RKNN3-Toolkit API 参考手册

文件标识：RK-YH-YF-412

发布版本：V1.0.0

日期：2026-01-10

文件密级：□绝密 □秘密 □内部资料 ■公开

瑞芯微电子股份有限公司

Rockchip Electronics Co., Ltd.

地址： 福建省福州市铜盘路软件园A区18号

网址： www.rock-chips.com

客户服务电话： +86-4007-700-590

客户服务传真： +86-591-83951833

客户服务邮箱： fae@rock-chips.com

前言

概述

本文是 RKNN3-Toolkit 的 API 参考手册。

RKNN3-Toolkit 是为用户提供在PC平台上进行模型转换、推理和性能评估的开发套件。

读者对象

本文档（本指南）主要适用于以下工程师：

技术支持工程师

软件开发工程师

修订记录

版本号	作者	修改日期	修改说明	核定人
V0.2.0	HPC团队	2025-8-6	初始版本	熊伟
V0.3.0b0	NN团队	2025-9-12	更新load_rknn/init_runtime/inference/list_devices接口说明	熊伟
V0.4.0b0	NN团队	2025-10-18	rknn.config 新增 rknn_build_parallelism llm_head_dtype和llm_head_quantized_method参数说明	熊伟
V0.5.0	NN团队	2025-12-01	支持连板精度分析，更新init_runtime/inference接口说明；新增kvcache_controller/query接口说明	熊伟
V1.0.0	NN团队	2026-01-10	正式版本	熊伟

目录

1 要求

1.1 适用芯片

RKNN3-Toolkit当前版本所支持芯片的型号如下：

- RK1820
- RK1828

1.2 系统依赖说明

使用RKNN3-Toolkit时需要满足以下运行环境要求：

操作系统版本	Ubuntu22.04（x64）	Ubuntu24.04（x64）
Python版本	3.10	3.12

注：

1. 具体python库依赖详见requirements*.txt
2. 本文档主要以Ubuntu 22.04 / Python3.10为例进行说明

1.3 适用的深度学习框架

RKNN3-Toolkit支持的深度学习框架包括Caffe、TensorFlow、TensorFlow Lite、ONNX、DarkNet和PyTorch。

它和各深度学习框架的版本对应关系如下：

RKNN3-Toolkit	Caffe	TensorFlow	TF Lite	ONNX	DarkNet	PyTorch
0.2.0	1.0	1.12.0~ 2.14.0	Schema version=3	1.17.0~ 1.18.0	Commit ID: 810d7f7	2.7.0~ 2.8.0
0.3.0b0	1.0	1.12.0~ 2.14.0	Schema version=3	1.17.0~ 1.18.0	Commit ID: 810d7f7	2.7.0~ 2.8.0
V0.4.0b0	1.0	1.12.0~ 2.14.0	Schema version=3	1.17.0~ 1.18.0	Commit ID: 810d7f7	2.7.0~ 2.8.0
V0.5.0	1.0	1.12.0~ 2.14.0	Schema version=3	1.17.0~ 1.18.0	Commit ID: 810d7f7	2.7.0~ 2.8.0
V1.0.0	1.0	1.12.0~ 2.14.0	Schema version=3	1.17.0~ 1.18.0	Commit ID: 810d7f7	2.7.0~ 2.8.0

注：

1. 由于TensorFlow版本兼容特性，TensorFlow 1.12.0之前版本的导出的pb文件，理论上也是支持的。关于TensorFlow版本兼容性的更多信息，可以参考官方资料：https://www.tensorflow.org/guide/versions?hl=zh-cn

2. 因为TFLite不同版本的schema之间是互不兼容的，所以构建TFLite模型时使用与RKNN3-Toolkit不同版本的schema可能导致加载失败。
3. RKNN3-Toolkit使用的Caffe protocol是基于berkeley官方修改的protocol：https://github.com/BVLC/caffe/tree/master/src/caffe/proto，commit值为828dd10，RKNN3-Toolkit在此基础上新增了一些OP。
4. ONNX release version和opset version、IR version之间的关系参考onnxruntime官网说明：https://github.com/microsoft/onnxruntime/blob/v1.6.0/docs/Versioning.md
5. DarkNet官方Github链接：https://github.com/pjreddie/darknet。RKNN3-Toolkit现在的转换规则是基于master分支的最新提交（commit值：810d7f7）制定的。
6. 加载PyTorch模型（torchscript模型）时，推荐使用相同版本的PyTorch导出模型并转为RKNN模型，前后版本不一致时有可能会导致转RKNN模型失败。

2 API详细说明

2.1 RKNN初始化及释放

在使用RKNN3-Toolkit的所有API接口时，都需要先调用RKNN()方法初始化RKNN对象，不再使用该对象时通过调用该对象的release()方法进行释放。

初始化RKNN对象时，可以设置verbose和verbose_file参数，以打印详细的日志信息。其中verbose参数指定是否要打印详细日志信息；如果设置了verbose_file参数，且verbose参数值为True，日志信息还将写到该参数指定的文件中。

举例如下：

```
# 打印详细的日志信息
rknn = RKNN(verbose=True)

…

rknn.release()
```

2.2 模型配置

在构建RKNN模型之前，需要先对模型进行通道均值、量化图片RGB2BGR转换、量化类型等的配置，这些操作可以通过config接口进行配置。

API	config
描述	设置模型转换参数。
参数	**mean_values**：输入的均值。参数格式是一个列表，列表中包含一个或多个均值子列表，多输入模型对应多个子列表，每个子列表的长度与该输入的通道数一致，例如[[128,128,128]]，表示一个输入的三个通道的值减去128。 默认值为None，表示所有的mean值为0。
	std_values：输入的归一化值。参数格式是一个列表，列表中包含一个或多个归一化值子列表，多输入模型对应多个子列表，每个子列表的长度与该输入的通道数一致，例如[[128,128,128]]，表示设置一个输入的三个通道的值减去均值后再除以128。 默认值为None，表示所有的std值为1。
	quant_img_RGB2BGR：表示在加载量化图像时是否需要先做RGB2BGR的操作。如果有多个输入，则用列表包含起来，如[True, True, False]。默认值为False。 该配置一般用在Caffe的模型上，Caffe模型训练时大多会先对数据集图像进行RGB2BGR转换，此时需将该配置设为True。 另外，该配置只对量化图像格式为jpg/png/bmp有效，npy格式读取时会忽略该配置，因此当模型输入为BGR时，npy也需要为BGR格式。 **该配置仅用于在量化阶段（build接口）读取量化图像或量化精度分析（accuracy_analysis接口），并不会保存在最终的RKNN模型中，因此如果模型的输入为BGR，则在调用RKNN3-Toolkit的inference或C-API的run函数之前，需要保证传入的图像数据也为BGR格式。**

API	config
	quantized_dtype：量化类型，目前支持的量化类型有w8a8、w4a16。默认值为w8a8。 **- w8a8**：权重为8bit非对称量化精度，激活值为8bit非对称量化精度。 **- w4a16**：权重为4bit非对称量化精度，激活值为16bit浮点精度。
	quantized_algorithm：计算每一层的量化参数时采用的量化算法，目前支持的量化算法有：normal，mmse，kl_divergence，gdq及grq。默认值为normal。 **normal**量化算法的特点是速度较快，推荐量化数据集量一般为20-100张左右，更多的数据集量下精度未必会有进一步提升。 **mmse**量化算法由于采用暴力迭代的方式，速度较慢，但通常会比normal具有更高的精度，推荐量化数据集量一般为20-50张左右，用户也可以根据量化时间长短对量化数据集量进行适当增减。 **kl_divergence**量化算法所用时间会比normal多一些，但比mmse会少很多，在某些场景下（feature分布不均匀时）可以得到较好的改善效果，推荐量化数据集量一般为20-100张左右。 **gdq**量化算法只在w4a16下有效，能有效的提高w4a16的权重量化精度，推荐量化数据集量为200张以上。因对算力要求高，须使用GPU进行加速运算。 **grq**量化算法只在w4a16下有效，也能有效的提高w4a16的权重量化精度，推荐量化数据集量为200张以上。同样因对算力要求高，须使用GPU进行加速运算，但grq相比gdq算法速度更快且显存占用更低，因此可优先尝试grq算法。
	quantized_method：目前支持layer，channel或者group{SIZE}。默认值为channel。 - layer：每层的weight只有一套量化参数； - channel：每层的weight的每个输出通道都有一套量化参数，通常情况下channel会比layer精度更高。 - group{SIZE}：在channel的基础上，每个输出通道的weight在输入通道上又按{SIZE}细分为多个小组，每个小组有各自的一套量化参数，通常情况下group{SIZE}会比channel精度更高。{SIZE}是32到256之间的32的倍数值，如group32或group128。目前group{SIZE}只在quantized_dtype为w4a16下有效。
	optimization_level：模型优化等级。默认值为3。 通过修改模型优化等级，可以关掉部分或全部模型转换过程中使用到的优化规则。该参数的默认值为3，打开所有优化选项。值为2或1时关闭一部分可能会对部分模型精度产生影响的优化选项，值为0时关闭所有优化选项。
	target_platform：指定RKNN模型是基于哪个目标芯片平台生成的。目前支持“rk1820”。该参数对大小写不敏感。默认值为None。
	model_pruning：对模型进行无损剪枝。对于权重稀疏的模型，可以减小转换后RKNN模型的大小和计算量。默认值为False。
	dynamic_input：用于根据用户指定的多组输入shape，来模拟动态输入的功能。格式为[[input0_shapeA, input1_shapeA, ...], [input0_shapeB, input1_shapeB, ...], ...]。 默认值为None。 假设原始模型只有一个输入，shape为[1,3,224,224]，或者原始模型的输入shape本身就是动态的，如shape为[1,3,height,width]或[1,3,-1,-1]，但部署的时候，需要该模型支持3种不同的输入shape，如[1,3,224,224], [1,3,192,192]和[1,3,160,160]，此时可以设置dynamic_input=[[[1,3,224,224]], [[1,3,192,192]], [[1,3,160,160]]]，转换成RKNN模型后进行推理时，需传入对应shape的输入数据。 注： **1. 需要原始模型本身支持动态输入才可开启此功能，否则会报错。** **2. 如果原始模型输入shape本身就是动态的，则只有动态的轴可以设置不同的值。**

API	config
	remove_reshape：删除模型的输入和输出中可能存在的Reshape的OP，以提高模型运行时性能（因为目前很多平台的Reshape是跑在cpu上，相对较慢）。 默认为 False。 注：**开启后可能会修改模型的输入或输出节点的shape，需要留意观察转换过程中的warning打印，并在部署时也需要考虑输入和输出shape变化的影响。**
	core_num：模型需要用到的NPU核心数。RK1820的NPU核心数范围为1到8，默认值为0，表示自动。 注： 1. Transformer类模型可以根据需要设置使用的NPU核心数，其他模型（如CNN模型）建议设为1。 2. 自动模式下，在使用load_llm加载的模型会使用8个NPU核心，其余情况使用1个NPU核心。
	rknn_build_parallelism：编译 rknn 模型的并行线程数，默认为 -1，表示自动配置，自动配置时不会使用超过32线程。
	distribute_strategy：模型拆分策略。目前支持'less_subgraph', 'best_perf', 'less_mem'，默认值为'less_mem'。
	input_attrs： 模型部署时的输入属性，默认值为{}。格式为：{input_name: {attr_name: attr_value, ...}, ...}, 例如：{'input_name': {'dtype': 'float32', 'layout': 'NCHW', 'shape': [1, 3, 224, 224]}, ...} 其中attr_name的可选项为dtype, layout, shape, stride等。
	output_attrs： 模型部署时的输出属性，默认值为{}。格式为：{output_name: {attr_name: attr_value, ...}, ...}, 例如：{'output_name': {'dtype': 'float32', 'layout': 'NCHW', 'shape': [1, 1000]}, ...} 其中attr_name的可选项为dtype, layout, shape, stride等。
	kvcache_store_method：指定 Kvcache 保存模式。配置为 “GroupQuant” 表示对 kvcache 进行量化处理节省内存。配置为 “Normal” 时表示 kvcache 维持 fp16 的保存格式。默认为 “Normal”。
	kvcache_dtype： 指定 Kvcache 保存及计算的数据类型。在 “GroupQuant” 模式下，支持 “**Int4_to_F16**”，表示保存为 int4 数据，计算过程使用 fp16 计算；在 “Normal” 模式下，支持 "**Float16**"，表示保存和计算均使用 fp16。默认为 “Float16”
	kvcache_group_size：指定 Kvcache 保存的分组数。当 Kvcache_dtype 为 Int4_to_F16 时，kvcache_group_size 需配置为 16。
	kvcache_residual_depth：指定 Kvcache 的缓存区大小，当缓存区填满时，会按照 kvcache_dtype 指定类型存入 kvcache 保存区域。当 kvcache_dtype 为 “Float16” 时，要求 kvcache_residual_depth 为 32；当 kvcache_dtype 为 “Int4_to_F16” 时，要求 kvcache_residual_depth 为 64。默认为 32。
	max_ctx_len：指定 Kvcache 最大长度，影响模型长文本下的推理结果以及 kvcache 的内存占用。推理时，如果超过 kvcache 长度，则会发生 kvcache 覆写行为，覆写行为遵循先进先出原则。目前由于内存规划的原因，只支持在转模型阶段指定 Kvcache 长度。默认为 1024。
	max_position_embeddings：指定最大 position id 大小，影响 rope 最大编码位置。需留意该参数不能大于原模型训练时配置的最大 rope 编码，否则可能引入异常行为。默认为 8192。

API	config
	subgraph_dtype_target：将模型中指定片段的推理精度类型配置为目标类型。该参数目前仅在模型开启量化时生效，且目前只支持指定对应片段使用 w16a16（等同于 Float16）进行推理。 参数为两级list，例如 [[subgraph0_inputs, subgraph0_outputs, dtype], [subgraph1_inputs, subgraph1_outputs, dtype], ...]，其中subgraph_inputs 为包含片段一个或多个输入 tensor 名称的 list，subgraph_outputs 为包含片段一个或多个输出 tensor 名称的 list，tensor 名称可从精度分析功能生成的表格中获取。dtype 表示推理精度类型。 注1: 特别的，当 subgraph_inputs、subgraph_outputs 为某个op的输入输出，此时相当于指定该 op 使用分配的推理精度类型。 注2: 目前不强制要求 subgraph_inputs、subgraph_outputs 必须包含对应片段的所有输入输出，此时片段中的部分 op 会使用对应的类型，判断逻辑为 op 的输入与 subgraph_inputs 有推理关联、op 的输出与 subgraph_outputs 有推理关联，此时会将该 op 配置为指定的推理精度类型。
	llm_config：配置 LLM 相关优化项，配置格式为字典，详细配置说明见2.2.1小章节。配置后需要调用 rknn.load_llm 接口加载模型才会生效。
返回值	无。

举例如下：

```
# model config
rknn.config(mean_values=[[103.94, 116.78, 123.68]],
            std_values=[[58.82, 58.82, 58.82]],
            target_platform='rk1820')
```

2.2.1 llm_config 配置项

针对 LLM 模型，`rknn.config`新增了 `llm_config`参数，LLM config 主要有4个字典子项，分别为`attention_config`, `llm_head`, `vocab`, `keep_one_logit`。

---

attention_config：用于配置 KV Cache 缓存机制、配置 Attention op 的推理机制，接受类型为列表，列表中允许存在多个字典指定多种 Attention 结构的具体参数。Attention_config 的子字典定义如下：

配置	功能	数据类型:默认值	备注
mask_name	指定 mask 名称，在后续流程中 Attention op 会通过识别 mask 输入来确定该 Attention op 所属的相关配置。 **该参数不可缺省！**	String: 'attention_mask'	不用类型的 Attention 结构不允许共享同一个 mask 输入

配置	功能	数据类型:默认值	备注
kvcache_buffer_len	指定 KV Cache 缓存长度。推理时若上下文长度超过设定长度时，遵循先进先出的原则，进行循环覆写。配置数值越大时，KV Cache占用内存越大，长文本任务的表现理论上会更好。	Int:1024	数值要求32对齐
kvcache_dtype	指定上下文保存及计算数据类型，影响推理性能及缓存内存占用大小。 内存占用上, Float16 > Int8_to_FP16 > Int4_to_FP16	String: 'Float16'	支持 Float16, Int8_to_FP16, Int4_to_FP16
attention_type	指定 Attention 类型	String: 'FullAttention'	支持 FullAttention, SlidingAttention. (当前版本暂不支持 SlidingAttention)
sliding_window_size	指定 Sliding Attention 的滑窗长度。当 attention 类型为 FullAttention 时，此参数不生效	Int: -1	/
position_name	指定 position id 输入的名字，用于提取位置编码特征值。对于早期的模型，可能不存在位置编码，此时允许缺省，配置为 None.	String: 'position_ids'	不同类型的 Attention 结构不允许共享同一个 position_ids 输入
max_position_embeddings	指定位置编码的最大长度。推理时若上下文长度超过设定长度时，会抛出错误。	Int: 8192	数值要求32对齐
mrope_type	当 LLM 模型为特殊的 mrope 类型时，可以配置并开启 mrope 功能。目前支持 ['Qwen2.5-VL', 'Qwen3-VL'] 类型。**注意**，开启 mrope 时必须配置 position_name，且这个 position_name 是单维的输入，开启 mrope 功能后 toolkit 会自动生成一个 [seq_len, 3] 维度的输入	Strings: None	/

配置	功能	数据类型:默认值	备注
mrope_section	配置 mrope section 信息。对于 Qwen2.5-VL、Qwen3-VL 模型，可在模型的配置 config.json 中找到对应的配置值。	list: None	/
mrope_new_id_name	配置 mrope 新生成的 position 名称。对于当前支持的 ['Qwen2.5-VL', 'Qwen3-VL']，会生成维度为 [seq_len, 3] 的新输入。	Strings: None	/

llm_head：用于配置并开启 LLM 模型 Head 层的优化行为。（Head层通常为单一一个参数量较大的Linear层）。暂不支持多个 LLM Head 层，后续会完善并支持。

配置名	功能	数据类型:默认值	备注
name	配置 llm head 层的权重名称	String: auto	目前仅支持 auto，暂不支持指定
device	配置 llm head 层的计算设备	String: rk1820	支持 rk1820, rk3588, rk3576
quantized_dtype	配置 llm head 层的量化类型。llm head 层规模较大，采用量化可以有效降低内存占用大小	String: w6a16	支持 w16a16, w4a16, w6a16, w8a8. 当平台为 rk3576 或 rk3588 时，会被强制更改为 w8a8.
quantized_method	配置 llm head 层的量化方式。	String: group32	支持 layer, channel, group{Size}. 当平台为 rk3576 或 rk3588 时，会被强制更改为 channel.

vocab：用于配置并开启 LLM 模型对字典层的优化行为。（**当前暂未实装该参数对应的功能**）

配置名	功能	数据类型:默认值	备注
index_name	配置 vocab 层的 index 输入名称	String: auto	目前仅支持 auto，暂不支持指定
tie_embedding	配置 vocab 层的 tie embedding 参数，tie_embedding 为 True 时意味着 vocab 和 llm_head 共享同一个数组。开启后则不再需要额外保存 vocab，但由于 llm_head 存在量化行为，反量化为 float16 的 embedding 时会降低推理性能。	Bool: False	当前版本暂不支持
store_on_compute_device	配置 vocab 是否保存在计算设备上。vocab 保存在计算设备上时可以减少每轮 LLM 推理所需要的传输耗时、但会大幅增加计算设备的内存占用。	Bool: False	当前版本暂不支持

keep_one_logit：用于配置并开启对输出的指定轴进行取单值行为（插入gather op）。对于大多数 LLM 模型，在输出采样时只需要对最后一个有效token输出进行采样即可。以输入token 序列长度为 100 为例，则输出的第100个序列为有效输出，其他输出均被舍弃。故可以将这个舍弃行为提前到 llm_head 前面，减少 llm head 线性 Linear 层的计算量、减少计算过程中所需要的内存占用。

配置名	功能	数据类型:默认值	备注
output_name	指定输出名称	String: None	为None 时则不生效
idx_name	生成 gather indices 输入的名称，并加入到模型的输入	String: num_logit_to_keep	/
axis	指定 gather 生效轴	Int: None	/

举例如下：

```
from rknn.api import RKNN, DEFAULT_RKNN_LLM_CONFIG
rknn = RKNN(verbose=True)
my_config = DEFAULT_RKNN_LLM_CONFIG.copy()
rknn.config(target_platform='rk1820', llm_config=my_config)

# default config is
DEFAULT_RKNN_LLM_CONFIG = {
'attention_config': [{                                  # accept multi internal kvcache
as a list of dicts
```

```
    'mask_name'              : 'attention_mask',            # required, this is used for
recognize which attn belong to this config
    'kvcache_buffer_len'     : 1024,                        # feed int
    'kvcache_dtype'          : 'Float16',                   # Float16, Int8_to_F16,
Int4_to_F16
    'attention_type'         : 'FullAttention',             # FullAttention,
SlidingAttention
    'sliding_window_size'    : -1,                          # feed int, -1 mean disable.
    'position_name'          : 'position_ids',              # allow None if no position_ids
input, positon id here was used for extract rope.
    'max_position_embeddings': 8192,
    'mrope_type'             : None,                        # 'Qwen2.5-VL', 'Qwen3-VL'
    'mrope_section'          : None,                        # [24,20,20]
    'mrope_new_id_name'      : None,                        # new name to insert position id
input
}],
'llm_head':[{
    'name'                  : 'auto',                       # auto or specify head tensor
name, if multiple heads in model, please specify head tensor name
    'device'                : 'RK1820',                     # RK3588, RK3576,
    'quantized_dtype'       : 'w6a16',                      # w16a16, w4a16, w6a16, w8a8
    'quantized_method'      : 'group32',                    # layer, channel, group{SIZE}
}],
'vocab':[{
    'index_name'            : 'auto',                       # auto or specify embedding
tensor name, if multiple embeddings in model, please specify embedding tensor name
    'tie_embeddings'         : False,                       #! True, False, current not
support
    'store_on_compute_device': False,                       # True, False
}],
'keep_one_logit': [{                                        # accept multi internal kvcache
as list(dict)
    'output_name': None,
    'idx_name'   : 'num_logits_to_keep',
    'axis'       : None,
}],
}
```

2.3 模型加载

RKNN3-Toolkit目前支持Caffe、TensorFlow、TensorFlow Lite、ONNX、DarkNet、PyTorch等模型的加载转换，这些模型在加载时需调用对应的接口，以下为这些接口的详细说明。

2.3.1 Caffe模型加载接口

API	load_caffe
描述	加载Caffe模型。
参数	**model**：Caffe模型文件（.prototxt后缀文件）路径。
	blobs：Caffe模型的二进制数据文件（.caffemodel后缀文件）路径。

API	load_caffe
	input_name：Caffe模型存在多输入时，可以通过该参数指定输入层名的顺序，形如['input1','input2','input3']，注意名字需要与模型输入名一致；默认值为None，表示按Caffe模型文件（.prototxt后缀文件）自动给定。
返回值	0：导入成功。
	-1：导入失败。

举例如下：

```
# 从当前路径加载mobilenet_v2模型
ret = rknn.load_caffe(model='./mobilenet_v2.prototxt',
                      blobs='./mobilenet_v2.caffemodel')
```

2.3.2 TensorFlow模型加载接口

API	load_tensorflow
描述	加载TensorFlow模型。
参数	**tf_pb**：TensorFlow模型文件（.pb后缀）路径。
	inputs：模型的输入节点（tensor名），支持多个输入节点。所有输入节点名放在一个列表中。
	input_size_list：每个输入节点对应的shape，所有输入shape放在一个列表中。如示例中的ssd_mobilenet_v1模型，其输入节点对应的输入shape是[[1, 300, 300, 3]]。
	outputs：模型的输出节点（tensor名），支持多个输出节点。所有输出节点名放在一个列表中。
	input_is_nchw：模型的输入的layout是否已经是NCHW。默认值为False，表示默认输入layout为NHWC。
返回值	0：导入成功。
	-1：导入失败。

举例如下：

```
# 从当前目录加载ssd_mobilenet_v1_coco_2017_11_17模型
ret = rknn.load_tensorflow(tf_pb='./ssd_mobilenet_v1_coco_2017_11_17.pb',
                           inputs=['Preprocessor/sub'],
                           outputs=['concat', 'concat_1'],
                           input_size_list=[[300, 300, 3]])
```

2.3.3 TensorFlow Lite模型加载接口

API	load_tflite
描述	加载TensorFlow Lite模型。
参数	**model**：TensorFlow Lite模型文件（.tflite后缀）路径。
	input_is_nchw：模型的输入的layout是否已经是NCHW。默认值为False，即默认输入layout为NHWC。
返回值	0：导入成功。
	-1：导入失败。

举例如下：

```
# 从当前目录加载mobilenet_v1模型
ret = rknn.load_tflite(model='./mobilenet_v1.tflite')
```

2.3.4 ONNX模型加载

API	load_onnx
描述	加载ONNX模型。
参数	**model**：ONNX模型文件（.onnx后缀）路径。
	inputs：模型输入节点（tensor名），支持多个输入节点，所有输入节点名放在一个列表中。默认值为None，表示从模型里获取。
	input_size_list：每个输入节点对应的shape，所有输入shape放在一个列表中。如inputs有设置，则input_size_list也需要被设置。默认值为None。
	input_initial_val：设置模型输入的初始值，格式为ndarray的列表。默认值为None。 主要用于将某些输入固化为常量，对于不需要固化为常量的输入可以设为None，如[None, np.array([1])]。
	outputs：模型的输出节点（tensor名），支持多个输出节点，所有输出节点名放在一个列表中。默认值为None，表示从模型里获取。
返回值	0：导入成功。
	-1：导入失败。

举例如下：

```
# 从当前目录加载arcface模型
ret = rknn.load_onnx(model='./arcface.onnx')
```

2.3.5 DarkNet模型加载接口

API	load_darknet
描述	加载DarkNet模型。
参数	**model**：DarkNet模型文件（.cfg后缀）路径。
	weight：权重文件（.weights后缀）路径。
返回值	0：导入成功。
	-1：导入失败。

举例如下：

```
# 从当前目录加载yolov3-tiny模型
ret = rknn.load_darknet(model='./yolov3-tiny.cfg',
                        weight='./yolov3.weights')
```

2.3.6 PyTorch模型加载接口

API	load_pytorch
描述	加载PyTorch模型。 支持量化感知训练（QAT）模型，但需要将torch版本更新至1.9.0以上。
参数	**model**：PyTorch模型文件（.pt后缀）路径，而且需要是torchscript格式的模型。
	input_size_list：每个输入节点对应的shape，所有输入shape放在一个列表中。
返回值	0：导入成功。
	-1：导入失败。

举例如下：

```
# 从当前目录加载resnet18模型
ret = rknn.load_pytorch(model='./resnet18.pt',
                        input_size_list=[[1,3,224,224]])
```

2.3.7 LLM模型加载接口

API	load_llm
描述	加载LLM模型。 LLM模型参考rkllm3_model_zoo工程的示例导出。（例如 rkllm3_model_zoo/examples/Qwen2_5/python/export_llm.py）
参数	**model**：LLM模型文件（.onnx后缀）路径。
	config：LLM模型的配置文件路径。
	seq_lens：LLM模型输入sequence length列表，默认为[1,128]。

API	load_llm
	llm_head_dtype：LLM模型Head权重的数据类型，目前支持的数据类型有int4、int6、int8、float16。
	llm_head_quantized_method：LLM模型Head权重的量化方法，目前支持group{SIZE}。{SIZE}是32到256之间的32的倍数值，如group32或group128。当**llm_head_dtype**为float16时，忽略该参数。
返回值	0：导入成功。
	-1：导入失败。

举例如下：

```
# 从当前目录加载Qwen2.5-0.5B模型
ret = rknn.load_llm(model='./Qwen2.5-0.5B.onnx',
                    config='./Qwen2.5-0.5B.pkl')
```

2.4 构建RKNN模型

API	build
描述	构建RKNN模型。
参数	**do_quantization**：是否对模型进行量化。默认值为True。
	dataset：用于量化校正的数据集。目前支持文本文件格式，用户可以把用于校正的图片（jpg或png格式）或npy文件路径放到一个.txt文件中。文本文件里每一行一条路径信息。如： a.jpg b.jpg 或 a.npy b.npy 如有多个输入，则每个输入对应的文件用空格隔开，如： a.jpg a2.jpg b.jpg b2.jpg 或 a.npy a2.npy b.npy b2.npy 注：量化图片建议选择与预测场景较吻合的图片。
返回值	0：构建成功。
	-1：构建失败。

举例如下：

```
# 构建RKNN模型，并且做量化
ret = rknn.build(do_quantization=True, dataset='./dataset.txt')
```

2.5 导出RKNN模型

通过本工具构建的RKNN模型通过该接口可以导出存储为RKNN模型文件，用于模型部署。

API	export_rknn
描述	将RKNN模型保存到指定文件中（.rknn后缀）且将模型权重保存在同一文件路径中（.weight后缀）。 如save_ctx为True，则也会保存上下文信息（.rknn.ctx后缀）
参数	**export_path**：导出模型文件的路径。
	save_ctx：是否导出上下文信息（配合load_rknn的load_ctx参数使用）。默认值为False。
返回值	0：导出成功。
	-1：导出失败。

举例如下：

```
# 将构建好的RKNN模型保存到当前路径的mobilenet_v1.rknn文件中，模型权重保存在当前路径的
mobilenet_v1.weight文件中
ret = rknn.export_rknn(export_path='./mobilenet_v1.rknn')
```

2.6 加载RKNN模型

API	load_rknn
描述	加载RKNN模型。 加载完RKNN模型后，不需要再进行模型配置、模型加载和构建RKNN模型的步骤。 当load_ctx为False时，加载后的模型仅限于连接NPU硬件进行推理或获取性能数据等，不能用于模拟器或精度分析等。 当load_ctx为True时，加载后的模型不仅可连接NPU硬件进行推理或获取性能数据等，也可用于模拟器或精度分析等。
参数	**model_path**：RKNN模型文件路径。
	weight_path：RKNN模型权重文件路径。
	load_ctx：是否加载上下文信息。默认值为False。
返回值	0：加载成功。
	-1：加载失败。

举例如下：

```
# 从当前路径加载mobilenet_v1.rknn模型
ret = rknn.load_rknn(model_path='./mobilenet_v1.rknn', weight_path='./mobilenet_v1.weight')
```

2.7 初始化运行时环境

在模型推理或性能评估之前，必须先初始化运行时环境，明确模型的运行平台（具体的目标硬件平台或软件模拟器）。

API	init_runtime
描述	初始化运行时环境。
参数	**target**：目标硬件平台，默认为None，即模型在模拟器上运行; 当指定平台后为连板运行。
	device_id：设备编号，当PC连接多台设备时，需通过该参数指定目标设备编号，设备ID必须为"rk3576" 或 "rk3588"。可通过调用 list_devices 接口查询可用设备列表。默认值为None。
	core_mask：模型连板推理的运行核心，如果连板推理时，需要指定该参数，支持从0x1到0xff的掩码值。默认值为-1。
返回值	0：初始化运行时环境成功。
	-1：初始化运行时环境失败。

注：**当前仅支持 PC 通过 ADB 模式连接设备。该设备的 SoC 必须为 RK3576 或 RK3588；且该设备需与目标板 RK1820/RK1828 通过 PCIe 或 USB 互联。**

举例如下：

```
# 初始化运行时环境
ret = rknn.init_runtime(target=None)
```

2.8 模型推理和精度分析

在进行模型推理前，必须先构建或加载一个RKNN模型。

API	inference
描述	对当前模型进行推理，并返回推理结果。
参数	**inputs**：待推理的输入列表，格式为ndarray。
	data_format：输入数据的layout列表，“nchw”或“nhwc”，只对4维的输入有效。默认值为None，表示所有输入的layout都为NHWC。
	accuracy_analysis：是否开启逐层精度分析。默认值为False。当init_runtime中设置target参数则进行连板精度分析，连扳精度分析时在`config`中需配置`profile_mode=True`
	output_dir：精度分析文件的输出目录，所有快照都保存在该目录。默认值为'./snapshot'。 如果init_runtime中没有设置target，在output_dir下会输出： - simulator目录：保存整个量化模型在simulator上完整运行时每一层的结果（已转成float32）； - golden目录：保存整个浮点模型在simulator上完整跑下来时每一层的结果; - error_analysis.txt：记录simulator上量化模型逐层运行时每一层的结果与golden浮点模型逐层运行时每一层的结果的余弦距离（entire_error cosine），以及量化模型取上一层的浮点结果作为输入时，输出与浮点模型的余弦距离（single_error cosine），更详细的信息请查看error_analysis.txt文件。 如果init_runtime中有设置target，则在output_dir里还会多输出： - runtime目录：保存整个量化模型在NPU上完整运行时每一层的结果（已转成float32）。 - error_analysis.txt：在上述记录的内容的基础上，还会记录量化模型在simulator上逐层运行时每一层的结果与NPU上逐层运行时每一层的结果的余弦距离（entire_error cosine）等信息，更详细的信息请查看error_analysis.txt文件。
返回值	results：推理结果，类型是ndarray list。

举例如下：

对于分类模型，如mobilenet_v1，代码如下（完整代码参考examples/tflite/mobilenet_v1）：

```
# 使用模型对图片进行推理，得到TOP5结果
outputs = rknn.inference(inputs=[img])
show_outputs(outputs)
```

输出的TOP5结果如下：

```
-----TOP 5-----
[ 156] score:0.928223 class:"Shih-Tzu"
[ 155] score:0.063171 class:"Pekinese, Pekingese, Peke"
[ 205] score:0.004299 class:"Lhasa, Lhasa apso"
[ 284] score:0.003096 class:"Persian cat"
[ 285] score:0.000171 class:"Siamese cat, Siamese"
```

2.9 查询功能

API	query
描述	查询模型相关信息或状态。
参数	**cmd**：查询命令的类型，格式为字符串。
返回值	results：查询的结果。

2.9.1 查询命令的类型

命令	描述
QUERY_ROPE_CACHE	查询模型rope输入的sin/cos的cache。

举例如下：

```
# 查询模型rope输入的sin/cos的cache
rope_cache = rknn.query('QUERY_ROPE_CACHE')
```

2.10 KVCache控制器

2.10.1 获取控制kvcache的输入

API	kvcache_controller.generate_kvcache_control_tensors
描述	获取控制kvcache的输入和模型动态shape的id。
参数	**input_seq_len**： 模型输入token长度。
参数	**system_prompt_len**：系统提示词的token长度，默认值为0。
返回值	返回控制kvcache的输入的列表和模型动态shape的id，如果需要多轮推理，则返回列表长度大于1。

举例如下：

```
attention_inputs, dynamic_idx =
rknn.kvcache_controller.generate_kvcache_control_tensors(input_seq_len)
```

2.10.2 清空kvcache状态

API	kvcache_controller.clear_kvcache_status
描述	清空kvcache状态。
参数	无。
返回值	无。

举例如下：

```
rknn.kvcache_controller.clear_kvcache_status()
```

2.11 评估模型性能

API	eval_perf
描述	获取模型性能。 模型必须运行在与PC连接的指定设备上。 在`config`中需配置`profile_mode=True`
参数	**log_level**：打印性能信息等级，默认值为0。0：仅打印汇总表；1：打印每个 Op 的详细信息表和汇总表；2：打印每个 Op 的详细信息包括时间、周期和带宽及汇总表
	dynamic_idx：动态输入的索引值，仅在动态shape中有效。默认值为0。
返回值	0：运行成功。
	-1：运行失败。

举例如下：

```
# 对模型性能进行评估
=ret = rknn.eval_perf()
```

2.12 获取内存使用情况

API	eval_memory
描述	获取模型在硬件平台运行时的内存使用情况。 模型必须运行在与PC连接的指定设备上。
参数	无
返回值	0：运行成功。
	-1：运行失败。

举例如下：

```
# 对模型内存使用情况进行评估
ret = rknn.eval_memory()
```

2.13 获取设备列表

API	list_devices
描述	列出已连接的 RK3576 / RK3588 / RK1820。 注：目前设备连接模式为ADB。
参数	无。

API	list_devices
返回值	返回adb_devices列表，如果设备为空，则返回空列表。

举例如下：

```
rknn.list_devices()
```

返回的设备列表信息如下：

```
************************
all device(s) with adb mode:
VD46C3KM6N
************************
```