推理引擎 OpenPPL 实战训练营

商汤 sensetime

OpenPPL 量化工具实践

2022年3月16日星期三

课程安排

课程安排	主讲人	课程时间
第一期：商汤自研AI推理引擎 OpenPPL 的实践之路	高洋	2021年12月07日
第二期：编程工作坊：基于 OpenPPL 的模型推理与应用部署	欧国宇	2021年12月16日
第三期：OpenPPL之通用架构下的性能优化概要	许志耿	2021年12月29日
第四期：模型大小与推理速度的那些事儿	田子宸	2022年01月06日
第五期：OpenPPL CUDA技术解析	李天健	2022年01月13日
第六期：性能调优实战（x86篇）	梁杰鑫	2022年02月17日
第七期：OpenPPL RISC-V 指令集初探	焦明俊	2022年02月24日
第八期：OpenPPL 在 ARM Server 上的技术实践	邱君仪	2022年03月03日
第九期：OpenPPL 量化工具实践	**纪喆**	**2022年03月15日**

「商汤学术」公众号

可以回复“抽奖”试试哦

主讲人介绍

纪　喆　商汤科技异构计算工程师

- 硕士毕业于北京大学
- 商汤科技异构计算工程师，目前在商汤科技高性能计算部门负责参与多平台的量化工具链开发

模型离线量化工具PPQ

纪喆 HPC

总览

- 量化概述
- 量化工具框架
- 量化工具图调度与融合
- 部署平台量化适配
  - 模拟器量化优化
  - 部署平台量化对齐
- 量化性能展示

量化概述

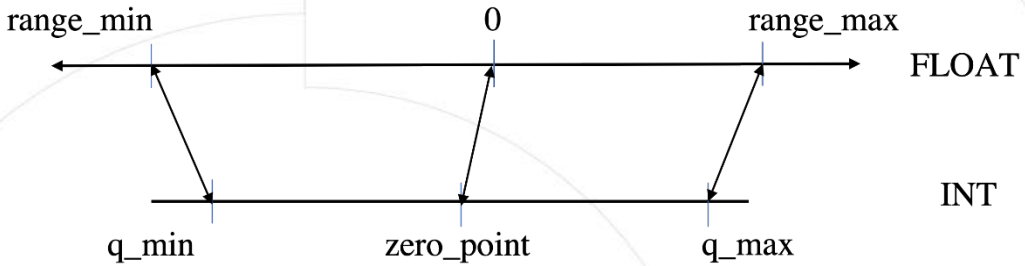
量化：将连续的信号取值，离散化为有限个取值的过程

数学表达：

$$s = (range\_max - range\_min) \,/\, (2^{bits} - 1)$$

$$q_x = clip(\, round\left(\frac{x}{s}\right) + z, q_{max}, q_{min})$$

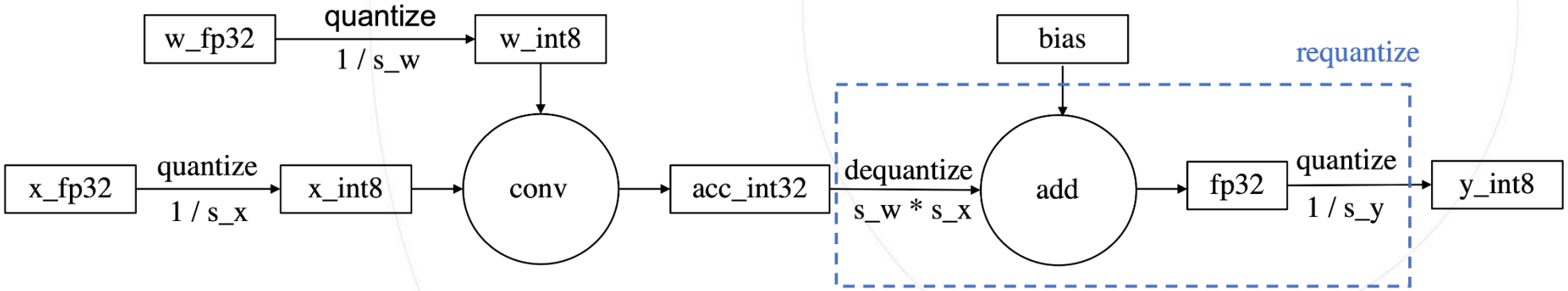
$$deq_x = (q_x - z) * s$$



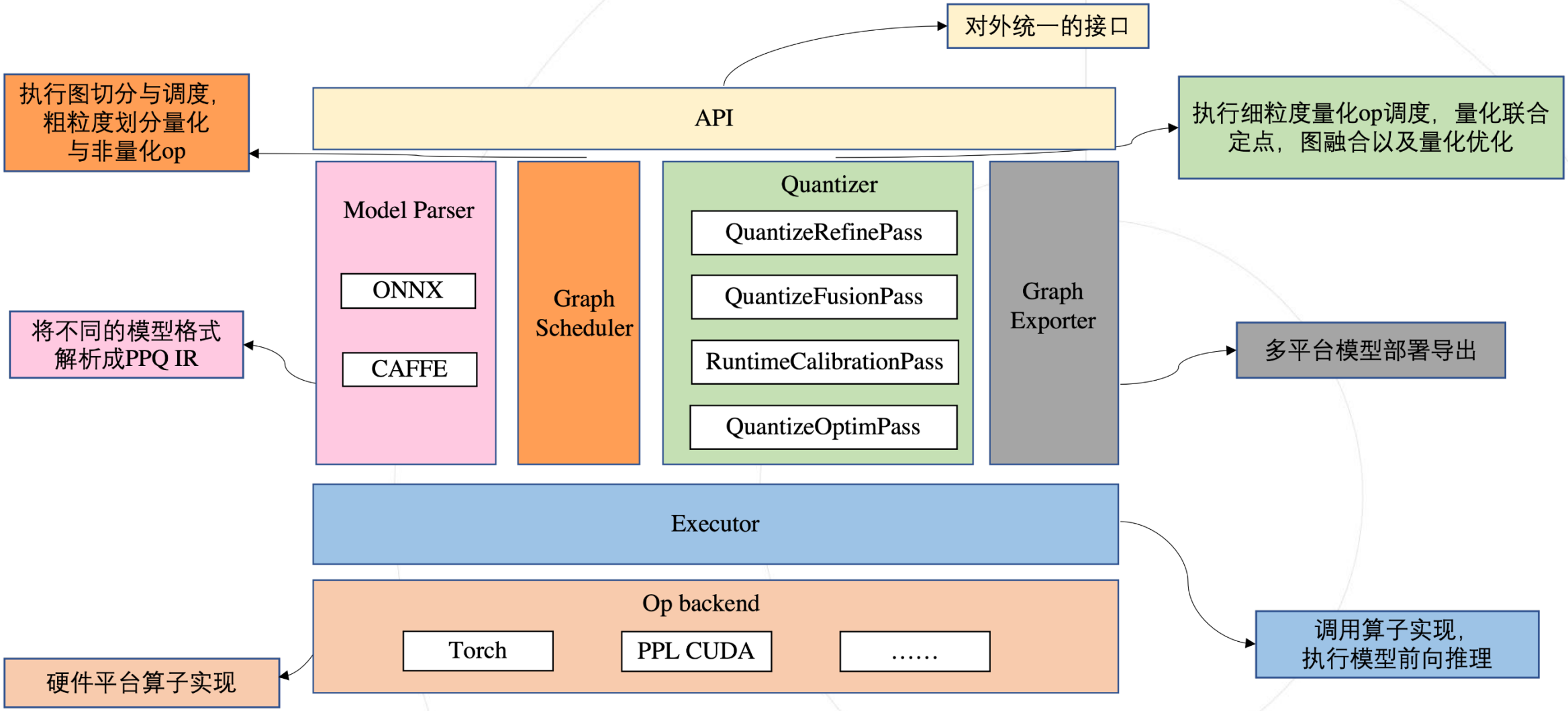
以Conv量化计算为例：

$$q_y = \frac{s_w s_x}{s_y}[(q_w - z_w) * (q_x - z_x) + q_{bias}] + z_y$$

对于不同硬件平台：$\frac{s_w s_x}{s_y} \approx \frac{K}{2^{shift}} \approx power\ of\ 2$



量化工具框架


对外统一的接口
执行图切分与调度，
粗粒度划分量化
与非量化op
API
执行细粒度量化op调度，量化联合
定点，图融合以及量化优化
Model Parser
ONNX
CAFFE
Graph
Scheduler
Quantizer
QuantizeRefinePass
QuantizeFusionPass
RuntimeCalibrationPass
QuantizeOptimPass
Graph
Exporter
将不同的模型格式
解析成PPQ IR
多平台模型部署导出
Executor
Op backend
Torch
PPL CUDA
……
硬件平台算子实现
调用算子实现，
执行模型前向推理

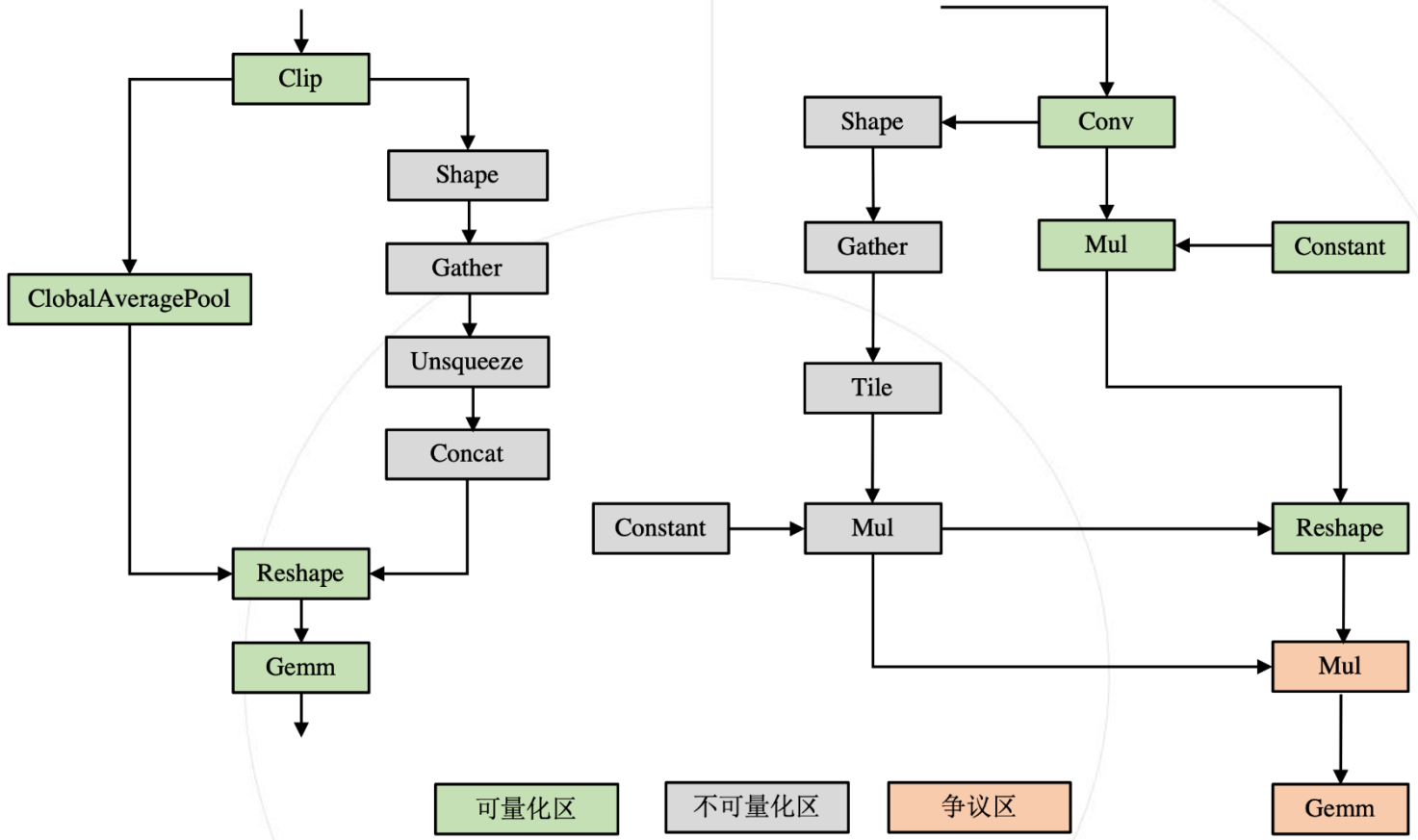
量化工具图调度

图调度主要解决：

- 动态输入模型的shape相关算子调度
- 检测模型的后处理算子调度

Graph Dispatcher将算子分为三类：

- 不可量化区
  - 算子与shape或者index相关
- 可量化区
  - 算子可被量化，conv/gemm等算子
- 争议区
  - 算子输入为不可量化区输出



Graph Dispatcher的三种调度逻辑：

- 激进式调度：量化争议区算子
- 保守式调度：不量化争议区算子
- pplnn调度：只量化conv-conv链路及其相关算子

量化工具图融合

PPQ使用TensorQuantizationConfig类描述数值量化细节

```python
class TensorQuantizationConfig(Serializable):
  def __init__(
      self,
      policy: QuantizationPolicy,
      rounding: RoundingPolicy,
      num_of_bits: int,
      quant_min: int,
      quant_max: int,
      scale: Any,
      offset: Any,
      observer_algorithm: str,
      detail: Any = None,
      inplace: bool = False,
      state: QuantizationStates = QuantizationStates.INITIAL
  ):
      assert num_of_bits <= 32, 'Cannot quantize a tensor with more than 32 bits.'
      assert num_of_bits >= 2, 'Cannot quantize a tensor with less than 2 bits.'

      self._policy = policy
      self._num_of_bits = num_of_bits
      self._scale = scale
      self._offset = offset
      self.state = state
      self._rounding = rounding
      self._quant_min = quant_min
      self._quant_max = quant_max
      self.observer_algorithm = observer_algorithm
      self.inplace = inplace
      self.detail = {} if detail is None else detail
      self._father_config = self # union-find
      self._hash = self.__create_hash()
      super().__init__()
```

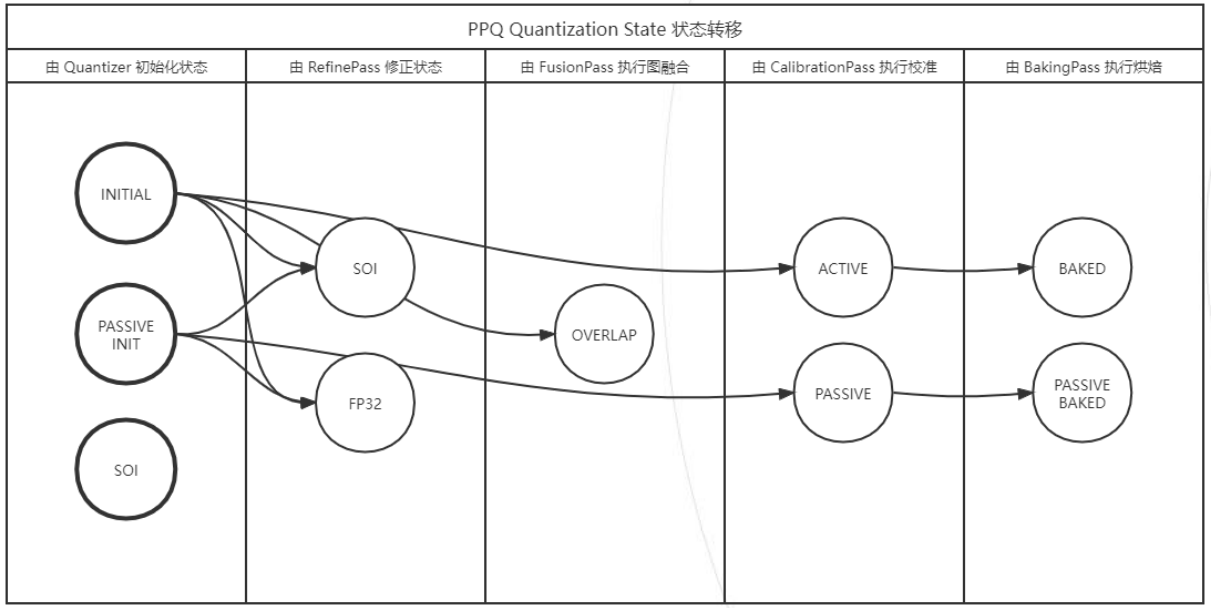
```python
class QuantizationStates(Enum):
    INITIAL      = 1    # 量化参数刚刚被初始化，当前 config 不生效，数据不能被使用
    BAKED        = 2    # 只针对参数量化，表示参数已经被静态量化，当前 config 不生效，数据可以直接使用
    OVERLAPPED   = 3    # 只针对activation量化，表示数据流的量化由其他 config 管理，当前 config 不生效
    JOINT        = 4    # 只针对activation量化，表示当前数据流处于强制联合定点，当前 config 生效
    DEACTIVATED  = 5    # 表示当前 config 不生效
    ACTIVATED    = 6    # 表示当前 config 生效
    DEQUANTIZED  = 7    # 表示当前 config 处于解量化状态，解量化是 PPQ 的一个系统操作
    SOI          = 8    # 表示这一路输入与 Shape or index 相关，不量化
    PASSIVE      = 9    # 表示这一路输入被动量化，如 bias，clip value 等
    PASSIVE_INIT = 10   # 表示这一路输入被动量化，并且刚刚初始化不能被使用
    PASSIVE_BAKED = 11 # 被动量化且静态量化，当前config不生效，数据可以直接使用
    FP32         = 12   # 表示这一路输入直接为FP32浮点数
```

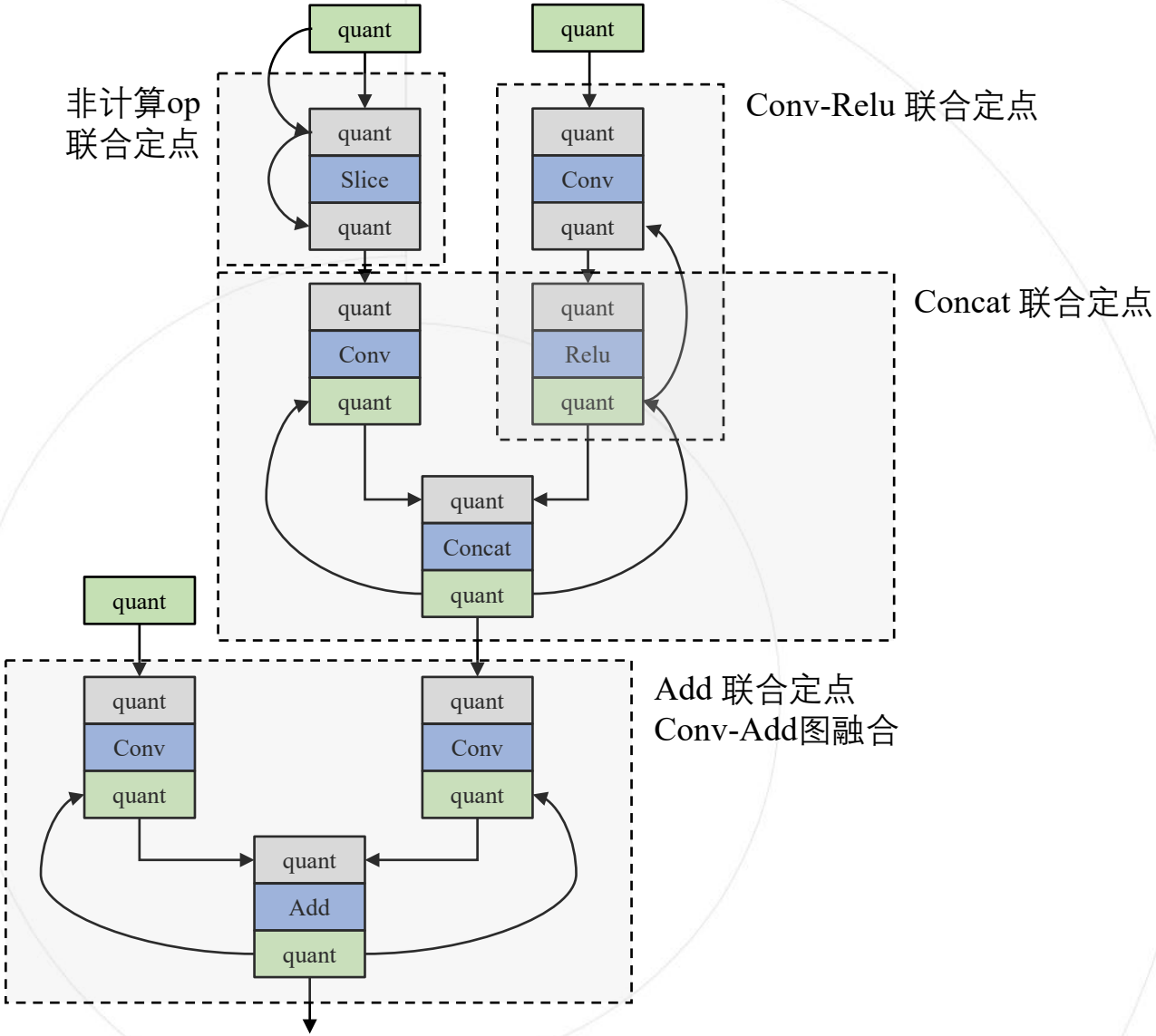
量化工具图融合

Quantizer(量化器)在PPQ中扮演中枢的角色:

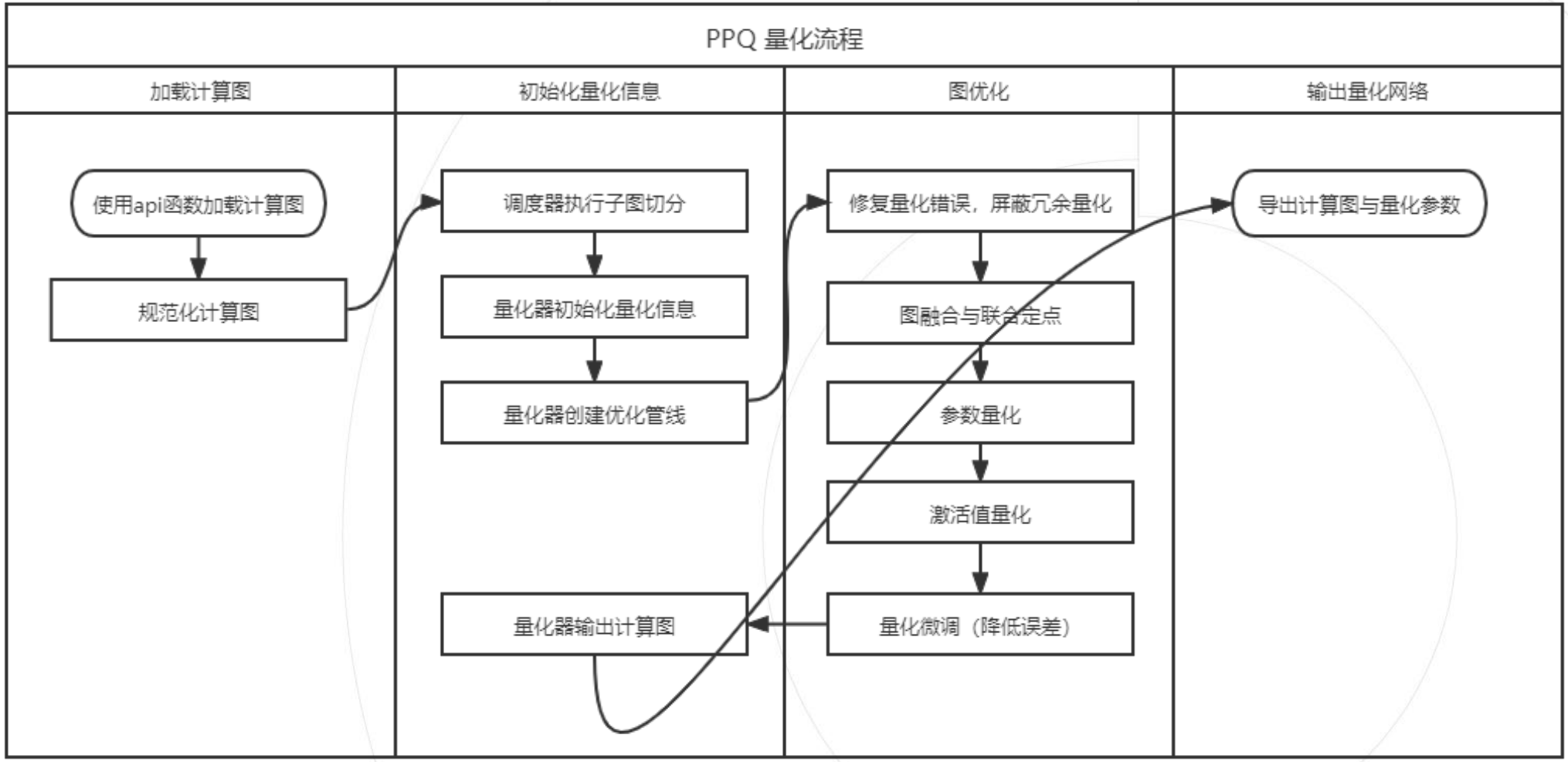
- 执行初始化及细粒度量化op调度
- 执行量化相关的联合定点与图融合
- 执行量化标定及优化

上述优化由Quantization Optimization Pipeline与Quantization Optimization Pass完成。用户可自定义优化Pass，实现新的优化算法。





量化工具流程图


PPQ 量化流程
加载计算图
初始化量化信息
图优化
输出量化网络
使用api函数加载计算图
规范化计算图
调度器执行子图切分
量化器初始化量化信息
量化器创建优化管线
量化器输出计算图
修复量化错误，屏蔽冗余量化
图融合与联合定点
参数量化
激活值量化
量化微调（降低误差）
导出计算图与量化参数

部署平台量化适配

目标平台	量化策略	量化位宽	图融合策略	取整策略	部署平台
PPL_CUDA_INT8, TensorRT	逐通道线性对称量化(参数)，逐层线性对称量化(激活值)	8 bit(weight, activation), 32 bit(bias, bias 执行浮点运算)	Conv(Gemm)-Batchnorm 融合，计算节点与激活节点 融合，Conv - Add 融合，跨越非计算节点联合定点，Concat 联合定点	ROUND_TO_NEAR_EVEN	PPL_CUDA_INT8, TensorRT
NXP_INT8	逐通道线性对称量化(参数，Power-of-2)，逐层线性对称量化(激活值，Power-of-2)	8 bit(weight, activation), 32 bit(bias)	Conv(Gemm)-Batchnorm 融合，计算节点与激活节点 融合，跨越非计算节点联合定点，Concat 联合定点	ROUND_HALF_UP，对于输入使用 ROUND_HALF_DOWN	NXP_INT8
DSP_INT8	逐层线性非对称量化	8 bit(weight, activation), 32 bit(bias)	Conv(Gemm)-Batchnorm 融合，计算节点与激活节点 融合，跨越非计算节点联合定点，Concat 联合定点	ROUND_TO_NEAR_EVEN	DSP_INT8, SNPE

部署平台量化适配(模拟器量化优化)

- 选择基础量化平台，标定方法，完成初步量化，并观察量化输出相似度

```
PPQ Quantization Config Refine Pass Running......PASSED.
PPQ Quantization Fusion Pass Running......PASSED.
PPQ Quantize Point Reduce Pass Running......PASSED.
PPQ Parameter Quantization Pass Running......PASSED.
Calibration Progress(Phase 1): 100%|                                        | 200/200 [00:06<00:00, 32.36it/s]
PPQ Runtime Calibration Pass Running......PASSED.
Inplace Qunantization Setting Pass Running......PASSED.
PPQ Compel Quant Pass(For Add, Sub) Running......PASSED.
PPQ Passive Parameter Quantization Running......PASSED.
PPQ Parameter Baking Pass Running......PASSED.
Analysing Graphwise Quantization Error(Phrase 1):: 100%|                    | 200/200 [00:33<00:00,  5.94it/s]
Analysing Graphwise Quantization Error(Phrase 2):: 100%|                    | 200/200 [00:37<00:00,  5.27it/s]
Layer          | COSINE SIMILARITY
convolution90: |                    | 0.999319
convolution66: |                    | 0.998742
convolution84: |                    | 0.998616
convolution72: |                    | 0.997656
relu0:         |                    | 0.997274
convolution78: |                    | 0.996419
convolution57: |                    | 0.995116
relu40:        |                    | 0.993649
relu62:        |                    | 0.988435
```

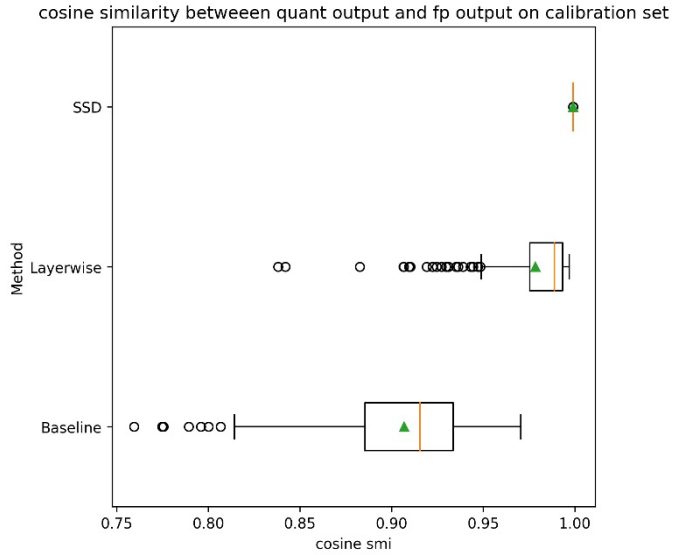
输出onnx模型和quantize.json

```
"quant_info": {
    "data": {
        "bit_width": 8,
        "per_channel": false,
        "quant_flag": true,
        "sym": true,
        "scale": 0.01907323014502432,
        "zero_point": 0,
        "tensor_min": -2.441373458563113,
        "tensor_max": 2.4223002284180883,
        "q_min": -128,
        "q_max": 127,
        "hash": 2679835901,
        "dominator": 2679835901
    },
```

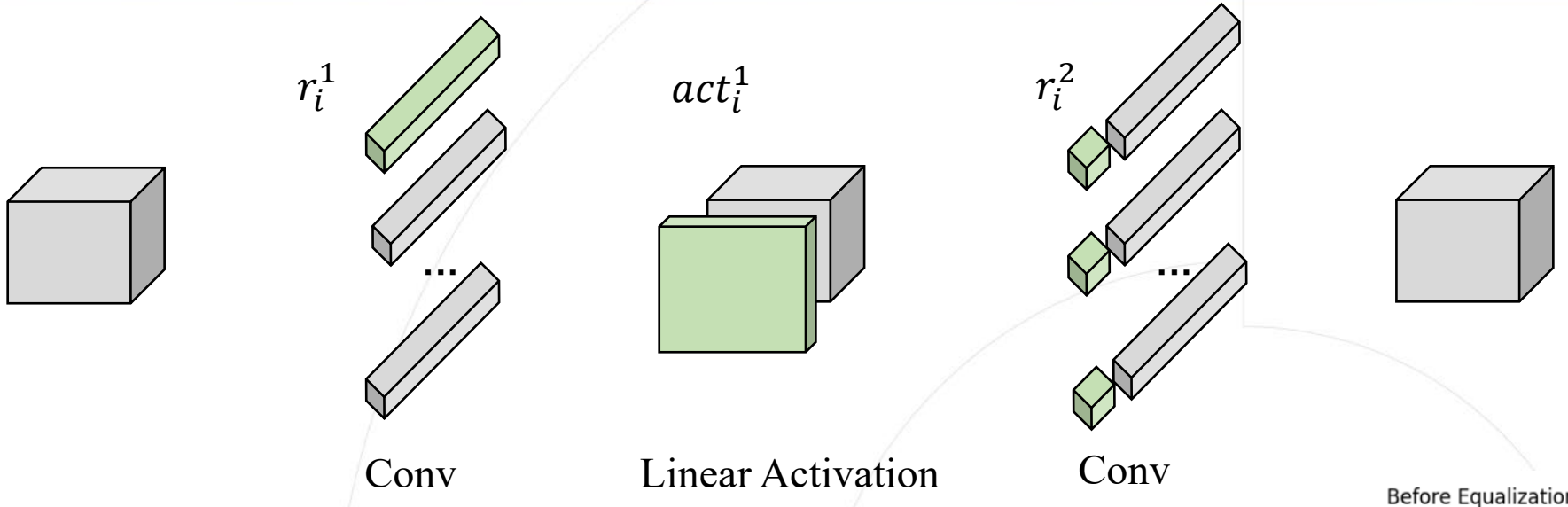
```
"op_info": {
    "resnetv17_conv0_fwd": {
        "data_type": "INT8"
    },
    "resnetv17_relu0_fwd": {
        "data_type": "INT8"
    },
    "resnetv17_pool0_fwd": {
        "data_type": "INT8"
    },
    "resnetv17_stage1_conv0_fwd": {
        "data_type": "INT8"
```

- 选择weight equalize算法，优化weight 异常值 (dfq, ssd)

```
INFO PPQ 2022-03-09 09:24:41 Now Processing Pair 50: convolution81--relu57--convolution84
INFO PPQ 2022-03-09 09:24:41 Collecting Activation Range for Pair...
INFO PPQ 2022-03-09 09:24:43 Collecting Done!
INFO PPQ 2022-03-09 09:24:52 DFQ step, Loss Before Equalization 0.0009678217465989292 || Loss After Equalization 0.0007199036190286279
INFO PPQ 2022-03-09 09:24:57 SSD Algo 1, Loss Before Equalization 0.0009678217465989292 || Loss After Equalization 0.000604239699896425
INFO PPQ 2022-03-09 09:25:01 SSD Algo 2, Loss Before Equalization 0.0009678217465989292 || Loss After Equalization 0.0005933891516178846
INFO PPQ 2022-03-09 09:25:06 SSD Algo 3, Loss Before Equalization 0.0009678217465989292 || Loss After Equalization 0.0007550014997832477
INFO PPQ 2022-03-09 09:25:06 SSD Algo 2 Activated, Loss Before Equalization 0.0009678217465989292 || Loss After Equalization 0.0005933891516178846
```



部署平台量化适配(模拟器量化优化)



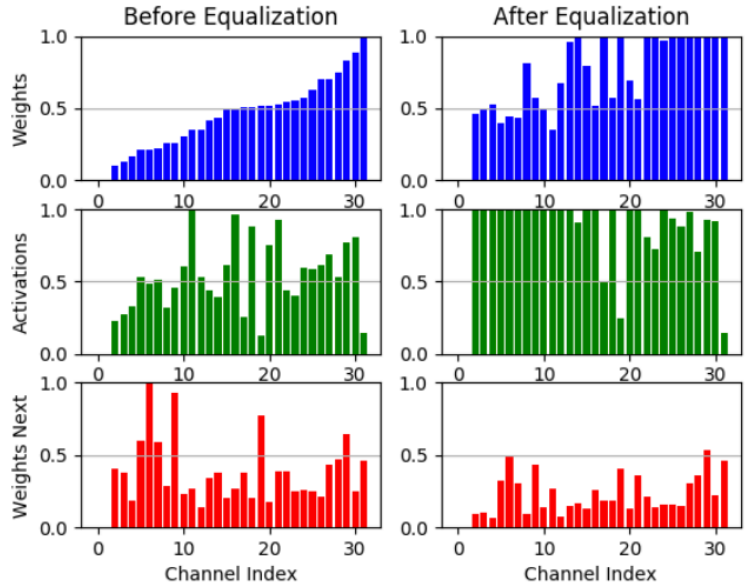
Weight equalize原理:

$$\begin{aligned}\mathbf{y} &= f(\mathbf{W}^{(2)} f(\mathbf{W}^{(1)}\mathbf{x}+\mathbf{b}^{(1)})+\mathbf{b}^{(2)}) \\ &= f(\mathbf{W}^{(2)}\mathbf{S}\hat{f}(\mathbf{S}^{-1}\mathbf{W}^{(1)}\mathbf{x}+\mathbf{S}^{-1}\mathbf{b}^{(1)})+\mathbf{b}^{(2)}) \\ &= f(\widehat{\mathbf{W}}^{(2)}\hat{f}(\widehat{\mathbf{W}}^{(1)}\mathbf{x}+\widehat{\mathbf{b}}^{(1)})+\mathbf{b}^{(2)})\end{aligned}$$

Scale选择:

$$S_{dfq} = \frac{1}{r^2}\sqrt{r^1 r^2}$$

$$S_{ssd} = \min\left(\frac{r^1_{max}}{r^1}, \frac{act^1_{max}}{act^1}, S_{max}\right)$$



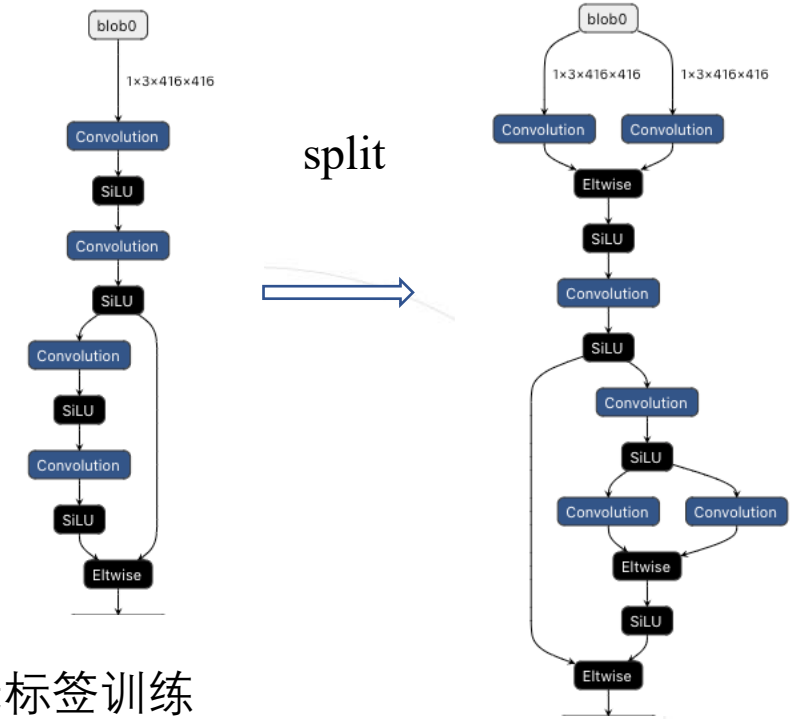
(a) One Step Equalization

部署平台量化适配(模拟器量化优化)

- 选择weight split算法，对量化误差较大层，进行拆分（适用per-layer）

```
INFO PPQ 2022-01-25 10:59:19 Weight split is applied to model_0_conv
INFO PPQ 2022-01-25 10:59:19 weight loss 0.001980482339859009
INFO PPQ 2022-01-25 10:59:32 Weight split loss from 0.001980482339859009 to 0.0006458574533462525
INFO PPQ 2022-01-25 10:59:32 Find split point 0.47999999999999976 for model_0_conv
```

```
INFO PPQ 2022-01-25 10:59:53 Weight split is applied to model_2_cv2_conv
INFO PPQ 2022-01-25 10:59:53 weight loss 0.0012222599983215331
INFO PPQ 2022-01-25 11:00:03 Weight split loss from 0.0012222599983215331 to 0.0004277980327606201 3
INFO PPQ 2022-01-25 11:00:03 Find split point 0.5249999999999998 for model_2_cv2_conv
```



- 选择 advanced optimize(adaquant + bias correction)等算法，进行无标签训练

$$Q_{out} = Quant(\ Quant(w + w_{offset}) * Quant(x) + Quant(bias))$$

$$F_{\text{out}} = w\ * x + bias$$

$$loss = MSE(Q_{out}\ - F_{out}) + \text{Reg}$$

部署平台量化适配(模拟器量化优化)

泰勒展开建模 Task Loss 误差：

$$\mathbb{E}[L(\mathbf{w}+\Delta\mathbf{w})]-\mathbb{E}[L(\mathbf{w})]\approx\Delta\mathbf{w}^{\mathrm{T}}\overline{\mathbf{g}}^{(\mathbf{w})}+\frac{1}{2}\Delta\mathbf{w}^{\mathrm{T}}\overline{\mathbf{H}}^{(\mathbf{w})}\Delta\mathbf{w}$$

为简化海森矩阵，AdaRound 做了如下假设：

- 假设不同层weight相互独立，H^{w} 简化至分块对角矩阵 H^{w^l}

$$\mathbf{H}^{(\mathbf{w}^{(\ell)})}=\mathbb{E}\left[\mathbf{x}^{(\ell-1)}\mathbf{x}^{(\ell-1)^T}\otimes\nabla^2_{\mathbf{z}^{(\ell)}}\mathcal{L}\right]$$

- 假设 $\nabla^2_{\mathrm{z}^{(l)}}L$ 为常数对角矩阵

最终简化至 Local Loss 误差：

$$\underset{\Delta\mathbf{w}^{(\ell)}}{\arg\min}\mathbb{E}\left[\Delta\mathbf{w}^{(\ell)^T}\mathbf{H}^{(\mathbf{w}^{(\ell)})}\Delta\mathbf{w}^{(\ell)}\right]=\underset{\mathbf{V}}{\arg\min}\left\|f_a(\mathbf{W}\mathbf{x})-f_a(\widetilde{\mathbf{W}}\hat{\mathbf{x}})\right\|_F^2+\lambda f_{\mathrm{reg}}(\mathbf{V})$$

缺点：当量化至更低比特时，上述假设并不成立，需设计更合理的优化代理函数(Brecq等)。

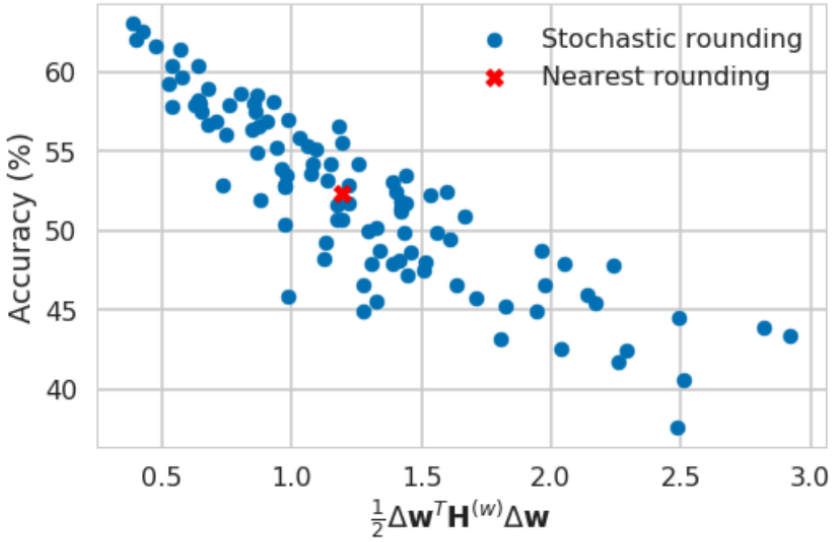
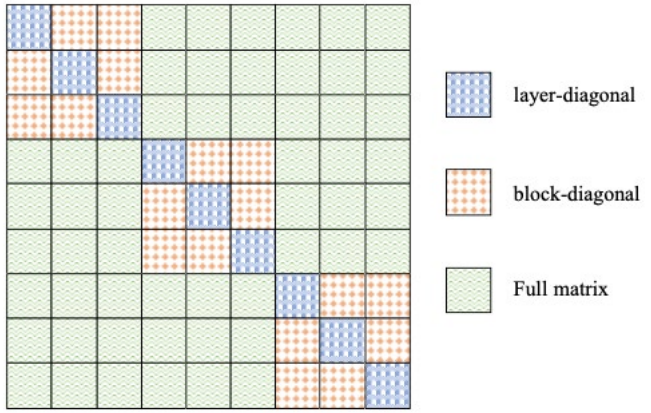


Figure 1. Correlation between the cost in (13) vs ImageNet validation accuracy (%) of 100 stochastic rounding vectors $\hat{\mathbf{w}}$ for 4-bit quantization of only the first layer of Resnet18.



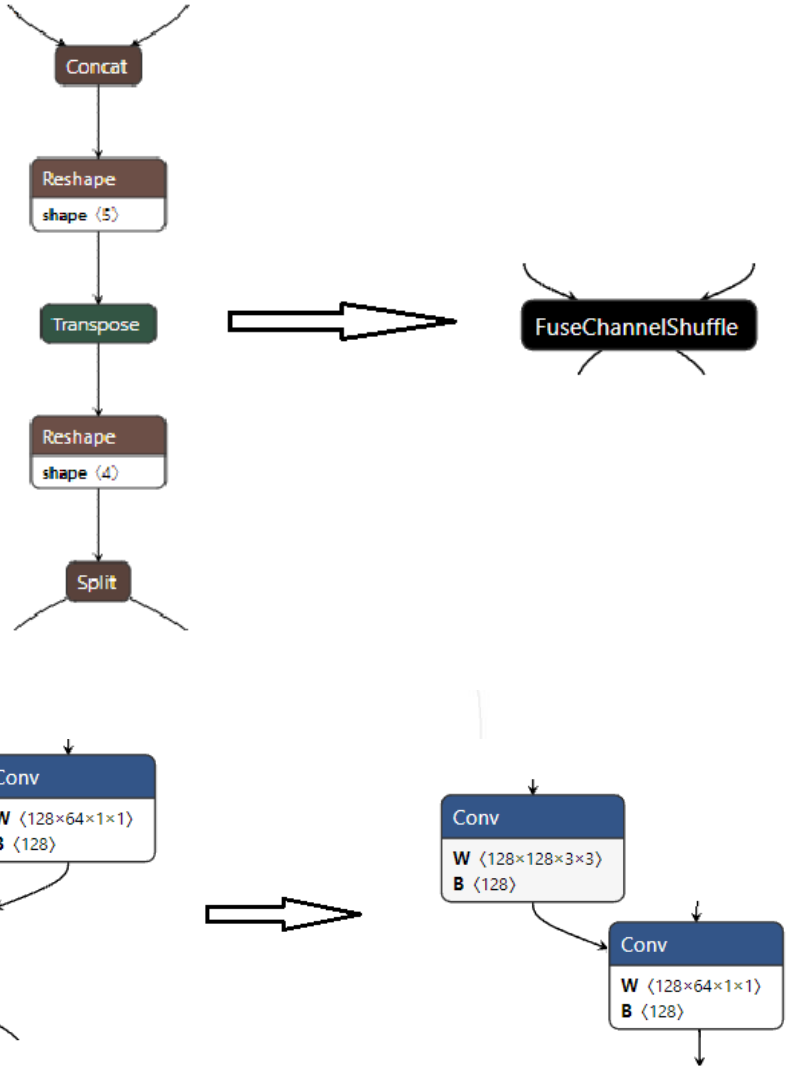
部署平台量化适配(部署平台量化对齐)

部署平台量化对齐:

- 图融合量化点对齐
  - 常见op融合
  - 非计算op联合定点
  - 计算op与内存类op融合
- 非对称per-layer量化平台，推理库微调minmax，对齐zero_point
- Round 方式
- 硬件推理库错误
  - 高通Qnn 1.4, 8w16a量化，Sub算子计算错误
  - NXP Apex, 8w8a 量化，部分case Conv计算错误
  - 高通SNPE 1.43, 部分Conv + Prelu计算错误

量化推理库:

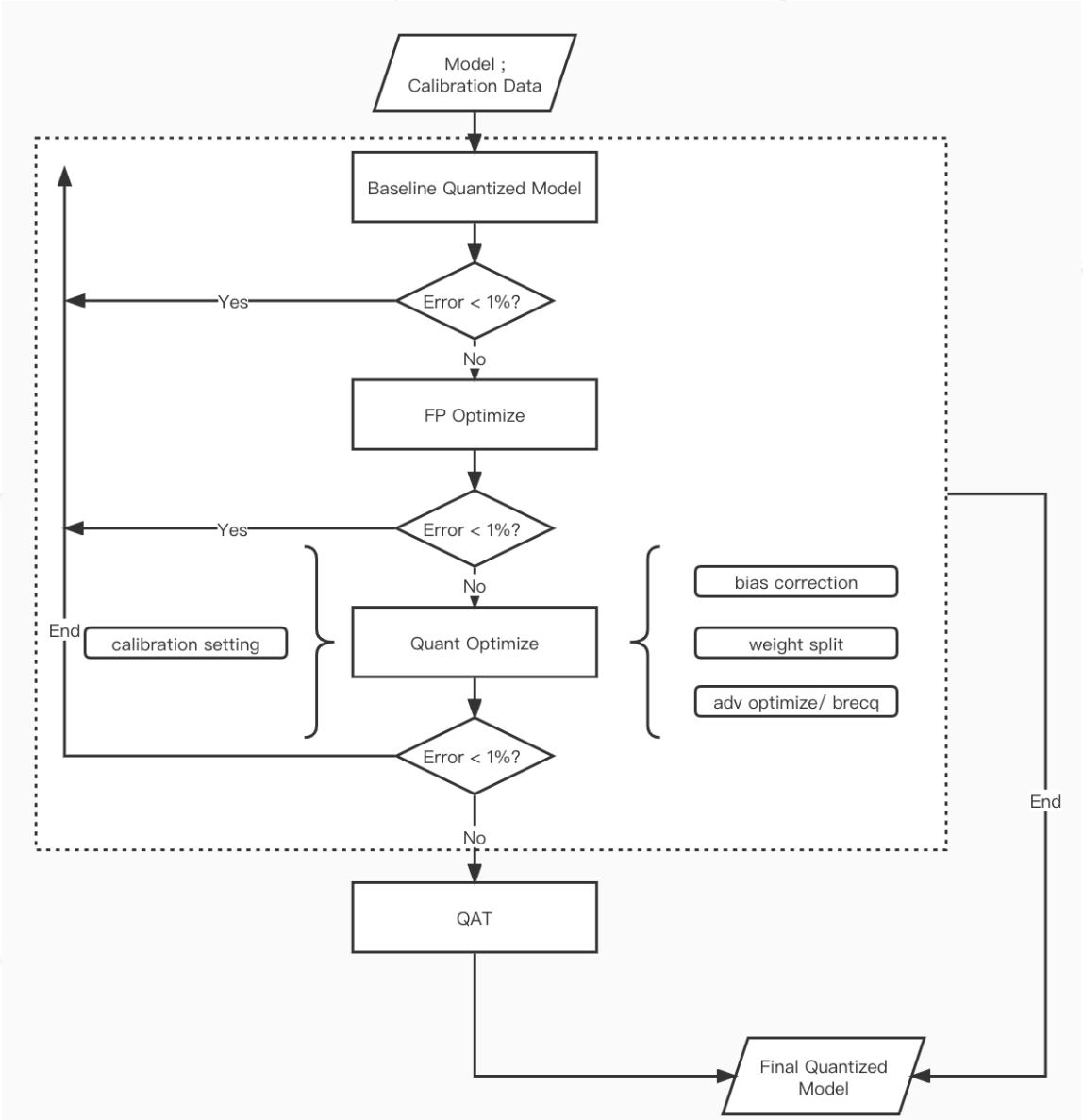
- 量化方案（per-tensor/per-layer, sym, 等等）
- 支持外部写入量化参数
- 支持dump模型中间结果
- 图融合及联合定点策略
- 算子实现细节（非必要）



部署平台量化适配(部署平台量化对齐)

误差衡量标准:
避免搭建各种任务的测试pipeline，我们基于标定集衡量cosine_similarity(target_int8, ppq_int8, fp32)

- 保证模拟器cos(ppq_int8, ppq_fp32) > 0.99
- 若cos(target_int8, ppq_int8) > 0.99, 则模拟器与推理库对齐。否则需重新对齐推理库量化细节
- 若cos(target_int8, fp32) > 0.99, 则量化精度正常
- 若sdk测试精度异常，需排查标定集采样是否均匀及测试pipeline



Benchmark

- 典型业务模型
- 开源模型 (github readme)

模型名称	模型功能	后端	fp32精度	baseline int8精度	ppq optimize 精度
petdet	宠物检测	snpe, ppl3_dsp	0.75 recall	0.711	0.747 (+3.6%)
childdet	婴幼儿检测	snpe	0.853 recall	0.813	0.846 (+3.3%)
verify	人脸识别	snpe, ppl3_dsp	0.984 @1e-6 tpr @fpr	0.801	0.975 (+17.4%)
action	行为识别	snpe, ppl3_dsp, apex	0.931 recall	0.781	0.927 (+14.6%)
yolov3	目标检测	寒武纪	0.66 mAP	None	0.653
denoise	画质降噪	ppl3_dsp	cos(fp, dsp) > 0.99999		人眼判定
retinanet	目标检测	ppl_cuda	0.314 mAP		0.312
……					

Thanks for listening!

Q&A Time

实战训练营

最后一个中奖的机会！

感谢您对商汤学术公开课-OpenPPL系列课程的关注！

请您为课程留下宝贵的意见，本次课程反馈调查仅占用您1分钟时间，您的反馈对我们非常重要哦~

我们将从中抽取1名幸运观众，并赠送商汤小黑羊一只，欢迎大家踊跃反馈~

课程反馈