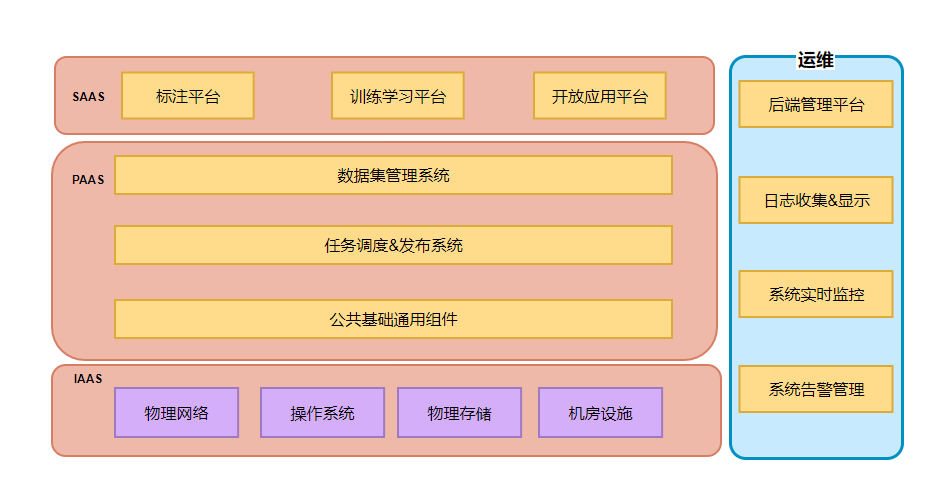
### MetaMind平台介绍



* SAAS层提供SAAS级别的云服务，用户通过SAAS的页面来操作平台
* PAAS容器调度平台
* IAAS 层资源需要用户方完整提供
* 运维 MetaMind维护管理

#### 计算机视觉技术

支持图像分类、检测、分割等任务；通过神经架构搜索技术组建出一定数量的模型进行训练，并且还可以针对重点、难点进行额外的优化与改进；模型分为高精度与高性能两类供客户选择，满足客户需求；适用于大部分分类场景，如细粒度分类、常规分类以及小样本分类。团队积累了大量顶会竞赛经验，在优化模型方面有我们独到的见解，精简训练过程，大大减少了使用者的门槛；通过简单的操作就能完成所有流程，可以让没有算法经验的人也可以达到算法工程师的效果，节省了大量人力物力。

#### 机器学习技术

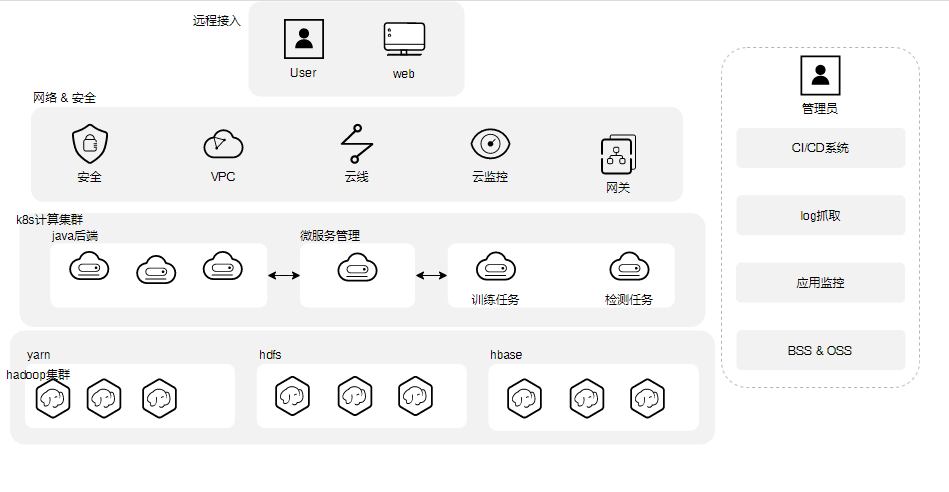
支持多种机器学习任务包括分类，回归，时序任务等。 在数据输入时，我们基本只需要的原始数据文件与几个必要的配置信息，支持多表(一个主表多个副表)输入。在模型发布时，平台支持以api的方式云发布与离线发布。在自动数据预处理方面，我们除了做空值填充，编码等处理；还会做组合编码，分层采样等特殊处理，有利于提升效果；还会使用cython和numpy加速以及支持数据可视化分析。 特征工程方面，我们积累了很多大赛冠军经验，有丰富的特征库，它能自动生成很多特征并可以快速筛选有效特征。模型方面，我们会跑多个模型对比效果，自动选择模型以及支持多种模型融合的方式；另外有一套自己的调参策略，能快速调出效果表现好的参数。

#### 自然语言处理技术

平台支持文本分类、文本匹配、序列标注三个自然语言处理方向，依据用户上传的数据集和用户对模型性能的要求，自动化选择符合该数据的最优模型。针对选择的模型进行文本的预处理，参数优化来提高模型最终的性能。并结合预训练模型BERT来构建高精度模型，满足用户对模型高精度的要求。对训练好的网络能够在平台自动化部署，节省人力以及时间成本。

### 用户环境对接参考

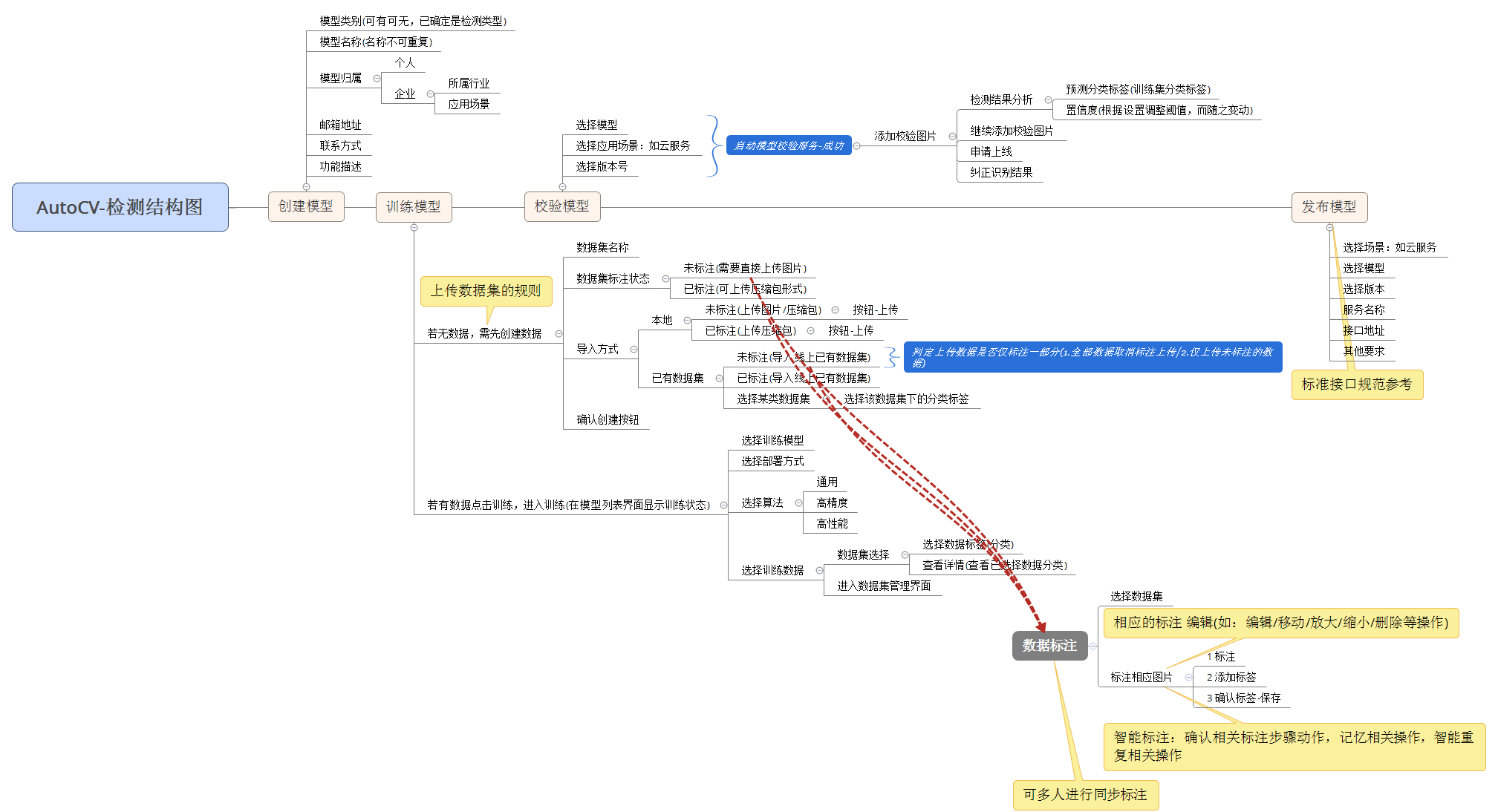
* 物理视图



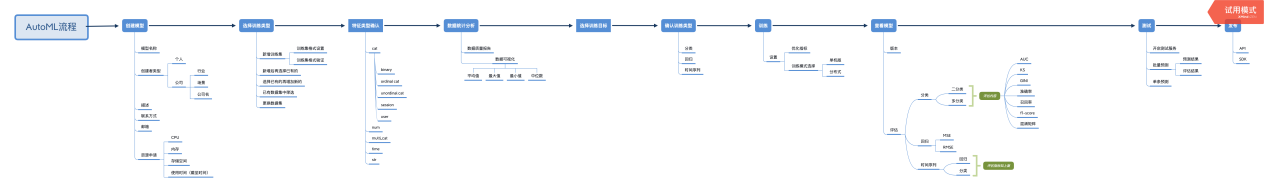
## 子系统功能定义

### WEB操作页面流程

* autocv的用户操作视图示例

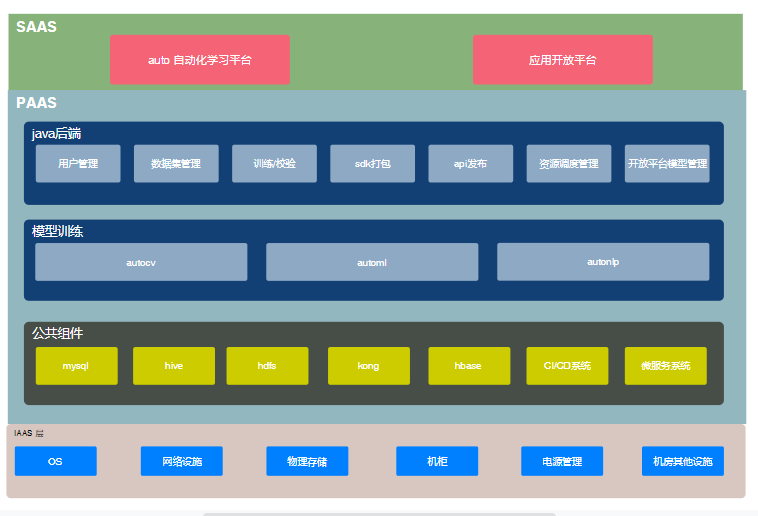


* automl的用户操作视图示例



### JAVA后端平台分解

#### 软件组件略图



#### 软件组件描述

##### Java后端

平台系统的控制核心，负责用户管理，数据集管理，训练任务管理，api发布，模型管理等功能。

##### 模型训练

目前适配有主流的cv，ml，nlp三个方向的训练任务

##### 公共组件

基础的开源组件

## 配置使用管理

以下内容是目前已经交付给用户的一份安装示例

管理口 | label | 备注

--------- | ----------- | -------

`10.16.101.38` | autotrain=true | `用于模型训练任务`

`10.16.101.39` | autotrain=true | `用于模型训练任务`

`10.16.101.48` | autotrain=true | `用于模型训练任务`

`10.16.101.49` | autotrain=true | `用于模型训练任务`

`10.16.101.58` | gpushare=true | `用于AI开放应用服务`

`10.16.101.59` | gpushare=true | `用于AI开放应用服务`

### kubernetes集群安装部署

#### 1） docker 安装

#### **2） nvidia docker2 安装，以及nvidia runtime配置**

#### **3） apt 源配置**

        deb http://mirrors.aliyun.com/ubuntu/ xenial main restricted

        deb http://mirrors.aliyun.com/ubuntu/ xenial-updates main restricted

        deb http://mirrors.aliyun.com/ubuntu/ xenial universe

        deb http://mirrors.aliyun.com/ubuntu/ xenial-updates universe

        deb http://mirrors.aliyun.com/ubuntu/ xenial multiverse

        deb http://mirrors.aliyun.com/ubuntu/ xenial-updates multiverse

        deb http://mirrors.aliyun.com/ubuntu/ xenial-backports main restricted universe  multiverse

        deb https://mirrors.aliyun.com/kubernetes/apt kubernetes-xenial main

#### **4）Install kubelet kubeadm kubectl组件**

        apt-get update &&  \

        apt-get install -y kubelet kubeadm kubectl --allow-unauthenticated

#### **5） kubeadm add master**

        kubeadm init --pod-network-cidr=10.244.0.0/16 --apiserver-advertise-address=10.16.32.49

        kubeadm token create

        kubeadm token list

    获取ca证书sha256编码hash值

    openssl x509 -pubkey -in /etc/kubernetes/pki/ca.crt | openssl rsa -pubin -outform der 2>/dev/null | openssl dgst -sha256 -hex | sed 's/^.\* //'

#### **6） 加入worker 节点**

    kubeadm join 192.168.0.200:6443 --token rw4enn.mvk547juq7qi2b5f --discovery-token-ca-cert-hash sha256:ba260d5191213382a806a9a7d92c9e6bb09061847c7914b1ac584d0c69471579

#### **7） 选择网络插件**

     参照： https://kubernetes.io/docs/concepts/cluster-administration/addons/

### 调度器定制化部署

#### 1）nvidia默认调度器部署

        kubectl apply -f nvidia-device-plugin.yml

    note： 需要提前再需要调度的nvidia机器上打上标签：

        kubectl label nodes 10.2.2.123 gpushare=false

#### 2）gpu共享调度器部署

    note： 需要提前再需要调度的nvidia机器上打上标签：

        kubectl label nodes 10.2.2.123 gpushare=true

        1. 替换 scheduler-policy-config.json kube-scheduler.yaml 两个文件

        2. Install ali scheduler ： https://github.com/AliyunContainerService/gpushare-scheduler-extender/blob/master/docs/install.md

        3. Install ali device plugin： https://github.com/AliyunContainerService/gpushare-scheduler-extender/blob/master/docs/install.md

### Hadoop数据存储

\* 安装脚本： http://203.195.212.206/hadoop/hadoop\_install.sh

#### 存储hdfs

    启动namenode： ${HADOOP\_HOME}/sbin/hadoop-daemon.sh start namenode

    启动datanode   ${HADOOP\_HOME}/sbin/hadoop-daemon.sh start datanode

#### 存储hbase

    一键启动脚本  ${HBASE\_HOME}/bin/start-hbase.sh

### 对外API网关

#### 1. download rpm package

    https://docs.konghq.com/install/centos/

#### 2. Install rpm package

```shell

 $  sudo yum install epel-release

 $  sudo yum install kong-2.0.3.\*.noarch.rpm --nogpgcheck

```

#### 3. create your dabase for your kong

```shell

    CREATE USER kong; CREATE DATABASE kong OWNER kong;

    $ kong migrations bootstrap [-c /path/to/kong.conf]

```

#### 4. start your kong gateway

```shell

    kong start [-c /path/to/kong.conf]

```

### 后端服务&前端页面程序安装

参照git地址提供的安装脚本： <http://10.16.32.59:30001/root/auto-deploy.git>

### 系统监控

参照git地址提供的安装脚本： <http://10.16.32.59:30001/root/auto-deploy.git>

## 技术支持、培训与服务

### 用户使用文档

重点参照下面文档：

* 《图像分类操作文档.docx》
* 《目标检测.docx》
* 《ML操作文档.docx》
* 《NLP操作文档.docx》

### 工单系统支持

请登录console.deepblueai.com填写发送工单

### 日常保障

专项技术服务组将长期复杂该项目的日常维护，系统运行期间，维护工程师将进行每年的例行系统检查，最大程度地避免系统隐患的发生。

### 紧急保障

本产品中的所有素材文件原始材料在维护期间均开通专门的服务器进行整理和备份，当由于客户人为操作或其他未知因素产生数据资料丢失时，我们会及时提供备份文件和资料将客户的损失减少到最低。

### 响应时间

|  |  |  |
| --- | --- | --- |
| 故障级别 | 定义 | 深兰故障响应时间 |
| 一级故障 | 主要指系统在运行中出现系统瘫痪或服务中断，导致设备的基本功能不能实现或全面退化的故障 | 0.5小时内响应，在2小时内提出故障解决方案，4小时内恢复设备正常运行 |
| 二级故障 | 主要指系统在运行中出现的故障具有潜在的系统瘫痪或服务中断的危险，并可能导致设备的基本功能不能实现或全面退化 | 1小时内响应，在2小时内提出故障解决方案，4小时内恢复设备正常运行 |
| 三级故障 | 主要指系统在运行中出现的直接影响服务、导致系统性能或服务部分退化的故障 | 1小时内响应，在2 小时内提出故障解决方案，4小时内恢复设备正常运行 |
| 四级故障 | 主要指系统在运行中出现的，断续或间接的影响系统功能和服务的故障 | 1小时内响应，在3小时内提出故障解决方案，4小时内恢复设备正常运行 |

## 授权

### 标注平台

待定。

### 训练平台

待定。

### AI开放平台

一般意义按照用户调用接口次数收费，具体参照产品文档。