(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 110971928 B
(45)授权公告日 2022.03.25

(21)申请号 201811161528.1

(22)申请日 2018.09.30

(65)同一申请的已公布的文献号
申请公布号 CN 110971928 A

(43)申请公布日 2020.04.07

(73)专利权人 武汉斗鱼网络科技有限公司
地址 430000 湖北省武汉市东湖开发区软件园东路1号软件产业4.1期B1栋11楼

(72)发明人 周志刚 胡卫谊

(74)专利代理机构 北京众达德权知识产权代理有限公司 11570
代理人 徐松

(51)Int.Cl.
H04N 21/234(2011.01)
H04N 21/8547(2011.01)
H04N 21/2187(2011.01)
H04N 21/235(2011.01)
H04N 21/258(2011.01)
G06V 10/74(2022.01)

(56)对比文件
US 2015264063 A1,2015.09.17
CN 106686395 A,2017.05.17
CN 102306287 A,2012.01.04
CN 107222780 A,2017.09.29
US 2015117765 A1,2015.04.30
US 2018276541 A1,2018.09.27
US 2005008225 A1,2005.01.13

审查员 曹珊珊

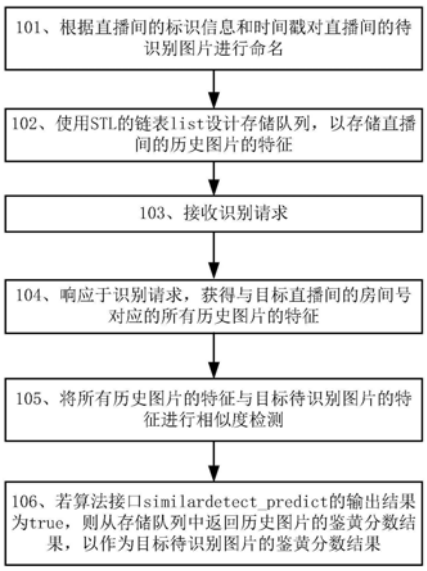
权利要求书2页 说明书20页 附图2页

(54)发明名称

一种图片识别方法及相关装置

(57)摘要

本发明实施例公开了一种图片识别方法及相关装置，用于降低GPU的识别资源并提高了鉴黄审核效率。本发明实施例方法包括：根据直播间的标识信息和时间戳对直播间的待识别图片进行命名；使用标准模板库STL的链表list设计存储队列，以存储直播间的历史图片的特征，存储队列包括N张图片；接收识别请求，识别请求用于请求识别目标直播间中的目标待识别图片，目标直播间包含于直播间；响应于识别请求，获得与目标直播间的房间号对应的所有历史图片的特征；调用算法接口similardetect_predict将所有历史图片的特征与目标待识别图片的特征进行相似度检测；若算法接口similardetect_predict的输出结果为true，则从存储队列中返回历史图片的鉴黄分数结果，以作为目标待识别图片的鉴黄分数结果。



CN 110971928 B

1.一种图片识别方法,其特征在于,包括:

根据直播间的标识信息和时间戳对所述直播间的待识别图片进行命名;

使用标准模板库STL的链表list设计存储队列,以存储所述直播间的历史图片的特征,所述存储队列包括N张图片,所述直播间的历史图片的特征包括所述历史图片的鉴黄分数结果;

接收识别请求,所述识别请求用于请求识别目标直播间中的目标待识别图片,所述目标直播间包含于所述直播间;

响应于所述识别请求,获得与所述目标直播间的房间号对应的所有历史图片的特征;

调用算法接口similardetect_predict将所述所有历史图片的特征与所述目标待识别图片的特征进行相似度检测;

若所述算法接口similardetect_predict的输出结果为true,则从所述存储队列中返回所述历史图片的鉴黄分数结果,以作为所述目标待识别图片的鉴黄分数结果。

2.根据权利要求1所述的方法,其特征在于,所述使用STL的链表list设计存储队列包括:

设计结构ImageData存储所述历史图片和所述历史图片的特征,所述结构ImageData包括所述历史图片的原始数据,所述历史图片的名称,所述历史图片的特征数据,和所述历史图片的鉴黄分数结果;

根据所述结构ImageData定义链表变量list＜ImageData*＞listImageData存储各直播间的历史图片的特征。

3.根据权利要求2所述的方法,其特征在于,所述获得与所述目标直播间的房间号对应的所有历史图片的特征包括:

确定与所述目标直播间的房间号对应的目标链表,所述目标链表包括所述目标直播间的历史图片;

遍历所述目标链表以获得所述所有历史图片的特征。

4.根据权利要求3所述的方法,其特征在于,所述确定与所述目标直播间的房间号对应的目标链表包括:

定义迭代器Map＜int,list＜ImageData*＞*＞::iterator itr;

调用map容器的find函数通过目标直播间的房间号nRoomId来查找对应的所有特征值的目标链表,并返回查找到的所述目标链表。

5.根据权利要求3所述的方法,其特征在于,所述调用算法接口similardetect_predict将所述所有历史图片的特征与所述目标待识别图片的特征进行相似度检测包括:

定义所述目标链表的迭代器;

通过函数itr＝list1mageData.begain()初始化所述目标链表的迭代器,所述itr用于表示所述目标链表的迭代器;

通过for循环函数遍历所述目标链表的迭代器,

并调用所述算法接口similardetect_predict的函数Int Result＝similardetect_predict((*itr).ImageFeature,ImageFeature)将所述所有历史图片的特征与所述目标待识别图片的特征进行相似度检测,其中,所述(*itr).ImageFeature用于表示所述历史图片的特征,所述ImageFeature用于表示所述目标待识别图片的特征。

6.根据权利要求4所述的方法,其特征在于,所述方法还包括:

若所述算法接口similardetect_predict的输出结果为false,则将存在时间最久的历史图片的特征删除,并将所述目标待识别图片的特征更新到所述存储队列中。

7.根据权利要求6所述的方法,其特征在于,所述将存在时间最久的历史图片的特征删除,并将所述目标待识别图片的特征更新到所述存储队列中包括:

调用函数listImageData.pop_front()将所述存在时间最久的历史图片的特征删除;

调用函数listImageData,push_back(ImageData)将所述目标待识别图片的特征存储到所述目标链表中。

8.一种鉴黄服务器,其特征在于,包括:

命名单元,用于根据直播间的标识信息和时间戳对所述直播间的待识别图片进行命名;

存储单元,用于使用标准模板库STL的链表list设计存储队列,以存储所述直播间的历史图片的特征,所述存储队列包括N张图片,所述直播间的历史图片的特征包括所述历史图片的鉴黄分数结果;

收发单元,用于接收识别请求,所述识别请求用于请求识别目标直播间中的目标待识别图片,所述目标直播间包含于所述直播间;

获取单元,用于响应于所述识别请求,获得与所述目标直播间的房间号对应的所有历史图片的特征;

检测单元,用于调用算法接口similardetect_predict将所述所有历史图片的特征与所述目标待识别图片的特征进行相似度检测;

返回单元,用于若所述算法接口similardetect_predict的输出结果为true,则从所述存储队列中返回所述历史图片的鉴黄分数结果,以作为所述目标待识别图片的鉴黄分数结果。

9.一种计算机可读存储介质,包括指令,当其在计算机上运行时,使得计算机执行如权利要求1-7任意一项所述的方法。

10.一种电子设备,包括存储器、处理器,其特征在于,所述处理器用于执行存储器中存储的计算机程序时实现如权利要求1-7任意一项所述的方法。

一种图片识别方法及相关装置

技术领域

[0001] 本发明涉及开发平台领域,尤其涉及一种图片识别方法及相关装置。

背景技术

[0002] 对于直播平台来说,需要对直播平台所直播的内容进行审核,以确认直播的内容是正规合法的内容,而不能是色情直播。现有技术中,确认直播的内容是否正规合法,主要是依赖于审核人员,需要去定时的查看每个直播间是否存在违规的直播内容。显然,现有技术耗时耗力,且审核的效率不高。

[0003] 因此,如何更加智能的识别直播间所直播的内容是否正规合法,即提高审核效率,是现在急需解决的问题。

发明内容

[0004] 本发明实施例提供了一种图片识别方法及相关装置,用于降低GPU的识别资源并提高了鉴黄审核效率。

[0005] 本发明实施例的第一方面提供了一种图片识别方法,包括:根据直播间的标识信息和时间戳对所述直播间的待识别图片进行命名;使用标准模板库STL的链表list设计存储队列,以存储所述直播间的历史图片的特征,所述存储队列包括N张图片,所述直播间的历史图片的特征包括所述历史图片的鉴黄分数结果;接收识别请求,所述识别请求用于请求识别目标直播间中的目标待识别图片,所述目标直播间包含于所述直播间;响应于所述识别请求,获得与所述目标直播间的房间号对应的所有历史图片的特征;调用算法接口similardetect_predict将所述所有历史图片的特征与所述目标待识别图片的特征进行相似度检测;若所述算法接口similardetect_predict的输出结果为true,则从所述存储队列中返回所述历史图片的鉴黄分数结果,以作为所述目标待识别图片的鉴黄分数结果。

[0006] 在一种可能的实施例中,所述使用STL的链表list设计存储队列包括:设计结构ImageData存储所述历史图片和所述历史图片的特征,所述结构ImageData包括所述历史图片的原始数据,所述历史图片的名称,所述历史图片的特征数据,和所述历史图片的鉴黄分数结果;根据所述结构ImageData定义链表变量list<ImageData*>listImageData存储各直播间的历史图片的特征。

[0007] 在一种可能的实施例中,所述获得与所述目标直播间的房间号对应的所有历史图片的特征包括:确定与所述目标直播间的房间号对应的目标链表,所述目标链表包括所述目标直播间的历史图片;遍历所述目标链表以获得所述所有历史图片的特征。

[0008] 在一种可能的实施例中,所述确定与所述目标直播间的房间号对应的目标链表包括:定义迭代器Map<int,list<ImageData*>*>::iterator itr;调用map容器的find函数itr=mapListData.find(nRoomId)查找与所述目标直播间的房间号对应的目标链表,所述nRoomId用于表示所述目标直播间的房间号,所述find函数返回的结果为所述目标链表。

[0009] 在一种可能的实施例中,所述调用算法接口similardetect_predict将所述所有

历史图片的特征与所述目标待识别图片的特征进行相似度检测包括:定义所述目标链表的迭代器;通过函数itr＝listImageData.begain()初始化所述目标链表的迭代器,所述itr用于表示所述目标链表的迭代器;通过for循环函数遍历所述目标链表的迭代器,并调用所述算法接口similardetect_predict的函数Int Result＝similardetect_predict((*itr).ImageFeature,ImageFeature)将所述所有历史图片的特征与所述目标待识别图片的特征进行相似度检测,其中,所述(*itr).ImageFeature用于表示所述历史图片的特征,所述ImageFeature用于表示所述目标待识别图片的特征。

[0010] 在一种可能的实施例中,所述方法还包括:若所述算法接口similardetect_predict的输出结果为false,则将存在时间最久的历史图片的特征删除,并将所述目标待识别图片的特征更新到所述存储队列中。

[0011] 在一种可能的实施例中,所述将存在时间最久的历史图片的特征删除,并将所述目标待识别图片的特征更新到所述存储队列中包括:调用函数listImageData.pop_front()将所述存在时间最久的历史图片的特征删除;调用函数listImageData,push_back(ImageData)将所述目标待识别图片的特征存储到所述目标链表中。

[0012] 本发明实施例的第二方面提供了一种鉴黄服务器,包括:命名单元,用于根据直播间的标识信息和时间戳对所述直播间的待识别图片进行命名;存储单元,用于使用标准模板库STL的链表list设计存储队列,以存储所述直播间的历史图片的特征,所述存储队列包括N张图片,所述直播间的历史图片的特征包括所述历史图片的鉴黄分数结果;收发单元,用于接收识别请求,所述识别请求用于请求识别目标直播间中的目标待识别图片,所述目标直播间包含于所述直播间;获取单元,用于响应于所述识别请求,获得与所述目标直播间的房间号对应的所有历史图片的特征;检测单元,用于调用算法接口similardetect_predict将所述所有历史图片的特征与所述目标待识别图片的特征进行相似度检测;返回单元,用于若所述算法接口similardetect_predict的输出结果为true,则从所述存储队列中返回所述历史图片的鉴黄分数结果,以作为所述目标待识别图片的鉴黄分数结果。

[0013] 本发明第三方面提供了一种电子设备,包括存储器、处理器,其特征在于,所述处理器用于执行存储器中存储的计算机管理类程序时实现如上述任意一项所述的方法的步骤。

[0014] 本发明第四方面提供了一种计算机可读存储介质,其上存储有计算机管理类程序,其特征在于:所述计算机管理类程序被处理器执行时实现如上述任意一项所述的方法的步骤。

[0015] 从以上技术方案可以看出,本发明实施例具有以下优点:根据直播间的标识信息和时间戳对所述直播间的待识别图片进行命名;使用标准模板库STL的链表list设计存储队列,以存储所述直播间的历史图片的特征,所述存储队列包括N张图片,所述直播间的历史图片的特征包括所述历史图片的鉴黄分数结果;接收识别请求,所述识别请求用于请求识别目标直播间中的目标待识别图片,所述目标直播间包含于所述直播间;响应于所述识别请求,获得与所述目标直播间的房间号对应的所有历史图片的特征;调用算法接口similardetect_predict将所述所有历史图片的特征与所述目标待识别图片的特征进行相似度检测;若所述算法接口similardetect_predict的输出结果为true,则从所述存储队列中返回所述历史图片的鉴黄分数结果,以作为所述目标待识别图片的鉴黄分数结果。本发

明实施例中，基于图片若相似的，其对应的识别结果也是相似的这样的考虑，通过GPU的算法来对图片进行相似度的识别，如果图片相似则直接返回之前的识别结果，从而不会调用到GPU的识别资源，并减少产生瓶颈的可能性，即本申请实施例降低GPU的识别资源并提高了鉴黄审核效率。

附图说明

[0016] 图1为本发明实施例提供的一种可能的图片识别方法的流程图；

[0017] 图2为本发明实施例提供的一种可能的鉴黄服务器的结构示意图；

[0018] 图3为本发明实施例提供的一种可能的电子设备的硬件结构示意图；

[0019] 图4为本发明实施例提供的一种可能的计算机可读存储介质的硬件结构示意图。

具体实施方式

[0020] 本发明实施例提供了一种图片识别方法及相关装置，用于降低GPU的识别资源并提高了鉴黄审核效率。

[0021] 下面将结合本发明实施例中的附图，对本发明实施例中的技术方案进行清楚、完整地描述，显然，所描述的实施例仅仅是本发明一部分实施例，而不是全部的实施例。基于本发明中的实施例，本领域技术人员在没有做出创造性劳动前提下所获得的所有其他实施例，都属于本发明保护的范围。

[0022] 本申请的说明书和权利要求书及上述附图中的术语“第一”、“第二”、“第三”、“第四”等(如果存在)是用于区别类似的对象，而不必用于描述特定的顺序或先后次序。应该理解这样使用的数据在适当情况下可以互换，以便这里描述的实施例能够以除了在这里图示或描述的内容以外的顺序实施。此外，术语“包括”和“具有”以及他们的任何变形，意图在于覆盖不排他的包含，例如，包含了一系列步骤或单元的过程、方法、系统、产品或设备不必限于清楚地列出的那些步骤或单元，而是可包括没有清楚地列出的或对于这些过程、方法、产品或设备固有的其它步骤或单元。

[0023] 对于本申请的鉴黄识别服务来说，每种图片都会使用图形处理器(graphics processing unit，GPU)资源，而GPU是整个识别中的最核心的资源，同时是也是最容易产生瓶颈的资源，而通过整个系统的运行，本文发现对于直播间来说，采取使用预置时长的策略来截取视频中的图片进行识别，发现对于很多情况下，连续的一段时间内，直播间的内容没有发送很大的变化。

[0024] 有鉴于此，本申请提供了一种图片识别方法，请参阅图1，为本发明实施例提供的一种图片识别方法的流程图，具体包括：

[0025] 101、根据直播间的标识信息和时间戳对直播间的待识别图片进行命名；

[0026] 对于直播间来说，相似图片的识别只会对同一个直播间来说，不同的直播间直播的内容不一样，所以也不存在相似图的识别问题。那么对于每个直播间来说直播间号是直播平台唯一区分每个直播间的标识信息，故对于每个直播间需要进行识别的图片本文则会对图片进行命名，其中命名包括该图片所属的直播间号加上时间戳，从而即可以知道该图片所属的直播间，也不会存在图片有重名的情况，还可以知道图片产生的时间。具体实现如下：

[0027] ImageName＝RoomId+Time()；

[0028] 其中ImageName是图片的名称;RoomId是房间号;Time()则是获取系统时间戳。

[0029] 因此通过上述方式,实现了根据直播间的标识信息和时间戳对直播间的待识别图片进行命名,该待识别图片为直播间内需要进行鉴黄识别的图片。

[0030] 102、使用STL的链表list设计存储队列,以存储直播间的历史图片的特征；

[0031] 对于直播间来说,需要设计一个队列来存储历史特征。由于当前识别的图片可能和之前的图片是相似的,所以本文设置队列长度是N张图片,如10张图片,每次都和这10张图片进行对比。可以理解的是,队列长度设置太长或者太短都会影响效果,如果设置太长,那么对比会消耗大量时间,如果设置太短那么有可能是相似的没有对比到,因此队列长度的设置可以基于实际应用场景。本文使用STL的链表list来存储图片的历史特征,具体实现如下：

[0032] Struct ImageData{

[0033] Image;ImageName;ImageFeature;Score;

[0034] }

[0035] 其中,该结构ImageData来存储一个图片的特征和图片,其中结构中包括图片的原始数据Image,图片的名称ImageName,图片的特征数据ImageFeature,Score则是图片识别的分数值。

[0036] 然后定义一个链表变量list<ImageData*>listImageData来存储一个房间的历史特征。对于整个直播网站来说,需要查询所有的直播间,所以本文使用了STL的键值对容器map来存储所有直播间的历史特征,其中键使用房间号,而值则使用了链表变量,具体实现函数包括:Map<int,list<ImageData*>*>mapListData。

[0037] 103、接收识别请求；

[0038] 本申请实施例中,使用HTTP的请求形式,通过向服务器发送识别请求,该识别请求用于请求识别目标直播间中的目标待识别图片,目标直播间包含于直播间。需要说明的是,该识别请求中需要携带上房间号,图片ID,图片内容。具体实现如下：

[0039] 本实施例中,基于开源的BOOST库提供的HTTP请求来创建HTTP的识别请求,包括：

[0040] 创建一个请求的数据流存储对象boost::asio::streambuf request,将其绑定到STL的输出流对象上std::ostream request_stream(&request),然后输入一些http的头信息：

[0041] request_stream<<"POST"<<url<<"HTTP/1.0\r\n",其中url是鉴黄服务器的http接口地址。

[0042] request_stream<<"Host:"<<host<<":"<<port<<"\r\n",其中host则对应的是接口名称,port对应的是端口号。

[0043] request_stream<<"Accept:*/*\r\n";

[0044] request_stream<<"Content-Length:"<<data.length()<<"\r\n";

[0045] request_stream<<"Content-Type:application/x-www-form-urlencoded\r\n";

[0046] request_stream<<"Connection:close\r\n\r\n";

[0047] 再输入房间号,图片ID号和图片内容,图片高度和图片宽带,以得到该识别请求：

[0048] request_stream<<roomid;request_stream<<pictureId;request_stream<<

pictoureData;

[0049] request_stream<<width;request_stream<<height;

[0050] 在得到识别请求后,将该请求消息发送出去,使得服务器接收识别请求,具体地,通过boost提供的接口boost::asio::write(socket,request)将识别请求通过socket方式发送到服务器上。

[0051] 104、响应于识别请求,获得与目标直播间的房间号对应的所有历史图片的特征;

[0052] 接收到识别请求后,响应于该识别请求,获得与目标直播间的房间号对应的所有历史图片的特征。为获得历史图片的特征,在这之前,需要编写接口来实现读取历史特征和更新历史特征。当直播间有一张图片需要进行识别时,首先会依据该图片的所属房间号来拉取对应的所有历史特征值,即获得与目标直播间的房间号对应的所有历史图片的特征。其中本文则编写了接口GetImagData来获取目标直播间的所有特征值,输入参数为房间号nRoomId,具体实现函数包括:

[0053] list<ImageData*>*GetImagData(int nRoomId){

[0054] 首先定义一个迭代器:

[0055] Map<int,list<ImageData*>*>::iterator itr;

[0056] 然后调用map容器的find函数通过房间号来查找对应的所有特征值的链表,并返回查找到的结果:

[0057] Itr=mapListData.find(nRoomId);

[0058] Return(*itr).second;

[0059] }

[0060] 在得到了目标直播间的所有特征值的链表后,即获得与目标直播间的房间号对应的所有历史图片的特征。

[0061] 105、将所有历史图片的特征与目标待识别图片的特征进行相似度检测;

[0062] 106、若算法接口similardetect_predict的输出结果为true,则从存储队列中返回历史图片的鉴黄分数结果,以作为目标待识别图片的鉴黄分数结果。

[0063] 在获得与目标直播间的房间号对应的所有历史图片的特征后,可以遍历链表来一一匹配图像的特征值。首先定义一个链表的迭代器itr,具体实现函数如下:

[0064] list<ImageData*>::itrator itr;

[0065] 然后初始化迭代器:itr=listImageData.begain();

[0066] 进而编写For循环来对比特征值是否相似:

[0067] For(;itr!=listImageData.end();++itr){

[0068] 其中调用了相似度检测的算法接口similardetect_predict,其输入参数有2个,一个是链表中的图片特征,另一个则是当前需要对比的图片的特征,其中返回值Result则是返回结果,需要说明的是,如果返回值是true则表明相似,否则不相似。

[0069] Int Result=similardetect_predict((*itr).ImageFeature,ImageFeature);

[0070] If(Result==True){

[0071] Return True;

[0072] }

[0073] }。

[0074] 若算法接口similardetect_predict的输出结果为true,则返回历史图片的鉴黄分数结果,以作为目标待识别图片的鉴黄分数结果。

[0075] 可选的,如果当前图片和历史特征对比都没有相似的,则说明当前直播间的直播内容变化很大,所以需要将历史特征进行更新,踢掉存在时间最久的特征,并且将当前的特征更新到特征库中。具体则是通过调用其接口pop_front函数listImageData.pop_front()来踢掉存在时间最久的特征数据;然后通过函数listImageData.push_back(ImageData)将最新的特征存储到链表中。

[0076] 需要说明的是,本发明实施例中,加入了相似图的识别流程后,那么整个鉴黄的识别流程则发生了变化,鉴黄服务框架收到鉴黄的请求后,首先会则会获取该直播间的历史特征,并将请求与历史特征进行比较,如果匹配上,则直接返回历史特征图片识别的鉴黄分数结果,那么此流程则终止,不需要进行后续的基于GPU的深度学习的鉴黄识别模块,从而可以节省GPU资源,而整个匹配都是使用CPU进行计算。而如果请求在历史特征库中都没有找到相似图片,则整个流程需要使用GPU的深度学习的鉴黄识别模块,并且将识别的结果更新到历史特征库中,从而历史特征库总是保存最近的图片识别结果。

[0077] 本发明实施例中,基于图片若相似的,其对应的识别结果也是相似的这样的考虑,通过GPU的算法来对图片进行相似度的识别,如果图片相似则直接返回之前的识别结果,从而不会调用到GPU的识别资源,并减少产生瓶颈的可能性。

[0078] 实施例2、一种违规图片的识别方法;

[0079] 实际应用中,随着人工智能深度学习的普及,现在都会采用深度学习来智能的识别图片中是否包含有色情内容。通常深度学习比较消耗计算机资源,而对于直播平台来说,同时存在上万个直播间,并且要不间断的每隔预置时长30秒对直播间内容进行截图,查看其是否包含色情内容。因此会通过专用的深度学习机器,其具有大量的图形处理器(graphics processing unit,GPU)计算资源。对于鉴黄服务框架则要利用好深度学习机器,并且使得程序更具有健壮性,同时又能满足对直播平台的所有直播间都能够覆盖到。因此本文提供一种违规图片的识别方法,具体实现包括以下:

[0080] 步骤1、直播平台服务器遍历所有直播间,以获得各直播间的直播信息;

[0081] 本发明实施例,通过设计一个对直播平台的直播间内容是否涉及违规内容的服务化框架的平台,那么对于在线直播的每一个直播间都需要对其进行违规内容的智能识别。那么直播平台服务器则需要去遍历所有当前在直播的直播间,然后依次对每个直播间请求鉴黄服务,得到鉴黄结果。具体实现如下:

[0082] 对于当前直播平台的每一个直播间,直播平台会设计各直播间的直播信息,包括直播间的房间号,直播状态(是否在直播),直播间的名称即房间号,直播间的分区,直播间的视频流地址等其他信息。为了效率方面的提升,本申请实施例中,设计了一个Redis,其中Redis是一个开源的使用ANSI C语言编写、支持网络、可基于内存亦可持久化的日志型、Key-Value数据库,作为内存的数据存储和查询数据库。由于每个直播间房间号都是不一样的,所以本实施例中,使用房间号做为KEY值,而房间的其他信息做为Value来存储。

[0083] 需要说明的是,本申请实施例中,会使用一个map来存储直播间的直播信息。包括创建一个map对象:Map<string,string>mapInfo;然后将直播间的直播信息都存储到map对象中,例如mapInfo[stauts]=1,其中status标示当前直播间是否开播,status为1表示开

播,status为0表示关播;例如Mapinfo[name]＝“666”,其中name表示直播间的名称;例如mapinfo[type]＝“户外”;其中type表示直播间的分类;mapinfo[url]＝www.douyu.aaaaa.com,其中url表示视频流的地址。

[0084] 再创建一个redis对象redisContext来链接内存数据库,具体函数表现包括:redisContext*context＝redisConnect("127.0.0.1",6379),其中"127.0.0.1"表示IP地址,6379表示端口号。

[0085] 可选的,链接完内存数据库后,可使用redis对象redisCommand来存储这个数据,具体函数表现包括:redisCommand(context,"SET key:%s%s",roomid,Mapinfo),其中参数context表示之前创建的对象,roomid表示房间号作为key值,mapinfo则是直播间的其他信息。

[0086] 生成了redis的内存数据库后,则可以从中获取到所有的直播间信息,并且从中取出当前正在直播的直播间的视频流地址。

[0087] 步骤2、直播平台服务器从待审核直播间的直播视频中截取关键帧图片;

[0088] 在步骤101中从内存数据库可以得到每个直播间的直播信息,从而得到了当前正在直播的直播间的视频流地址信息。有了视频流地址,则可以从视频中截取关键帧的图片内容作为后续的图片鉴黄识别,即依据图片内容通过深度学习的模型来识别图片是否包含违规内容。

[0089] 本申请实施例中,可以使用开源的视频编解码库FFMPEG来对视频截取关键帧图片,该关键帧图片用于进行鉴黄识别。首先调用FFMPEG的库初始化接口av_register_all()来初始化。接下来调用应用程序编程接口(application programming interface,API)函数av_open_input_file(&pFormatCtx,url,NULL,0,NULL)打开视频流。进而得到视频流的上下文指针,即pCodecCtx＝pFormatCtx->streams[videoStream]->codec;创建一帧视频帧对象,即pFrame＝avcodec_alloc_frame();再创建一帧图像帧对象pFrameRGB＝avcodec_alloc_frame(),再创建得到视频帧对象和图像帧对象后,再填充视频帧,具体函数表现包括:

[0090] avpicture_fill((AVPicture*)pFrameRGB,buffer,PIX_FMT_RGB24,pCodecCtx->width,pCodecCtx->height)。

[0091] 填充视频帧后,解码该视频帧,具体解码方式可通过以下函数实现:

[0092] avcodec_decode_video(pCodecCtx,pFrame,&frameFinished,packet.data,packet.size),从中可以得到当前视频的一个关键帧,函数表现如下:sws_scale(pSWSCtx,pFrame->data,pFrame->linesize,0,pCodecCtx->height,pFrameRGB->data,pFrameRGB->linesize)。

[0093] 步骤3、直播平台服务器向鉴黄服务器发送请求消息;

[0094] 直播平台服务器截取到了视频流的关键帧图片,获取到了对应的图片内容后,接下来则是设计如何将图片内容来调用鉴黄服务,即请求鉴黄服务器对图片内容进行鉴黄识别。本申请实施例中,使用HTTP的请求形式,直播平台服务器通过向鉴黄服务器发送请求消息,以请求鉴黄服务器的HTTP服务来对图片进行鉴黄,并请求鉴黄服务返回鉴黄结果。需要说明的是,该请求消息中需要携带上房间号,图片ID,图片内容。具体实现如下:

[0095] 本实施例中,基于开源的BOOST库提供的HTTP请求来创建HTTP的请求消息,包括:

[0096] 创建一个请求的数据流存储对象boost::asio::streambuf request,将其绑定到STL的输出流对象上std::ostream request_stream(&request),然后输入一些http的头信息:

[0097] request_stream<<"POST"<<url<<"HTTP/1.0\r\n",其中url是鉴黄服务器的http接口地址。

[0098] request_stream<<"Host:"<<host<<":"<<port<<"\r\n",其中host则对应的是接口名称,port对应的是端口号。

[0099] request_stream<<"Accept:*/*\r\n";request_stream<<"Content-Length:"<<data.length()<<"\r\n";request_stream<<"Content-Type:application/x-www-form-urlencoded\r\n";

[0100] request_stream<<"Connection:close\r\n\r\n";

[0101] 再输入房间号,图片ID号和图片内容,图片高度和图片宽带,以得到该请求消息:

[0102] request_stream<<roomid;request_stream<<pictureId;request_stream<<pictoureData;

[0103] request_stream<<width;request_stream<<height;

[0104] 在得到请求消息后,将该请求消息发送出去,具体地,通过boost提供的接口boost::asio::write(socket,request)将请求消息通过socket方式发送到鉴黄服务器上。

[0105] 步骤4、鉴黄服务器搭建接收接口;

[0106] 需要说明的是,本申请实施例提供的是一个鉴黄服务,服务是以提供对外的http的接口,服务不断的接收http的请求来完成对应的请求的功能,并在请求中调用鉴黄模块来实现对关键帧图片的鉴黄结果。所以本步骤中,会搭建一个鉴黄服务的HTTP的服务程序来接收请求鉴黄的请求消息。类似的,本申请实施例中,基于开源库BOOST来搭建一个HTTP的服务器程序来接收请求消息。

[0107] 首先创建boost的对象,具体实现函数包括:boost::asio::ip::tcp::resolver resolver(io_service_);然后绑定服务器的ip和端口号,具体实现函数包括:boost::asio::ip::tcp::resolver::query query(address,port,其中address表示服务器的ip,port表示端口号。然后绑定该端口以进行侦听,具体地,实现函数包括:通过函数acceptor_.bind(endpoint)实现端口绑定;通过函数acceptor_.listen()实现侦听。

[0108] 当帧听到请求后,创建异步处理接口来处理请求,具体函数实现如下:acceptor_.async_accept(new_connection_->socket(),boost::bind(&server::handle_accept,this,boost::asio::placeholders::error));需要说明的是,异步处理接口则在接口server::handle_accept中处理。另外,在此接口server::handle_accept中,创建一个链接对象开始处理请求,创建链接对象的实现函数包括:connection_manager_.start(new_connection_)。

[0109] 在创建了链接对象后,需要对请求进行解析,即首先创建解析对象,具体实现函数包括:result;request_parser::result_type result;然后调用prase方法来进行解析,具体函数实现包括:request_parser_.parse(request_,buffer_.data(),buffer_.data()+bytes_transferred)。在解析完请求消息后,从中获取到对应的参数信息包括房价号,图片ID,图片内容,图片宽度和图片高度等。

[0110] 步骤5、鉴黄服务器创建鉴黄识别服务功能得到图片识别结果。

[0111] 鉴黄服务器接收到鉴黄图片的请求消息，并获取了请求消息的内容后，需要去调用鉴黄模块来识别待识别图片即关键帧图片，并得到待识别图片的识别结果。首先，创建鉴黄模块的句柄对象，void*pHandle＝ai_create()；然后通过调用其接口ai_create来创建句柄pHandle。得到句柄后，需要初始化鉴黄模块，具体的，可调用其对应的接口ai_init来初始化，ai_init(pHandle,ModuleFilePath,binGPU)，其中pHandle是创建的模块的句柄，ModuleFilePath是模型的文件名称，binGPU是选择的GPU芯片的编号。在初始化鉴黄模块后，调用其接口ai_predict来识别图片，并得到图片的打分，分值越高则说明违规的可能性越大，具体函数实现如下:ai_predict(pHandle,image,width,height,score)；其中参数pHandle是创建的句柄，image是图片的内容，width是图片的宽带，height是图片的高度，且识别结果存储在score字段，即通过score可以知道图片是否涉及违规。

[0112] 步骤6、直播平台服务器接收鉴黄服务器发送的鉴别结果；

[0113] 步骤7、直播平台服务器确定打分超过预设数值；

[0114] 步骤8、直播平台服务器确认关键帧图片违规，并对关键帧图片对应的直播间进行预置处理。

[0115] 在鉴黄服务器得到了鉴别结果后，则需要将该鉴别结果返回给请求者即直播平台服务器，需要说明的是，鉴别结果则是该待识别图片的打分的分值，该分值则表明了待识别图片是违规内容的分值。例如，满分是100分，本文则设定为鉴别结果为90分及以上的待识别图片为违规图片。具体实现如下：

[0116] 鉴黄服务器得到了请求消息后则调用鉴黄服务器的识别模块，根据识别模块得到了识别结果。接下来则需要将识别结果返回给直播平台服务器，具体数值则通过http的返回给直播平台服务器。具体地，先定义一个返回的buffers，函数实现包括：std::vector<boost::asio::const_buffer>buffers。再填充相应的头信息：

[0117] buffers.push_back(boost::asio::buffer(h.name))；

[0118] buffers.push_back(boost::asio::buffer(misc_strings::name_value_separator))。

[0119] 在填充完相应的头信息后，继续填充相应的结果请求数据，具体函数实现如下：buffers.push_back(boost::asio::buffer(content))，其中content即为相应的结果。本申请实施例中，则需要将识别的分数填入到相应结果中。

[0120] 在得到了请求结果后，鉴黄服务器会将鉴黄识别模块识别的分数返回给直播平台服务器，对于直播间内容审核人员则可以得到审核的分数，例如系统可以自动的对识别分数高的认为是违禁的直播间推送给审核人员，从而可以人为的做二次确认，如果确认有问题则可以对直播间进行警告或者处罚。另外，本实施例可以通过自动识别过滤出不需要人为确认的直播间，从而减少审核人员的工作量。

[0121] 实施例3、一种管理多台识别机器的方法

[0122] 另外，实际应用中，对于鉴黄服务框架来说，由于直播间数量比较多，并且对于每一个直播间都是采取每隔预置时长如30秒钟会截取直播间的视频中的一张图片来调用鉴黄服务，那么对于一台识别服务化的机器则满足不了线上的需求。因此需要有多台基于GPU的识别机器(深度学习使用GPU进行大量的计算)。而对于识别服务框架则需要设计一种方

法来依据请求来调度不同的识别机器，同时对于多台机器也可以起到容灾备份的目的。因此本申请实施例还提供了一种管理多台识别机器的方法，具体的实现方法如下所示：

[0123] 步骤1、设计基于每台机器进行编号和IP分配。

[0124] 首先对于每一台鉴黄机器，都具有CPU资源和GPU资源，那么每台机器都可以独立的进行鉴黄图片的识别得到识别的分数。因此首先需要对每台机器进行编号以及分配其固定的IP地址，具体编号则可以按照机器编写序号。例如机器1编写其序号为NO1，然后对于的IP地址则可以分配为192.168.1.1；对于机器2则可以编写序号为NO2，其对应的IP地址也可以分配为192.168.1.2；故对于每一台机器都可以按照此种方法进行编号和分配IP地址，需要注意的是，必须需要保障每台机器的编号都是唯一的，并且IP也是唯一的。

[0125] 步骤2、设计一台机器做为主机器，即master机器。

[0126] 挑选多台机器中的其中一台机器做为master机器，其具有管理其他机器的智能，相当于是一个管理中心。

[0127] 步骤3、设计数据结构来管理所有的机器注册和注销。

[0128] 本文在设计一个内存数据结构来对所有机器的注册和注销进行管理。首先定义一个结构体AiMachine表明是一台机器。

[0129] Struct AiMachine{

[0130] String name;String ip;String no;

[0131] }

[0132] 其中该数据结构包含有机器的名称name，机器的IP地址ip，机器的编号no。定义好机器信息后，接下来本文定义一个管理类Manager来管理机器的注册和注销：

[0133] Class Manager{

[0134] 首先定义一个存储变量来存储所有的机器，本文可以使用STL的list结构来存储机器的结构信息，具体函数实现如下：

[0135] Stl::list<AiMachine>m_list;

[0136] 其中list链表则存储所有注册的机器信息。

[0137] 再编写接口来注册一台机器，具体函数实现如下：

[0138] Void Register(const AiMachine&data){

[0139] 本文则通过调用list链表的push_back来存储机器信息，从而实现了注册一台机器到管理类中，具体函数实现如下：

[0140] m_list.push_back(data)；

[0141] }

[0142] Void UnRegister(cont string&name){

[0143] 需要说明的是，本文还可以通过机器的名称来注销一台机器，包括：

[0144] 首先定义一个迭代器list<AiMachine>::iterator itr;

[0145] 然后对迭代器进行初始化itr＝m_list.begin()；

[0146] 接下来则遍历链表查找该台机器进行注销：

[0147] For(;itr!＝m_list.end()；++itr){

[0148] 需要说明的是，itr!＝m_list.end()表示不是链表的结尾。

[0149] 通过if函数对比机器的名称是否一致，如果一致则说明是该台机器，则通过链表

的删除接口erase来删除该台机器，具体实现函数如下：

```
If ((*itr).name == name){
    m_list.erase(itr);
    Return;
}
```

[0150]

[0151] 需要说明的是，本申请实施例中，还可以通过IP地址来进行注销的接口，具体实现函数如下：

[0152] Void UnRegister(cont string&ip){

[0153] 首先定义一个迭代器，实现函数包括：list<AiMachine>::iterator itr;

[0154] 然后对迭代器进行初始化，实现函数包括：itr＝m_list.begin();

[0155] 接下来则遍历链表查找该台机器进行注销。

[0156] For(;itr!＝m_list.end();++itr){

[0157] 其中itr!＝m_list.end()表示不是链表的结尾。

[0158] 再通过If函数对比机器的ip是否一致，如果一致则说明是该台机器，则通过链表的删除接口erase来删除该台机器，具体实现函数如下：

```
If ((*itr).ip== ip){
    m_list.erase(itr);
    Return; }
}
}
```

[0159]

[0160] 最后，再编写获取实例个数的接口，具体实现函数如下：

[0161] Int Getcount(){

[0162] Return m_list.size();

[0163] }

[0164] }

[0165] 步骤4、设计副机器即非master机器的注册。

[0166] 首先本文挑选了一台机器做为master机器用来管理其他的机器，每台非master机器都有master机器的IP地址，本文设计当非master机器开机启动后，则会请求连接master机器，从而告知master机器现在有一台鉴黄识别机器启动了，需要注册到识别机器群中，使得master后续可以调度识别任务到此台机器中。具体实现本文则可以通过在master创建一个传输控制协议(transmission control protocol,TCP)网络通道，非master机器启动后，创建TCP通道来连接master机器，并告知master该非master机器的名称，非master机器的编号。具体可以设计这样一条协议，使得当非master机器启动后则发送该协议到master机器中，具体的，该协议可为如下协议：

[0167] type@＝register/name@＝machine1/no@＝3/。

[0168] 当master机器收到此协议后，则通过管理类Manager来管理机器的注册。即首先定义一个AiMachine data对象，然后给其进行赋值，具体实现函数如下：

[0169] Data.name＝machine1;data.no＝3;data.ip＝ip;

[0170] 其中ip地址则可以通过链接过来的socket获取到,然后管理类Manager mgr;可以通过其注册方法mgr.register(data)来实现注册。

[0171] 步骤5、设计非master机器的注销。

[0172] 当任意一台非master机器注册后,如果非master机器存在问题或者其他原因可以主动的进行注销,注销后则master机器不会再往其发送图片鉴黄识别任务。具体注销也是通过发送一条注销的协议type@＝unregister/;那么master机器收到后则可以对该台机器进行注销。当master收到此协议后,则通过管理类Manager来管理机器的注销。其中ip地址则可以通过连接过来的socket获取到。然后管理类Manager mgr通过其注册方法mgr.unregister(ip)来实现注销。

[0173] 步骤6、设计非master机器的心跳包活策略。

[0174] 需要说明的是,当任意一台非master机器注册后,如果非master机器由于程序原因导致崩溃或者卡死等其他原因,并且最终其由于有问题所以并不会进行注销操作,这样则会导致其处于假死状态或者机器已经关闭了。那么如果其没有在master机器进行注销,将导致master机器以为此机器一直存在,并不断的发送识别任务,而最终所有的识别任务都失效了。因此需要设计一种心跳保活的策略,非master每隔预置时长如30s则向master机器发送一个心跳包,同时master机器也会给对方回复一个心跳包,从而只要机器存活,则双方不断的发送心跳包,告知对方都存活。当任意一方收不到对方的心跳包后则认为对方出现了问题,则会重现进行注销和注册。可选的,本实施例中,心跳协议则可以设计成type@＝keeplive/,可以理解的是,master机器和非master机器双方都是发送这样的协议内容。

[0175] 实施例4、一种任务的调度方法;

[0176] 另外,对于鉴黄服务框架来说,当直播间同时开播数量比较多时会造成请求非常多,并且很有可能同一时间的请求量非常大,这样会导致很多请求超时得不到处理。那么基于有多台识别服务的机器来说,如何合理的设计使得所有的请求任务不会造成超时或者丢弃,同时对于多台识别机器又能满负荷的进行识别任务,并且不会导致一台识别任务很多,而其他机器又处于空闲状态。基于此本文在将所有的鉴黄识别请求在master机器上设计成了缓存队列的形式,同时在master机器上也设计了负载均衡的调度控制,使得任务尽量均匀的分布到不同的机器上,同时当有机器处于停机或者崩溃时,调度可以将任务分发到其他机器,而不影响整个机器的性能,同时当任务超时或者识别失败时,master机器可以将任务再次发送给其他机器进行识别,防止失败任务丢弃掉。因此本文还提供了一种任务的调度方法,具体的实现方法如下所示:

[0177] 步骤1、设计请求任务的数据结构打包成一个任务。

[0178] 首先对每一个请求,需要将请求进行打包成一个任务,从而后续将任务放到任务队列中,有请求过来时,master机器则将请求打包成任务放入到任务队列中。Master机器后续则从任务队列中不断的取任务来分配到其他识别服务机器上。具体设计如下:

[0179] Struct Task{

[0180] Int Count;Int imagId;Int RoomId;Data imagedata;String imagename;Int score;Int type;

[0181] }

[0182] 收到一个图片识别请求则会将其转换成这样一个结构的对象,即一个这样的结构对象就是一个识别任务,识别完成后,则将结果赋值给此对象的识别分数score。

[0183] 步骤2、设计请求任务的队列。

[0184] 当同一时刻有大量的请求到来时,为了不会丢失请求,本文设计了任务队列。当大量请求来时,先将请求转换成一个识别任务,然后将任务放入到任务队列中,这样任务队列可以将同时并发的请求转换成非并发的请求。Master机器则从任务队列中取出任务来调度识别机器来进行识别。具体实现如下:

[0185] 封装了一个C++语言的类来完成任务队列的存储删除取任务:

[0186] Class TaskQueue{

[0187] 由于队列是不断的进行插入任务和取任务删除任务,所以本文使用STL的list链表来作为任务的队列,具体实现函数如下:

[0188] std::list<Task>lst_buffer;

[0189] 接着对于任务队列为了防止多线程操作,造成队列数据污染,所以需要加入锁来进行多线程的同步互斥,那么定义一个互斥变量Mutex write_mutex和往队列存储任务的接口PostTask,具体实现函数如下:

[0190] Void PostTask(Task t){

[0191] 首先定义一个自动锁对象ScopedLock lock,然后其传入本文定义的互斥变量write_mutex,具体实现函数包括:

[0192] ScopedLock lock(write_mutex);

[0193] 然后调用list的接口push_back来将任务存入到队列的尾端。

[0194] lst_buffer.push_back(t);

[0195] }

[0196] 接下来我们需要编写取任务队列的接口ReadTask,具体实现函数如下:

[0197] Task ReadTask(){

[0198] 类似的,首先定义一个自动锁对象ScopedLock lock,然后其传入本文定义的互斥变量write_mutex。

[0199] ScopedLock lock(write_mutex);

[0200] 然后从队列中取出队列头部的一个任务t:

[0201] Task t=lst_buffer.front();

[0202] 接着则调用接口pop_front将队列头部的任务删除掉:

[0203] lst_buffer.pop_front();

[0204] 然后返回取出的任务:Return t;

[0205] }。

[0206] 步骤3、Master机器创建线程不断的从队列取任务。

[0207] Master机器则需要从队列中不断的获取任务,为后续来进行任务的调度。本文则会创建一个独立的线程来不断的取任务,具体包括:

[0208] 首先创建一个线程对象Thread thread;

[0209] 然后线程对象绑定其执行函数。

[0210] thread(boost::bind(&Run,NULL));

[0211] 绑定后,此线程则会去执行Run函数,接下来编写Run函数:

[0212] Void Run(){

[0213] 需要说明的是,该Run函数是一个while循环来不断的从任务队列中取任务,具体实现如下:

[0214] While(true){

[0215] 首先从队列获取任务t:

[0216] Task t=ReadTask();

[0217] 如果队列中没有任务,则线程会暂停一会,减少CPU的消耗。

[0218] If(t==NULL){

[0219] Sleep(10);

[0220] }

[0221] 如果队列中有任务,则后续会进行任务调度。

[0222] Else{

[0223] }。

[0224] 步骤4、Master机器进行任务调度。

[0225] Master机器从队列中获取到了可以执行的任务后,则需要进行任务的调度。系统中搭建了多个识别服务的机器,并且master机器对识别服务的所有实例进行了管理,那么此时则可以获取到所有存活的实例来进行任务的调度。

[0226] 首先创建一个任务实例的管理对象Manager:Manager mgr;

[0227] 在通过调用接口Getcount()获取现有实例的总数:Int nCount=mgr.Getcount();然后本文的调度策略可以使用随机调度,从现有识别任务中进行随机的调度。从理论上来说,随机调度是可以分布比较均匀的。

[0228] 具体实现如下:

[0229] 依据时间设置随机种子。

[0230] 首先获取当前系统时间,即Time time=Time();

[0231] 然后调用系统函数srand来设置随机种子:srand(time);接着则生成一个随机数nRand:Int nRand=Rand();然后从随机数据中对实例个数求余数,那么余数是谁则落在那个实例上去调度识别任务:Int no=nRand%nCount;从而取出一个任务则会调用一次生成随机数据,来调度任务。再通过序号从识别实例中获取对应的实例:AiMachine ai=Mgr.GetAiMachine(no);然后调用该实例来进行任务的识别。

[0232] 实施例5、一种调度GPU的方法。

[0233] 对于本文的鉴黄识别服务来说,每种图片都会使用GPU资源,而GPU是整个识别中的最核心的资源,同时也是最容易产生瓶颈的资源。那么在整个鉴黄识别服务中,本文则提出了一种充分利用GPU的方案。本文首先会获取系统中GPU显卡的数量,然后对所有的识别任务设计一个任务队列,本文的调度模型中,则会不定时的获取每块GPU显卡当前的使用率,依据使用率,来分配任务到对应的GPU上进行处理。从而防止有的GPU处于空闲,而有的GPU又处于满负荷运行,使得调度更为合理和高效。具体实现方案如下:

[0234] 步骤1、获取系统中GPU数量。

[0235] 首先本文需要获取当前系统GPU的数量,从而为后续GPU的调度提供调度的数量。

具体则通过显卡提供的API函数EnumNvidiaDisplayHandle来枚举当前GPU数量。具体实现如下：

[0236] 设计一个数组来存储所有的GPU的句柄。

[0237] NvPhysicalGpuHandle gpuhandle[Max]＝{0}；

[0238] 本文编写一个循环来遍历GPU,查看其是否存在,如果不存在则说明是最后一个。首先定义一个显卡数量变量,并初始化为0:int nCount＝0;

[0239] 然后编写循环来遍历所有的显卡GPU,具体实现函数如下：

[0240] for(int nIndex＝0;nIndex<0xFFFFFFFF;++nIndex){

[0241] 再通过调用系统提供的API函数EnumNvidiaDisplayHandle来判断当前枚举的GPU是否存在,如果存在返回值则是NvStatus_OK,否则则是失败。如果返回成功,本文则将数量加1,如果返回失败则直接返回nCount值,即表示GPU的数量,具体实现函数如下：

[0242] if(EnumNvidiaDisplayHandle(nIndex,&nvDisplayCardHandle)＝＝NvStatus_OK){

[0243] gpuhandle[nCount]＝nvDisplayCardHandle;

[0244] 如果存在,则将当前GPU的句柄存储到句柄数组中。

```
nCount++;
}else{
```

[0245]

```
        return nCount;
    }
}。
```

[0246] 步骤2、获取系统中每块GPU的使用率。

[0247] 在获取到当前系统GPU的数量和GPU的句柄数组后,需要编写接口来查询每块GPU当前的使用率,从而后续则可以依据使用率来进行任务调度。具体实现如下：

[0248] 首先定义一个数组来存储当前GPU的使用率,int GpuUsage[Max]＝{0}；

[0249] 并且此数组对应于上个步骤中的GPU的句柄数组,接下来编写for循环来遍历GPU的所有句柄：

[0250] 首先定义一个遍历起始下标int nNo＝0,并且在上个步骤中获取到了显卡GPU的数量：

[0251] for(nNo＝0;nNo<nCount;++nNo){

[0252] 接下来则调用系统API函数GPU_GetUsages来获取显卡的使用率,具体实现函数如下：

[0253] GPU_GetUsages(pCardInfo->sGpuInfo[nIndex].nvGpuHandle,pnvUsages)；

[0254] GpuUsage[nNo]＝pnvUsages->nUsage;

[0255] }

[0256] 因此通过循环则可以得到所有GPU的使用率。

[0257] 需要注意的是,由于使用率是一个动态变化的,所以需要通过定时器来不断的获取使用率,从而实时的进行更新。本文则编写一个定时器函数来每间隔预置时长如10秒钟来获取一次使用率,具体实现如下：

[0258] 首先编写一个定时器的回调函数,此函数则是定时器到时会去执行的函数;Void TimerFunc(){

[0259] 再调用上面编写的获取GPU使用率的功能来更新每块显卡的使用率:

[0260] for(nNo＝0;nNo<nCount;++nNo){

[0261] 接下来则调用系统API函数GPU_GetUsages来获取显卡的使用率,具体实现函数如下:

[0262] GPU_GetUsages(pCardInfo->sGpuInfo[nIndex].nvGpuHandle,pnvUsages);

[0263] GpuUsage[nNo]＝pnvUsages->nUsage;

[0264] }

[0265] }

[0266] 接下来调用系统API函数SetTimer来创建一个定时器,其函数原型如下:

[0267] UINT_PTR SetTimer(

[0268] HWND hWnd,

[0269] UINT_PTR nIDEvent,UINT nElapse,

[0270] TIMERPROC lpTimerFunc//

[0271]);

[0272] 其中,hwnd用于表示窗口句柄;nIDEvent用于表示定时器ID,多个定时器时,可以通过该ID判断是哪个定时器;nElapse用于表示时间间隔,该单位为毫秒;lpTimerFunc用于表示回调函数。

[0273] 具体调用则是通过函数SetTimer(NULL,0,10000,TimerFunc);其中窗口句柄设置为NULL,ID则设置为0,时间间隔则设置为10秒即10000毫秒,定时器的回调函数则设置为之前编写的TimerFunc。

[0274] 步骤3、查找GPU使用率最小的编号。

[0275] 在分配任务时,需要获取当前GPU使用率最小的显卡来分配执行任务。具体实现如下:首先本文定义一个当前GPU使用的最小的变量nMinUsage,并且初始化为第一个GPU显卡,Int nMinUsage＝GpuUsage[0];

[0276] 并且定义一个变量来记录当前最小的GPU的编号int nMinNo＝0;

[0277] 接下来则遍历循环查找使用率最小的GPU,需要说明的是,由于第一块已经赋值给nMinUsage,此时循环则从第二块GPU显卡开始。

[0278] for(int nNo＝1;nNo<nCount;++nNo){

[0279] 如果当前遍历的GPU的使用率低于最小的使用率则替换掉当前最小的GPU使用率,并且记录其编号,具体实现函数如下:

[0280] if(GpuUsage[nNo]<nMinUsage){

[0281] nMinUsage＝GpuUsage[nNo];

[0282] nMinNo＝nNo。

[0283] }

[0284] }

[0285] 通过以上查询即可得到当前最小的GPU使用率的编号nMinNo。

[0286] 步骤4、任务调度。

[0287] 本文会对任务使用一个任务队列来缓存所有的识别任务,并且会使用队列中的任务来调度到对应的GPU进行执行。

[0288] 首先创建一个线程对象Thread thread;

[0289] 然后线程对象绑定其执行函数:thread(boost::bind(&Run,NULL))。绑定后,此线程则会去执行Run函数,接下来编写Run函数。

[0290] Void Run(){

[0291] 此函数则是一个while循环来不断的从任务队列中取任务,具体实现函数如下:

[0292] While(true){

[0293] 首先从队列获取任务t:

[0294] Task t=ReadTask();

[0295] 如果队列中没有任务,则线程会暂停一会,减少CPU的消耗,有任务,则会调用之前编写的接口来获取当前GPU使用率最小的GPU的编号,然后从将该任务分配到该GPU进行执行:

[0296] If(t==NULL){

[0297] Sleep(10);

[0298] }

[0299] Else{

[0300] }。

[0301] 上面从图片识别方法的角度对本发明实施例进行了描述,下面从鉴黄服务器的角度对本发明实施例进行描述。

[0302] 请参阅图2,图2为本发明实施例提供的一种可能的鉴黄服务器的实施例示意图,其中,该获取装置具体包括:

[0303] 命名单元201,用于根据直播间的标识信息和时间戳对所述直播间的待识别图片进行命名;

[0304] 存储单元202,用于使用标准模板库STL的链表list设计存储队列,以存储所述直播间的历史图片的特征,所述存储队列包括N张图片,所述直播间的历史图片的特征包括所述历史图片的鉴黄分数结果;

[0305] 收发单元203,用于接收识别请求,所述识别请求用于请求识别目标直播间中的目标待识别图片,所述目标直播间包含于所述直播间;

[0306] 获取单元204,用于响应于所述识别请求,获得与所述目标直播间的房间号对应的所有历史图片的特征;

[0307] 检测单元205,用于调用算法接口similardetect_predict将所述所有历史图片的特征与所述目标待识别图片的特征进行相似度检测;

[0308] 返回单元206,用于若所述算法接口similardetect_predict的输出结果为true,则从所述存储队列中返回所述历史图片的鉴黄分数结果,以作为所述目标待识别图片的鉴黄分数结果。

[0309] 请参阅图3,图3为本发明实施例提供的电子设备的实施例示意图。

[0310] 如图3所示,本发明实施例提供了一种电子设备,包括存储器310、处理器320及存储在存储器320上并可在处理器320上运行的计算机程序311,处理器320执行计算机程序

311时实现以下步骤:根据直播间的标识信息和时间戳对所述直播间的待识别图片进行命名;使用标准模板库STL的链表list设计存储队列,以存储所述直播间的历史图片的特征,所述存储队列包括N张图片,所述直播间的历史图片的特征包括所述历史图片的鉴黄分数结果;接收识别请求,所述识别请求用于请求识别目标直播间中的目标待识别图片,所述目标直播间包含于所述直播间;响应于所述识别请求,获得与所述目标直播间的房间号对应的所有历史图片的特征;调用算法接口similardetect_predict将所述所有历史图片的特征与所述目标待识别图片的特征进行相似度检测;若所述算法接口similardetect_predict的输出结果为true,则从所述存储队列中返回所述历史图片的鉴黄分数结果,以作为所述目标待识别图片的鉴黄分数结果。

[0311] 可选的,在一种可能的实施例中,所述处理器320具体用于:设计结构ImageData存储所述历史图片和所述历史图片的特征,所述结构ImageData包括所述历史图片的原始数据,所述历史图片的名称,所述历史图片的特征数据,和所述历史图片的鉴黄分数结果;根据所述结构ImageData定义链表变量list<ImageData*>listImageData存储各直播间的历史图片的特征。

[0312] 可选的,在一种可能的实施例中,所述处理器320具体用于:确定与所述目标直播间的房间号对应的目标链表,所述目标链表包括所述目标直播间的历史图片;遍历所述目标链表以获得所述所有历史图片的特征。

[0313] 可选的,在一种可能的实施例中,所述处理器320具体用于:定义迭代器Map<int,list<ImageData*>*>::iterator itr;调用map容器的find函数itr=mapListData.find(nRoomId)查找与所述目标直播间的房间号对应的目标链表,所述nRoomId用于表示所述目标直播间的房间号,所述find函数返回的结果为所述目标链表。

[0314] 可选的,在一种可能的实施例中,所述处理器320具体用于:定义所述目标链表的迭代器;通过函数itr=listImageData.begain()初始化所述目标链表的迭代器,所述itr用于表示所述目标链表的迭代器;通过for循环函数遍历所述目标链表的迭代器,并调用所述算法接口similardetect_predict的函数Int Result=similardetect_predict((*itr).ImageFeature,ImageFeature)将所述所有历史图片的特征与所述目标待识别图片的特征进行相似度检测,其中,所述(*itr).ImageFeature用于表示所述历史图片的特征,所述ImageFeature用于表示所述目标待识别图片的特征。

[0315] 可选的,在一种可能的实施例中,所述处理器320还用于:若所述算法接口similardetect_predict的输出结果为false,则将存在时间最久的历史图片的特征删除,并将所述目标待识别图片的特征更新到所述存储队列中。

[0316] 可选的,在一种可能的实施例中,所述处理器320具体用于:调用函数listImageData.pop_front()将所述存在时间最久的历史图片的特征删除;调用函数listImageData,push_back(ImageData)将所述目标待识别图片的特征存储到所述目标链表中。

[0317] 由于本实施例所介绍的电子设备为实施本发明实施例中一种鉴黄服务器所采用的设备,故而基于本发明实施例中所介绍的方法,本领域所属技术人员能够了解本实施例的电子设备的具体实施方式以及其各种变化形式,所以在此对于该电子设备如何实现本发明实施例中的方法不再详细介绍,只要本领域所属技术人员实施本发明实施例中的方法所

采用的设备，都属于本发明所欲保护的范围。

[0318] 请参阅图4，图4为本发明实施例提供的一种计算机可读存储介质的实施例示意图。

[0319] 如图4所示，本实施例提供了一种计算机可读存储介质400，其上存储有计算机程序411，该计算机程序411被处理器执行时实现如下步骤：根据直播间的标识信息和时间戳对所述直播间的待识别图片进行命名；使用标准模板库STL的链表list设计存储队列，以存储所述直播间的历史图片的特征，所述存储队列包括N张图片，所述直播间的历史图片的特征包括所述历史图片的鉴黄分数结果；接收识别请求，所述识别请求用于请求识别目标直播间中的目标待识别图片，所述目标直播间包含于所述直播间；响应于所述识别请求，获得与所述目标直播间的房间号对应的所有历史图片的特征；调用算法接口similardetect_predict将所述所有历史图片的特征与所述目标待识别图片的特征进行相似度检测；若所述算法接口similardetect_predict的输出结果为true，则从所述存储队列中返回所述历史图片的鉴黄分数结果，以作为所述目标待识别图片的鉴黄分数结果。

[0320] 可选的，在一种可能的实施例中，该计算机程序411被处理器执行时具体用于实现如下步骤：设计结构ImageData存储所述历史图片和所述历史图片的特征，所述结构ImageData包括所述历史图片的原始数据，所述历史图片的名称，所述历史图片的特征数据，和所述历史图片的鉴黄分数结果；根据所述结构ImageData定义链表变量list<ImageData*>listImageData存储各直播间的历史图片的特征。

[0321] 可选的，在一种可能的实施例中，该计算机程序411被处理器执行时具体用于实现如下步骤：确定与所述目标直播间的房间号对应的目标链表，所述目标链表包括所述目标直播间的历史图片；遍历所述目标链表以获得所述所有历史图片的特征。

[0322] 可选的，在一种可能的实施例中，该计算机程序411被处理器执行时具体用于实现如下步骤：定义迭代器Map<int,list<ImageData*>*>::iterator itr；调用map容器的find函数itr＝mapListData.find(nRoomId)查找与所述目标直播间的房间号对应的目标链表，所述nRoomId用于表示所述目标直播间的房间号，所述find函数返回的结果为所述目标链表。

[0323] 可选的，在一种可能的实施例中，该计算机程序411被处理器执行时还用于实现如下步骤：定义所述目标链表的迭代器；通过函数itr＝listImageData.begain()初始化所述目标链表的迭代器，所述itr用于表示所述目标链表的迭代器；通过for循环函数遍历所述目标链表的迭代器，并调用所述算法接口similardetect_predict的函数Int Result＝similardetect_predict((*itr).ImageFeature,ImageFeature)将所述所有历史图片的特征与所述目标待识别图片的特征进行相似度检测，其中，所述(*itr).ImageFeature用于表示所述历史图片的特征，所述ImageFeature用于表示所述目标待识别图片的特征。

[0324] 可选的，在一种可能的实施例中，该计算机程序411被处理器执行时还用于实现如下步骤：若所述算法接口similardetect_predict的输出结果为false，则将存在时间最久的历史图片的特征删除，并将所述目标待识别图片的特征更新到所述存储队列中。

[0325] 可选的，在一种可能的实施例中，该计算机程序411被处理器执行时具体用于实现如下步骤：调用函数listImageData.pop_front()将所述存在时间最久的历史图片的特征删除；调用函数listImageData,push_back(ImageData)将所述目标待识别图片的特征存储

到所述目标链表中。

[0326] 需要说明的是，在上述实施例中，对各个实施例的描述都各有侧重，某个实施例中没有详细描述的部分，可以参见其它实施例的相关描述。

[0327] 本领域内的技术人员应明白，本发明的实施例可提供为方法、系统、或计算机程序产品。因此，本发明可采用完全硬件实施例、完全软件实施例、或结合软件和硬件方面的实施例的形式。而且，本发明可采用在一个或多个其中包含有计算机可用程序代码的计算机可用存储介质(包括但不限于磁盘存储器、CD-ROM、光学存储器等)上实施的计算机程序产品的形式。

[0328] 本发明是参照根据本发明实施例的方法、设备(系统)、和计算机程序产品的流程图和/或方框图来描述。应理解可由计算机程序指令实现流程图和/或方框图中的每一流程和/或方框、以及流程图和/或方框图中的流程和/或方框的结合。可提供这些计算机程序指令到通用计算机、专用计算机、嵌入式计算机或者其他可编程数据处理设备的处理器以产生一个机器，使得通过计算机或其他可编程数据处理设备的处理器执行的指令产生用于实现在流程图一个流程或多个流程和/或方框图一个方框或多个方框中指定的功能的装置。

[0329] 这些计算机程序指令也可存储在能引导计算机或其他可编程数据处理设备以特定方式工作的计算机可读存储器中，使得存储在该计算机可读存储器中的指令产生包括指令装置的制造品，该指令装置实现在流程图一个流程或多个流程和/或方框图一个方框或多个方框中指定的功能。

[0330] 这些计算机程序指令也可装载到计算机或其他可编程数据处理设备上，使得在计算机或其他可编程设备上执行一系列操作步骤以产生计算机实现的处理，从而在计算机或其他可编程设备上执行的指令提供用于实现在流程图一个流程或多个流程和/或方框图一个方框或多个方框中指定的功能的步骤。

[0331] 尽管已描述了本发明的优选实施例，但本领域内的技术人员一旦得知了基本创造概念，则可对这些实施例作出另外的变更和修改。所以，所附权利要求意欲解释为包括优选实施例以及落入本发明范围的所有变更和修改。

[0332] 显然，本领域的技术人员可以对本发明进行各种改动和变型而不脱离本发明的精神和范围。这样，倘若本发明的这些修改和变型属于本发明权利要求及其等同技术的范围之内，则本发明也意图包括这些改动和变型在内。

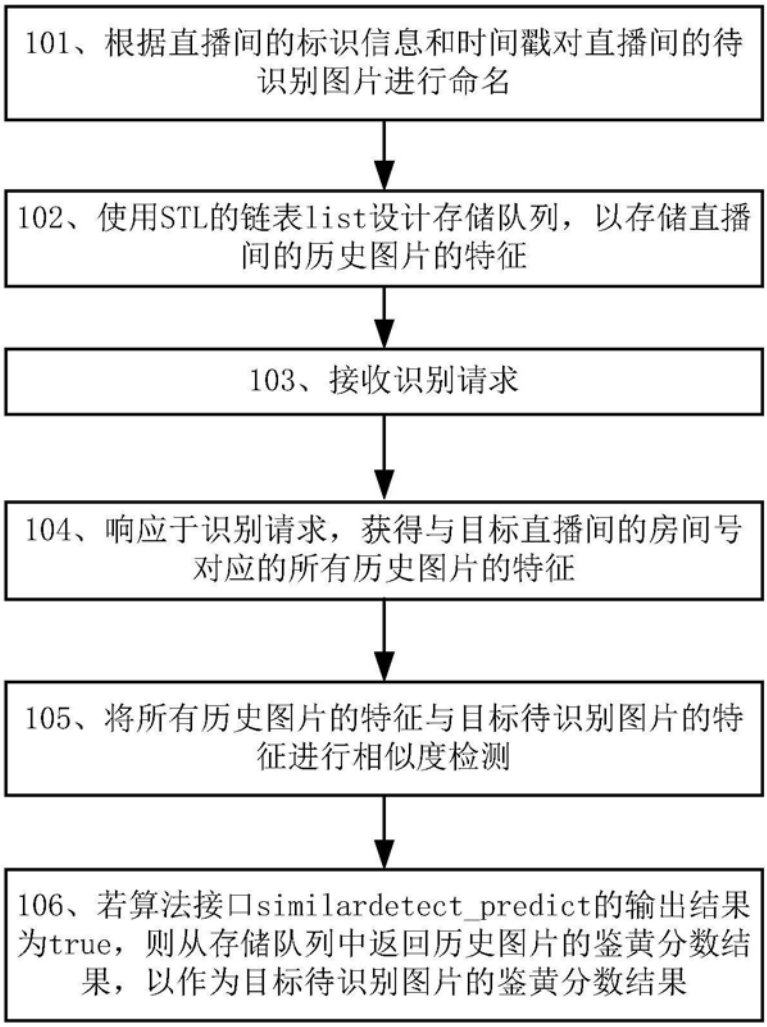

101、根据直播间的标识信息和时间戳对直播间的待识别图片进行命名
102、使用STL的链表list设计存储队列，以存储直播间的历史图片的特征
103、接收识别请求
104、响应于识别请求，获得与目标直播间的房间号对应的所有历史图片的特征
105、将所有历史图片的特征与目标待识别图片的特征进行相似度检测
106、若算法接口similardetect_predict的输出结果为true，则从存储队列中返回历史图片的鉴黄分数结果，以作为目标待识别图片的鉴黄分数结果

图1

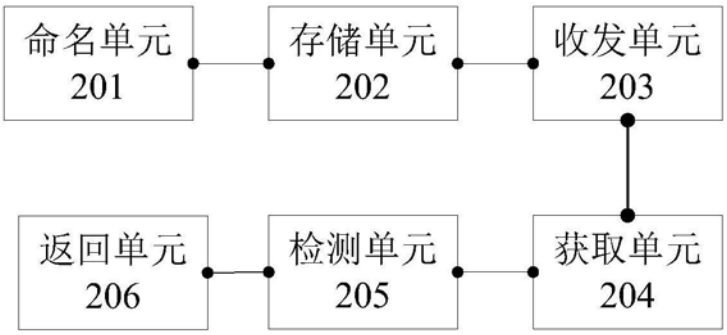

命名单元
201
存储单元
202
收发单元
203
获取单元
204
检测单元
205
返回单元
206

图2

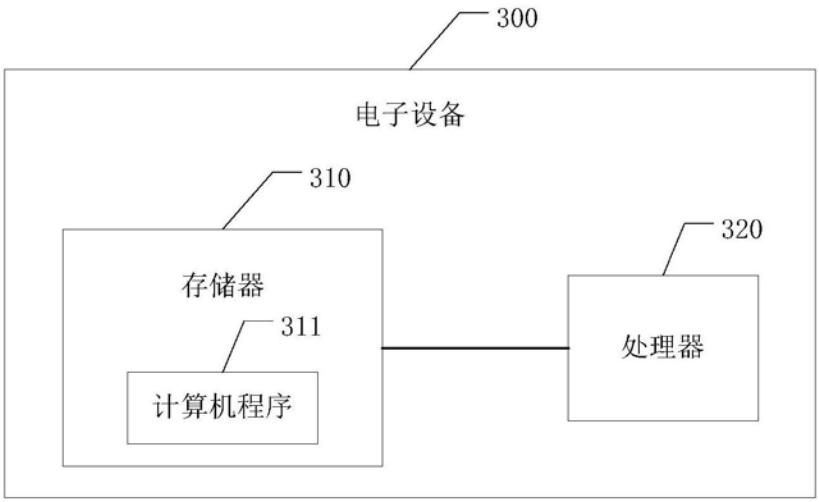


图3

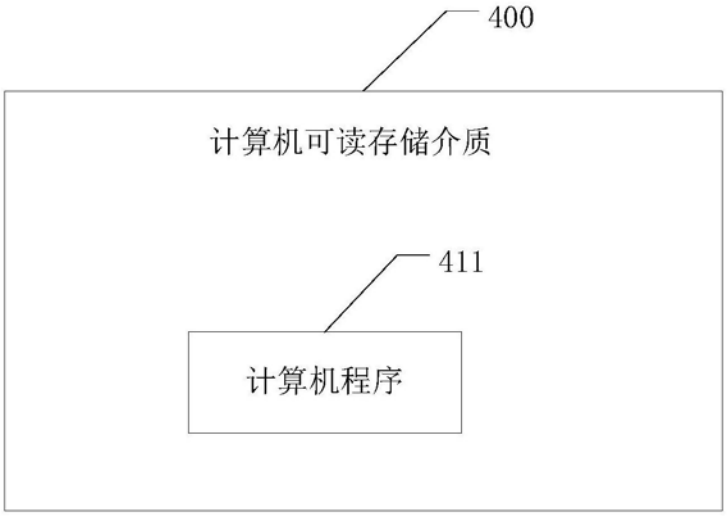


图4