软件需求规格说明书

基于Flask的深度学习自动化部署系统

|  |  |  |  |
| --- | --- | --- | --- |
| 编写 | 许京爽，张文斌，聂磊 | 日期 | 2020年3月22日 |
| 校对 | 聂磊 | 日期 | 2020年3月25日 |

北京航空航天大学 计算机学院

软件工程综合实验课 C组

二〇二〇年三月二十五日

**版本变更历史**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 版本号 | 日期 | 修改章节 | 修改说明 | 修改人 | 审核人 |
| 1.0 | 2020.3.25 |  | 完成第一版 | 许京爽，张文斌，聂磊 | 聂磊 |

**目录**

[1 引言 1](#_Toc36119471)

[1.1 文档概述 1](#_Toc36119472)

[1.2 背景 1](#_Toc36119473)

[1.3 术语和缩略语 1](#_Toc36119474)

[1.4 参考资料 2](#_Toc36119475)

[2 任务概述 3](#_Toc36119476)

[2.1 目标 3](#_Toc36119477)

[2.2 用户特点 3](#_Toc36119478)

[2.3 假定与约束 3](#_Toc36119479)

[3 需求 4](#_Toc36119480)

[3.1 业务需求 4](#_Toc36119481)

[3.2 功能需求 4](#_Toc36119482)

[3.3 非功能需求 5](#_Toc36119483)

[4 运行环境规定 6](#_Toc36119484)

[4.1 硬件资源 6](#_Toc36119485)

[4.2 软件资源 6](#_Toc36119486)

[4.3 通信接口 6](#_Toc36119487)

[5. 合格性规定 6](#_Toc36119488)

[5.1 软件测试规定 6](#_Toc36119489)

[5.2 软件演示规定 7](#_Toc36119490)

[附录A Flask框架分析 7](#_Toc36119491)

[A.1 框架介绍 7](#_Toc36119492)

[A.2 框架核心 7](#_Toc36119493)

[A.3 Flask通用扩展包 8](#_Toc36119494)

# 1 引言

## 1.1 文档概述

本文档是C组开发项目《基于Flask的深度学习自动化部署系统》的需求规格说明书，通过对项目预期功能的评估，在明确定义和用户需求的基础上，进一步确定软件的业务需求、功能需求、非功能需求，以及软件对运行环境的要求。

本轮需求分析过程以Flask开源框架及相关参考资料为输入，分析软件设计需求，结合软件工程综合实验具体要求，输出软件需求规格说明书。

本文档是为软件设计人员、开发人员、测试人员、项目验收方编写的。可作为后续软件架构设计的依据，软件开发过程的参考材料，测试过程的评价指标，以及项目验收达标的依据。

## 1.2 背景

Flask是Python常用的web开发框架，因其具有很多优点而受到广泛的使用。它非常的灵活、轻便、安全且容易上手，能够很好地结合MVC模式进行开发，具有高度定制性，用户可以根据自己的需求来添加相应的功能，在保持核心结构简单的同时，实现功能的丰富与扩展。因其具有强大的插件库，可以让用户实现个性化的网站定制，开发出功能强大的网站。

## 1.3 术语和缩略语

本需求分析文档所用到的专业术语与缩略语如表1所示。

**表1 专业术语及缩略语**

|  |  |
| --- | --- |
| 术语/缩略语 | 描述 |
| Web应用框架 | Web应用框架（Web application framework）是一种开发框架，用来支持动态网站、网络应用程序及网络服务的开发，有助于减轻网页开发时共通性活动的工作负荷。 |
| Python | Python是一种跨平台的计算机程序设计语言，具有面向对象和动态类型的特性。Python最初被设计用于编写自动化脚本，随着版本的不断更新和语言新功能的添加，越多被用于独立的、大型项目的开发。 |
| WSGI | Web服务器网关接口（Python Web Server Gateway Interface，缩写为WSGI）是为Python语言定义的Web服务器和Web应用程序或框架之间的一种简单而通用的接口。 |
| 深度学习 | 深度学习是机器学习领域中一个新的研究方向，通过学习样本数据的内在规律和表示层次，让机器能够像人一样具有分析学习能力，能够识别文字、图像和声音等数据。 |
| 模型部署 | 将训练完毕的深度学习模型放置在特定硬件平台上，辅以合适的软件环境，使该模型能够被用户使用。 |
| HTTP | 超文本传输协议（英语：HyperText Transfer Protocol，缩写：HTTP）是一种用于分布式、协作式和超媒体信息系统的应用层协议。HTTP是万维网的数据通信的基础。 |
| REST API | 表现层状态转换（英语：Representational State Transfer，缩写：REST）是一种万维网软件架构风格，目的是便于不同软件/程序在网络中互相传递信息。表现层状态转换是根基于超文本传输协议（HTTP）之上而确定的一组约束和属性，是一种设计提供万维网络服务的软件构建风格。 |
| GET请求 | 向指定服务器发出“显示”请求，通常用于读取服务器中的数据。 |
| POST请求 | 向指定服务器提交数据，请求服务器进行处理（例如提交表单或者上传文件）。 |

## 1.4 参考资料

本文档的撰写过程中参照了如下标准：

GB/T 9385-2008 计算机软件需求规格说明

GB/T 8567-2006 计算机软件文档编制规范

GJB 438B-2009 军用软件开发文档通用要求

# 2 任务概述

## 2.1 目标

高效的深度学习模型需要通过实际部署才能在生产生活中发挥作用。常用的部署方式是将模型包装成Web服务，向外界提供基于HTTP协议的接口，从而提供各类预测功能和服务。对于算法研究人员而言，将深度学习模型部署为Web服务的学习成本较高，通常需要工程开发人员协助部署，这一定程度上降低了算法研发和实际部署的效率。本项目旨在通过自动化部署深度模型，降低算法研究人员实际部署模型的门槛，提高深度学习模型迭代研发和部署的效率。

考虑到语言和部署效率的因素，项目将采用基于Python语言的轻量级Web框架flask和容器技术作为实际的模型部署工具，同时构建一套自动化部署系统供用户使用，包括了前端交互页面，后端系统和数据库三部分，用户上传模型文件并通过界面操作即可完成模型的部署，无需使用者具备Web开发和部署的相关知识和经验。本系统将为算法研究人员提供高效可扩展的深度学习模型部署服务。

## 2.2 用户特点

本软件系统的用户应当具有深度学习的相关理论和实践知识，能够正确地将待解决的问题以恰当的深度学习模型进行描述，并将训练完毕的神经网络模型上传至本系统。

此外，用户通过本系统完成模型的部署后，模型的使用方应当能够正确地通过REST API访问本软件提供的服务接口，以上传待验证数据（例如照片信息），并获取神经网络的输出（例如物体识别的结果）。

## 2.3 假定与约束

本软件系统假定用户待部署的神经网络模型的功能为静态图像识别。用户将训练完毕的模型通过网页上传至本系统后，本系统将模型部署在自身服务器上，并返回一个REST API给用户。

用户可在需要使用该模型的恰当场景下调用此REST API，通过POST请求将待识别的图片通过API发送至服务器，服务器将通过HTTP响应返回图片的识别结果。

# 3 需求

## 3.1 业务需求

本项目是基于Flask框架搭建的深度学习模型自动化部署系统，主要面向的是具有模型部署需求的算法研究人员，为他们提供方便快捷的模型部署服务。

用户的需求是希望通过上传训练好的模型能够实现“一键部署”。通过分析，我们总结出了三个功能需求：模型上传，自定义配置和模型部署。用户可以通过模型上传功能将训练好的模型文件上传到系统中。之后用户通过自定义配置功能对模型的部署进行相关个性化的配置。模型部署功能允许用户对上传的模型进行“一键部署”，部署成功后会返回给用户模型服务的API供用户调用。

1. 模型上传

系统应当提供给用户上传模型文件的接口，同时还需要支持多种格式的模型文件的上传。由系统组织并存储这些模型文件，用户不需要关心具体的存储位置和方式，系统确保上传成功的模型文件存储可靠性，供后续部署使用。

1. 自定义配置

不同模型需要的运行环境和依赖是不同的，为了更好的支持用户部署，系统应支持用户对模型部署时的配置项进行自定义设置。包括运行时环境、第三方库、编译环境、副本数量等。

1. 模型部署

在上传模型，自定义配置之后，系统要支持用户“一键部署”，“多次部署”。具体来说，系统需要提供便捷的部署接口，例如部署按钮等，用户通过点击或提交表单等方式便能够部署模型。此外，还需要支持同一个模型部署多个实例。

1. 模型使用

用户完成模型的部署后，系统允许用户在合适的场景访问服务器的API，将待检测图片发送至服务器，服务器运行神经网络模型后，将图片检测结果返回给用户。

## 3.2 功能需求

1. 注册登录

用户可在本软件系统提供的服务网页上注册用户，注册过程中需提供用户名、密码。用户完成注册后，可通过注册时登记的用户名和密码执行登录操作，以此获得部署任务的创建权限，以及对神经网络模型的上传、管理、部署权限。

1. 创建任务

用户可通过在页面中点击“创建任务“按钮，建立一个新的模型部署任务。

1. 上传模型

用户完成任务的创建后，可通过点击网页的“上传模型“按钮，选择本地已经训练好的神经网络模型，将其上传至服务器。

1. 自定义配置

用户完成任务的创建后，可在任务管理页面中，自定义模型的运行时环境、第三方库、编译环境、副本数量。

1. 部署模型

用户完成模型的上传和自定义配置后，可通过点击“开始部署“按钮，完成模型部署的过程，并获得服务器返回的REST API.

1. 使用模型

用户可通过访问步骤e)获得的REST API使用已经上传好的模型，并获取模型的运行结果。

## 3.3 非功能需求

1. 兼容性

作为一个模型自动化部署工具，客户的服务器类型，软件环境多种多样，本系统需要兼容不同的部署环境。系统需要将自身的依赖项以及代码打包完整，采用容器技术部署在用户的服务器资源上，这样不会影响用户服务器本身的环境和在运行的程序。

1. 可扩展性

随着用户部署的模型服务数量增多，用户可能需要新增服务器资源。系统需要支持这种可插拔的扩展方式，同一调度服务器资源，均衡部署模型服务。

1. 容灾性

系统需要对某一个服务器的宕机所导致的服务不可用做出响应。具体来说，某一服务器节点的宕机会导致部署在其上的模型服务失效，为了确保用户部署的模型服务的可用性，系统需要监控每个服务器节点的状态，并在节点失效时对其上的模型服务进行重新部署，以保证模型服务的可用性。

# 4 运行环境规定

## 4.1 硬件资源

运行本软件系统所需的最小硬件资源：

Intel Core i5 9600K或同等级CPU；

2 GB 随机存储器（RAM）；

40 GB可用硬盘空间。

## 4.2 软件资源

运行本软件系统所需的软件资源：

类Unix操作系统；

Docker；

Kubernetes.

## 4.3 通信接口

运行本软件系统所需的通信接口：

1000Mbps互联网连接；

TCP/IP协议栈。

# 5. 合格性规定

## 5.1 软件测试规定

软件在测试过程中应当符合以下规定：

1. 在单元测试时，网页前端、网页后端、部署系统这三个独立的单元模块能够实现预期功能；
2. 在系统测试时，软件能够长时间运行在服务器上，接受用户的访问请求，平均无故障时间大于300小时。

## 5.2 软件演示规定

在软件的演示过程中，应当表现出如下过程：

1. 演示者访问软件的服务提供网页，在网页上注册并登录账户；
2. 演示者登录账户后，点击“创建任务“按钮，新建一个部署任务，并正确完成已训练模型的上传和自定义配置；
3. 演示者点击“部署模型”按钮，获得访问该模型的REST API；
4. 演示者通过调用此REST API上传待验证图片，并获得模型的预期运行结果。

# 附录A Flask框架分析

## A.1 框架介绍

Flask是当下流行的Web框架，它是用Python实现的。Flask最显著的特点包括它是一个“微”框架(microframework)。“微”意味着Flask旨在保持核心功能的简单，因此它又是一个易于扩展的框架。

在“微”框架的情况下，Flask不包含数据库抽象层、表单验证，或者是其它任何已有多种库可以胜任的功能。然而，Flask支持用扩展来给本项目添加这些功能，无需局限于特定的框架功能，让该项目更具有个性化的特征。众多Flask扩展包提供了数据库集成、表单验证、上传处理以及各种各样的开放认证技术等功能，甚至可以根据项目需求自行开发一些基于Flask框架的功能包，具有极强的拓展性和定制性，同时Flask可以很好地结合MVC模式进行开发，开发人员分工合作，小型的团队在短时间内就可以完成功能丰富的中小型网站或Web服务的实现。这些特性，使得它在Web开发方面变得非常流行，也成为本项目的首选框架。

## A.2 框架核心

Flask框架的核心包括两个部分，Werkzeug和Jinja2，Werkzeug是一个WSGI(Python Web Server Gateway Interface)的工具包，它可以作为一个Web框架的底层库，其中封装了很多Web服务的工具，包括最重要的会话模块的Response、Resquest等。

Jinja2是Python语言的一个通用模版引擎，具有一系列强大的功能的同时还具有灵活性、快速性、安全性的优点，因此被广泛使用。其灵活性主要体现在Jinja2提供了控制结构、表达式和继承等，快速性在于其仅有控制结构，不允许在模版中编写太多业务逻辑，可读性很强。

## A.3 Flask通用扩展包

Flask作为一个微框架，具有极强的可扩展性，因此下面介绍本项目中会用到的Flask的扩展插件。

1. Flask-SQLAlchemy数据库扩展包

数据库是长期存储在计算机内，大量有组织可共享的数据的集合。SQLAlchemy是一个关系型数据库框架，它提供了高层的ORM(Object Relational Mapping)和底层的原生数据库的操作， 实际上是对数据库的抽象，让开发者不用直接和 SQL 语句打交道，而是通过 Python对象来操作数据库，在舍弃一些性能开销的同时，换来的是开发效率的较大提升。而Flask-SQLAlchemy则是一个基于SQLAlchemy操作的flask扩展包。

1. Flask-WTF表单扩展包

Flask\_WTF是Flask框架的表单验证模块，可以很方便地生成表单，也可以当做json数据交互的验证工具，支持热插拔。Flask-WTF其实是对wtforms组件的封装，而wtfroms其实是一个支持多种web框架的form组件，主要用于对用户请求数据的进行验证，依照功能类别来说wtforms分别由表2所示的几个类别构成。

**表2 wtforms的构成**

|  |  |
| --- | --- |
| Forms | 主要用于表单验证、字段定义、HTML生成，并把各种验证流程聚集在一起进行验证 |
| Fields | 主要负责渲染(生成HTML)和数据转换 |
| Validator | 主要用于验证用户输入的数据的合法性。比如Length验证器可以用于验证输入数据的长度 |
| Widgets | html插件，允许使用者在字段中通过该字典自定义html小部件 |
| Meta | 用于使用者自定义wtforms功能，例如csrf功能开启 |
| Extensions | 丰富的扩展库，可以与其他框架结合使用 |