

How to lighten AI models

Tricks, Theory, Measures and Hardware

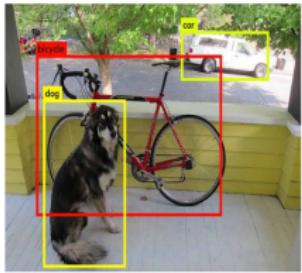
Chaigneau Yanis and Loustau Sébastien

GreenAI U.P.P.A.

20-06-2022



Today's AI



General principle of today's AI : Machine Learning

$x \longrightarrow \boxed{\text{nature}} \longrightarrow y$

General principle of today's AI : Machine Learning

$$x \longrightarrow \boxed{\text{nature}} \longrightarrow y$$

- predict response y given x ,
- describe $x \sim y$,
- optimize response y from x .

General principle of today's AI : Machine Learning

$$x \longrightarrow \boxed{\text{nature}} \longrightarrow y$$

- predict response y given x ,
- describe $x \sim y$,
- optimize response y from x .

$$x \longrightarrow \boxed{\text{algorithm}} \longrightarrow \hat{y}$$

General principle of today's AI : Machine Learning

$$x \longrightarrow \boxed{\text{nature}} \longrightarrow y$$

- predict response y given x ,
- describe $x \sim y$,
- optimize response y from x .

$$x \longrightarrow \boxed{\text{algorithm}} \longrightarrow \hat{y}$$



Machine Learning paradigms

Statistical Learning

Data are stored in a large database.

The model is built thanks to a training set.

Machine Learning paradigms

Statistical Learning

Data are stored in a large database.

The model is built thanks to a training set. The model is evaluated on a test set.

Machine Learning paradigms

Statistical Learning

Data are stored in a large database.

The model is built thanks to a training set. The model is evaluated on a test set.

The model is deployed in a constraint environment.

Online Learning

Data arrives sequentially.

Machine Learning paradigms

Statistical Learning

Data are stored in a large database.

The model is built thanks to a training set. The model is evaluated on a test set.

The model is deployed in a constraint environment.

Online Learning

Data arrives sequentially. The model is learning on the fly.

Machine Learning paradigms

Statistical Learning

Data are stored in a large database.

The model is built thanks to a training set. The model is evaluated on a test set.

The model is deployed in a constraint environment.

Online Learning

Data arrives sequentially. The model is learning on the fly.

The model is deployed and continues learning on a constraint environment.

New Moore's law

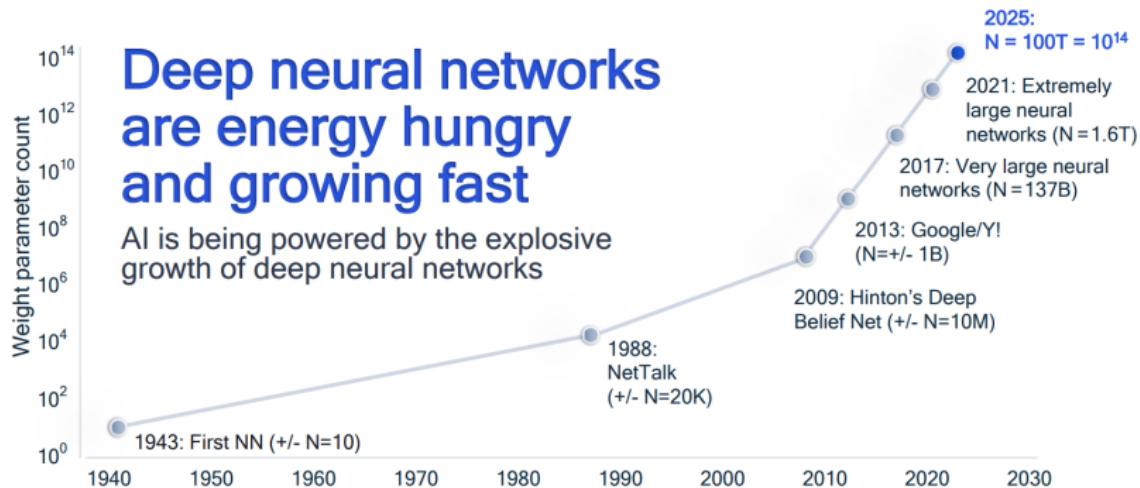


Figure 1: Exponential growth of Deep Learning models
[Fournarakis, 2021a]

Reminder: Impact of Deep Learning

Common carbon footprint benchmarks

in lbs of CO₂ equivalent

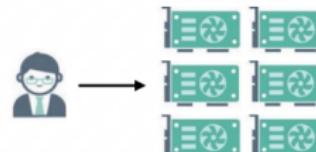
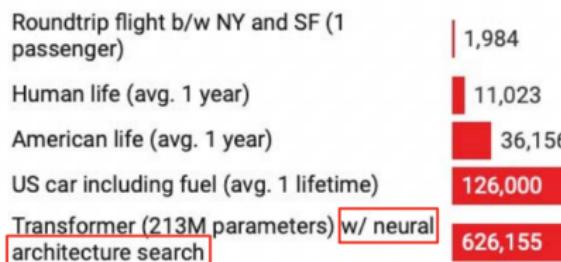


Figure 2: Carbon footprint comparative study between a neural network and activities [Han, 2021].

The impact of Machine Learning is non negligible → Reduce the size of models !

① Tricks and limits

② More maths

③ AIPowerMeter

④ Tiny ML

⑤ Conclusion and Perspectives

1 Tricks and limits

2 More maths

3 AIPowerMeter

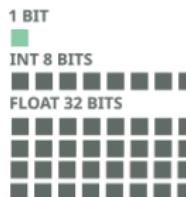
4 Tiny ML

5 Conclusion and Perspectives

Quantization

Low bitwidth approaches

- Originally from [Courbariaux et al., 2015] - BinaryConnect,
- Weights and I/O in [Bulat and Tzimiropoulos, 2019] - XNOR-nets ++,
- Quantization in [Zhou et al., 2016, Fournarakis, 2021a].



BNN's details

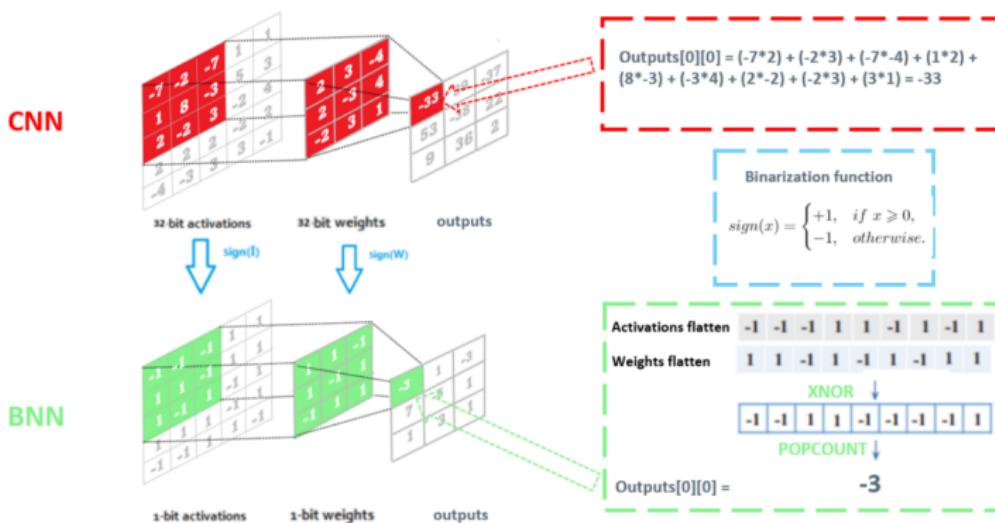


Figure 3: CNN and Binarization [Yuan and Agaian, 2021]

Backpropagation of a BNN

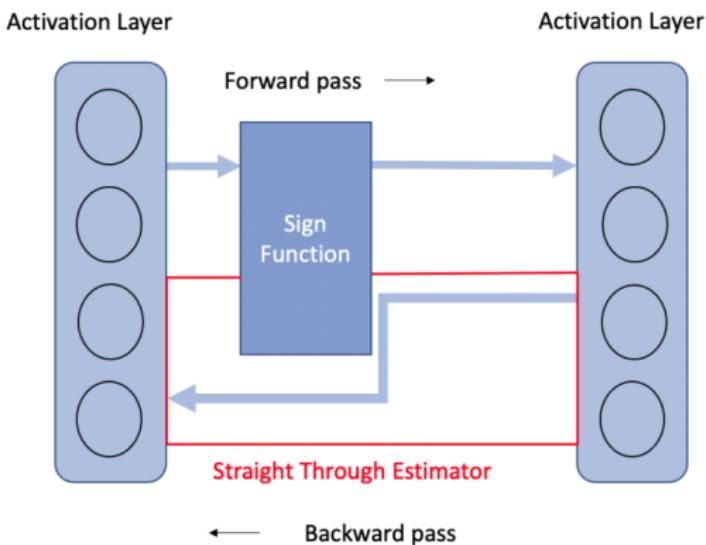


Figure 4: STE estimator in the backprog

Figure source :

Limits of BNN's

- STE estimator and continuous gradient accumulations,
- No gain observed in [Courbariaux et al., 2015] or [Bulat and Tzimiropoulos, 2019],

Limits of BNN's

- STE estimator and continuous gradient accumulations,
- No gain observed in [Courbariaux et al., 2015] or [Bulat and Tzimiropoulos, 2019],
- Larq Compute Engine to the rescue.

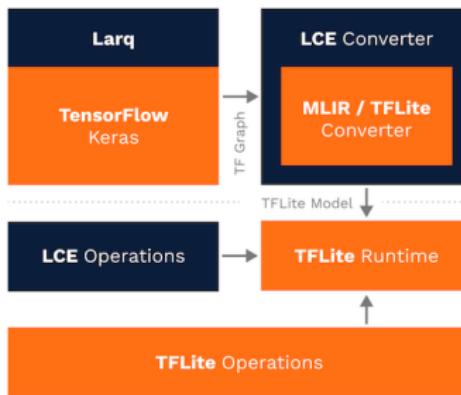


Figure 5: Larq Compute Engine workflow from training to deployment

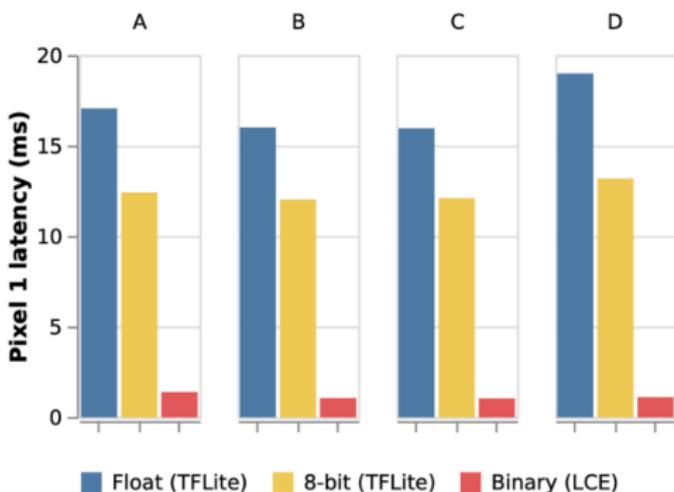
Larq Compute Engine of TFLite model

Principle

- train your model on a standard CPU/GPU,
- convert the model in tflite format,
- a python module ([GitHub link here](#)) based on a C++ routine is used (tiling, vectorization and parallelism) for fast inference.

More details on a blog post [here](#).

Larq Compute Engine gain



h * w * in * out convolutions are:
■ Blue bar: Float (TFLite)
■ Yellow bar: 8-bit (TFLite)
■ Red bar: Binary (LCE)

- h * w * in * out convolutions are:
- (A) $56 \times 56 \times 64 \times 64$; (B) $28 \times 28 \times 128 \times 128$;
 - (C) $14 \times 14 \times 256 \times 256$; (D) $7 \times 7 \times 256 \times 256$

Pruning - origin

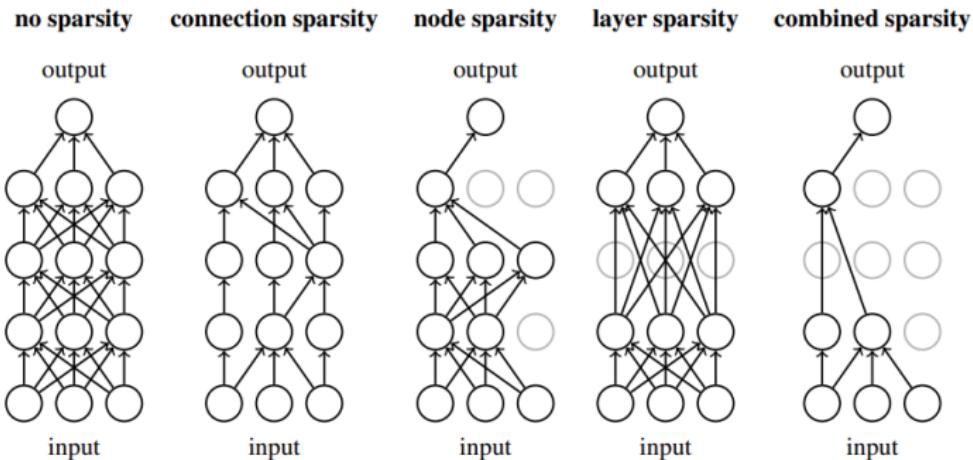
Originated in Leo Breiman's book Classification And Regression Tree (1984):

- build a maximal decision tree T_{\max}
- solve the following optimization problem:

$$\arg \min_{T \subset T_{\max}} \left\{ \sum_{\text{node} \in T} \sum_{x_i \in \text{node}} (y_i - \bar{y}_{\text{node}})^2 + \alpha \|T\|_0 \right\}$$

- goal: reduce overfitting !

Pruning NNs



Pruning NNs - pruned literature

- originated in [Mozer and Smolensky, 1988] with relevance coefficients and Optimal Brain Damage in [LeCun et al., 1990] : **pruning after training**
- recent advances in [Lee et al., 2018], [de Jorge et al., 2020] : **pruning at init**
- statistical approach for layer sparsity or **pruning during training** in [Hebiri and Lederer, 2020, Bellec et al., 2018].

SNIP measures

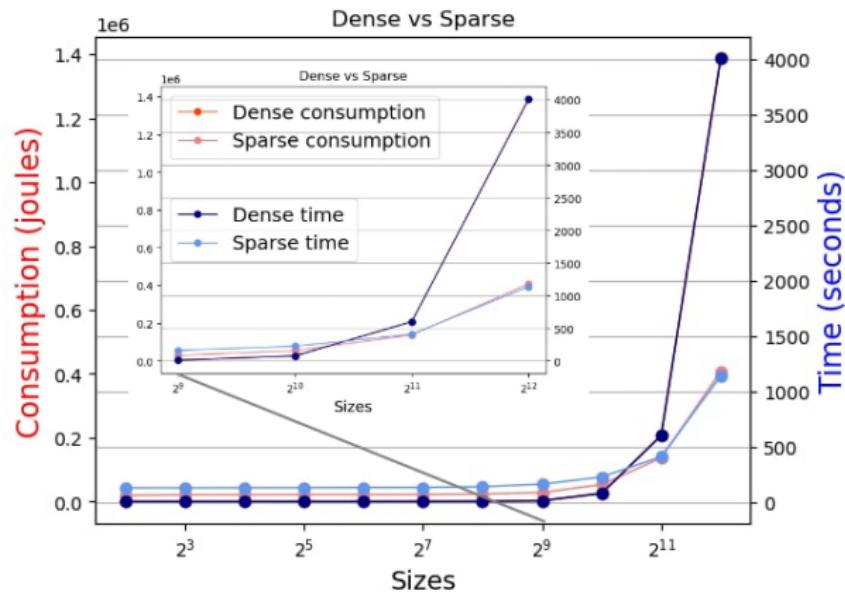
We test a Python implementation of SNIP.

Architecture	Dataset	Pruning ?	Parameters	Time (hh:mm:ss)	Max precision (%)	Total consumption (Wh)
vgg-D	CIFAR-10	no	15,239,872	1:40:18	93.55	785
		yes (95%)	761,994	1:39:03	93.13	771
LeNet-5-Caffe	MNIST	no	430,500	30:18	99.42	145.5
		yes (98%)	8,610	28:26	99.15	145.28

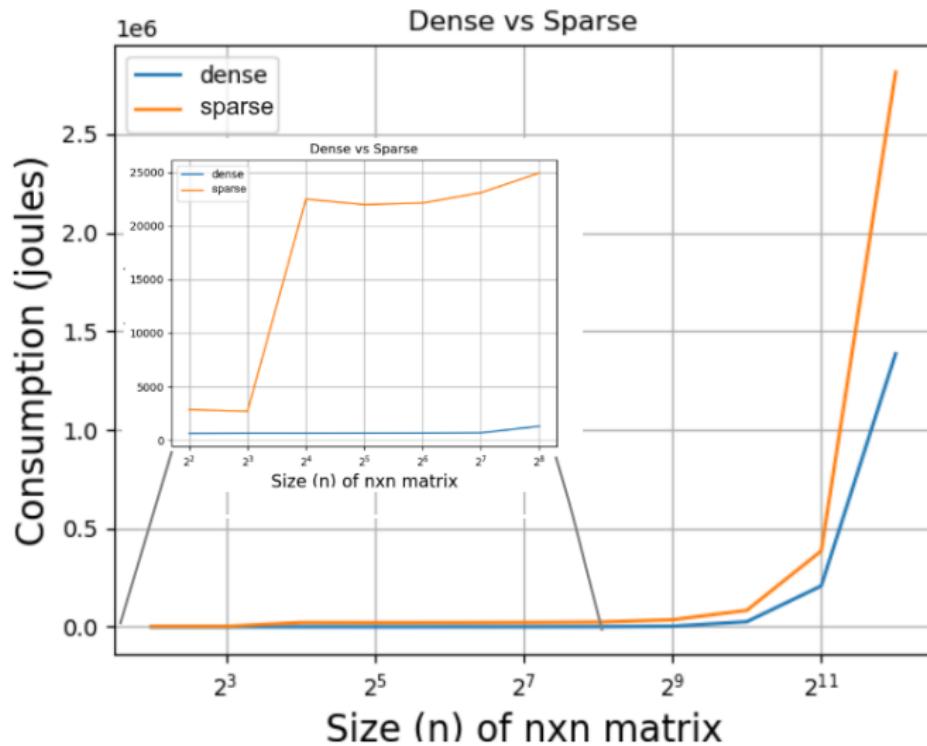
No gain since units are zeroing.

Pytorch Sparse with $1/n$ sparsity index

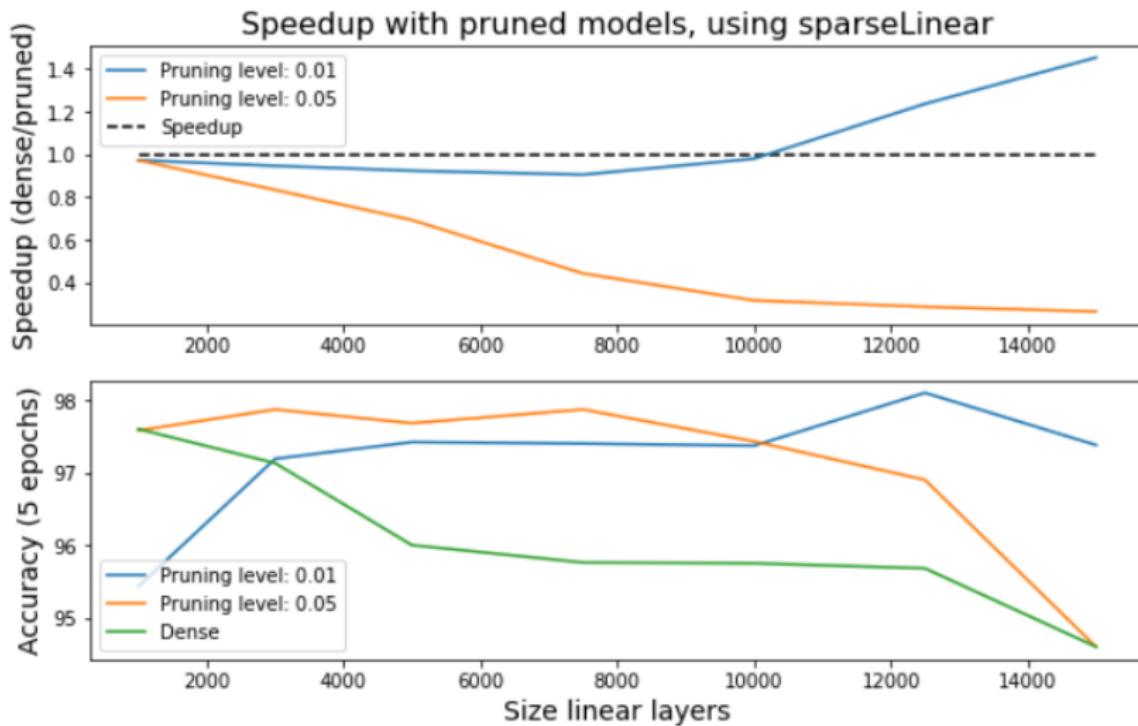
We test PyTorch Sparse library for sparse matrix multiplication.



Pytorch Sparse with 1% of sparsity

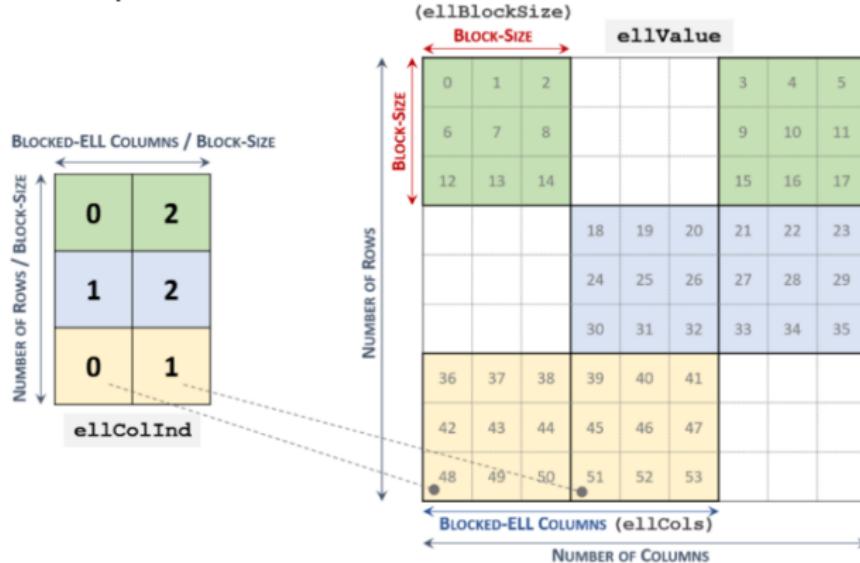


SNIP with PyTorch.Sparse



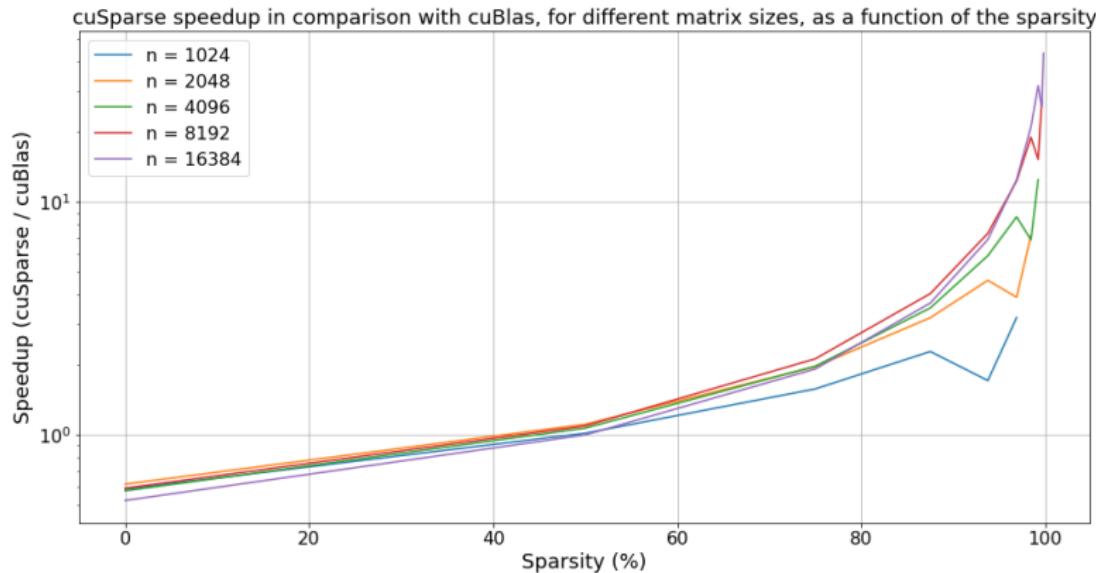
cuSparse

The cuSparse 11.4.0 release provides “a new high-performance block sparse matrix multiplication routine” for newer GPUs, with the help of the **Blocked-ELL format**.

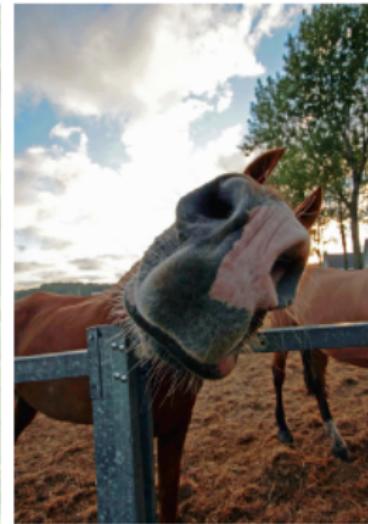


read the cuSPARSE Library doc from Nvidia.

Comparison between cuSparse SpMM and cuBlas hGeMM

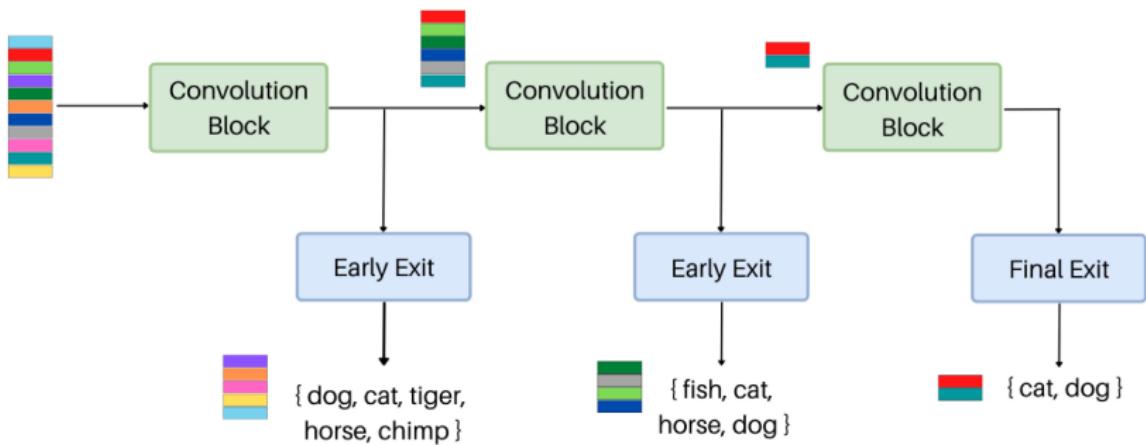


Early Exits - motivation

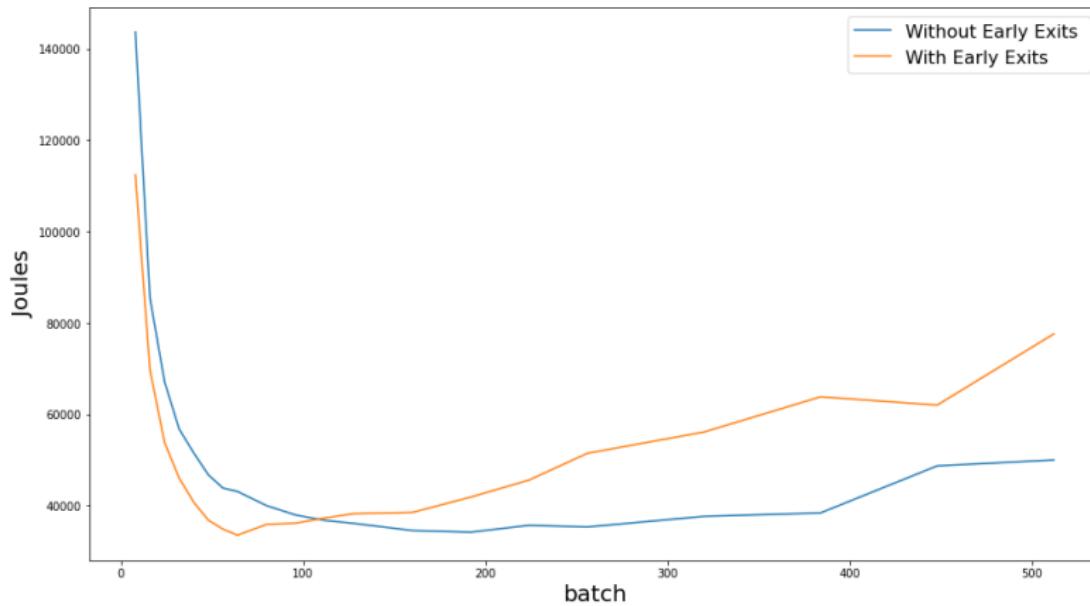


Easy and hard image from [Huang et al., 2017], Copyright Pixel Addict and Doyle.

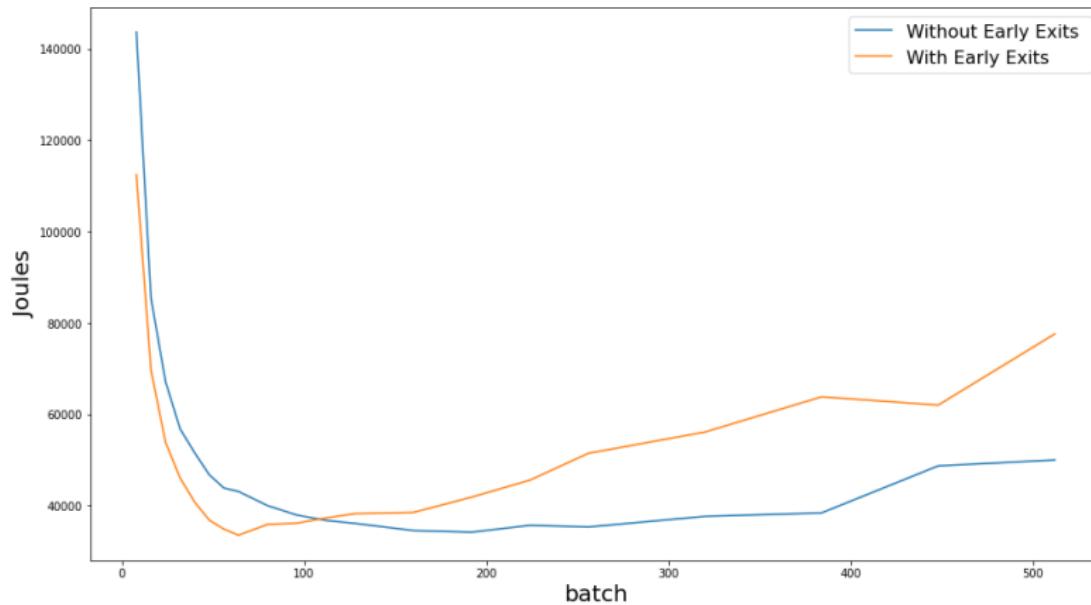
Early Exits - principle



Early Exists - Measure



Early Exists - Paul



Tricks and limits - Conclusion

- BNN: real gain with Larq (see details here)
- Pruning: real gain with block-sparsity (see details here)
- Early Exits: some gain observed here, nice for model parallelism (work in progress).

Tricks and limits - Conclusion

- Difficult to observe real gain,
- Hardware dependent,
- SGD dependent.

1 Tricks and limits

2 More maths

3 AI Power Meter

4 Tiny ML

5 Conclusion and Perspectives

Deep Learning is Optimization

- Optimization problem to deal with:

$$\min_{W=(W_1, \dots, W_L) \in \mathbb{R}^p} \sum_{i=1}^n \ell(g_W(X_i), Y_i),$$

where the number of parameters p is huge and n is big.

- Solved by Stochastic Gradient descent:

$$W^{(t+1)} = W^{(t)} - \alpha \nabla_W \left(\sum_{i=1}^n \ell(g_W(X_i), Y_i) \right) [W^{(t)}]$$

SGD - Origin

A convex function is dually defined as:

$$f(y) \geq f(x) + \nabla f(x) \cdot (y - x), \forall x, y.$$

For $y = \arg \min f(x)$, we have:

$$\boxed{-\nabla f(x) \cdot (y - x) \geq 0}.$$

SGD - Origin

A convex function is dually defined as:

$$f(y) \geq f(x) + \nabla f(x) \cdot (y - x), \forall x, y.$$

For $y = \arg \min f(x)$, we have:

$$\boxed{-\nabla f(x) \cdot (y - x) \geq 0}.$$

WARNING : No constraint about energy consumption

Gradient to Mirror descent

Gradient descent can be written as:

$$W^{(t+1)} := \arg \min_{W \in \mathbb{R}^p} \left\{ \eta \nabla f(W^{(t)}) \cdot W + \frac{\|W - W^{(t)}\|^2}{2} \right\}.$$

Gradient to Mirror descent

Gradient descent can be written as:

$$W^{(t+1)} := \arg \min_{W \in \mathbb{R}^p} \left\{ \eta \nabla f(W^{(t)}) \cdot W + \frac{\|W - W^{(t)}\|^2}{2} \right\}.$$

⇒ no localization and pure Euclidean setting

Gradient to Mirror descent

Mirror descent solves:

$$W^{(t+1)} := \arg \min_W \left\{ \eta \nabla f(W^{(t)}) \cdot W + \mathcal{B}_\Phi(W, W^{(t)}) \right\},$$

where $\mathcal{B}_\Phi(W, W^{(t)}) = \Phi(W) - \Phi(W^{(t)}) - \nabla \Phi(W^{(t)}) \cdot (W - W^{(t)})$
is a Bregman divergence.

Gradient to Mirror descent

Mirror descent solves:

$$W^{(t+1)} := \arg \min_W \left\{ \eta \nabla f(W^{(t)}) \cdot W + \mathcal{B}_\Phi(W, W^{(t)}) \right\},$$

where $\mathcal{B}_\Phi(W, W^{(t)}) = \Phi(W) - \Phi(W^{(t)}) - \nabla \Phi(W^{(t)}) \cdot (W - W^{(t)})$
is a Bregman divergence.

- For $\Phi(W) = \frac{\|W\|^2}{2}$, mirror descent \Leftrightarrow gradient descent,

Gradient to Mirror descent

Mirror descent solves:

$$W^{(t+1)} := \arg \min_W \left\{ \eta \nabla f(W^{(t)}) \cdot W + \mathcal{B}_\Phi(W, W^{(t)}) \right\},$$

where $\mathcal{B}_\Phi(W, W^{(t)}) = \Phi(W) - \Phi(W^{(t)}) - \nabla \Phi(W^{(t)}) \cdot (W - W^{(t)})$ is a Bregman divergence.

- For $\Phi(W) = \frac{\|W\|^2}{2}$, mirror descent \Leftrightarrow gradient descent,
- $\mathcal{B}_\Phi(W, W^{(t)}) = \|W - W^{(t)}\|_{\nabla^2 \Phi(\omega_t)}^2$ by Taylor approximation,

Gradient to Mirror descent

Mirror descent solves:

$$W^{(t+1)} := \arg \min_W \left\{ \eta \nabla f(W^{(t)}) \cdot W + \mathcal{B}_\Phi(W, W^{(t)}) \right\},$$

where $\mathcal{B}_\Phi(W, W^{(t)}) = \Phi(W) - \Phi(W^{(t)}) - \nabla \Phi(W^{(t)}) \cdot (W - W^{(t)})$ is a Bregman divergence.

- For $\Phi(W) = \frac{\|W\|^2}{2}$, mirror descent \Leftrightarrow gradient descent,
- $\mathcal{B}_\Phi(W, W^{(t)}) = \|W - W^{(t)}\|_{\nabla^2 \Phi(\omega_t)}^2$ by Taylor approximation,

Sparsity induced NNs

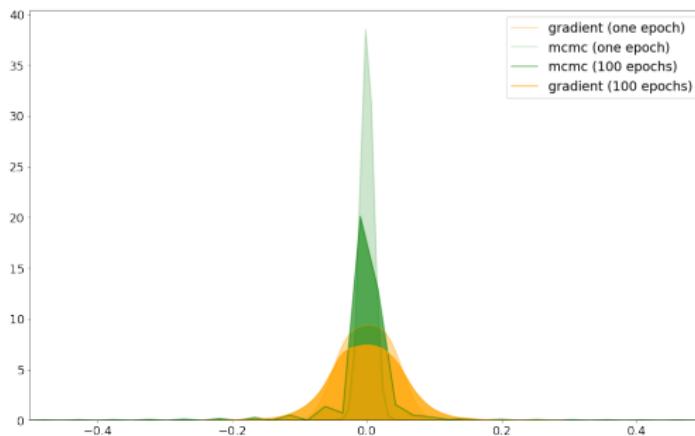
We are looking for a distribution ρ solution of:

$$\min_{\rho} \sum_{i=1}^n \mathbb{E}_{W \sim \rho} \ell(g_W(X_i), Y_i).$$

- For an entropy potential $\Phi(\rho) = \int \rho \log \rho$, we have $\mathcal{B}_\Phi(\rho, \pi) = \mathcal{K}(\rho, \pi)$,
- By chosen $\rho^{(n)}$ a sparsity prior, we get the following risk bound:

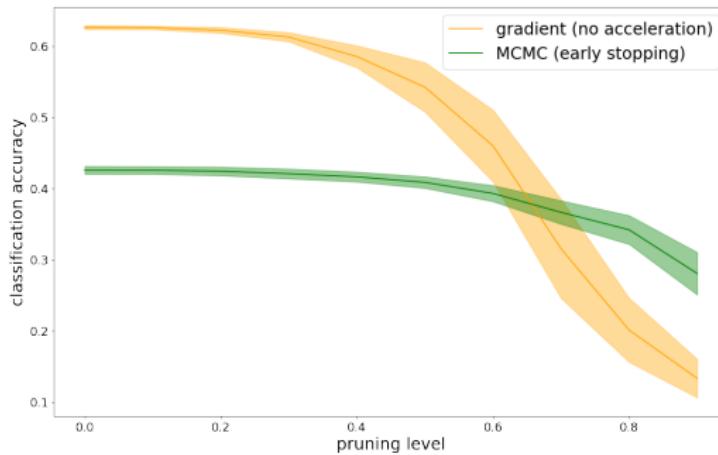
$$\sum_{i=1}^n \mathbb{E}_{W \sim \rho^{(n)}} \ell(g_W(X_i), Y_i) \leq \inf_{W \in \mathbb{R}^p} \left\{ \sum_{i=1}^n \ell(g_W(X_i), Y_i) + \alpha \|W\|_0 \right\}.$$

Resistance to pruning on CIFAR-10



- CNN with 60,000 params,
- SGD with batch size 256 and no acceleration,
- MCMC with 200 iterations by epoch.

Resistance to pruning on CIFAR-10



- CNN with 60,000 params,
- SGD with batch size 256 and no acceleration,
- MCMC with 200 iterations by epoch.

Greedy (RJ)-MCMC algorithm

Initialization : $\mathbf{w}_1 \sim \pi$. Parameter $\lambda > 0$.

For $m = 1, \dots, M$ **do**

For $k = 1, \dots, N$ **do**

- Pick a layer $\ell \in \{1, \dots, L\}$ at random,
- Propose $\tilde{\mathbf{w}} \sim p(\cdot | \mathbf{w}_k)$,
- Accept $\mathbf{w}_{k+1} = \tilde{\mathbf{w}}$ with proba:

$$\rho = \frac{\exp\{-\lambda \sum_{t \in \mathcal{I}_m} \ell(y_t, g_{\tilde{\mathbf{w}}}(x_t))\}}{\exp\{-\lambda \sum_{t \in \mathcal{I}_m} \ell(y_t, g_{\mathbf{w}_k}(x_t))\}} \frac{\pi(\tilde{\mathbf{w}})}{\pi(\mathbf{w}_k)}.$$

More maths - conclusion

With another mathematical framework, we get a simple Accept/Reject optimizer with several nice conclusion:

- (block)-sparsity induced NNs is easy,
- hybrid optimization is natural,
- model parallelism is easier,
- training optimization \Leftrightarrow fast inference.

1 Tricks and limits

2 More maths

3 AIPowerMeter

4 Tiny ML

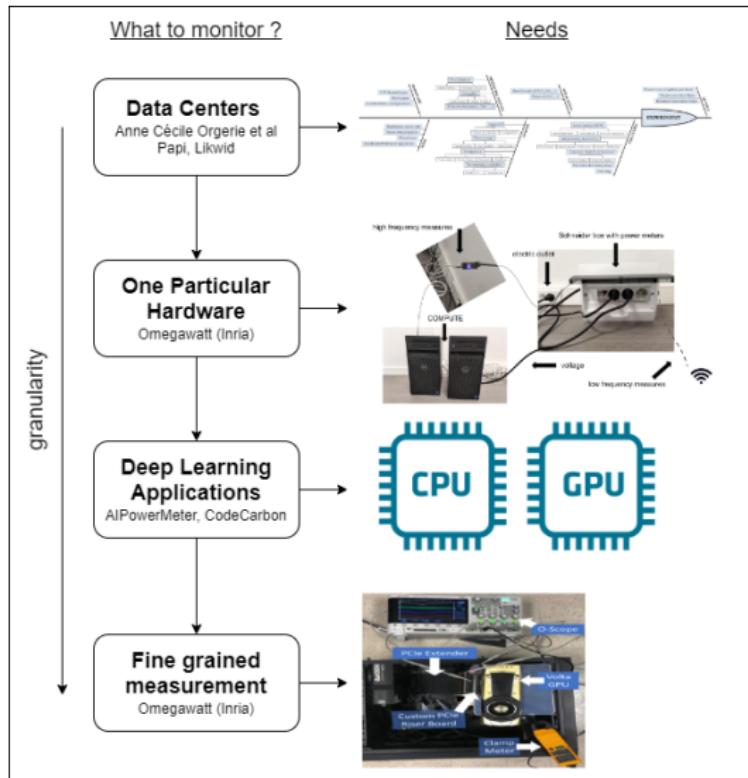
5 Conclusion and Perspectives

Energy

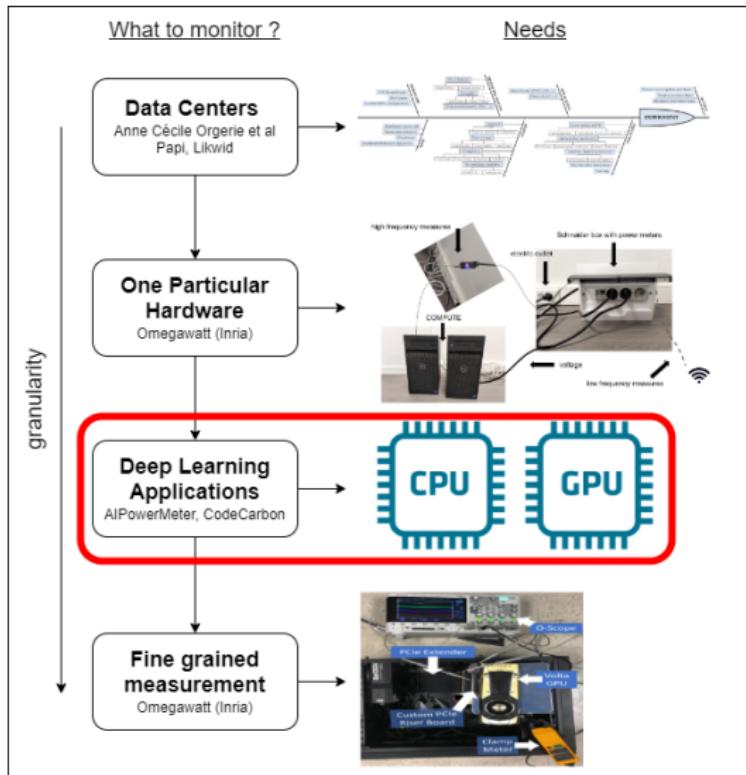
"You can't improve what you don't measure"

- What is energy ? Quantitative property that is transferred to a physical system (through work, heat or light).
- Measured in Joules: $1J \rightarrow$ The energy dissipated as heat when an electric current of one ampere passes through a resistance of one ohm for one second.
- What is power ? \rightarrow "The amount of energy transferred or converted per unit time" $\rightarrow J/s = W$
- kWh ? \rightarrow One kW sustained for one hour $\rightarrow 1kWh = 3.6MJ$

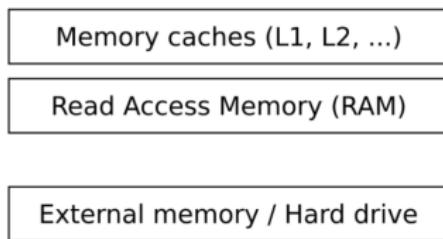
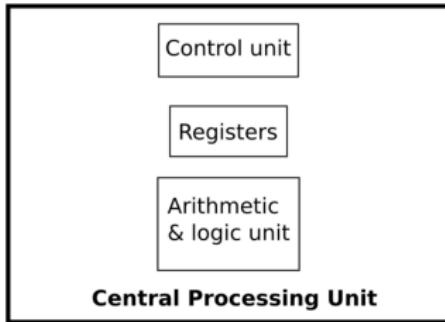
Energy consumption monitoring in IT



Energy consumption monitoring in IT

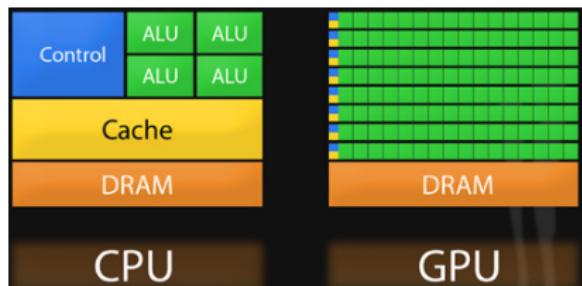


CPU



- Instructions set : boolean, floating operations
- Registers : fast memory used by the ALU (10-100 registers with 8-64 bits)
- Memory: Closer to the cpu → smaller and faster
- Moving data up and down the memory hierarchy costs time and power → optimizations

GPU



- GPU : Consumes more than the whole computer !
- Thousands of cores to enable parallelism
- Higher latency, Higher memory throughput

More power hungry and requires a CPU → BUT Energy efficient since the computations is faster

Monitoring the energy consumption of a Deep Learning algorithm

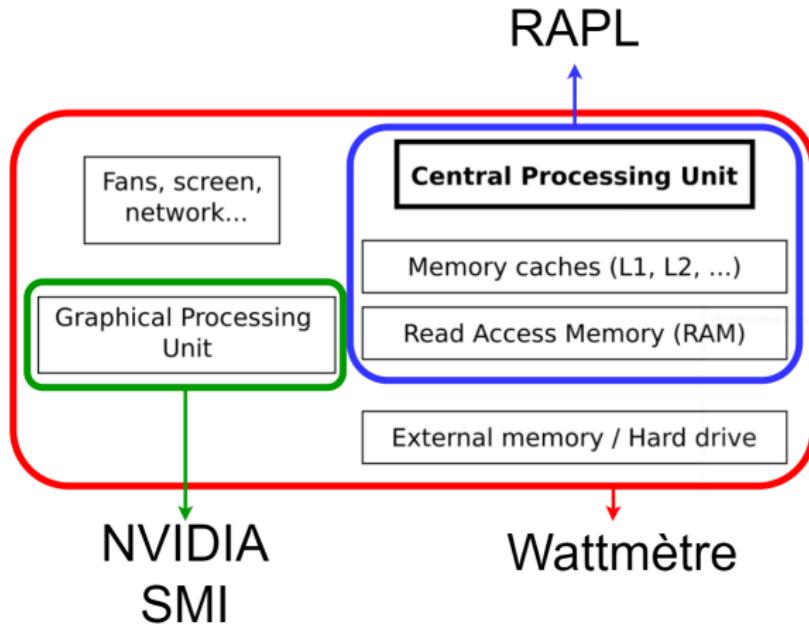
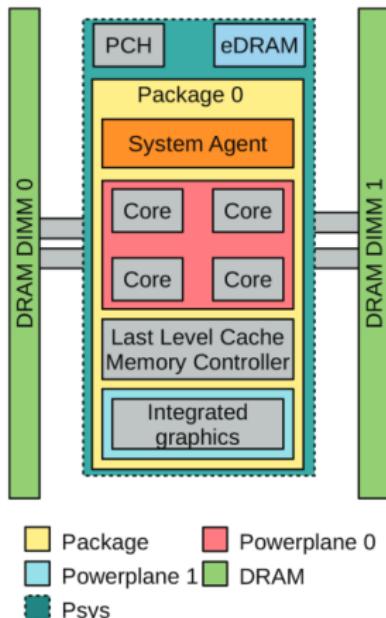


Figure 6: Different manners to monitor each components

RAPL for CPU energy consumption monitoring



- Running Average Power Limit (Sandy bridge architecture in 2011)
- Reports the accumulated energy consumption recording at 1000Hz
- Monitor the energy consumption of different parts: CPU, RAM, System on Chip energy consumption, Processor graphics on the socket.

- Command:

```
sudo chmod -R 755 /sys/class/power
```

Figure 7: K. N. Khan et al.
2018

NVIDIA-SMI to monitor GPU consumption

NVIDIA-SMI 410.79		Driver Version: 410.79		CUDA Version: 10.0		
GPU	Name	Persistence-M	Bus-Id	Disp.A	Volatile Uncorr. ECC	
Fan	Temp	Perf	Pwr:Usage/Cap	Memory-Usage		GPU-Util Compute M.
0	TITAN Xp	Off	00000000:01:00.0	Off	1861MiB / 12194MiB	0% Default
ERR1	54C	P2	ERR1 / 250W			
1	TITAN Xp	Off	00000000:02:00.0	Off	11097MiB / 12196MiB	0% Default
50%	69C	P2	74W / 250W			
2	TITAN Xp	Off	00000000:05:00.0	Off	11299MiB / 12196MiB	0% Default
62%	B1C	P2	86W / 250W			
Processes:		GPU Memory Usage				
GPU	PID	Type	Process name			
0	1535	C	-	617MiB		
0	17875	C	-	617MiB		
0	23889	C	-	617MiB		

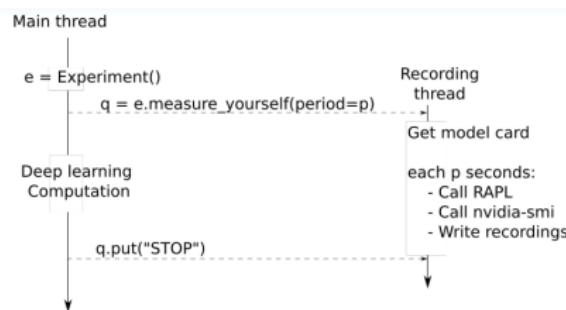
Figure 8: K. N. Khan et al. 2018

- NVIDIA System Management Interface (CUDA)
- +/- 5% accuracy of current power draw. Memory usage per gpu and per process
- Command:

```
nvidia-smi -q -x
```

In practice: AI PowerMeter

- Developed by GreenAI UPPA, to measure the efficiency of your deep learning recording CPU and GPU
- By and for data scientists
- It uses Nvidia-smi and RAPL as well as psUtil (to compute by program)



<https://github.com/GreenAI-Uppa/AIPowerMeter>

AI PowerMeter vs WattMetre

Global comparison of consumption measure on PAM with multiprocessing

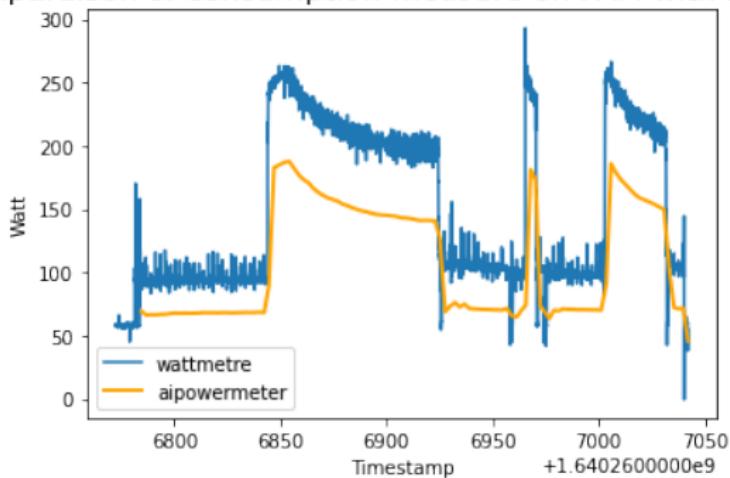
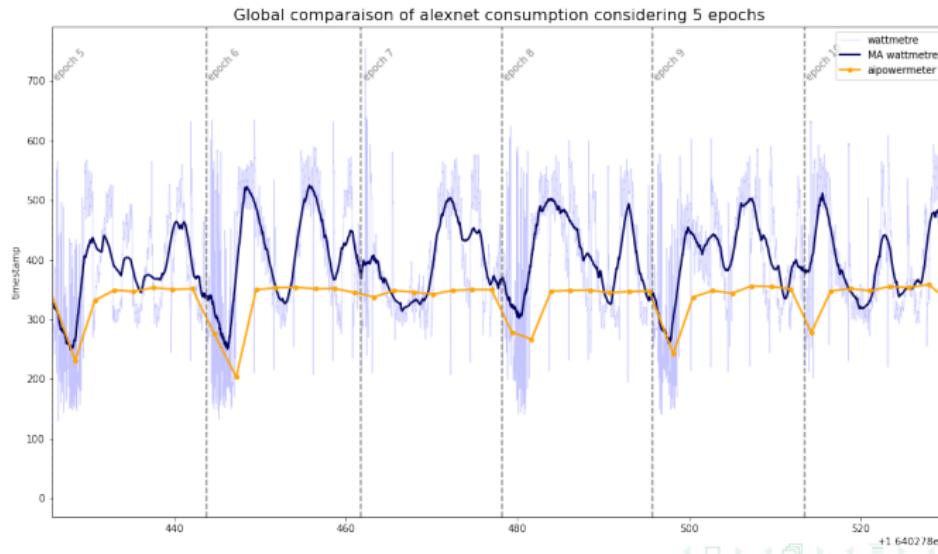


Figure 9: Wattmeter is more precise, but AiPowermeter permits to monitor the evolution of the power supply.

Application

- Example of the training of AlexNet for image classification.
- Energy consumed during training: $672kJ = 0.187kWh$
($\simeq 11gCO2eq$)



Monitor smaller, optimized, hardwares

- In the lab, we are interested in embedded systems: Raspberry Pi, Jetson Cards or micro-controllers
- Constraints exist to monitor (GPU, RAPL not available for ARM processors, Nvidia-SMI not working on Jetson...)

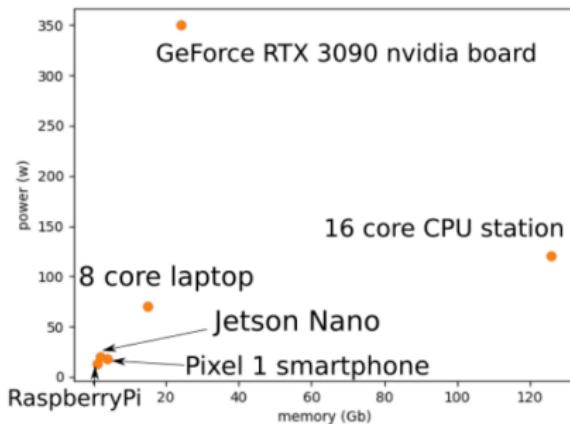


Figure 11: Find the better compromise between power and memory !

1 Tricks and limits

2 More maths

3 AI Power Meter

4 Tiny ML

5 Conclusion and Perspectives

ML field

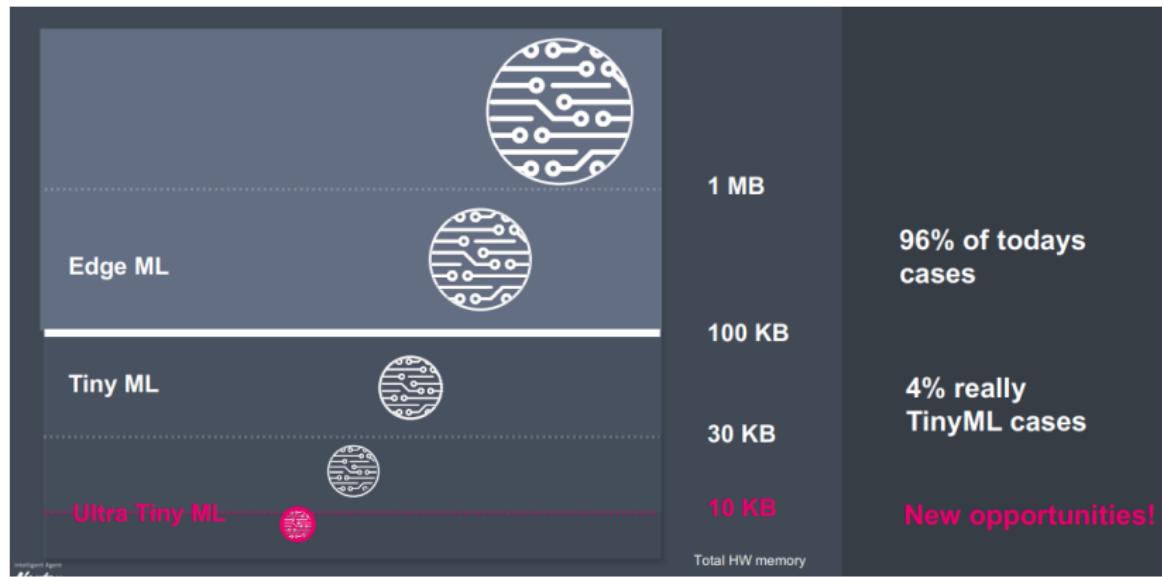


Figure 12: Benchmark of the tiny ML field [NEUTON.AI, 2022]

What is Tiny ML?

- What is tinyML ?

TinyML: When a neural network model can be run at an energy cost of below 1 mW

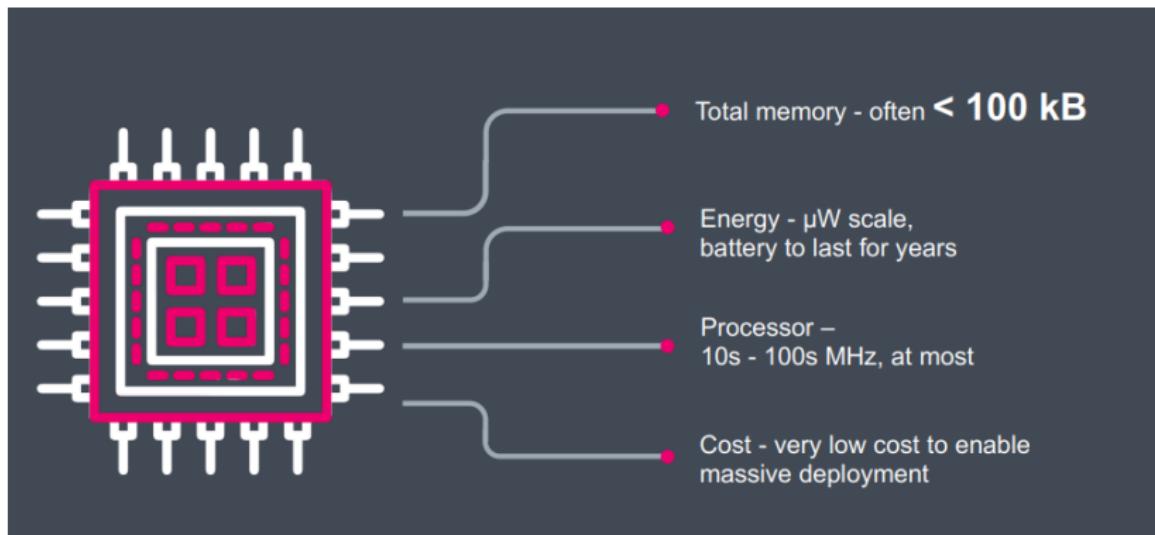


Figure 13: Definition of TinyML by Pete Warden[NEUTON.AI, 2022]

Tiny ML ?

- This presentation is highly inspired by the book

TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers by Warden and Situnayake [Pete Warden, 2019].



Figure 14: Different models for different capacities [Han, 2021]

It is also inspired by many presentations yielded at the TinyML Summit every year (March 2022)

Why ?

- Function – wanting a smart device to act quickly and locally (independent of the Internet).
- Cost – accomplishing this with simple, lower cost hardware.
- Privacy – not wanting to share all sensor data externally.
- Efficiency – smaller device form-factor, energy-harvesting or longer battery life.

Limitations in terms of hardware

- Decrease in energy consumption → limitations in sRAM memory, flash memory, microprocessor capacities

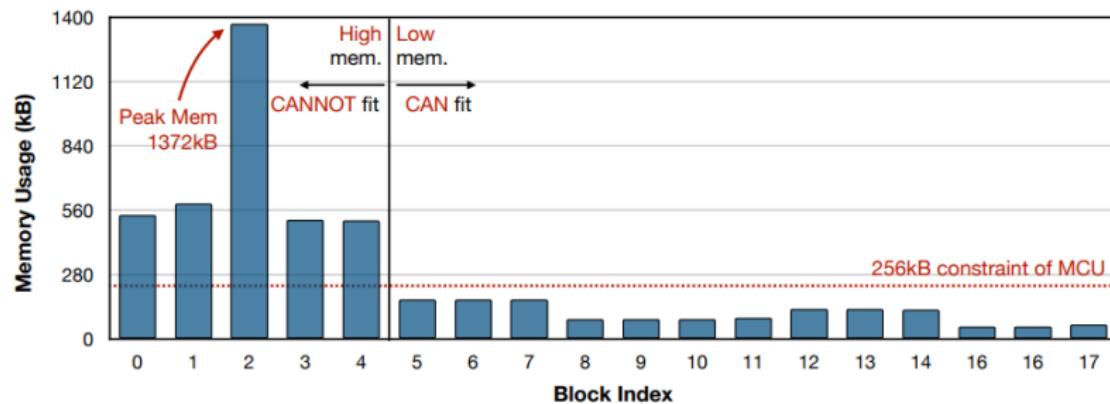


Figure 15: Per-block memory usage of MobileNetV2 [Ji Lin, 2021]

→ What is important is the **Peak memory** !

Micro-controllers

- A typical microcontroller system consists of a processor core, an on-chip SRAM block and an on-chip embedded flash
- Constraints
 - Peak memory usage of the model computations < memory usage.
 - Number of parameters in the model < flash memory storage
 - Model size and the peak memory < 250 KB each;
 - CNN computation < 60 million multiply-adds per inference at high accuracy

Comparison between hardwares

Micro-controller	Price	Memory	Specificities
Arduino Nano 33 BLE Sense	29,70€	256 kB	
SparkFun Edge	\$16.50	384kB	
ST Microelectronics STM32F746G Discovery kit	\$54.0	340 kB	Screen / included camera

Table 1: Main micro-controllers on the market for tinyML



(a) SparkFun Edge

(b) Arduino Nano

(c) ST

Arduino Nano 33 BLE Sense: Components

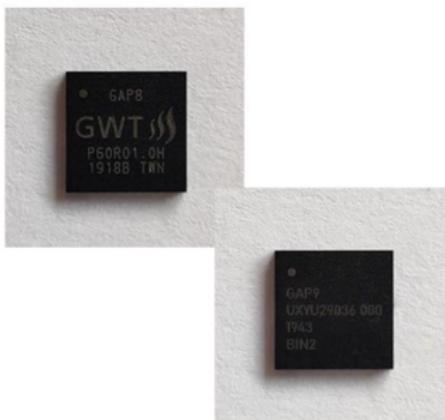
Sensor	Power
IMU	$1mW$
Weather (humidity, and temperature)	$5\mu W$
barometric sensor	$10 \mu W$
microphone	$300\mu W$
Gesture, proximity, light	?
Bluetooth® Low Energy connectivity	40 mW

Table 2: Components integrated

Possibility to connect many sensors such as cameras for recognition
(1 mW at 30 FPS for 320×320 -pixel monochrome image sensor).

Applications of tinyML

Visible Image	
Sound	
IR Image	
Radar	
Bio-sensor	
Gyro/Accel	



Wearables / Hearables



Battery-powered consumer electronics



IoT Sensors

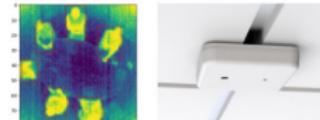


Figure 17: Different possible applications of tinyML [Fournarakis, 2021b]

More precision: Patch-Based Learning

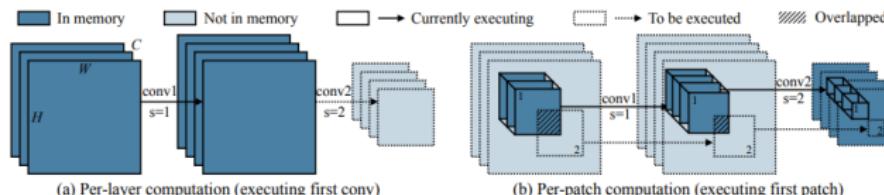


Figure 18: Apply filters on patch only: Patch-Based Learning
[Ji Lin, 2021]

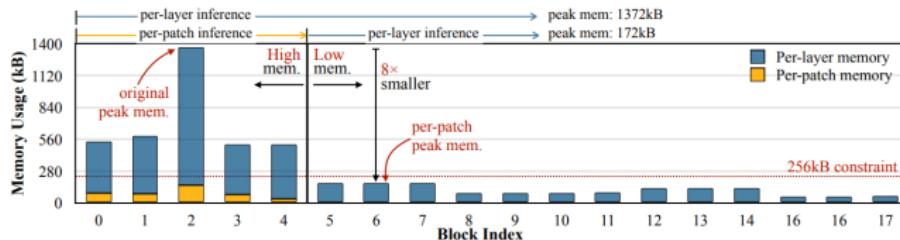


Figure 19: Reduce the peak memory ! Tradeoff between overall computation time and performances

1 Tricks and limits

2 More maths

3 AI Power Meter

4 Tiny ML

5 Conclusion and Perspectives

Conclusion

- Pruning block sparse binarisation
- Mesurer
- Plus de maths
- Mesurer l'inference

References I

[Bellec et al., 2018] Bellec, G., Kappel, D., Maass, W., and Legenstein, R. (2018).

Deep rewiring: Training very sparse deep networks.

In *International Conference on Learning Representations*.

[Bulat and Tzimiropoulos, 2019] Bulat, A. and Tzimiropoulos, G. (2019).

Xnor-net++: Improved binary neural networks.

arXiv preprint arXiv:1909.13863.

References II

[Courbariaux et al., 2015] Courbariaux, M., Bengio, Y., and David, J.-P. (2015).

Binaryconnect: Training deep neural networks with binary weights during propagations.

In *Advances in neural information processing systems*, pages 3123–3131.

[de Jorge et al., 2020] de Jorge, P., Sanyal, A., Behl, H. S., Torr, P. H., Rogez, G., and Dokania, P. K. (2020).

Progressive skeletonization: Trimming more fat from a network at initialization.

arXiv preprint arXiv:2006.09081.

[Fournarakis, 2021a] Fournarakis, M. (2021a).

A practical guide to neural network quantization.

References III

[Fournarakis, 2021b] Fournarakis, M. (2021b).

A practical guide to neural network quantization.

[Han, 2021] Han, S. (2021).

Putting ai on a diet: Tinyml and efficient deep learning.

[Hebiri and Lederer, 2020] Hebiri, M. and Lederer, J. (2020).

Layer sparsity in neural networks.

[Huang et al., 2017] Huang, G., Chen, D., Li, T., Wu, F., van der

Maaten, L., and Weinberger, K. Q. (2017).

Multi-scale dense networks for resource efficient image classification.

References IV

[Ji Lin, 2021] Ji Lin, Wei-Ming Chen, H. C. C. G. S. H. (2021).

Mcunetv2: Memory-efficient patch-based inference for tiny deep learning.

[LeCun et al., 1990] LeCun, Y., Denker, J. S., and Solla, S. A. (1990).

Optimal brain damage.

pages 598–605.

[Lee et al., 2018] Lee, N., Ajanthan, T., and Torr, P. H. (2018).

Snip: Single-shot network pruning based on connection sensitivity.

arXiv preprint arXiv:1810.02340.

References V

[Mozer and Smolensky, 1988] Mozer, M. C. and Smolensky, P. (1988).

Skeletonization: A technique for trimming the fat from a network via relevance assessment.

In *Proceedings of the 1st International Conference on Neural Information Processing Systems*, pages 107–115.

[NEUTON.AI, 2022] NEUTON.AI (2022).

A novel approach to building exceptionally tiny models without loss of accuracy.

[Pete Warden, 2019] Pete Warden, D. S. (2019).

TinyML: Machine learning with tensorflow lite on arduino and ultra-low-power microcontrollers.

References VI

[Yuan and Agaian, 2021] Yuan, C. and Agaian, S. S. (2021).

A comprehensive review of binary neural network.

[Zhou et al., 2016] Zhou, S., Wu, Y., Ni, Z., Zhou, X., Wen, H., and Zou, Y. (2016).

Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients.

arXiv preprint arXiv:1606.06160.