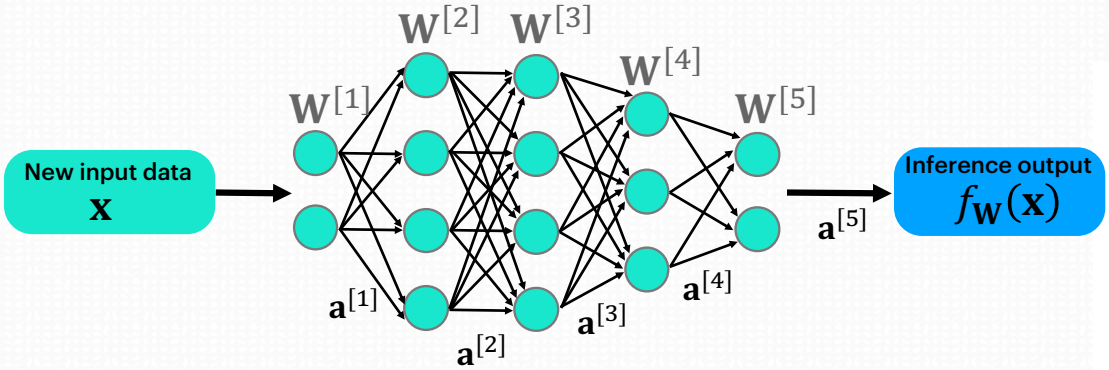
Compressing Deep Neural Networks for Deployment or Training

Hardware Accelerators that Compute Just Right!

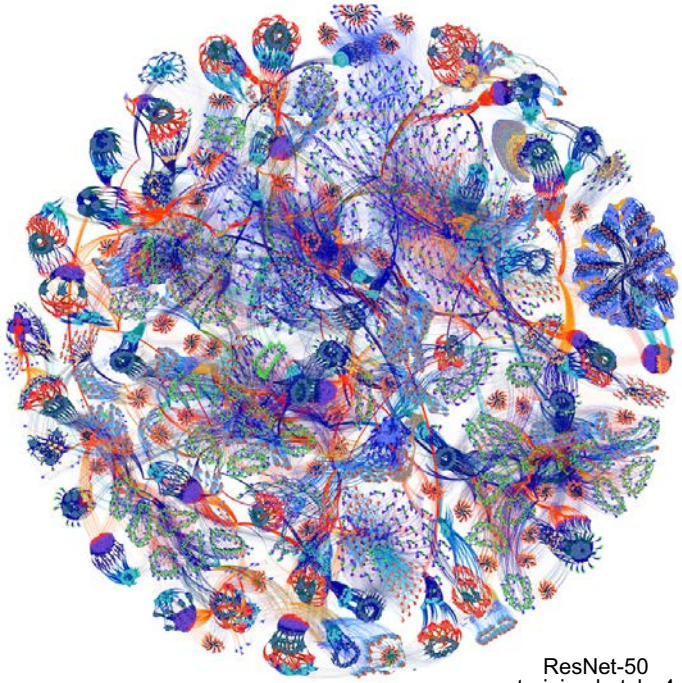
Olivier Sentieys
Univ. Rennes, Inria, IRISA
olivier.sentieys@irisa.fr

joint work with Silviu Filip, Léo Pradels, Cédric Gernigon, Sami Ben Ali, Mariko Tatsumi, Guy Lemieux

Complexity Issues of Deep Neural Networks



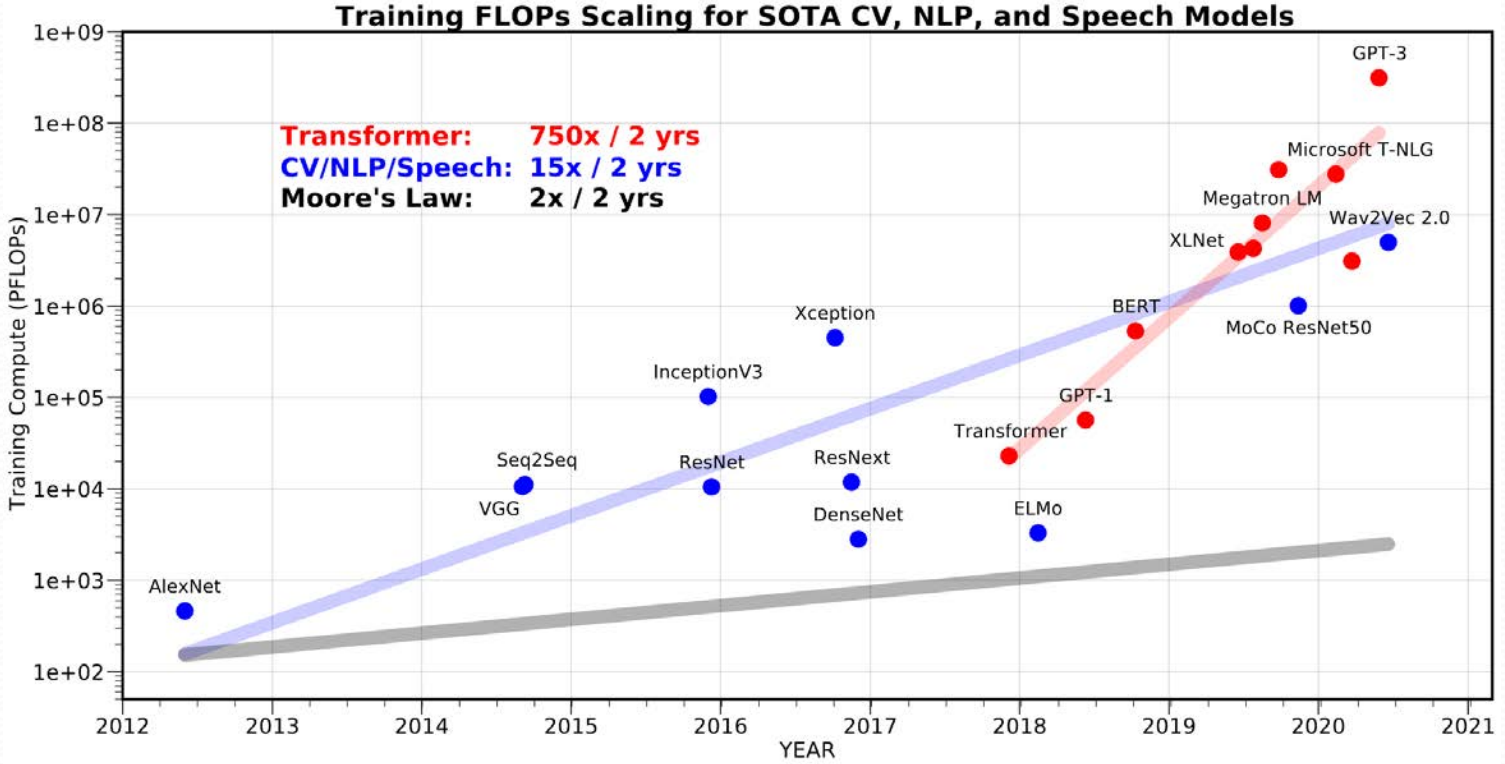
- Two main tasks
  - training - determine set of network parameters to solve a *task* (minimize a loss on a *training set*)
  - inference - given an input, compute (forward propagate) using the trained network



Poplar® graph

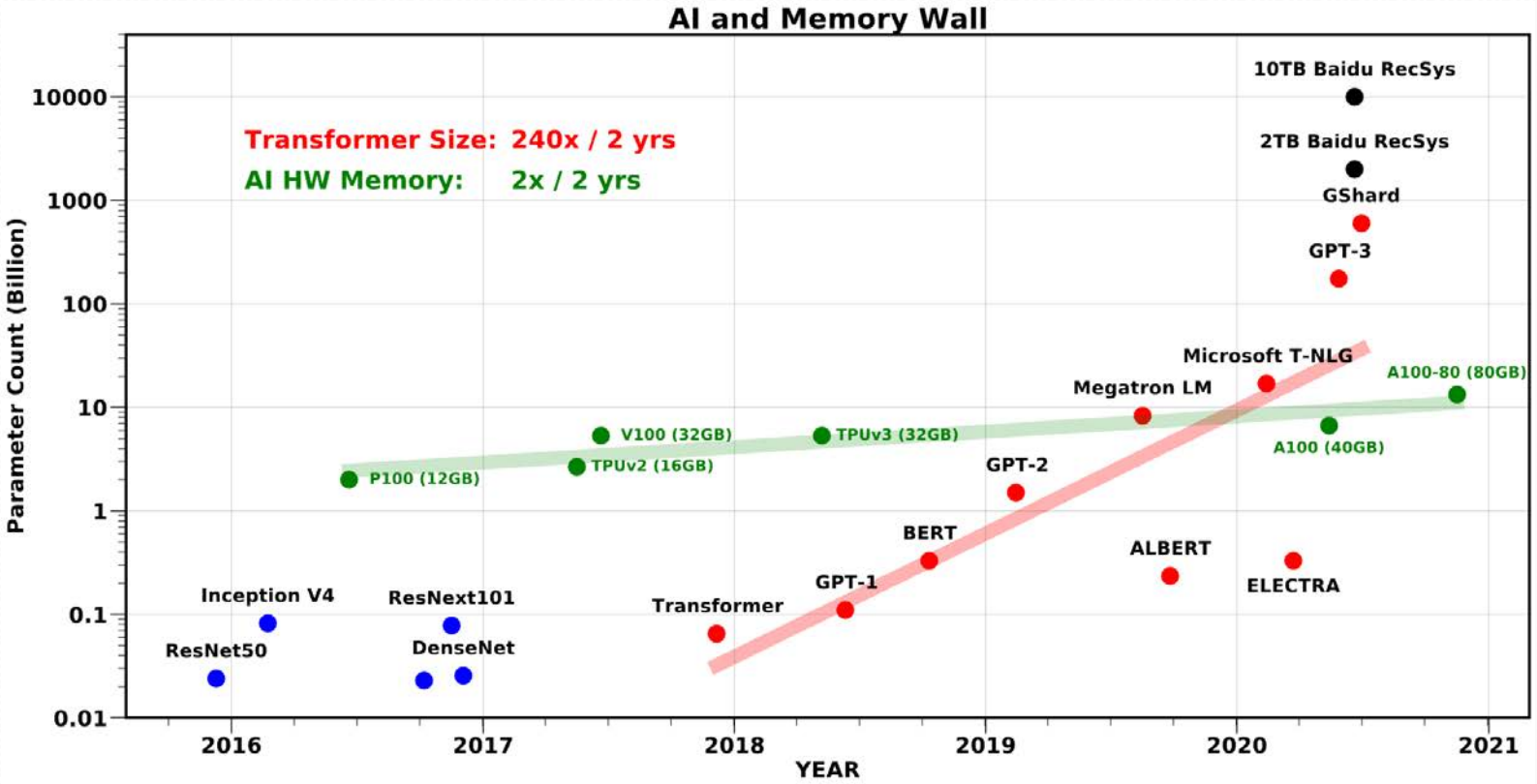
Computing power demand of AI

- is higher than what computer architectures can bring



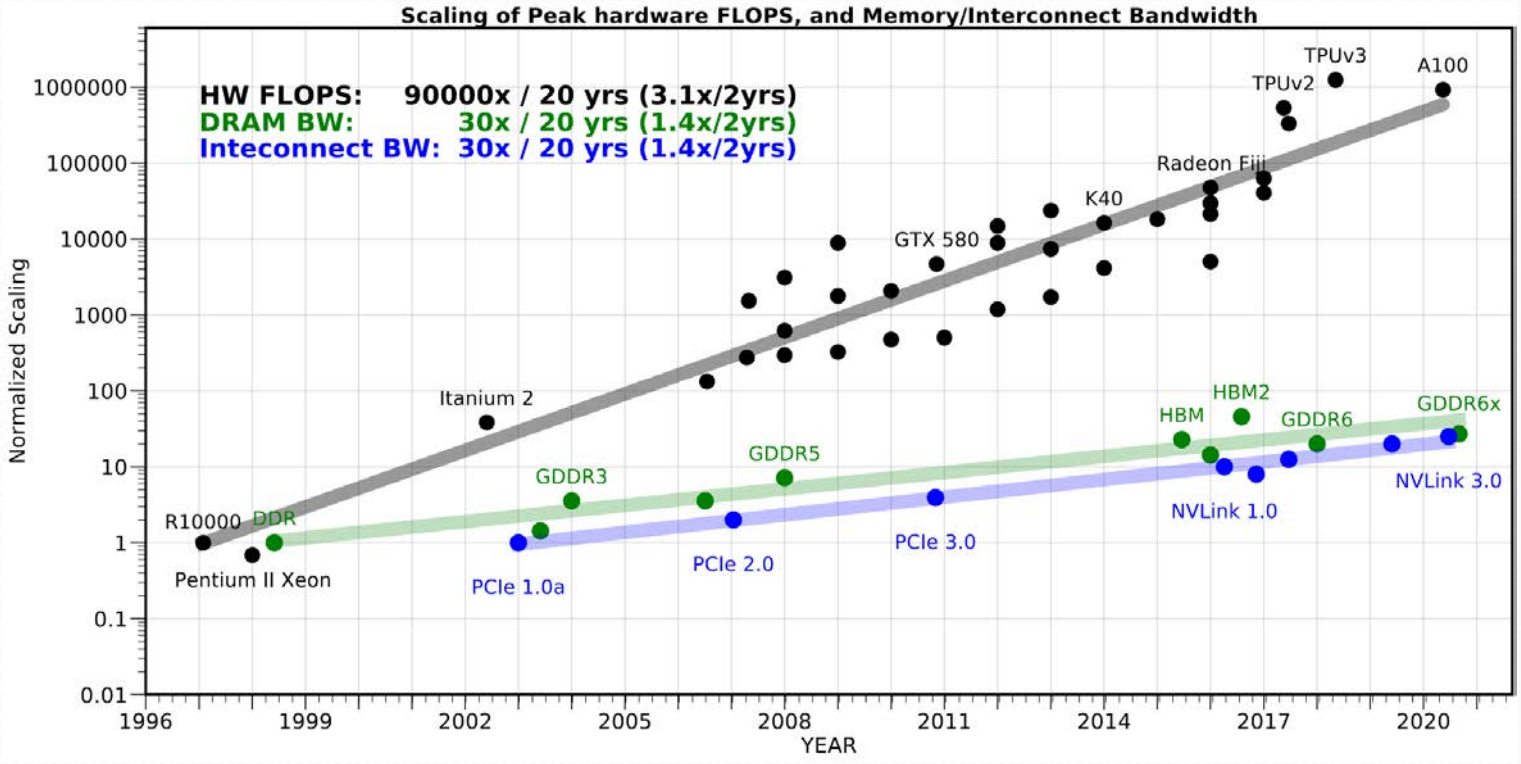
Evolution of the number of parameters

- is much higher than available (on-chip) memory capacity

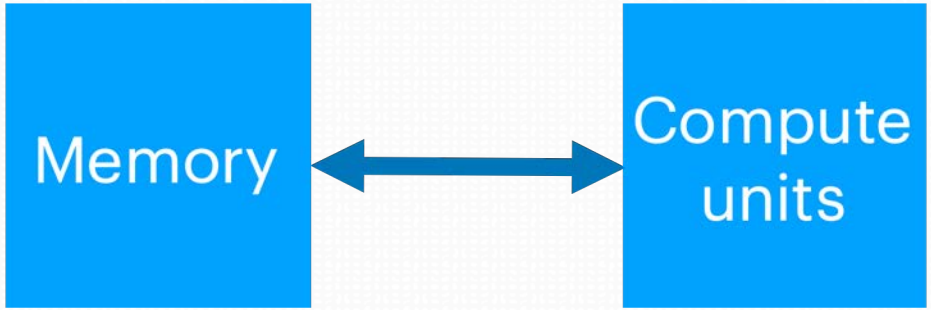


Evolution of bandwidth

- is much slower than FLOPS



Memory Bottleneck



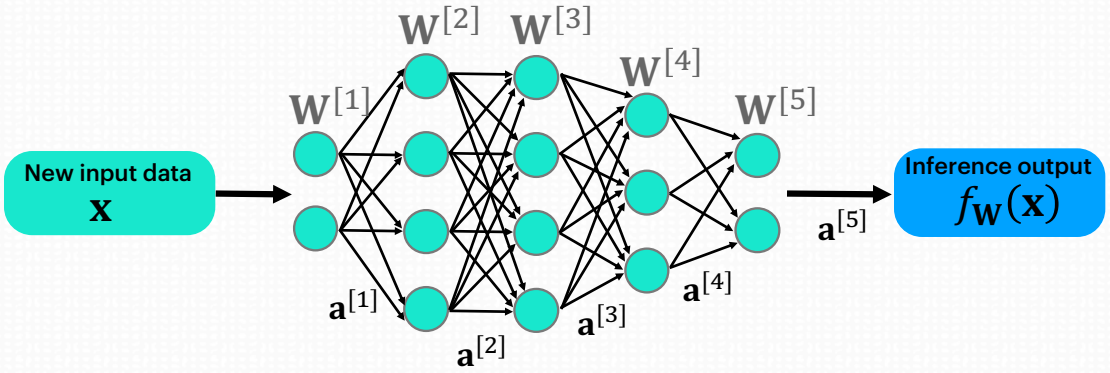
Data movement

- move input data & model from memory to compute units
- send partial results back to memory

Computations

- vector/matrix manipulations
- done on CPU, GPU, or custom accelerators (e.g., FPGA, ASIC)

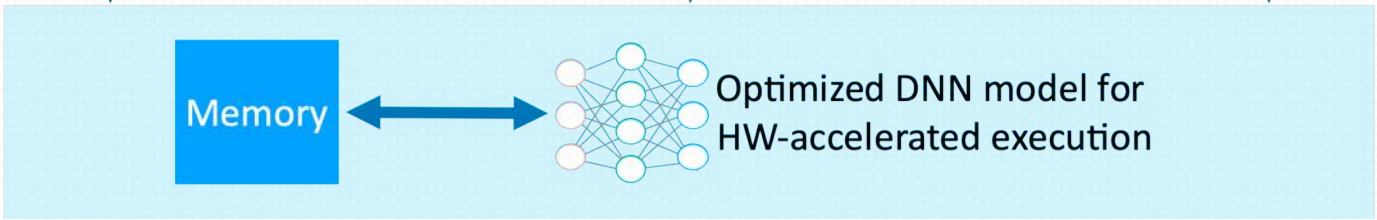
Need for DNN Compression



Compression
prune model parameters, while keeping target accuracy

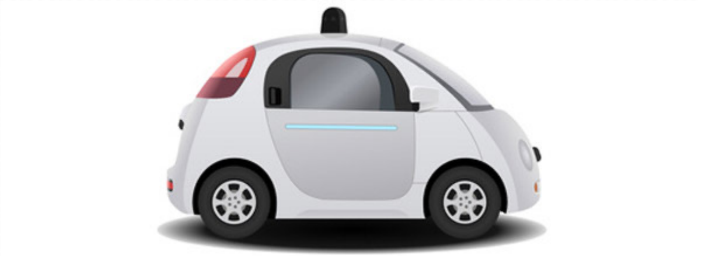
Quantization
reduce arithmetic bit width and compute, while keeping target accuracy

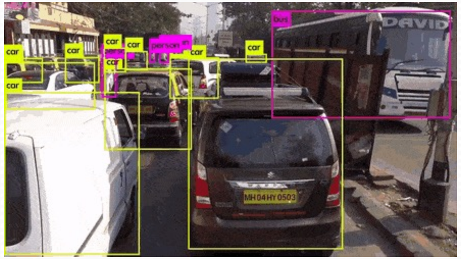
Compilation
determine how to compile DNN models for efficient HW execution



Need for DNN Compression

- From Sensors to Clouds

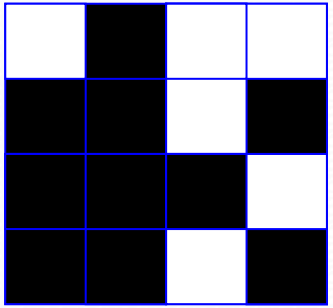




- For both Inference and Training

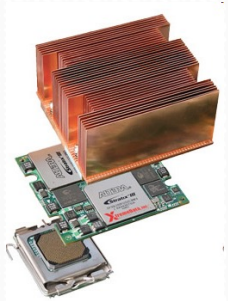
Computer Architectures

- Energy consumption is a major issue
- Utilization wall
- End of Moore's law…



Dark Silicon

- Domain-Specific Architectures are the road ahead
  - Energy efficiency requires deeply specialized hardware
  - which also may come with pain from the programmer



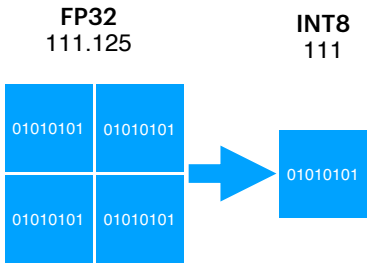
What is DNN quantization?

- store network parameters in low precision
- store/compute intermediate signals in low precision
- store/compute back propagated gradients in low precision

Quantization effects: the good

Memory usage

storage needed for weights and activations is proportional to bit width



Power and energy consumption

energy is significantly reduced for both computations and memory accesses

ADD energy (pJ)			
INT8	INT32	FP16	FP32
0.03	0.1	0.4	0.9
30x energy reduction			

MULT energy (pJ)			
INT8	INT32	FP16	FP32
0.2	3.1	1.1	3.7
18.5x energy reduction			

Memory access energy (pJ)	
Cache (64-bit)	
8KB	10
32KB	20
1MB	100
DRAM	1300-2600
Up to 4x More due to locality	

Latency

less memory access and simpler computations lead to faster execution and reduced latency

Silicon area

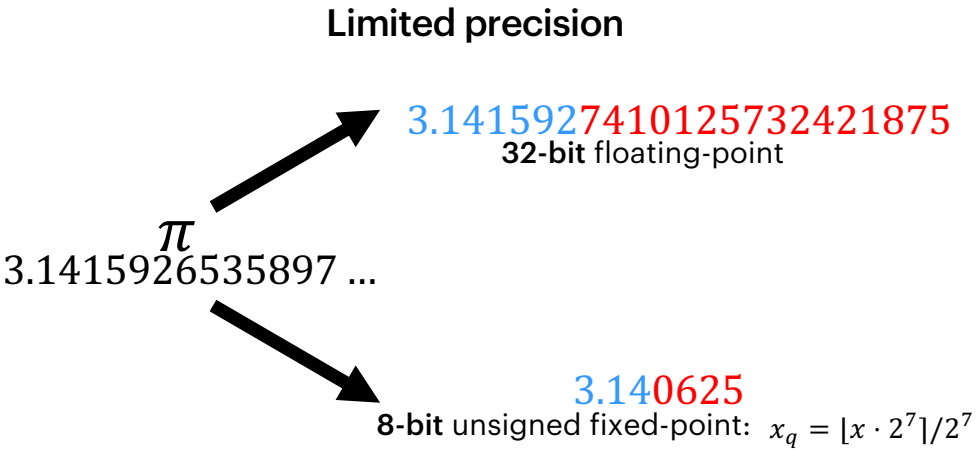
8-bit arithmetic and below requires much less area

MULT area (μm^2)			
INT8	INT32	FP16	FP32
282	3495	1640	7700
27x area reduction			

ADD area (μm^2)			
INT8	INT32	FP16	FP32
36	137	1360	4184
116x area reduction			

Sources: Mark Horowitz (Stanford), energy based on ASIC, area based on TSMC 45nm process
Wikimedia Commons

Quantization effects: the good and the bad

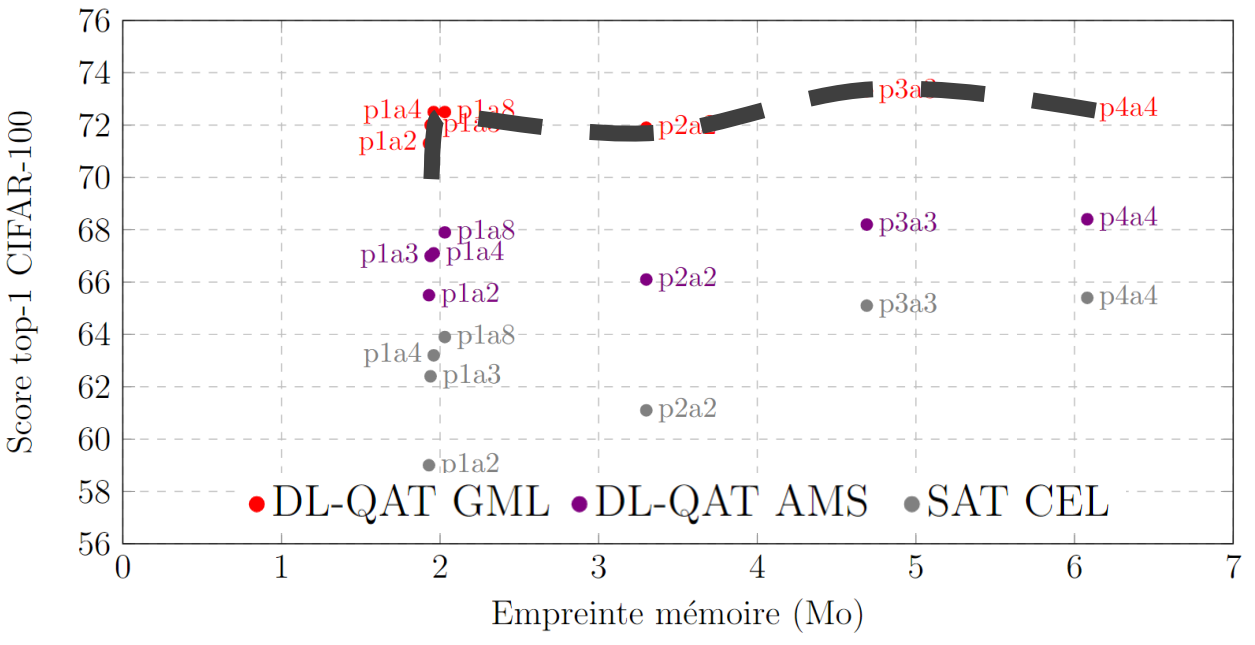


DNN Quantization Methods

- PTQ: Post-Training Quantization
- QAT: Quantization-Aware Training

$p\{n\}a\{m\}$
weights: n bits
activations: m bits

Resnet-18
CIFAR-100



Adaptive QAT

- Learning the bit-width during training
  - Both weights and activations

Hardware Loss:

$N_t \in \mathbb{R}_+^*$: bit-width, a new parameter to optimize

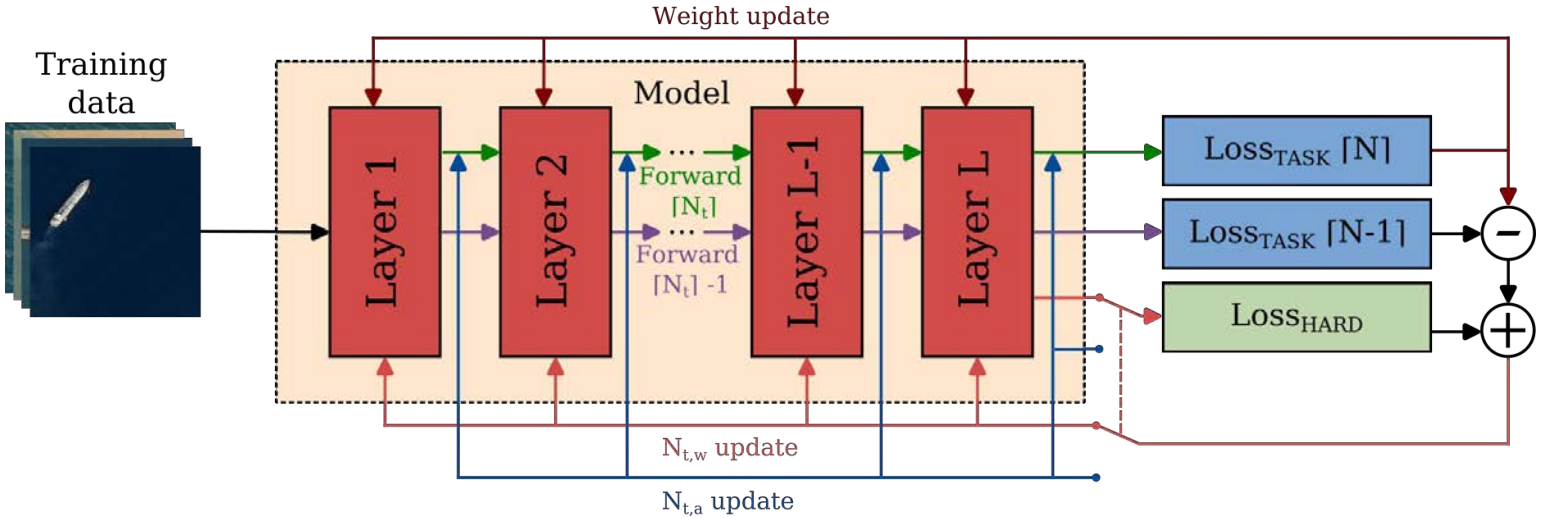
$$L_{Hard} = \alpha \cdot \sum_{l=1}^{L-1} \lceil N_t^{(l)} \rceil$$

Total loss:

$$L_{Total} = \lambda \cdot L_{Task}[N_t] + (1-\lambda) \cdot L_{Hard}[N_t]$$

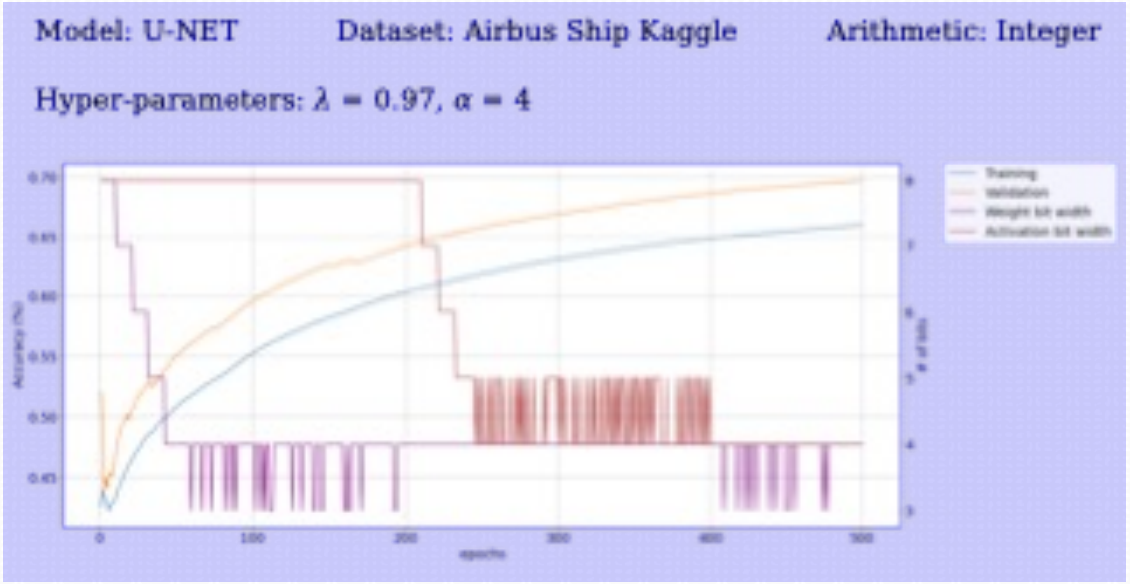
Bit-width gradient approximation:

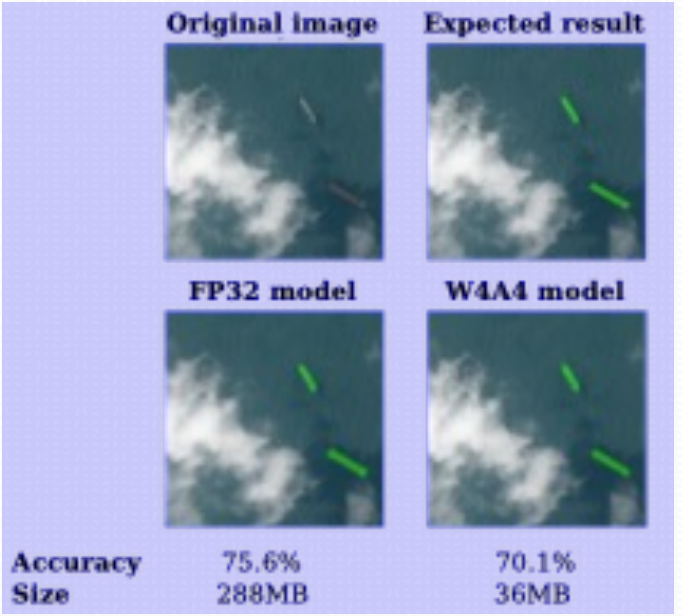
$$\lambda \cdot (L_{Task}[N_t] - L_{Task}[N_t - 1]) + (1-\lambda) \cdot \frac{\partial L_{Hard}[N_t]}{\partial \lceil N_t \rceil}$$



Adaptive QAT

- Learning the bit-width during training
  - Both weights and activations





Even Worse for Training…

- Carbon footprint of DNN training

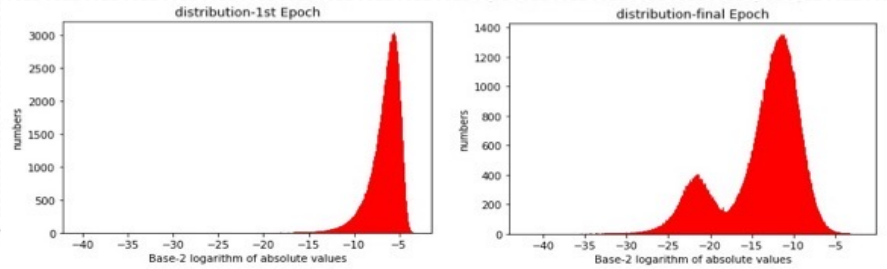
Analyzing the carbon footprint of current natural-language processing models shows an alarming trend: ***training one huge model for machine translation emits the same amount of CO2 as five cars in their lifetimes (fuel included)*** [Strubell *et al.*, ACL 2019]

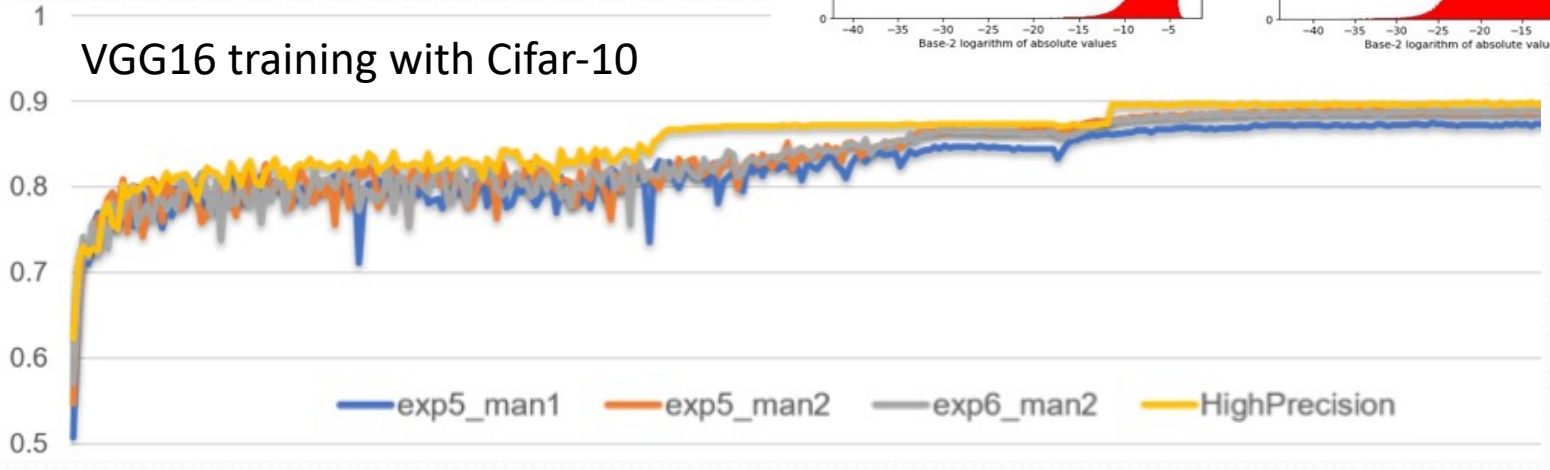
  - Many more operations than inference
  - More pressure on memory access
  - Much more difficult to accelerate

Need for a Significant Reduction of the Carbon Footprint of Neural Network Training Hardware

Low-Precision Training of DNNs

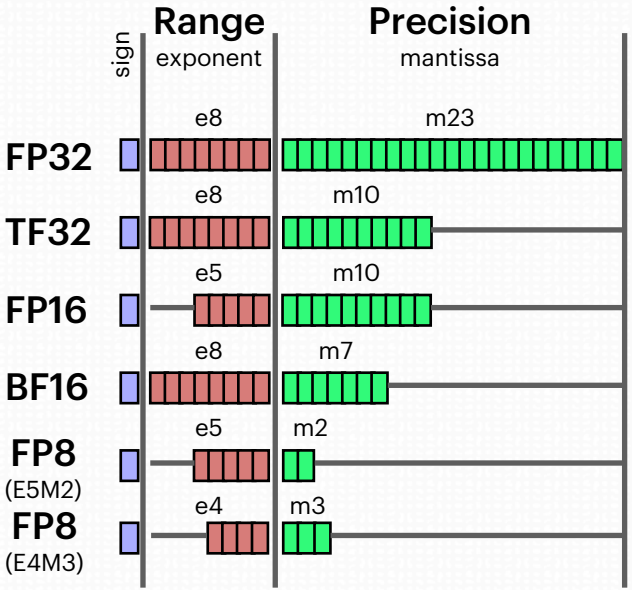
- Explore mixed numerical precision hardware
  - Low-precision floating-point, variable-precision variants, building the accelerator



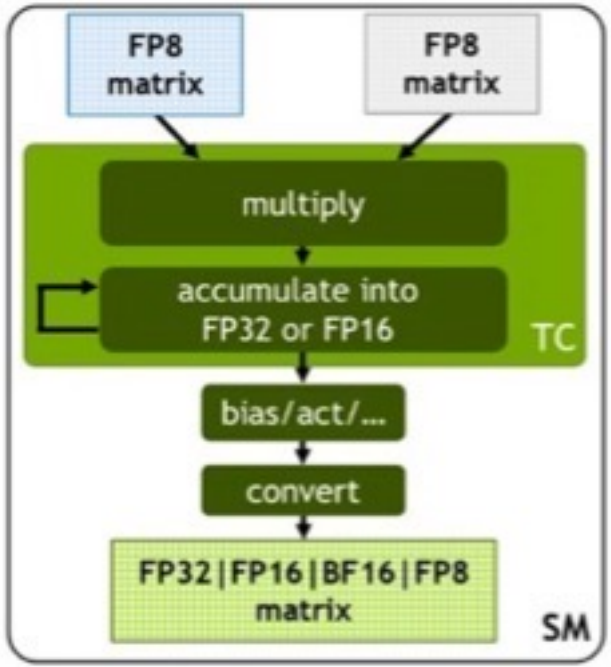


Arithmetic Support in Latest Chips?

- Various trade-offs in terms of range, precision, performance

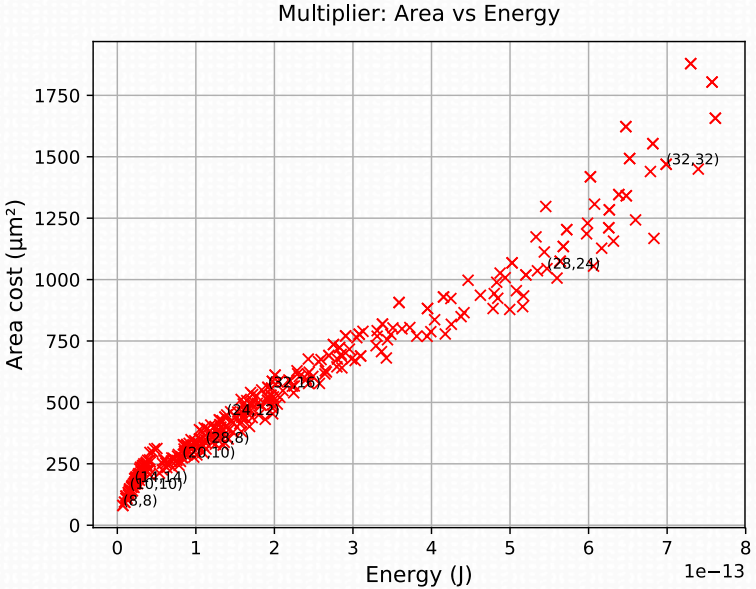


- Nvidia Hopper GH100 GPU
  - FP8 support in tensor cores provides up to 4x speedup



Energy Gains of Low Precision

- Multiplier (float)

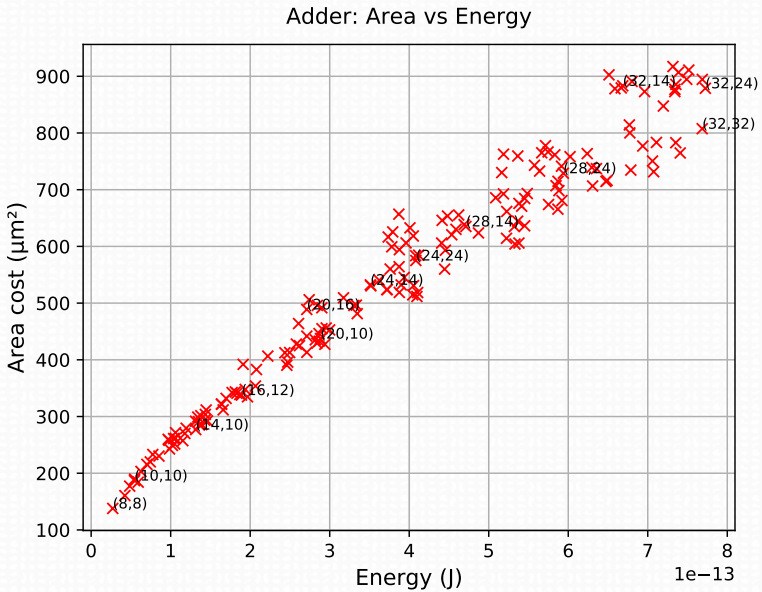


>200x

8 bits

32 bits

- Adder (float)



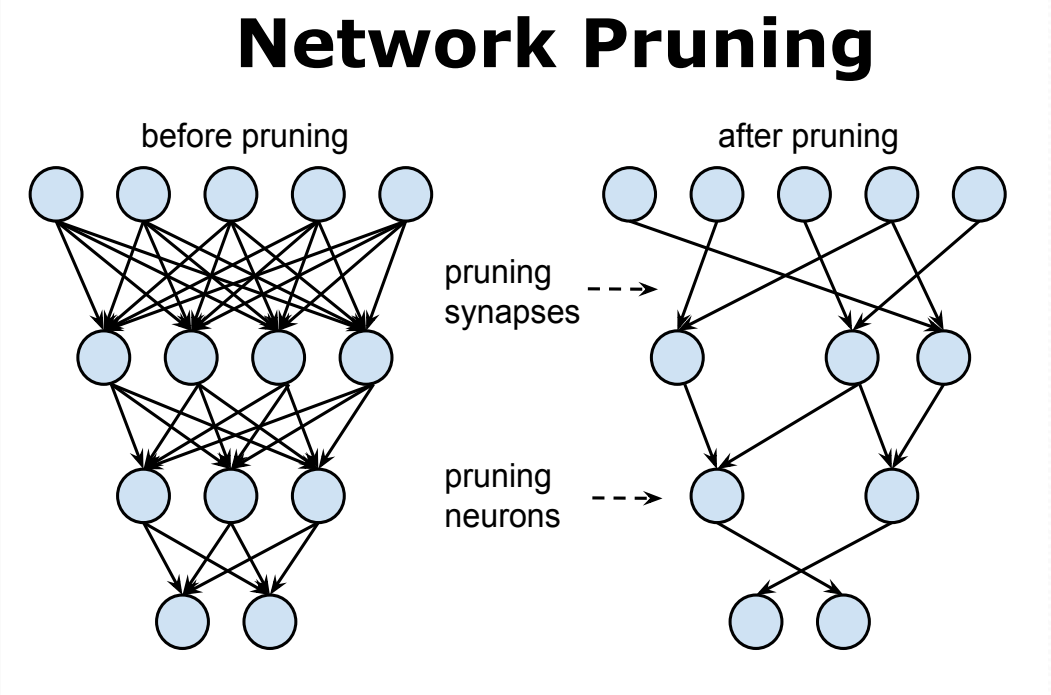
>30x

8 bits

32 bits

DNN Pruning

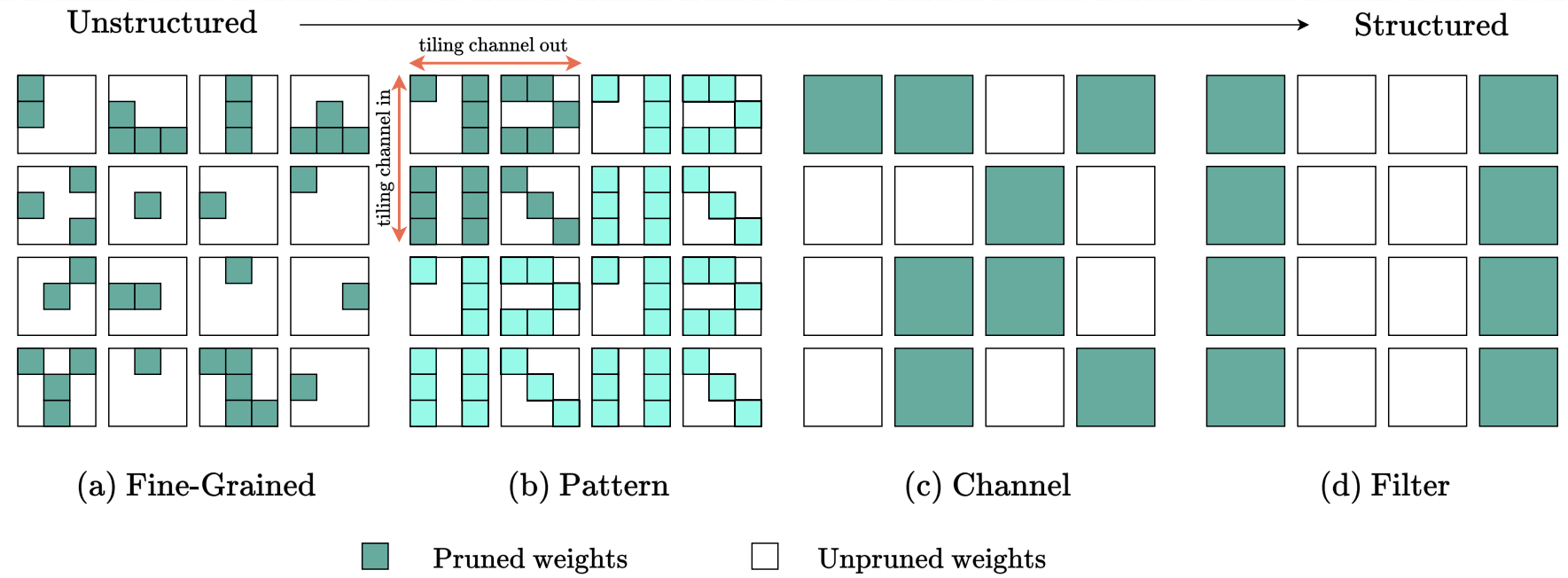
a	b	0	0	e
f	0	0	i	0
0	l	m	0	o
p	0	r	0	t
u	0	0	0	y



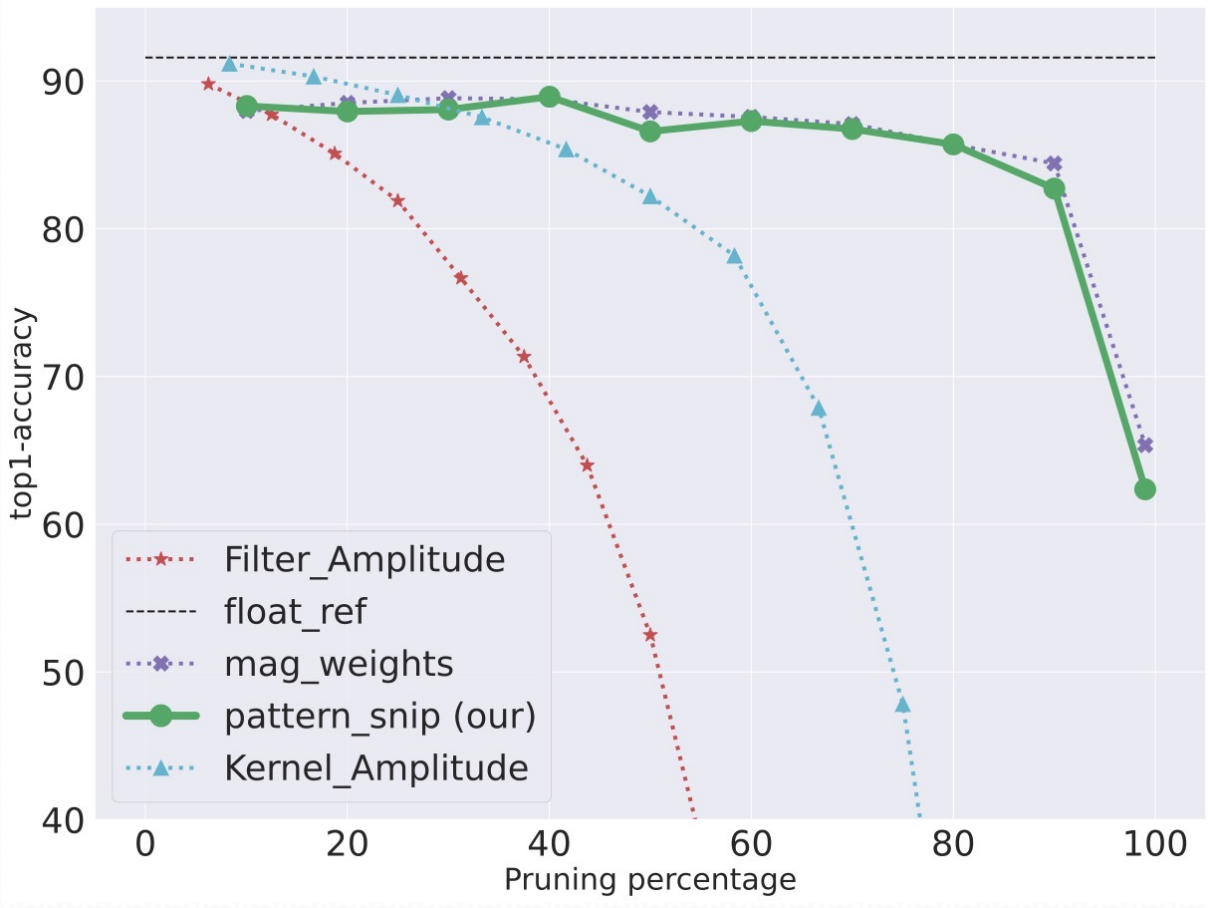
- Pruning does not always translate into energy savings

DNN Pruning

- Structured pruning provides higher efficiency



Hardware-Aware Pattern Pruning



Key Takeaways

- Energy efficiency requires deeply specialized hardware
  - Basic tasks of DNNs are easy to accelerate
- Deep knowledge of the hardware is required to propose energy-efficient models and techniques
  - Hardware-aware optimizations are mandatory
  - Structured pruning
  - Scaling quantization to large models, leverage mixed-precision
  - Efficient data-free quantization
  - Low-precision training (e.g., FP8)
- Of course, I am sorry (and not responsible?) for any rebound effect due to this work...

TARAN Team at a Glance

Rennes

Lannion

- *Domain-Specific Computers* in the post Moore's law era

- IRISA/INRIA
- ~35 people, Rennes and Lannion campuses
- from INRIA, Univ. Rennes 1, ENS Rennes

- Domain-specific computing architectures and compilers
- Energy efficiency, fault tolerance, security