特效能力API开发中的背景分割

目录

[1依赖软件列表介绍 1](#_Toc62141191)

[2 服务器需求 1](#_Toc62141192)

[3 组件安装 2](#_Toc62141193)

[anaconda环境安装（在本地环境配置）： 2](#_Toc62141194)

[python环境安装（在本地环境配置） 2](#_Toc62141195)

[docker中使用tensorflow-serving（远程服务器环境配置） 2](#_Toc62141196)

[将模型部署到服务器中（远程服务器环境配置） 4](#_Toc62141197)

[4 通信设计 4](#_Toc62141198)

[5 程序安装运行 5](#_Toc62141199)

[6 代码说明 5](#_Toc62141200)

[7 QA 5](#_Toc62141201)

# 1依赖软件列表介绍

《特效能力API开发中的背景分割》项目涉及的第三方软件包括：

Tensorflow serving，Python及相关库

涉及安装包包括：

Docker

Tensorflow serving

Python 3.7 (所需安装库在requirements.txt中展现)

# 2 服务器需求

服务器部署相关详细信息如下：

|  |  |  |
| --- | --- | --- |
| 服务器名称 | 服务器描述 | 服务器数量 |
| 图像分割模型服务器 | 需要部署到基于Ubuntu18.04 LTS 64位的服务器中，至少为单核心，内存2GB以上，带宽1Mbps以上 | 1台 |

# 3 组件安装

以下步骤都是基于Ubuntu系统的操作步骤，分为本地环境配置和远程服务器环境配置。

## anaconda环境安装（在本地环境配置）：

anaconda是python的环境管理器，能够管理不同的版本python文件库与依赖，起到隔离沙盒的作用。

前往anaconda官网，下载[Ananconda installer for Linux](https://www.anaconda.com/products/individual#linux)。

运行安装脚本：

bash ~/Downloads/Anaconda3-2020.02-Linux-x86\_64.sh

点击同意license agreement之后，“Do you wish the installer to initialize Anaconda3 by running conda init?”，选择“yes”。

安装完成之后会显示：“Thank you for installing Anaconda<2 or 3>!”

## python环境安装（在本地环境配置）

采用python3.7，创建名称为“python37\_seg”的环境名称：

conda create -n python37\_seg python=3.7

进入创建的环境：

conda activate python37\_seg

进入项目所在的目录，安装requirement.txt的依赖

pip install -r requirements.txt

保留这个打开的窗口，不要关闭

## docker中使用tensorflow-serving（远程服务器环境配置）

docker的安装：

更新apt包目录索引：

sudo apt-get update

下载相应包：

sudo apt-get install \

apt-transport-https \

ca-certificates \

curl \

gnupg-agent \

software-properties-common

添加Docker的官方GPG key：

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

确认你目前在使用的key：

sudo apt-key fingerprint 0EBFCD88

应该输出类似信息：

pub   rsa4096 2017-02-22 [SCEA]

      9DC8 5822 9FC7 DD38 854A  E2D8 8D81 803C 0EBF CD88

uid           [ unknown] Docker Release (CE deb) <docker@docker.com>

sub   rsa4096 2017-02-22 [S]

设置使用稳定版本的docker库：

sudo add-apt-repository \

   "deb [arch=amd64] https://download.docker.com/linux/ubuntu \

   $(lsb\_release -cs) \

   stable"

先更新库索引之后下载使用docker：

sudo apt-get update

sudo apt-get install docker-ce docker-ce-cli containerd.io

确认docker正确安装，运行hello world实例程序，如果docker正确运行，那么会输出hello world文本。

sudo docker run hello-world

在docker中安装tensorflow serving：

docker pull tensorflow/serving

使用以下命令可以查看已经安装的镜像：

docker image ls

使用以下命令查看运行或者曾经运行的容器：

docker ps -a

## 将模型部署到服务器中（远程服务器环境配置）

首先将模型文件目录“export”上传到服务器的/home/user/目录下（user是你自己的用户名称）

执行代码，监听8501端口，使用restful api:

docker run -p 8501:8501 \

--mount type=bind,\

source=/home/user/export,\

target=/models/export/ \

-e MODEL\_NAME=export -t tensorflow/serving &

如果出现event loop的输出，说明接口监听成功打开。

# 4 通信设计

通信设计流程如下：

# 5 程序安装运行

在组件部署安装完毕之后，在本地项目目录运行：

python client\_rest.py

端口信息：8501，restful api的通信端口。

具体通信内容与代码查看client\_rest当中注释。

# 6 代码与接口说明

这部分介绍核心代码参数设定、接口定义等，为二次开发提供一个指导和说明。

* test\_images

**描述：**这个文件夹包括需要测试的图片。

* client\_rest.py

**描述：**包括了和服务器通信的核心代码，也包括图像的后处理部分（滤波操作、图片的尺寸调整等）

**关键变量说明：**

|  |  |  |
| --- | --- | --- |
| **变量名称** | **类型与作用域** | **描述** |
| path | string/全局 | 被分割的肖像路径 |
| pathBack | string/全局 | 替换的背景 |
| body | dict/全局 | 访问的数据体 |
| mask | np.array/全局 | 生成的肖像图掩码数据 |
| imgx | np.array/全局 | 读取的肖像图数据 |
| background | np.array/全局 | 读取的背景图数据 |
| combine\_img | np.array/全局 | 生成的分割合并结果 |

**接口（代码）使用说明：**

无特别说明，修改path与pathBack路径的值来指向图片路径。

* utils.py

**描述：**图像后处理的一些函数

**关键变量说明：**无

**接口（代码）使用说明：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **函数名** | **描述** | **参数** | **返回值** | **可能的异常** |
| resize\_image | 改变图片大小通过opencv并根据网络中的核卷积大小设定被分割图像尺寸 | **img**:输入图像，**ndarray**格式。 **reshape\_size**:图像大小尺寸设定。 | 返回被opencv处理过尺寸的图片（np.array） | 可能的IOError,修改输入的图片格式 |
| padding | 填充图片边缘  如果图像不满足核维度，将图片边缘填充到指定维度 | **img**:输入图像，ndarray格式，type为unit8 **dimension**:核维度，此处模型应为512，默认224  **channel**:图像通道数 **show\_pad**:是否显示图片填充边缘 | 返回填充过后的图片（np.array） | / |
| resize\_background  \_image | 根据需要截取的人像调整背景图像的大小  如果原图像尺寸大于人像尺寸那么直接返回背景图片，  如果小于则调整到与背景图片一致的大小 | **img**:输入的人像图像  **back\_img**:输入的背景图像 | 处理过的图像 | / |

# 7 QA

1. 在项目目录下的export的“1”目录是什么意思呢？

这个是默认的版本号，tensorflow serving根据此选择最大的版本进行部署，1代表第一个版本（version），除此之外tensorflow serving支持版本的热更新，在不停机的情况下，上传对应新的目录，tensorflow会自动部署，例如再上传一个模型在“2”这个目录下。

1. 服务器处理通信的很慢怎么办？

这个主要受限于服务器的带宽、处理性能等要素，在1Mbps单核2GB的低性能服务器上，将模型部署于服务器上，处理4k分辨率的图片，所用时间大概是40s左右（包括下载和上传时间）。通过性能分析，如果将图片部署于本地实验室服务器上，通信时间大大减少，可以将时间缩减到数秒之内。对于服务器提升方面，首要应该考虑的是带宽的大小。