Dockerfile标准

一、基础镜像

dockerfile中通过from关键字指定基础镜像，基础镜像可以是网上已用的，也可以自己制作（详见模型镜像制作过程文档）

```
From 基础镜像
```

二、环境变量

由于模型被制作成镜像后，就无法手动在代码中修改参数，因此通过代码中通过读取环境变量的方式设置这些参数，这样在平台中使用这些参数时，我们可以通过修改环境变量来达到修改模型参数的目的

环境变量分为两种：

A类：一种是在dockerfile中使用的，在dockerfile中使用是为了上传模型时为平台提供某种信息

B类：一种是只要在代码中使用的，起到修改模型参数的作用

1、适配数据类型

A类环境变量

该环境变量用于标识此镜像中的模型支持何种数据类型

环境变量名为 MODEL_DATA_TYPE

可选值及其意义为：

```
音频audio、视频video、点云pointcloud、图片image
```

2、模型功能

A类环境变量

环境变量名称用于标识该镜像中的模型具有哪些功能，以下依次为训练功能，模型转换功能，推理功能

```
MODEL_TRAINING   训练功能
MODEL_CONVERSION     模型转换功能
MODEL_INFERENCE 推理/部署功能
```

环境变量可选值：true为具有该功能，false表示不具有该功能

3、模型参数

A类环境变量

模型运行时的超参

环境变量名 模型需要的超参以HP（hyper param）为前缀，目前平台可识别的超参为:

```
HP_LEARNING_RATE      学习率

HP_RATE_DECAY     学习率衰减

HP_WEIGHT_DECAY 权重衰减

HP_MOMENTUN 动量

HP_ITERATIONS     迭代次数
```

模型转换用的超参以CONVERT为前缀：

```
CONVERT_RKNN_QUANTIZED_DTYPE

CONVERT_RKNN_QUANTIZED_IMG_NUM

CONVERT_REORDER_CHANNEL
```

环境变量可选值：环境变量的值代表改超参的变量类型及其取值范围信息

```
格式：  ENV 超参名称 变量类型:取值范围:默认值:输入规范要求

其中变量类型可选值：numeric, boolean, enum, string
对于数值类型取值范围，其中闭区间[a,b]表示为该超参的取值范围，a、b分别为取值上界和下界，若不存在界则对应位置无数值，如[a,]表示该超参取值范围需大于等于a，开区间同理，只是开区间无法取到断点值。
对于布尔类型，取值范围为true和false，取值范围字段不用给出
对于枚举类型，格式为  enum:{枚举值a,枚举值b}:枚举值a

缺少的内容为空，也使用 ':'  与其他部分分隔开

例子：
ENV  HP_LEARNING_RATE  numeric:(,):0.01
该环境变量表示，HP_LEARNING_RATE这个超参为数值型（numeric）的参数，取值范围为负无穷大到正无穷大,0.01为默认值
ENV  CONVERT_RKNN_QUANTIZED_DTYPE enum:{dynamic_fixed_point-i8,dynamic_fixed_point-i16,asymmetric_quantized-u8}:asymmetric_quantized-u8
该环境变量表示,CONVERT_RKNN_QUANTIZED_DTYPE这个参数是枚举型（enum）的参数，默认值为‘asymmetric_quantized-u8'，可接受的值包括‘asymmetric_quantized-u8','dynamic_fixed_point-i8','dynamic_fixed_point-i16’
ENV  CONVERT_RKNN_MEAN_VALUES  string:::输入按如下公式处理`y=(x-mean)/std`，默认值为'0_0_0'，输入的字符串以'#'分割不同的输入，以'_'分割每一个输入的不同通道，如‘0 1 2#3 4’
```

4、模型当前功能

B类环境变量

如2中所看到的，模型功能可能不止一种，因此需要通过读取环境变量，获知模型目前需要执行它的哪一种功能

环境变量名为 MODEL_WORKING_MODE

可取值及其意义如下：

```
1（训练模式），2（模型推理），3（模型转换），4（手动验证）,5（质检功能）
```

需要注意的是其为B类环境变量，不需要在dockerfile中使用，因为它只是用于告诉模型代码目前模型应该以哪种功能模式启动

例如在python中，通过代码model_working_mode = int(os.getenv("MODEL_WORKING_MODE"))获取该环境变量，这样通过if-else之类的判断语句来执行不同的代码功能（训练、转换、推理/部署）

```
if model_working_mode == 1:
        train() //执行模型训练代码
elif model_working_mode == 2:
        convert() //执行模型推理代码
```

5、GPU资源需求

标识模型是否需要gpu资源

A类环境变量

环境变量名称：USE_GPU

可取值及其意义：

```
true    模型需要gpu环境
false   模型不需要gpu环境
```

6、原始流地址

B类环境变量

在部署时，模型需要获取需要进行推理的原始流地址

环境变量名称：monitorUrl

可取值及其意义：

原始流的rtsp地址，如 rtsp://admin:lab315315@210.30.97.174:33100

7、推理流地址

B类环境变量

原始流经过经过推理后，需要一个推理流地址，将推理结果流写入到该地址中

环境变量名称：streamUrl

可取值及其意义：

原始流的rtsp地址，如 rtsp://210.30.96.101:31004/modelVersion_3/monitor_1

8、自定义配置信息

B类环境变量

模型需要的自定义配置信息

环境变量名称：customConfig

9、报警文件存储位置

B类环境变量

模型进行报警时可能存储的报警文件，如报警图片，需要存储到某个文件夹中，该变量用于指示模型代码需要存储报警文件的位置

环境变量名称： IMG_SAVE_PATH

可取值及其意义：

例如：/root/alert 代表模型代码需要存储报警文件存储中/root/alert 目录下

9、算法名称

B类环境变量

算法名称

环境变量名称： algName

10、摄像头名称

B类环境变量

摄像头名称

环境变量名称： monitorName

11、平台ip及报警接口、配置上报的端口

B类环境变量

模型代码在报警时，不仅需要将报警文件存储到**IMG_SAVE_PATH**环境变量指示的目录中，还需要发送报警信息给平台

环境变量名称： IP、PORT　　其中IP为平台ip地址，port为报警接口的端口

通过以下方式拼接出报警接口

```
http://ipStr:portStr/ma/daemon/alertUpload
其中ipStr、portStr为通过环境变量 IP、PORT的值
```

通过以下方式拼接出配置上报接口

```
http://ip:port/ma/daemon/configUpload
```

三、模型相关规范

1、相关文件夹

模型从/dataset中读取数据集

若模型有权重文件生成，则存储在/weight目录下。若要导入权重文件，也从/weight目录下导入

2、代码规范

通过读取环境变量的方式获取第一节中的参数

（1）数据集读取

算法从/dataset/train中读取训练用数据集，/dataset/test中读取测试用数据集，/dataset/validation中读取验证用数据集，/dataset/user_validation/images 用于支持用户上传图片进行验证，同时将推理结果保存在/dataset/user_validation/output中，用于前端展示。/dataset目录结构如下：

```
--dataset
  |--train
    |--训练用数据集
  |--test
    |--测试用数据集
  |--validation
    |--验证用数据集
  |--user_validation
    |--用户上传验证用文件夹
```

具体各文件夹结构如下：

```
--train
  |--annotations
    |--所有的标签
  |--images
    |--所有的图片
```

```
--test
  |--annotations
    |--所有的标签
  |--images
    |--所有的图片
```

```
--validation
  |--annotations
    |--所有的标签
  |--images
    |--所有的图片
```

```
--user_validation
  |--images
    |--所有的图片
  |--output
    |--验证结果
```

如果使用了数据集切分功能，也由平台将切分后数据集挂载至这三个目录 train、test、validation，目录中的images依旧存储所有图片，但annotations目录中只存储切分后的标注文件

对于Ultralytics YOLO 格式的数据集，在dataset目录下还将有一个yaml文件，如下：

```
--dataset
  |--data.yaml
  |--train
    |--训练用数据集
  |--test
    |--测试用数据集
  |--validation
    |--验证用数据集
  |--user_validation
    |--用户上传验证用文件夹
```

data.yaml内容用于说明数据集的类目字典，如下

```
# Classes (80 COCO classes)
names:
  0: person
  1: bicycle
  2: car
  # ...
  77: teddy bear
  78: hair drier
  79: toothbrush
```

（2）权重文件导入导出

若模型有权重文件生成，则存储在/weight/output目录下。若要导入权重文件，从/weight/input目录下导入，模型转换后的模型存储在/weight/output_convert

```
--weight
  |--input
    |--预训练的模型挂载在该目录，代码从中导入预训练模型的文件
  |--output
    |--训练后生产的模型文件存储在该目录下
  |--output_convert
    |--进行模型转换后生成的模型文件存储在该目录下
```

（3）打印规范

训练模型过程打印内容：

当前的Epoch 当前loss 当前accuracy

```
 Epoch:  0 | train loss: 2.3073 | test accuracy: 0.10

 Epoch:  1 | train loss: 2.2824 | test accuracy: 0.10

 Epoch:  2 | train loss: 2.2693 | test accuracy: 0.10
```

测试打印内容：

```
Total sample number: 460 图片总数
tested: 100, correct: 100, accuracy: 0.02  打印时已测试的图片数，准确的图片数，准确率
```

```
Total sample number: 460
tested: 100, correct: 100, accuracy: 0.02
tested: 200, correct: 200, accuracy: 0.03
tested: 300, correct: 300, accuracy: 0.04
tested: 400, correct: 400, accuracy: 0.06
tested: 460, correct: 460, accuracy: 0.06
```

四、dockerfile示例

```
FROM 基础镜像

ENV MODEL_DATA_TYPE picture  //表示算法适用的数据类型为图片

ENV  MODEL_TRAINING true   //表示改算法有训练功能
ENV  MODEL_CONVERSION true   //表示改算法有模型转换功能
ENV  MODEL_INFERENCE false  //表示改算法有部署推理功能

ENV  HP_LEARNING_RATE  numeric: (,)         //超参  数值型 无取值范围限制
ENV  HP_RATE_DECAY  numeric:(,)            //超参  数值型 无取值范围限制
ENV  HP_ITERATIONS  numeric:[100,500]     //超参  数值型 取值范围为[100,500]

ENTRYPOINT python alg.py   //代码入口
```

以上为dockerfile文件内容
若算法生成报警图片/视频等信息，则存储在/alert目录下