Qualcomm

June 23, 2019

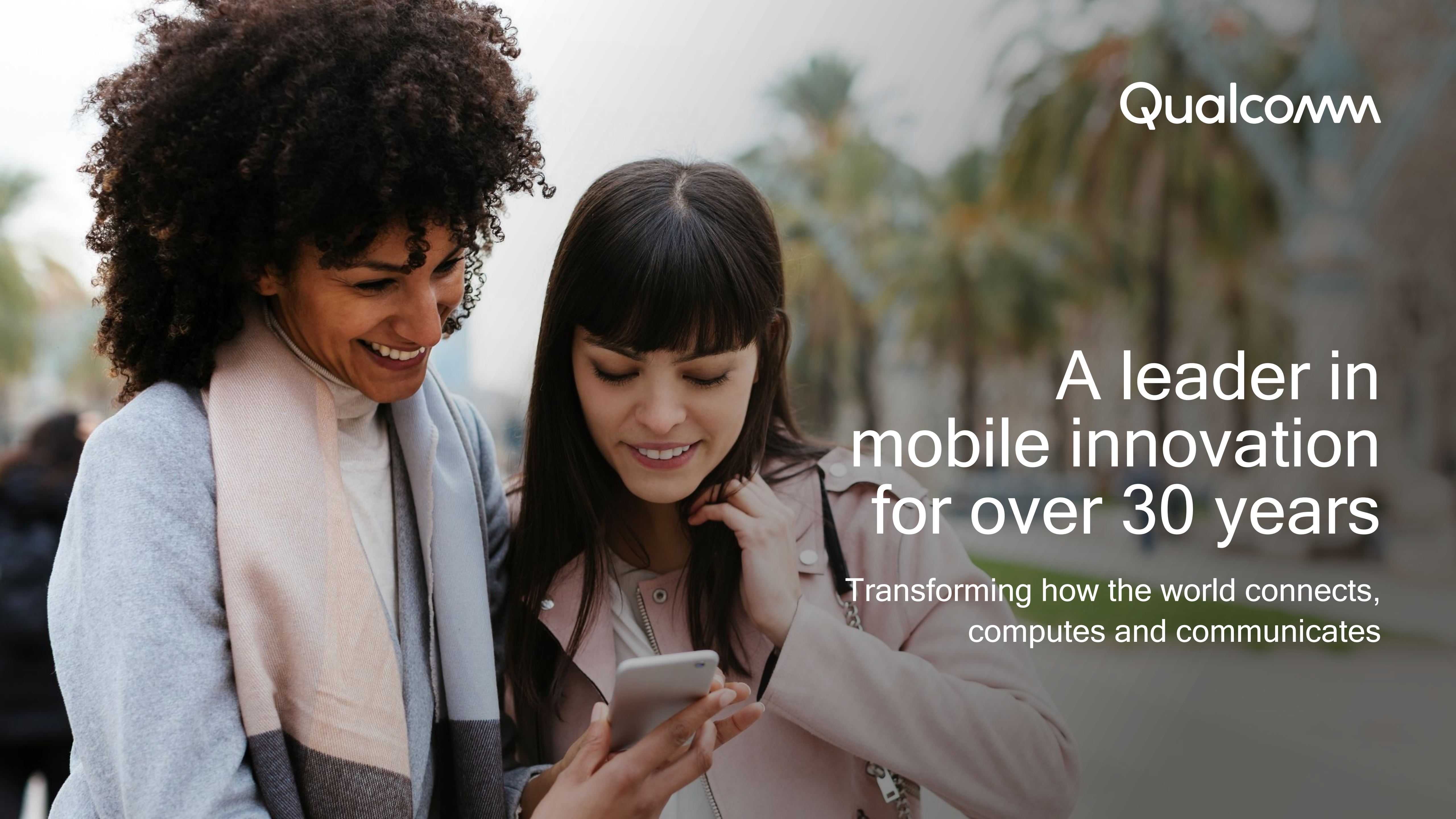
Phoenix, AZ

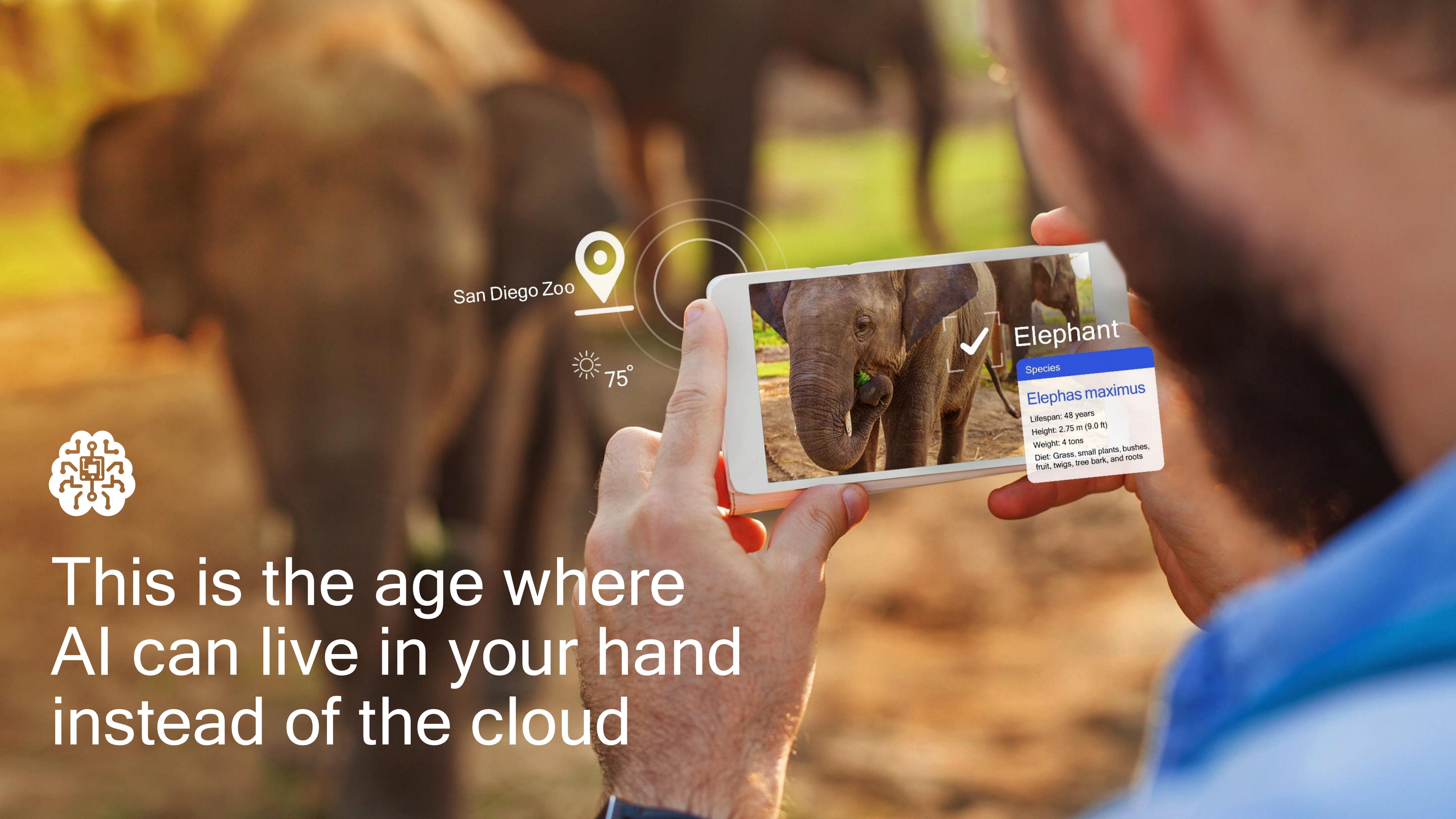
Quantization without fine-tuning

Harris Teague

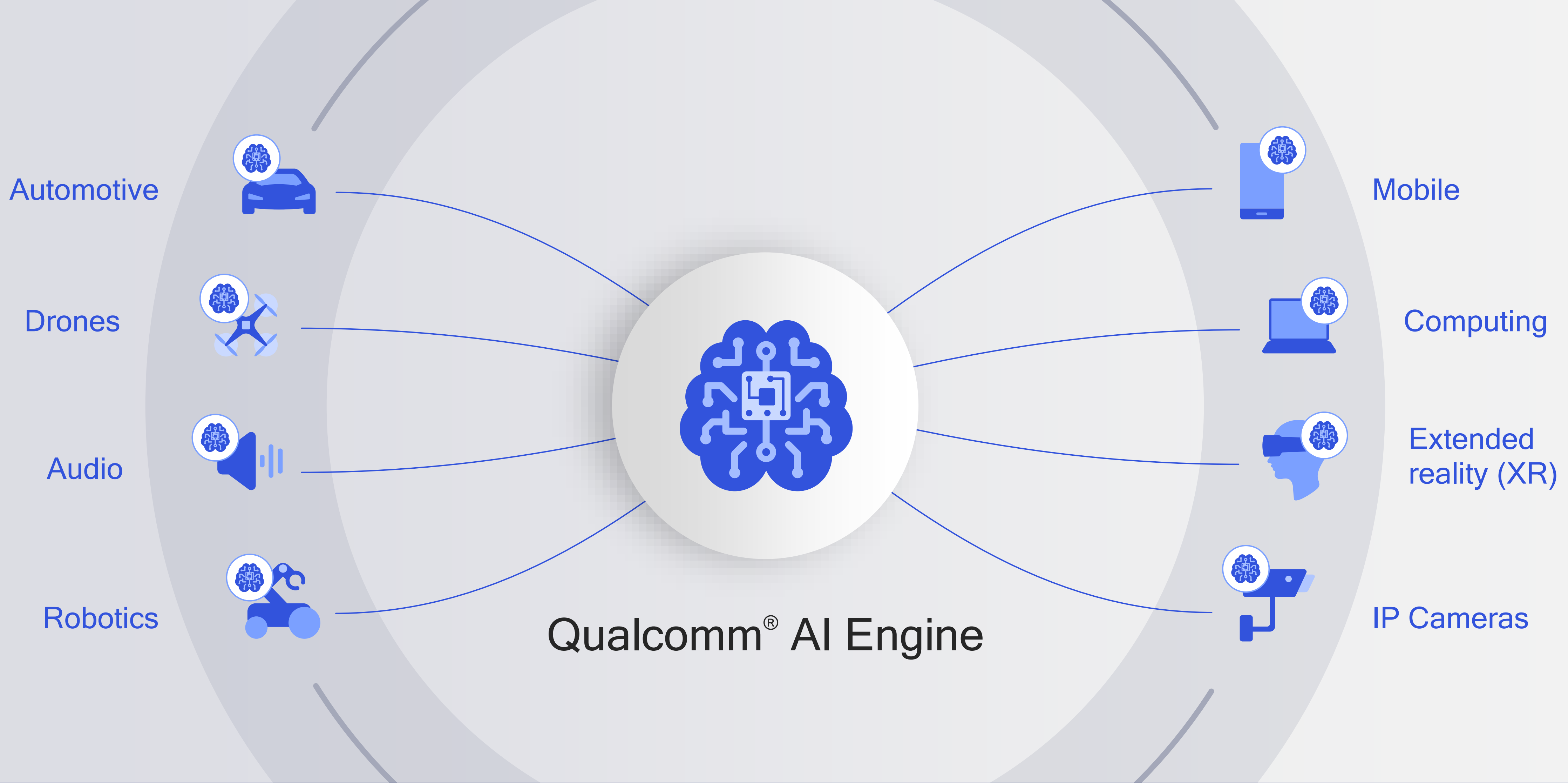
Principal Engineer

Qualcomm AI Research


Qualcomm
A leader in
mobile innovation
for over 30 years
Transforming how the world connects,
computes and communicates



This is the age where AI can live in your hand instead of the cloud



Pioneering on-device intelligence

Qualcomm AI Engine is a product of Qualcomm Technologies Inc. and/or its subsidiaries.

Quantization

Quantization is the most effective way of improving

- Power Consumption
- Latency
- Memory usage
- Area needed

Per hour of sweat spent

8bit everything would be preferable

M. Horowitz, "Computing's energy problem (and what we can do about it)", *ISSCC '14*, pp. 10-14.

Energy based on ASIC | Area based on TSMC45nm

Memory	Energy (pJ)
Cache	(64bit)
8KB	10
32KB	20
1MB	100
DRAM	1300-2600

Operation	Energy (pJ)	Area (μm^2)
int8 addition	0.03	36
int16 addition	0.05	67
int32 addition	0.1	137
float16 addition	0.4	1,360
float32 addition	0.9	4,184
int8 multiplication	0.2	282
int32multiplication	3.1	3,495
float16multiplication	1.1	1,640
float32multiplication	3.7	7,700

8-bit solutions in practice

Level 1

- No data
- No backpropagation
- No architecture changes

Level 2

- Data needed
- No backpropagation
- No architecture changes

Level 3

- Data needed
- Backpropagation
- No architecture changes

Level 4

- Data needed
- Backpropagation
- Architecture changes

In decreasing order of practicality for many users/customers

Is every model 8-bit quantizable?

Model	Top1 accuracy	Top1 quantized
InceptionV3	0.78	0.78
NasnetMobile	0.74	0.722
Resnet 50	0.756	0.75
MobileNetV2	0.749	0.004

Imagenet – INT8 Asymmetric quantization weights+activations. No fine-tuning.

Selecting [min, max] based on data

Methods exist to increase performance, but they rely on architecture changes/fine-tuning

Results from Jacob et al. 2018 Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only inference

Better methods for quantization without fine-tuning

Markus Nagel

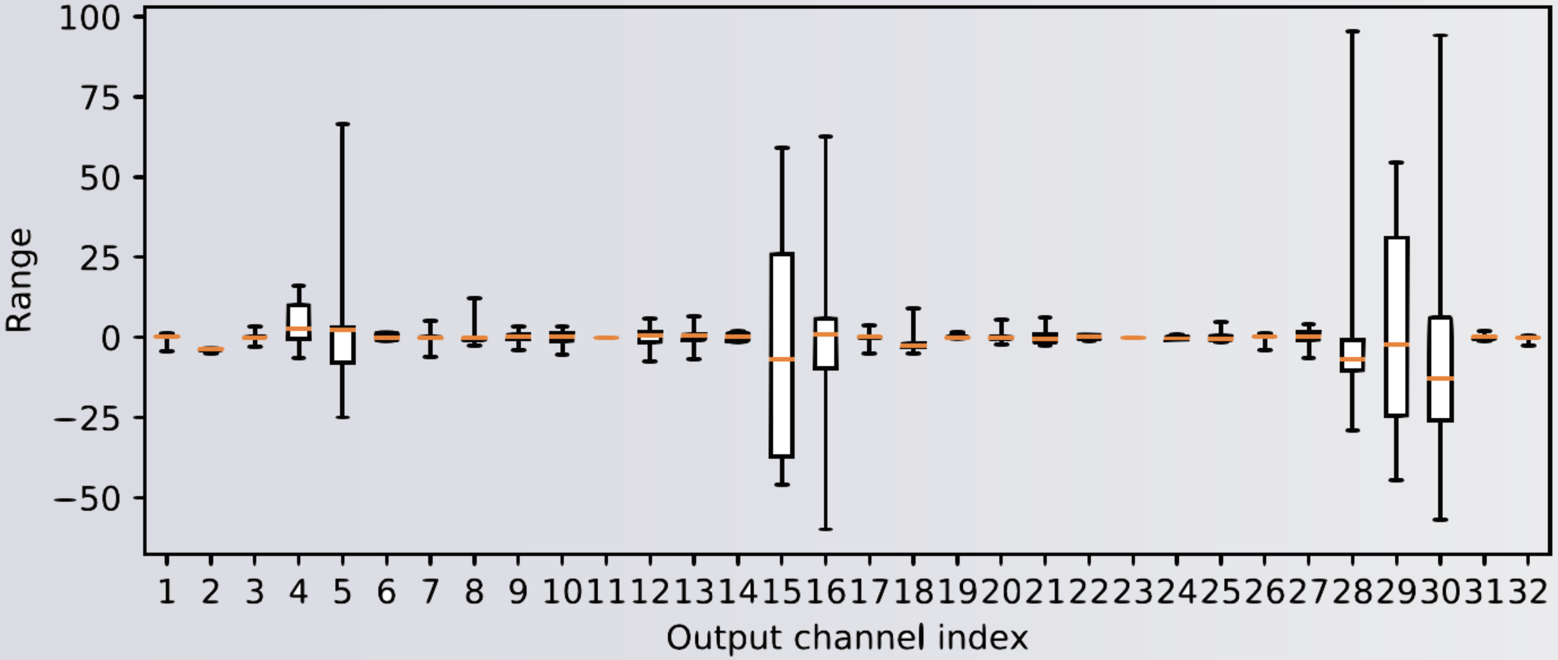
Mart van Baalen

Tijmen Blankevoort

Max Welling

- Developed methods for quantization without use of data
- Excellent hassle-free quantization results without needing any training

Problem 1: Imbalance in weight ranges per output



Distributions of weights in 2nd layer of MobileNetV2 (Imagenet)

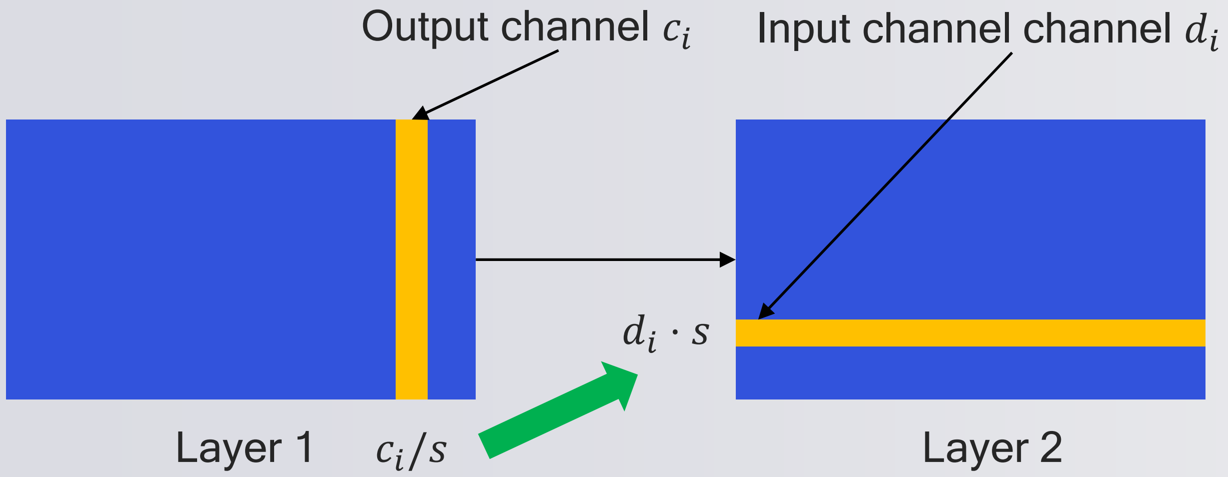
We've seen large difference in the ranges for each output of a layer

Large quantization grids decrease performance for the smaller ranges catastrophically

Per-channel quantization [1] solves this problem, but not supported on all hardware

[1] Krishnamoorthi 2018. Quantizing deep convolutional networks for efficient inference: A whitepaper

Cross-layer equalization (CLE) procedure



If $r_i^{(j)}$ is the range of layer j, for output/input I, we can scale as follows to optimize:

$$s_i = \frac{1}{r_i}\sqrt{\left(r_i^{(1)} r_i^{(2)}\right)}$$

- Works for networks with (P)ReLU activations
- Balance out the scaling factors between any layers that are ‘simply connected’ by scaling

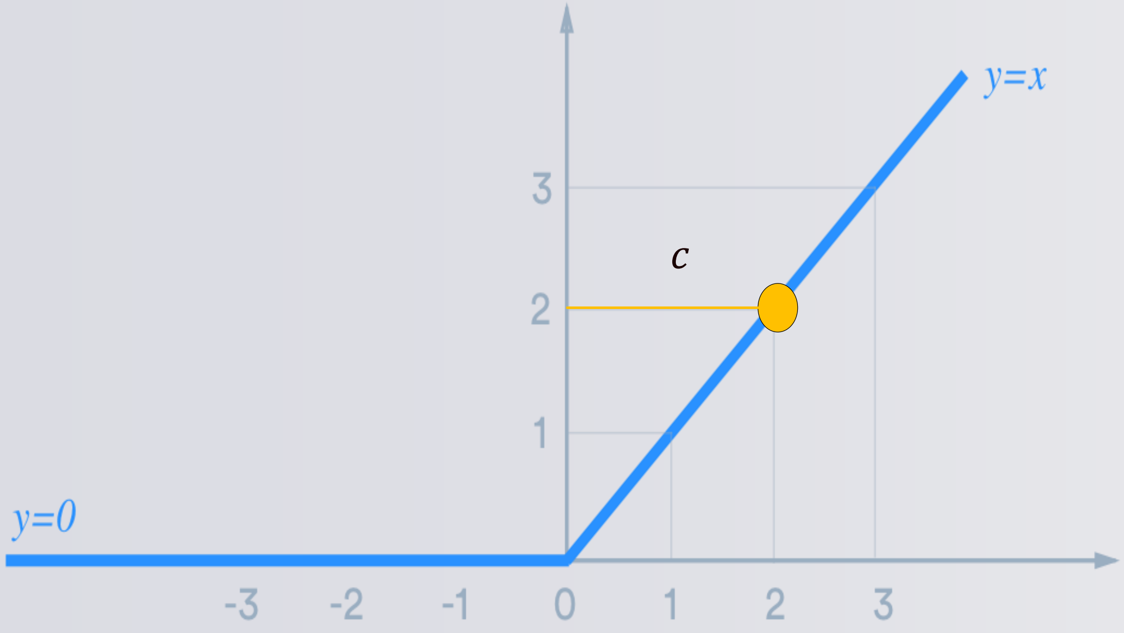
For residual blocks we only scale the layers in a block

Method also proposed by [1], work done concurrently

[1] Meller et al. 2019. Same, Same but Different - Recovering Neural Network Quantization Error through weight factorization

Bias absorption

We can decrease large activation ranges by moving some of the range to the next layer



Consider one output activation for a Layer:

$$relu(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b})$$

Where the activations are always minimally $\boldsymbol{c}$ (c is minimally 0)

For r the ReLU function we have

$$r(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b} - \boldsymbol{c}) = r(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) - \boldsymbol{c}$$

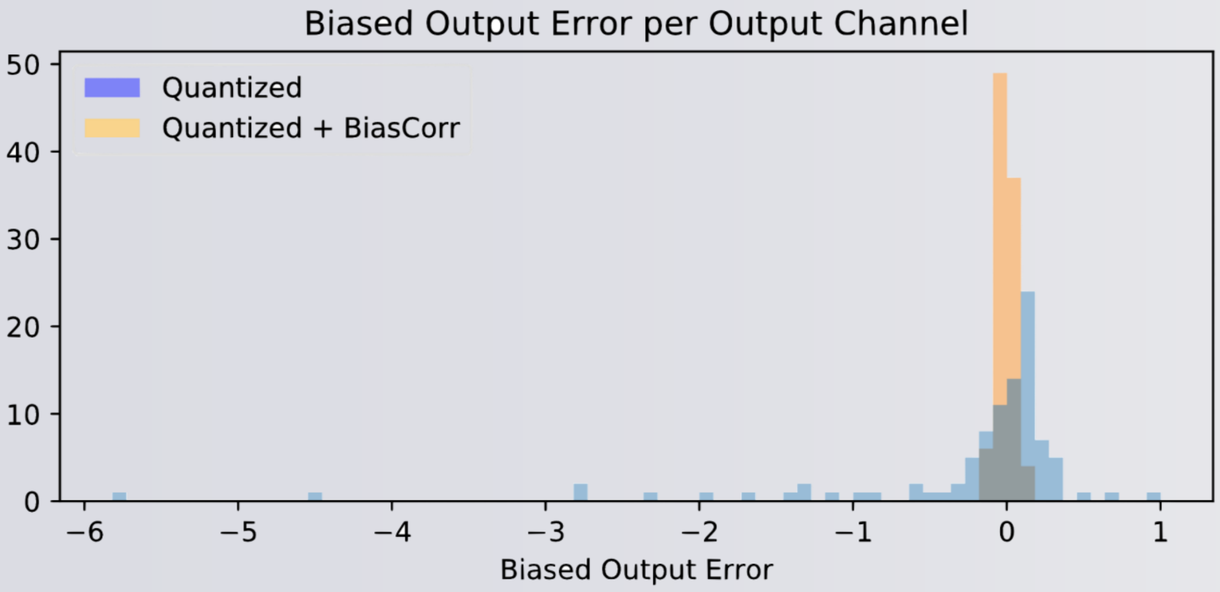
We move c from layer 1, and include it in the bias of the next layer 2.

This procedure generally helps performance after equalization

Problem 2: Quantization introduces biased error

$$Kernel = \begin{bmatrix} -0.1 & -0.2 & -0.1 \\ -0.1 & 0.1 & 0.2 \\ 0.1 & -0.2 & -0.1 \end{bmatrix} \qquad \sum_i K_i = -0.4$$

Illustrative example



- Quantization could lead to a biased error: $\mathbb{E}[y] - \mathbb{E}[\tilde{y}] \neq 0$
- Biased error's effects are more detrimental to the network compared to the same magnitude of unbiased error.
- A biased error can also be introduced by clipping weights/activations (Sometimes inadvertently because of wrong ranges!)

Solve with bias correction

Given $\boldsymbol{W}$ a weight matrix, and a quantized approximation $\widetilde{W}$ we can write

$$\boldsymbol{W} = \widetilde{\boldsymbol{W}} + \boldsymbol{\epsilon}$$

The bias of an output is given as

$$\mathbb{E}[\boldsymbol{y}] - \mathbb{E}[\widetilde{\boldsymbol{y}}] = \boldsymbol{\epsilon}\mathbb{E}[\boldsymbol{x}]$$

Key idea: Bias correction

We find $\boldsymbol{\epsilon}\mathbb{E}[\boldsymbol{x}]$ and subtract it from the output after quantization to correct for the bias effect!

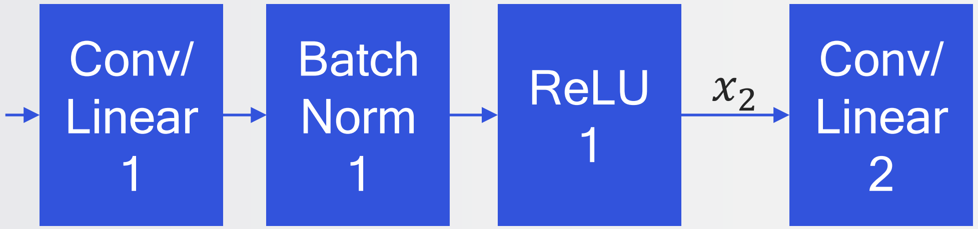
Finding $\mathbb{E}[x]$

- With Data:

  Simply calculate $\mathbb{E}[\boldsymbol{y}] - \mathbb{E}[\widetilde{\boldsymbol{y}}]$

- Without Data:

  We can use Batch-Normalization statistics!



This gives us:

$$\mathbb{E}[x_2] = \gamma\mathcal{N}\left(\frac{-\beta}{\gamma}\right) + \beta[1 - \Phi(\frac{-\beta}{\gamma})]$$

Where $\mathcal{N}(x)$ is used to denote the normal $\mathcal{N}(x|0,1)$ PDF and $\Phi(x)$ the normal CDF

Dataless activation range setting

Models are sensitive to proper activation range setting

Mistakes with range setting on data are the same as clipping activations, which has strong detrimental effects

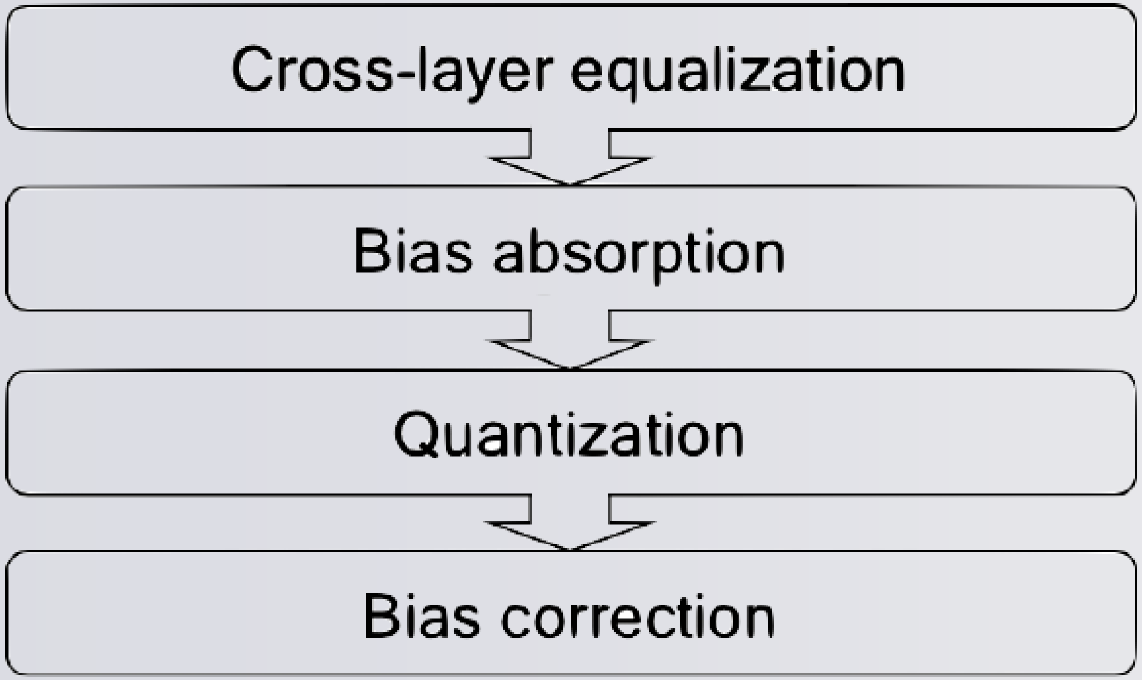
Solution

We set the activation ranges to 6σ based on the batch normalization parameters

Mobilenet V1 - Imagenet	INT8 - top 1 acc
DFQ	70.24%
+ act 4σ	70.30%
+ act 5σ	70.36%
+ act 6σ	**70.51%**
+ act 7σ	70.43%

Example of benefit of activation setting

Data-Free quantization approach



Flow diagram of the data-free quantization method

Results reported henceforth as DFQ indicate the combination of both methods

No data was used anywhere.

Algorithms work well with data too!

Results

- All with INT8 weights/activations
- Asymmetric quantization
- 6 sigma activation ranges used for everything

MobileNetV2 results

A significant amount of performance regained. Although not close to original, it shows that quantization error bias is definitely a problem

	Top1 Float32	Top1 INT8	Delta
Original Model	71.72%	0.12%	-71.6%
+Bias correction	71.72%	52.02%	-20.7%
Clip [-15,15]	67.06%	56.68%	-15.04%
+Bias correction	71.15%	70.43%	-1.29%
Equalization + Absorption	71.57%	70.92%	-0.8%
+ Bias correction	71.57%	71.19%	-0.53%
+ Bias correction /w data	71.57%	71.41%	-0.31%

Clipping hurts model performance significantly. In FP32 we regain a lot of lost accuracy. In INT8, Clipping + bias correction baseline is quite strong. [-15,15] chosen arbitrarily

Bias correction improves on top of CLE

Data version: only -0.3% error degradation!

Imagenet results

~D - no data needed
~BP no backprop needed
~AC no architecture changes needed

	~D	~BP	~AC	MobileNetV2 FP32	MobileNetV2 INT8	MobileNetV1 FP32	MobileNetV1 INT8	ResNet18 FP32	ResNet18 INT8	ResNet18 INT6
DFQ (ours)	✓	✓	✓	71.7%	**71.2%**	70.8%	**70.5%**	69.7%	**69.7%**	66.3%
Per-layer [18]	✓	✓	✓	71.9%	0.1%	70.9%	0.1%	69.7%	69.2%*	63.8%*
Per-channel [18]	✓	✓	✓	71.9%	69.7%	70.9%	70.3%	69.7%	69.6%*	**67.5%***
QT [16] ^	✗	✗	✓	71.9%	70.9%	70.9%	70.0%	-	**70.3%†**	67.3%†
SR+DR†	✗	✗	✓	-	-	-	71.3%	-	68.2%	59.3%
QMN [30]	✗	✗	✗	-	-	70.8%	68.0%	-	-	-
RQ [21]	✗	✗	✗	-	-	-	**70.4%**	-	69.9%	**68.6%**

Per-channel is Krishnamoorthi 2018
QT is Jacob et al. 2017
SR+DR is stochastic rounding + dynamic ranges (Louizos et al. 2019)
QMN is Qualcomm mobilenet architecture (Sheng et al. 2018)
RQ is Relaxed quantization (Louizos et al. 2019)

Other CV tasks

DeepLab V3	mIoU Float32	mIoU INT8	Delta
Asymmetric quant	72.94	41.4	-31.54
DFQ	72.45	**72.33**	**-0.12**
Per-channel	72.94	71.44	-1.5

Semantic Segmentation - DeeplabV3+ MobilenetV2 backend. Pascal VOC

Mobilenet-SSD	mAP Float32	mAP INT8	Delta
Asymmetric quant	68.47	10.63	-57.84
DFQ	68.56	**67.91**	**-0.56**
Per-channel	68.47	67.52	-0.95

Object detection MobilenetV2 SSD-lite Pascal VOC

Conclusion

- Applying DFQ procedure gives equal or better performance compared to per-channel quantization. Per-tensor quantization can be used instead.
- Fully data-free approach works just as well as methods that fine-tune or train from scratch.
- Works for any convolutional architecture.
- Method is very simple to apply, single API call. Can always be tried!

Paper available on Arxiv:

Data-Free Quantization through Weight Equalization and Bias Correction

https://arxiv.org/abs/1906.04721

Go forth and quantize!

Qualcomm

Thank you

Follow us on:

For more information, visit us at:

www.qualcomm.com & www.qualcomm.com/blog