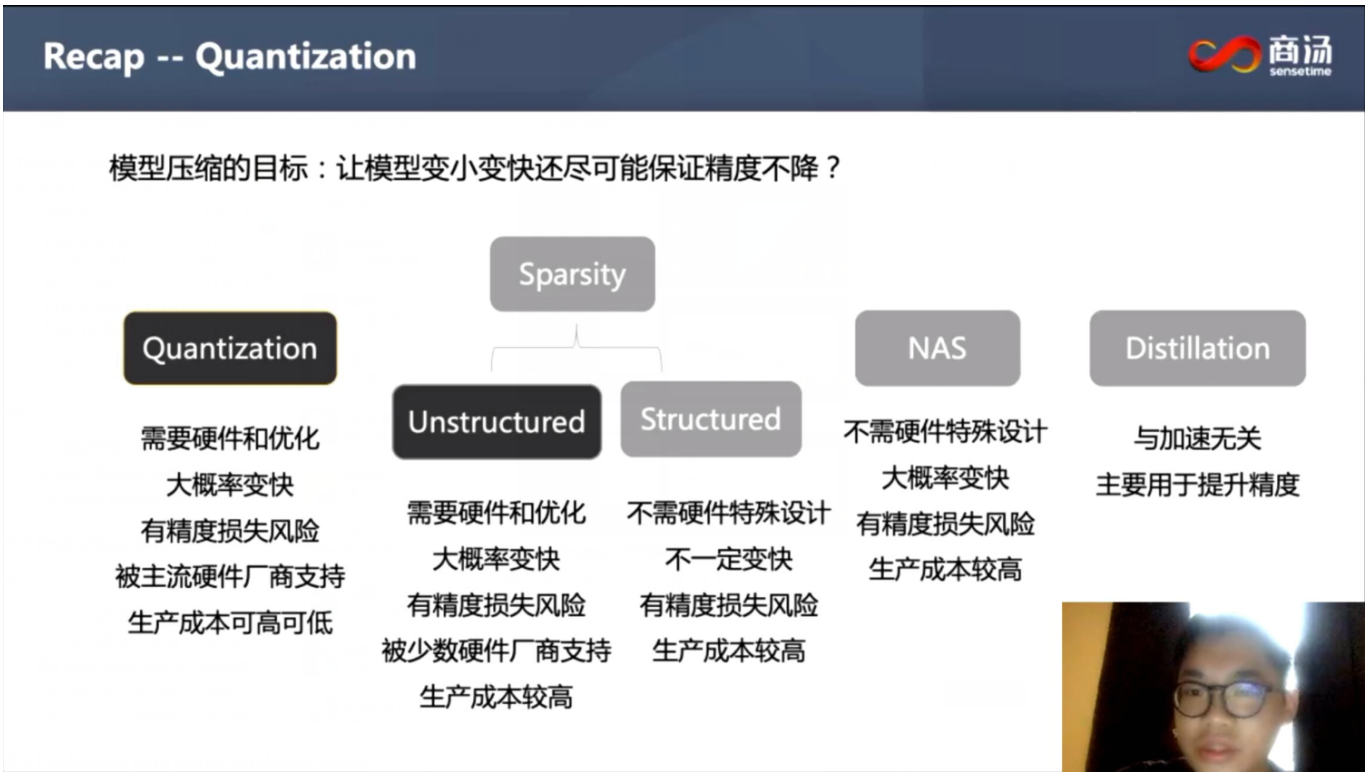
21/9/9商汤QAT talk

2021/9/9

首先是概览：

量化感知训练算法概览

- 主要是一些preliminary和前情提要：



Recap -- Quantization

- **均匀量化**
  - **二值化**
    - xnor + popcount理论峰值比float32高
    - 引入额外的quantizer，可用SIMD方式加速
  - **线性量化(对称、非对称、Ristretto)**
    - arm/x86/nvGPU均支持高效的8-bit计算，TensorCore支持4bit计算
    - 引入额外的quantizer/de-quantizer，可用SIMD方式加速

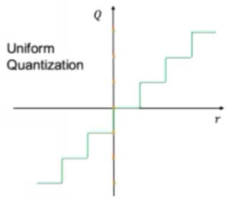
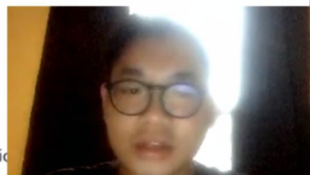


Figure taken from Gholami et al., 2021, A Survey of Quantization Methods for Effi



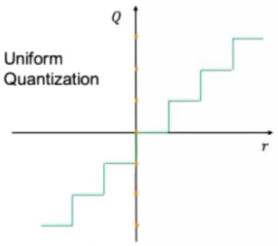
Quantization Function

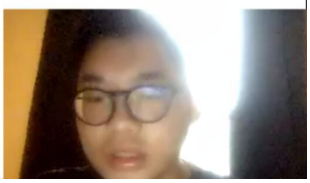
$$\bar{\boldsymbol{w}} = \text{clip}(\left\lfloor \frac{\boldsymbol{w}}{s} \right\rceil + z, N_{min}, N_{max}), \quad \hat{\boldsymbol{w}} = s \cdot (\bar{\boldsymbol{w}} - z)$$

Quantize：将全精度参数w映射到整数

De-Quantize：将整数映射到全精度范围

Backward: 量化后 $\hat{w}$与w 的导数为1





$$\bar{\boldsymbol{w}} = \text{clip}(\left\lfloor \frac{\boldsymbol{w}}{s} \right\rceil + z, N_{min}, N_{max}), \quad \hat{\boldsymbol{w}} = s \cdot (\bar{\boldsymbol{w}} - z)$$

- Symmetric量化： 零点z始终为0，而asymmetric量化中z可以为其他整数
- Per-tensor量化：w矩阵中使用一个步长和零点，perchannel会对每个w_i分配一个步长和零点
- s的格式：通常情况下s是FP32，但是部分硬件会使用2次幂的步长，即s=2^n
- 无符号量化：学术集为了处理ReLU激活值全部非负的情况，将Nmin设置为0，Nmax设置为2^b-1，这样就不会浪费一般的整数范围。然而真实硬件通常不会这样定义，只存在对称和非对称的设置。

- MQBench只讨论均匀量化，非均匀量化需要特殊硬件设计支持，不易部署。
- 后面简单介绍了DoReFa-Net、PACT、QIL、LSQ（sota）四种量化算法，这些算法的学术设置有以下特点：
  - 对A与W都进行量化，首层与最后一层保留精度较高
  - 逐层、对称量化，激活值采用unsigned量化，这是因为经过ReLU激活值都变成正数，但是在硬件部署时不支持这种本质上是**非对称**的量化方式（只支持"绝对"对称量化，既量化区间关于0点对称，学术上对A的4-bit量化实际上要浪费掉符号位，只有3-bit可用）
  - **无法真正部署到硬件上**（和hardware backend不匹配，比如是否为对称量化、是否是均匀量化、是否per-tensor/channel）
  - 算法间采用了**不同的训练设置**，有些算法可能有训练上额外的优势
- DoReFa-Net：
  - Basic quantize operation:

    Forward: $r_o = \frac{1}{2^k - 1} \text{round}((2^k - 1) r_i)$

    Backward: $\frac{\partial c}{\partial r_i} = \frac{\partial c}{\partial r_o}.$
  - Weight quantization:

    Forward: $r_o = f_\omega^k(r_i) = 2\,\text{quantize}_\text{k}(\frac{\tanh(r_i)}{2\max(|\tanh(r_i)|)} + \frac{1}{2}) - 1.$

    Backward: $\frac{\partial c}{\partial r_i} = \frac{\partial r_o}{\partial r_i}\frac{\partial c}{\partial r_o}$ [4]
  - Activation quantization:

    $$f_\alpha^k(r) = \text{quantize}_\text{k}(r).$$

  - 先压缩到[0, 1]范围内再量化W与A（weight quantization/activation quantization就是压缩量化范围）
  - Activation的处理由于经过ReLU(没有负半轴)再Clip到[0, 1]，不需要复杂的变化
  - 取值范围限制在[0, 1]太局限
- PACT：
  - W量化方式与DoReFa一致，考虑到每一层对应activation的最优截断范围不一样，逐channel学最优的截断范围(将activation的范围从[0, 1]拓宽)
  - 没有修改weight量化方式，是其不足

- QIL：

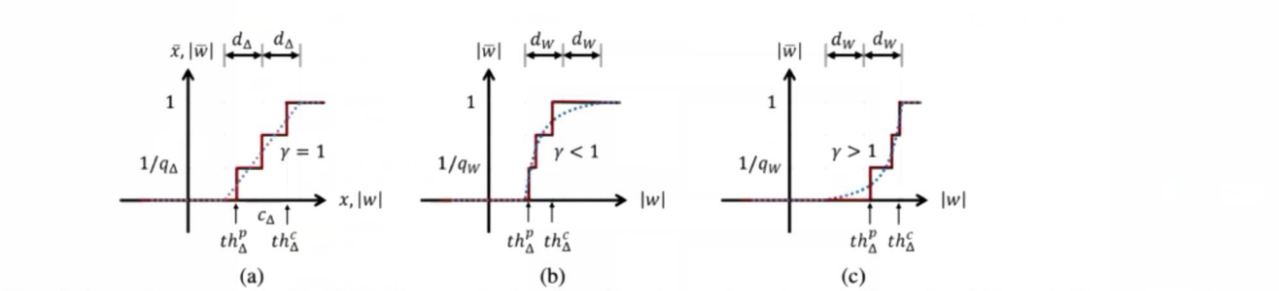
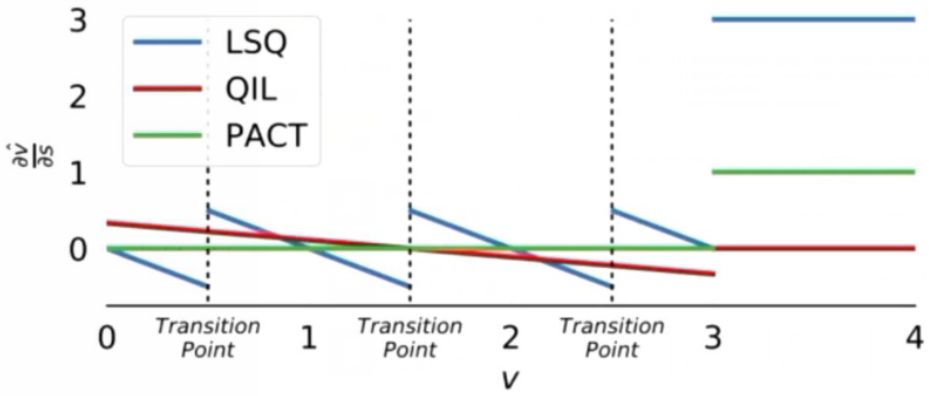


Figure 2. A quantizer as a combination of a transformer and a discretizer with various γ where (a) $\gamma = 1$, (b) $\gamma < 1$, and (c) $\gamma > 1$. The blue dotted lines indicate the transformers, and the red solid lines are their corresponding quantizers. The th^p_Δ and the th^c_Δ represent the pruning and clipping thresholds, respectively.

$$\hat{w} = \begin{cases} 0 & |w| < c_W - d_W \\ \text{sign}(w) & |w| > c_W + d_W \\ (\alpha_W |w| + \beta_W)^\gamma \cdot \text{sign}(w) & otherwise, \end{cases} \qquad \hat{x} = \begin{cases} 0 & x < c_X - d_X \\ 1 & x > \\ \alpha_X x + \beta_X & oth \end{cases}$$

  - 学习C_w(量化区间中心点)和D_w(量化区间半径)，另外根据这两个参数计算\alpha_w和\beta_w（对应斜率和偏移）
  - \gamma在指数位上，表示非均匀量化，实际上学不出来会训崩，复现的时候也没加\gamma，本来意义可能也不大

- LSQ：



$$\bar{v} = \lfloor clip(v/s, -Q_N, Q_P) \rceil, \qquad \hat{v} = \bar{v} \times s.$$

$$\frac{\partial \hat{v}}{\partial s} = \begin{cases} -v/s + \lfloor v/s \rceil & \text{if } -Q_N < v/s < Q_P \\ -Q_N & \text{if } v/s \leq -Q_N \\ Q_P & \text{if } v/s \geq Q_P \end{cases}$$

  - 改变了求scale/stepsize的导数的范围

可部署的量化感知训练

- 总结了学术设置和可部署量化模型之间的区别：
  - **量化函数配置不一致**：学术论文普遍采用per-tensor, symmetric的量化函数，而不同的硬件有自己的量化函数配置；一些量化算法在per-channel的情况下表现不佳；学术量化认为A的量化是无符号的，硬件中只有有符号量化（浪费1bit）
  - **激活值量化插入点不同**：学术量化只考虑量化卷积层，不考虑element-wise加等

- **没折叠BN**

- 硬件后端设置：

$$\bar{\boldsymbol{w}} = \text{clip}(\left\lfloor \frac{\boldsymbol{w}}{s} \right\rceil + z, N_{min}, N_{max}), \quad \hat{\boldsymbol{w}} = s \cdot (\bar{\boldsymbol{w}} - z)$$

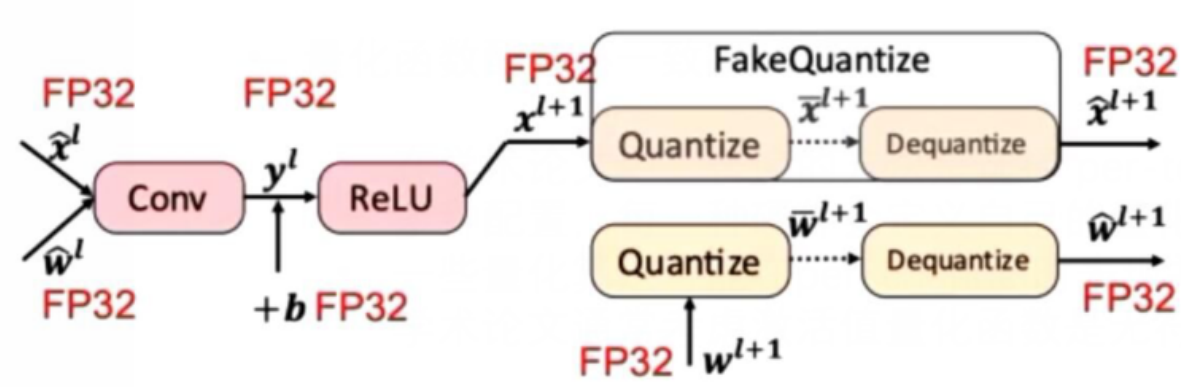
Table 2: Comparison of (1) the different hardware we selected and (2) the different QAT algorithms. *Infer. Lib.* is the inference library; *FBN* means whether fold BN. ❋ means undeployable originally, but can be deployable when certain requirements are satisfied.

Infer. Lib.	Provider	HW Type	Hardware	s Form.	Granularity	Symmetry	Graph	FBN
TensorRT [22]	NVIDIA	GPU	Tesla T4/P4	FP32	Per-channel	Symmetric	2	✓
ACL [23]	HUAWEI	ASIC	Ascend310	FP32	Per-channel	Asymmetric	1	✓
TVM [25]	OctoML	CPU	ARM	POT	Per-tensor	Symmetric	3	✓
SNPE [24]	Qualcomm	DSP	Snapdragon	FP32	Per-tensor	Asymmetric	3	✓
FBGEMM [26]	Facebook	CPU	X86	FP32	Per-channel	Asymmetric	3	✓

  - FBGEMM是服务器上CPU的?
  - TVM POT -> 2次幂 scale -> 级数加法/移位 -> 减少乘法
  - 对称量化更快（没有零点偏移，只是乘）
  - 非对称量化可以改成无符号量化

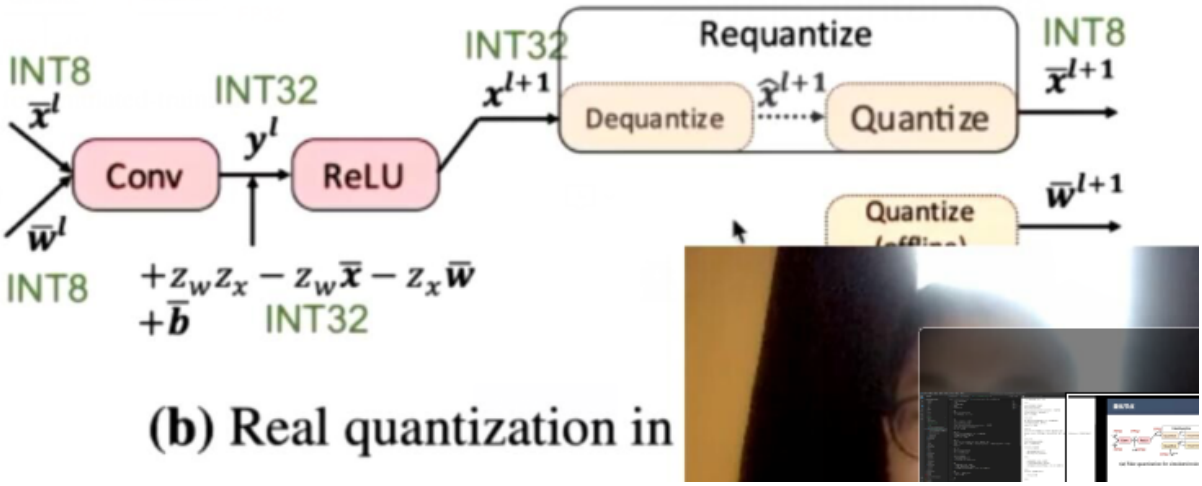
- 模拟量化（软件仿真）和硬件量化计算图：

  - 模拟量化（Fake Quantization）：



(a) Fake quantization for simulated-training.

- 硬件量化：



  - x_hat虽然是FP32，但是是离散取值
  - 真实量化需要重新计算scale所以要先dequantize，但实际上只有requantize一步，不会拆成去量化和量化两步

- 量化节点的区别：

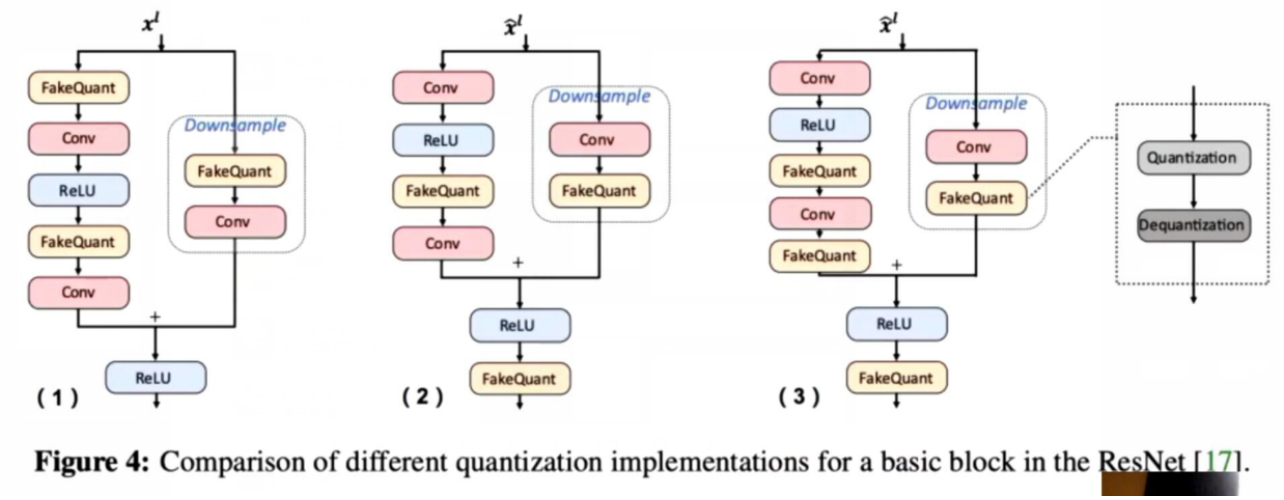


Figure 4: Comparison of different quantization implementations for a basic block in the ResNet [17].

  - （1）是学术设置，只保证参与卷积运算的参数与激活值是定点的，element-wise以全精度计算；这种方法很慢，比如在shortcut和normal path中要独立量化两次，无加速效果；
  - （2）（3）方法区别在于加法前是否量化，（3）更通用
  - tensorRT -> （2）；ACL -> （1）（非通用平台）； 其他的-> （3）

- BN折叠策略的区别：

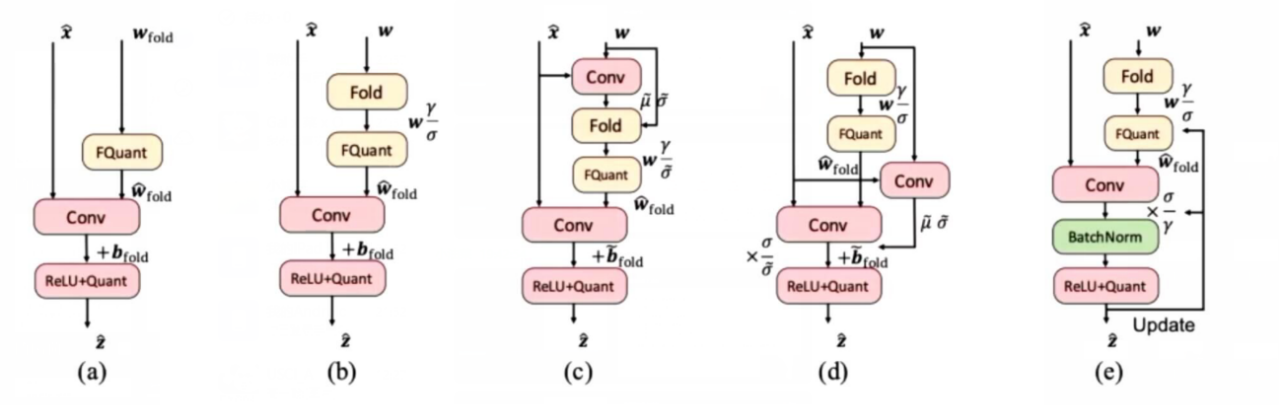
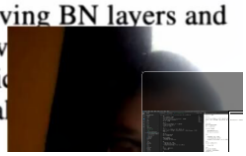


Figure 3: Comparison of different Batch Normalization folding technologies. (a) Removing BN layers and directly update $\boldsymbol{w}_{\text{fold}}$. (b) BN folding without any statistics update. (c) BN folding with tw folding with running statistics and also requires two convolutions. (e) folding running statisti BN layer in training. *Graph (bcde) can be transformed to (a) during inference.* FQuant=Fa



  - （1）直接fold进Conv，调优时没有BN
  - （2）不更新BN统计量，fold之后调优时继续优化affine参数
  - （3）google 两次卷积 第一次计算更新统计量 再折叠进Conv 再量化
  - （4）google 第一次卷积计算移动统计量，fold时统计量抖动较少
  - （5）pytorch官方 反乘折叠系数 推理时可以消掉插入的BN

可复现的量化感知训练

- 部分学术论文无法复现精度的原因有：
  - 没开源
  - 训练配置不一样（预训练的全精度模型训练程度不同、不同lr schedule、不同batch size与训练时长、不同wd、etc.）
  - 软件库发展

模型量化标准集（MQBench）

- MQBench统一配置与预训练模型，消除训练偏差（具体设置不关心）
- 实验结果一（没截到图），统一学术训练设置，4-bit QAT，结果差距非常小，在ResNet-50上效果更明显/5个模型4个最优解 -> **不同算法差距不大**&更好的训练配置导致结果差异；MQBench没有方差分析，有的话差距应该更小。

- 实验结果二，对齐硬件后端的设置，4-bit QAT，算法不分好坏，有好有坏，也没有最坏的，小结论：

Table 7: 4-bit Quantization-Aware Training benchmark on the ImageNet dataset, given different algorithms, hardware inference libraries, and architectures. "NC" means not converged. Red and Green numbers denotes the decrease and increase of the hardware deployable quantization.

Model	Method	Paper Acc.	Academic	TensorRT	ACL	TVM	SNPE	FBGEMM	Avg. HW
ResNet-18 FP: 71.0	LSQ [15]	71.1 / 70.7[1]	**70.7**	69.3(1.4)	70.2(0.5)	67.7(3.0)	**69.7(1.0)**	**69.8**(0.9)	69.3±0.87
	DSQ [28]	69.6	70.0	66.9(3.1)	69.7(0.3)	67.1(2.9)	68.9(1.1)	68.9(1.1)	68.3±1.10
	PACT [30]	69.2	70.5	69.1(1.4)	**70.4(0.1)**	57.5(13.0)	69.3(1.2)	69.7(**0.8**)	67.2±4.87
	DoReFa [31]	68.1[2]	**70.7**	**69.6(1.1)**	**70.4**(0.3)	**68.2(2.5)**	68.9(1.8)	69.7(1.0)	**69.4**±0.75
ResNet-50 FP: 77.0	LSQ [15]	76.7	**77.4**	**76.3**(1.1)	**76.5**(0.9)	**75.9(1.5)**	**76.2**(1.2)	76.4(1.0)	**76.3**±0.21
	DSQ [28]	N/A	76.4	74.8(1.6)	76.2(0.2)	74.4(2.0)	75.9(**0.5**)	76.0(0.4)	75.5±0.72
	PACT [30]	76.5	76.3	**76.3(0.0)**	76.1(0.2)	NC	NC	**76.6(0.3)**	45.8±37.4
	DoReFa [31]	71.4[2]	76.4	76.2(0.2)	76.3(**0.1**)	NC	NC	75.9(0.5)	45.7±37.3
MobileNetV2 FP: 72.6	LSQ [15]	66.3[3]	70.6	66.1(4.5)	68.1(2.5)	**64.5(6.1)**	**66.3(4.3)**	65.5(5.1)	**66.1**±1.18
	DSQ [28]	64.8	69.6	48.4(21.2)	68.3(1.3)	29.4(39.8)	41.3(28.3)	50.7(18.9)	47.6±12.7
	PACT [30]	61.4[4]	**70.7**	**66.5(4.2)**	**70.3(0.4)**	48.1(22.6)	60.3(10.4)	**66.5(4.2)**	62.3±7.8
	DoReFa [31]	N/A	NC	NC	NC	NC	NC	NC	0±0
EfficientNet-Lite0 FP: 75.3	LSQ [15]	N/A	72.6	67.0(5.6)	65.5(7.1)	**65.0(7.6)**	**68.6(4.0)**	66.9(6.7)	**66.6**±1.27
	DSQ [28]	N/A	72.6	35.1(37.5)	69.6(3.0)	NC	7.5(65.1)	45.9(26.7)	31.6±25.5
	PACT [30]	N/A	**73.0**	**68.2(4.8)**	**72.6(0.4)**	45.9(27.1)	56.5(16.5)	**69.0(4.0)**	62.4±9.88
	DoReFa [31]	N/A	NC	NC	NC	NC	NC	NC	0±
RegNetX-600MF FP: 73.7	LSQ [15]	N/A	72.7	**72.5(0.2)**	72.8(0.1)	**70.0(2.7)**	**72.5(0.2)**	**72.5**(0.2)	**72**
	DSQ [28]	N/A	71.7	68.6(2.1)	71.4(0.3)	64.5(7.2)	70.0(1.7)	70.0(1.7)	68
	PACT [30]	N/A	72.2	72.0(**0.2**)	**73.3(1.1)**	NC	NC	**72.5(0.3)**	43
	DoReFa [31]	N/A	**72.9**	72.4(0.5)	73.2(0.3)	NC	NC	72.2(0.7)	43

[1,2,3,4] Accuracy reported in [30, 42, 43, 44], respectively.

  - PACT->对per channel友好
  - LSQ -> per tensor友好 但是在per channel中不太好
  - DoReFa也不差，训崩了可能是[0, 1]范围的问题 -> rule-based与learning based差不多
  - 对齐硬件之后和学术设置下结果差距很大，基本上会掉点
- 实验结果三，BN折叠：

Table 5: Comparison of the accuracy on a 4-bit quantized ResNet-18 and MobileNetV2, using LSQ [15] and PACT [30], given different folding strategies ("-1" denotes normal BN training without folding, others are folding strategies introduced in Sec. 3.2.); "NC" denotes Not Converged; "*" denotes asynchronous statistics.

Model	ResNet-18							MobileNetV2						
Folding Strategy	-1	0	1	2	3	4	4*	-1	0	1	2	3	4	4*
LSQ	70.7	69.8	70.1	70.2	70.3	**70.4**	70.1	70.6	69.5	69.9	70.0	**70.1**	**70.1**	64.8
PACT	70.5	NC	NC	NC	NC	**67.8**	65.5	70.7	NC	NC	NC	NC	**60.8**	NC

  - BN折叠会影响量化算法的效果
  - 右上角带*表示异步统计量，即不同步分布式训练中BN层的统计量 -> 显著影响性能，因为BN会fold进Conv层，不同步会导致weight不同

Q&A

- LSQ/LSQ+按照论文里不能部署，需要手动调整
- 硬件量化集成为requantize，而不会有dequantize和quantize
- swish/hardswish实际上不好用，ReLU好用
- 学术上几乎没有做检测分割的，baseline比较少，任务也挺难
- github还会更新，以及doc
- softmax的量化在NLP中有， Q-bert有量化
- NLP transformer可以部署，CV transformer量化做的人还少