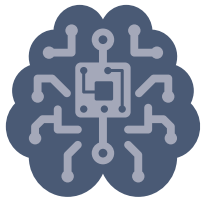
March 25, 2018

Qualcomm

A Quantization-Friendly Separable Convolution for MobileNets

Tao Sheng, Chen Feng, Shaojie Zhuo,
Xiaopeng Zhang, Liang Shen, Mickey Aleksic
Qualcomm



Outline

Model quantization is not the full story of fixed-point inferencing

- Quantization and the problems in MobileNetV1
  - Why quantization is important
  - TF8 quantization
  - How to measure quantization loss
  - MobileNetV1 and separable convolution
  - The problems
- Quantization Loss Analysis
  - Large Quantization Loss caused by BatchNorm in Depthwise Conv Layer
  - ReLU6, a troublemaker?
- A Quantization-friendly Model
- Experimental Results

Quantization and the problems in MobileNetV1

Why quantization is important

- Quantization has become extremely useful when doing deep learning inferencing on mobile devices, especially the hardware only capable of fix-point computation, like DSP/NPU, which can achieve the best power efficiency.
- 8-bit quantization works well on previous baseline deep learning models, like AlexNet, VGG, GoogleNet V1-V4, and ResNet because they are all over-parameterized by design to achieve best accuracy in ImageNet competition.
- Industry starts to push the deep learning to the edge devices, and invented the deep learning models at the tradeoff between accuracy and efficiency, eg. Google's MobileNets.
- 8-bit quantization didn't work for MobileNet models released from Google: (http://download.tensorflow.org/models/mobilenet_v1_1.0_224_2017_06_14.tar.gz)

Quantization and the problems in MobileNet

Static Tensorflow 8-bit (TF8) Quantization

1. Given known min/max for input, weights, and output, calculate their scale Δ and offset δ

$$x_{quant8} = \left[\frac{x_{float} - x_{min}}{x_{max} - x_{min}} \times 255\right] = \left[\frac{x_{float}}{\Delta_x}\right] - \delta_x, \qquad where\ \Delta_x = \frac{x_{max} - x_{min}}{255} \quad and \quad \delta_x = \left[\frac{x_{min}}{\Delta_x}\right]$$

2. Add offset δ to quantized input and weights for computing their accumulated result

$$accum_{float} = \sum(x_{float} * w_{float})$$

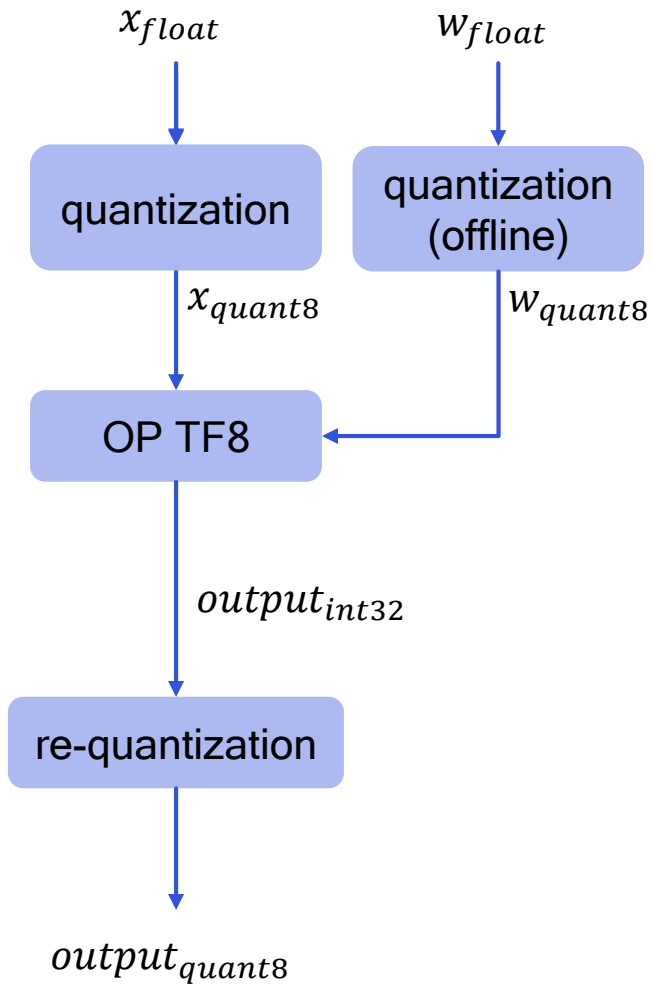
$$\doteq \Delta_x \Delta_w \sum (x_{quant8} + \delta_x)(w_{quant8} + \delta_w)$$

$$= \Delta_x \Delta_w\ accum_{int32}$$

3. Multiply the accumulated result with $\frac{\Delta_x \Delta_w}{\Delta_{output}}$, and then subtract to get re-quantized output

$$output_{quant8} = \left[\frac{1}{\Delta_{output}} accum_{float}\right] - \delta_{output}$$

$$= \left[\frac{\Delta_x \Delta_w}{\Delta_{output}} accum_{int32}\right] - \delta_{output}$$



Quantization and the problems in MobileNet V1

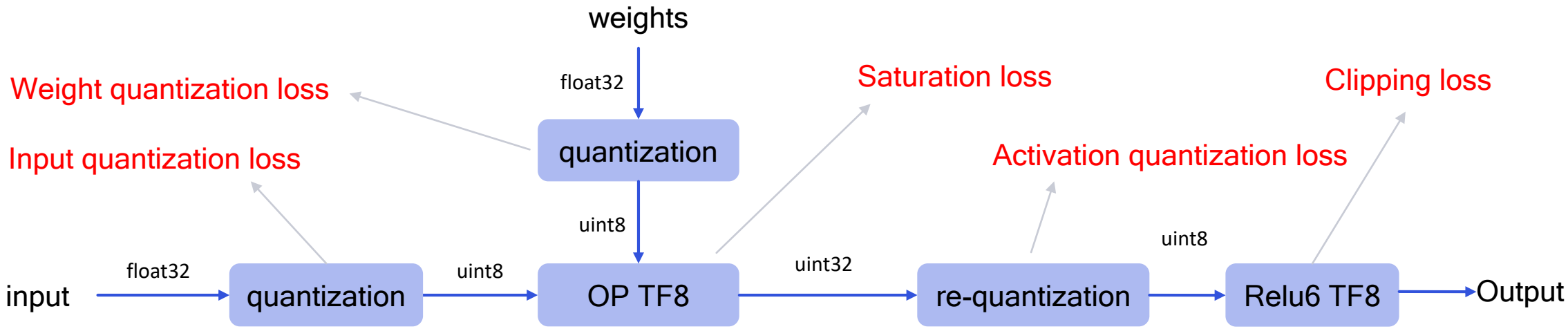
How to measure quantization loss

Signal-to-Quantization-Noise Ratio (SQNR)

A quantizer can be evaluated by its SQNR, defined as the power of the unquantized signal $\boldsymbol{x}$ divided by the power of the quantization error.

$$SQNR = 10 \cdot log_{10}\left(\frac{\sum x^2}{\sum (x - x_{quant})^2}\right) \quad in\ dB$$

- SQNR is tightly coupled with signal distribution.
- For linear quantizer, SQNR is higher when signal distribution is more uniform, and is lower when otherwise.



Quantization and the problems in MobileNet

Key of MobileNet is use of separable convolution

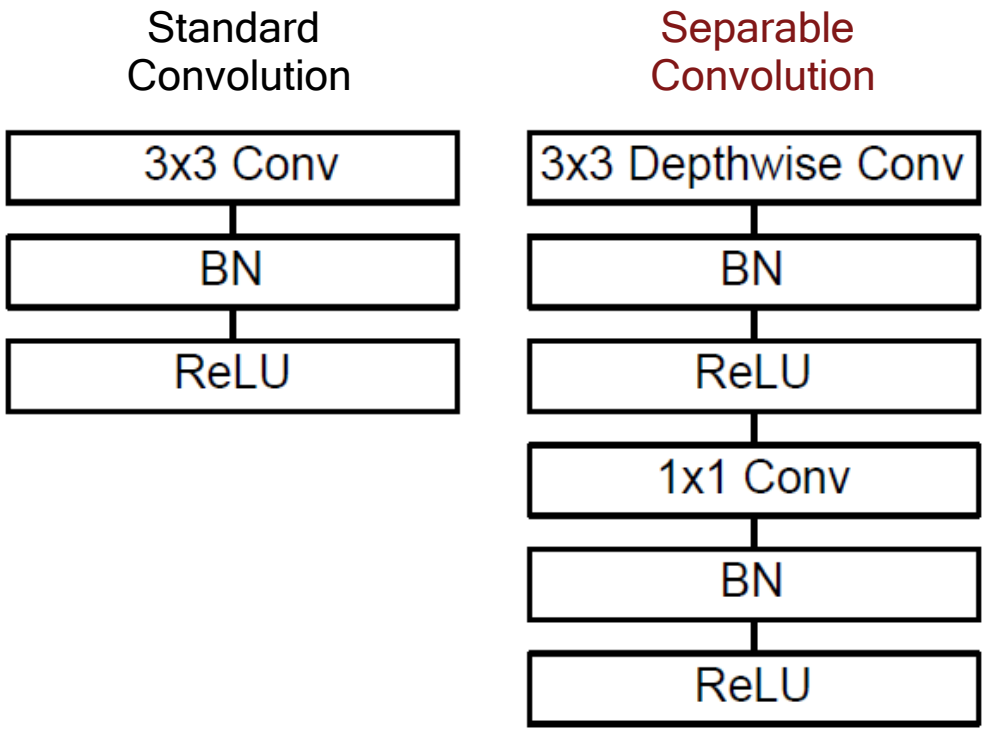


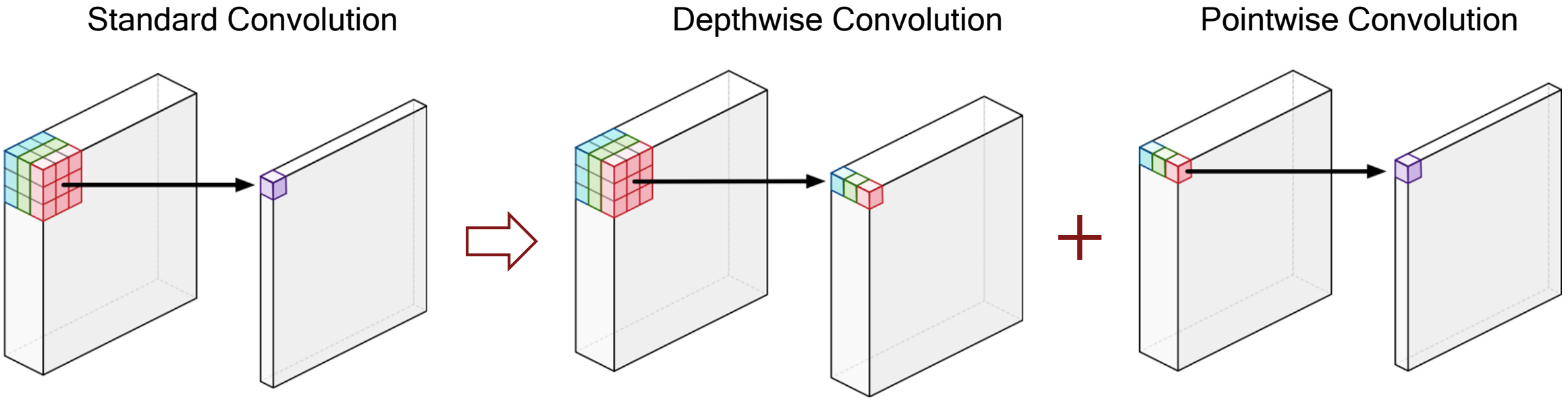
Table 1. MobileNet Body Architecture

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

"MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications", Apr17, 2017, Google, https://arxiv.org/abs/1704.04861

Quantization and the problems in MobileNet

Standard convolution vs. Separable convolution



More parameters	Much less parameters
Computation expensive	Computation efficient
Full correlation across channels	No correlation across channels + Single point correlation across channels

Images are from http://machinethink.net/blog/googles-mobile-net-architecture-on-iphone/

Quantization and the problems in MobileNet

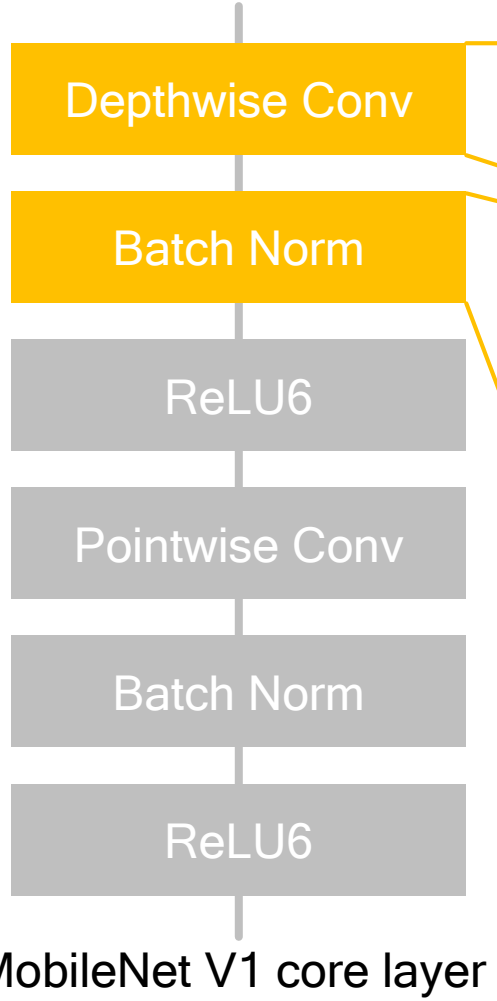
TF8 Quantization corrupts MobileNet's feature representation

Top-1 Accuracy on ImageNet 2012 validation set

	Float Pipeline	TF8 Pipeline	Comments
InceptionV3	78.00%	76.92%	Standard Convolution Only
MobileNet_v1_1.0_224	70.50%	1.80%	Mainly Separable Convolution
An example 	**white wolf 0.946334** timber wolf 0.0277307 ice bear 0.0125607 Eskimo dog 0.004571 dingo 0.00328183	window shade 0.328094 window screen 0.203663 paper towel 0.0602134 handkerchief 0.0286787 doormat 0.0232018	Float model gives the correct prediction but prediction from TF8 model is totally wrong

Quantization Loss Analysis

Large Quantization Loss caused by BatchNorm in Depthwise Conv Layer

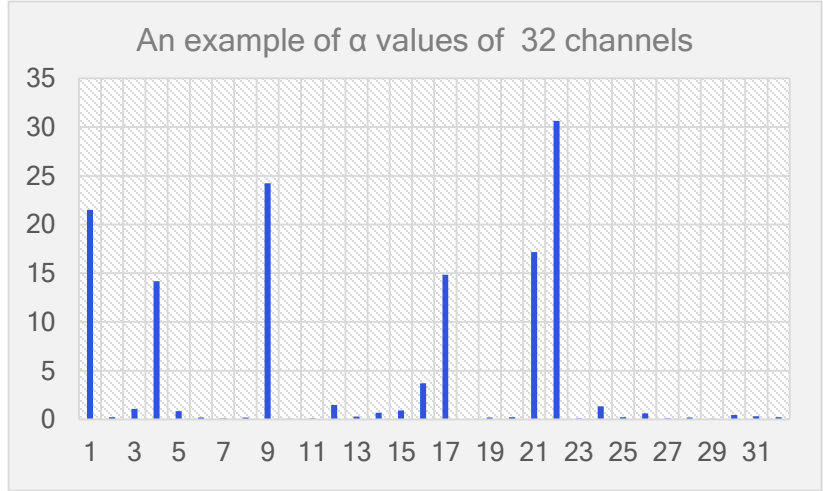


MobileNet V1 core layer

Without correlation crossing channels, depthwise convolution may prone to produce all-zero channels, which have zero variance ($\sigma = 0$).

$$y = \frac{\gamma*(x-\mu)}{\sigma} + b$$
$$= \frac{\gamma}{\sigma} * x + b - \frac{\gamma * \mu}{\sigma}$$
$$= \alpha * x + \beta$$

Zero variance introduces a very large non-uniform distribution range for α , thus dramatically increases quantization loss of layers.

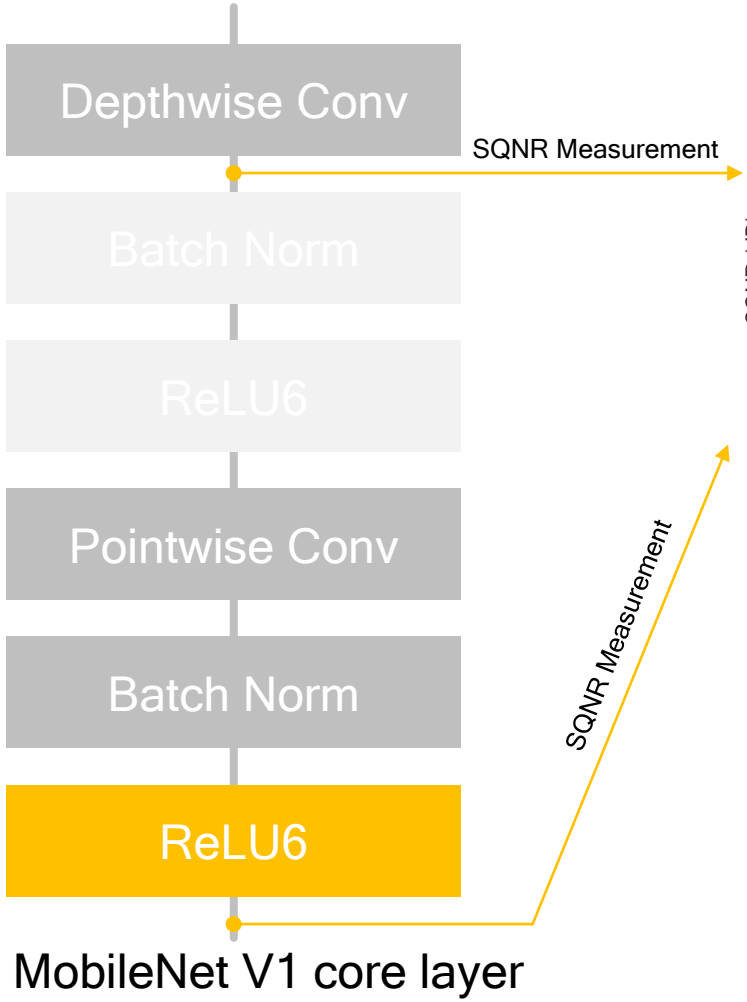


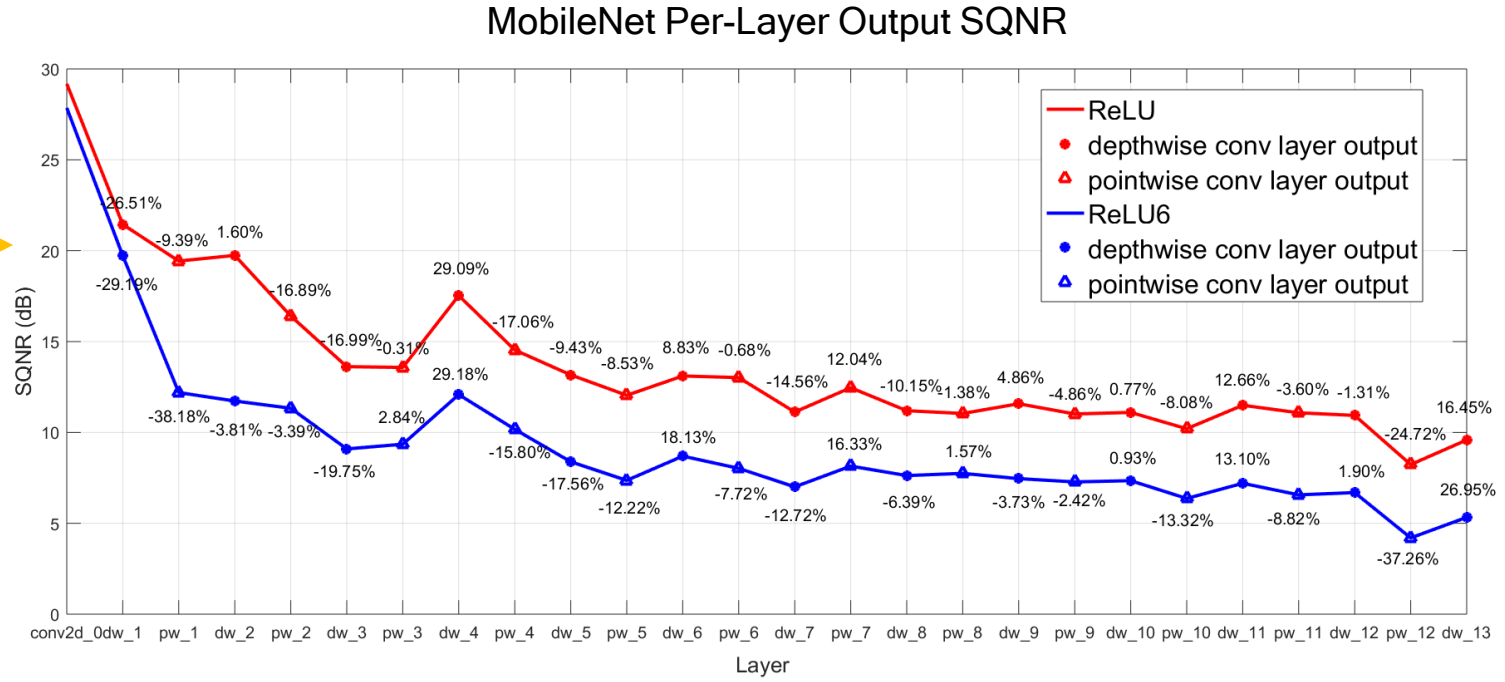
Option 1: Properly handling zero variance, the Top1 accuracy can be improved from 1.8% to ~50% without retraining.

Option 2: Removing the BN and Relu6 in depthwise conv layer, and then retrain the Mobilenet V1, the Top1 accuracy can be improved up to ~61.5%.

Quantization Loss Analysis

ReLU6, a troublemaker?

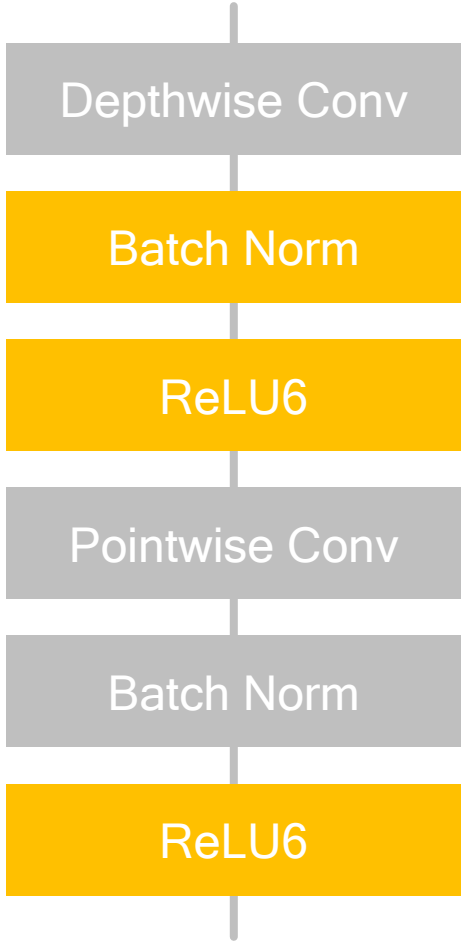




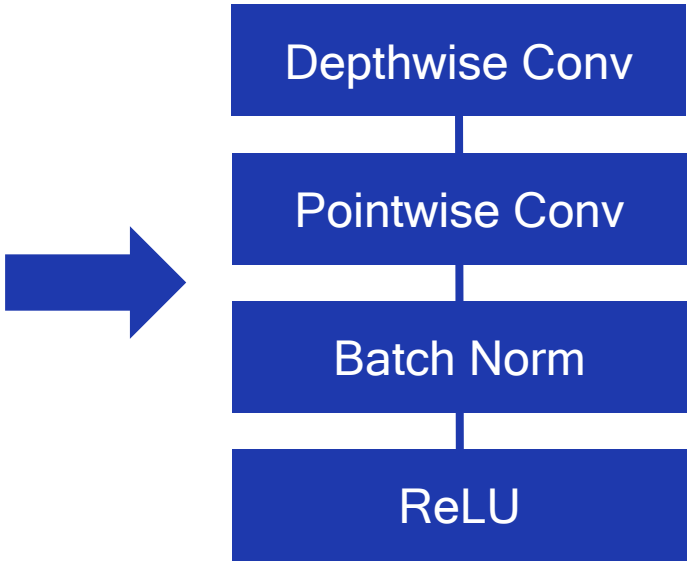
This figure shows an averaged per-layer output SQNR of MobileNet V1 by using ReLU and ReLU6 as different activation functions. 1000 images are randomly selected from ImageNet validation set (one in each class). A huge SQNR drop is observed in the first pointwise convolution layer while using ReLU6. From our experiments, by simply replacing ReLU6 with ReLU, the top-1 accuracy on ImageNet dataset can be increased from 61.50% to 67.80%.

A Quantization-friendly Model

Re-architect the separable convolution to make it quantization friendly



MobileNet core layer



Our quantization-friendly core layer

- Remove Batch Normalization (BN) and ReLU6 from depthwise convolution layer
- Replace the ReLU6 by ReLU in first Conv2d_0 layer and all the pointwise convolution layers
- Enable the L2-Regularization on the depthwise convolution weights

Download: https://github.qualcomm.com/WoT/MobileNet_Models

Experimental Results

The fixed-point inferencing performance is very close to float-point inferencing

Models	Dataset	Top-1 Accuracy (mAP)	
		Float Pipeline	TF8 Pipeline
MobileNet 1.0 Slim	ImageNet2012	70.50%	1.80%
Remove BN and ReLU6 in Depthwise Conv		70.55%	61.50%
Replace all ReLU6 by ReLU		70.80%	67.80%
Enable L2-Regularization on Depthwise weights		70.77%	68.03%

Qualcomm

Thank you!

Follow us on:

For more information, visit us at:

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