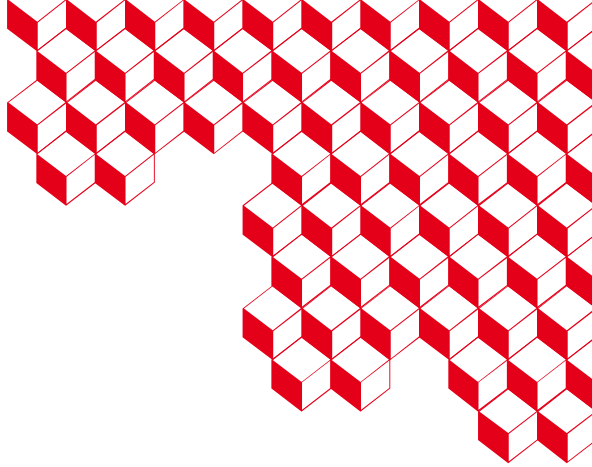


Quantized Neural Networks for edge implementation

Yannick Malot

JRAF 2023

Introduction

DNN, powerful but heavy models

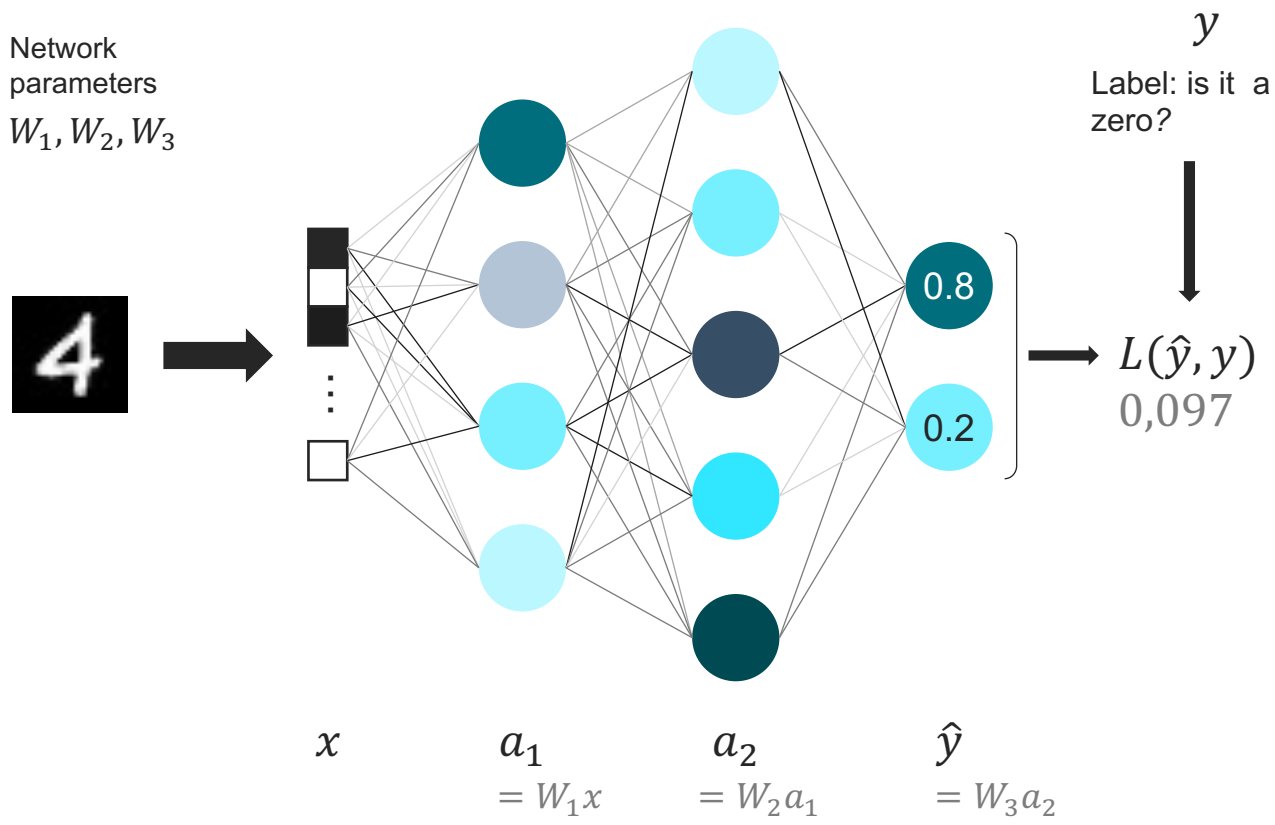
A neural network is a nesting of matrix products (parameters)

Parameters:

- Numerous (deep networks)
- In high dimension

⇒ Important storage need

Scheme – Functioning of a neural network



Compress a DNN

Needed memory > memory available on edge devices

⇒ need for **compression**

Several possibilities:

- fewer parameters, smarter operations;
- pruning: remove nodes;
- **quantization**: reduce numerical precision of parameters

Model	Memory (parameters)
ResNet-18	45 MB
ResNet-50	98 MB
MobileNet-v2	14 MB
EfficientNet-b0	21 MB
Yolo v7	148 MB
RetinaNet	152 MB
GPT3-small	16 GB

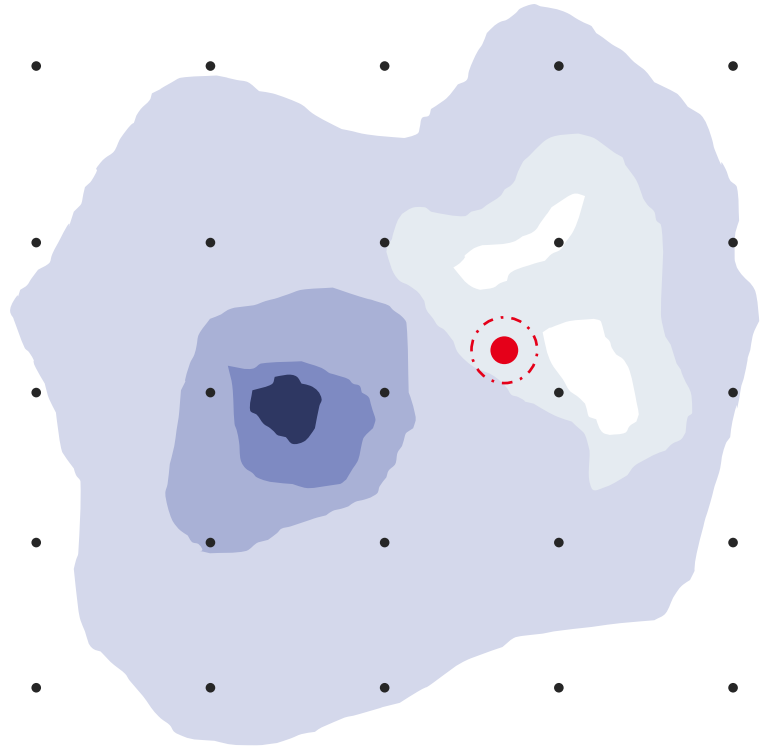
source: torchvision documentation

Quantization

A parameter is quantized if it lies in a **discrete** and **finite** set (vs. continu et infini)

- Quantization of a trained network (*Post Training Quantization*)
- Training iteratively with quantized weights (*Quantization Aware* Training)
- Integer-only training: parameters, activatoins and gradients

Scheme – Optimization of a function of a continuous parameter



1. Quantization Aware Training

Concepts and *benchmark*

Quantization Aware Training

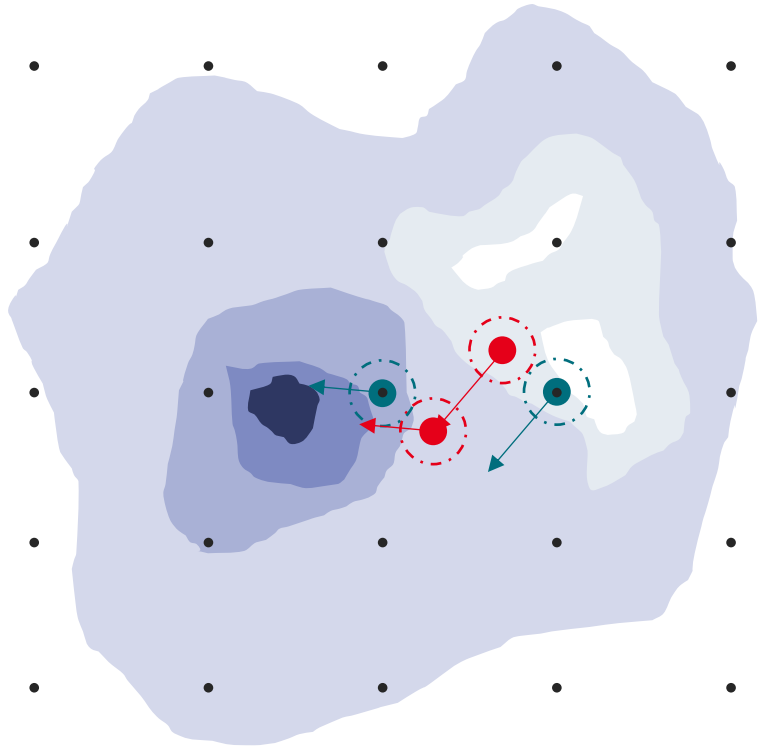
- Latent parameter

- Quantized parameter

  (iterative)

- Learning signal: gradient at the quantized point

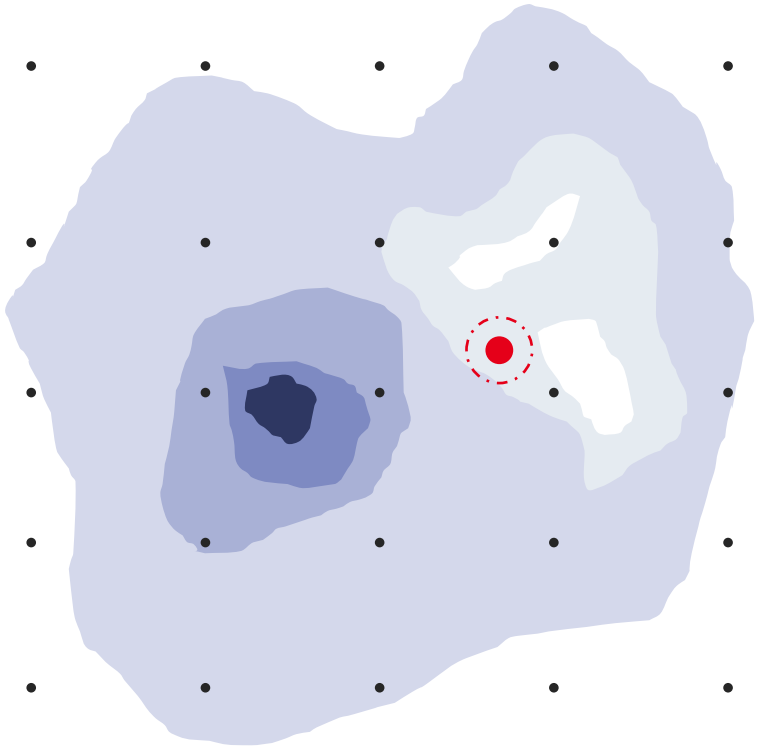
Scheme – QAT Learning



Diversity of QAT approaches

- Discretization set
  - Linear / uniform
  - Logarithmic
  - Any centroids

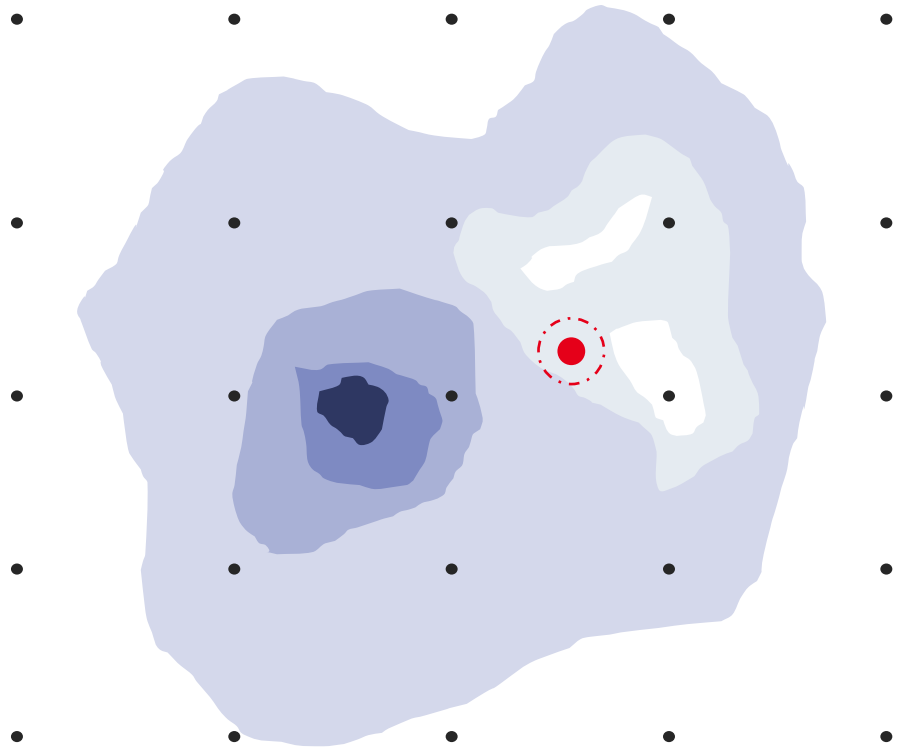
Scheme – QAT Learning



Diversité des approches

- Discretization set
  - Linear / uniform
  - Logarithmic
  - Any centroids

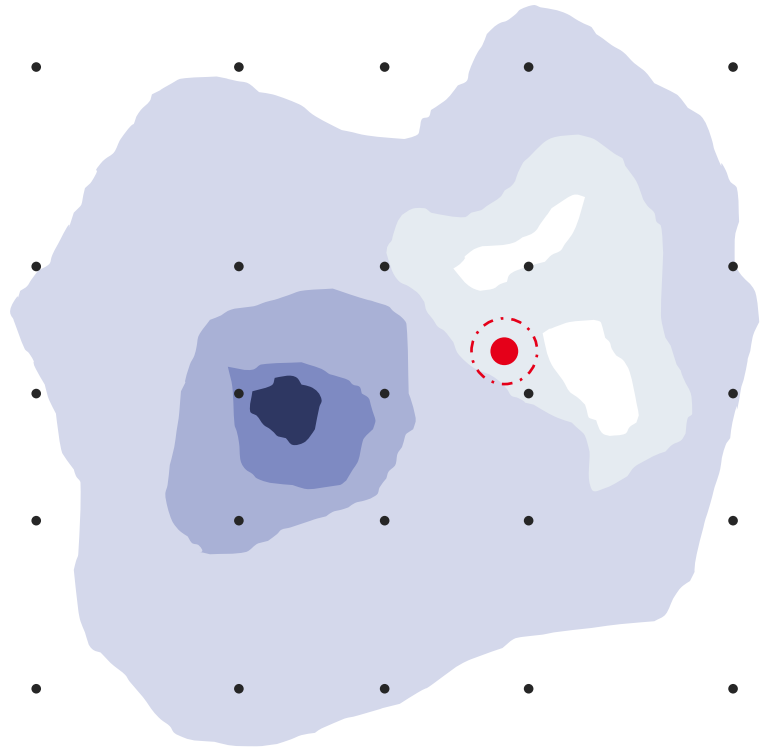
Scheme – QAT Learning



Diversité des approches

- Discretization set
  - Linear / uniform
  - Logarithmic
  - Any centroids

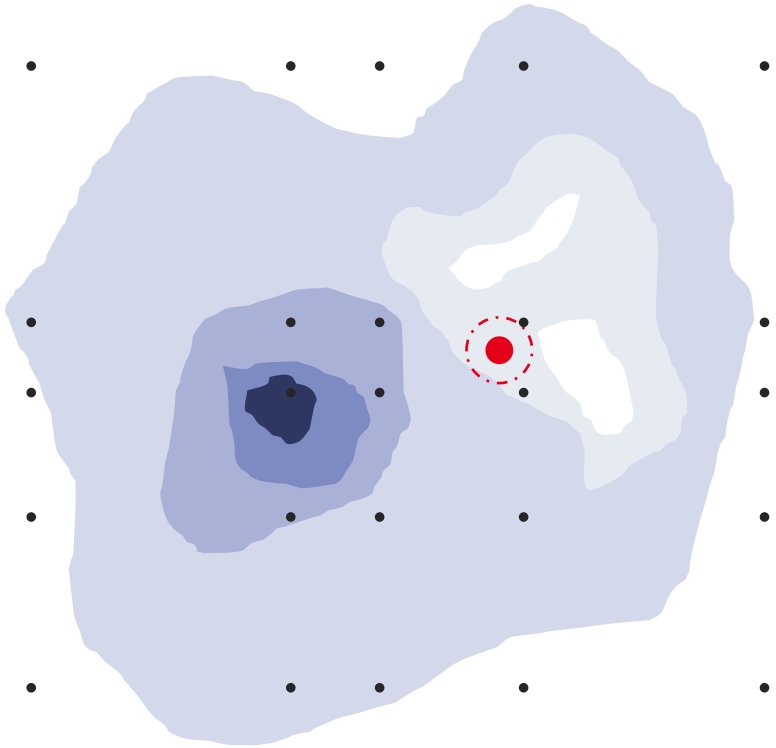
Scheme – QAT Learning



Diversité des approches

- Discretization set
  - Linear / uniform
  - Logarithmic
  - Any centroids

Scheme – QAT Learning



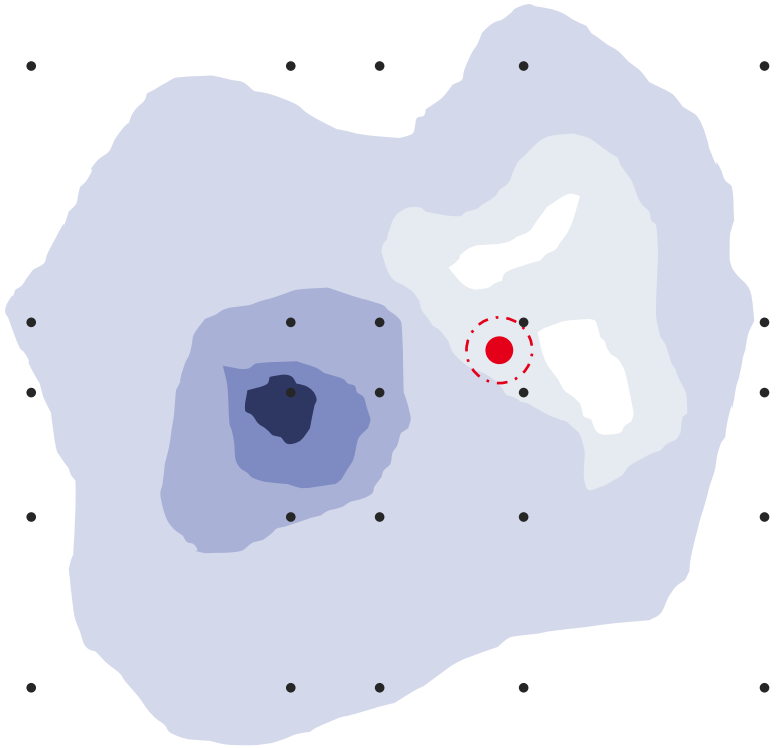
Diversité des approches

- Discretization set
  - Linear / uniform
  - Logarithmic
  - Any centroids

- Additional parameters
  - Scaling
  - Learnt discretization set

More complex approaches = additional settings to tune

Scheme – QAT Learning



QAT benchmark

Linear approaches can reach 90% accuracy, while more complicated approaches do not necessarily perform better

Test classification error on test data (W3A3)

ResNet20 network on CIFAR-10 data. Best epoch selected on 5% validation data, averaged over 10 runs. Optimizer SGD, lr 10^{-1}, 200 epochs

Méthode	**Taux d'erreur moyen**
Baseline (32 bits)	8.5%
Binarized NN (1 bit)	34%
Ternary Weight Net (2 bits)	27%
Quantized NN	19.5%
Learned Step Quantization	10%
DoReFa Net	12%
Scale Adjusted Training	11%
LQ-Net	11.5%
Scalar centroids	13%

2. Numerical precision

Which should we chose?

Number of parameters and memory

Model with N_p parameters and an activation with maximum size N_a

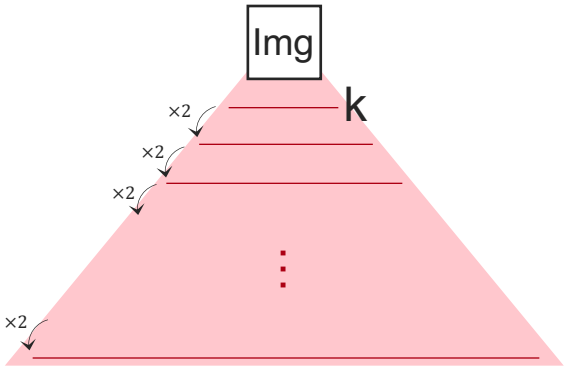
Memory needed for an inference:

parameters b_p bits, activations b_a bits

$$Mem = N_p \times b_p + N_a \times b_a$$

$$N_p = f(k)$$
$$N_a = g(k)$$

$$b_a = \begin{cases} b_p & if\ b_p > 3 \\ 4 & else \end{cases}$$



Number of parameters and memory

Model with N_p parameters and an activation with maximum size N_a

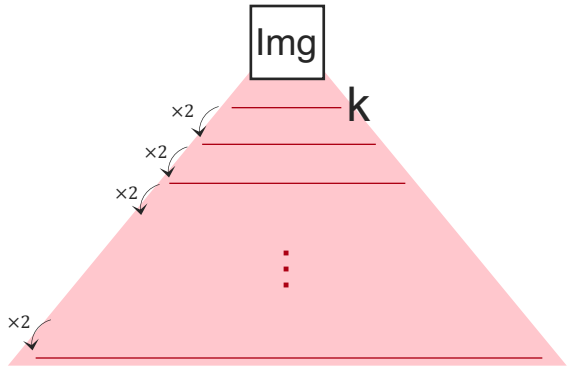
Memory needed for an inference:

parameters b_p bits, activations b_a bits

$$N_p = f(k)$$
$$N_a = g(k)$$

$$b_a = \begin{cases} b_p & if\ b_p > 3 \\ 4 & else \end{cases}$$

$$Mem = f(k) \times b_p + g(k) \times \max(b_p, 4)$$



Number of parameters and memory

Model with N_p parameters and an activation with maximum size N_a

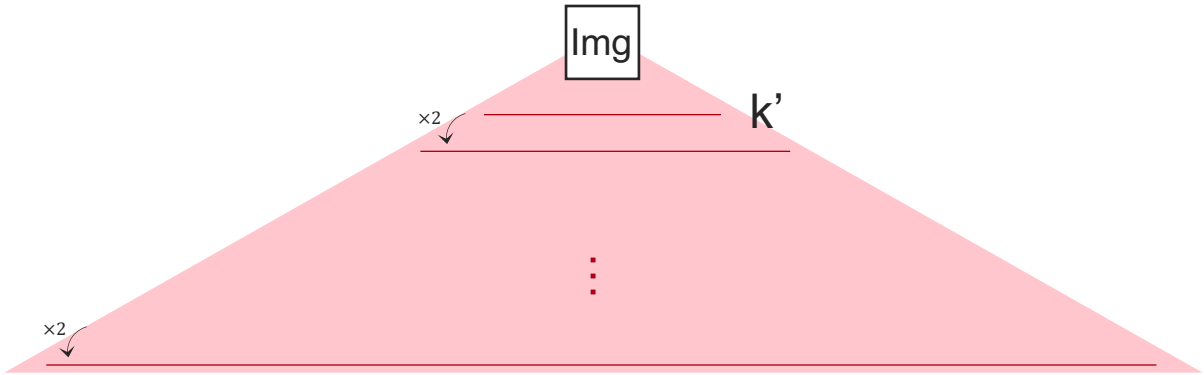
Memory needed for an inference:

parameters b_p bits, activations b_a bits

$$N_p = f(k)$$
$$N_a = g(k)$$

$$b_a = \begin{cases} b_p & if\ b_p > 3 \\ 4 & else \end{cases}$$

$$Mem = f(k) \times b_p + g(k) \times \max(b_p, 4)$$



Tradeoff #parameters - precision

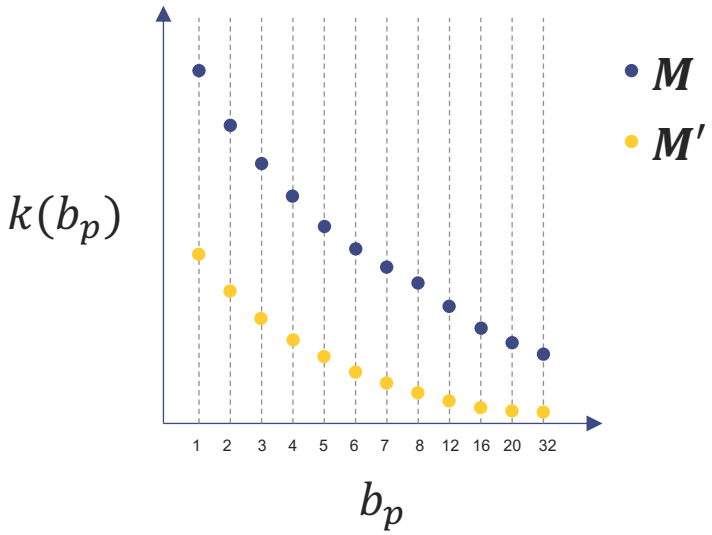
Fixed computational budget $\boldsymbol{M}$

Relation between the size of first convolutional filter and numerical precision:

$$k(b_p, \boldsymbol{M}) = \underset{k}{\operatorname{argmax}}\, Mem(k, b_p)$$
$$t.q.\ \ Mem(k, b_p) \leq \boldsymbol{M}$$

« take as many parameters as the memory budget $\boldsymbol{M}$ allows for with precision b_p »

$$Mem(k, b_p) = f(k) \times b_p + g(k) \times \max(b_p, 4)$$



Optimal numerical precision: 3-4 bits ?

Model: ResNet 20, CIFAR-10 data

Baseline memory of the model: $\boldsymbol{M_0}$

Observe model accuracy when the numerical precision varies (jointly with the number of parameters)

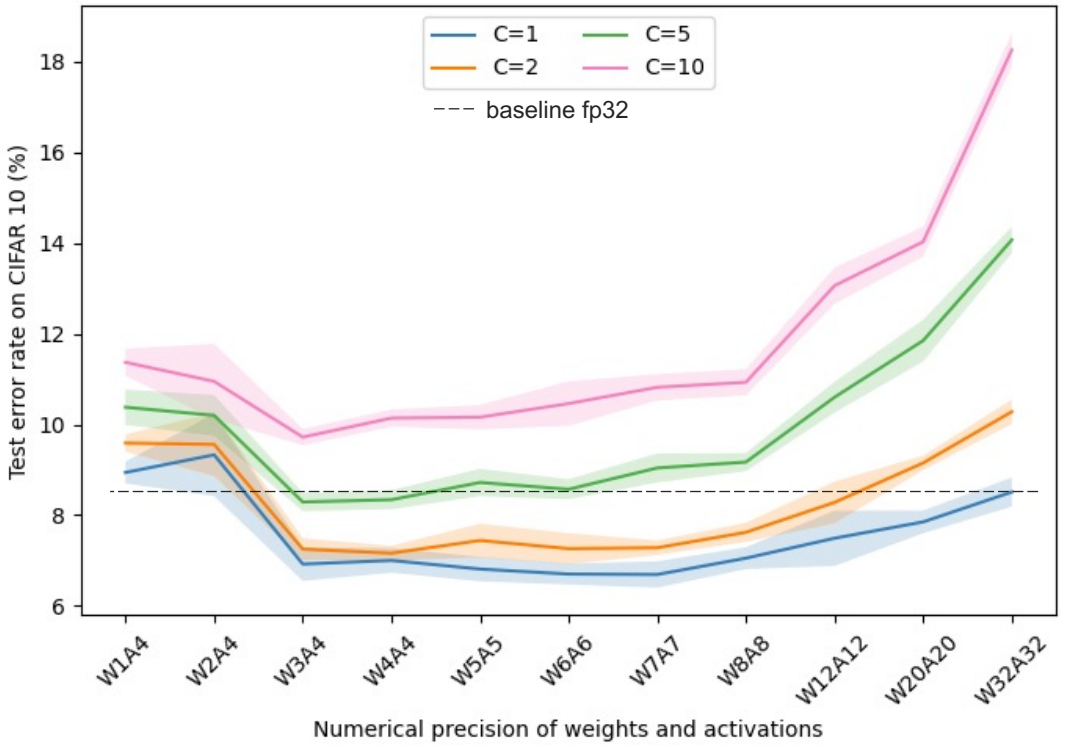
with memory budget $\boldsymbol{M_0}/C$

for given compression ratios C

Test classification error on test data (W3A3)

ResNet20 with fixed memory budget, for different compression ratios

CIFAR-10 data; Optimizer SGD, lr 10^{-1}, 200 epochs



3. Towards on-device training

Contraints for on-device training

more elements to keep in memory

In inference:

- quantized weights (*b* bits)
- activation (intermediary calculation, *b'* bits)

During training:

- latent parameters (32 bits)
- optimizer (32 bits)
- all activations (*b'* bits)

Table: Numerical precision of each element and needed memory
ResNet-20, CIFAR-10 data, batch size B

W32A32	Inférence	Mémoire (Mb)	Apprentissage	Mémoire (Mb)
Activations	+ grande	0,07	toutes	0,77 × B
Paramètres	tous	1,08	tous	1,08
Optimizer	N.A.	0	~paramètres	1,08
Total		**1,15**		**2,16 + 0,77 B**

W3A3	Inférence	Mémoire (Mb)	Apprentissage	Mémoire (Mb)
Activations	+ grande	0,006	toutes	0,07 × B
Paramètres	tous	0,10	Tous (32 bits)	1,08
Optimizer	N.A.	0	~paramètres	1,08
Total		**0,11**		**2,16 + 0,07 B**

All-integer training

Works well in 8 bits (see NITI)

How to go down to 4 bits?

- keep gradients in 8 bits
- random jumps from state to others

Figure: All-integer training
From NITI

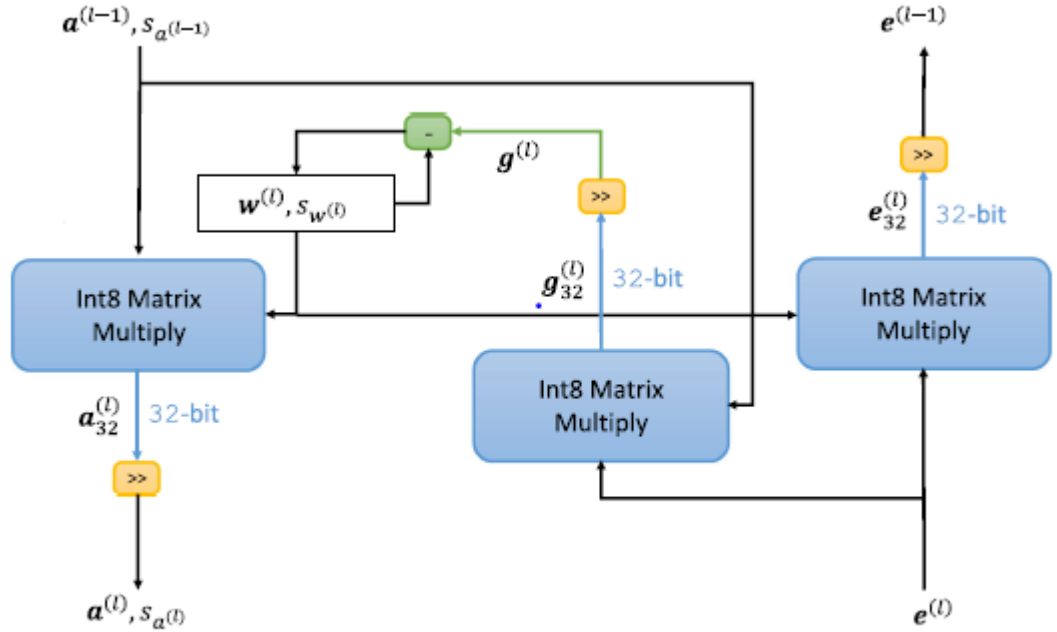


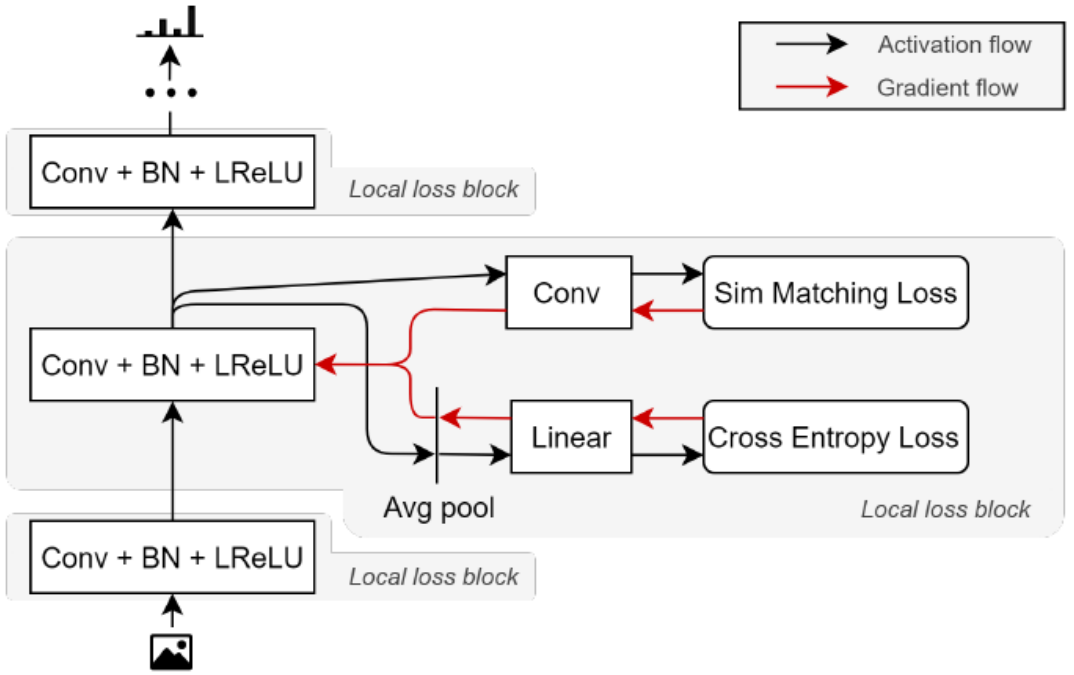
Fig. 1. Integer layer forward and backward pass.

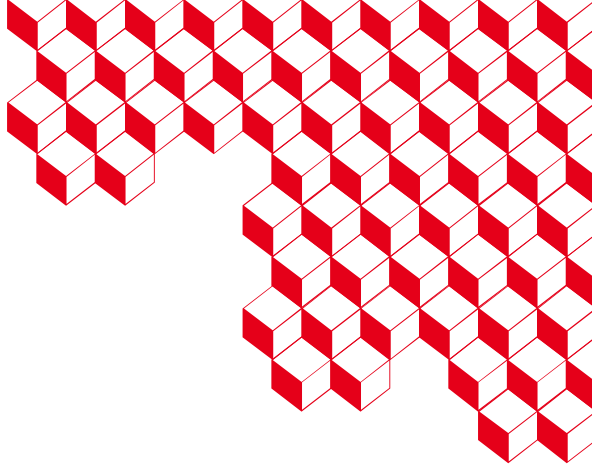
Local loss

Training with blocks

- No complete backpropagation
- QAT on the block being trained
- So far, 30% test error on Cifar-10

Figure: Training with local error signals
From Nokland et al.





Q&R

Direction: Kim Thang Nguyen (LIG)

Supervision: Thomas Mesquida (CEA), Sylvain Bouveret (LIG)

References 1/3

[1] Y. Li, X. Dong, et W. Wang, « Additive Powers-of-Two Quantization: An Efficient Non-uniform Discretization for Neural Networks », arXiv, arXiv:1909.13144, févr. 2020. Consulté le: 15 mai 2023. [En ligne]. Disponible sur: http://arxiv.org/abs/1909.13144

[2] P. Stock, A. Joulin, R. Gribonval, B. Graham, et H. Jégou, « And the Bit Goes Down: Revisiting the Quantization of Neural Networks », arXiv, arXiv:1907.05686, nov. 2020. Consulté le: 15 mai 2023. [En ligne]. Disponible sur: http://arxiv.org/abs/1907.05686

[3] S. Oh, H. Sim, S. Lee, et J. Lee, « Automated Log-Scale Quantization for Low-Cost Deep Neural Networks », in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Nashville, TN, USA: IEEE, juin 2021, p. 742-751. doi: 10.1109/CVPR46437.2021.00080.

[4] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, et Y. Bengio, « Binarized Neural Networks: Training Deep Neural Networks with Weights and Activations Constrained to +1 or -1 », arXiv, arXiv:1602.02830, mars 2016. Consulté le: 15 mai 2023. [En ligne]. Disponible sur: http://arxiv.org/abs/1602.02830

[5] R. Gong *et al.*, « Differentiable Soft Quantization: Bridging Full-Precision and Low-Bit Neural Networks », in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, Seoul, Korea (South): IEEE, oct. 2019, p. 4851-4860. doi: 10.1109/ICCV.2019.00495.

[6] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, et Y. Zou, « DoReFa-Net: Training Low Bitwidth Convolutional Neural Networks with Low Bitwidth Gradients », arXiv, arXiv:1606.06160, févr. 2018. Consulté le: 15 mai 2023. [En ligne]. Disponible sur: http://arxiv.org/abs/1606.06160

[7] A. Zhou, A. Yao, Y. Guo, L. Xu, et Y. Chen, « INCREMENTAL NETWORK QUANTIZATION: TOWARDS LOSSLESS CNNS WITH LOW-PRECISION WEIGHTS », 2017.

[8] S. K. Esser, J. L. McKinstry, D. Bablani, R. Appuswamy, et D. S. Modha, « Learned Step Size Quantization », arXiv, arXiv:1902.08153, mai 2020. Consulté le: 15 mai 2023. [En ligne]. Disponible sur: http://arxiv.org/abs/1902.08153

References 2/3

[9] S. Jung *et al.*, « Learning to Quantize Deep Networks by Optimizing Quantization Intervals With Task Loss », in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA: IEEE, juin 2019, p. 4345-4354. doi: 10.1109/CVPR.2019.00448.

[10] D. Zhang, J. Yang, D. Ye, et G. Hua, « LQ-Nets: Learned Quantization for Highly Accurate and Compact Deep Neural Networks », in *Computer Vision – ECCV 2018*, vol. 11212, V. Ferrari, M. Hebert, C. Sminchisescu, et Y. Weiss, Éd., in Lecture Notes in Computer Science, vol. 11212. , Cham: Springer International Publishing, 2018, p. 373-390. doi: 10.1007/978-3-030-01237-3_23.

[11] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, et Y. Bengio, « Quantized Neural Networks: Training Neural Networks with Low Precision Weights and Activations ».

[12] B. Liu, F. Li, X. Wang, B. Zhang, et J. Yan, « Ternary Weight Networks », in *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Rhodes Island, Greece: IEEE, juin 2023, p. 1-5. doi: 10.1109/ICASSP49357.2023.10094626.

[13] Q. Jin, L. Yang, et Z. Liao, « Towards Efficient Training for Neural Network Quantization », arXiv, arXiv:1912.10207, déc. 2019. Consulté le: 15 mai 2023. [En ligne]. Disponible sur: http://arxiv.org/abs/1912.10207

[14] C. Zhu, S. Han, H. Mao, et W. J. Dally, « Trained Ternary Quantization », arXiv, arXiv:1612.01064, févr. 2017. Consulté le: 15 mai 2023. [En ligne]. Disponible sur: http://arxiv.org/abs/1612.01064

[15] A. Fan *et al.*, « Training with Quantization Noise for Extreme Model Compression », arXiv, arXiv:2004.07320, févr. 2021. Consulté le: 15 mai 2023. [En ligne]. Disponible sur: http://arxiv.org/abs/2004.07320

[16] C. Baskin *et al.*, « UNIQ: Uniform Noise Injection for Non-Uniform Quantization of Neural Networks », *ACM Trans. Comput. Syst.*, vol. 37, nº 1-4, p. 1-15, nov. 2019, doi: 10.1145/3444943.

[17] M. Rastegari, V. Ordonez, J. Redmon, et A. Farhadi, « XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks », in *Computer Vision – ECCV 2016*, vol. 9908, B. Leibe, J. Matas, N. Sebe, et M. Welling, Éd., in Lecture Notes in Computer Science, vol. 9908. , Cham: Springer International Publishing, 2016, p. 525-542. doi: 10.1007/978-3-319-46493-0_32.

References 2/3

[18] K. He, X. Zhang, S. Ren, et J. Sun, « Deep Residual Learning for Image Recognition », arXiv, arXiv:1512.03385, déc. 2015. Consulté le: 22 juin 2023. [En ligne]. Disponible sur: http://arxiv.org/abs/1512.03385

[19] M. Wang, S. Rasoulinezhad, P. H. W. Leong, et H. K.-H. So, « NITI: Training Integer Neural Networks Using Integer-Only Arithmetic », *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, nº 11, p. 3249-3261, nov. 2022, doi: 10.1109/TPDS.2022.3149787.

[20] A. Nøkland et L. H. Eidnes, « Training Neural Networks with Local Error Signals », arXiv, arXiv:1901.06656, mai 2019. Consulté le: 17 octobre 2022. [En ligne]. Disponible sur: http://arxiv.org/abs/1901.06656