aicity sdk 模型配置文件编写指南

版本： v0.1.0

时间：2021/10/14

编者：AI计算组

目录

---

1. 简介

aicity sdk 是一个用于深度学习模型推理部署的API库；支持多种平台，如cpu， gpu， atlas；支持多种模型格式，如mxnet，acl，trt，onnx；具备低代码开发特性，内置常用前后处理和推理算子，在将通用接口集成调通后，用户只需要根据模型特点编写一个简单的配置文件即可支持新模型推理运行。

2. 版本说明和约束

v0.1.0-20211014:

- 同v0.0.1, v0.1.0版本改动不影响这部分

v0.0.1-20210917:

- 支持atlas平台的yolov3和yolov5检测算法
- 暂不支持trt和onnx模型
- gpu版本未严格自测

3. 快速上手

以ALTAS平台city检测算法acl格式模型为例

步骤1. 准备好模型文件

如：city-det-acl-b1-0.0.0.bin

> 模型发布前要完成格式转换-打包和加密等操作。

步骤2. 创建一个模型配置文件

city-det-acl-b1-0.0.0.json

> 配置文件采用json格式编写

添加一个大括号作为json主体，具体内容信息都是key:value键值对, 键值对之间用','隔开

```
{
  "key1":value1,
  "key2":value2
}
```

步骤3. 填写模型基础信息

```
  "model_file": "city-det-acl-b1-0.0.0.bin",
  "task_type": "DETECT",
  "model_format": "ACL",
  "class_names":
["type1","type2","type3","type4","type5","type6","type7","type8","type9","type10
","type11","type12","type13"],
  "class_num": 13,
  "batch_size": 1,
  "channel_num": 3,
  "image_height": 416,
  "image_width": 416,
```

模型基础信息主要包括模型文件、类型、目标类别和NCHW等信息。

"model_file"就是让指定模型文件名，这里就是步骤1中准备好的模型文件city-det-acl-b1-0.0.0.bin

> 具体的参数信息详情见后文，下同。

步骤4. 填写前处理算子信息

```
 "preprocess":{
    "operator": "PreResize",
    "padding_type":"FULL"
  },
```

根据模型具体情况选择合适的前处理算子，并配置需要的参数。

前处理缩放用到的图像大小会，sdk内部会从模型基础信息里自动获取。

步骤5. 填写模型推理算子信息

```
  "model_infer":{
    "operator": "InferAcl",
    "output_num": 3
  },
```

根据模型具体情况选择正确的推理算子，并配置需要的参数。

模型是ATLAS平台acl格式的，所以这里选择"InferAcl"算子。

这个模型是yolov3的，有3个输出，所以配置output_num为3

步骤6. 填写后处理算子信息

```
  "postprocess": {
    "operator": "PostDetectResultYolov3",
    "conf_thresh":0.1,
    "nms_thresh": 0.45,
    "anchors": [8, 13,  11, 23,  25, 22,  17, 38,  29, 52,  56, 38,  50, 91,
93, 81, 145,199]
  }
```

根据模型具体情况选择合适的后处理算子，并配置需要的参数。

yolov3的模型后处理算子这里选择"PostDetectResultYolov3"。

anchors也配置到该字段里面。

步骤7. 完成模型配置文件编写

完整的例子如下

```
{
  "model_file": "city-det-acl-b1-0.0.0.bin",
  "task_type": "DETECT",
  "model_format": "ACL",
  "class_names":
["type1","type2","type3","type4","type5","type6","type7","type8","type9","type10
","type11","type12","type13"],
  "class_num": 13,
  "batch_size": 1,
  "channel_num": 3,
  "image_height": 416,
```

```
  "image_width": 416,
  "preprocess":{
    "operator": "PreResize",
    "padding_type":"FULL"
  },
  "model_infer":{
    "operator": "InferAcl",
    "output_num": 3
  },
  "postprocess": {
    "operator": "PostDetectResultYolov3",
    "conf_thresh":0.1,
    "nms_thresh": 0.45,
    "anchors": [8, 13,  11, 23,  25, 22,  17, 38,  29, 52,  56, 38,  50, 91,
 93, 81, 145,199]
  }
}
```

步骤8. 跟模型文件一起发布

city-det-acl-b1-0.0.0.bin

city-det-acl-b1-0.0.0.json

让配置文件跟模型文件保持在一个路径下面，sdk具体处理时会根据配置文件找到模型文件。

4. 模型基础信息

参数	类型	必须/可选	说明	示例
model_file	string	必须	模型文件，模型文件与配置文件一一对应	"model_file":"city-det-acl-b1-0.0.0.bin"
task_type	string	必须	任务类型，模型要处理的任务类型主要有检测, 分类, 语义分割等, 目前sdk支持的任务类型选项包括{"DETECT", "CLASSIFY"}, atlas平台目前仅支持"DETECT"	"task_type":"DETECT"
model_format	string	必须	模型格式，模型格式主要有mxnet, atlas acl, trt, onnx等, 目前sdk支持的模型格式选项包括{"MXNET", "ACL"}, atlas平台仅支持"ACL"	"model_format": "ACL"
class_names	string-array	必须	类别数组，检测和分类模型的输出结果都对应了一组目标类别，具体的类别名称在这里配置	"class_names": ["person","dog"]
class_num	int	必须	类别数量，为了校验类别名数组是否有漏填或多填	"class_num": 2
batch_size	int	必须	batch大小，要跟模型实际batch大小一致，常见的为1或8	"batch_size": 8
channel_num	int	必须	通道数，目前只支持通道数为3的	"channel_num": 3
image_height	int	必须	图像高度，模型输入层要求的图像高度	"image_height": 416
image_width	int	必须	图像宽度，模型输入层要求的图像宽度	"image_width": 416

5. 前处理算子

根据模型实际情况，选择对应的前处理算子。

5.1 PreResize

对输入图像进行缩放，

> 涉及的模型输入大小等参数在模型基础信息里配置，算子里不需要重复配置。

参数	类型	必须/可选	说明	示例
padding_type	string	必须	缩放和填充方式，目前支持的选项包括{"FULL", "LEFT_TOP", "CENTER"}, FULL——将原图像的宽和高按各自比例缩放到模型输入大小，完全平铺填充。 LEFT_TOP——锁定原图像宽高比，按长边的比例缩放，靠左/上填充到目标空间。 CENTER——锁定原图像宽高比，按长边的比例缩放，居中填充到目标空间。	"padding_type":"FULL"

根据模型具体情况选择合适的padding_type，如yolov3可能是"FULL"，而yolov5可能是"CENTER"。

版本约束：

> 目前该算子仅支持atlas平台，gpu暂不支持

5.2 PreResizeNormalize

对图像进行缩放和标准化以及通道分离，

参数	类型	必须/可选	说明	示例
mean	float-array	必须	均值数组，取值(0.0-1.0)，数据会先除以255进行归一化，然后再减均值	"mean":[0.485, 0.456, 0.406]
std	float-array	必须	方差数组，归一化、减均值后的数据，除以方差得到标准化数据	"std": [0.229, 0.224, 0.225]
padding_type	string	可选	缩放和填充方式，选项包括{"FULL", "LEFT_TOP", "CENTER"}	"padding_type":"CENTER"

版本约束：

> 目前该算子仅支持gpu平台，atlas暂不支持
>
> padding_type目前仅支持"CENTER"

> 仅支持BGR数据输入，输出3通道分离

5.3 PreCropResize

将图像先安装roi进行裁剪，然后缩放并填充到目标空间

参数	类型	必须/可选	说明	示例
padding_type	string	必须	缩放和填充方式，目前支持的选项包括{"FULL", "LEFT_TOP", "CENTER"}	"padding_type":"FULL"

版本约束：

> 目前该算子暂不可用

6. 模型推理算子

根据模型格式和平台类型，选择正确的推理算子。

6.1 InferAcl

atlas平台acl格式的模型需要选择该推理算子

参数	类型	必须/可选	说明	示例
output_num	int	必须	模型输出个数	"output_num": 3

使用约束：

> 该算子支持atlas平台acl格式的模型

6.2 InferMxnet

mxnet格式的模型需要选择该推理算子

参数	类型	必须/可选	说明	示例
output_num	int	必须	模型输出个数	"output_num": 3

使用约束：

> 该算子支持mxnet格式的模型推理，

> 支持CPU或GPU执行模式，TRT模式暂不支持

7. 后处理算子

7.1 PostDetectResultYolov3

yolov3检测模型后处理算子

参数	类型	必须/可选	说明	示例
conf_thresh	float	必须	检测置信度阈值	"conf_thresh":0.3
nms_thresh	float	必须	nms阈值	"nms_thresh": 0.45
anchors	int-array	必须	anchors数组	"anchors": [8,13, 11,23, 25,22, 17,38, 29,52, 56,38, 50,91, 93,81, 145,199]

请注意anchors数组正确填写，勿遗漏或多填。

使用约束：

> 该算子支持yolov3模型后处理
>
> 目前仅支持模型输出个数为3的情况

7.2 PostDetectResultYolov5

yolov5检测模型后处理算子

参数	类型	必须/可选	说明	示例
conf_thresh	float	必须	检测置信度阈值	"conf_thresh":0.3
nms_thresh	float	必须	nms阈值	"nms_thresh": 0.35
anchors	int-array	必须	anchors数组	"anchors":[10,13, 16,30, 33,23, 30,61, 62,45, 59,119, 116,90, 156,198, 373,326]

请注意anchors数组正确填写，勿遗漏或多填。

使用约束：

> 该算子支持yolov5模型后处理
>
> 目前仅支持模型输出个数为3的情况

7.3 PostDetectResultCenternet

centernet检测模型后处理算子

参数	类型	必须/可选	说明	示例
conf_thresh	float	必须	检测置信度阈值	"conf_thresh":0.3
nms_thresh	float	必须	nms阈值	"nms_thresh": 0.35
min_size	int	必须	目标框大小阈值	"min_size": 20

使用约束：

> 该算子支持centernet模型后处理
>
> 目前仅支持CPU GPU模式， ATLAS暂不支持

8. 样例

8.1 acl-b1

```
{
  "model_file": "kitchen-uniform-det-acl-b1-0.0.1.bin",
  "task_type": "DETECT",
  "model_format": "ACL",
  "class_names":
["kitchen_hat","non_kitchen_hat","kitchen_uniform","non_kitchen_uniform"],
  "class_num": 4,
  "batch_size": 1,
  "channel_num": 3,
  "image_height": 640,
  "image_width": 640,
  "preprocess":{
    "operator": "PreResize",
    "padding_type":"LEFT_TOP"
  },
  "model_infer":{
    "operator": "InferAcl",
    "output_num": 3
  },
  "postprocess": {
    "operator": "PostDetectResultYolov5",
    "conf_thresh":0.3,
    "nms_thresh": 0.35,
    "anchors": [15, 18, 26, 31, 39, 47, 41, 93, 63, 74, 63, 142, 108, 133, 93,
209, 138, 247]
```

```
  }
}
```

> 注意batch_size跟模型的对应关系

8.2 acl-b8

```
{
  "model_file": "road-damage-det-acl-b8-0.0.1.bin",
  "task_type": "DETECT",
  "model_format": "ACL",
  "class_names": ["road_damage"],
  "class_num": 1,
  "batch_size": 8,
  "channel_num": 3,
  "image_height": 640,
  "image_width": 640,
  "preprocess":{
    "operator": "PreResize",
    "padding_type":"CENTER"
  },
  "model_infer":{
    "operator": "InferAcl",
    "output_num": 3
  },
  "postprocess": {
    "operator": "PostDetectResultYolov5",
    "conf_thresh":0.3,
    "nms_thresh": 0.35,
    "anchors": [10,13, 16,30, 33,23,  30,61, 62,45, 59,119,  116,90, 156,198,
373,326]
  }
}
```

8.3 mxnet-b1

```
{
  "model_file": "communitydog-det-mxnet-b1-0.0.1.bin",
  "task_type": "DETECT",
  "model_format": "MXNET",
  "class_names": ["person","dog"],
  "class_num": 2,
  "batch_size": 1,
  "channel_num": 3,
  "image_height": 512,
  "image_width": 512,
  "preprocess":{
    "operator": "PreResizeNormalize",
    "mean":[0.485, 0.456, 0.406],
    "std": [0.229, 0.224, 0.225]
  },
  "model_infer":{
    "operator": "InferMxnet",
    "output_num": 3
```

```
  },
  "postprocess": {
    "operator": "PostDetectResultCenternet",
    "conf_thresh":0.4,
    "nms_thresh": 0.5,
    "min_size": 20
  }
}
```

————结束