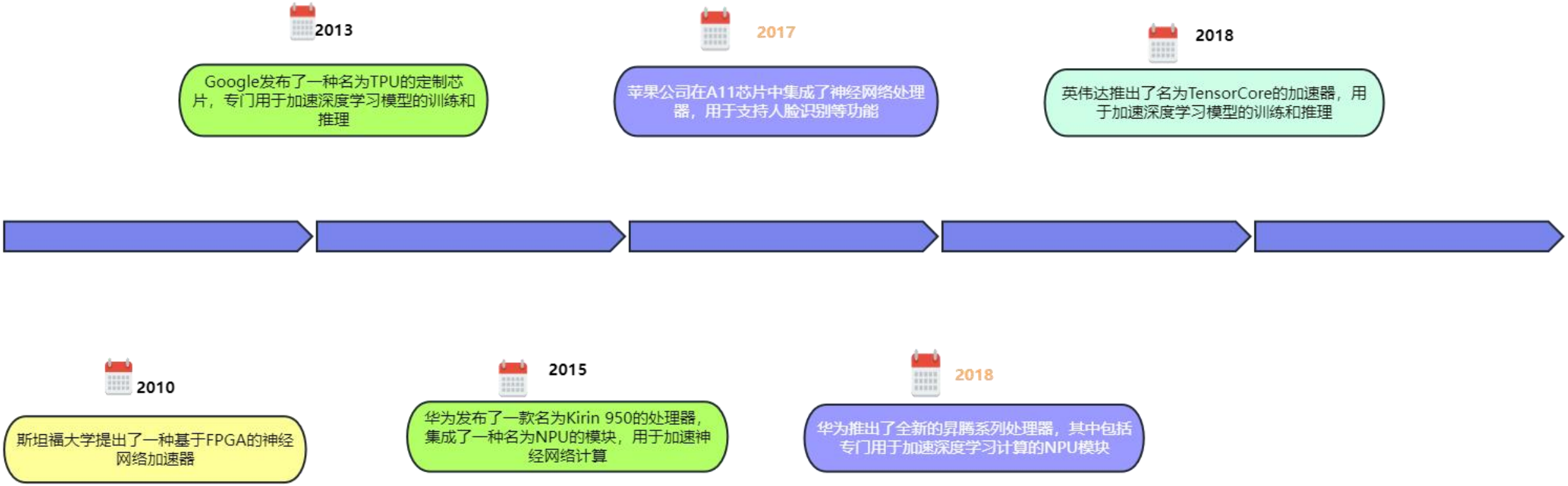
RKNPU教程

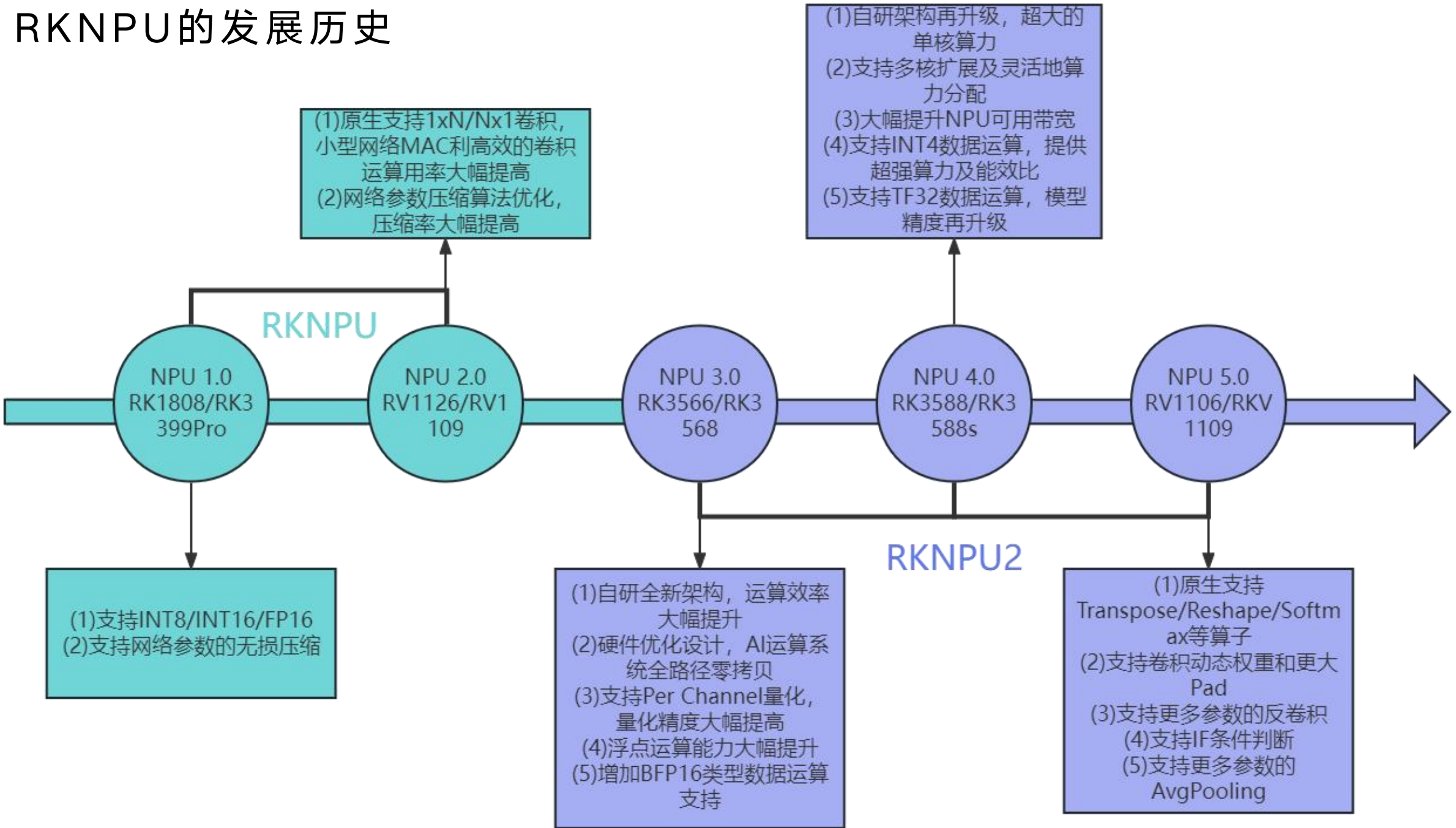
01 初识RKNPU

TOPEET 迅为

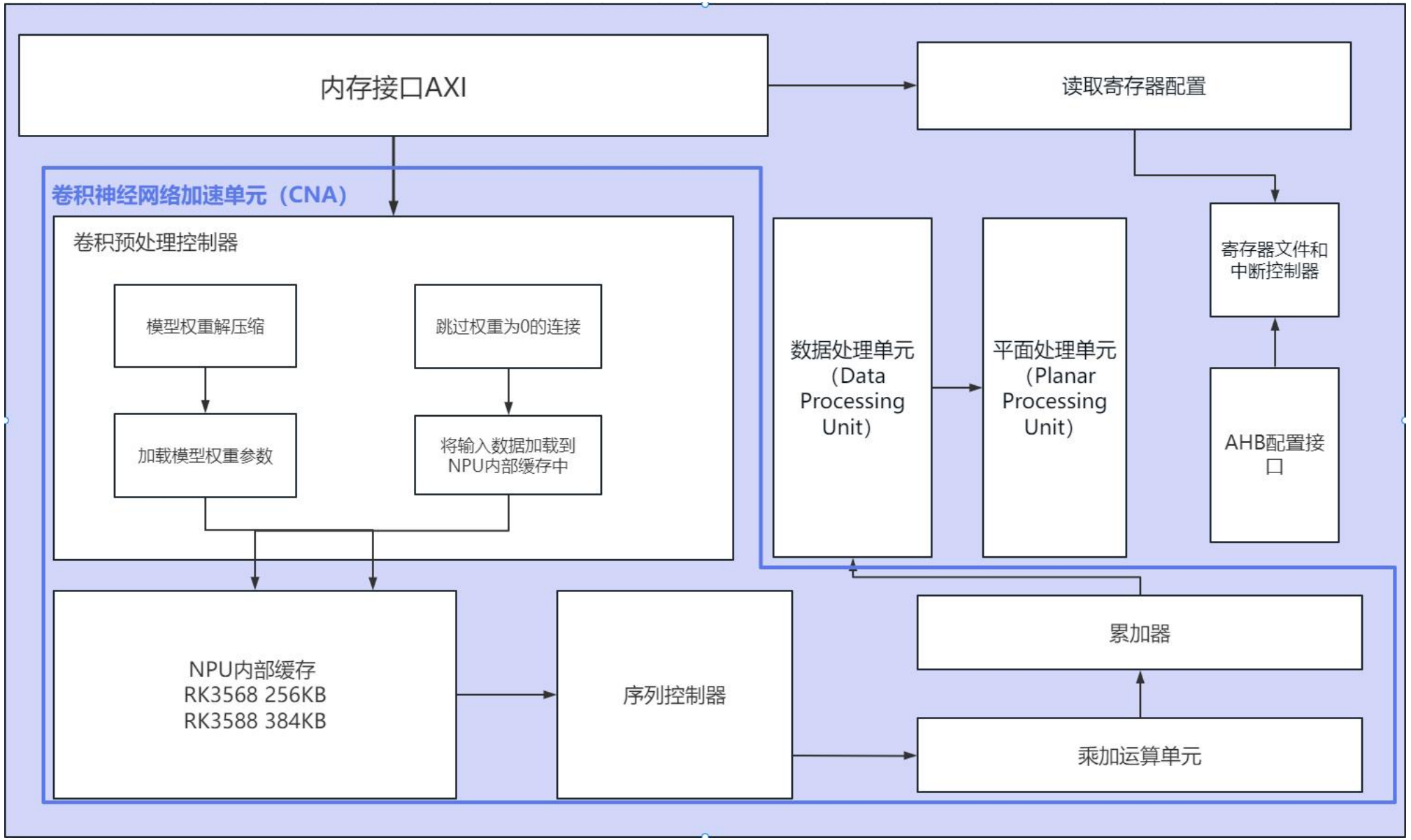
NPU的由来


2010
斯坦福大学提出了一种基于FPGA的神经网络加速器
2013
Google发布了一种名为TPU的定制芯片，专门用于加速深度学习模型的训练和推理
2015
华为发布了一款名为Kirin 950的处理器，集成了一种名为NPU的模块，用于加速神经网络计算
2017
苹果公司在A11芯片中集成了神经网络处理器，用于支持人脸识别等功能
2018
华为推出了全新的昇腾系列处理器，其中包括专门用于加速深度学习计算的NPU模块
2018
英伟达推出了名为TensorCore的加速器，用于加速深度学习模型的训练和推理

RKNPU的发展历史


(1)自研架构再升级，超大的单核算力
(2)支持多核扩展及灵活地算力分配
(3)大幅提升NPU可用带宽
(4)支持INT4数据运算，提供超强算力及能效比
(5)支持TF32数据运算，模型精度再升级
(1)原生支持1xN/Nx1卷积，小型网络MAC利高效的卷积运算用率大幅提高
(2)网络参数压缩算法优化，压缩率大幅提高
RKNPU
NPU 1.0 RK1808/RK3399Pro
NPU 2.0 RV1126/RV1109
NPU 3.0 RK3566/RK3568
NPU 4.0 RK3588/RK3588s
NPU 5.0 RV1106/RKV1109
(1)支持INT8/INT16/FP16
(2)支持网络参数的无损压缩
RKNPU2
(1)自研全新架构，运算效率大幅提升
(2)硬件优化设计，AI运算系统全路径零拷贝
(3)支持Per Channel量化，量化精度大幅提高
(4)浮点运算能力大幅提升
(5)增加BFP16类型数据运算支持
(1)原生支持Transpose/Reshape/Softmax等算子
(2)支持卷积动态权重和更大Pad
(3)支持更多参数的反卷积
(4)支持IF条件判断
(5)支持更多参数的AvgPooling

RKNPU单核架构


内存接口AXI
读取寄存器配置
卷积神经网络加速单元（CNA）
卷积预处理控制器
模型权重解压缩
跳过权重为0的连接
加载模型权重参数
将输入数据加载到NPU内部缓存中
数据处理单元（Data Processing Unit）
平面处理单元（Planar Processing Unit）
寄存器文件和中断控制器
AHB配置接口
NPU内部缓存
RK3568 256KB
RK3588 384KB
序列控制器
累加器
乘加运算单元

RKNPU性能计算

NPU算力是指每秒可以处理的运算次数，通常以*TOPS（Tera Operations Per Second）*为单位进行衡量。

每个周期的理论峰值性能为*Perf=MACs*2(ops/cycle)*。其中MACs表示每个周期内可以进行的乘加操作次数，而算力指的是运算的次数，所以要将乘加操作分解为一次乘法和一次加法，也就是最后乘以2的由来。

当NPU的频率为 *f* 时，每秒的理论峰值性能为*Perf=MACs*2*f(ops/s)*。

以RK3588 int 8 数据类型为例进行性能计算演示：

（1）RK3588每个周期可进行*1024x3*个int8 MAC操作

（2）RK3588 的NPU频率为*1G HZ*

（3）理论峰值性能计算公式为*Perf=MACs*2*f(ops/s)*

综上RK3588 int 8 理论峰值性能为Perf=1024x3x2x1G = 6 TOPS

RKNPU应用场景

计算机视觉

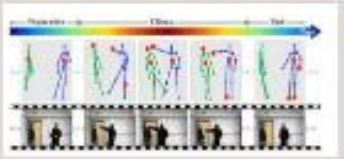
图像分类

目标检测

图像分割

人脸识别

行为识别



语音识别

语音识别

语音合成

说话人识别

自然语言处理

情感分析

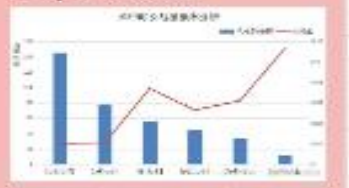
文本生成

文本分类

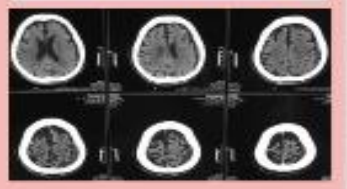
机器翻译

医疗保健

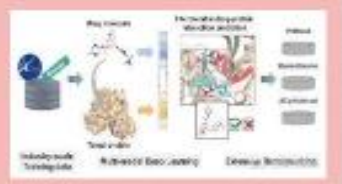
疾病预测



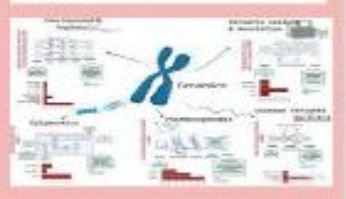
医疗影像分析



药物发现

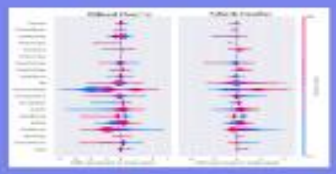


基因组学



金融服务

风险评估



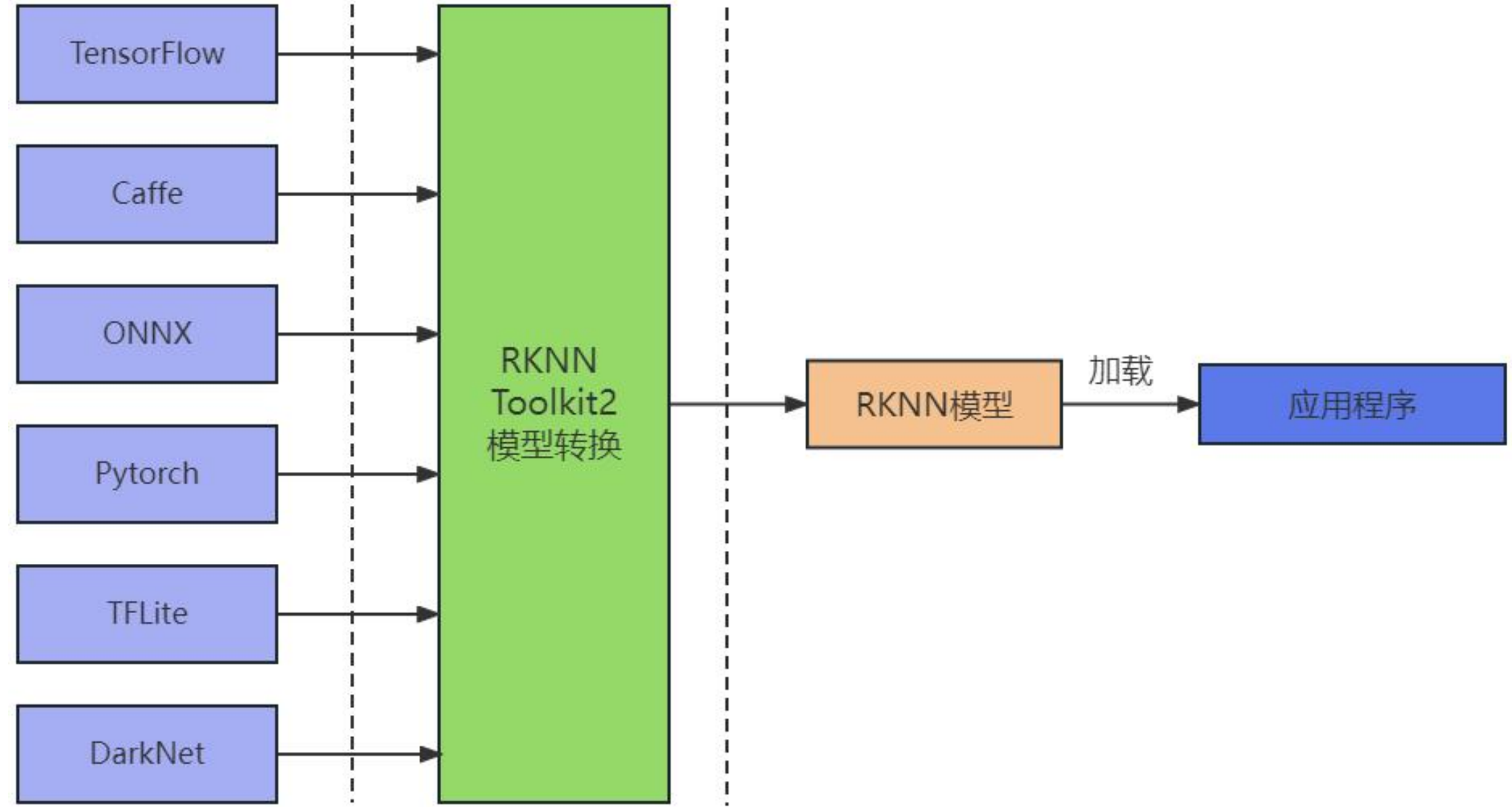
欺诈检测

股票预测

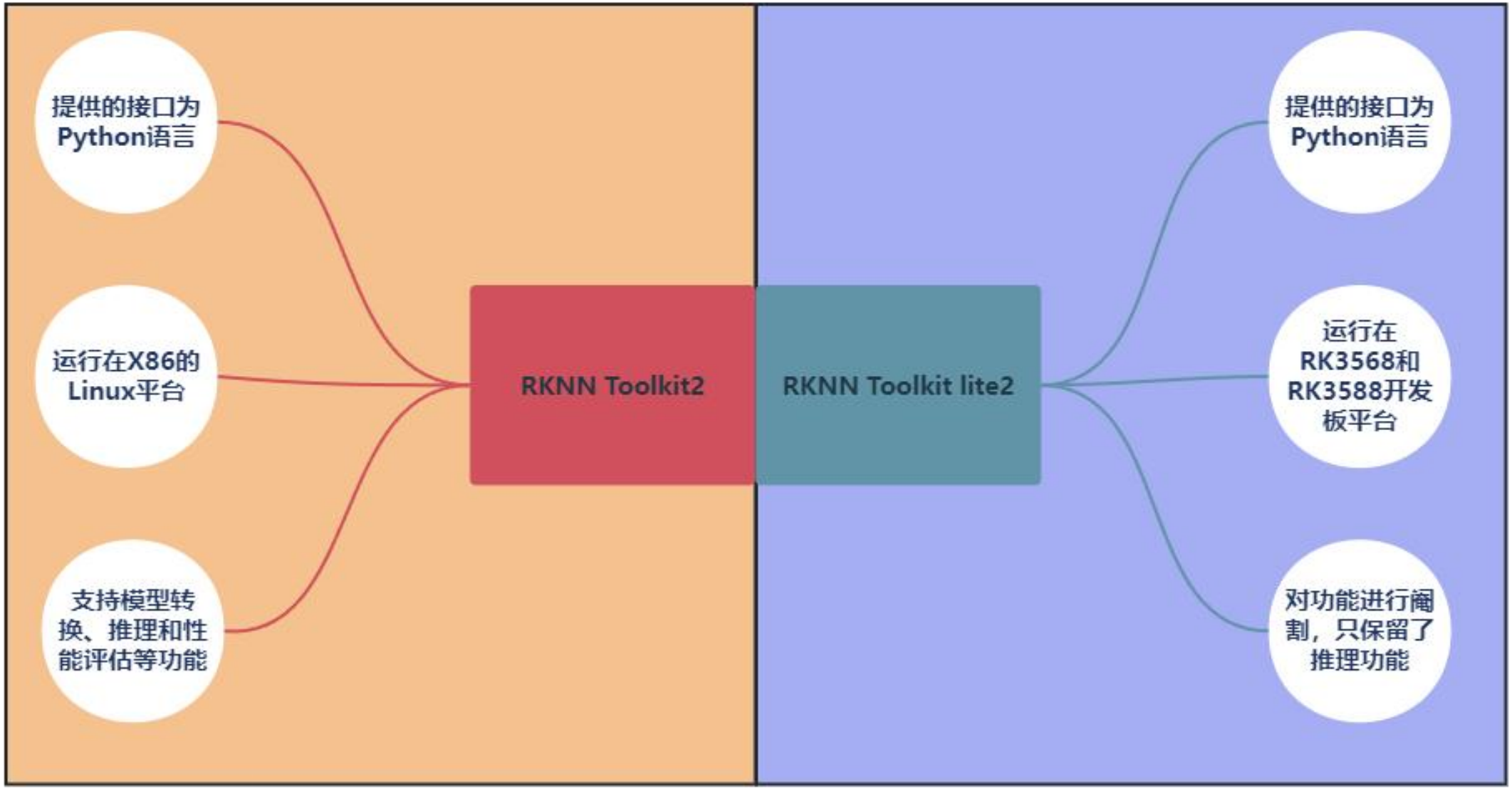
投资组合优化

RKNPU 推理软件框架

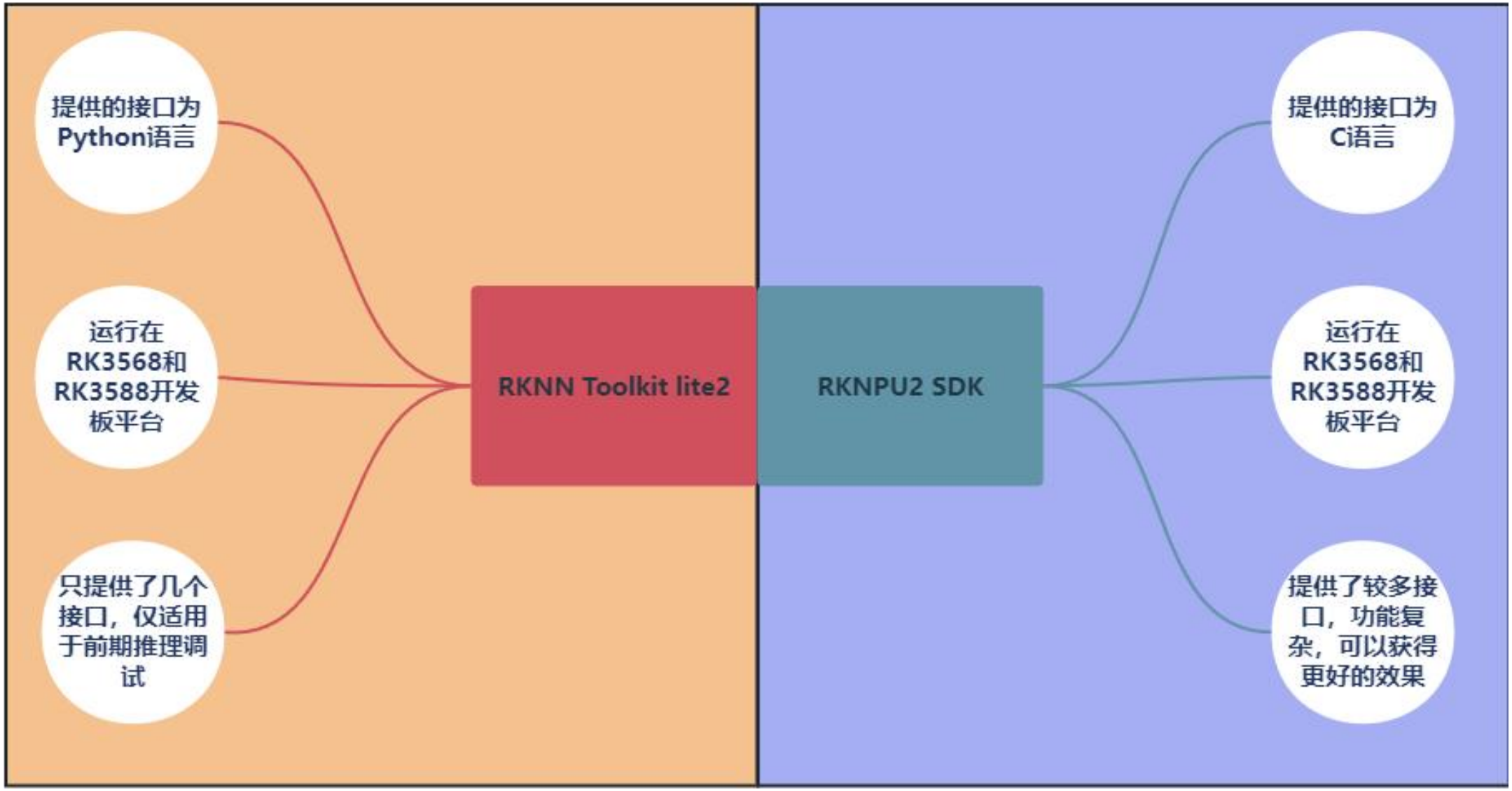
RKNN 模型

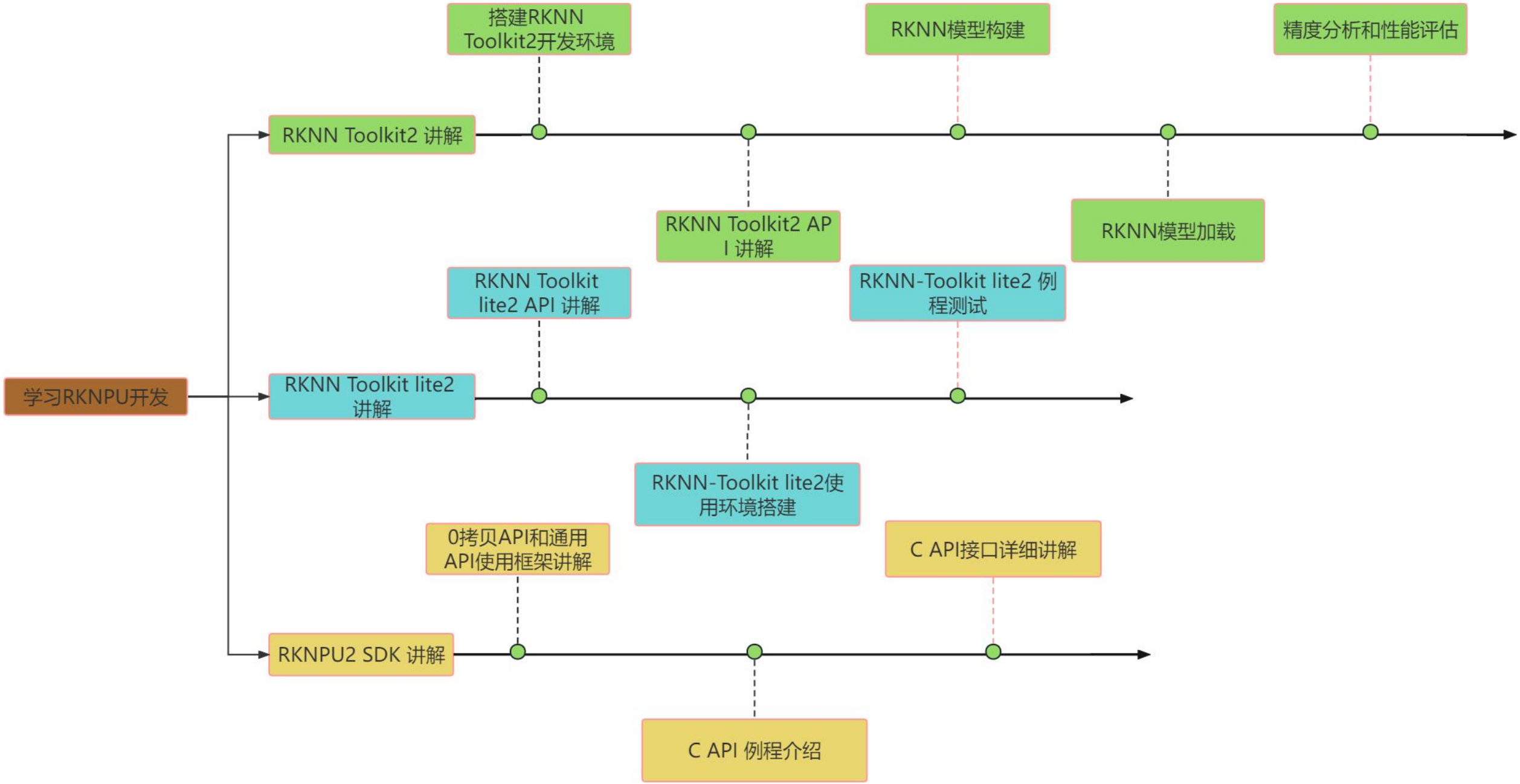

TensorFlow
Caffe
ONNX
Pytorch
TFLite
DarkNet
RKNN
Toolkit2
模型转换
RKNN模型
加载
应用程序

RKNN Toolkit 2和RKNN Toolkit lite2工具对比

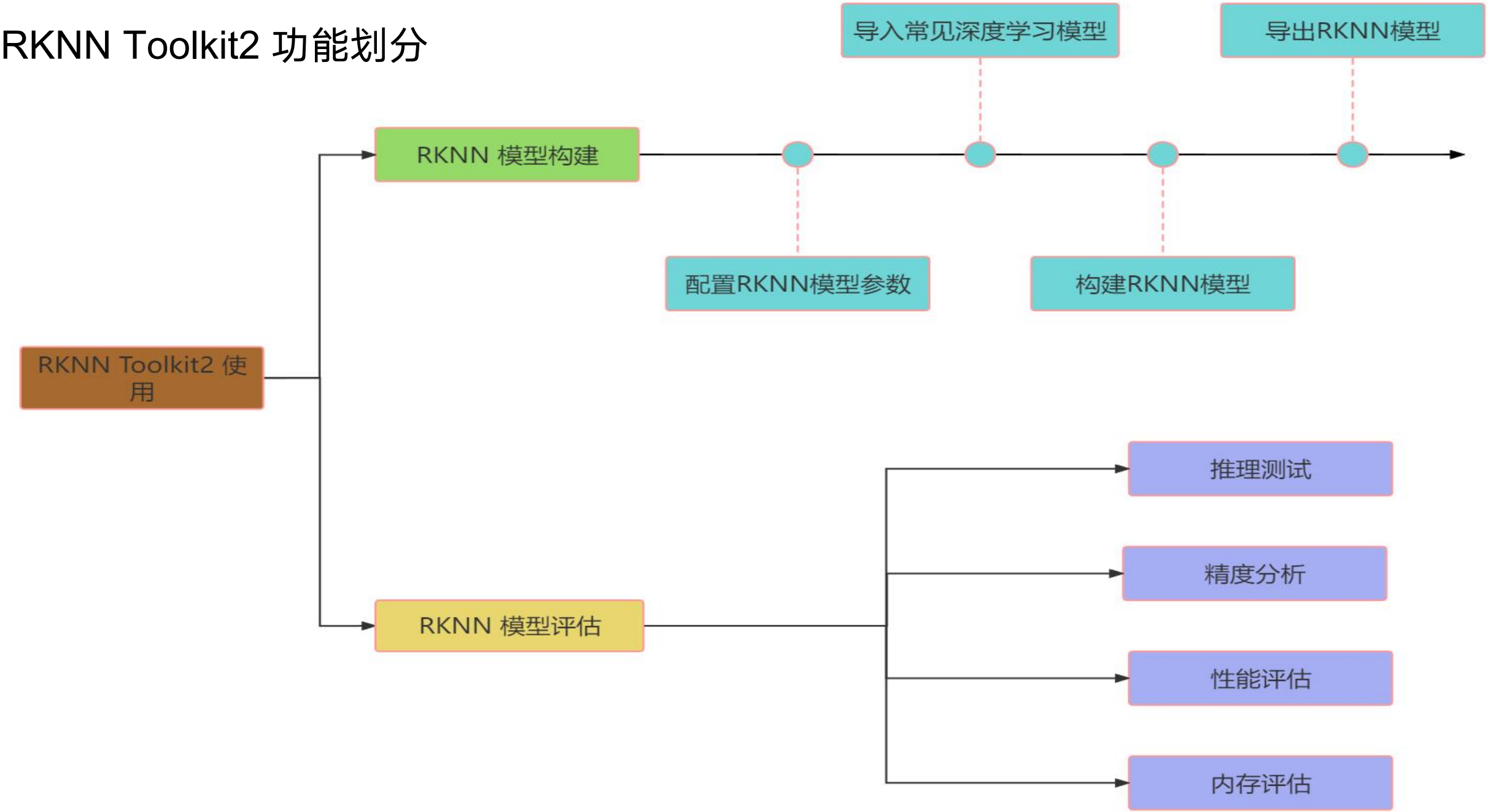


RKNN Toolkit lite2 和RKNPU2 SDK对比

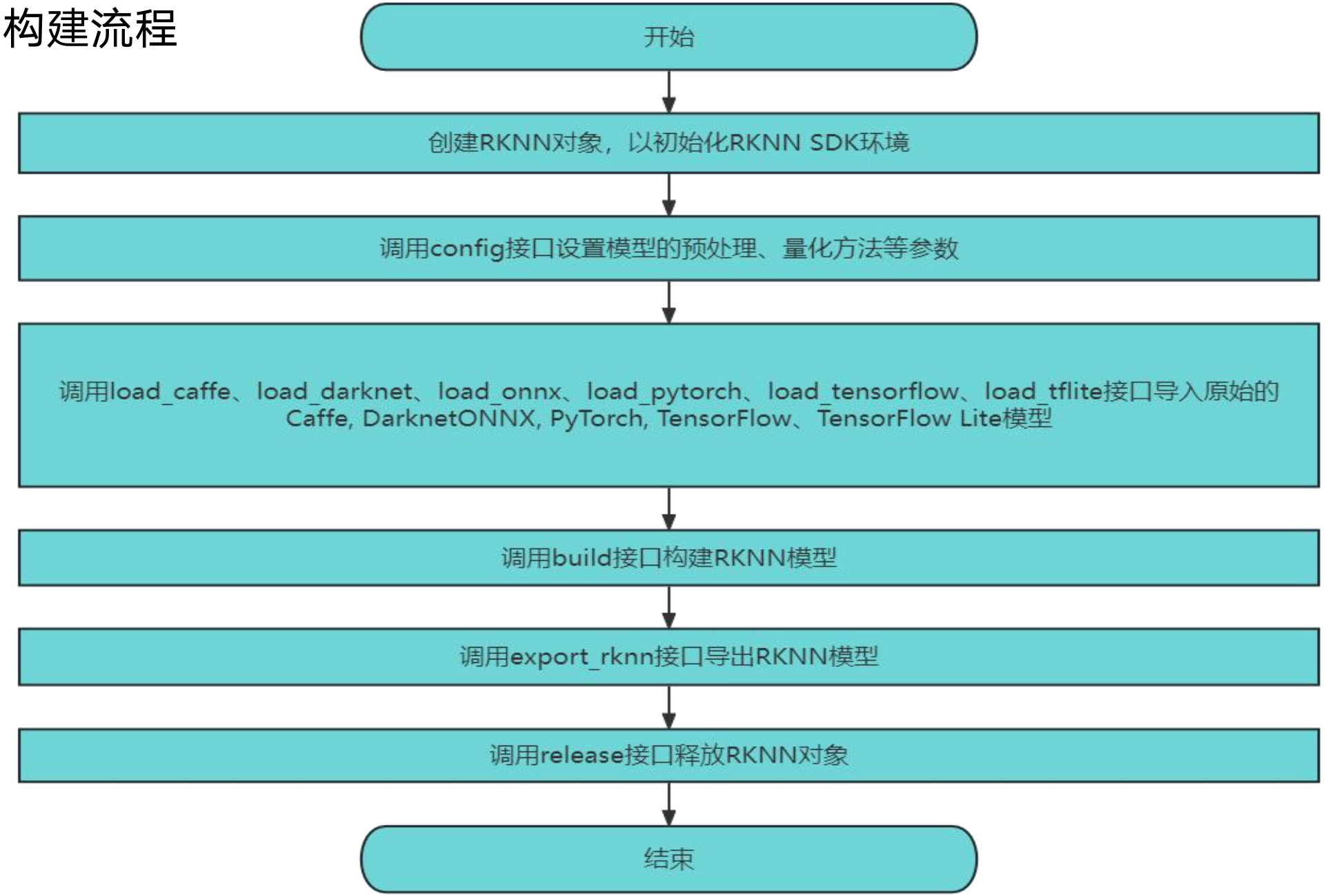



学习RKNPU开发
RKNN Toolkit2 讲解
搭建RKNN Toolkit2开发环境
RKNN Toolkit2 API 讲解
RKNN模型构建
RKNN模型加载
精度分析和性能评估
RKNN Toolkit lite2 讲解
RKNN Toolkit lite2 API 讲解
RKNN-Toolkit lite2使用环境搭建
RKNN-Toolkit lite2 例程测试
RKNPU2 SDK 讲解
0拷贝API和通用API使用框架讲解
C API 例程介绍
C API接口详细讲解

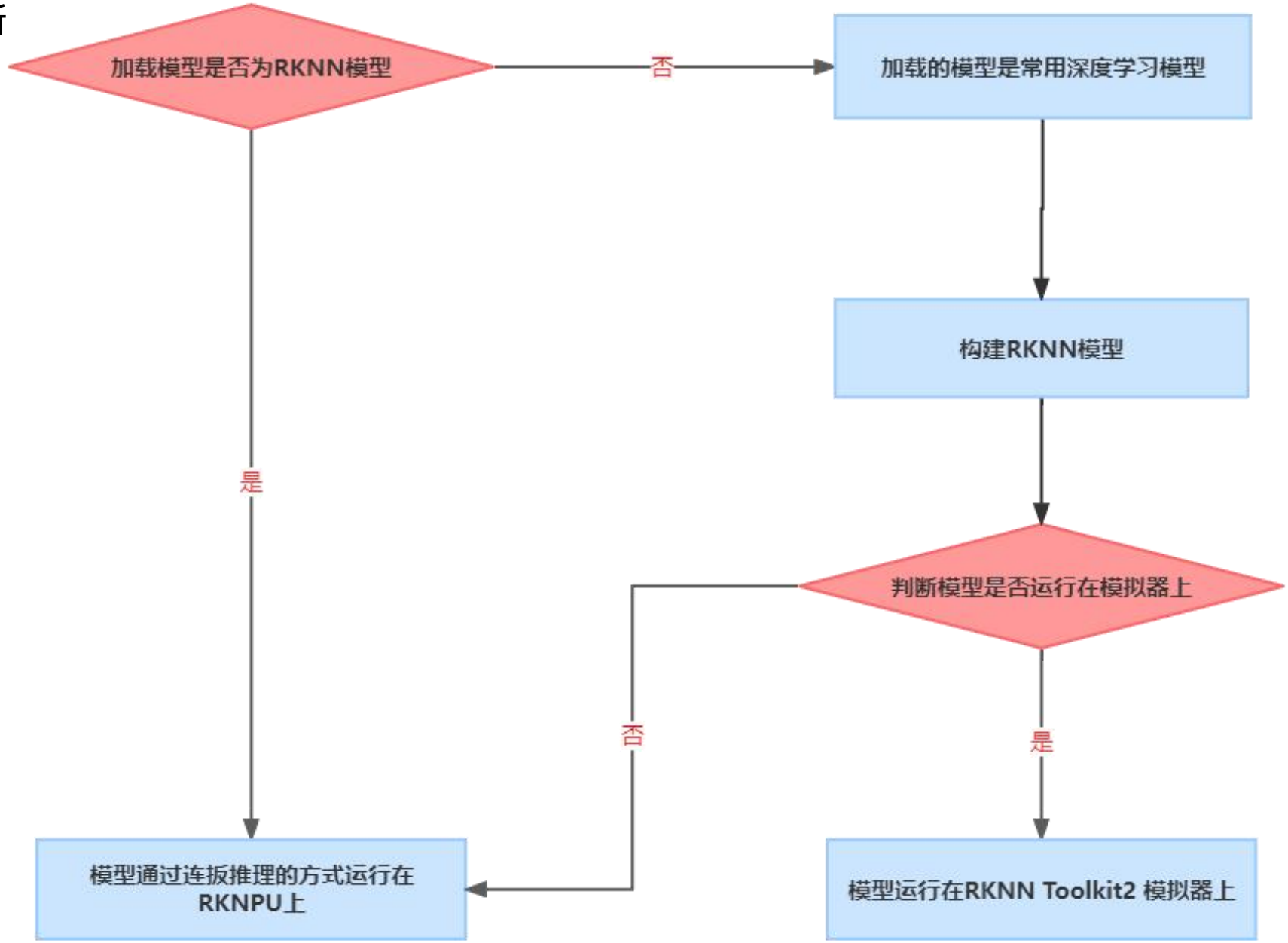
RKNN Toolkit2 功能划分



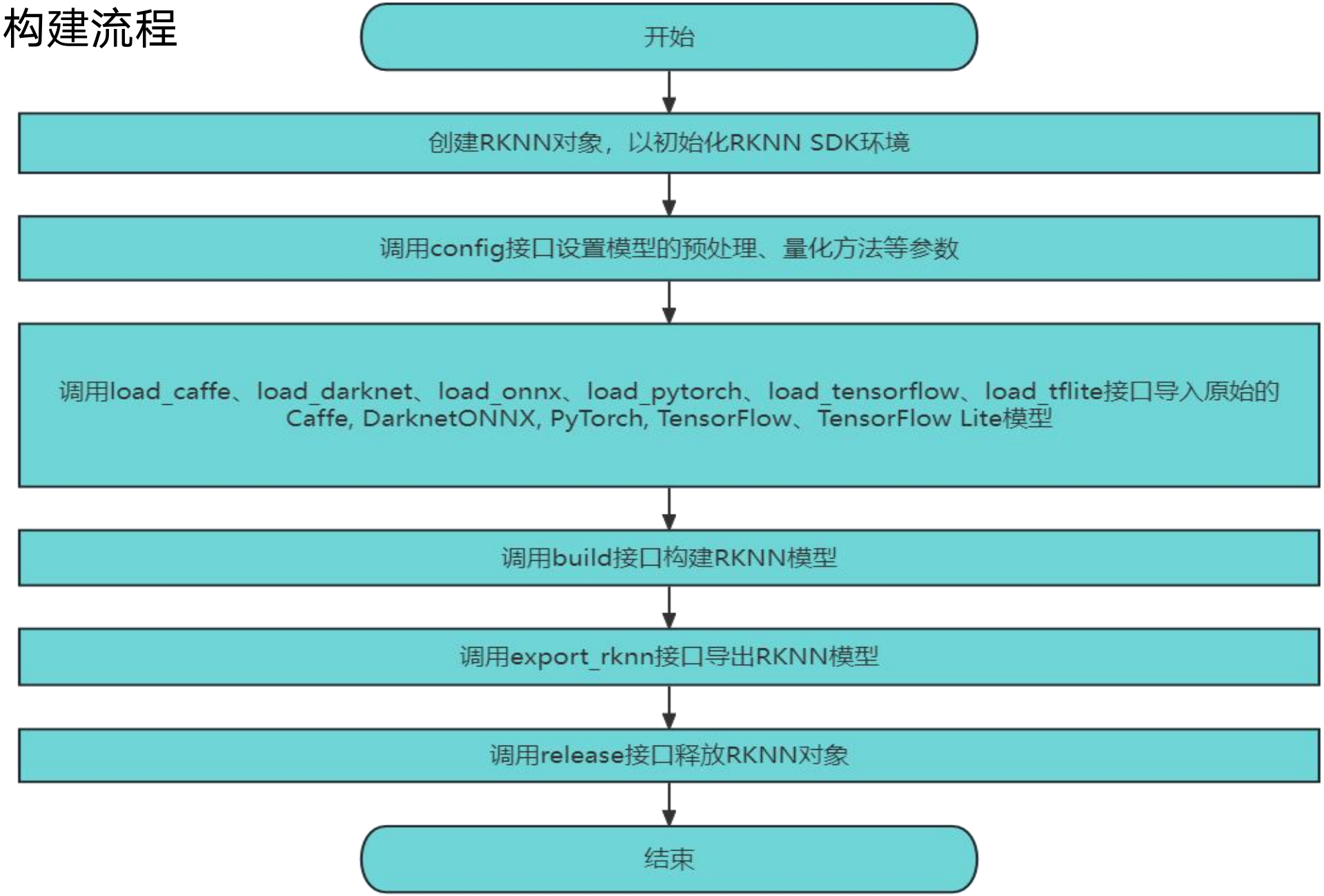
RKNN 模型构建流程


开始
创建RKNN对象，以初始化RKNN SDK环境
调用config接口设置模型的预处理、量化方法等参数
调用load_caffe、load_darknet、load_onnx、load_pytorch、load_tensorflow、load_tflite接口导入原始的Caffe, DarknetONNX, PyTorch, TensorFlow、TensorFlow Lite模型
调用build接口构建RKNN模型
调用export_rknn接口导出RKNN模型
调用release接口释放RKNN对象
结束

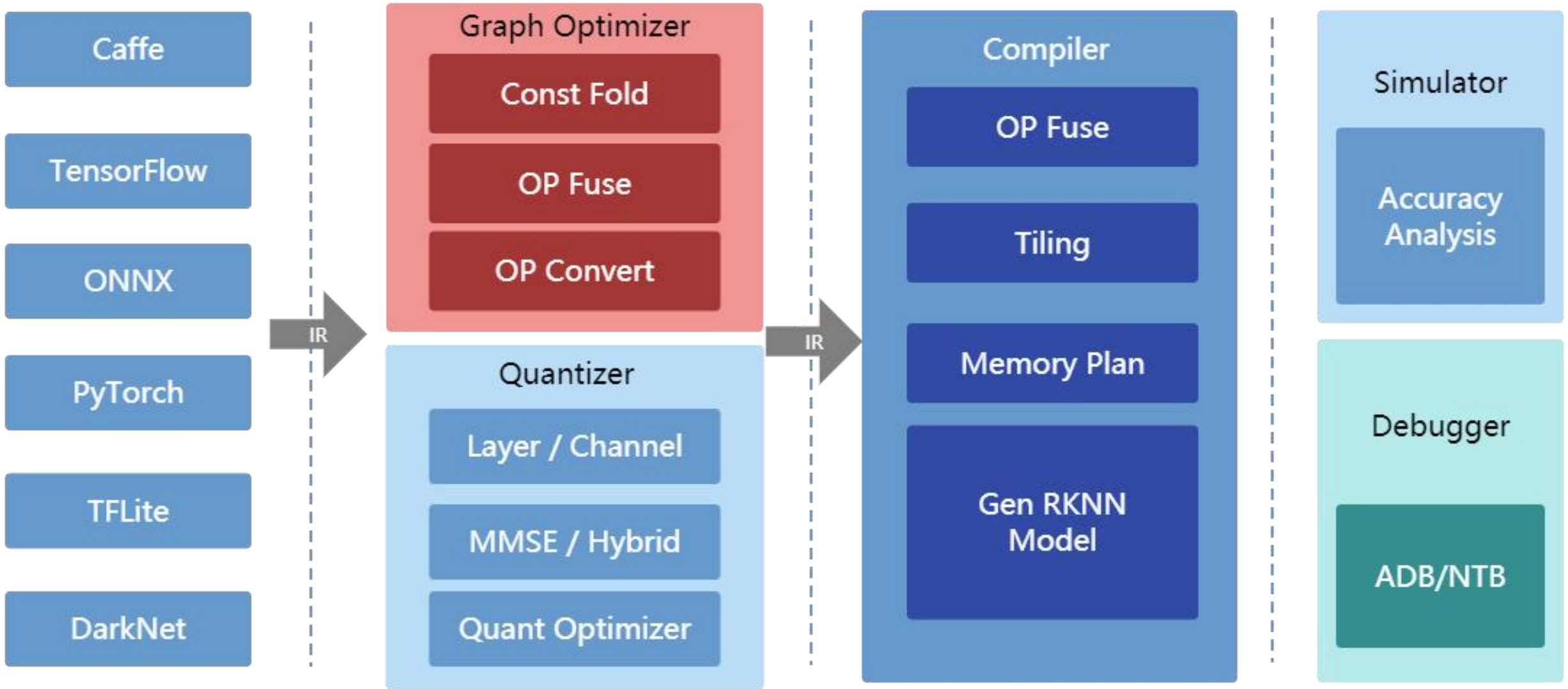
模型运行方式判断


加载模型是否为RKNN模型
否
加载的模型是常用深度学习模型
构建RKNN模型
判断模型是否运行在模拟器上
是
否
模型通过连扳推理的方式运行在RKNPU上
模型运行在RKNN Toolkit2 模拟器上

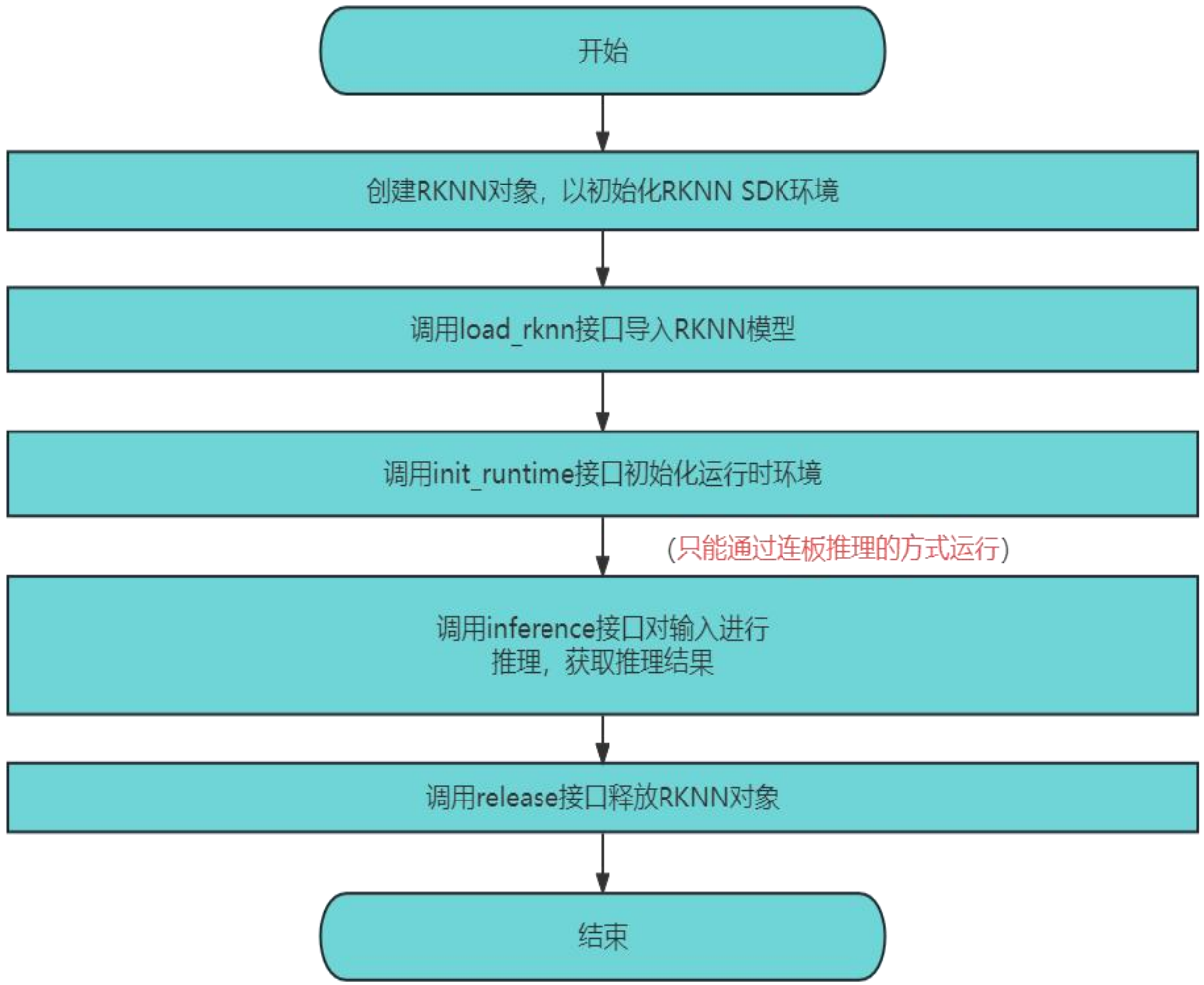
RKNN 模型构建流程


开始
创建RKNN对象，以初始化RKNN SDK环境
调用config接口设置模型的预处理、量化方法等参数
调用load_caffe、load_darknet、load_onnx、load_pytorch、load_tensorflow、load_tflite接口导入原始的Caffe, DarknetONNX, PyTorch, TensorFlow、TensorFlow Lite模型
调用build接口构建RKNN模型
调用export_rknn接口导出RKNN模型
调用release接口释放RKNN对象
结束

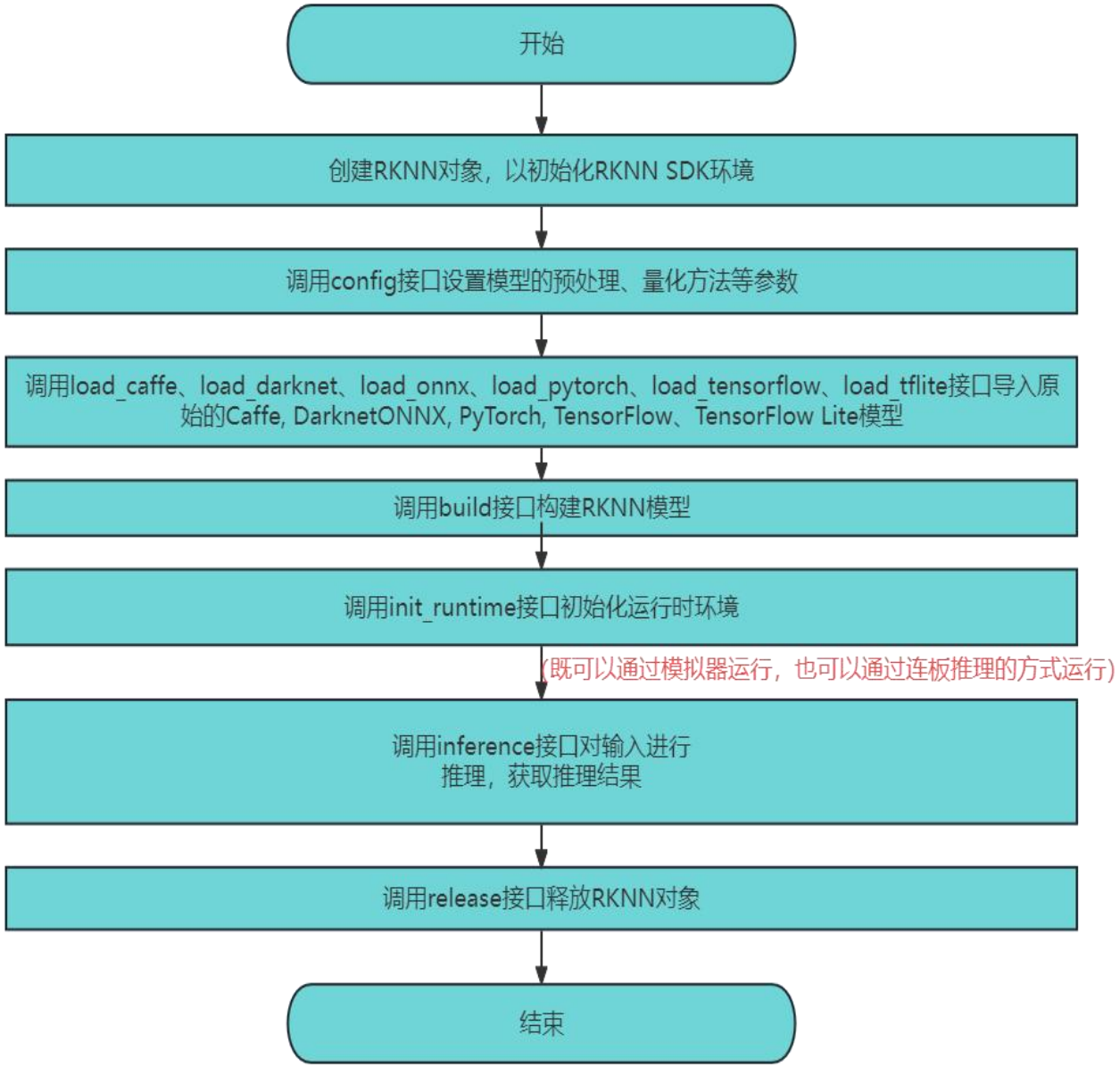
RKNN 模型构建框图


Caffe
TensorFlow
ONNX
PyTorch
TFLite
DarkNet
IR
Graph Optimizer
Const Fold
OP Fuse
OP Convert
Quantizer
Layer / Channel
MMSE / Hybrid
Quant Optimizer
IR
Compiler
OP Fuse
Tiling
Memory Plan
Gen RKNN Model
Simulator
Accuracy Analysis
Debugger
ADB/NTB

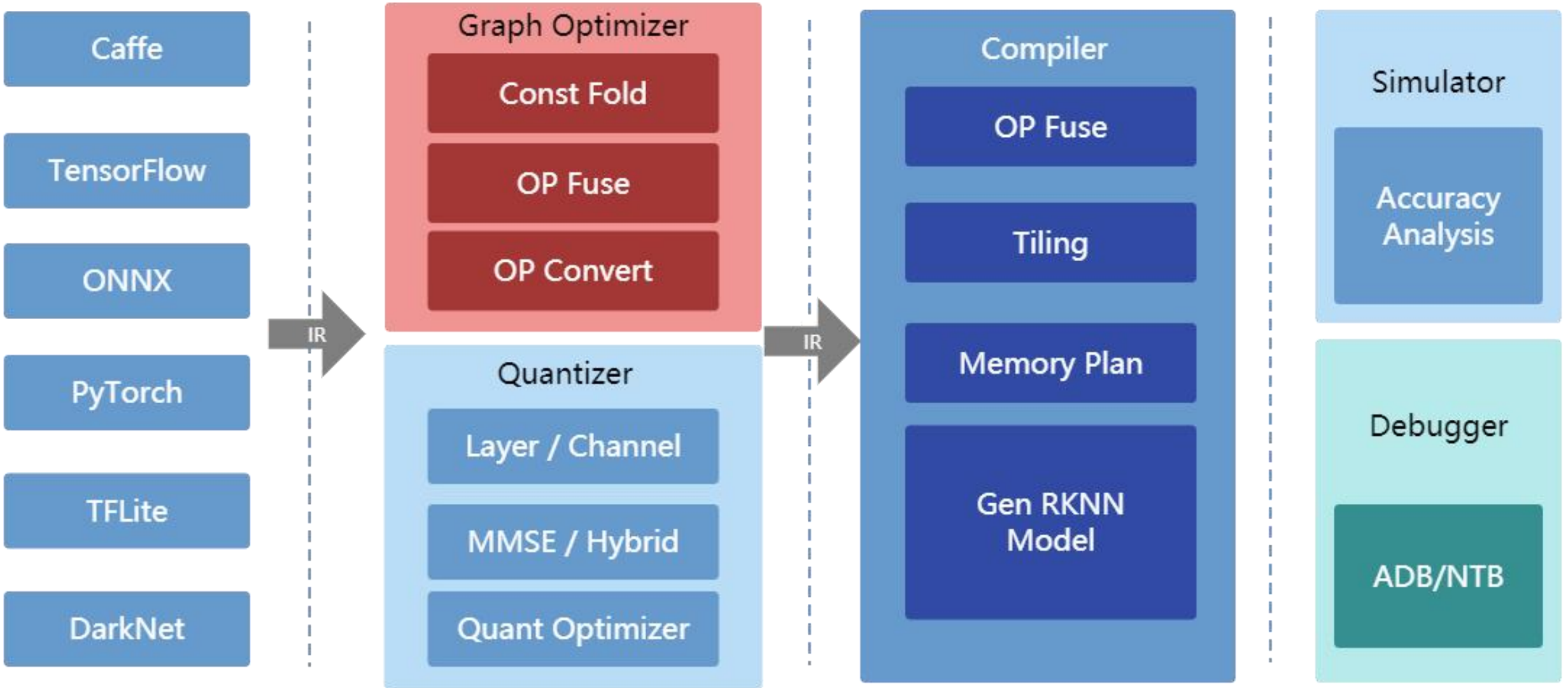
加载RKNN模型进行模型推理



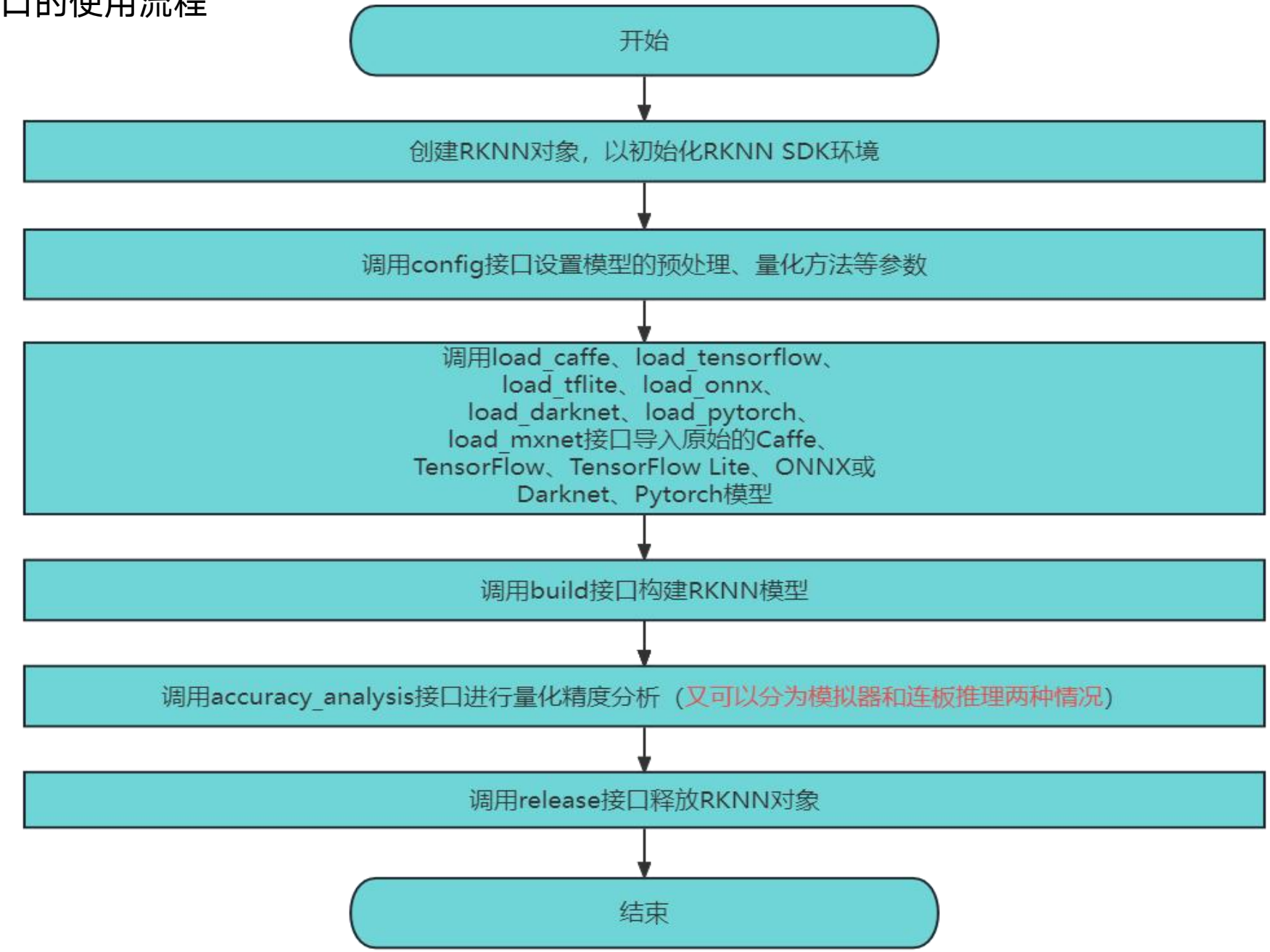
加载非RKNN模型进行模型推理



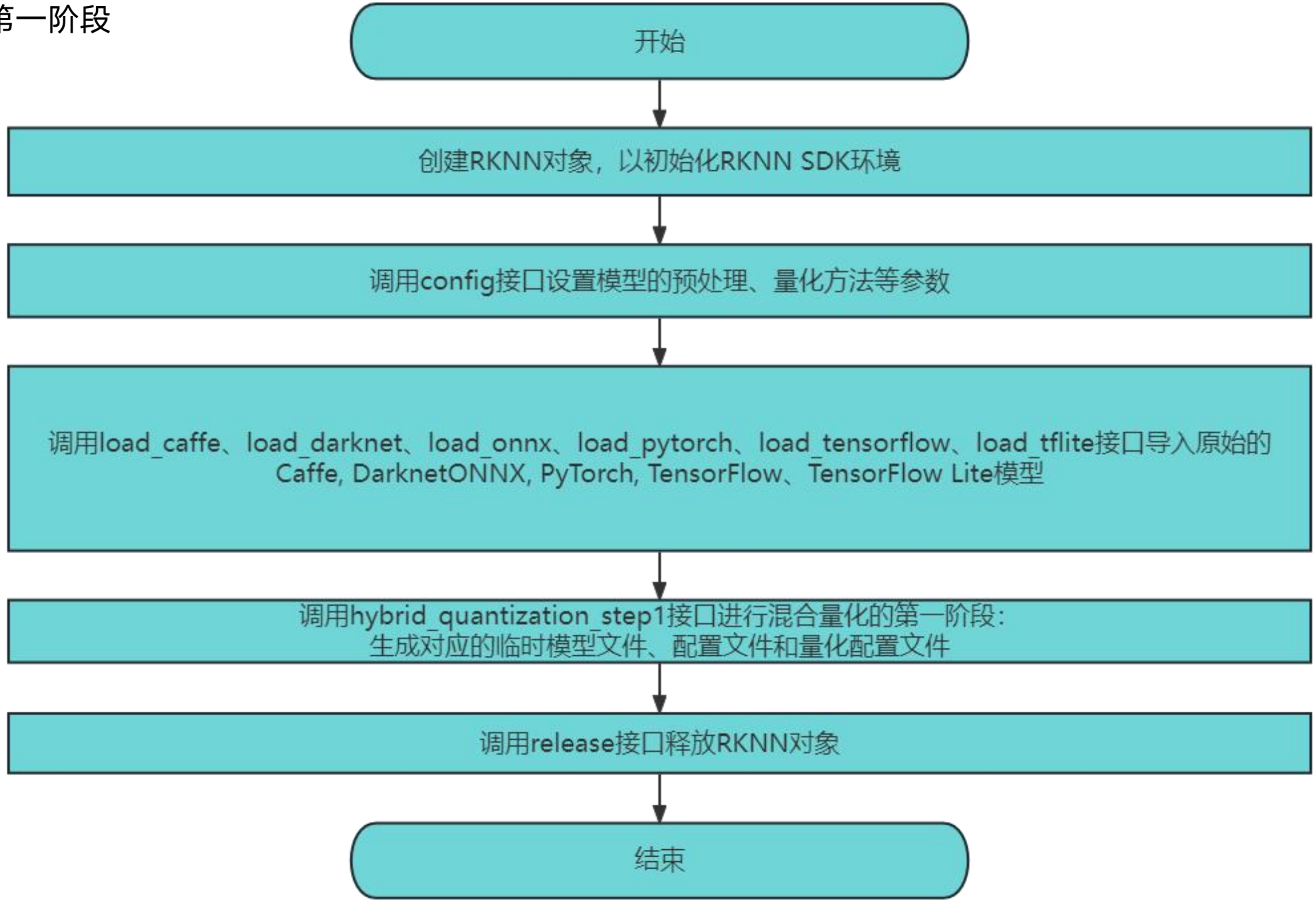
RKNN 模型构建框图


Caffe
TensorFlow
ONNX
PyTorch
TFLite
DarkNet
IR
Graph Optimizer
Const Fold
OP Fuse
OP Convert
Quantizer
Layer / Channel
MMSE / Hybrid
Quant Optimizer
IR
Compiler
OP Fuse
Tiling
Memory Plan
Gen RKNN Model
Simulator
Accuracy Analysis
Debugger
ADB/NTB

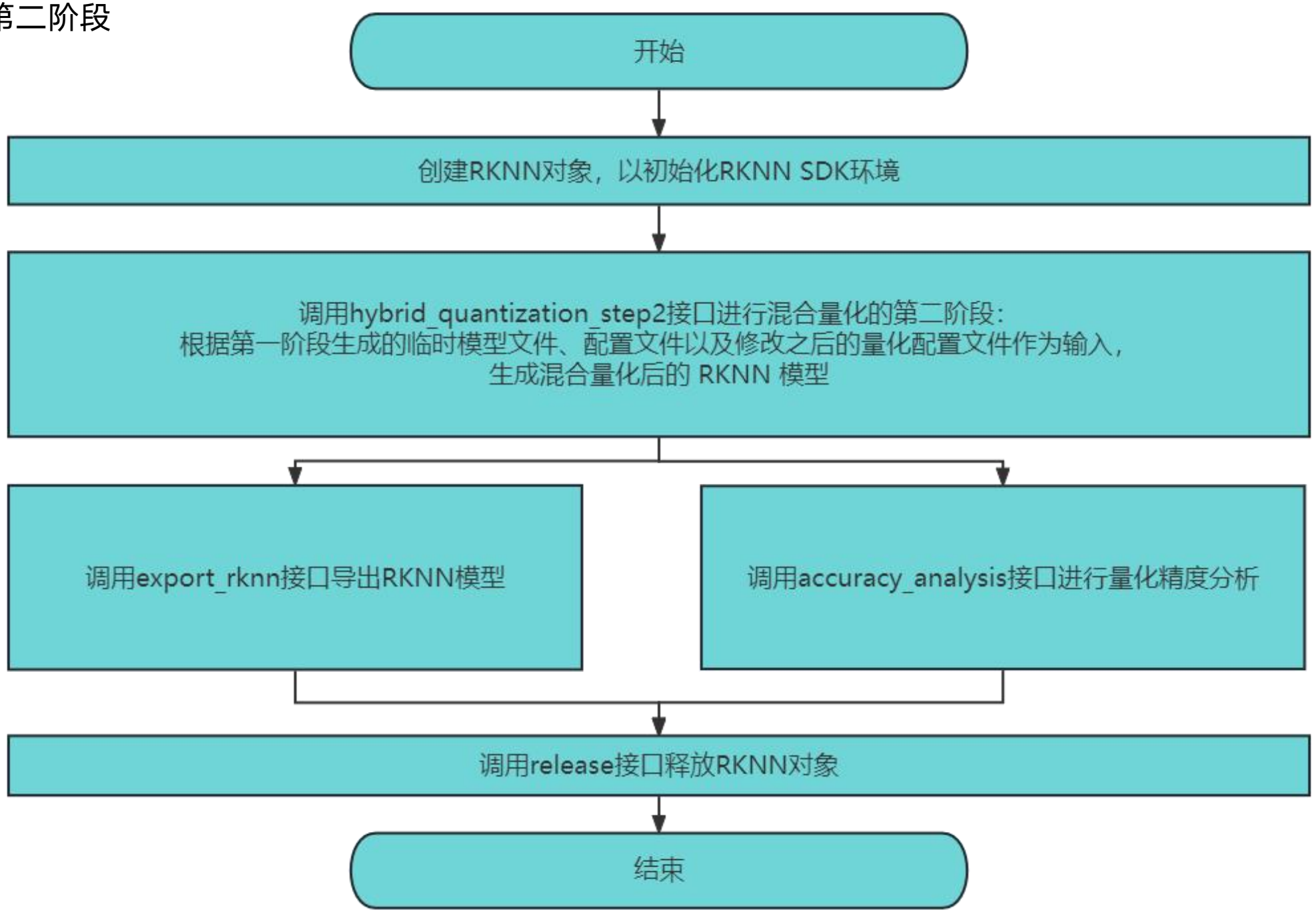
量化精度分析接口的使用流程



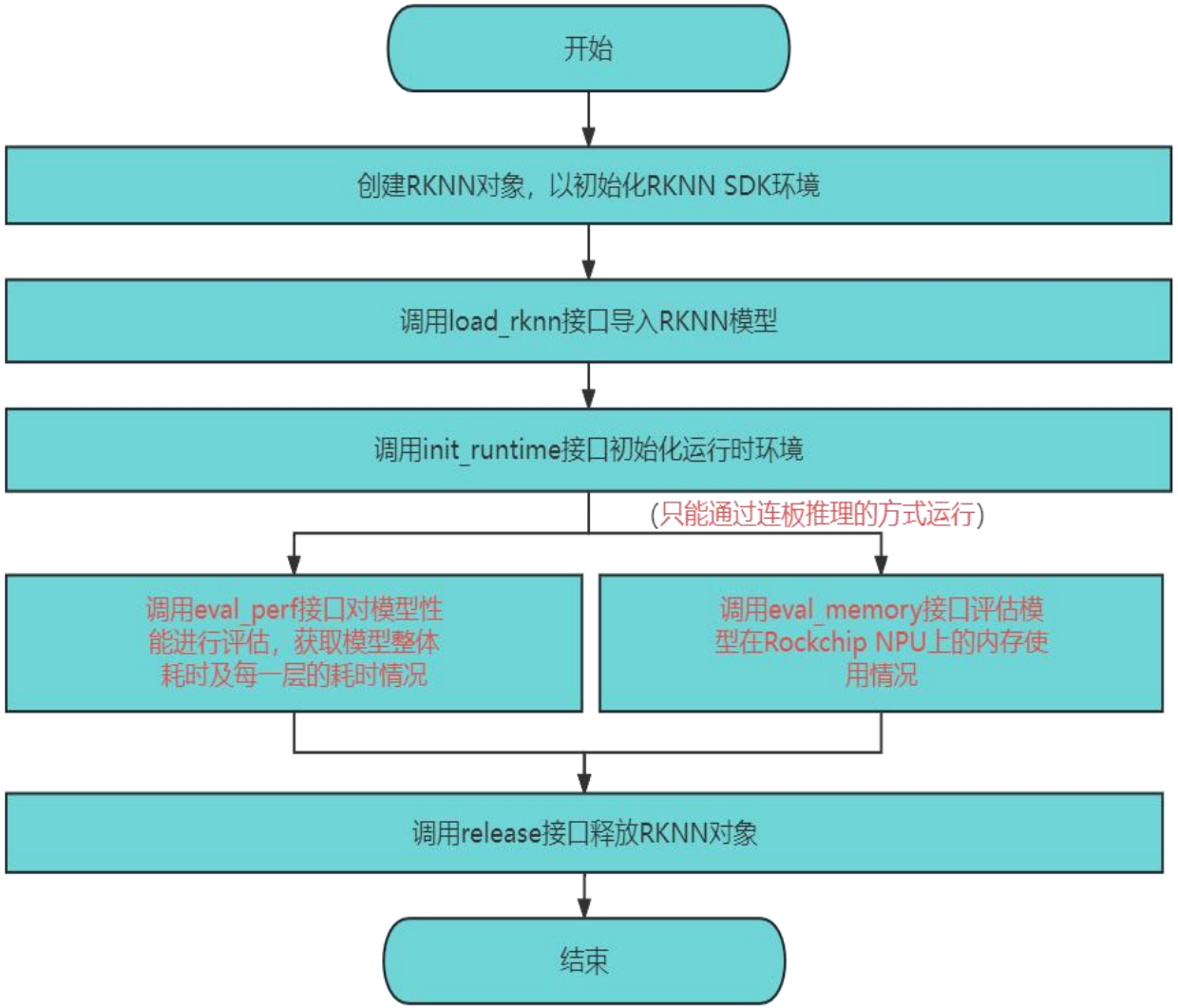
混合量化第一阶段

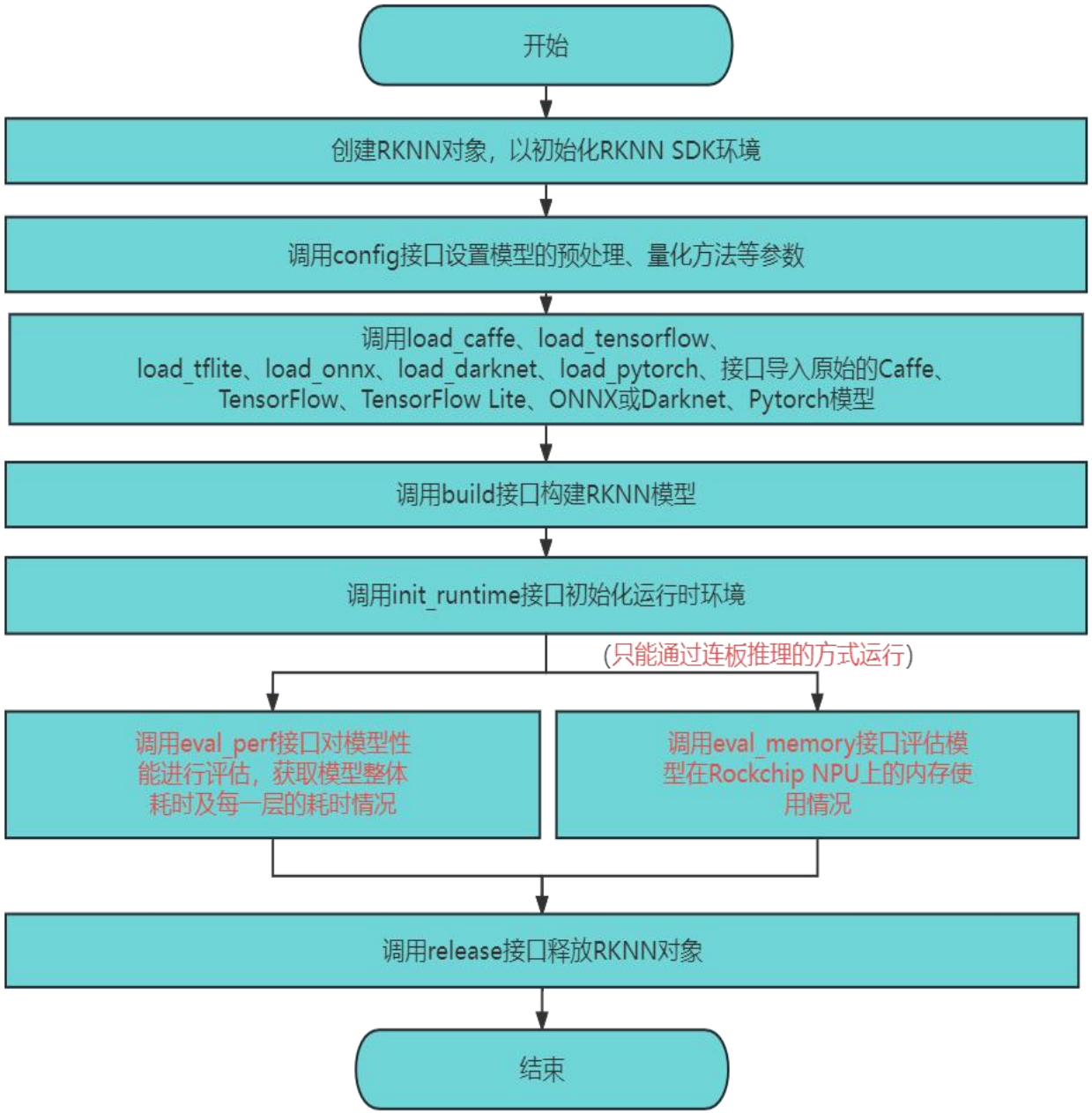

开始
创建RKNN对象，以初始化RKNN SDK环境
调用config接口设置模型的预处理、量化方法等参数
调用load_caffe、load_darknet、load_onnx、load_pytorch、load_tensorflow、load_tflite接口导入原始的Caffe, DarknetONNX, PyTorch, TensorFlow、TensorFlow Lite模型
调用hybrid_quantization_step1接口进行混合量化的第一阶段:
生成对应的临时模型文件、配置文件和量化配置文件
调用release接口释放RKNN对象
结束

混合量化第二阶段

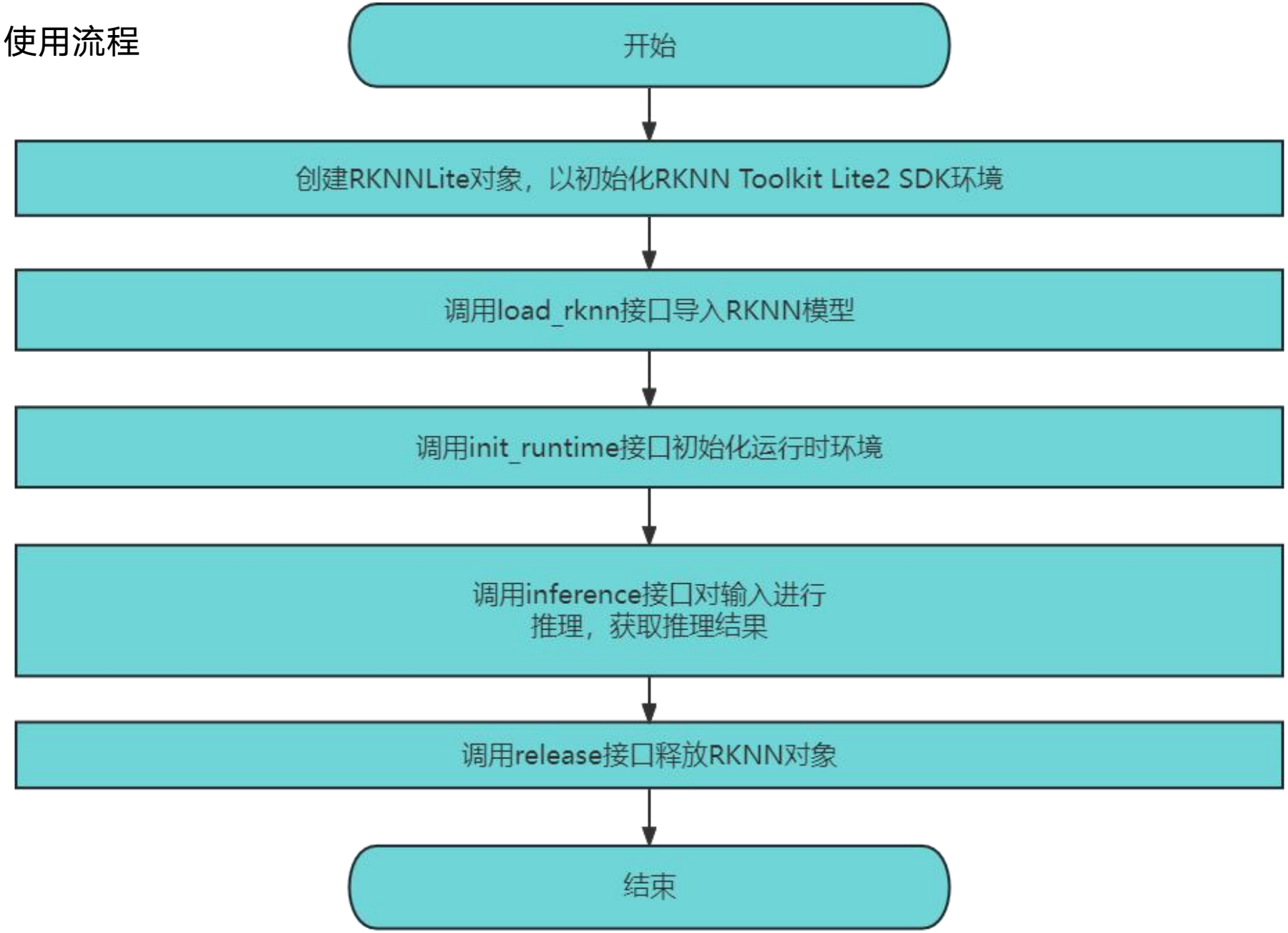

开始
创建RKNN对象，以初始化RKNN SDK环境
调用hybrid_quantization_step2接口进行混合量化的第二阶段：
根据第一阶段生成的临时模型文件、配置文件以及修改之后的量化配置文件作为输入，
生成混合量化后的 RKNN 模型
调用export_rknn接口导出RKNN模型
调用accuracy_analysis接口进行量化精度分析
调用release接口释放RKNN对象
结束

性能评估和内存评估

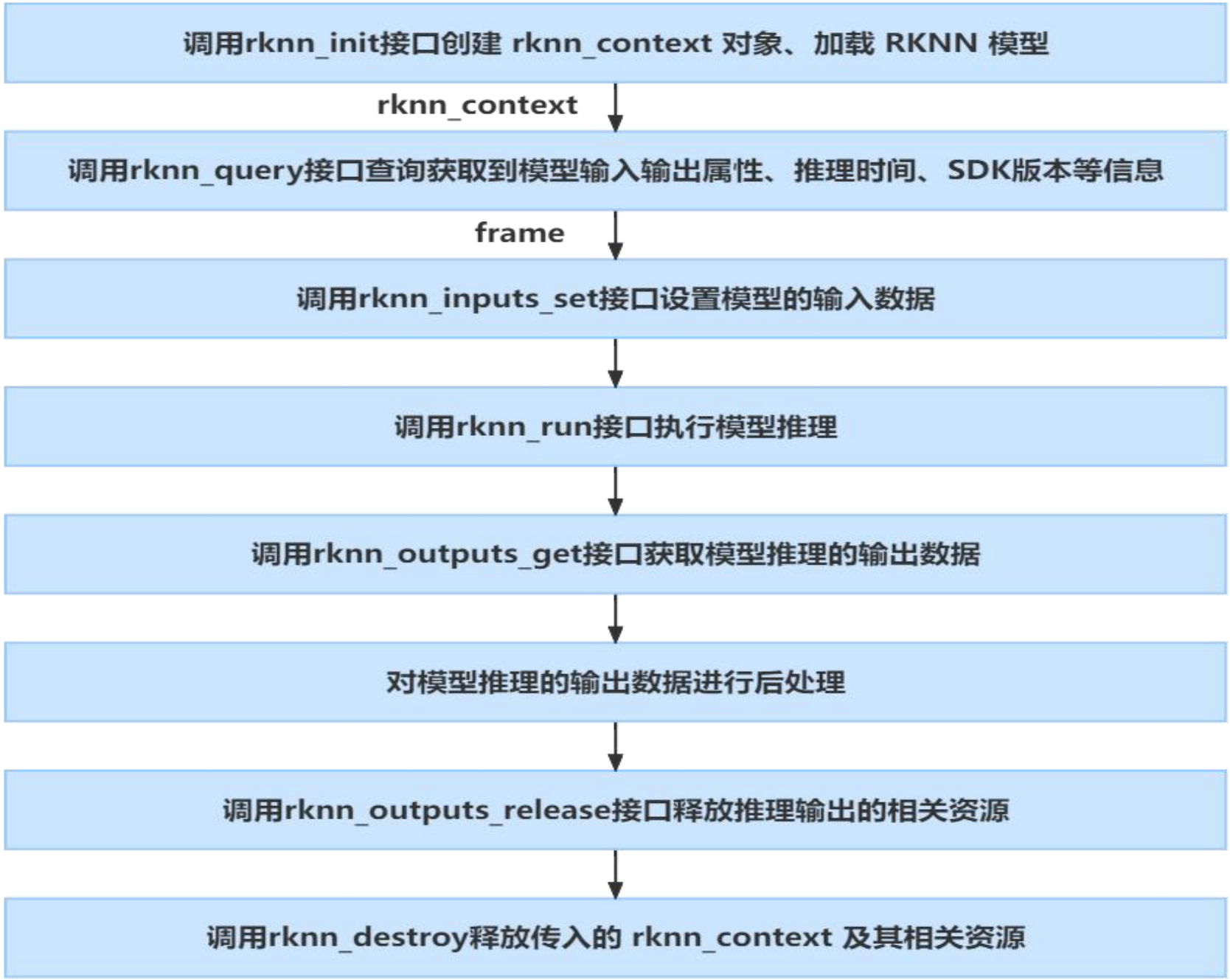

开始
创建RKNN对象，以初始化RKNN SDK环境
调用load_rknn接口导入RKNN模型
调用init_runtime接口初始化运行时环境
(只能通过连板推理的方式运行)
调用eval_perf接口对模型性能进行评估，获取模型整体耗时及每一层的耗时情况
调用eval_memory接口评估模型在Rockchip NPU上的内存使用情况
调用release接口释放RKNN对象
结束


开始
创建RKNN对象，以初始化RKNN SDK环境
调用config接口设置模型的预处理、量化方法等参数
调用load_caffe、load_tensorflow、load_tflite、load_onnx、load_darknet、load_pytorch、接口导入原始的Caffe、TensorFlow、TensorFlow Lite、ONNX或Darknet、Pytorch模型
调用build接口构建RKNN模型
调用init_runtime接口初始化运行时环境
(只能通过连板推理的方式运行)
调用eval_perf接口对模型性能进行评估，获取模型整体耗时及每一层的耗时情况
调用eval_memory接口评估模型在Rockchip NPU上的内存使用情况
调用release接口释放RKNN对象
结束

RKNN Toolkit lite2 使用流程



RKNPU2 通用API使用流程：



零拷贝API使用流程


rknn_init接口创建 rknn_context 对象、加载 RKNN 模型
rknn_context
调用rknn_query接口查询获取到模型输入输出属性、推理时间、SDK版本等信息
调用rknn_create_mem接口申请内存
调用rknn_set_io_mem接口让NPU 使用上一步申请的内存
调用rknn_run接口执行模型推理
对模型推理的输出数据进行后处理
调用rknn_destroy_mem毁rknn_tensor_mem结构体，释放用户分配的内存
调用rknn_destroy释放传入的 rknn_context 及其相关资源