME | VARCHAR2(30) | 战队名称 | Y |
| SHOW\_TYPE | NUMBER(2) | 直播展示方式1:横屏 、2：竖屏 | Y |
| PLAY\_COEFFICIENT | NUMBER(11,1) | 播放次系数。播放次数\*播放次系数 = 展示播放次数 | Y |

1. **视频报错表game\_video\_error**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 字段类型 | 表字典描述 | 是否必填 |
| ID | NUMBER(12) | 主键 | 是 |
| VIDEO\_ID | NUMBER(12) | 视频ID | 是 |
| error\_describe | Varchar(1024) | 报错描述 | 是 |
| person\_label | Varchar(1024) | 报错内容，角色标签id | 否 |
| scence\_label | Varchar(1024) | 报错内容，场景标签id | 否 |
| create\_TIME | DATE | 创建时间 | 否 |
| create\_USER | VARCHAR2(100) | 创建人 | 否 |
| UPDATE\_TIME | DATE | 更新时间 | 否 |
| UPDATE\_USER | VARCHAR2(100) | 更新人 | 否 |

### 视频标签管理

#### 数据表

游戏场景标签表（game\_scene\_label）、游戏角色标签表（game\_role\_label）

#### 主要功能

运营人员在咪咕游戏后台管理系统中通过【视频标签管理】页面对视频标签记录进行维护，标签分为游戏角色、游戏场景两大类。

**一、视频标签管理页**



【标签管理页】为一级页面，点击展开“游戏角色”、“游戏场景”两个二级页面，点击“游戏角色”、“游戏场景”展示对应页面，再次点击则关闭页面。

**二、游戏角色**

**查询条件**

【标签ID】输入标签ID进行精确匹配，不输入则为空。

【游戏角色】输入游戏角色名称进行精准匹配，不输入则为空。

**标签列表**

【新增】点击新增拉起“游戏角色新增弹窗”，填写相关信息后进行添加，详见“游戏角色新增弹窗”

【修改】用户选择标签后点击“修改”，则拉起带有原有内容的“游戏角色新增弹窗”，用户编辑后点击“提交”后进行更新，点击“取消”则保持原有内容不变。如果未选择任何标签点击“修改”则提示用户“请至少选择一条数据”，点击“确定”关闭弹窗。

【删除】用户选择标签后点击“删除”，弹窗提示用户“是否确认删除”，点击“确定”进行删除，点击“取消”则关闭弹窗。

**三、游戏场景**

**查询条件**

【标签ID】输入标签ID进行精确匹配，不输入则为空。

【游戏场景】输入游戏场景进行精准匹配，不输入则为空。

**标签列表**

【新增】点击新增拉起“游戏场景新增弹窗”，填写相关信息后进行添加，详见“游戏场景新增弹窗”

【修改】用户选择标签后点击“修改”，则拉起带有原有内容的“游戏场景新增弹窗”，用户编辑后点击“提交”后进行更新，点击“取消”则保持原有内容不变。如果未选择任何标签点击“修改”则提示用户“请至少选择一条数据”，点击“确定”关闭弹窗。

【删除】用户选择标签后点击“删除”，弹窗提示用户“是否确认删除”，点击“确定”进行删除，点击“取消”则关闭弹窗。

**四、游戏角色新增弹窗**

注：以下内容均不可为空，否则不可进行提交。



【游戏角色】用户输入游戏角色名称，最多70个字符。

【性别】下拉进行选择，默认为空，枚举值有：男、女、其他

【角色类型】用户输入类型，最多70个字符，可同时输入多种角色类型。

【角色截图】点击“浏览”添加格式为gif/jpg/png/jpeg格式的图片。

【提交】如果上述内容均已填完，点击进行提交，如果有任意内容为空则点击无效。

【取消】点击取消关闭弹窗。

**五、游戏场景新增弹窗**

注：以下内容均不可为空，否则不可进行提交。

****

【游戏场景】用户输入游戏场景名称，最多70个字符。

【场景得分】下拉选择，默认为空，枚举值为：1,2,3,4,5

【场景描述】用户输入最多70个字符。

【场景截图】点击“浏览”添加格式为gif/jpg/png/jpeg格式的图片。

【提交】如果上述内容均已填完，点击进行提交，如果有任意内容为空则点击无效。

【取消】点击取消关闭弹窗。

### AI视频合成

#### 数据表

游戏短视频表（game\_short\_video）、游戏视频标签对应表（game\_video\_label）、游戏视频报错表（game\_video\_error）

#### 主要功能

运营人员在咪咕游戏后台管理系统中通过【AI合成视频】页面查询、预览、合成AI模型生成的精彩短视频，将短视频合成为长视频。

**一、审核短视频页**

****

**查询条件**

【视频ID】输入进行精确匹配，不输入则为空。

【原视频ID】输入视频ID进行精确匹配，不输入则为空。

【游戏场景】输入游戏场景进行精确匹配，游戏场景需要经过“游戏场景表”进行“增删改”。

【游戏角色】输入游戏角色进行精确匹配，游戏角色需要经过“游戏角色表”进行“增删改”。

【角色类型】下拉进行选择，默认为“全部”，枚举值包括：坦克、战士、刺客、辅助、法师、射手，一个视频可能有多个角色类型，进行模糊匹配。

【角色性别】下拉进行选择，默认为“全部”，枚举值包括：男、女、其他，进行精确匹配。

【查询】点击“查询”根据已输入的搜索条件进行筛选

【报错】同“关联视频页”的“报错”按钮

**视频列表**

【短视频ID】OMS后台通过AI前置平台“短视频返回接口”拿到VID，然后自动生成一一对应的“短视频ID”。

【游戏场景】OMS后台通过AI前置平台“短视频返回接口”拿到每一个短视频的游戏场景。

【游戏角色】OMS后台通过AI前置平台“短视频返回接口”拿到每一个短视频的游戏角色。

【短视频时长】OMS后台通过AI前置平台“短视频返回接口”拿到每一个短视频的时长，精确到秒。

【角色类型】OMS后台根据从“短视频返回接口”拿到的游戏角色，在“游戏角色表”中匹配得到对应的角色类型，一个角色可能对应多个角色类型，

【角色性别】OMS后台根据从“短视频返回接口”拿到的游戏角色，在“游戏角色表”中匹配得到对应的角色性别，一个角色只会对应一个性别类型。

【视频内容】OMS后台通过AI前置平台“短视频返回接口”拿到每一个短视频的播放地址。

**二、AI合成视频页**

****

**查询条件**

【视频ID】系统生成，不可为空，输入进行精确匹配；

【视频名称】输入进行精确匹配,不输入则为空白。

【游戏角色】输入进行模糊匹配，不输入则为空白。

【所属业务ID】在进行视频“新增”（详见“新增”解释）时，用户关联额对应业务ID，输入ID进行精确匹配。

【游戏场景】输入进行模糊匹配。

【采编状态】下拉框展示，枚举值有“采编”、“非采编”、“三方渠道引入”、“直播回放”；

【查询】点击“查询”根据已输入的搜索条件进行筛选

**视频列表**

【新增】详见“新增AI合成视频页”

【修改】选择视频后，点击“修改”则弹出带有原有信息的“新增AI合成视频页”，点击“提交”后刷新原有内容。如果用户为选择任何视频并点击“修改”则提示用户“请至少选择一条数据”，点击“确定”关闭弹窗。

【删除】选择视频后点击“删除”，弹窗提示用户“是否确认删除”，点击“确定”进行删除，点击“取消”则关闭弹窗。

【生成】选择视频后点击“生成”，如果该视频没有关联短视频则弹窗提示“请关联至少一个视频”，点击“确定”关闭弹窗。如果以关联视频内容，则将该视频内容转至“视频管理页”，视频ID、视频名称、视频发布者ID、视频发布者昵称、所属业务ID、采编状态、视频时长、视频封面均读取当前视频对应字段内容。分辨率为“短视频返回接口”返回的分辨率值。

【视频ID】系统生成ID，不可为空。

【视频名称】在进行“AI长视频页”创建长视频内容时进行编辑，

【游戏角色】视频列表页的任意一行右键“关联游戏”（详见“关联游戏”解释），添加的所有短视频内包含的游戏角色，按照视频顺序对角色顺序进行排序，尚未关联视频时显示为空。

【游戏场景】在视频列表页的任意一行右键“关联游戏”（详见“关联游戏”解释），添加的所有短视频内包含的游戏场景，按照视频顺序对场景顺序进行排序，尚未关联视频时显示为空。

【视频时长】在视频列表页的任意一行右键“关联游戏”（详见“关联游戏”解释），添加的所有短视频的时长总和，尚未关联视频时显示为零，精确到秒。

【创建时间】在”AI合成视频页”点击“新增”（详见“新增”解释），进行视频相关信息填写后提交生成页面时间，精确到秒。

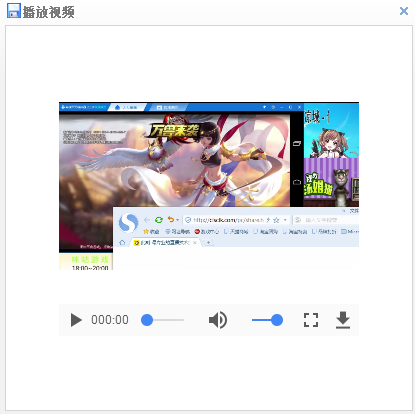
【所属业务ID】为用户在“新增AI合成视频页”中填写的“所属业务ID”

【采编状态】为用户在“新增AI合成视频页”中填写的“采编状态”

【预览】点击“预览”则弹出“视频播放页”（详见“视频播放页”）进行视频播放，尚未关联视频时显示为空。

【关联游戏】视频列表页的任意一行右键选择“关联游戏”，则弹出关联视频页面如下所示，进行视频关联。

**三、AI合成视频预览页**



通过该页面可进行短视频、AI合成视频的播放，点击右上角“叉号”进行关闭。

**四、新增AI合成视频页**



注：以下字段均为必填字段，有任意内容为空则不可提交。

【视频名称】输入名称长度必须在1-70之间

【视频发布者ID】用户输入的ID必须是在用户信息管理内存在的用户ID.

【视频发布者昵称】当输入正确的用户ID后，点击空白框则会自动同步对应的昵称信息。

【所属业务ID】点击“关联游戏”进入关联游戏页面进行关联，相关页面与“视频管理”页面中的“关联游戏”功能保持一致。

【采编状态】下拉选择，默认为空白,枚举值包含：非采编、采编、三方渠道引入、直播回放。

【视频封面】点击浏览进行视频封面选择，支持gif、jpg、png、jpeg四种格式。

【提交】点击提交则在“AI合成视频页”生成新的视频。

【取消】点击取消则返回上一页面。

**五、关联视频页**



**查询条件**

【短视频ID】输入进行精确匹配，不输入则为空。

【游戏场景】输入游戏场景进行精确匹配，游戏场景需要经过“游戏场景表”进行“增删改”。

【原视频ID】输入视频ID进行精确匹配，不输入则为空。

【游戏角色】输入游戏角色进行精确匹配，游戏角色需要经过“游戏角色表”进行“增删改”。

【查询】点击“查询”根据已输入的搜索条件进行筛选

**视频列表**

【关联新视频】点击在“短视频页面”进行短视频内容的筛选，具体操作见“短视频页”。

【取消关联】选择多个视频后，点击“取消关联”则在该页面对视频内容进行删除。如果用户未选择视频且点击“取消关联”则提示用户“请至少选择一条数据”，点击“确定”关闭弹窗。

【短视频ID】进行视频关联时读取“短视频页”对应的短视频ID

【游戏场景】进行视频关联时读取“短视频页”对应的游戏场景

【游戏角色】进行视频关联时读取“短视频页”对应的游戏角色

【短视频时长】进行视频关联时读取“短视频页”对应的短视频时长

【长视频ID】进行视频关联时读取“短视频页”对应的长视频ID

【角色类型】进行视频关联时读取“短视频页”对应的角色类型

【角色性别】进行视频关联时读取“短视频页”对应的角色性别

【视频内容】进行视频关联时读取“短视频页”对应的视频内容，点击可通过“视频播放页”进行播放。

【展示顺序】用户选择视频后默认按照“短视频页”视频顺序进行展示，但是可以通过点击“展示顺序”框输入“1/2/3”等顺序数字进行排序，合成的长视频按照此顺序进行短视频播放。

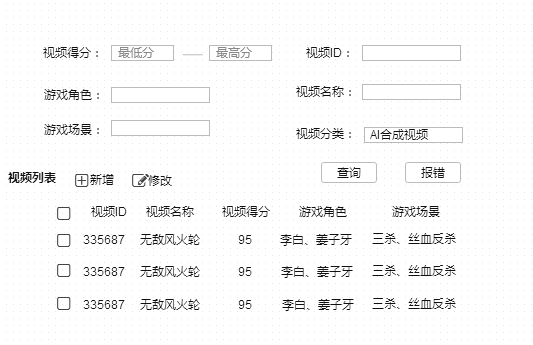
### AI视频发布

#### 数据表

游戏视频表（game\_video）、游戏视频标签对应表（game\_video\_label）

#### 主要功能

运营人员在咪咕游戏后台管理系统中通过【视频管理】页面查询、发布AI合成视频到前端产品。如发现AI合成视频识别标签错误，可向AI前置服务平台报错，AI前置服务平台更新样本库后，将样本上传AI云平台；算法人员登录AI云平台重新训练模型，完成训练后AI前置服务平台重新载入新模型。



注：此部分查询条件及视频列表字段均是在原有“视频管理”页面的基础上进行添加，如有重复则不再添加。

**查询条件：**

【视频得分】输入最低分、最高分进行视频筛选，无最高限制，不输入则不限制，上线值和下线值均包含在搜索范围内。

【视频ID】输入视频ID进行查询，精确匹配。

【游戏角色】输入角色名称进行精确匹配，游戏角色需要经过“游戏角色表”进行“增删改”。

【游戏场景】输入业务侧定义的场景名称进行精确匹配，游戏场景需经过“游戏场景表”进行“增删改”。

【视频分类】所有经过AI化短视频合成的长视频类型均默认为“AI合成视频”，且不可编辑。

【查询】点击“查询”根据已输入的搜索条件进行筛选

【报错】选择多个视频内容，点击“报错”则弹出报错弹窗，如下所示，页面读取“视频ID”、“视频名称”，“报错”内容需用户进行手动编辑，且为必填项，存在任意一个为空则不可提交。点击“取消”则返回原有界面，只针对AI合成视频，如有其它类型视频则不弹出报错页面。



## 系统接口

### 数据上传接口

* 功能描述：将训练样本视频及配套标注、视频元数据说明文件由互娱大数据平台上传至AI云平台，上传结果需要记录日志。
* 接口方式：FTP，使用现有的大数据平台与各子公司之间的数据传输通道。
* 传输数据：视频源文件、标注文件、视频元数据说明文件

### 模型发布接口

* 功能描述：将训练生成的tensorflow或caffe模型文件、模型元数据说明文件从AI云平台下载至AI前置服务平台，在AI前置服务平台上发布模型，解析并保存模型元数据，下载和发布模型需要记录日志。
* 接口方式：FTP，使用现有的大数据平台与各子公司之间的数据传输通道。
* 传输数据：模型文件、模型元数据说明文件

### 模型使用接口

* 功能描述：咪咕游戏服务端调用互娱大数据平台，互娱大数据平台再调用AI前置服务平台的业务调度模块，获取视频理解AI能力服务，识别视频中的游戏标签、精彩短视频及其封面图地址。
* 接口方式：RESTFUL
* 传输数据：参见第3.2.2节。

### 标签同步接口

* 功能描述：运营人员在咪咕游戏的标签管理界面中维护游戏标签数据后，咪咕游戏服务端实时将标签记录发送至互娱大数据平台，用于样本标注。
* 接口方式：Kafka
* 传输数据：参见第3.1.2节

### 视频同步接口

* 功能描述：咪咕游戏服务端每小时将新增的游戏视频存储地址发送给互娱大数据平台，互娱大数据负责下载视频源文件再调用模型使用接口分析视频标签，拆解成短视频。
* 接口方式：Kafka
* 传输数据：参见第3.2.1节

### 结果输出接口

* 功能描述：互娱大数据平台调用AI模型获取游戏视频标签、短视频地址后，将短视频上传咪咕云存储，再将视频标签、视频地址发送回咪咕游戏服务端。
* 接口方式：Kafka
* 传输数据：参见第3.2.3节

### 模型反馈接口

* 功能描述：咪咕游戏运营人员审核视频时，如发现模型计算错误，则将错误视频和对应的正确标签发送回互娱大数据平台，将错误视频作为样本进行重新标注、重新训练模型。
* 接口方式：Kafka
* 传输数据：参见第3.2.4节

# 系统响应码

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **场景** | **响应码** | **说明** |
|  | 调用接口 | 0000 | 返回成功 |
|  | 调用接口 | 0001 | 输入参数为空 |
|  | 调用接口 | 0002 | 输入参数格式错误 |
|  | 调用接口 | 0003 | 输入参数类型错误 |
|  | 调用接口 | 0004 | 输入文件路径不存在 |
|  | 调用接口 | 0005 | 模型错误 |
|  | 调用接口 | 0006 | 图片格式异常 |
|  | 调用接口 | 0007 | 视频格式异常 |
|  | 调用接口 | 0008 | 处理视频异常 |
|  | 调用接口 | 0009 | 输出图片异常 |
|  | 调用接口 | 0010 | 输出视频异常 |
|  | 调用接口 | 0011 | 输出推荐路线异常 |
|  | 调用接口 | 0012 | 请求地址无响应 |