Trained Quantization Thresholds (TQT)

for Accurate and Efficient Fixed-Point Inference of Deep Neural Networks

Sambhav Jain[^*], Albert Gural[#*], Michael Wu[^], Chris Dick[^]

[^]Xilinx Inc., [#]Stanford University (*equal contribution)

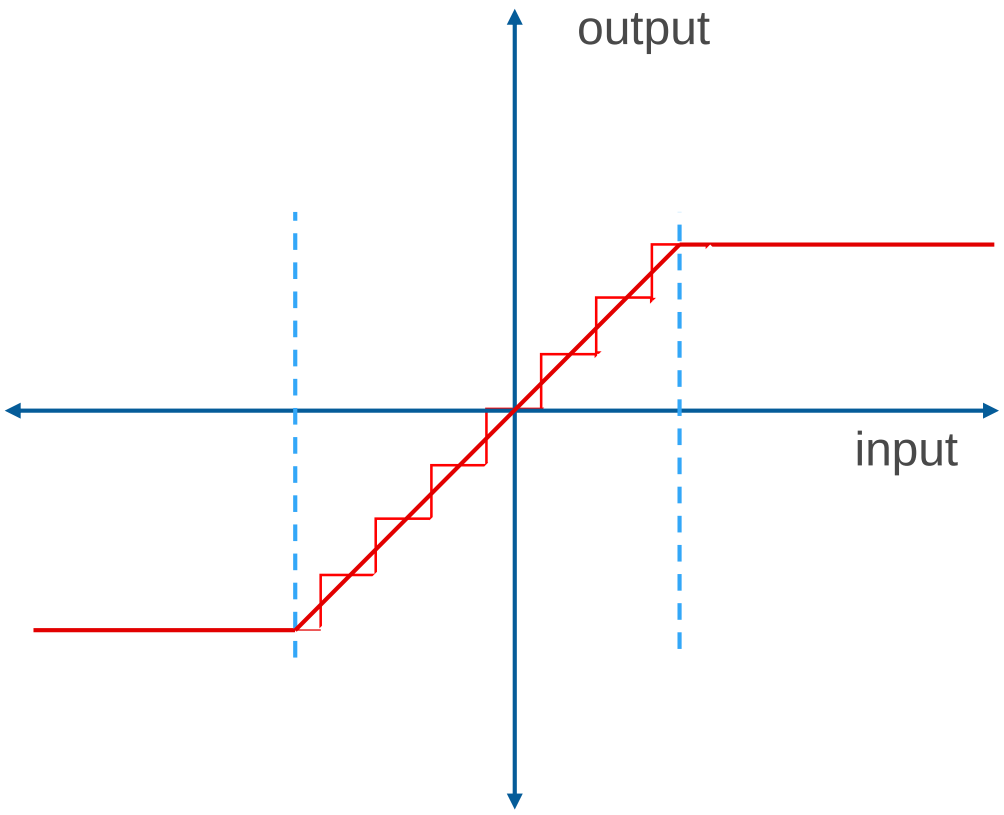
March 3, 2020

© Copyright 2020 Xilinx

Background & Motivation

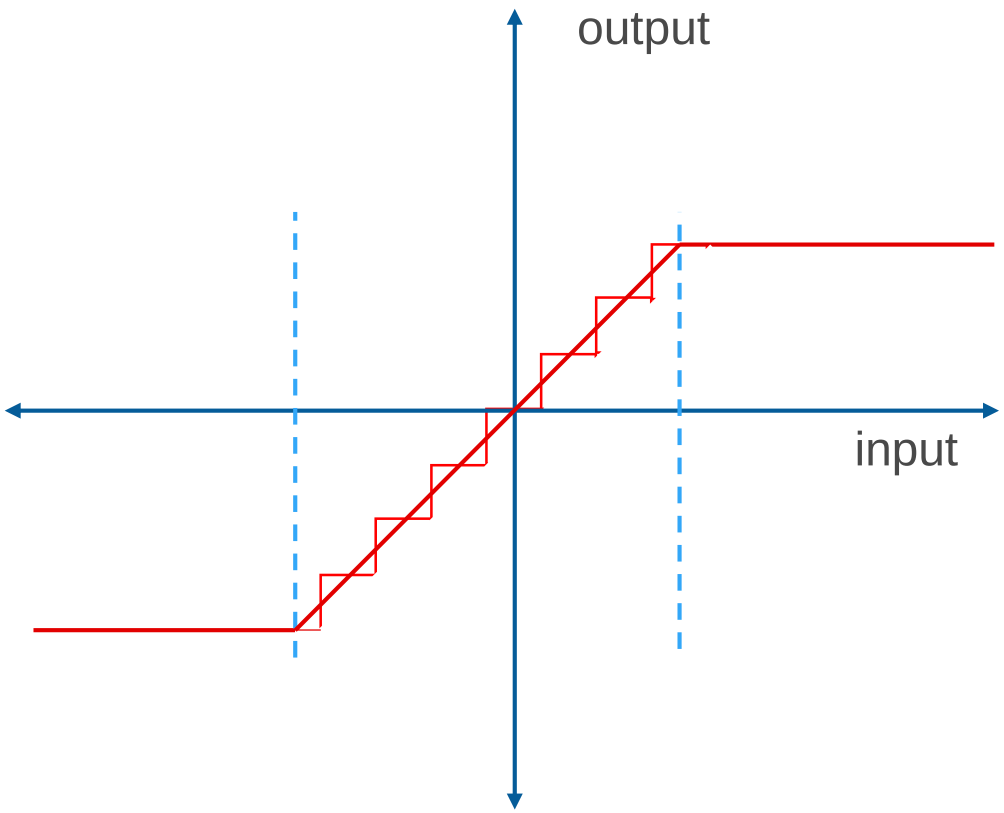
© Copyright 2020 Xilinx

Uniform Quantization



© Copyright 2020 Xilinx

Uniform Quantization



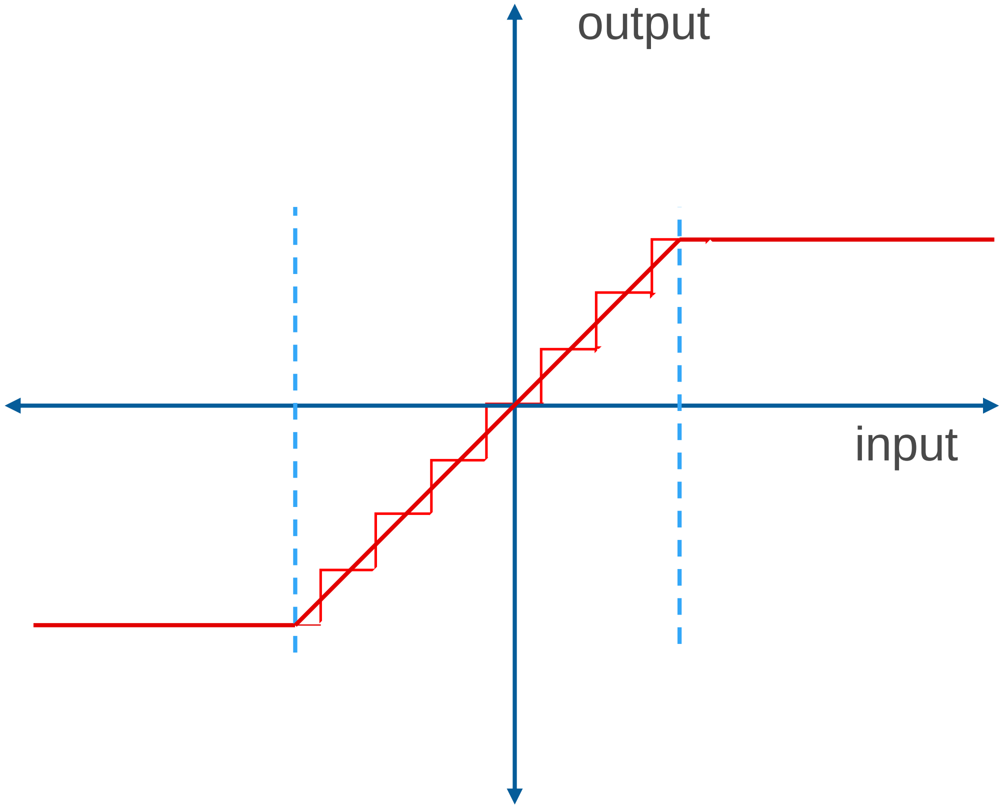
Affine Mapping

$$r = s \cdot (q - z)$$

real domain quantized domain

© Copyright 2020 Xilinx

Uniform Quantization



Affine Mapping

quantization parameters

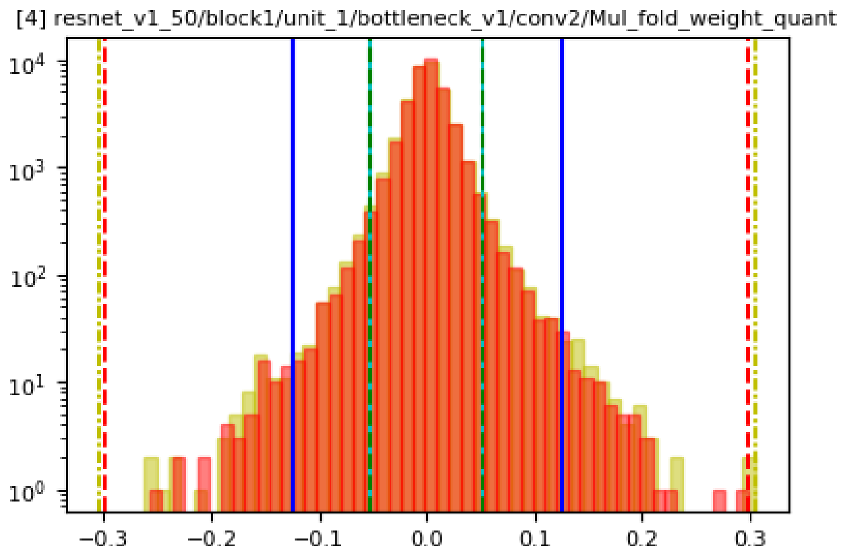
$$r = s \cdot (q - z)$$

real domain

quantized domain

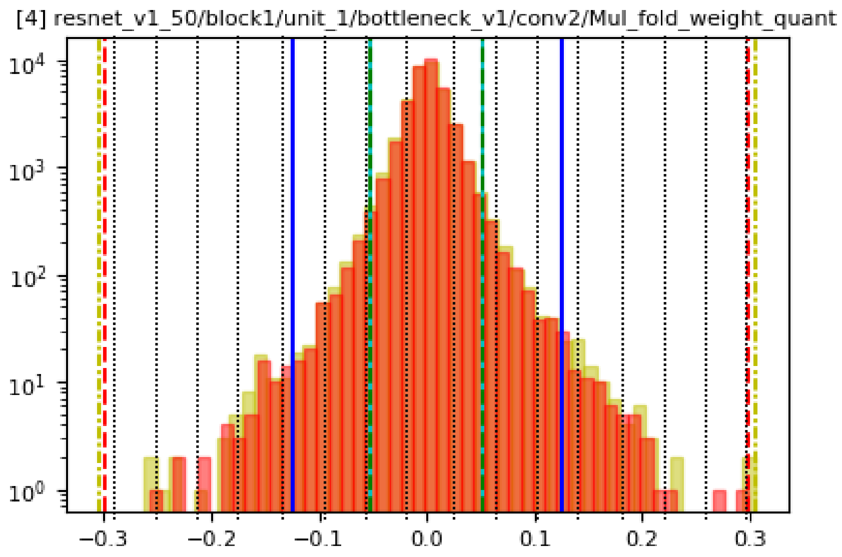
© Copyright 2020 Xilinx

Threshold Selection



© Copyright 2020 Xilinx

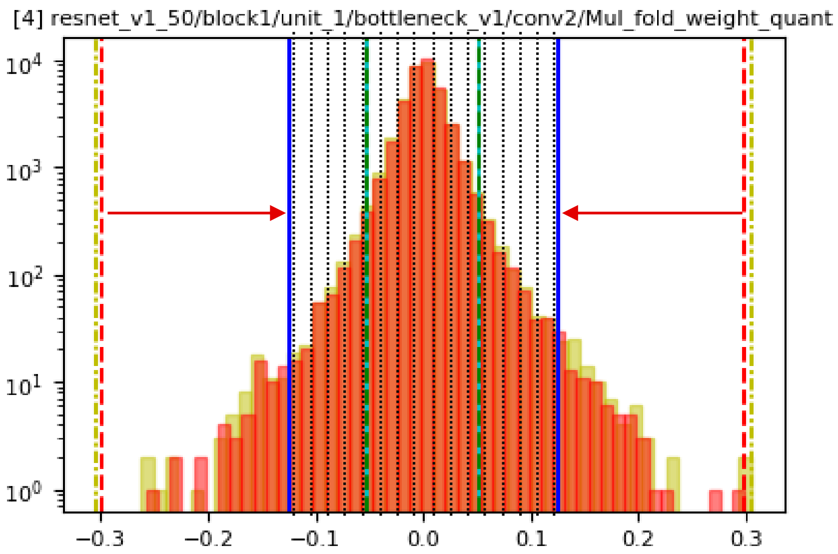
Threshold Selection



- Say 4-bit quantizer
  - 16 quantization levels
  - Clipping thresholds: dotted red lines (min, max)
  - Poor utilization of available precision

© Copyright 2020 Xilinx

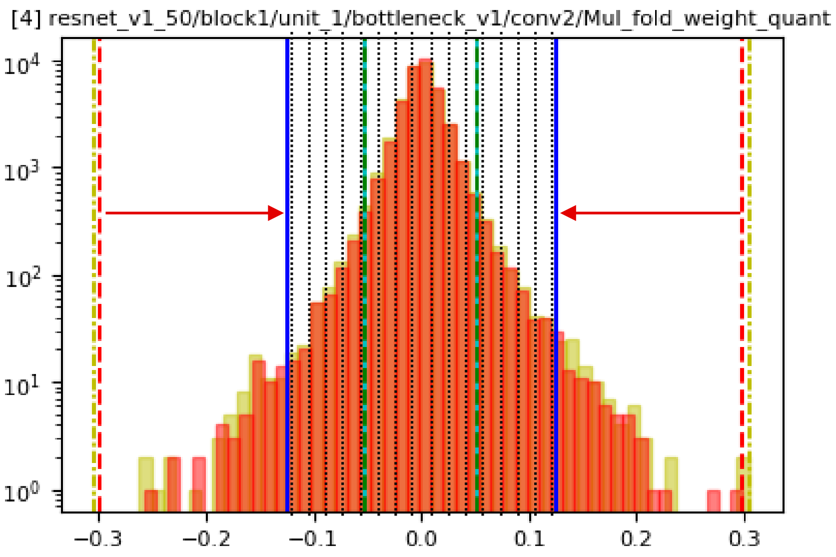
Threshold Selection



- Say 4-bit quantizer
  - 16 quantization levels
  - Clipping thresholds: blue lines
  - Better utilization of available precision

© Copyright 2020 Xilinx

Threshold Selection



1 Statistical Methods

- Calibration
- KL divergence minimization
- SQNR maximization
- Percentile / nSD initialization
- …

2 Gradient Descent Methods

- Google's QAT (Jacob et al., 2017)
- IBM's PACT (Choi et al., 2018)
- Xilinx's TQT (Jain et al., 2019)
- …

© Copyright 2020 Xilinx

Quantizer “Degrees of Freedom”

General case

Special case

Hardware Friendliness

Accuracy (Easy Network)

Accuracy (Hard Network)

© Copyright 2020 Xilinx

Quantizer "Degrees of Freedom"

General case

$$r = s \cdot (q - z)$$

asymmetric ⟷ symmetric

Special case

$$r = s \cdot q$$

Hardware Friendliness

Accuracy (Easy Network)

Accuracy (Hard Network)

© Copyright 2020 Xilinx

XILINX

Quantizer "Degrees of Freedom"

General case			Special case
$r = s \cdot (q - z)$	asymmetric	symmetric	$r = s \cdot q$
$s, z \in \mathbb{R}^c$	per-channel	per-tensor	$s, z \in \mathbb{R}$

Hardware Friendliness

Accuracy (Easy Network)

Accuracy (Hard Network)

© Copyright 2020 Xilinx

Quantizer “Degrees of Freedom”

General case			Special case
$r = s \cdot (q - z)$	asymmetric	symmetric	$r = s \cdot q$
$s, z \in \mathbb{R}^c$	per-channel	per-tensor	$s, z \in \mathbb{R}$
$s \in \mathbb{R}^+$	real-valued scaling	power-of-2 scaling	$s = 2^{-f}$

Hardware Friendliness

Accuracy (Easy Network)

Accuracy (Hard Network)

© Copyright 2020 Xilinx

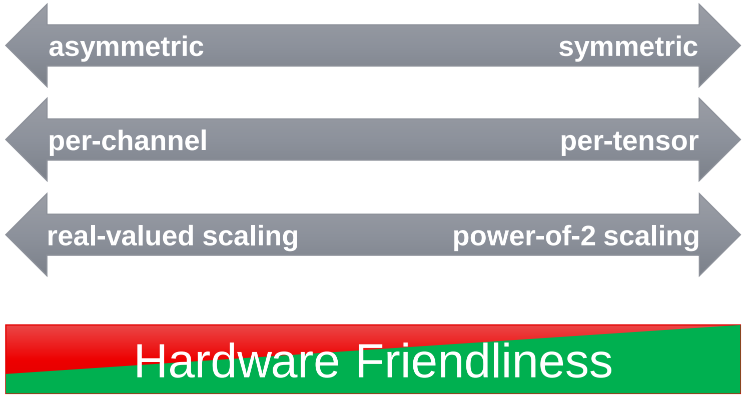
MobileNets are hard to quantize

real-valued scaling

Network	Asymmetric, per-layer (Post Training Quantization)	Symmetric , per-channel (Post Training Quantization)	Asymmetric, per-layer (Quantization Aware Training)	Symmetric, per-channel (Quantization Aware Training)	Floating Point
Mobilenet-v1_1_224	0.001	0.591	0.70	0.707	0.709
Mobilenet-v2_1_224	0.001	0.698	0.709	0.711	0.719
Nasnet-Mobile	0.722	0.721	0.73	0.73	0.74
Mobilenet-v2_1.4_224	0.004	0.74	0.735	0.745	0.749
Inception-v3	0.78	0.78	0.78	0.78	0.78
Resnet-v1_50	0.75	0.751	0.75	0.75	0.752
Resnet-v2_50	0.75	0.75	0.75	0.75	0.756
Resnet-v1_152	0.766	0.762	0.765	0.762	0.768
Resnet-v2_152	0.761	0.76	0.76	0.76	0.778

(Krishnamoorthi, 2018)

Degrees of Freedom



© Copyright 2020 Xilinx

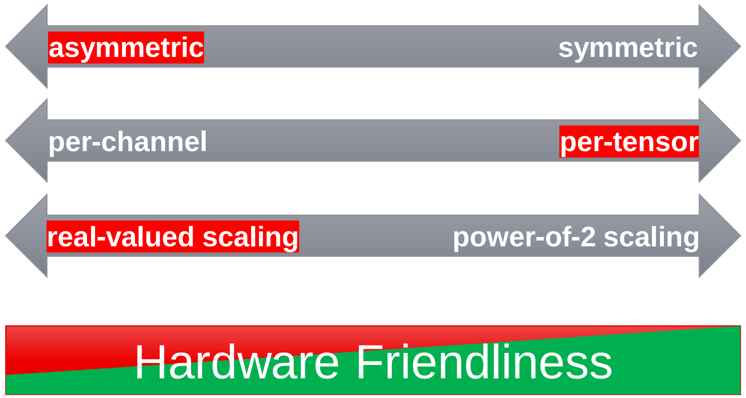
MobileNets are hard to quantize

real-valued scaling

Network	Asymmetric, per-layer (Post Training Quantization)	Symmetric , per-channel (Post Training Quantization)	Asymmetric, per-layer (Quantiza-tion Aware Training)	Symmetric, per-channel (Quantiza-tion Aware Training)	Floating Point
Mobilenet-v1_1_224	0.001	0.591	0.70	0.707	0.709
Mobilenet-v2_1_224	0.001	0.698	0.709	0.711	0.719
Nasnet-Mobile	0.722	0.721	0.73	0.73	0.74
Mobilenet-v2_1.4_224	0.004	0.74	0.735	0.745	0.749
Inception-v3	0.78	0.78	0.78	0.78	0.78
Resnet-v1_50	0.75	0.751	0.75	0.75	0.752
Resnet-v2_50	0.75	0.75	0.75	0.75	0.756
Resnet-v1_152	0.766	0.762	0.765	0.762	0.768
Resnet-v2_152	0.761	0.76	0.76	0.76	0.778

(Krishnamoorthi, 2018)

Degrees of Freedom



© Copyright 2020 Xilinx

MobileNets are hard to quantize

real-valued scaling

Network	Asymmetric, per-layer (Post Training Quantization)	Symmetric , per-channel (Post Training Quantization)	Asymmetric, per-layer (Quantization Aware Training)	Symmetric, per-channel (Quantization Aware Training)	Floating Point
Mobilenet-v1_1_224	0.001	0.591	0.70	0.707	0.709
Mobilenet-v2_1_224	0.001	0.698	0.709	0.711	0.719
Nasnet-Mobile	0.722	0.721	0.73	0.73	0.74
Mobilenet-v2_1.4_224	0.004	0.74	0.735	0.745	0.749
Inception-v3	0.78	0.78	0.78	0.78	0.78
Resnet-v1_50	0.75	0.751	0.75	0.75	0.752
Resnet-v2_50	0.75	0.75	0.75	0.75	0.756
Resnet-v1_152	0.766	0.762	0.765	0.762	0.768
Resnet-v2_152	0.761	0.76	0.76	0.76	0.778

(Krishnamoorthi, 2018)

Degrees of Freedom

asymmetric — symmetric

per-channel — per-tensor

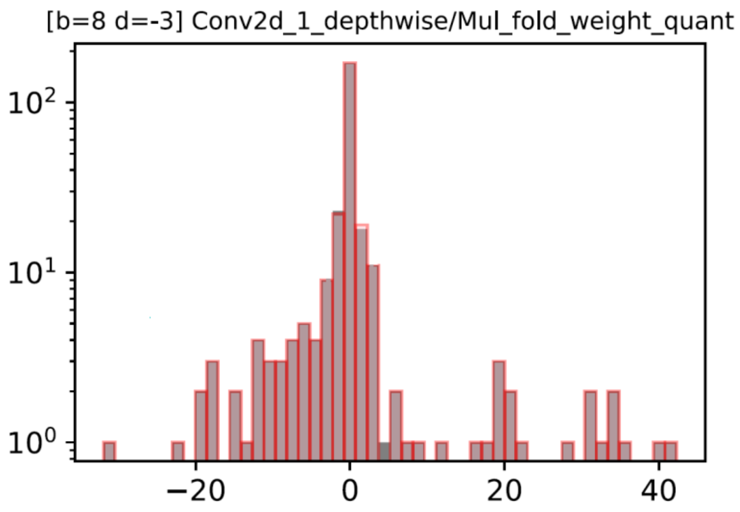
real-valued scaling — power-of-2 scaling

Hardware Friendliness

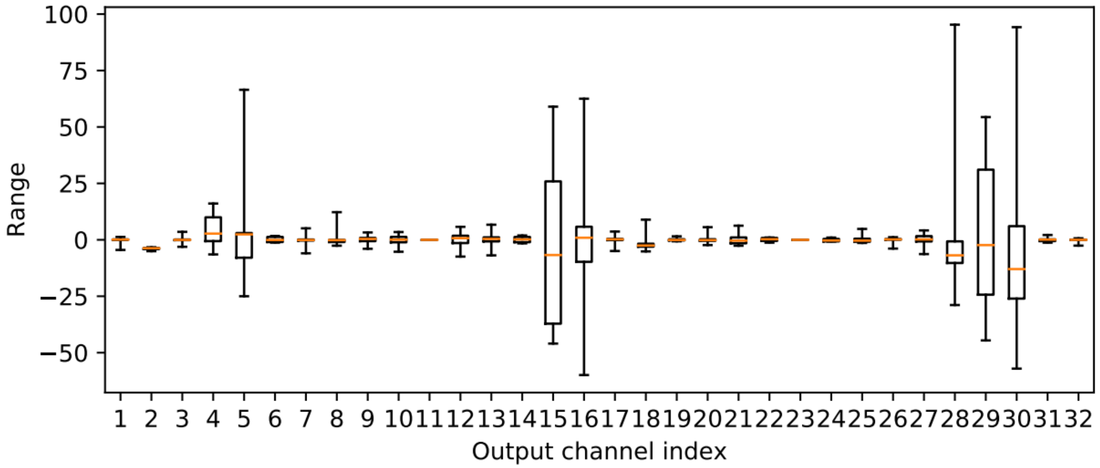
© Copyright 2020 Xilinx

XILINX

MobileNets are hard to quantize – Why?



Weight distribution in first depthwise separable layer of MobileNet v1



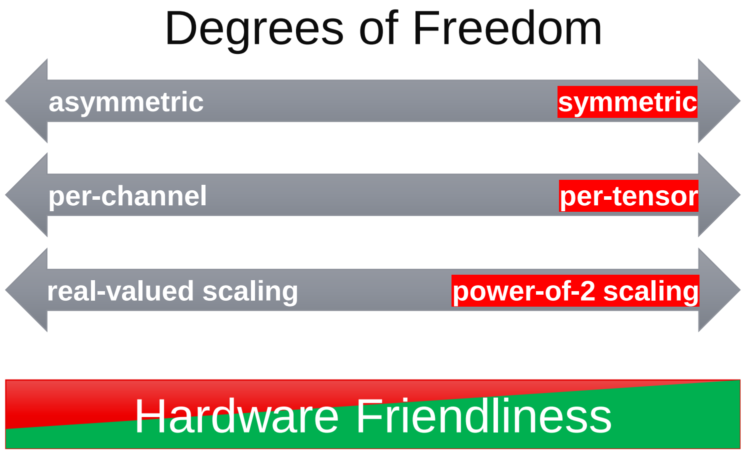
Dynamic range of weights (per-channel) in first depthwise separable layer of MobileNet v2 (Nagel et al., 2019)

© Copyright 2020 Xilinx

With TQT: MobileNets can be quantized well

ours

Method	Precision	Quantization Scheme	Top-1
MobileNet v1 1.0 224			
QAT	FP32		70.9
	INT8	per-channel, symmetric, real scaling	70.7
	INT8	per-tensor, asymmetric, real scaling	70.0
TQT	FP32		71.1
	INT8	per-tensor, symmetric, p-of-2 scaling	**71.1**
MobileNet v2 1.0 224			
QAT	FP32		71.9
	INT8	per-channel, symmetric, real scaling	71.1
	INT8	per-tensor, asymmetric, real scaling	70.9
TQT	FP32		71.7
	INT8	per-tensor, symmetric, p-of-2 scaling	**71.8**



© Copyright 2020 Xilinx

Trained Quantization Thresholds

© Copyright 2020 Xilinx

Implementation

Forward Pass

$$q(x;s) := \text{clip}\left(\left\lfloor \frac{x}{s} \right\rceil; n, p\right) \cdot s$$

where $n = -2^{b-1}$, $p = 2^{b-1} - 1$ and $s = \frac{2^{\lceil \log_2 t \rceil}}{2^{b-1}}$ for signed data; $n = 0$, $p = 2^b - 1$ and $s = \frac{2^{\lceil \log_2 t \rceil}}{2^b}$ for unsigned data.

Backward Pass

© Copyright 2020 Xilinx

Implementation

Forward Pass

$$q(x;s) := \begin{cases} \left\lfloor \frac{x}{s} \right\rceil \cdot s & \text{if } n \le \left\lfloor \frac{x}{s} \right\rceil \le p, \\ n \cdot s & \text{if } \left\lfloor \frac{x}{s} \right\rceil < n, \\ p \cdot s & \text{if } \left\lfloor \frac{x}{s} \right\rceil > p. \end{cases}$$

where $n = -2^{b-1}$, $p = 2^{b-1} - 1$ and $s = \frac{2^{\lceil \log_2 t \rceil}}{2^{b-1}}$ for signed data; $n = 0$, $p = 2^b - 1$ and $s = \frac{2^{\lceil \log_2 t \rceil}}{2^b}$ for unsigned data.

Backward Pass

© Copyright 2020 Xilinx

Implementation

Forward Pass

$$q(x;s) := \begin{cases} \left\lfloor \frac{x}{s} \right\rceil \cdot s & \text{if } n \le \left\lfloor \frac{x}{s} \right\rceil \le p, \\ n \cdot s & \text{if } \left\lfloor \frac{x}{s} \right\rceil < n, \\ p \cdot s & \text{if } \left\lfloor \frac{x}{s} \right\rceil > p. \end{cases}$$

where $n = -2^{b-1}$, $p = 2^{b-1} - 1$ and $s = \frac{2^{\lceil \log_2 t \rceil}}{2^{b-1}}$ for signed data; $n = 0$, $p = 2^b - 1$ and $s = \frac{2^{\lceil \log_2 t \rceil}}{2^b}$ for unsigned data.

Backward Pass

$$\frac{\partial}{\partial x}\lfloor x \rceil = \frac{\partial}{\partial x}\lceil x \rceil = 1, \text{ but } \lfloor x \rceil \neq x \text{ and } \lceil x \rceil \neq x$$ (Straight-Through Estimator)

In the backward pass, approximate gradients of round/ceil to 1, without approximating round/ceil to be identity

© Copyright 2020 Xilinx

Implementation

Forward Pass

$$q(x;s) := \begin{cases} \left\lfloor \dfrac{x}{s} \right\rceil \cdot s & \text{if } n \le \left\lfloor \frac{x}{s} \right\rceil \le p, \\ n \cdot s & \text{if } \left\lfloor \frac{x}{s} \right\rceil < n, \\ p \cdot s & \text{if } \left\lfloor \frac{x}{s} \right\rceil > p. \end{cases}$$

where $n = -2^{b-1}$, $p = 2^{b-1} - 1$ and $s = \frac{2^{\lceil \log_2 t \rceil}}{2^{b-1}}$ for signed data; $n = 0$, $p = 2^b - 1$ and $s = \frac{2^{\lceil \log_2 t \rceil}}{2^b}$ for unsigned data.

Backward Pass

non-zero!

$$\nabla_s q(x;s) := \begin{cases} \boxed{\left\lfloor \dfrac{x}{s} \right\rceil - \dfrac{x}{s}} & \text{if } n \le \left\lfloor \frac{x}{s} \right\rceil \le p \\ n & \text{if } \left\lfloor \frac{x}{s} \right\rceil < n \\ p & \text{if } \left\lfloor \frac{x}{s} \right\rceil > p \end{cases}$$

$$\frac{\partial}{\partial x} \lfloor x \rceil = \frac{\partial}{\partial x} \lceil x \rceil = 1, \text{ but } \lfloor x \rceil \ne x \text{ and } \lceil x \rceil \ne x$$ (Straight-Through Estimator)

© Copyright 2020 Xilinx

Implementation

Forward Pass

$$q(x;s) := \begin{cases} \left\lfloor \frac{x}{s} \right\rceil \cdot s & \text{if } n \le \left\lfloor \frac{x}{s} \right\rceil \le p, \\ n \cdot s & \text{if } \left\lfloor \frac{x}{s} \right\rceil < n, \\ p \cdot s & \text{if } \left\lfloor \frac{x}{s} \right\rceil > p. \end{cases}$$

where $n = -2^{b-1}$, $p = 2^{b-1} - 1$ and $s = \frac{2^{\lceil \log_2 t \rceil}}{2^{b-1}}$ for signed data; $n = 0$, $p = 2^b - 1$ and $s = \frac{2^{\lceil \log_2 t \rceil}}{2^b}$ for unsigned data.

Backward Pass

non-zero!

$$\nabla_s q(x;s) := \begin{cases} \boxed{\left\lfloor \frac{x}{s} \right\rceil - \frac{x}{s}} & \text{if } n \le \left\lfloor \frac{x}{s} \right\rceil \le p \\ n & \text{if } \left\lfloor \frac{x}{s} \right\rceil < n \\ p & \text{if } \left\lfloor \frac{x}{s} \right\rceil > p \end{cases}$$

$$\nabla_x q(x;s) := \begin{cases} 1 & \text{if } n \le \left\lfloor \frac{x}{s} \right\rceil \le p, \\ 0 & \text{otherwise.} \end{cases}$$

$$\frac{\partial}{\partial x} \lfloor x \rceil = \frac{\partial}{\partial x} \lceil x \rfloor = 1, \text{ but } \lfloor x \rceil \neq x \text{ and } \lceil x \rfloor \neq x$$ (Straight-Through Estimator)

© Copyright 2020 Xilinx

XILINX

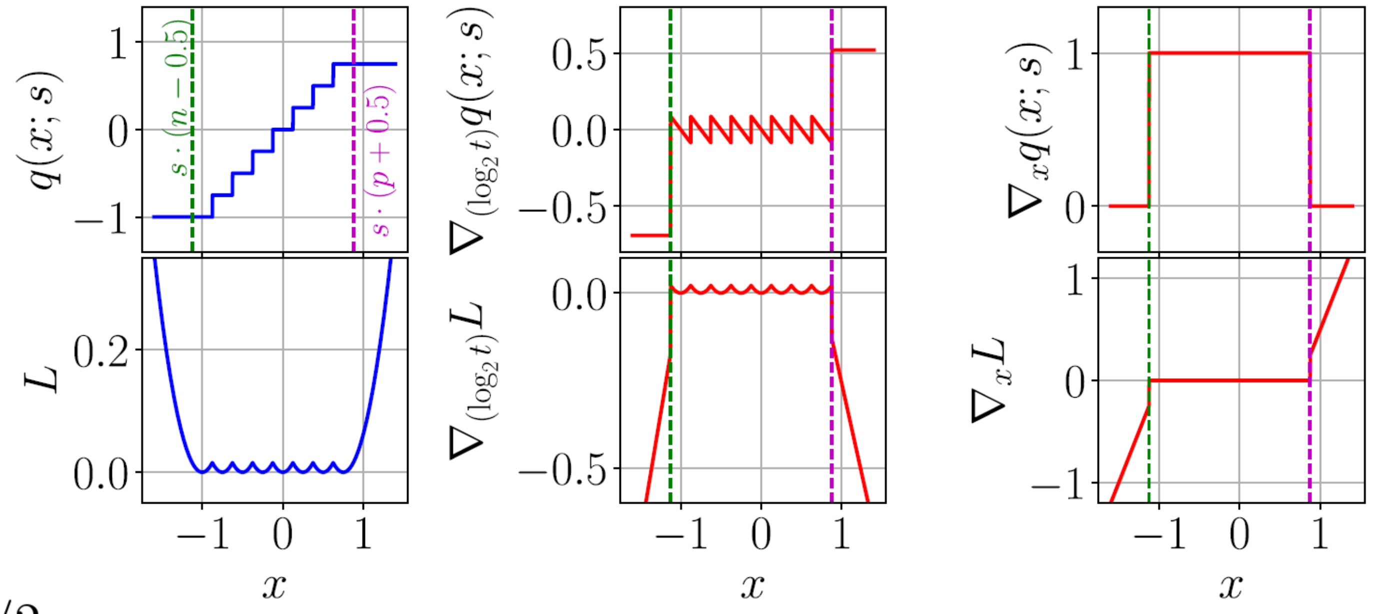
Transfer Curves

$$\nabla_{(\log_2 t)} q(x;s) := s\ \ln(2) \cdot \begin{cases} \left\lfloor \frac{x}{s} \right\rceil - \frac{x}{s} & \text{if } n \leq \left\lfloor \frac{x}{s} \right\rceil \leq p, \\ n & \text{if } \left\lfloor \frac{x}{s} \right\rceil < n, \\ p & \text{if } \left\lfloor \frac{x}{s} \right\rceil > p \end{cases}$$

$$\nabla_x q(x;s) := \begin{cases} 1 & \text{if } n \leq \left\lfloor \frac{x}{s} \right\rceil \leq p, \\ 0 & \text{otherwise.} \end{cases}$$

Threshold Gradient

Input Gradient

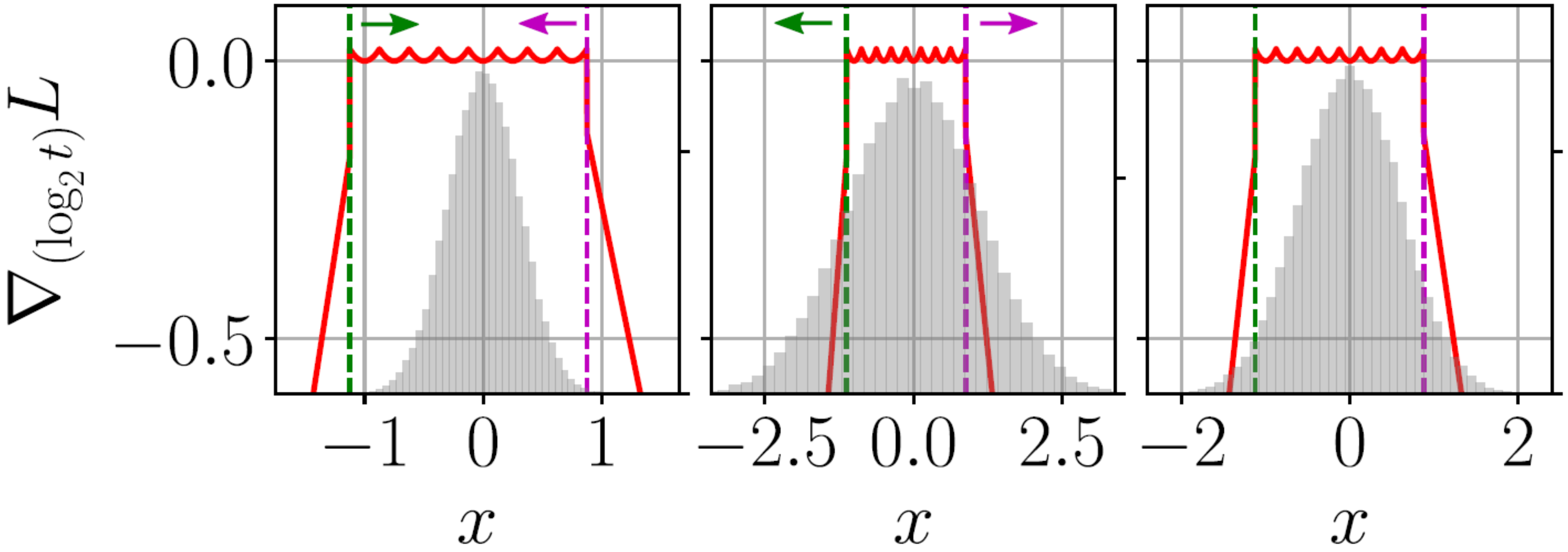


Toy L_2 loss

$$L = \left(q(x;s) - x\right)^2 / 2$$

© Copyright 2020 Xilinx

Range Precision Trade-off



$$\log_2 t := \log_2 t - \alpha \cdot \nabla_{\log_2 t} L$$ (Update Rule)

© Copyright 2020 Xilinx

Clipped Threshold Gradients

PACT's threshold gradients:

$$\frac{\partial y_q(x;\alpha)}{\partial \alpha} = \begin{cases} 0 & x \in (-\infty, \alpha) \\ 1 & x \in [\alpha, +\infty) \end{cases}$$

© Copyright 2020 Xilinx

Clipped Threshold Gradients

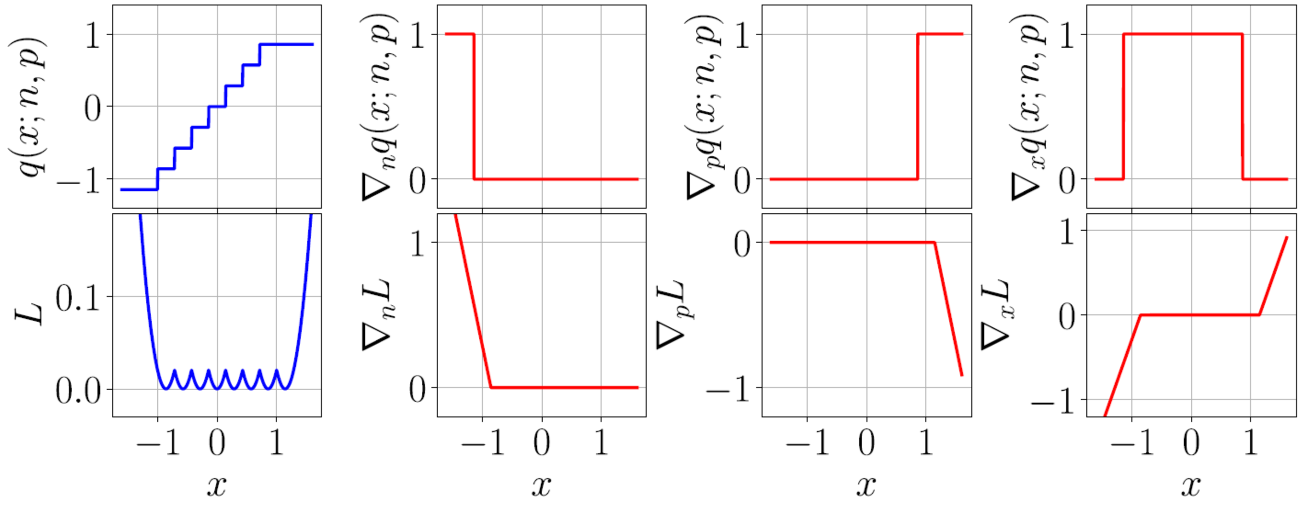
PACT's threshold gradients

$$\frac{\partial y_q(x;\alpha)}{\partial \alpha} = \begin{cases} 0 & x \in (-\infty, \alpha) \\ 1 & x \in [\alpha, +\infty) \end{cases}$$

QAT's threshold gradients (FakeQuant)

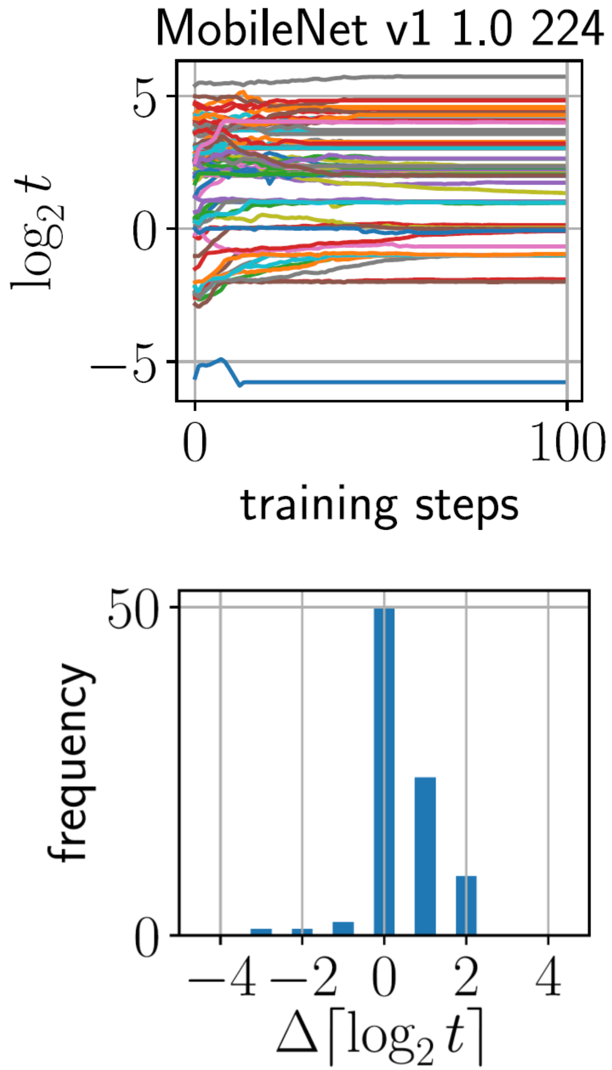
$$q(x;n,p) := \left\lfloor \frac{\text{clip}(x;n,p) - n}{\frac{p-n}{2^b-1}} \right\rceil \cdot \frac{p-n}{2^b-1} + n$$

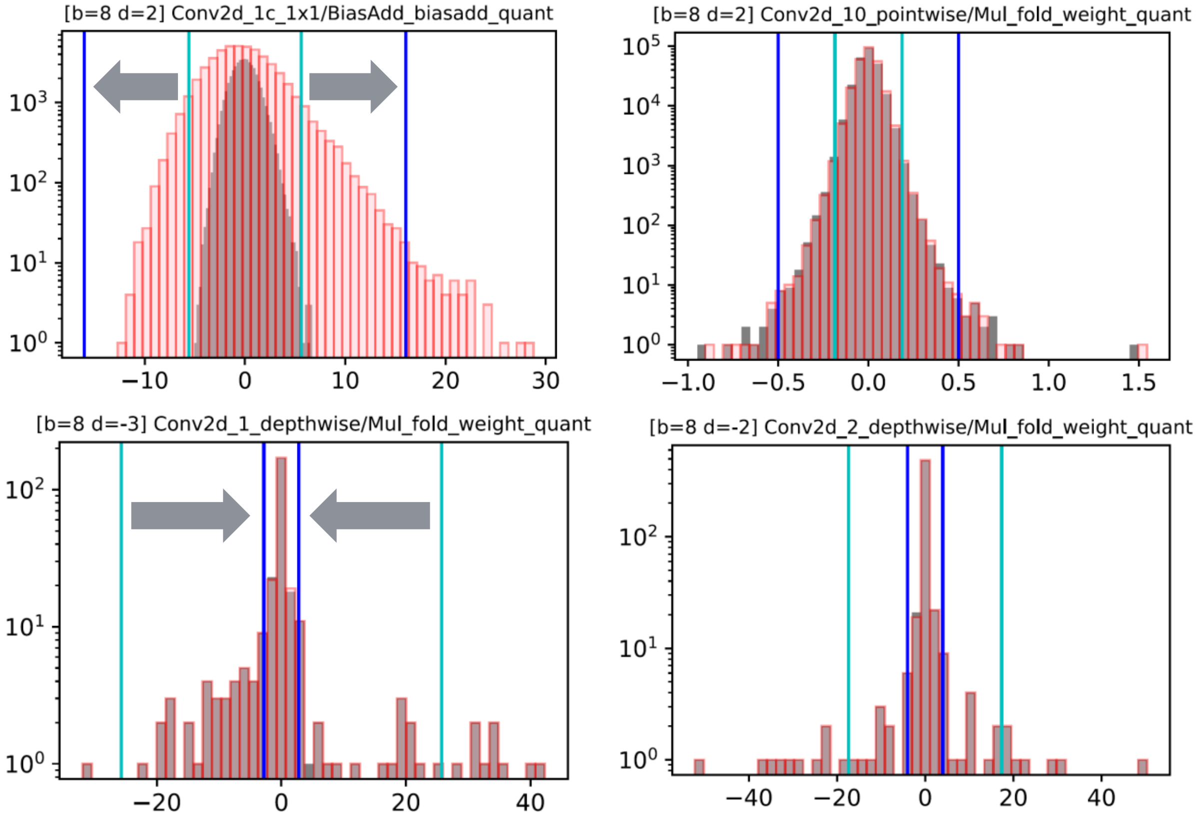
$$\frac{\partial q(x;n,p)}{\partial p} = \begin{cases} 0 & x \in (-\infty, p) \\ 1 & x \in [p, +\infty) \end{cases}$$



© Copyright 2020 Xilinx

Distributions after TQT retraining





© Copyright 2020 Xilinx

Results

Mode		Precision	Bit-width (W/A)	Accuracy (%) top-1	Accuracy (%) top-5	Epochs
MobileNet v1 1.0 224						
Static		FP32	32/32	71.0	90.0	
Static		INT8	8/8	0.6	3.6	
Retrain	wt	FP32	32/32	71.1	90.0	3.4
Retrain	wt	INT8	8/8	67.0	87.9	4.6
Retrain	wt,th	INT8	8/8	71.1	90.0	2.1
Retrain	wt,th	INT4	4/8	–	–	
MobileNet v2 1.0 224						
Static		FP32	32/32	70.1	89.5	
Static		INT8	8/8	0.3	1.2	
Retrain	wt	FP32	32/32	71.7	90.7	3.2
Retrain	wt	INT8	8/8	68.2	89.0	2.7
Retrain	wt,th	INT8	8/8	71.8	90.6	2.2
Retrain	wt,th	INT4	4/8	–	–	

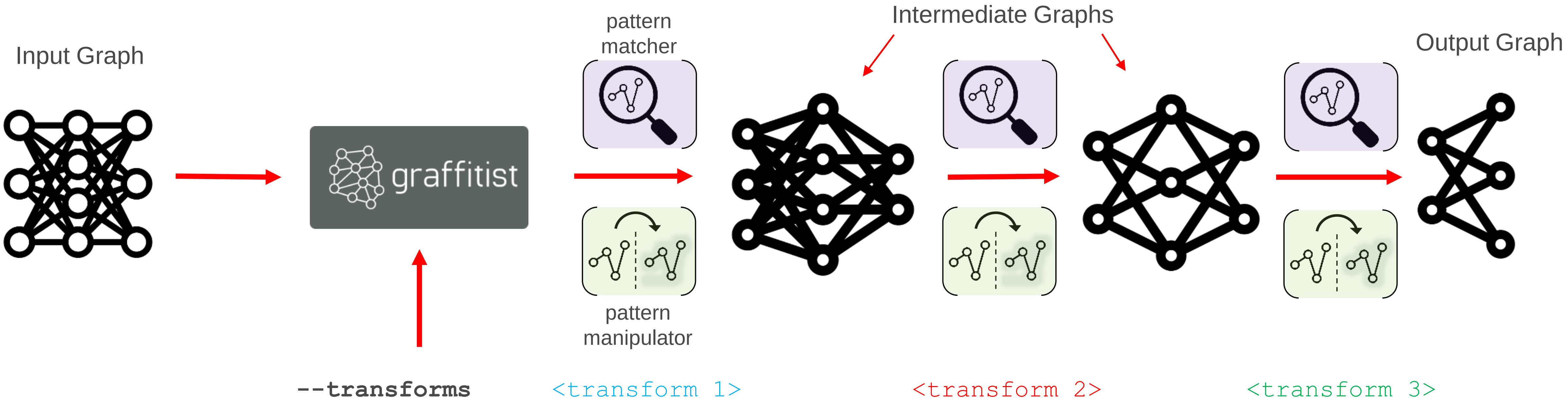
© Copyright 2020 Xilinx

graffitist

github.com/Xilinx/graffitist

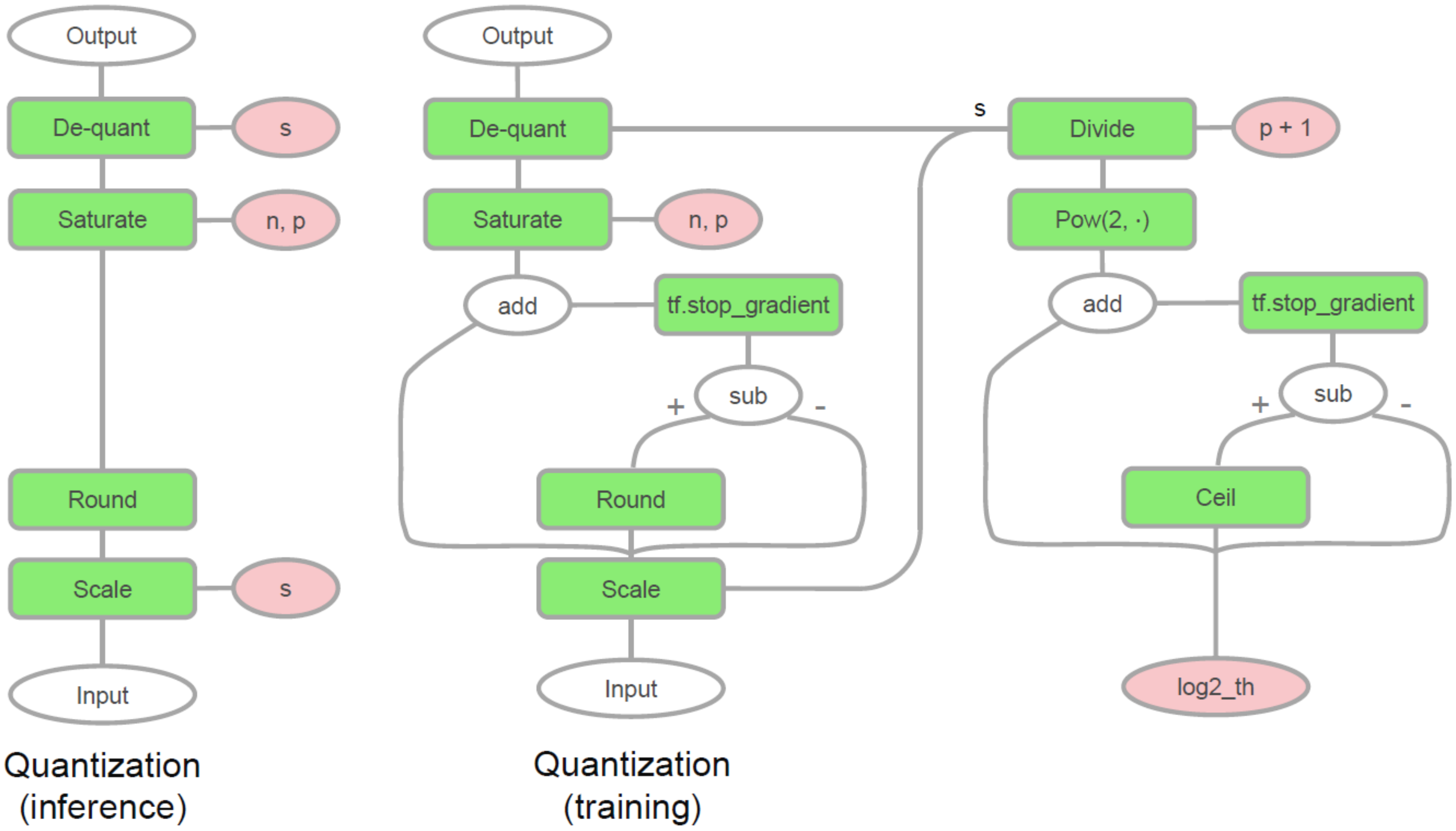
© Copyright 2020 Xilinx
XILINX

Tool for Neural Net Optimizations



© Copyright 2020 Xilinx

Quantization Layer for TQT (unfused)



© Copyright 2020 Xilinx

Layer Precisions

Conv/FC

$$q_8\left(q'_{16}\left(\sum\left(q_{8/4}(w)\cdot q_8(x)\right)\right)+q'_{16}(b)\right)$$

Avgpool

$$q_8\left(\sum\left(q_8(r)\cdot q_8(x)\right)\right)$$

Eltwise Add

$$q_8\left(q'_8(x)+q'_8(y)\right)$$

Concat

$$\text{concat}(q'_8(x),q'_8(y),q'_8(z))$$

© Copyright 2020 Xilinx

Graph Optimizations

- BatchNorm folding (adopt best practices from Jacob et al., 2017)
  - Ensure folded batch norms in training and inference graphs are mathematically equivalent
  - Apply batch norm corrections (reduce training jitter by switching between batch and moving average statistics)
  - Freeze batch norm moving mean and variance updates post convergence for improved accuracy
- Explicitly merging input scales for scale preserving ops such as concat, bias-add, eltwise-add, and maximum (for leaky relu)
- Collapsing concat-of-concat layers into single concat, splicing identity nodes
- Modeling average pool layers as depthwise conv layers with reciprocal multiplier as weights to enable quantization

© Copyright 2020 Xilinx

Thank You

© Copyright 2020 Xilinx

Backup

© Copyright 2020 Xilinx

TQT - Components

Input Graph

Training Platform (TF)

graffitist

Quantized Training Graph

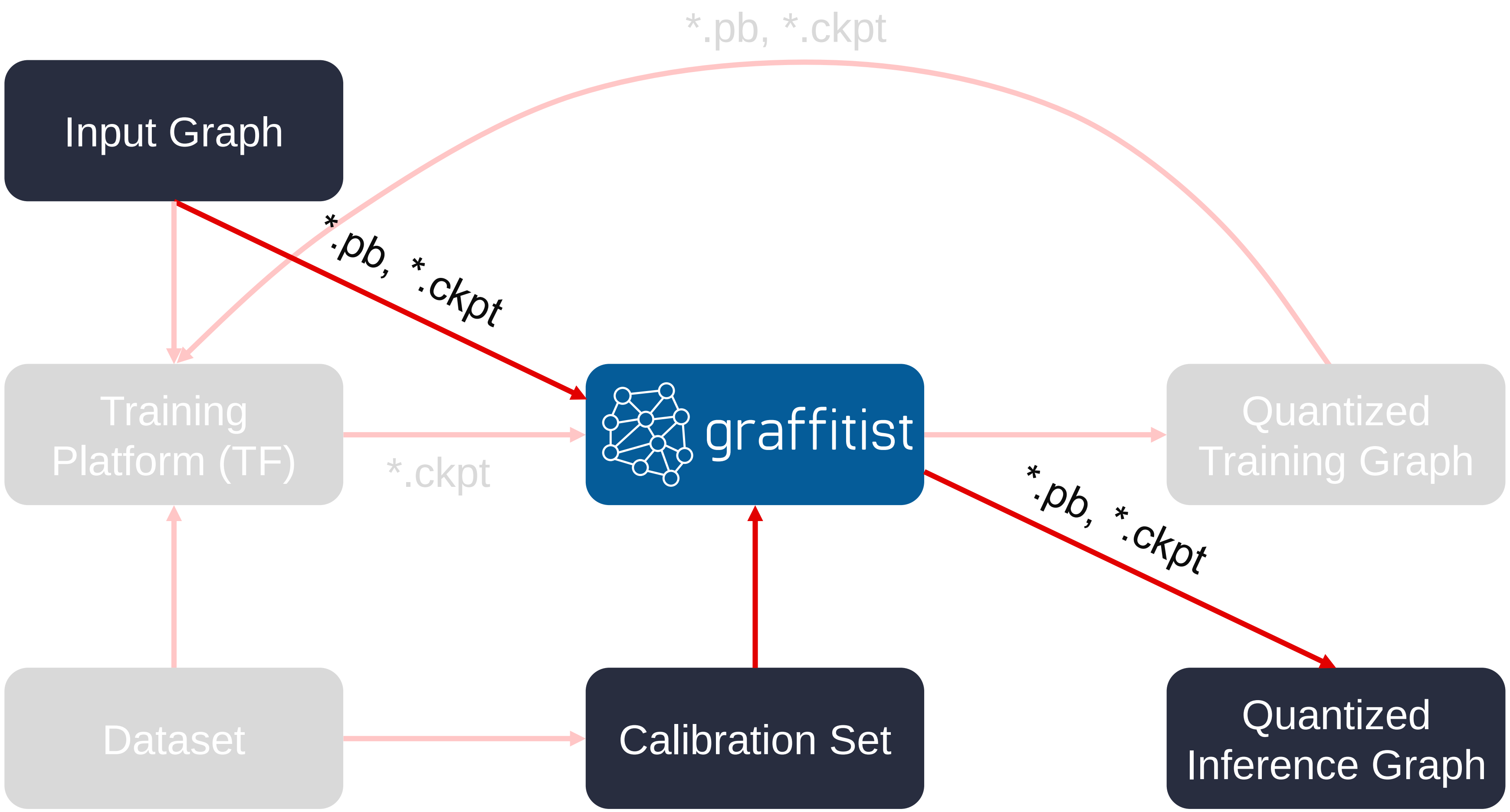
Dataset

Calibration Set

Quantized Inference Graph

© Copyright 2020 Xilinx

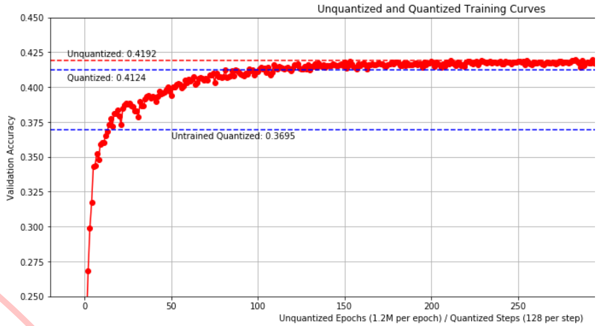
Static Mode

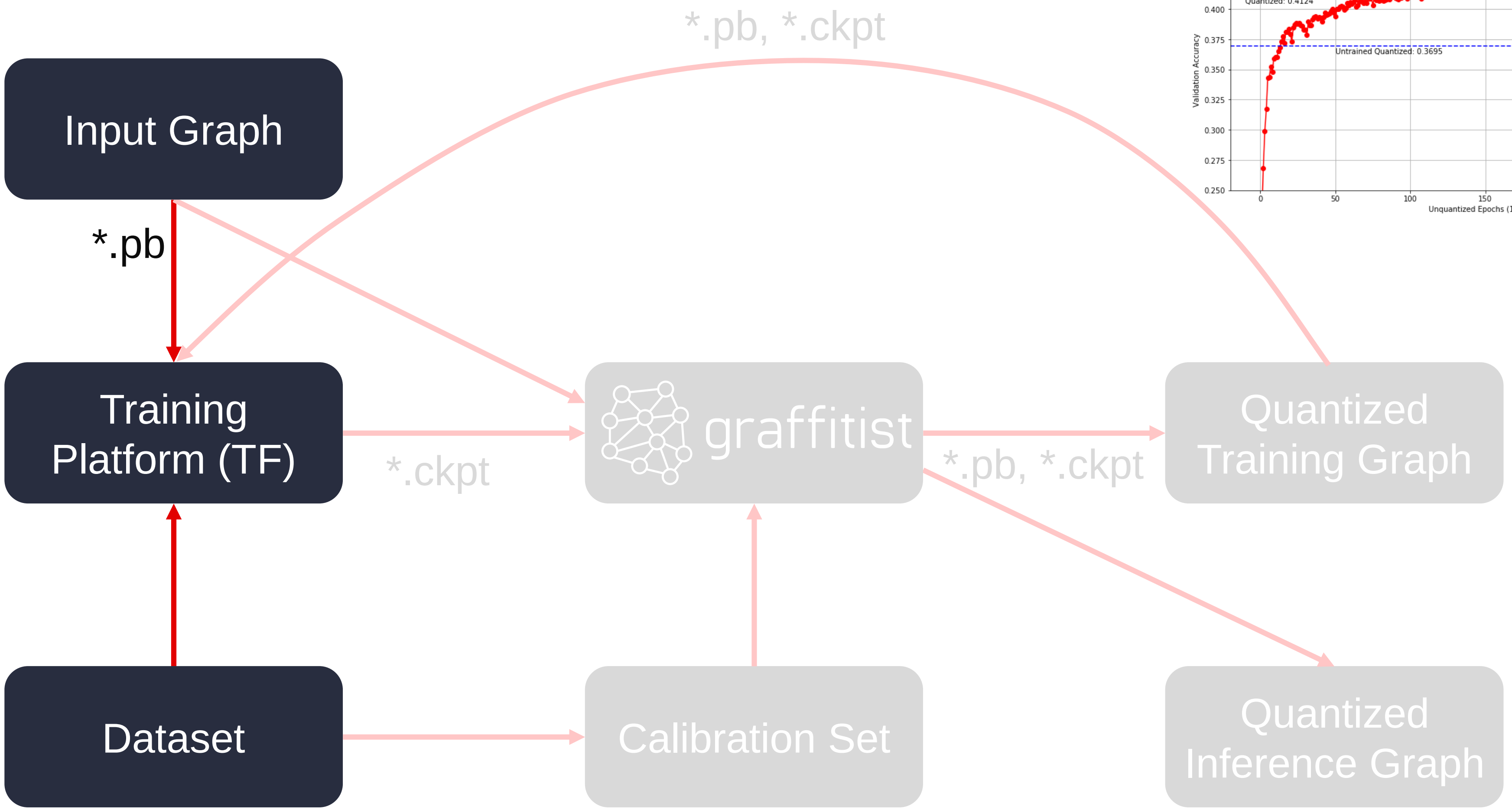


© Copyright 2020 Xilinx

Retrain Mode

Step 1: Train on the original input graph (or use pre-trained weights)

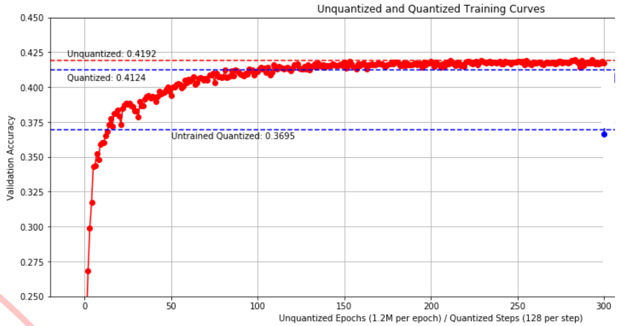


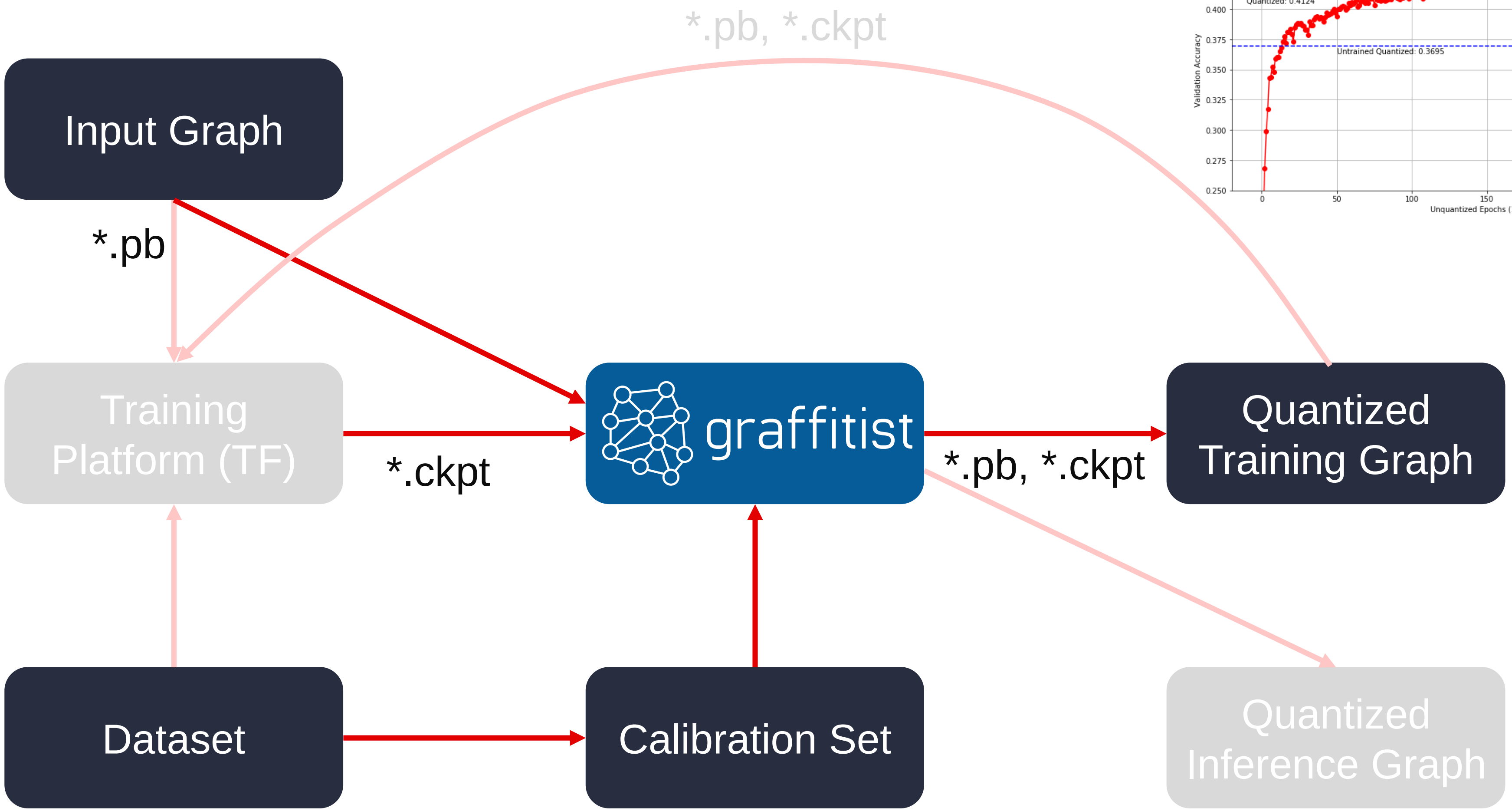


© Copyright 2020 Xilinx

Retrain Mode

Step 2: Generate quantized training graph; calibrate thresholds

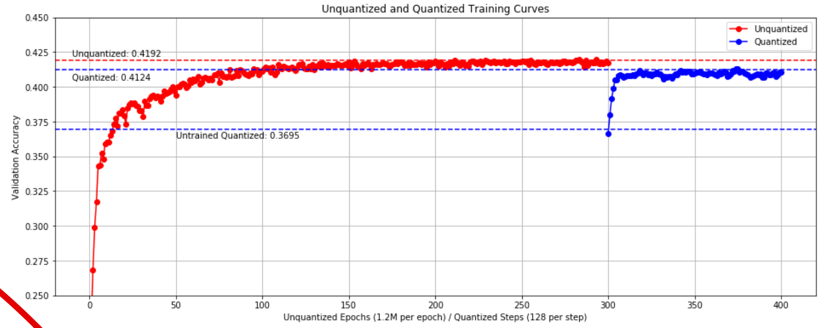


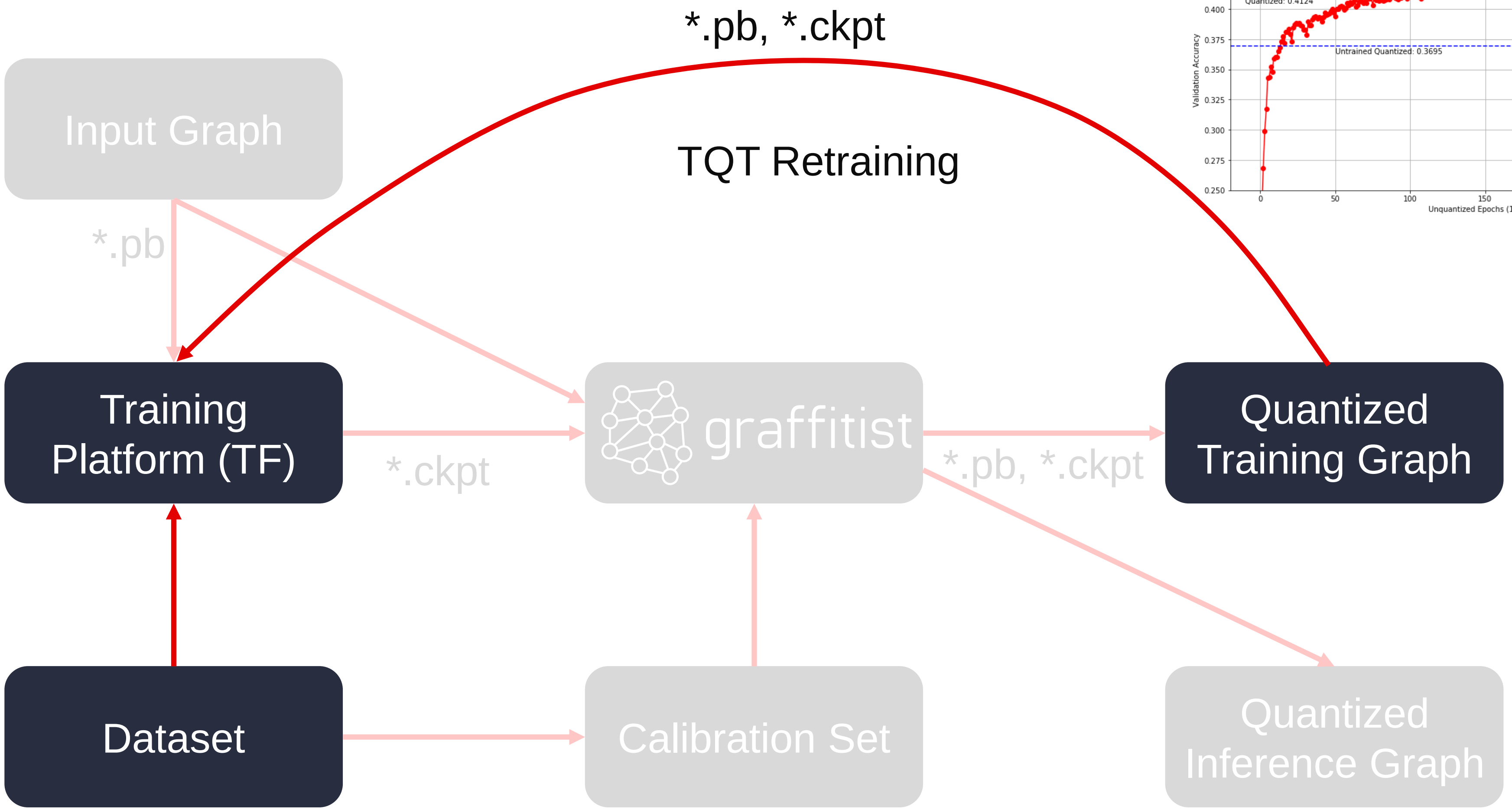


© Copyright 2020 Xilinx

Retrain Mode

Step 3: Retrain quantized training graph (learn weights & thresholds)

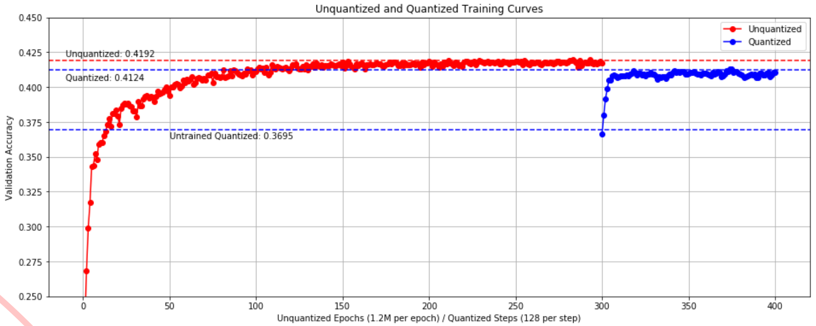


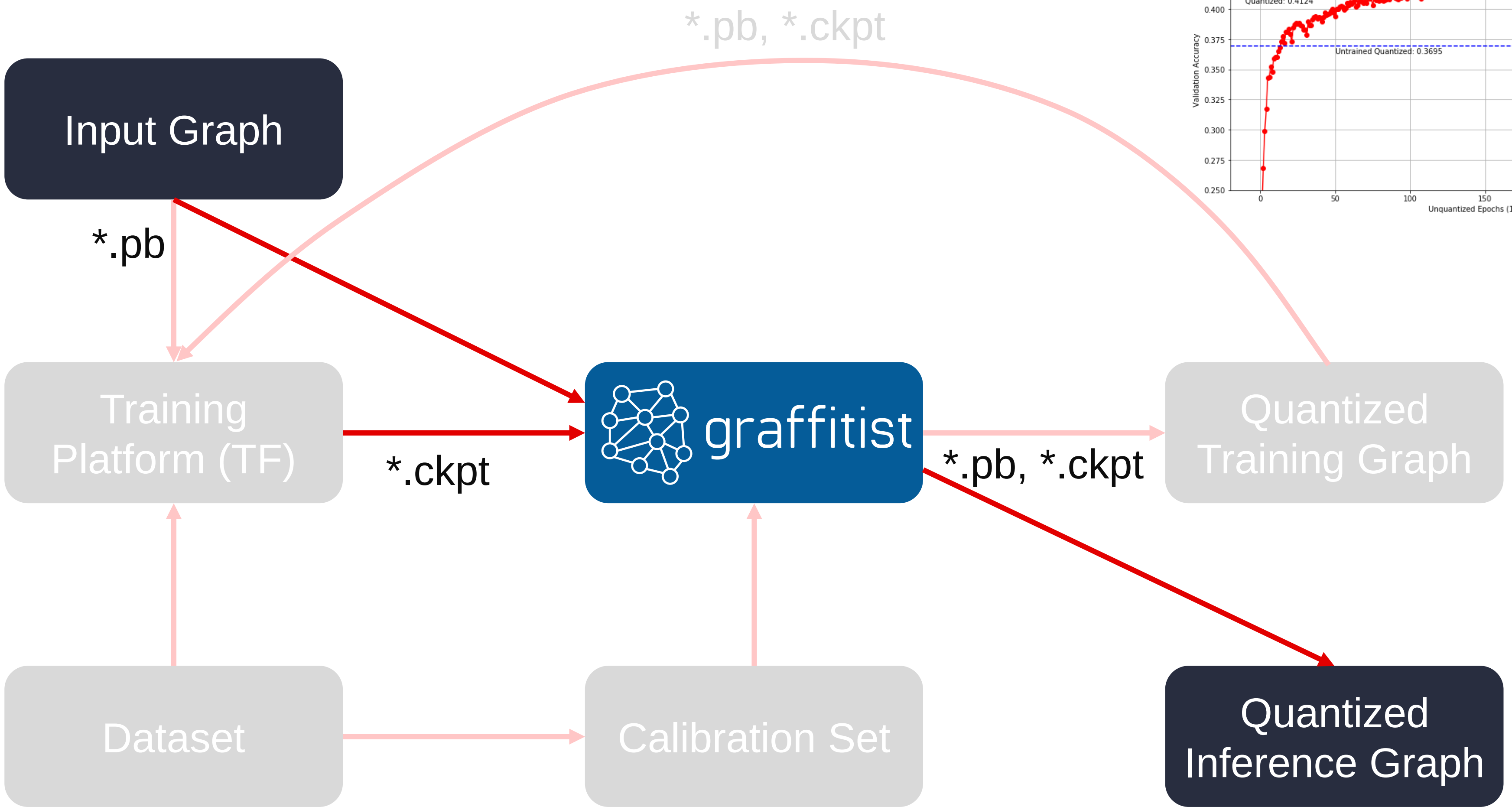


© Copyright 2020 Xilinx

Retrain Mode

Step 4: Generate quantized inference graph for compiler





© Copyright 2020 Xilinx