Quantization of neural networks

Alexander Fritzler

How to make inference of the neural network?

GPU

NPU

How to make inference of the neural network?

GPU

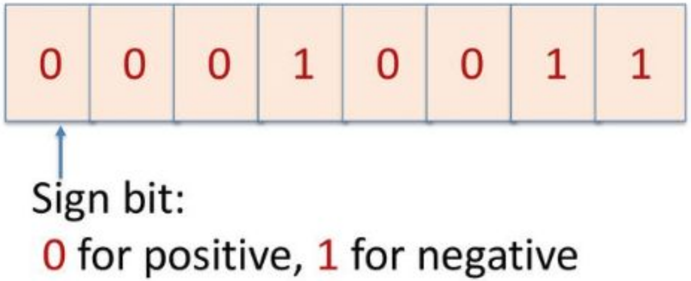
float32

NPU

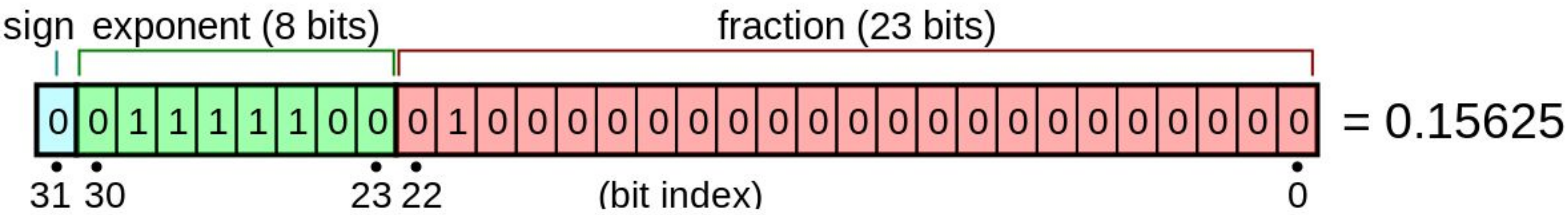
int8

How numbers are stored in memory?

int8



float32



The problem: the quality falls significantly

How to fix it?

Let’s consider a vector

Vector of floats

w

float32

Vector of numbers from **0** to **255**

How to transfer it?

Let’s consider a vector

Asymmetric quantization

We need: scale-parameter (Δ) and a zero-point (z)

Quantization operation

$$x_{int} = round(\frac{x}{\Delta}) + z$$

$$x_Q = clamp(0, N_{levels} - 1, x_{int})$$

Dequantization operation

$$x_{float} = (x_Q - z)\Delta$$

Let's consider a vector

Symmetric quantization

We need: scale-parameter (Δ)

Quantization operation

$$x_{int} = round(\frac{x}{\Delta})$$

$$x_Q = clamp(-N_{levels}/2, N_{levels}/2 - 1, x_{int}) \quad \text{if signed}$$

$$x_Q = clamp(0, N_{levels} - 1, x_{int}) \quad \text{if un-signed}$$

Dequantization operation

$$x_{out} = x_Q \Delta$$

How to choose the parameters?

It's easy: based on the distribution

How do we quantize?

Weights

Based on the minimum and maximum weight

We can do
1) Per-layer quantization
2) Per-channel (per-neuron) quantization

Activations

Requires data.

Based on the minimum and maximum (or some quantiles) of activations.

Quantized convolution

Naive:

$$y(k,l,n) = \Delta_w \Delta_x conv(w_Q(k,l,m;n) - z_w, x_Q(k,l,m) - z_x)$$

Advanced:

$$y(k,l,n) = conv(w_Q(k,l,m;n), x_Q(k,l,m)) - z_w \sum_{k=0}^{K-1} \sum_{l=0}^{K-1} \sum_{m=0}^{N-1} x_Q(k,l,m)$$
$$- z_x \sum_{k=0}^{K-1} \sum_{l=0}^{K-1} \sum_{m=0}^{N-1} w_Q(k,l,m;n) + z_x z_w$$

https://github.com/google/gemmlowp/blob/master/doc/quantization.md#implementation-of-quantized-matrix-multiplication

Post-training quantization pipeline

Step 1:

Train a full-sized model

Step 2:

Inserting observers, calibrate the model

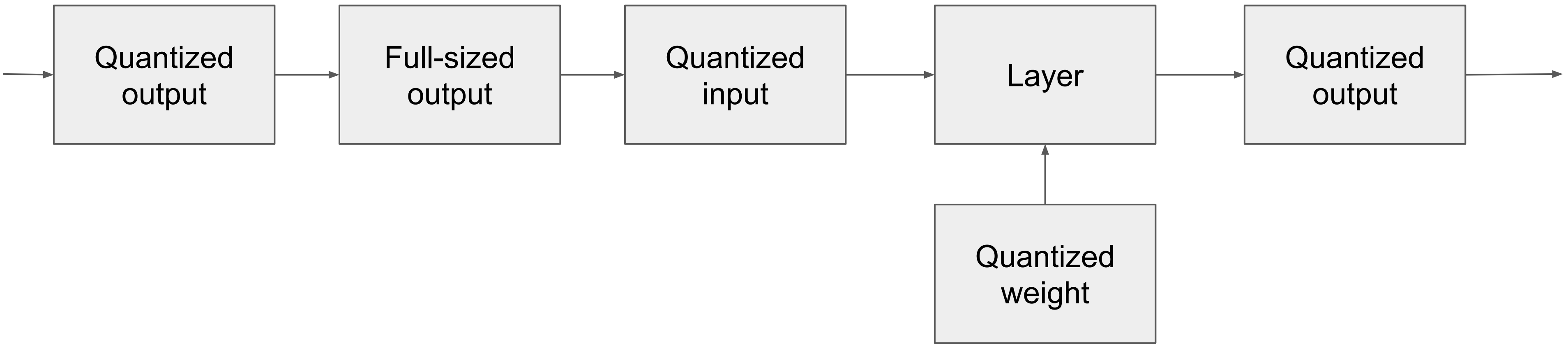
Step 3:

Compute the parameters of the quantization

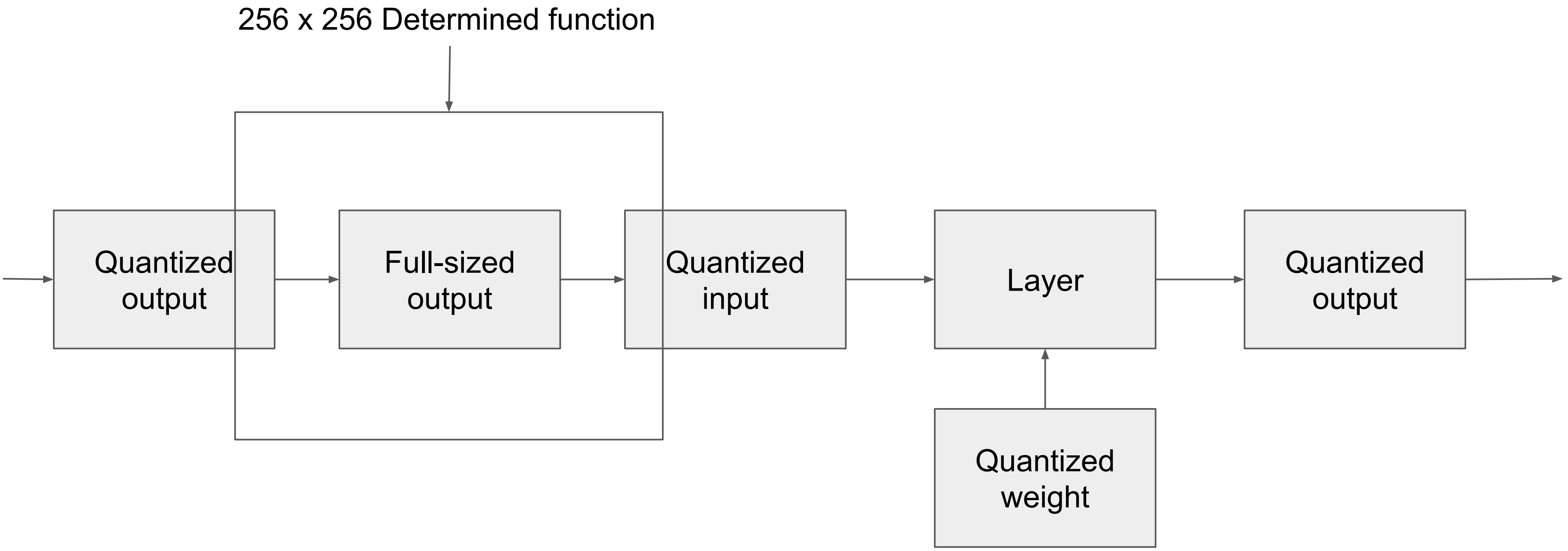
Step 4:

Inference of the quantized model

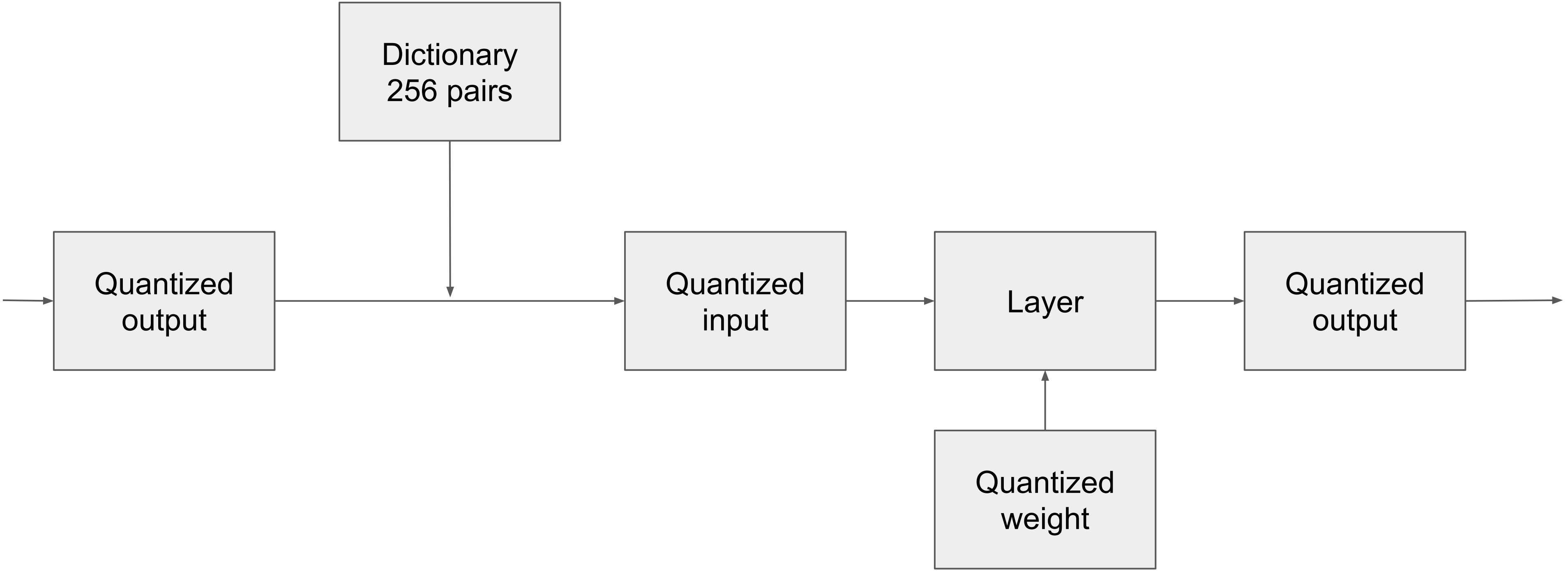
Quantized inference: theory and practice



Quantized inference: theory and practice



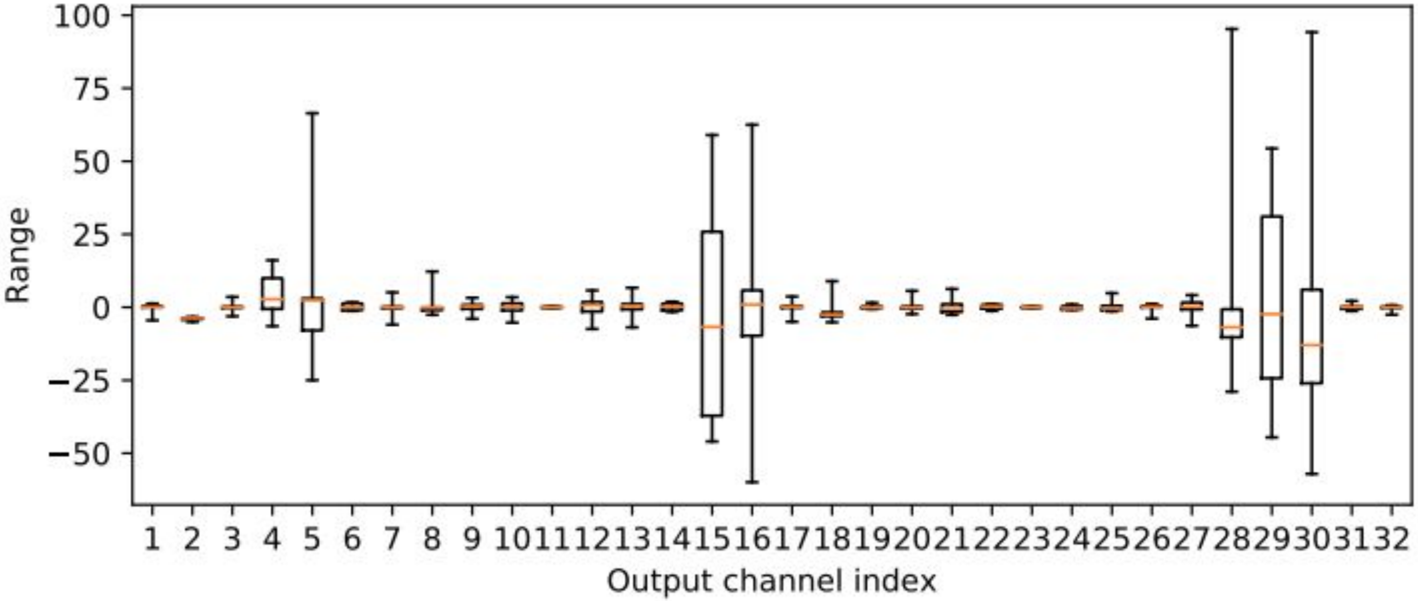
Quantized inference: theory and practice



Results

Network	Asymmetric, per-layer	Symmetric , per-channel	Asymmetric, per-channel	Floating Point
Mobilenetv1_1_224	0.001	0.591	0.704	0.709
Mobilenetv2_1_224	0.001	0.698	0.698	0.719
NasnetMobile	0.722	0.721	0.74	0.74
Mobilenetv2_1.4_224	0.004	0.74	0.74	0.749
Inceptionv3	0.78	0.78	0.78	0.78
Resnet_v1_50	0.75	0.751	0.752	0.752
Resnet_v2_50	0.75	0.75	0.75	0.756
Resnet_v1_152	0.766	0.763	0.762	0.768
Resnet_v2_152	0.761	0.76	0.77	0.778

Weights distribution



Bias correction

$\widetilde{\mathbf{y}} = \widetilde{\mathbf{W}}\mathbf{x}$ - **noised output**

$\widetilde{\mathbf{y}} = \mathbf{y} + \epsilon\mathbf{x}$ **, where** $\epsilon = \widetilde{\mathbf{W}} - \mathbf{W}$

$\mathbb{E}[\epsilon\mathbf{x}]_i \neq 0$ → The expectation of the output will be different

Solution: Compute $\mathbb{E}[\epsilon\mathbf{x}]$ empirically using data and subtract it from bias

Results

Model	MobileNetV2 SSD-lite	DeeplabV3+ (MobileNetV2 backend)
Original model	10.63	41.40
DFQ (ours)	**67.91**	**72.33**
Per-channel quantization	67.52	71.44
Original model (**FP32**)	68.47	72.94

Can we do better?

- Can we make the gap between the quantized model and full-sized model even smaller?

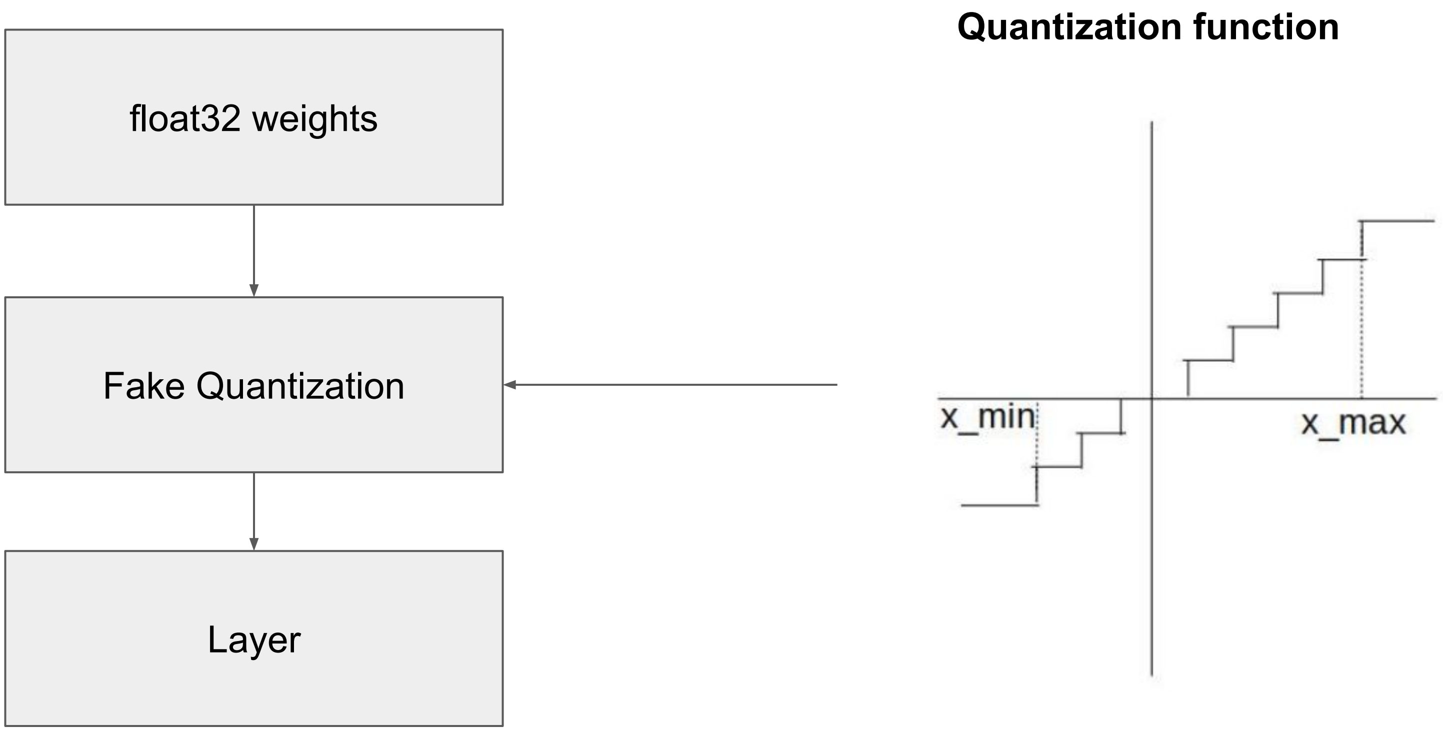
- Some models do not perform properly after the quantization

- Can we **train quantized model on our computer?**

“Quantization-Aware Training”

- How the model is stored?
- How do we do forward pass?
- How do we do backpropagation?
- How we choose the parameters of the quantization?

How the model is stored

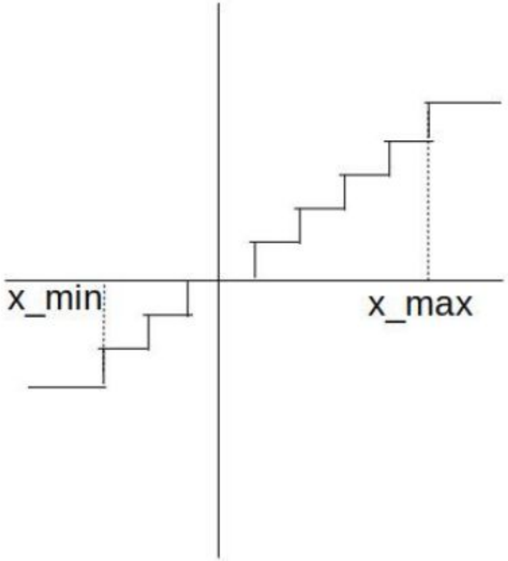


How do we do forward pass (with Pytorch)

	Works on GPU	8-bit computations	8-bit weights and activations
nn.quantized.modules.linear.Linear (nn.quantized.modules.conv.Conv2d)			
nn.qat.modules.linear.Linear (nn.qat.modules.conv.Conv2d)			

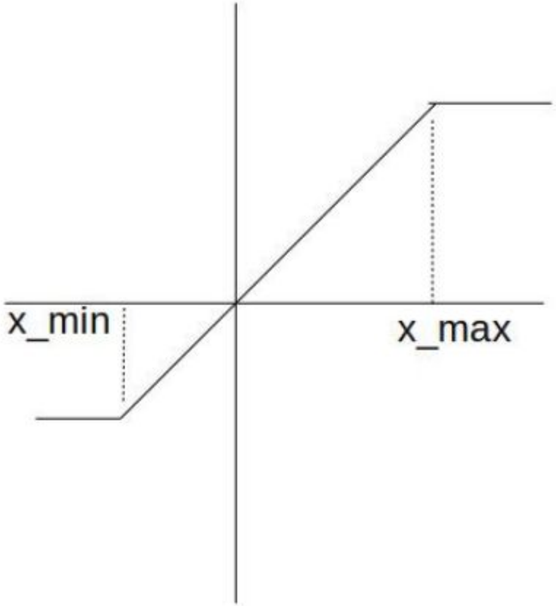
How do we do backpropagation

Quantization function (forward)



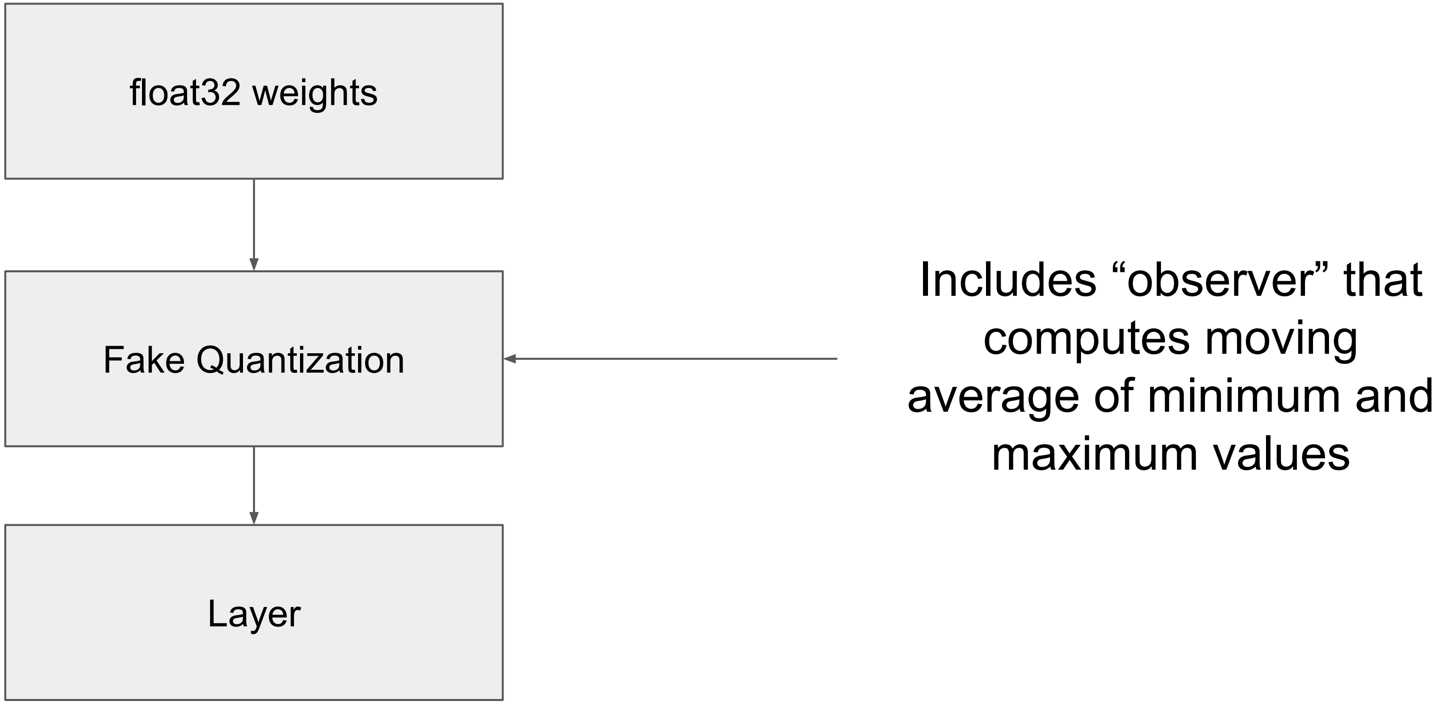
Derivative = 0

Quantization function (backward)



Derivative = const

How we choose the parameters of the quantization



Comparison of the inference time

Network \ Inference Platform	Floating point(CPU)	Fixed point (CPU)	Fixed point (HVX, NN-API)
Mobilenet_v1_1_224	155	68	16
Mobilenet_v2_1_224	105	63	15.5
Mobilenet_v1_1_224_SSD	312	152	
Inception_v3	1391	536	
Resnet_v1_50	874	440	
Resnet_v2_50	1667	1145	
Resnet_v1_152	2581	1274	
Resnet_v2_152	4885	3240	

References

- Raghuraman Krishnamoorthi. Quantizing deep convolutional networks for efficient inference: A whitepaper. arXiv preprint arXiv:1806.08342, Jun 2018.

- Markus Nagel, Mart van Baalen, Tijmen Blankevoort, and Max Welling. Data-free quantization through weight equalization and bias correction. arXiv preprint arXiv:1906.04721, 2019

- B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, “Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference,” Dec. 2017.

- https://pytorch.org/docs/stable/quantization.html