主讲人 ：正点原子团队
硬件平台：正点原子ATK-DLRV1126开发板

资料下载：www.openedv.com/docs/index.html
教学平台：www.yuanzige.com
天猫店铺：zhengdianyuanzi.tmall.com
技术论坛：www.openedv.com/forum.php
公众平台：正点原子

■ 模型的量化

1，模型的量化方法

2，为什么要对模型进行量化

3，RKNN Toolkit常规量化

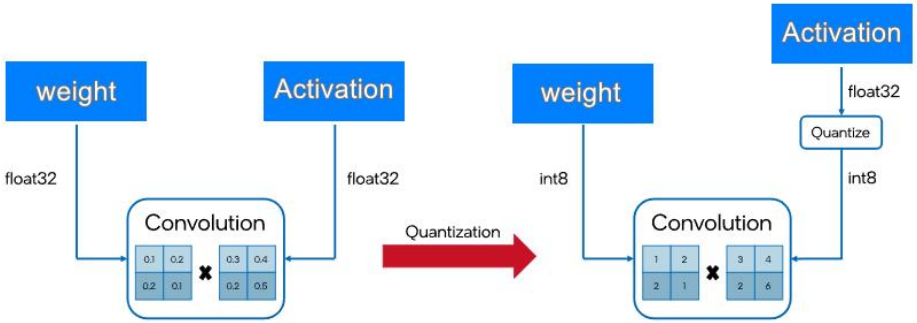
4，RKNN Toolkit混合量化

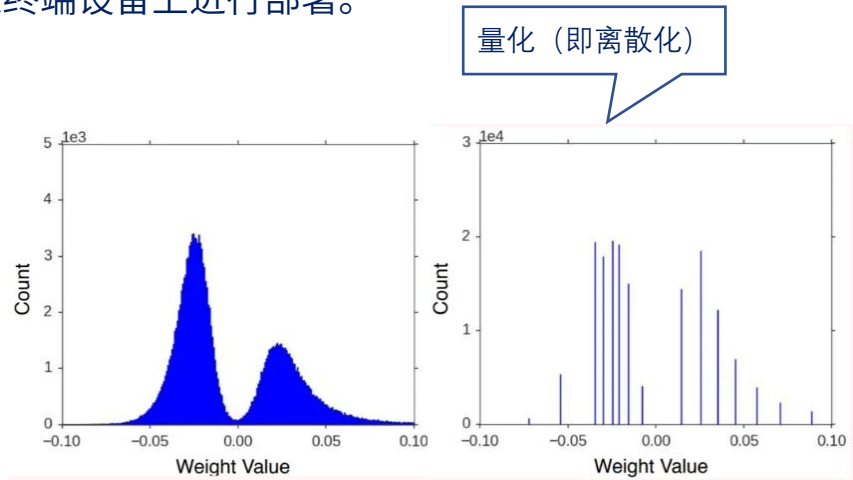
1、模型的量化方法

什么是模型的量化：

模型的量化，主要是指将模型中的权重和激活值，由浮点数据转换成低比特（更低位宽）的定点数据。

神经网络模型性能越高，引入的参数量和计算量就越大，通过模型的量化，有效减小模型的大小和计算强度，有效地减少模型的内存占用，这非常有利于模型在边缘终端设备上进行部署。





1、模型的量化方法

模型的量化方法主要有两种：

① 先基于各个深度学习框架训练得到一个没有量化的模型，再通过rknn-toolkit1.7.5工具来进行量化。

- 量化方式：训练后静态量化
- 支持的量化精度类型：int16，int8，uint8
- 支持的量化粒度：per-tensor（或 per-layer），不支持 per-channel 量化

② 使用各个深度学习框架导出量化后的模型，再通过rknn-toolkit工具直接把该已经量化的模型转换为RKNN模型。

- 量化方式：训练后静态量化、量化感知训练(QAT)
- 支持的量化精度类型：int8, uint8
- 支持的深度学习框架： PyToch(v1.9.0)、ONNX(Onnxruntime v1.5.1)、Tensorflow、TFLite

1、模型的量化方法

量化精度类型：

工具	量化方法	平台支持情况
rknn-toolkit2 1.5.2	默认仅支持非对称量化（asymmetric_quantized-8）	RK3566/RK3568/RK3562 RK3588/RK3588S RV1103/RV1106
rknn-toolkit 1.7.5 （自带可视化工具）	非对称量化（asymmetric_quantized-u8）→默认采用的量化方式	RK1808/RK1806 RV1126/RV1109
	动态定点量化（dynamic_fixed_point-i8和dynamic_fixed_point-i16）	RK3399pro

用uint8进行定点量化，可表示范围为 [0,255]（这种量化方式对精度的损失较小，对于某些模型来说，uint8方式比int8精度高）

用int8进行定点量化，可表示范围为[-127，127]（对于某些模型来说，int8方式比uint8精度高）

int16的量化方式，量化公式和int8的一样，只不过位宽是16位（对于某些量化到 8位后精度损失较大的模型，可以考虑使用此量化方式）

rknn-toolkit 1.7.5默认使用的是uint8，rknn-toolkit2 1.5.2里默认使用的 int8，两者默认的量化方式，其量化公式是一样的

混合量化：模型的某些层可以采用不同的量化精度类型（多于1种量化精度类型）

1、模型的量化方法

由浮点到定点的量化公式：$Q = \frac{R}{S} + Z$

由定点到浮点反量化公式：R = (Q - Z)*S

其中：

$S = \frac{R_{max} - R_{min}}{Q_{max} - Q_{min}}$　　$Z = Q_{max} - R_{max} / S$

Q表示量化后的定点值；R表示真实的浮点值；S表示定点量化后可表示的最小刻度；Z表示0浮点值对应的量化定点值。

例如，激活值范围为[-1.0, 5.0]，用int8进行模型量化，定点量化值范围为[-128, 127]，有一个真实的激活值为0.2（即R=0.2），求这个激活值0.2量化后的值，计算如下：

S = （5.0 + 1.0）/ （127 + 128）= 6.0 / 255 ≈ 0.2353　　Z = 127 - 5.0 / 0.2353 ≈ 105.7505

Q = 0.2 / 0.2353 + 105.7505 ≈ 106.6005

那么，该浮点激活值0.2经过int8量化后，得到的值为106.6005

2、为什么要对模型进行量化

神经网络模型的参数量（权重和激活值）是庞大的，所以计算量也是比较大的，这样的模型，不利于在资源受到限制（存储空间有限、内存有限、算力有限、带宽有限等等）的嵌入式系统和边缘设备上跑。

量化的优势：

① 有效压缩模型文件，节省磁盘存储空间，节省成本

经过量化操作，模型中的权重和激活值由float32类型转换为低位宽的数据类型，可以有效压缩模型文件， 从而节省磁盘存储空间。如，将float32变为uint8，压缩后，模型文件大小变为原来的四分之一。

② 使得模型能够更快地进行推理，可以加速模型的运行速度

更小的数据类型通常需要更少的位运算和存储器访问，采用低精度计算的好处是：可以减少计算量，可以减少计算需求，加快推理速度。

float32转为int16，理论上，速度可以提升2倍；

float32转为int8，理论上，可以达到原来4倍的速度；

2、为什么要对模型进行量化

神经网络模型的参数量（权重和激活值）是庞大的，所以计算量也是比较大的，这样的模型，不利于在资源受到限制（存储空间有限、内存有限、算力有限、带宽有限等等）的嵌入式系统和边缘设备上跑。

量化的优势：

③ 降低功耗

模型量化了以后，比如将float32类型量化为uint8类型，在推理时，完全是在8 位输入和输出上执行的，这可减少推理计算所需的计算资源，能更快地进行推理，从而间接地节省了功耗。

④ 节省内存和带宽

与没有量化之前的float32浮点运算相比，uint8类型的浮点运算只需要四分之一的内存和带宽，大提升了内存的访问效率，有效地避免了 RAM 访问的瓶颈。

2、为什么要对模型进行量化

量化的缺点：一个致命的缺点，即精度损失！

对模型进行量化，实际上是以牺牲模型精度为前提来提高模型在嵌入式设备上的运行效率和性能的。

模型的量化，在牺牲较少精度，或者精度损失在合理的范围之内时，如果在推理模型时对精度要求没有那么高的场合下，可以使用低精度的技术来完成。

理论上，模型推理的准确度（不过也和具体的硬件NPU有关）：

flaot32类型 > int16类型 > int8类型 >uint8类型

2、为什么要对模型进行量化

很多研究表明，对于很多CNN网络模型，使用float 16类型、int8类型、uint8类型来推理时，小的精度损失是有的，但没有特别特的精度损失，精度损失可以在接受的范围内：

图片大小	深度	Top-1 精度（flaot32类型）	Top-1 精度（uint8类型）
128	0.25	0.415	0.399
128	0.5	0.563	0.549
128	0.75	0.621	0.598
128	1	0.652	0.64
160	0.25	0.455	0.435
160	0.5	0.591	0.577
160	0.75	0.653	0.639
160	1	0.68	0.673
192	0.25	0.477	0.458
192	0.5	0.617	0.604
192	0.75	0.672	0.662
192	1	0.7	0.69
224	0.25	0.498	0.482
224	0.5	0.633	0.622
224	0.75	0.684	0.679
224	1	0.709	0.697

MobileNet 模型在 Imagenet 验证集上的 Top-1 精度对比（float32和uint8）

3、RKNN Toolkit常规量化

常规量化，指在模型中的权重和激活值只采用一种量化精度类型（例如，要么只使用uint8，要么只使用int8）。

```
from rknn.api import RKNN
rknn = RKNN()
```

调用rknn.config()配置模型，主要进行：输入的预处理（如归一化操作、通道顺序的调整）、量化类型选择、量化参数优化算法选择、模型优化等级配置、指定 RKNN 模型目标运行平台等

调用rknn.load_tensorflow()、rknn.load_tflite()、rknn.load_pytorch()、rknn.load_onnx()……等等接口，加载对应深度学习框架下的模型

调用rknn.build ()，根据加载的模型结构及权重参数，构建对应的 RKNN 模型，在这一部分会进行：是否对模型进行量化、确定量化的数据集、要推理的批量大小

```
rknn.export_rknn()
```

调用rknn.accuracy_analysis()，逐层对比浮点模型和量化模型的输出结果，输出余弦距离和欧式距离，用于分析量化模型精度下降原因

```
rknn.release()
```

代码参考：nomal_quantization.ipynb

3、RKNN Toolkit常规量化

normal_quantization_analysis目录	
entire_qnt目录	保存整个量化模型完整运行时每一层的结果（已转成 float32）
Fp32目录	保存整个浮点模型推理时每一层的结果
individual_qnt目录	量化模型拆成一层一层运行的结果，每一层推理时的输入是上一层用浮点模型推理的结果
qnt_npu_dump目录	保存量化模型逐层在 NPU 上实际运行时的结果（将量化模型拆成一层一层后逐个放到 NPU 上运行，所用的输入为浮点模型上一层的结果）
entire_qnt_error_analysis.txt	记录量化模型完整运算时每一层结果与浮点模型结果的余弦距离和欧氏距离
individual_qnt_error_analysis_on_npu.txt	记录量化模型逐层在NPU硬件设备上运行时每一层结果与浮点模型结果的余弦距离和欧氏距离（配置rknn.accuracy_analysis()的参数target以后，就生成该文件）
individual_qnt_error_analysis.txt	记录量化模型逐层运行时每一层的结果与浮点模型结果的余弦距离和欧氏距离
dump_data_distribute目录	
保存每一层的 weihgt/bias（如果有）和输出数据的直方图	

注意：

余弦距离越小（cosine_norm），表明量化后的精度下降得越厉害

欧式距离越大（eculidean_norm），表明量化后的精度下降得越厉害

4、RKNN Toolkit混合量化

混合量化，指在网络模型中，权重和激活值可以采用不止一种量化精度类型，有多种量化精度类型，例如，某些层的权重和激活值采用uint8量化精度类型，某些层的权重和激活值采用int16量化精度类型，这里就有两种量化精度类型在整个模型中混合使用了，这种方式就叫做模型的混合量化。

对于rknn-toolkit1.7.5版本的混合量化功能来说，目前支持如下三种用法：

① 将指定的量化层改成非量化层（如保持采用 float32 进行计算），但是因为 NPU 的浮点数算力较低，推理性能会下降。

② 将指定的非量化层改成量化层。

③ 修改指定量化层的量化参数。

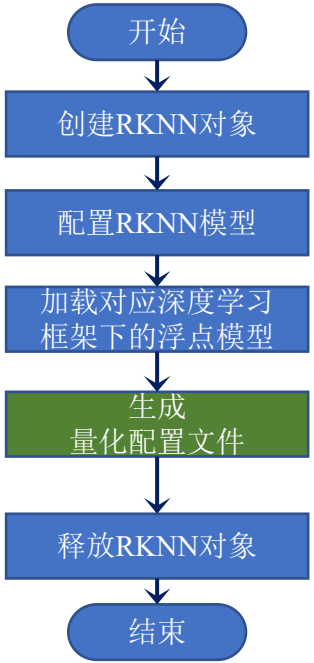
4、RKNN Toolkit混合量化

在很多场合下使用混合量化功能是很有必要的，特别是在后期部署YOLO模型，可能需要经常手动调整某些层的量化精度类型。

采用完全量化功能，即每层的权重和激活值都量化为同一种数据精度类型，虽然在提高模型推理速度的基础上尽量保证模型精度了，但是仍有一些层在量化以后出现精度下降较多的情况，影响了模型整体的精度，为了解决该问题，可以使用rknn-toolkit的混合量化功能：

即把那些精度不高的层进行调整，改为其它量化方式，这些层的精度提高了以后，整体模型的精度就得到提高了。

4、RKNN Toolkit混合量化



from rknn.api import RKNN
rknn = RKNN()

调用rknn.config()配置模型，主要进行：输入的预处理（如归一化操作、通道顺序的调整）、量化类型选择、量化参数优化算法选择、模型优化等级配置、指定 RKNN 模型目标运行平台等

调用rknn.load_tensorflow()、rknn.load_tflite()、rknn.load_pytorch()、rknn.load_onnx()……等等接口，加载对应深度学习框架下的模型

调用hybrid_quantization_step1()接口生成量化配置文件：
模型结构文件：.json文件；
权重数据文件：.data文件；
模型量化配置文件：.quantization.cfg文件；

rknn.release()

生成模型量化配置文件（.quantization.cfg文件）以后，可根据需要自己修改该文件，配置量化或者不量化的层以及对应的量化方式

混合量化第一步接口调用流程
参考demo：step1.ipynb

4、RKNN Toolkit混合量化

from rknn.api import RKNN
rknn = RKNN()

调用rknn.config()配置模型，主要进行：输入的预处理（如归一化操作、通道顺序的调整）、量化类型选择、量化参数优化算法选择、模型优化等级配置、指定 RKNN 模型目标运行平台等

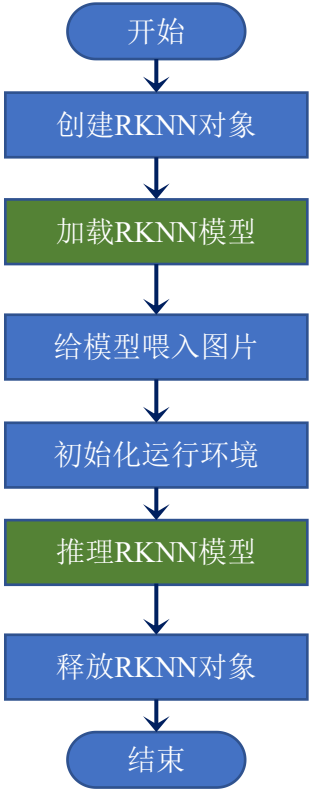
调用rknn.hybrid_quantization_step2()接口，根据第一步生成的文件（.json文件、.data文件和.quantization.cfg文件）来构建混合量化RKNN模型

rknn.export_rknn()

rknn.release()

混合量化第二步接口调用流程
参考demo：step2.ipynb

4、RKNN Toolkit混合量化



from rknn.api import RKNN
rknn = RKNN()

调用rknn.load_rknn ()接口加载混合量化后的RKNN模型

读取图片，给模型喂入输入图片

调用rknn.init_runtime()接口初始化运行环境（ADB连板推理，需要指定target目标平台）

调用rknn.inference()接口，完成模型的推理

rknn.release()

第三步，可以跑一下混合量化后的模型，查看推理的效果

参考demo： step3.ipynb

4、RKNN Toolkit混合量化

from rknn.api import RKNN
rknn = RKNN()

调用rknn. export_rknn_quant_params()加载RKNN模型，并将参数导出到指定的文件中（.yml文件）

rknn.release()

若要导出RKNN 模型量化参数，可参考该流程
参考demo： step4.ipynb

天猫店铺：https://zhengdianyuanzi.tmall.com