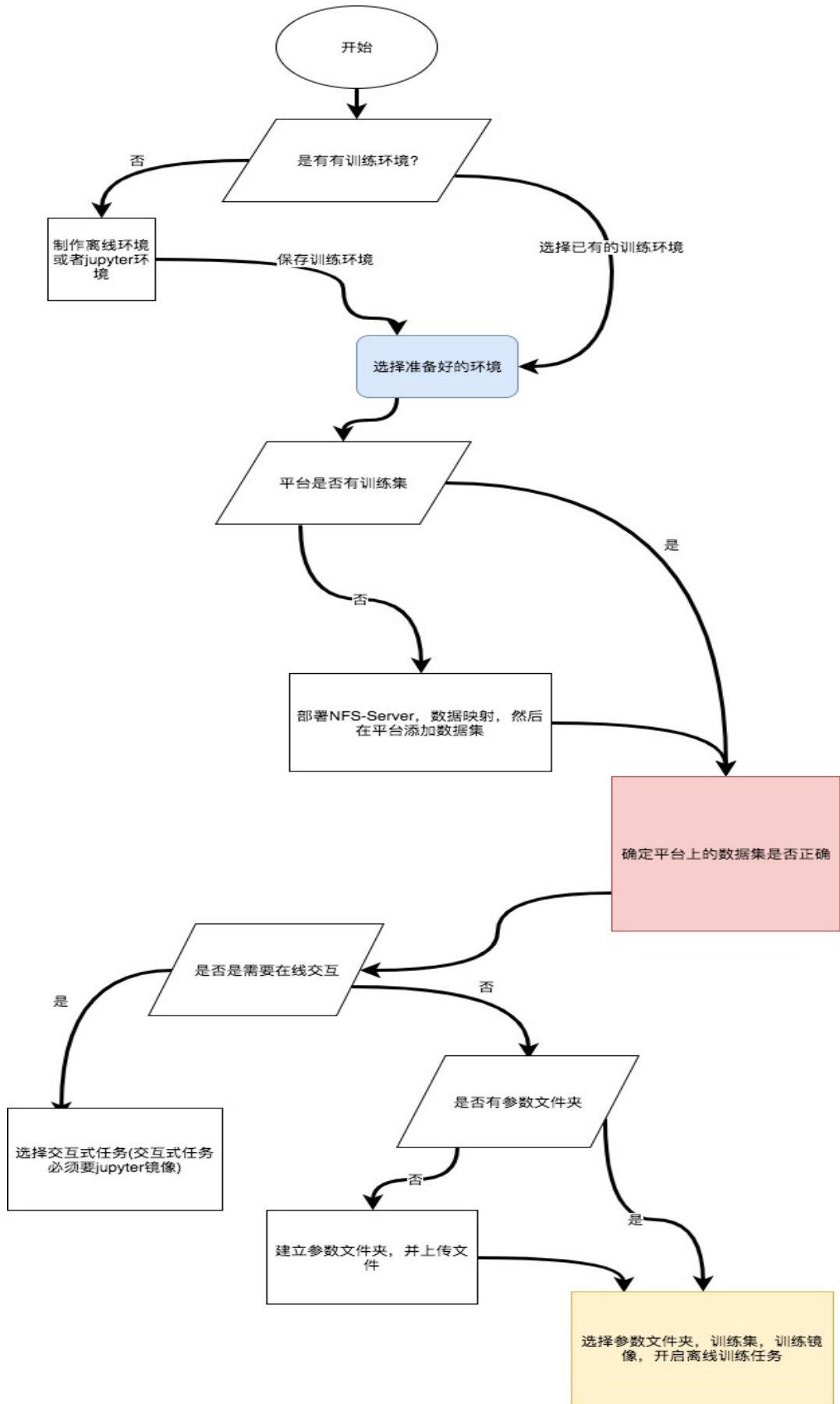


总体的深度学习平台应用的流程图如下：



所有的操作集中在训练中心：

训练中心

集成中心

应用中心

平台与服务监控中心

工作流

训练评测集管理

历史所有的训练任务的状态，日志以及管理操作等

训练评测参数文件夹管理

训练评测参数文件夹管理

训练评测结果管理

训练评测结果管理

Debug文件夹管理

Debug文件夹管理

训练与评测任务管理

查询，查看和管理所有的训练集 / 测试集

集成服务中心

集成服务中心

交互式训练

交互式训练

网络可视化

caffe网络配置的可视化工具

PR曲线查看模块

PR曲线查看模块

Step-Loss-ACC曲线查看模块

Step-Loss-ACC曲线查看模块

环境搭建平台

环境搭建平台

在进行训练或者评测之前，首先需要判断是否有可用的离线训练环境，如果没有可用的离线训练环境。选择“环境搭建平台”，基于已有的镜像建立一个新的镜像。

DEEPCLOUD

深度学习

数据中心

训练中心

集成中心

应用中心

平台与服务监控中心

工作流

DG-Net

环境搭建平台

1 生成contain\_id

repository:

运行

2 显示终端界面

3 commit

4 push

选择所需的镜像，进行环境的配置，然后对于已经配置好的环境进行提交（commit按钮）对于镜像的名字应该是”新的镜像所基于的原镜像名\_\_用户名 \_\_创建的时间“，对于对应的版本号“以v开头加上版本号”，最后推送到服务器中（push按钮），经过这些步骤，新的镜像就制作完成。

DEEPLINT  
格 灵 深 瞳

目 数据中心

🔧 训练中心

📁 集成中心

🔧 应用中心

📶 平台与服务监控中心

🏠 工作流

DG-Net

🔍 dgnet

环境搭建平台

✓ 生成contain\_id

repository: registry.deepglint.com/jupyter:v0.3

运行

✓ 显示终端界面

root@37160446f675:/train/execute# command

3 commit

\* repository: registry.deepglint.com/ jupyter\_hotpot\_20180101

\* tag: v0.1

commit

4 push

在代码调试阶段，选择“交互式训练”，在选中image的jupyter对应的版本号v:0.3，然后进行交互式训练修改代码。（密码是：deeplearning），交互式训练的创建可能会较慢，需要耐心等待一下，或者刷新一下。

DEEPLINT  
格 灵 深 瞳

目 数据中心

🔧 训练中心

📁 集成中心

🔧 应用中心

📶 平台与服务监控中心

🏠 工作流

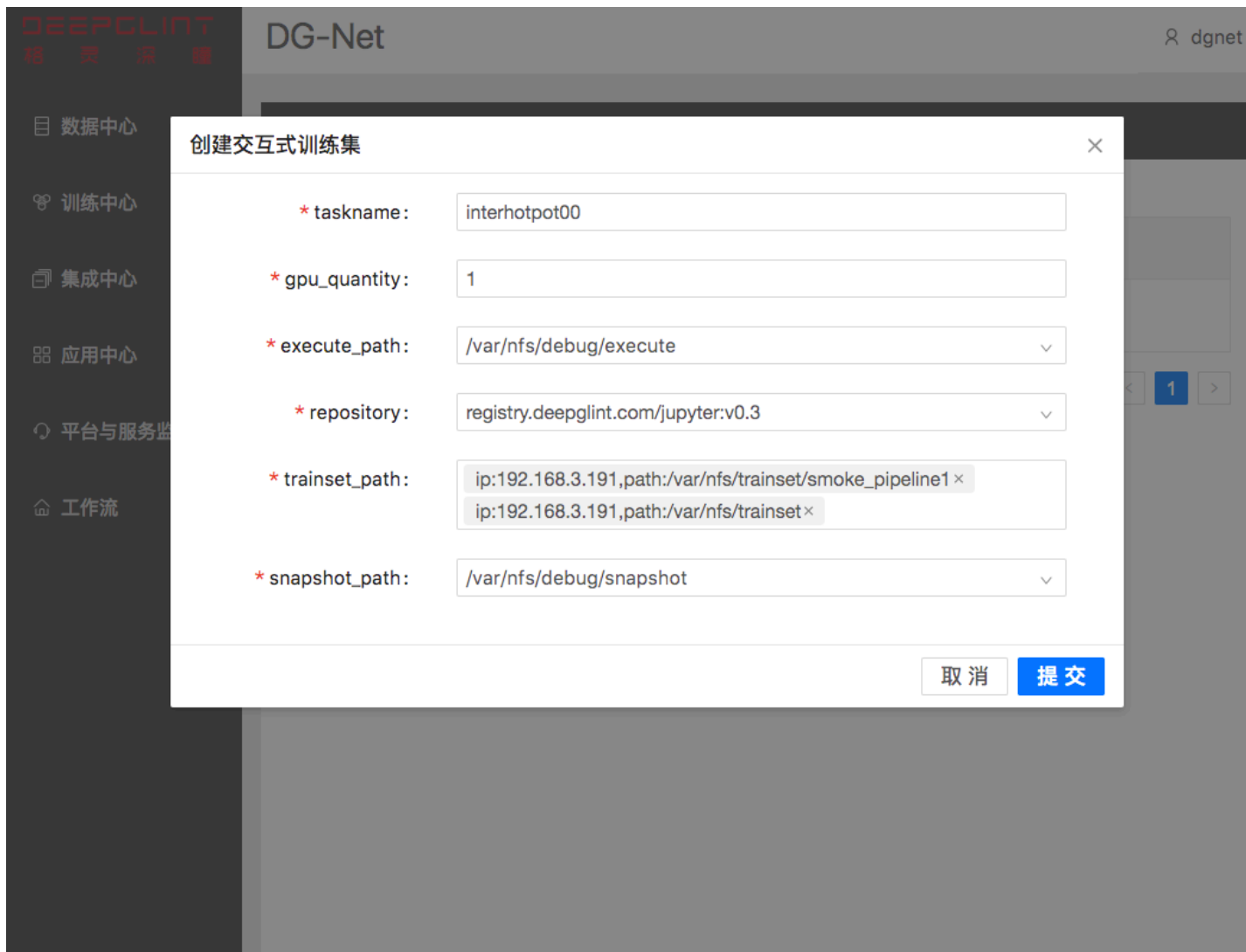
交互训练

创建交互式训练

id	name	socket	status	操作
1	inter100	192.168.3.9:31141	Running	<a href="#">↶ 进入</a> <a href="#">🗑 删除</a>
2	interhy	192.168.3.238:31080	Running	<a href="#">↶ 进入</a> <a href="#">🗑 删除</a>

[<](#) [1](#) [>](#)

点击创建交互式训练，填写对应的内容。对于taskname需要以“inter”为开头，后面接用户的名字，和一些用户喜欢的数字。因为每个taskname需要唯一，所以在设置最后的数字时候不要重复。



在进行交互式训练的时候，在本机（需要ubuntu或者centos的系统机）选择一个空的文件夹输入 `sudo mount -t nfs 192.168.3.191:/var/nfs/debug/execute ./` 然后将本地的运行代码等放入这个空的文件夹，这样你的代码就映射到交互式训练中，程序在交互式系统的位置为 `/train/execute/`

进行模型训练，分成如下几个步骤：

- 1.数据集需要在数据所在位置搭建nfs-server服务，然后添加数据到平台指定的位置：`"/var/nfs/trainset"`下，然后将映射后的文件目录增加到平台

目 数据中心

🔗 训练中心

📁 集成中心

🔧 应用中心

📊 平台与服务监控中心

🏠 工作流

训练评测集管理

训练集管理

评测集管理

add trainset

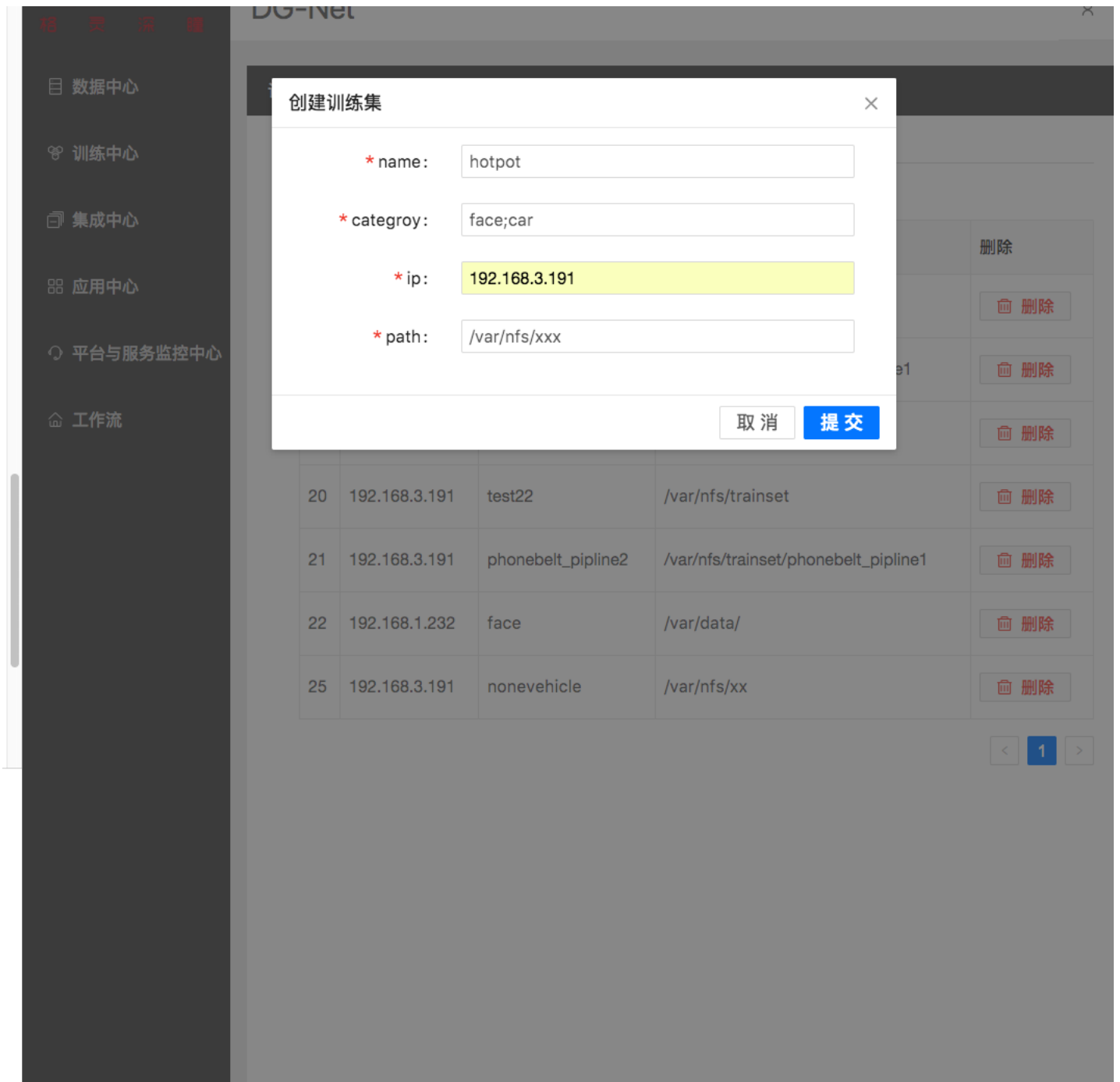
id	ip	name	path	删除
11	192.168.3.191	trainset2	/var/nfs/trainset	<div>🗑 删除</div>
12	192.168.3.191	smoke_pipeline1	/var/nfs/trainset/smoke_pipeline1	<div>🗑 删除</div>
18	192.168.3.191	train20171226	/var/nfs/trainset/origin	<div>🗑 删除</div>
20	192.168.3.191	test22	/var/nfs/trainset	<div>🗑 删除</div>
21	192.168.3.191	phonebelt_pipeline2	/var/nfs/trainset/phonebelt_pipeline1	<div>🗑 删除</div>
22	192.168.1.232	face	/var/data/	<div>🗑 删除</div>
25	192.168.3.191	nonevehicle	/var/nfs/xx	<div>🗑 删除</div>

<

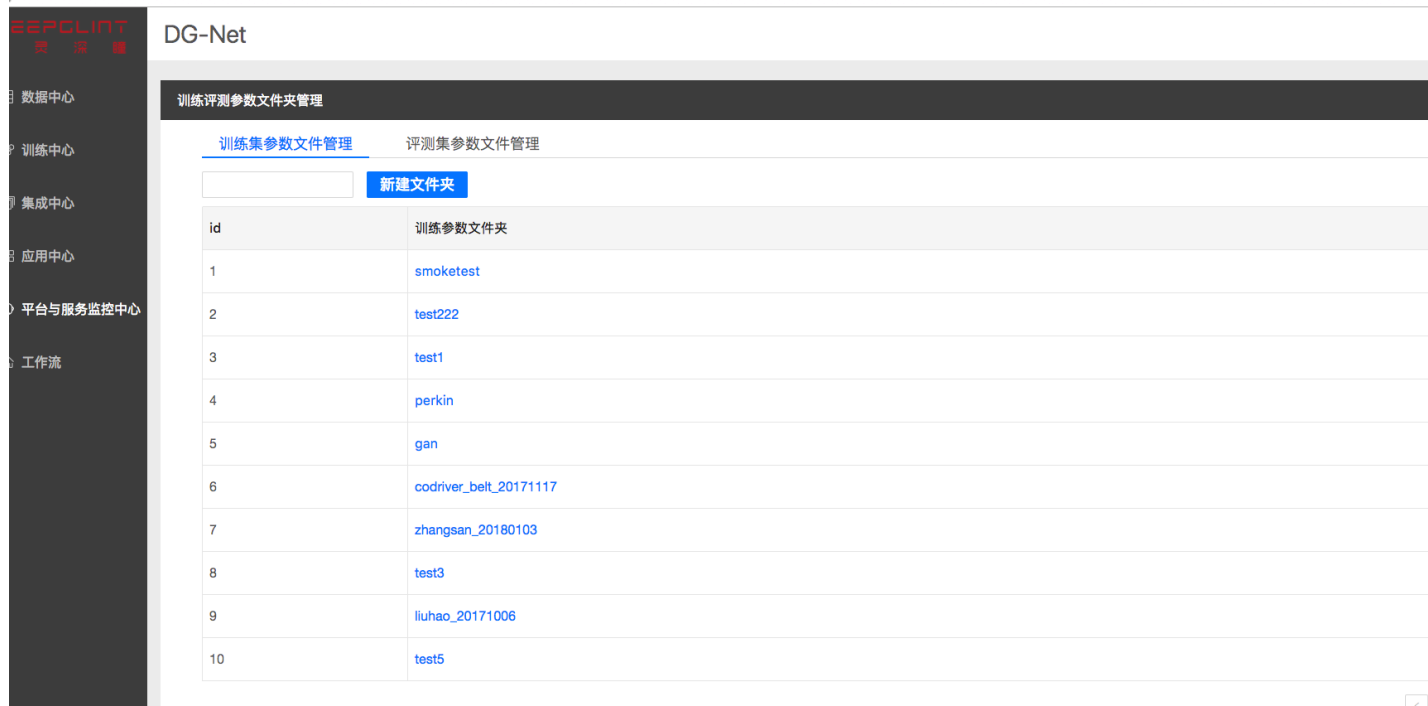
1

>

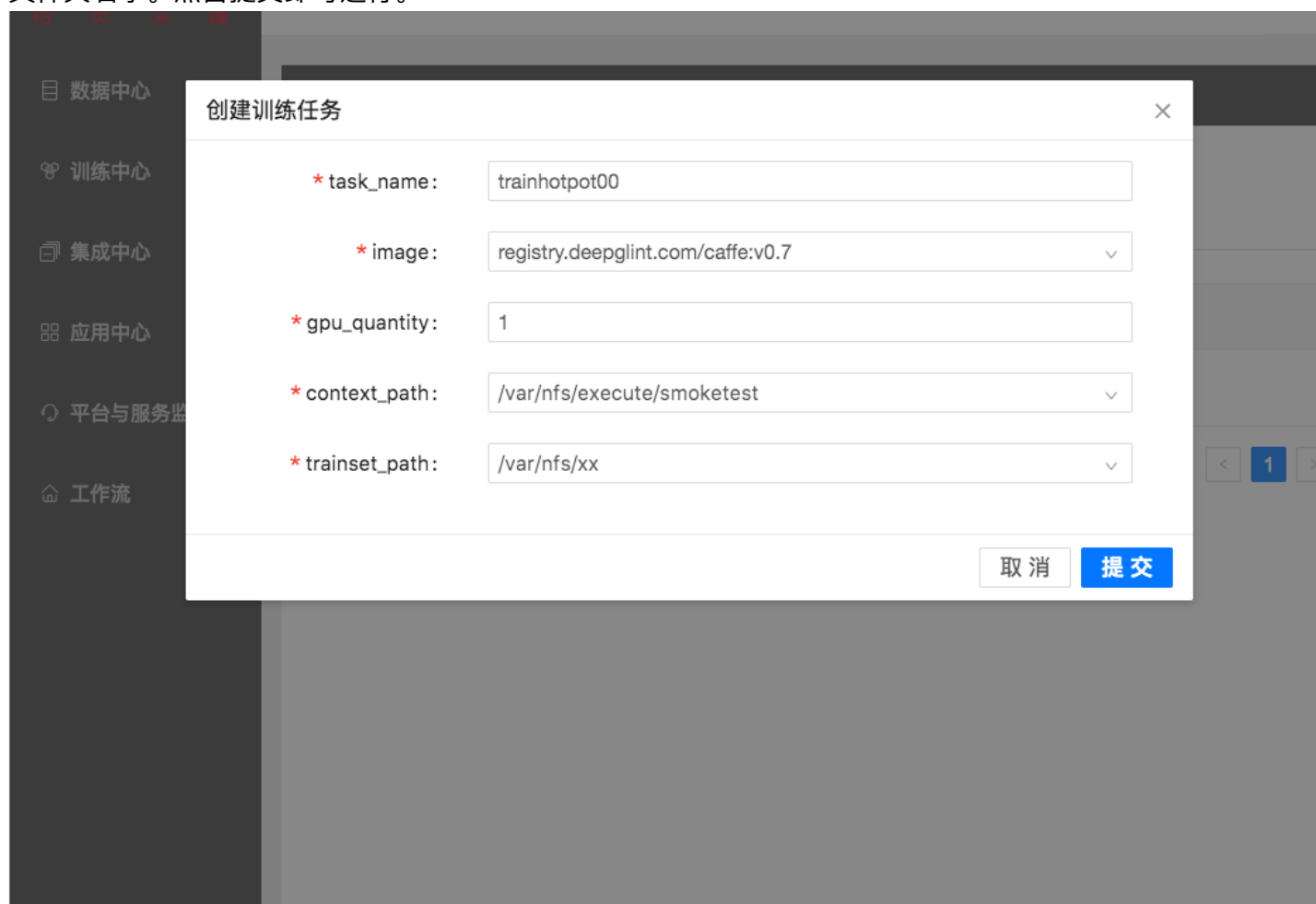
数据内容的添加：对于category是对应数据的类型，如人脸，机动车等。对于ip需要确定数据集目前存放的宿主机，参考总流程判定接下来需要的操作。



2.对于训练的代码需要上传，选择“训练评测参数文件夹管理”。建立自己的文件夹，点击建立的文件夹，将程序上传到自己的文件夹下（upload，目前只支持单文件上传）。



3.对于进行训练操作，选择“训练与评测任务管理”。选择“训练”，然后选中自己的训练模型用的数据文件，以及程序文件。具体参数可以参考图中，taskname需要以“train”开头，然后加上用户的名字，最后是一个数字。需要保证taskname的唯一性，所以数字不要重复。image选择自己最开始制作的离线环境。对于contextpath则是上一步将程序上传所建立的文件夹，对于trainsetpath则是第一步需要将数据上传时候建立的文件夹名字。点击提交即可运行。



4.最后选择“训练评测结果管理”，查看自己的训练模型结果

目 数据中心

步 训练中心

司 集成中心

器 应用中心

平台与服务监控中心

工作流

训练评测结果管理

训练结果文件管理

评测结果文件管理

新建文件夹

id	训练结果文件夹
1	<a href="#">train0001</a>
2	<a href="#">train20171226</a>
3	<a href="#">test100</a>
4	<a href="#">train01</a>
5	<a href="#">train</a>
6	<a href="#">train199</a>

对于进行评测的训练，过程同上，只是需要将训练改成评测即可。如果有不明白的地方可以随时联系 [yuhengli@deepglint.com](mailto:yuhengli@deepglint.com) 和 [yuboliu@deepglint.com](mailto:yuboliu@deepglint.com)



## FAQ:

---

- 数据如何上传:

搭建nfs-server的方式将数据映射到指定位置 (参考前文) 对于nfs-server的搭建参考: nfs搭建文档.pdf (随本说明文档一起发送)

- 平台内各个上传部分的位置:

模型训练部分 (包括在模型生产时候的训练和测试):

训练的执行脚本的位置 (需要一个train.sh,启动全部的测试程序, 将需要安装的第三方包pip install XXX 此类命令写入train.sh中,对于caffe由于有两种变异方式, make和cmake, base-image基于cmake程序, 所以调用方式为/opt/caffe/build/....而非/opt/caffe/.build\_release...这样的形式):

/train/execute/xxx

训练的被测试样本 (各种图片的位置):

/train/trainset/1/xxx

训练的模型输出位置 (model的位置):

/train/snapshot/xxx

模型测试部分 (此部分是已经生成的模型, 再次使用新数据进行测试):

测试的执行脚本的位置 (需要一个eval.sh,启动全部的测试程序):

/eval/execute/xxx

测试的被测试文件 (各种图片的位置):

/eval/evalset/xxx

测试的模型位置 (model的位置):

/eval/snapshot/xxx

测试的结果输出位置:

/eval/evalresult/xxx

平台下的caffe目录位置为:

/opt/caffe

- 需要将代码里的文件位置全部修改为平台内的绝对路径, 尽量不要使用相对路径。

## 名词解释:

离线训练环境, 一个包含训练环境的系统:

- 1.包括训练所需的caffe, mxnet, TensorFlow, pytorch的一个或者几个深度学习框架;
- 2.python所使用的非系统自带的安装包 (需要pip install 的安装包)
- 3.各种软连接 (ln -s) 等已经配置好

## 一些可避免的错误或者bug (不断更新) :

- 1.在写程序的调用位置时, 确定“/”是否需要添加, 很多报错的原因是找不到路径
- 2.尽量不要使用一些linux交互式的命令 如: fish, tmux, byobu等, 因为系统本身就处于交互式环境, 如果再加上交互式命令, 可能会有假死的bug出现。

如果哪里不明白, 可以随时进行交流。