

**本科生毕业设计[论文]**

**基于Docker技术构建机器学习PaaS平台**

院 系 光学与电子信息学院

专业班级 微电子科学与工程1401

姓 名 张超

学 号 U201414176

指导教师 邓前松

2018 年 6 月 08 日

**学位论文原创性声明**

本人郑重声明：所呈交的论文是本人在导师的指导下独立进行研究所取得的研究成果。除了文中特别加以标注引用的内容外，本论文不包括任何其他个人或集体已经发表或撰写的成果作品。本人完全意识到本声明的法律后果由本人承担。

作者签名： 年 月 日

**学位论文版权使用授权书**

本学位论文作者完全了解学校有关保障、使用学位论文的规定，同意学校保留并向有关学位论文管理部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权省级优秀学士论文评选机构将本学位论文的全部或部分内容编入有关数据进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

本学位论文属于 1、保密囗，在 年解密后适用本授权书

2、不保密囗 。

（请在以上相应方框内打“√”）

作者签名： 年 月 日

导师签名： 年 月 日

# 摘 要

近些年，越来越多的科研工作者和工程师开始从事机器学习的研究和开发，而机器学习的开发流程又十分复杂且对计算资源性能要求又非常高，需要投入大量人力和云计算服务租赁成本。本文针对这一现状，设计并开发了基于Docker技术构建的功能完善、使用便捷、十分灵活的机器学习PaaS平台。其主要特色如下：

（1）国内首次提出并设计实现了专用于机器学习开发的PaaS平台，该平台采用主流的WEB技术构建，成本低、拓展性强且功能强大。

（2）开创性地把Docker集群技术用于该PaaS平台，实现了云计算资源的弹性分配和高效使用，解决了云计算资源闲置和浪费的问题，大大降低了用户的云平台计算资源的租赁支出。

（3）创造性地运用了Heroku和Google App Engine架构的设计理念，简化了器学习开发流程，设计并实现了一个功能完善、使用便捷，性价比高的机器学习PaaS平台。

**关键词：**机器学习；云平台；PaaS；Docker；WEB技术；功能完善

# Abstract

In recent years, more and more researchers and engineers have begun to engage in the research and development of machine learning, and the development process of machine learning is very complicated and requires very high computational resource performance, so it requires a lot of manpower and cost, In view of this phenomenon, this paper designs and develops a machine learning PaaS platform based on Docker technology.

The platform uses Docker cluster technology to realize the rational allocation and utilization of computing resources, thereby reducing the computing resource cost. The PaaS platform built by WEB technology can greatly simplify the process of machine learning development and reduce labor costs. Learning from the architecture of Heroku and Google App Engine, the platform designs the PaaS platform for machine learning and optimizes and improves it based on the machine learning development process to provide a highly scalable, highly available, cost-effective and flexible Machine Learning PaaS .The PaaS platform has become the most comprehensive, most scalable, most powerful, low-cost PaaS platform in machine learning.

**Key Words：**Machine learning; PaaS; model training; cloud computing; Docker; dataset; GPU computing

**目 录**

[摘要 II](#_Toc484718687)

[Abstract III](#_Toc484718688)

[1 绪论 1](#_Toc484718689)

[1.1 算法性能初探 1](#_Toc484718690)

[1.2 本文算法应用实例 4](#_Toc484718691)

[1.3 本文结构 5](#_Toc484718692)

[2 聚类算法简介 7](#_Toc484718693)

[2.1 主流聚类算法 7](#_Toc484718694)

[2.2 基于密度的聚类算法 8](#_Toc484718695)

[2.3 Skinny-dip算法 9](#_Toc484718696)

[3 优化算法原理及分析 11](#_Toc484718697)

[3.1 小波变换 11](#_Toc484718698)

[3.2 算法原理 17](#_Toc484718699)

[3.2.1 量化数据空间 17](#_Toc484718700)

[3.2.2 多维小波变换 18](#_Toc484718701)

[3.2.3 阈值滤波 19](#_Toc484718702)

[3.2.4 进行聚类 21](#_Toc484718703)

[3.2.5 标注初始数据集 22](#_Toc484718704)

[3.3 本章小结 23](#_Toc484718705)

[4 实验结果分析 24](#_Toc484718706)

[4.1 人工数据集实验 24](#_Toc484718707)

[4.1.1 Skinny-dip数据集 24](#_Toc484718708)

[4.1.2 小波聚类数据集 26](#_Toc484718709)

[4.2 自然数据集实验 28](#_Toc484718710)

[4.3 复杂度分析 28](#_Toc484718711)

[5 总结与展望 31](#_Toc484718712)

[5.1 本文总结 31](#_Toc484718713)

[5.2 未来的改进与应用 31](#_Toc484718714)

[致 谢 33](#_Toc484718715)

[参考文献 34](#_Toc484718716)

# 1 绪论

近年来，随着机器学习中神经网络算法的深入发展以及计算机计算能力的提升，机器学习被更广泛地应用于人工智能各个领域，形成了模式识别、数据挖掘、统计学、计算机视觉、语音识别、自然语言处理等交叉学科，已经成为近些年来人工智能领域最受关注和应用最广泛的技术之一。越来越多的高校实验室，企业研究院等科研机构从事机器学习研究，但是建立机器学习平台是一件复杂的事情，需要高性能GPU、CPU、SSD、主机，还要维护一个容易出问题的机房。除此之外，配置相关软件环境，获取大容量的数据集都需要花费大量的精力。因此，如何简化机器学习研究和开发流程，减少运维工作，使研究者专注于算法研究，将成为机器学习开发过程中面临的一个重要问题。

1.1 发展现状

随着存储和服务器技术的快速发展和互联网行业的繁荣，计算资源成本越来越低，计算资源算力也越来越强大。这一趋势使得云计算得到广泛应用，所谓云计算，是指计算资源（CPU,GPU,存储）作为可以出租的通用工具按需提供给用户租用。过去的几年里，云计算的出现对整个信息技术行业产生了巨大的影响，谷歌、亚马逊、微软、阿里巴巴等大公司努力提供更强大、更可靠、更低成本的云计算平台，试图重塑他们的商业模式。相比于传统模式，云计算拥有诸多的优势：

没有前期投资， 运营成本低，可拓展性高，业务风险和维护成本低。

正是由于云计算的巨大优势，它被广泛应用于各个行业，绝大多数中小型公司都选择大型云服务提供商的服务器资源，而不是自建服务器机房，由此催生了各种类型的云计算服务商。

云计算服务通常意义上分为三类：基础设施即服务（IaaS，infrastructure as a service），平台即服务（PaaS,platform as a service），软件即服务（SaaS,software as a service）。如下图图1.1，表示了三层之间的关系。

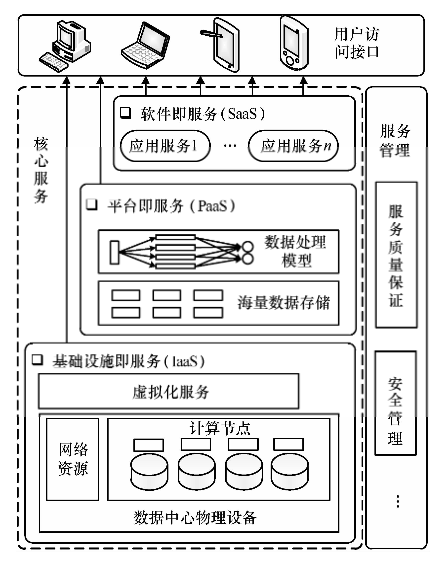


图1.1云计算服务层次架构图

1.2 PaaS平台现状

随着虚拟化技术的发展和云计算技术的日益成熟，越来越多的企业专注于为用户提供PaaS服务。一些 PaaS 供应商使用容器来减少为每个应用程序创建一个新的 VM 的开销, 从而降低了运行 PaaS 应用程序的成本, 同时在流程、网络和文件系统级别保持隔离，保证了可移植性和安全性。 PaaS提供了广泛的应用程序级详细服务, 并在云基础结构之上提供一个执行和开发环境。允许用户获得云服务, 而不需要购买和管理底层基础设施的成本和复杂性。平台允许开发者在云端实现或上传他们的应用程序, 这些应用程序就像普通的 Web 应用程序一样可以访问, 并且在使用量增长时自动升级。这使得开发者专注于生产开发而不是网络管理、存储和计算，这样可以极大地节约用户时间和精力，并为他们提供可靠且实用的服务。

促进规模经济的关键因素之一是多租赁。一般而言, 服务供应商会为每名客户管理一个专用的应用程式实例(即单一租约)。我们将多租赁定义为一种建筑风格, 使服务提供者能够在共享硬件和软件基础设施之外, 同时通过一个单一的应用程序实例，来为不同租户(即客户组织和公司)的最终用户提供服务。PaaS 平台能够使软件开发者在不需要处理支撑的中间件和基础架构的复杂性的情况下构建和提供 SaaS 应用程序。PaaS服务商可以大致分为以下三类：

1. 模仿和匹配流行企业应用服务器和中间件平台API的 PaaS 平台。例如, Windows Azure；Net 框架和 IIS 网络服务器, Red Hat OpenShift；基于 JBoss 平台, 在 WebLogic 服务器上运行的 Oracle Cloud和使用 VMware 和 Spring 技术的Cloud Foundry。

2. 聚焦 PaaS 平台, 旨在优化支持特定类型的云应用程序。这些平台通常部署其他的中间件和存储设施,它们的规模更大。Google App Engine，Heroku和 GigaSpaces 的 XAP 弹性应用平台属于这个类别。

3. 元数据驱动的 PaaS 平台。与聚焦 PaaS 平台类似, 这些平台的设计考虑到了 SaaS 应用程序。此外, 由元数据驱动的 PaaS 平台引入了更高级别的组合和配置接口, 使典型的中间件级别的抽象化。然而, 这限制了可以实现的应用程序的复杂性。例如 Force.com, WOLF以及 TCS InstantApps。

1.3 机器学习面临的问题

相比于高度抽象的更底层的IaaS，PaaS专注于垂直领域，更趋向定制化。例如Cloud Foundry专注于Web应用开发和数据库，Google App Engine专注于典型Web应用程序部署。XAP专注于于高性能和低延迟事务处理以及实时分析用例。而机器学习近些年才被广泛应用，因此还未出现一家成熟的PaaS服务商为机器学习研究者提供应用级服务。从而导致机器学习研究者只能使用IaaS服务商提供的云计算资源，一步步搭建定制化的机器学习环境。这一过程中，开发者不得不投入时间和精力来管理计算资源、搭建相应软件环境、部署应用程序，而这些工作本应该是相应的PaaS服务商来完成，开发者只需要专注于机器学习算法开发。

从2017年起，部分团队开始尝试搭建机器学习领域的PaaS平台，为机器学习研究者提供更精细化的服务。目前出现的机器学习PaaS平台主要有以下几类：

1. 提供训练环境以及周边功能，能够将机器学习模型快速转化为应用，如FloydHub，提供集成化的机器学习云计算服务，支持多种深度学习框架，包括Caffe、PyTorch、Chainer、MxNet、TensorFlow、Keras，专注于深度学习的PaaS平台，支持快速部署和训练深度学习模型，能够在一分钟内快速开始训练深度学习模型，并且支持训练模型的部署，转化为HTTP服务，同时引入公共项目和数据集来促进开源协作。

2. 仅提供机器学习云计算服务，Neptune.ml，和FloydHub类似，作为机器学习PaaS平台，提供了基本的快速部署和训练深度学习模型功能，同时提供Notenbook模式，方便调试和在线管理和监控。该平台提供的功能有限，拓展性也比较差。

3. 提供定制化的机器学习环境，提供某种特定深度学习框架的软件环境，实现快速训练和部署模型，有些甚至提供了可选择的数据集。典型应用有Google Cloud Machine Learning Engine，能让用户轻松构建可基于任意规模的任何数据类型工作的机器学习模型，可以利用强大的 TensorFlow 框架创建模型。该平台仅支持TensorFlow，适用范围狭窄。

以上现有的平台并不能解决目前机器学习研发所面临的问题，FlyodHub是其中最优秀的PaaS平台，但是由于其发布仅四个月，产品的可用性依然不够，在拓展性和性价比仍然有很大的优化空间。而Cloud Machine Learning Engine，Azure和Bluemix提供的深度学习平台的关键弱点是可扩展性，内部集成和与开源深度学习框架的互操作性差。Google Cloud ML目前无法执行在Caffe，Theano或Torch等流行框架下编写的模型。这个限制是一个致命性的缺点，TensorFlow只是机器学习中的一个比较流行的框架，并不能满足用户的的开发需求。此外，除了Microsoft在Azure堆栈上支持Azure ML的一些创造性工作外，大型云深度学习平台目前无法在内部部署环境中执行。而且从机器学习模型到可调用的工程应用中间还有许多事情要做，如何让算法开发者专注于算法开发，省去复杂的运维和软件开发工作，这是目前亟待解决的问题。

1.4 本文结构

在本文接下来的部分，我们将以如下顺序对我们所研究的问题以及平台的设计和实际使用和测试结果进行讲解。

首先我们将阐述与本文内容相关的工作，包括云计算的各类形式和层次、目前所面临的问题、市场上的解决方案。然后，我们将主要分析如何进一步解决问题，如何设计和开发这个PaaS平台，解决工作流问题。主要包括平台的功能设计、平台的技术实现方案。最后，我们会对全文进行总结，并给出现有阶段下本平台的不足和继续发展的方向，对未来的进一步优化进行展望。

本文结构如下图1-5所示。

图1-5 本文结构示意图

# 2 平台设计方案

在研究了目前机器学习的市场以及现有的云服务平台以后，本文设计并开发了用于机器学习模型训练的PaaS平台，为机器学习开发者提供低廉、快速、易用的机器学习云服务，使得开发者能够专注于算法开发而忽略环境搭建、运维等工作，利用Web技术和Docker技术实现自动化环境配置、任务训练、日志监控、自动化运维等流程，极大地提高机器学习开发效率，降低开发成本。

本章将按照一般机器学习开发所包含的三个主要过程来介绍该平台的设计和实现，主要包含：自动化环境配置、任务模型训练、自动化运维；最后介绍本平台为提升开发效率所做的一些核心功能的设计：可快速引用的分布式云端数据、便于用户操作的客户端。

2.1 自动化环境配置

搭建机器学习模型训练环境是一个复杂而又耗时的事情，本文设计的平台将这一过程做了最大限度的优化，使其完全自动化，同时尽可能为用户提供更多选择，使用户可以快速定制训练环境、获取计算资源，这样就保证了这一过程的高效性和灵活性。

2.1.1 环境配置流程

在一个典型的机器学习环境搭建过程中，用户一般需要完成以下三个环节，才能搭建一个可用的机器学习训练平台：

1. 购买相应的硬件设施，包括服务器、数据存储器、GPU等设备。部分用户会选择购买云计算服务商的云服务器等设备，也有部分用户会直接购买上述设备，后者还需要维护硬件的正常运行。
2. 配置、组装上述设备，使其构成完整的计算体系，包括数据盘的挂载、网络管理、安装操作系统等。
3. 搭建相应的机器学习软件环境，安装相应的软件包、机器学习库、必要的开发工具等。

本文设计的平台致力于将上述复杂过程自动化，并且非常灵活，可定制化程度高。平台为用户提供了初始化环境的界面，在这个界面，用户可以根据自己的需要选择相应的计算资源配置，一键生成所需要的计算环境。在该平台中， 过去需要复杂的搭建流程被简化为一步，并且可以在一分钟内完成。

2.2 完善的任务模型训练流程控制

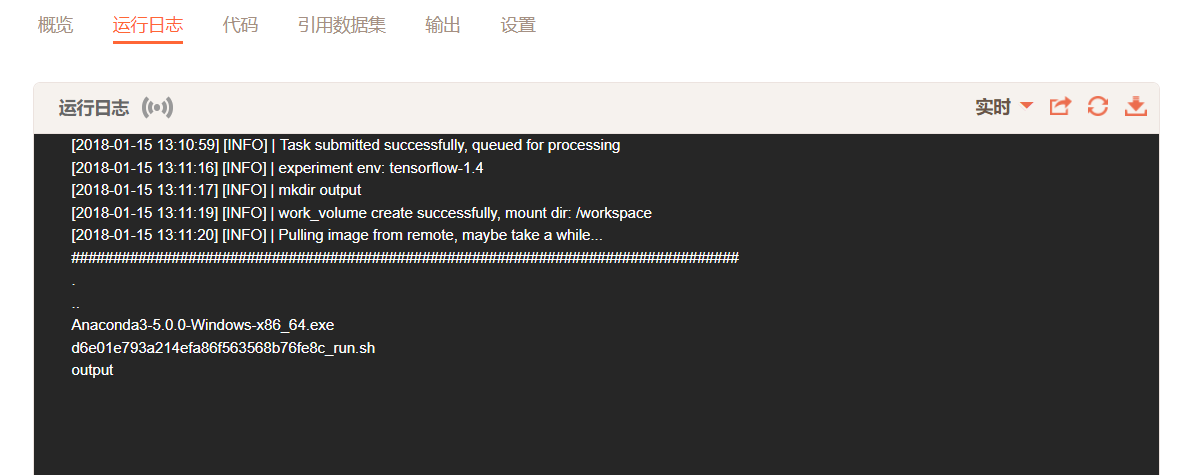
任务模型是为方便描述和开发自定义的概念，在该系统中，是指一次训练任务及其所包含的信息，包括相关程序文件及存储路径、训练任务所需软件环境及配置、训练任务的描述信息（用于区分各个模型，方便辨识，方便多人协作）、输出结果、日志等信息。一个任务模型包含了该训练任务的整个生命周期以及相关信息和文件。这是整个PaaS平台的核心工作流程之一，即帮助机器学习开发者训练机器学习模型。

该平台将一次机器学习模型训练任务抽象为一个任务模型，当用户提交一个训练模型的任务以后，系统将该任务放入调度队列，并初始化相应的软件环境开始训练模型。该任务模型定义有完整的生命周期，包括等待、进行中、失败、超时、结束。用户可通过监控和日志查看任务模型的状态，并且通过命令控制任务模型的状态。

为了方便调试和监控训练过程，该平台提供了实时日志和状态提醒功能，还提供了notebook调试模式。

1. 日志和状态监控，为了方便用户获知训练任务的状态信息，该平台利用kafka和pilot实现了日志模块，实时获取任务训练的进程和其他信息。同时，监控系统检测到任务模型的生命周期改变时，将会通过邮件、站内信等方式提醒用户。

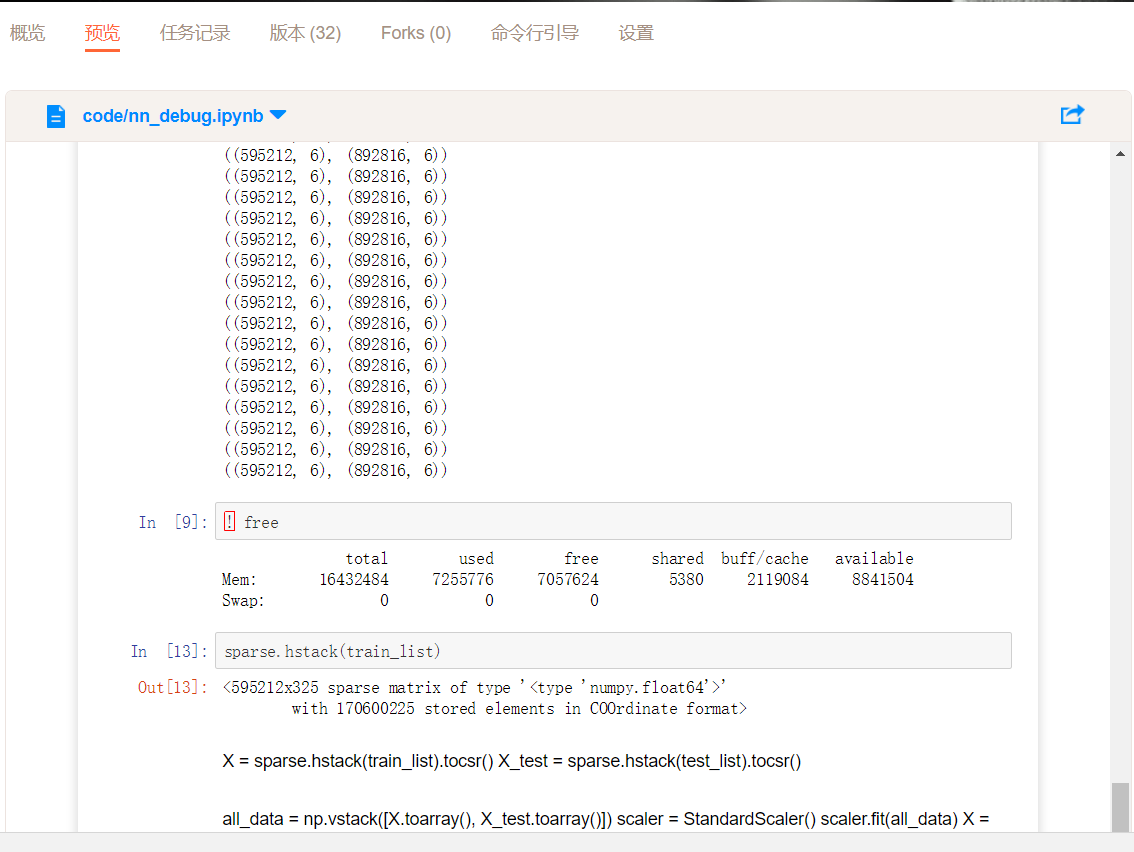
如图1所示，为用户提供实时日志流信息：



如图2所示，状态监控信息



1. Notebook调试模式，为用户提供交互式笔记本，方便用户调试和更改程序，查看和记录每一步的输出结果。如图3所示：



2.3 自动化部署和运维

在模型训练调试、测试完成以后，将模型转化为应用是一个繁琐的过程。在一个典型的转化过程中，部署WEB应用需要购买相应的WEB服务器，搭建Web服务环境，配置DNS解析和域名解析，设置IP，同时还需要做相应的运维工作，以保证服务的稳定性。该平台的设计讲这些复杂而又耗时耗力的工作自动化，平台为用户提供了一键部署功能，将机器学习模型转化成Web应用，以REST API接口对外提供服务。

2.3 分布式云端数据集

数据集是机器学习训练中最重要的资源之一，尤其是对于数据挖掘、NLP、机器翻译、图像识别等领域研究人员。由于机器学习所使用的数据集往往都比较大，少则几个G，多则几十个G，以保证模型得训练效果，这就使得数据集的传输和存储成为一个严重的问题。

目前市场上出现了一些专门用于管理和提供数据集的云服务提供商，但是他们只专注于提供和管理数据集，并不提供机器学习计算资源，这就使得他们的服务使用场景有限。比如受到网络的影响，传输和下载速度慢。

本文设计的平台将数据集模块独立出来，用户可以上传自有数据集用于训练，也可以引用公开数据集。平台收集了大量公开数据集，为用户提供便利。这种模式也将促进数据的开源，方便机器学习开发者的研究工作。

平台选用阿里云NAS专用数据存储服务器存储数据，并且采用了分布式存储架构，减少了数据集的复制和移动，显著减少等待时间，改善网络性能。用户仅使用一条命令就可以快速引用存储在云端的数据集，这大大减少了用户移动复制上传数据集所耗费的时间。

2.3 多用户界面

该PaaS平台聚焦于减少开发者的工作，提升工作效率，以灵活、简单、高效为核心理念，因此为用户提供了非常友好的用户界面。包括基于Python的CLI客户端，基于Web的网页端。

CLI客户端和Web网页端提供了非常友好而强大的功能。客户端兼容了目前主流的PC操作系统，包括Windows、MacOS、Linux，能够满足用户的觉得大多数需求。

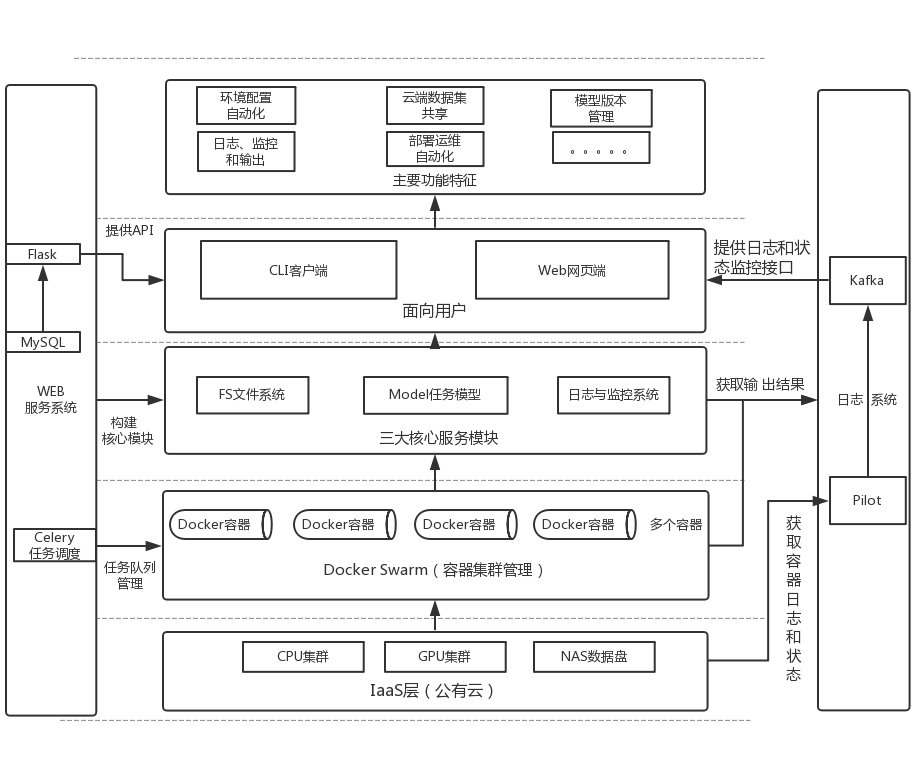
2.4 本章小结

本章的前三节按照模型训练所经历的三个核心环节介绍了本平台的核心功能，然后介绍了该平台十分重要的附加功能：分布式数据集管理，最后介绍了该平台的用户端的设计。由此可以看到，该平台致力于解决机器学习整个开发流程中的所面临的主要问题，为机器学习开发者提供一站式云计算服务，使开发者专注于核心的算法开发而忽略其他工作，从而提升开发效率，降低开发的人力成本和资金成本。

# 3 平台的技术实现及原理

本章将根据第二章所介绍的三个核心模块详细说明其实现原理和技术架构，主要包括：如何使用Docker技术实现环境配置的自动化，任务模型的生命周期的控制以及监控日志系统的实现，部署和运维自动化的实现，客户端的实现。由于这些功能模块的实现都比较复杂，所涉及的开源工具和库也很多，因此本章将略写部分非核心模块的实现。

该平台参考了Heroku和Google App Engine的架构，依据机器学习开发流程做相应的优化和改进，以实现一个高拓展性，高可用性，高性价比的灵活的机器学习PaaS平台。图4是该平台的系统架构图。IaaS层基于阿里云的CPU集群和GPU集群构建，数据存储选用阿里云NAS服务，用分布式存储系统来分发数据集和大文件。然后在IaaS层的基础上构建容器集群服务和Web服务：



3.1 环境配置和Docker容器技术

容器是在 Linux 环境中提供隔离和资源管理的一种手段。 这个术语来源于船运集装箱, 是储存和运输任何货物的标准方法。 操作系统容器提供了一种通用的方法, 可以将进程与系统的其他部分隔离开来。

Docker 是 PaaS 提供商 dotCloud 开源的一个基于 LXC 的高级容器引擎，可以被视为一个操作系统级虚拟机, 它在用户空间中作为一个独立的过程运行, 并与其他容器共享相同的内核。它对基础设施服务进行虚拟化和隔离，把应用需要的运行环境、缓存环境、数据库环境等封装起来，形成一个独立的开发环境。Docker 还有一个额外的镜像管理和一个分层的文件系统来优化容器的性能和减少磁盘空间。Docker使用 Docker daemon 管理单个节点中的多个容器。 通过命令行接口(CLI) , Docker 客户端可以管理包含操作的容器, 例如从注册表中抽取存储库、运行容器、将容器提交到新的镜像、按照配置文件初始化容器以及终止运行的容器。

3.1.1 Docker容器技术的应用

本文设计的平台选用Docker容器技术封装用户所需要的训练环境和服务，可以很方便地隔离出一个独立的用户空间，用于任务模型的训练和管理。选择Docker技术，为平台提供了以下优势和功能：

1. 独立、安全、易于管理的用户独立空间。Docker本身是一个操作系统上的容器。当用户提交一个训练模型的任务以后，本文的平台根据用户需求生成相应的配置文件，然后根据配置调用生成Docker容器的服务接口实例化一个容器，当用户的任务模型训练完成以后，该容器会被自动回收。
2. 节约资源，降低成本。在一般情况下，机器学习开发者往往会购买一个云端服务器，那么这个服务器就只属于个人，那必然会有闲置和性能过程的情况，这是资源浪费的一种情况。在该平台中，当用户提交的任务比较多时，一台服务器上将运行多个Docker容器实例，当该服务器资源被占满（CPU，存储器等），就会对当前节点上的服务器进行扩容，从而实现合理的资源调度和利用，这样就避免了计算资源闲置和性能过剩的情况。
3. 跨平台，可移植性好。Docker本身支持跨平台，可以运行在目前的主流操作系统上，因此这里构建的环境也支持跨平台。

3.1.2容器集群

本文的平台需要在一个主机上运行多个容器实例，以合理分配和共享资源。多个容器的管理和调度是一个非常复杂的难题。平台选用Docker Swarm构建容器和管理集群，通过把多个Docker Engine聚集在一起，形成一个大的Docker Engine，对外提供容器的集群服务。同时这个集群对外提供Swarm API，使得用户可以像使用Docker Engine一样使用Docker集群。Docker Client发送请求给Swarm，Swarm处理请求并发送至相应的Docker Node，Docker Node执行相应的操作并返回响应。当用户初始化并开始一个训练任务时，平台将任务放入调度队列，当队列任务被执行时，平台通过Swarm实例化一个Docker容器，并根据任务所需配置初始化容器环境，并将这个容器分配给该用户。当任务结束时，该容器会被自动回收，输出结果和相关文件存储到任务模型中，方便用户查看和复现。

3.2任务模型和监控

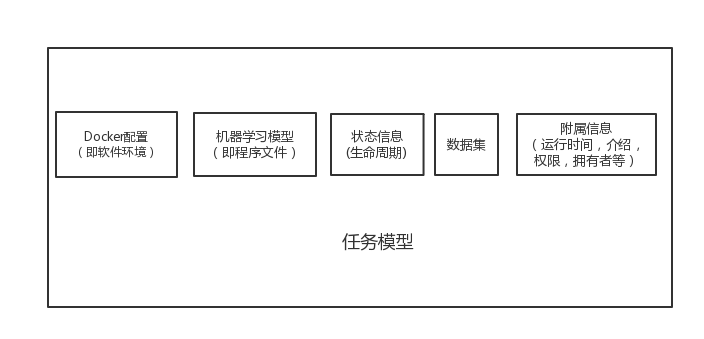
本文中的算法主要由以下五个步骤组成：

3.2.1 任务模型

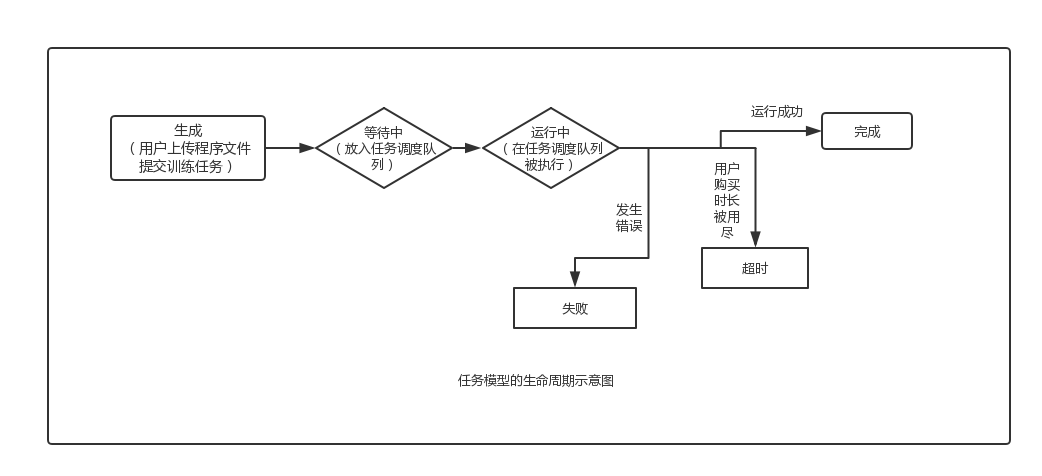
任务模型是为方便描述和开发自定义的概念，在该系统中，是指一次训练任务及其所包含的信息。一次训练任务以及其附带的文件、配置等信息抽象为一个任务模型进行管理，就可以实现Git一样的版本管理和控制，从而能实现多人协作以及版本回退，这样复现实验、追踪错误、查看历史训练结果等等功能都可以实现。

用户新建一个项目，上传相关机器学习模型文件，并提交了训练任务之后，该任务模型自动生成，并进入等待状态。这个过程中用户的操作是基于Falsk的WEB服务的，Flask将任务模型的相关信息记录进MySQL数据库中，用户提交任务时，Flask把该任务提交到Celery异步任务队列中，Celery按照优先级处理所有任务。当Celery任务队列里的该训练任务被执行时，Flask调用Swarm相关接口初始化一个Docker容器，该任务被放入这个Docker中运行，这时候开始日志系统开始读取并输出日志，记录相关信息。训练完成后，又记录其训练输出结果及训练时间，Flask将这些相关信息添加更新到数据库中，并调用Swarm接口将该任务模型所使用的容器被销毁。这个任务模型的生命周期就结束了，任务模型被永久存储在数据库中。

当用户需要复现实验、追踪错误的时候，再次运行该任务，已被记录的任务模型将按照存储的配置和信息实例化容器，构建和之前完全一样的环境，能极大地提高调试效率和快速追踪错误、异常。下图是一个任务模型的基本概念图：



下图是任务模型的生命周期示意图：



3.2.3 日志和异常监控

日志和监控系统主要包含三层服务，是基于开源框架Kafka和Pilot搭建的。Kafka是一种高吞吐量的分布式发布订阅消息系统，它的主要目的是建立一个数据处理框架, 以收集和传播大量信息, 包括用户行为数据，网站操作统计等等。 它最初由 LinkedIn 开发, 作为活动流和操作数据处理管道的基础。之后它成为 Apache 项目的一部分。 现在, 它被不同的公司用作信息系统和数据管道。 由于拥有一些独特而实用的设计，Kafka在信息传播方面有优势，这使得它与其他的信息系统不同。考虑到系统的可拓展性和未来的发展发展方向，该平台采用Kafka搭建日志系统，以应对未来可能出现的大量用户以及更复杂的海量消息推送，平台的日志和监控主要包含三层日志：

第一层：环境配置阶段。这一过程主要包含用户在平台上的一些操作、与用户相关的项目和数据集的动态、用户账户变动的信息。用户上传模型的程序文件以后，Flask根据用户操作记录任务模型状态，反馈给用户。

第二层：任务训练阶段。当任务模型在任务队列中被执行时，系统会为任务模型实例化一个Docker容器，此时，该模型就运行在Docker容器中。Docker的日志可以它分成两类，一类是stdout标准输出，另外一类是文件日志。stdout是写在标准输出里面的日志，另外的就是文件日志，存储在磁盘上。Docker允许将Docker日志路由到指定的第三方日志转发层，可将日志转发到Fluentd，syslog，GELF 或 NAT 服务器。本平台利用fluentd-pilot读取Docker容器的日志，并传递给Kafka，由Flask从Kafka中获取相关信息反馈给用户。

第三层：任务结束阶段（也可能失败）。Flask获取任务训练结束以后状态信息和错误信息，并通过邮件和客户端反馈给用户。

在任务训练阶段，可能会产生大量的标准输出日志，用户需要实时了解模型的训练状态和输出。该平台利用WebSocket简历实时数据通道，可以将训练状态和日志实时传递给用户，方便调试和开发。

模型训练过程中可能会出现异常，平台为用户提供了邮件和站内信提醒功能，用户可根据需要定制报警信息，这样可以保证整个过程更加可控，以避免时间浪费。

3.3 部署和运维自动化

从机器学习模型到应用级的服务是一个比较复杂且繁琐的转化过程，涉及Web开发、路由配置、IP设置、安全防护、负载均衡等内容，为了减少开发者工作量，平台提供一键模型发布功能，将上述过程自动化，转移为平台的工作，这样可以极大提高开发生产效率。

3.2.1 模型转化为应用

当模型训练完成以后，达到开发者的期望，开发者可以在平台上发布这个模型，转化为REST API风格的Web服务。平台将为这个模型实例化一个Docker容器，启动一个Web服务器，将外部端口进行映射，同时分配一个唯一的路由地址，即网站服务站点地址，同时配置异常报警程序、自动守护进程，这样就可以实现自动化运维和Web级服务。

3.4 用户端界面

为了方便用户操作，平台提供了跨平台的客户端和网页端用户界面，网页端和客户端都包含了所有的用户操作，用户可根据自己需求选择。

3.4.1 CLI客户端

CLI交互式命令行客户端基于Python开发，兼容Python2和Pyhon3，使用命令行进行各类操作，可与程序开发无缝结合。该命令行工具发布在Python包管理平台上，利用Python自带工具Pip安装、管理。由于Python本身支持跨平台，可以运行在主流操作系统上，因此该工具也具有跨平台的特性。

该工具被启动时，从命令行终端中获取用户输入，根据输入命令调用Web服务接口执行操作，当需要输出实时日志或者运行任务时，该工具会在后台运行一个进程，从而使用户可以进行其他操作。

3.4.2 网页端

网页端目前只适配了PC端，利用目前主流的前端框架Vue构建的前后端分离的系统，页面UI组建使用Element UI，采用了统一的设计风格，保证前端界面的美观。由于平台提供了公共项目和数据集以促进技术和资源开源，因此前端采用前后端同构技术，利用NodeJS在服务端渲染页面，从而保证页面的高性能，同时实现了搜索引擎优化，方便主流搜索引擎的搜索和收录。

3.3 本章小结

本章逐步分析了该平台的三个核心模块的实现原理以及所用到的开源框架和工具，利用本章所描述的技术方案，实现了一个高可用性的平台。可以看到，平台的核心目标是帮助机器学习开发者专注于算法开发，降低开发成本、资金成本，从而提升效率。因此平台的功能设计都是围绕这个目标完成，根据机器学习的整个流程做最大幅度的优化，同时考虑到用户多样化的需求，利用技术上的设计来实现平台的弹性和灵活性。

# 4 实践和测试结果分析

本文主要进行了三组实验：在Skinny-dip数据集的实验对比；在小波聚类数据集的实验对比；在自然数据集的实验对比。

这三组实验分别对应了对Skinny-dip算法有优势的数据集；普遍的人工数据集以及现实应用中的数据集。下文就将按照上述顺序以此解释这三组实验的原理并分析实验结果。最后，本章还将整体分析本文算法的运算复杂度并于其他算法的复杂度进行对比。

4.1 节约资源，降低成本

IaaS服务商直接为用户提供基础设施服务，即为用户提供不同配置的服务器、存储器等硬件服务，而PaaS服务商从底层搭建一个十分庞大的硬件设施服务，在这个基础上搭建软件环境，再将这些计算资源按照用户需求分配给各个用户，资源的统筹分配和协调由PaaS服务商来完成，从而大大提高资源利用率。

下面将用一个典型示例来比较这两种服务的差异，神经网络是机器学习中最常见的研究领域之一。循环神经网络((Recurrent Neural Networks，RNNs))是一类人工神经网络, 其中单元之间的连接形成一个有向的循环。 这使得它能够表现出动态的时间行为。 与前馈神经网络不同, RNNs 可以使用内存处理任意的输入序列。这使得它们适用于诸如未分割的、连接的手写识别或语音识别等任务。RNNs已经在众多自然语言处理(Natural Language Processing, NLP)中取得了巨大成功以及广泛应用，在词向量表达、语句合法性检查、词性标注等方面的应用都非常成功。本章的示例将用RNNs生成唐诗。

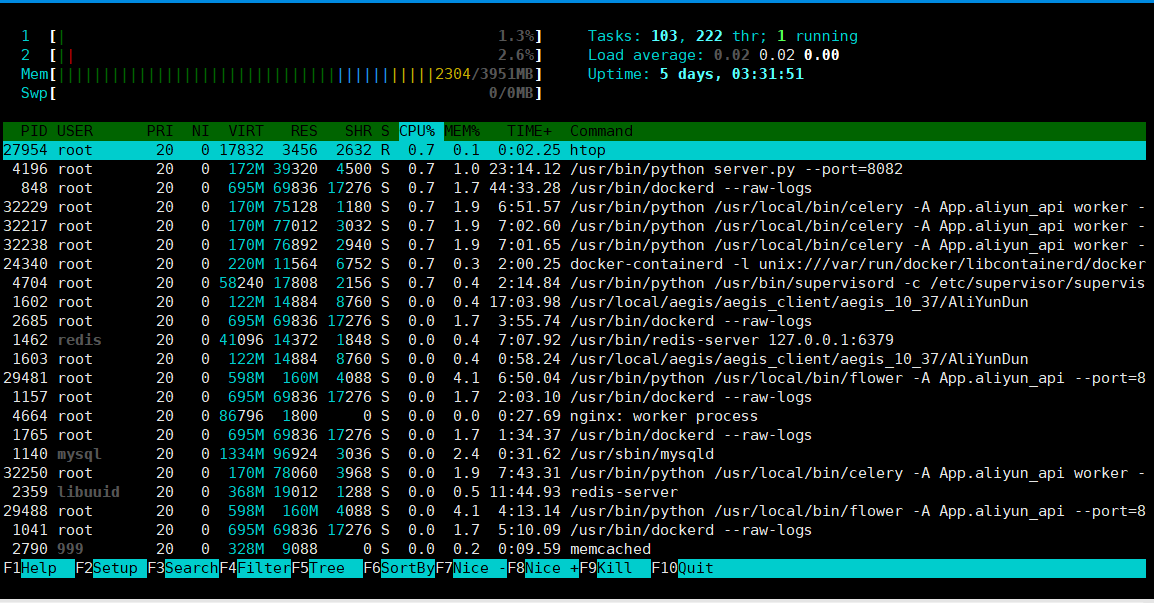
该模型引入了谷歌开源机器学习框架TensorFlow、Python开源数值计算库、Python内建集合模块collections，下图是主要程序的一部分：



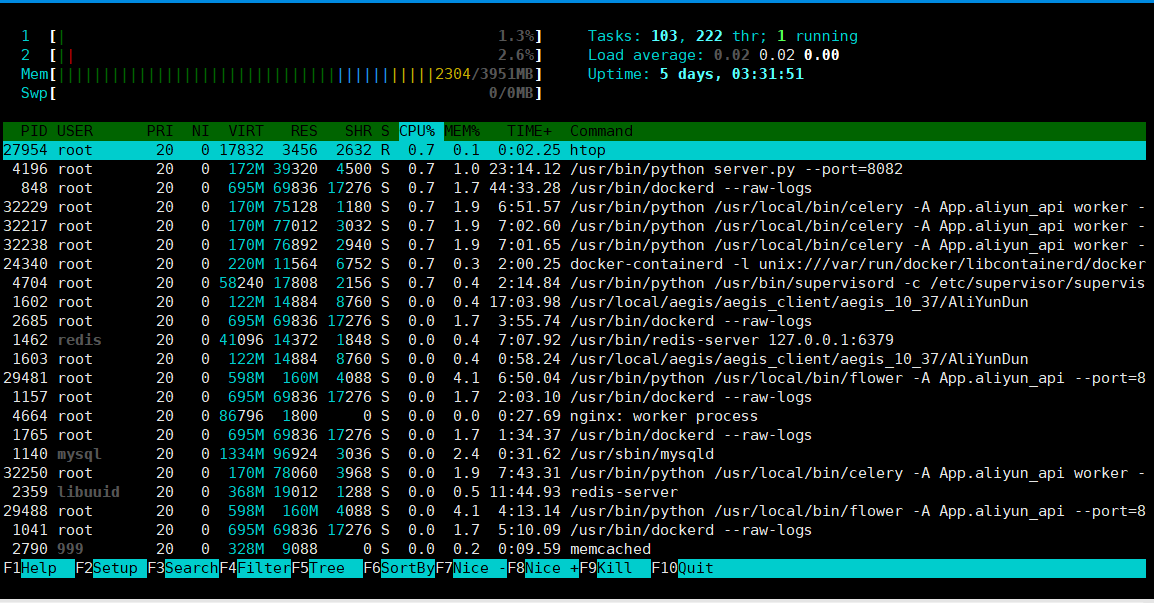
该平台提供的计算环境是:

* Ubuntu 14.04
* Intel(R) Xeon(R) CPU E5-2682 v4 @ 2.50GHz 双核
* 4 G RAM

下图是系统没有运行任何任务时候的资源占用情况



下图是进行该模型训练时的资源占用情况



在该计算环境中，利用htop工具获取系统资源占用信息以及主要进程，根据两张截图，在训练该模型的过程中，CPU两个核心占用率分别由1.3%、2.6%上升到2.6%/3.3%，Mem占用由2304M涨到了2309M，空闲了大量计算资源。

由于本平台目前是在阿里云服务器的基础上搭建的，而阿里云又是国内最大的云计算厂商。因此下面的服务器对比可以代表市场上的最普遍情况。该平台在运行一个月以后给出了定价方案，注册用户可以选普通用户和VIP用户（按月购买）两种套餐，也可以按训练时间购买，价格如下图所示。

而普通开发者目前只能选择购买云服务器或者实体服务器进行机器学习开发，以阿里云为例，购买最低配置的服务器需要45.00元每月，购买与该平台同等配置的服务器需要295.40元每月。

下表将对比亚马逊AWS，阿里云，腾讯云的最低支出：

|  |  |  |  |
| --- | --- | --- | --- |
| 服务商 | 本文的平台 | 阿里云 | 腾讯云 |
| 价格/元 |  | 295.00（每月） | 642.00（半年促销） |

由于本节的示例仅在CPU上运行，如果需要使用GPU，那么成本差距将进一步扩大。

下图对比GPU云计算资源价格（双核4G），腾讯云暂无对应配置机器：

|  |  |  |
| --- | --- | --- |
| 服务商 | 本文的平台 | 阿里云最低配置 |
| 价格/元 | 8.00（每小时） | 2441.45（每月） |

云计算服务厂商的发展已进入较稳定阶段，计算资源价格也趋于稳定，因此服务器的价格反应了最低的硬件成本和运维成本。从上述价格对比可以看出，由于使用了Docker技术将单个服务器进行分割，并且利用高效的算法对用户的计算任务进行合理分配和调度，从而极大提高了资源利用率，这样才降低了计算资源的成本，为用户节约了支出。

4.2 开发流程自动化，提高效率

平台的核心功能是帮助机器学习开发者提高开发效率，省掉与算法开发无关的工作。因此将开发流程自动化是本平台的核心优势，这一节将用一个示例来对比现有的机器学习开发流程和平台上的开发流程，展示开发效率和用户体验上的差别。以一个真实的比赛为案例京东JData算法大赛-高潜用户购买意向预测入门程序(starter code)，复现该算法模型训练过程。

4.2.1流程对比

在目前的开发过程中，要复现上述算法模型，需要完成以下步骤（由于实际情况各有不同，这里讲述最常见的开发流程）：

1. 购买服务器，对于高校实验室或者企业开发者，他们一般会选择购买实体服务器，维护一个用于开发的服务器机房，这里所包含的工作量非常大、成本也非常高，搭建完成以后相当于拥有了云服务器（他们本身是服务器的拥有者）。对于普通开发者，他们大部分会选择购买IaaS服务商的服务器，比如阿里云，亚马逊等等。由于云服务器厂商提供的服务器比较稳定，且配置简单，这里购买了阿里云的ECS服务器，下面将基于阿里云服务器上搭建软件环境。
2. 安装操作系统以及软件环境，在阿里云上可以直接选用官方镜像源装载操作系统，为方便操作，选用了Ubuntu。安装完成之后，利用Python安装包安装pandas,numpy,matplotlib,sklearn,xgboost等常用库，之后将代码和数据集利用Ftp或者Github上传到服务器，然后执行该训练任务。
3. Notebook调试模式，为了方便调试，需要安装jupyter，开启notebook调试模式。利用pip安装该工具，选定路径开启notebook。由于操作系统本身安全设置，这里需要配置防火墙、开启对应端口，之后才可以使用基于Web的服务。

如果基于该平台来实现算法调试和训练，只需要以下操作：

1. 注册账号，购买合适的用量包，安装CLI客户端。
2. 在客户端输入以下命令：

git clone https://github.com/RussellCloud/JData.git

cd JData

russell login

russell init --name JData

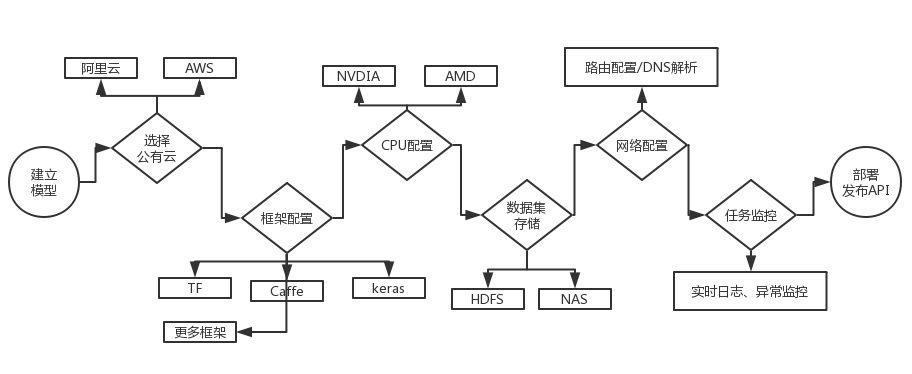
russell upload

russell run --mode jupyter –data id:data

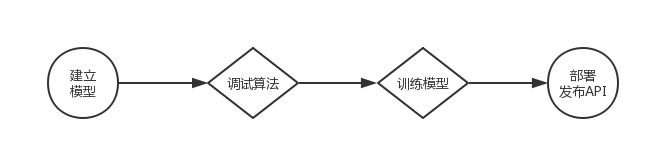
1. 在浏览器notebook模式下调试程序

4.2.2 结果分析

一般而言，开发一个基于深度学习的应用开发者首先需要去购买公有云服务，比如阿里云、AWS等，然后在云主机上安装相应的机器学习框架，常见的框架有Caffe、PyTorch、Chainer、MxNet、TensorFlow、Keras，接着，开发者需要配置相应的GPU，比如NVDIA、AMD。基本完成深度学习环境搭建之后，开发者开始训练机器学习模型，首先需要选择数据集，在训练数据集较大的情况下，需要使用云端数据盘，比如HDFS、NAS，达到理想的效果以后，再开始部署该机器学习应用，使得该模型能够对外提供服务。以提供HTTP的API服务为例，开发者除了完成上述流程以外，同时需要配置相关网络设置，如服务监控、域名解析，如果是大规模应用还需要考虑负载均衡。在要提供稳定的服务之外，开发者还需要做相应的运维工作基本流程如图3所示。通过以上流程才能完成一个基本的机器学习模型到应用的训练和部署流程。



本文所设计的PaaS平台将以上流程简化为两步，让用户在一分钟内完成模型环境搭建、训练和部署（模型训练时间与框架、模型复杂度、计算机性能有关，这里的时间不包括模型训练本身所用时间），基本流程如图5所示。该平台让用户专注于算法开发，在建立模型以后，将程序上传到该平台，选择公开数据集，一键开启任务训练模式，极速发布API。将服务器选择、环境搭建、数据集选择、应用部署、运维等工作转移到PaaS平台，由该平台来实现自动化配置。



**图5 简化后平台训练流程图**

4.3 功能完善

在机器学习学习的整个开流程中，平台为用户提供了尽可能多和使用的功能。借鉴了目前的一些机器学习云计算服务商的功能设计，例如UCI Machine Learning Repository的数据集功能，Github的版本控制及协作功能，Heroku等PaaS服务商的自动化部署功能。

目前市场上和本平台类似的产品还有Google Cloud Machine Learning Engine，TinyMind，Neptune Machine Learning Lab，下面将根据这几个平台目前实现的核心功能做详细的对比，来说明该平台的完善性。

TinyMind和Neptune Machine Learning Lab与本平台十分类似，在功能设计和系统架构方面都有共同之处，而Google Cloud Machine Learning Engine是高度定制化的，仅支持谷歌的机器学习框架，在应用范围上十分狭窄，功能也有限。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 功能 | 平台 | | | |
| 本平台 | TinyMind | Google | Neptune |
| 环境选择种类 | 11种框架，  2个版本Python | 4种框架  3个Python版本 | 仅支持一种框架 | 5种框架  1个Python版本 |
| Jupyter NoteBook | 支持 | 不支持 | 不支持 | 不支持 |
| 云端公开数据集 | 支持 | 不支持 | 仅支持特定数据集 | 不支持 |
| 并行任务调参 | 支持 | 不支持 | 不支持 | 不支持 |
| 版本管理  （兼容Git） | 支持 | 不支持 | 不支持 | 不支持 |
| Dashboard管理 | 提供网页界面管理项目、任务以及数据集 | 仅能管理数据集项目 | 包含简单操作 | 仅能管理数据集项目 |
| 日志 | 提供实时日志，日志下载 | 提供日志查看，但日志不全面 | 无 | 无 |
| 监控异常 | 提供站内信和邮件提醒 | 无 | 无 | 无 |
| 一键部署发布 | 支持并提供自动化运维 | 无 | 无 | 无 |

4.4 总结

根据上一节对比可以看到，该平台包括了其他平台的所有功能，并在他们的基础上做了改进和进一步的优化，可以为开发者提供一站式的服务，基本上解决了绝大部分开发者目前所面临的开发工作量大、附加工作多的问题。极大地降低了开发人力成本和资金成本。

# 5 总结与展望

5.1 本文总结

5.2 未来的改进与应用

# 致 谢

四年时光转瞬即逝，这样的感觉在我为文论写上最后一笔时不禁涌上心头。每到要告别的时候就想问问自己：这四年我有没有虚度，我有什么改变？

大学四年我后悔的事情不少。我后悔自己没有再努力一点，后悔自己没有再勇敢一点，后悔自己还有这么多没有尝试过的事情。但有一点我觉得自己很幸运，在大学我看到了许多我想成为的人。我想像蔡学长一样谈起自己的科研方向就能如数家珍，我想像田同学一样能工作在自己喜欢的岗位上，还有许许多多向他们一样的人在指引着我。

在华科我还遇到了最好的一群老师。在此我要特别感谢何琨老师，邓前松老师以及柯昌剑老师对于我研究工作的倾力指导与支持，还要感谢我的班主任付松年老师四年来对我学习生活上的许多帮助。

铁打的营盘流水的兵，在大学我们已经习惯了被称为学长，但进入了新的环境还要做回新人与学弟。这是又一次的未知，又一次的机遇。相信在华科四年中学到的知识能帮助我准备好下一次的征程。

# 参考文献

[1] Viola P, Wells W M. Alignment by maximization of mutual information[C]//Computer Vision, 1995. Fifth International Conference on. IEEE, 1995: 16-23.

[2] MacQueen J. Some methods for classification and analysis of multivariate observations[C]//Proceedings of the fifth Berkeley symposium on mathematical statistics and probability. 1967, 1(14): 281-297.

[3] Ester M, Kriegel H P, Sander J, et al. A density-based algorithm for discovering clusters in large spatial databases with noise[C]//Kdd. 1996, 96(34): 226-231.

[4] Maurus S, Plant C. Skinny-dip: Clustering in a Sea of Noise[C]//Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2016: 1055-1064.

[5] Kaufman L, Rousseeuw P. Clustering by means of medoids[M]. North-Holland, 1987.

[6] Dempster A P, Laird N M, Rubin D B. Maximum likelihood from incomplete data via the EM algorithm[J]. Journal of the royal statistical society. Series B (methodological), 1977: 1-38.

[7] Gower J C, Ross G J S. Minimum spanning trees and single linkage cluster analysis[J]. Applied statistics, 1969: 54-64.

[8] Zhang T, Ramakrishnan R, Livny M. BIRCH: an efficient data clustering method for very large databases[C]//ACM Sigmod Record. ACM, 1996, 25(2): 103-114.

[9] Wang W, Yang J, Muntz R. STING: A statistical information grid approach to spatial data mining[C]//VLDB. 1997, 97: 186-195.

[10] Hinneburg A, Keim D A. An efficient approach to clustering in large multimedia databases with noise[C]//KDD. 1998, 98: 58-65.

[11] Ram A, Jalal S, Jalal A S, et al. A density based algorithm for discovering density varied clusters in large spatial databases [J]. International Journal of Computer Applications, 2010, 3(6): 1-4.

[12] Liu P, Zhou D, Wu N. Varied density based spatial clustering of application with noise[C]//proceedings of IEEE Conference ICSSSM 2007 pg 528. 2007, 531.

[13] Xu X, Ester M, Kriegel H P, et al. A distribution-based clustering algorithm for mining in large spatial databases[C]//Data Engineering, 1998. Proceedings., 14th International Conference on. IEEE, 1998: 324-331.

[14] Birant D, Kut A. ST-DBSCAN: An algorithm for clustering spatial–temporal data[J]. Data & Knowledge Engineering, 2007, 60(1): 208-221.

[15] Parimala M, Lopez D, Senthilkumar N C. A survey on density based clustering algorithms for mining large spatial databases[J]. International Journal of Advanced Science and Technology, 2011, 31(1): 59-66.

[16] Sheikholeslami G, Chatterjee S, Zhang A. Wavecluster: A multi-resolution clustering approach for very large spatial databases[C]//VLDB. 1998, 98: 428-439.

[17] Grossmann A, Morlet J. Decomposition of Hardy functions into square integrable wavelets of constant shape[J]. SIAM journal on mathematical analysis, 1984, 15(4): 723-736.

[18] Mallat S G. A theory for multiresolution signal decomposition: the wavelet representation[J]. IEEE transactions on pattern analysis and machine intelligence, 1989, 11(7): 674-693.

[19] Hinneburg A, Keim D A. An efficient approach to clustering in large multimedia databases with noise[C]//KDD. 1998, 98: 58-65.

[20] Ankerst M, Breunig M M, Kriegel H P, et al. OPTICS: ordering points to identify the clustering structure[C]//ACM Sigmod record. ACM, 1999, 28(2): 49-60.

[21] Achtert E, Böhm C, Kröger P. DeLi-Clu: boosting robustness, completeness, usability, and efficiency of hierarchical clustering by a closest pair ranking[J]. Advances in knowledge discovery and data mining, 2006: 119-128.

[22] Liao W, Liu Y, Choudhary A. A grid-based clustering algorithm using adaptive mesh refinement[C]//7th Workshop on Mining Scientific and Engineering Datasets of SIAM International Conference on Data Mining. 2004: 61-69.