An Optimization Playground for Precision and Number Representation Tuning

The case of Approximate Deep Learning Accelerators

Olivier Sentieys

Univ. Rennes, Inria

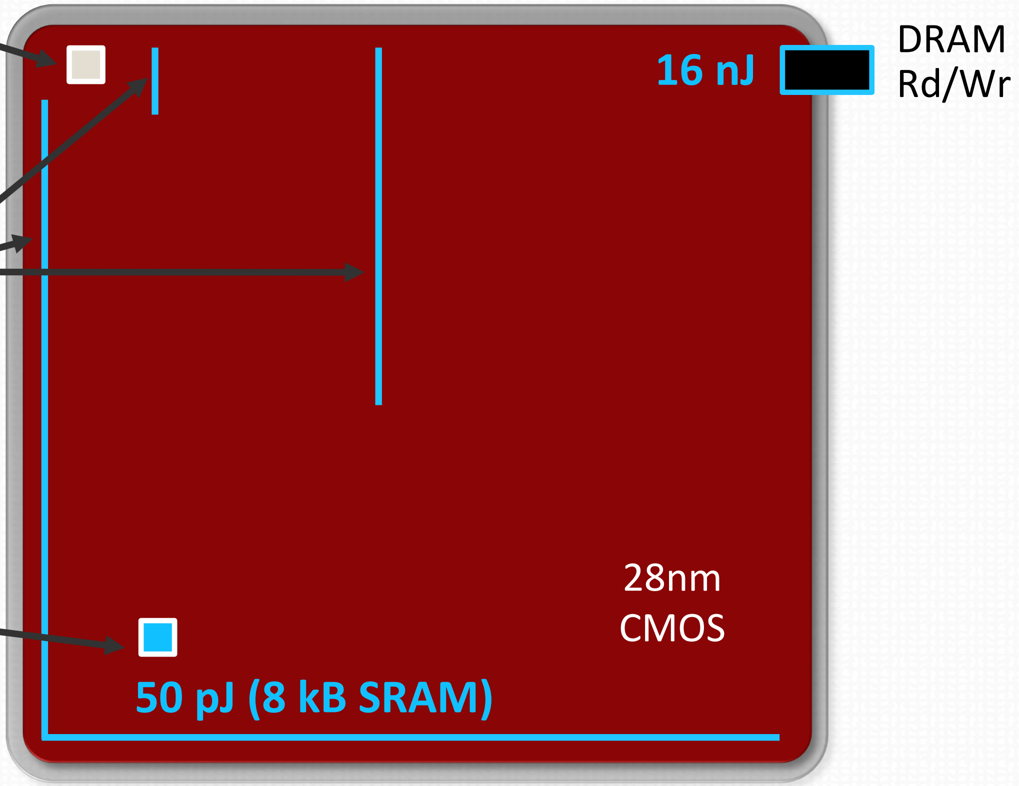
Energy Cost in a Processor/SoC

- 64-bit FPU: 20pJ/op
- 32-bit addition: 0.05pJ
- 16-bit multiply: 0.25pJ

- Wire energy
  - 240fJ/bit/mm per ⇃↾
  - 32 bits: 40pJ/word/mm
  - 8 bits: 10pJ/word/mm

- Memory/Register-File
  - Depends on word-length

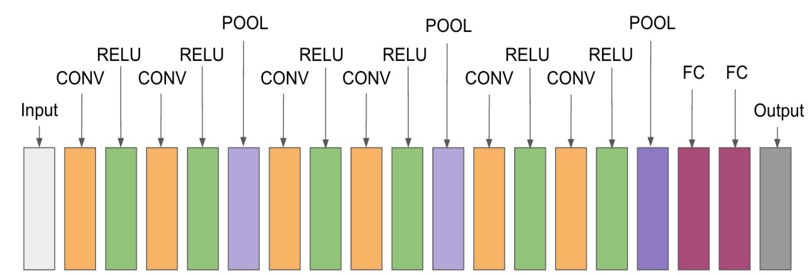
[Adapted from Dally, IPDPS'11]

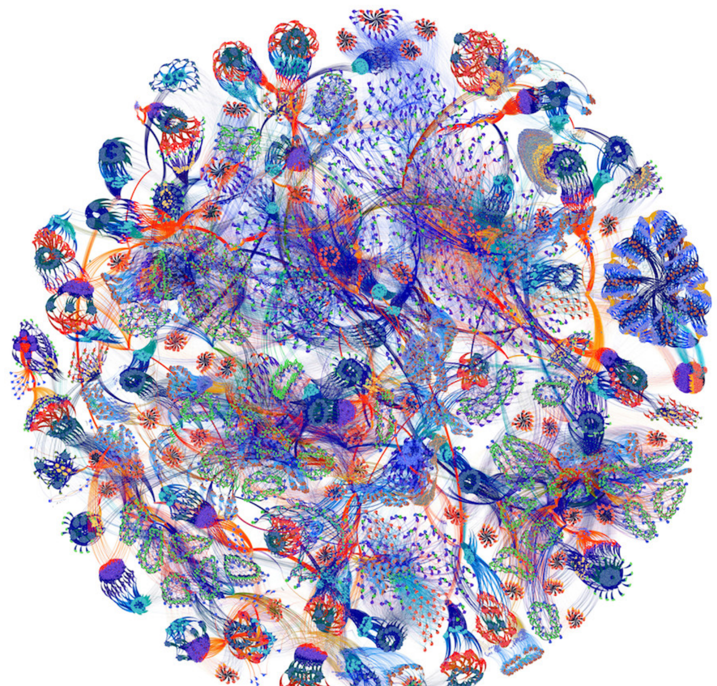


Energy strongly depends on data representation and size

Complexity Issues of Deep NNs

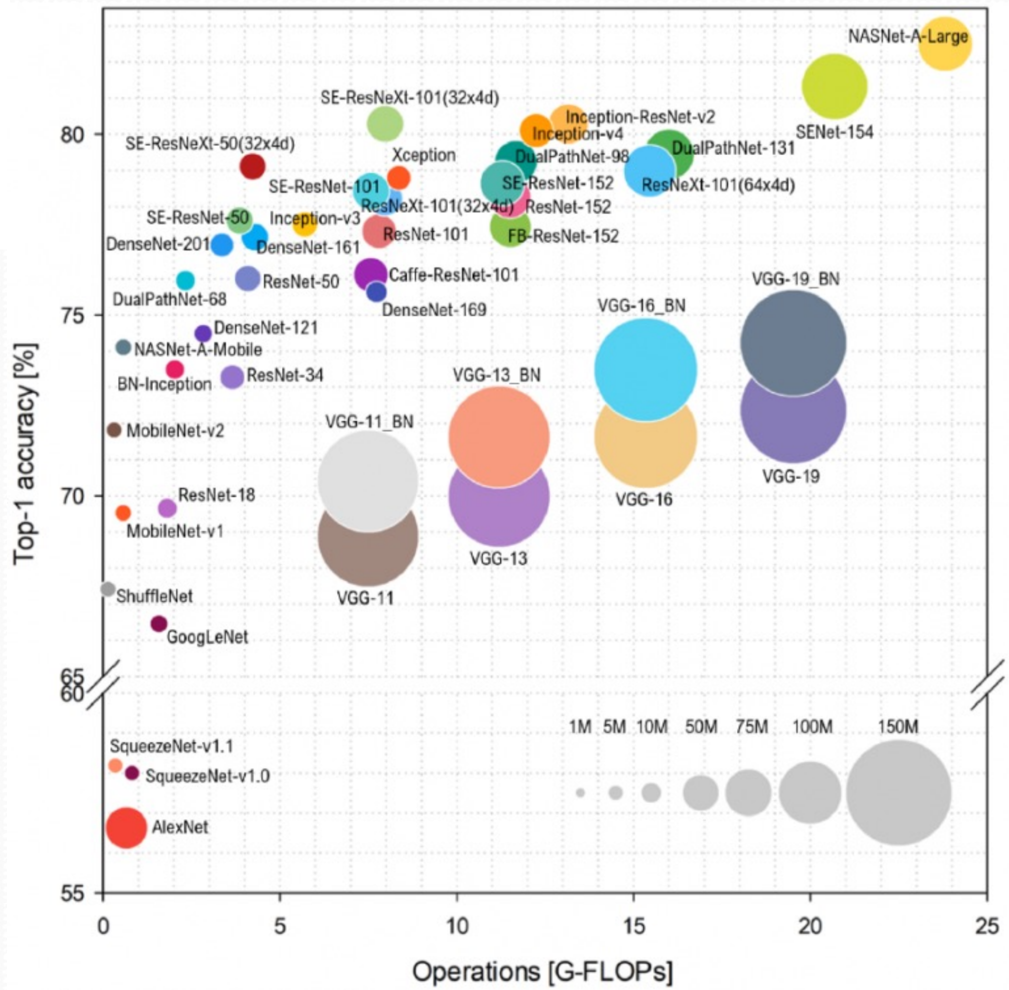
- Deep (Convolutional) Neural Networks





Poplar® graph

ResNet-50
training batch=4



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)***

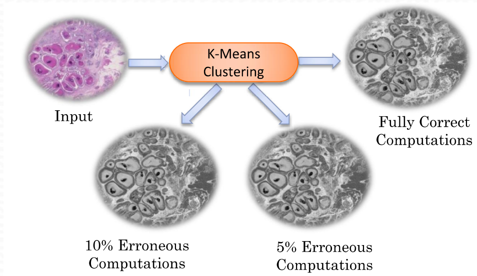
[Strubell *et al.*, ACL 2019]

- 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

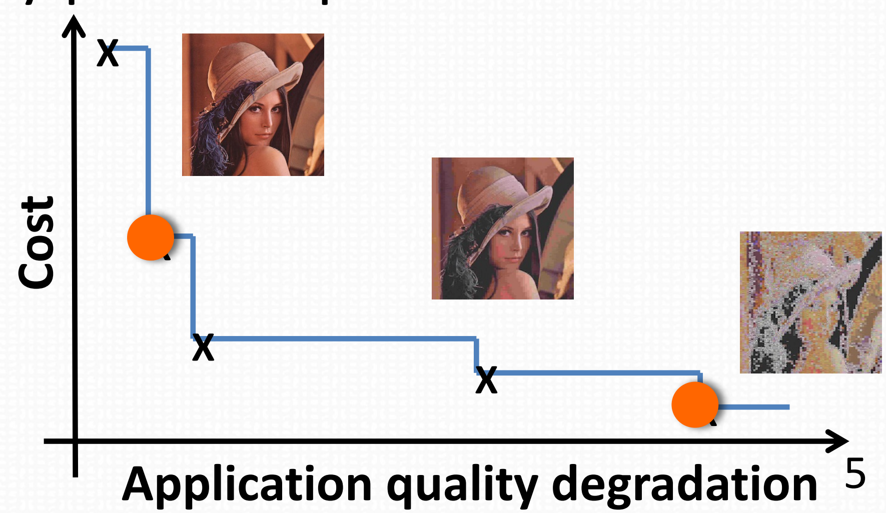
Approximate Computing

- Many applications are **error resilient**
  - media processing, data mining, machine learning, web search, etc.



- AxC performs **approximations** to reduce **energy** and increase execution speed while keeping **accuracy in acceptable limits**
  - Relaxing the need for fully precise operations
  - *Number representations and word-length*

- Design-time/run-time
- Different levels



Resilience of NNs?

Aoccdrnig to a rscheearch at Cmabrigde Uinervtisy, it deosn't mttaer in waht oredr the ltteers in a wrod are, the olny iprmoatnt tihng is taht the frist and lsat ltteer be at the rghit pclae. And we spnet hlaf our lfie larennig how to splel wrods. Amzanig, no!

[O. Temam, ISCA10]

- Our biological neurons are tolerant to computing errors and noisy inputs
- Quantization of parameters and computations provides benefits in throughput, energy, storage

This rest of this talk is about

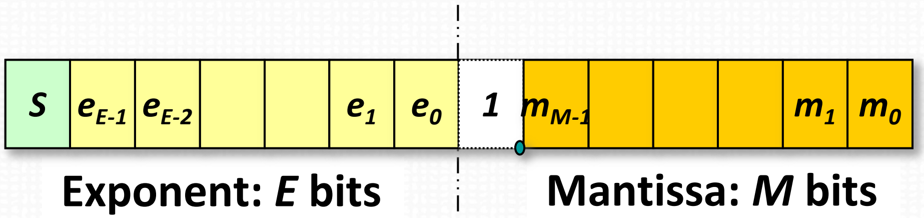
- Reducing the numerical precision of arithmetic operations is a general way to increase performance and energy efficiency in computing
  - How does this apply to DNNs?
  - Can we design low-precision accelerators for inference and training?
  - Can we do this precision tuning automatically?

Number Representations

- Floating-Point (FlP)

$$x = (-1)^s \times m \times 2^{e-127}$$

s: sign, m: mantissa, e: exponent



  – Easy to use
  – High dynamic range
  – IEEE 754

Format	e	m	bias
Single Precision	8	23	127
Double Precision	11	52	1023

- Fixed-Point (FxP)

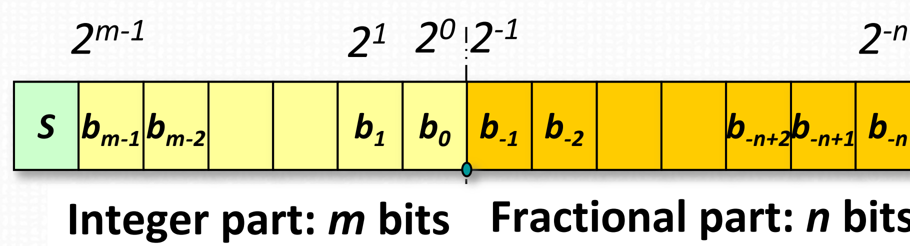
$$x = p \times K$$

p: integer, $K=2^{-n}$: fixed scale factor

  – Integer arithmetic
  – Efficient operators
    - Speed, power, cost
  – Hard to use...

$$x = s.(-2)^m + \sum_{i=-n}^{m-1} b_i.2^i$$

s: sign, m: magnitude, n: fractional

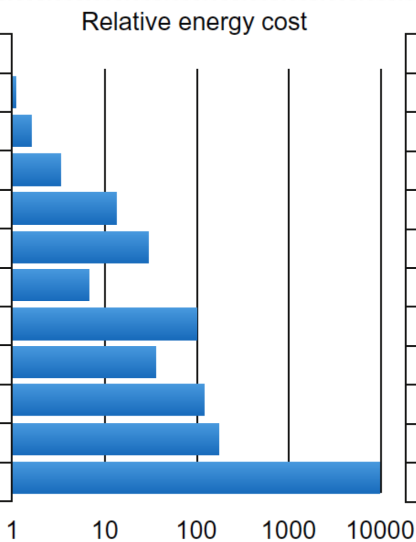


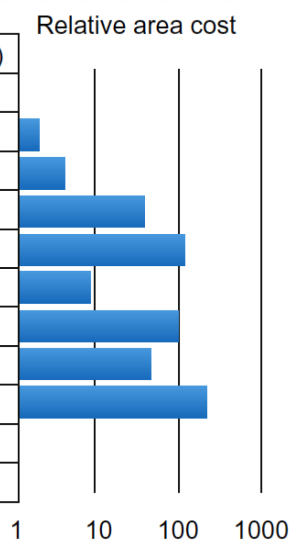
Number Representations

- Energy, delay, and area vary a lot between numeric formats and word-length

	Addition	Multiplication
8-bit integer	0.03pJ / 36μm^2	0.2pJ / 282μm^2
32-bit float	0.9pJ / 4184μm^2	3.7pJ / 7700μm^2

Operation:	Energy (pJ)	Area (μm^2)
8b Add	0.03	36
16b Add	0.05	67
32b Add	0.1	137
16b FB Add	0.4	1360
32b FB Add	0.9	4184
8b Mult	0.2	282
32b Mult	3.1	3495
16b FB Mult	1.1	1640
32b FB Mult	3.7	7700
32b SRAM Read (8KB)	5	N/A
32b DRAM Read	640	N/A



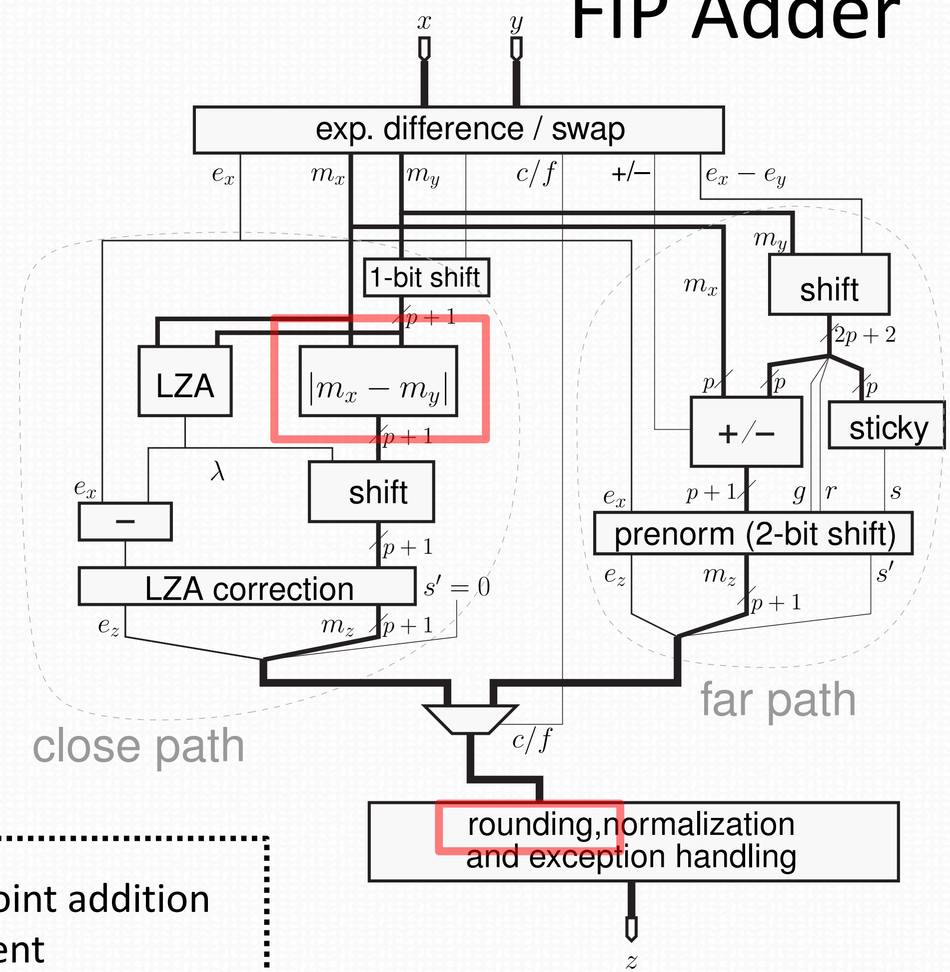


Energy numbers are from Mark Horowitz *Computing's Energy problem (and what we can do about it)*. ISSCC 2014
Area numbers are from synthesized result using Design compiler under TSMC 45nm tech node. FP units used DesignWare Library.

Floating-Point Arithmetic

- Floating-point hardware is doing the job for you!
- FlP operators are therefore more complex

FlP Adder



Fixed-point addition equivalent

[J.-M. Muller et al., Handbook of Floating-point arithmetic, Springer, 2009]

What can be customized?

- Of course precision
  - Exponent (E) and Mantissa (M) bit-width
  - *e* and *m* both impact accuracy
- Play with exponent bias
- Sub-normal numbers or not?
- 0, ∞, NaN?
- Rounding modes
  - to nearest, truncation, to 0/∞
- Inexact integer operators

LP-Floating-Point Multiplication

- Example: 7 bits, (2,5)

s_x e_x 1 0/1 $1.m_x = \{1.0; 1.5\}$

X s_y e_y 1 0/1 $1.m_y = \{1.0; 1.5\}$

$1.m_z = \{1.0; 1.5; 2.25\}$

$s_z = s_x$ **NXOR** s_y $e_z = e_x + e_y$ + (mx AND my)

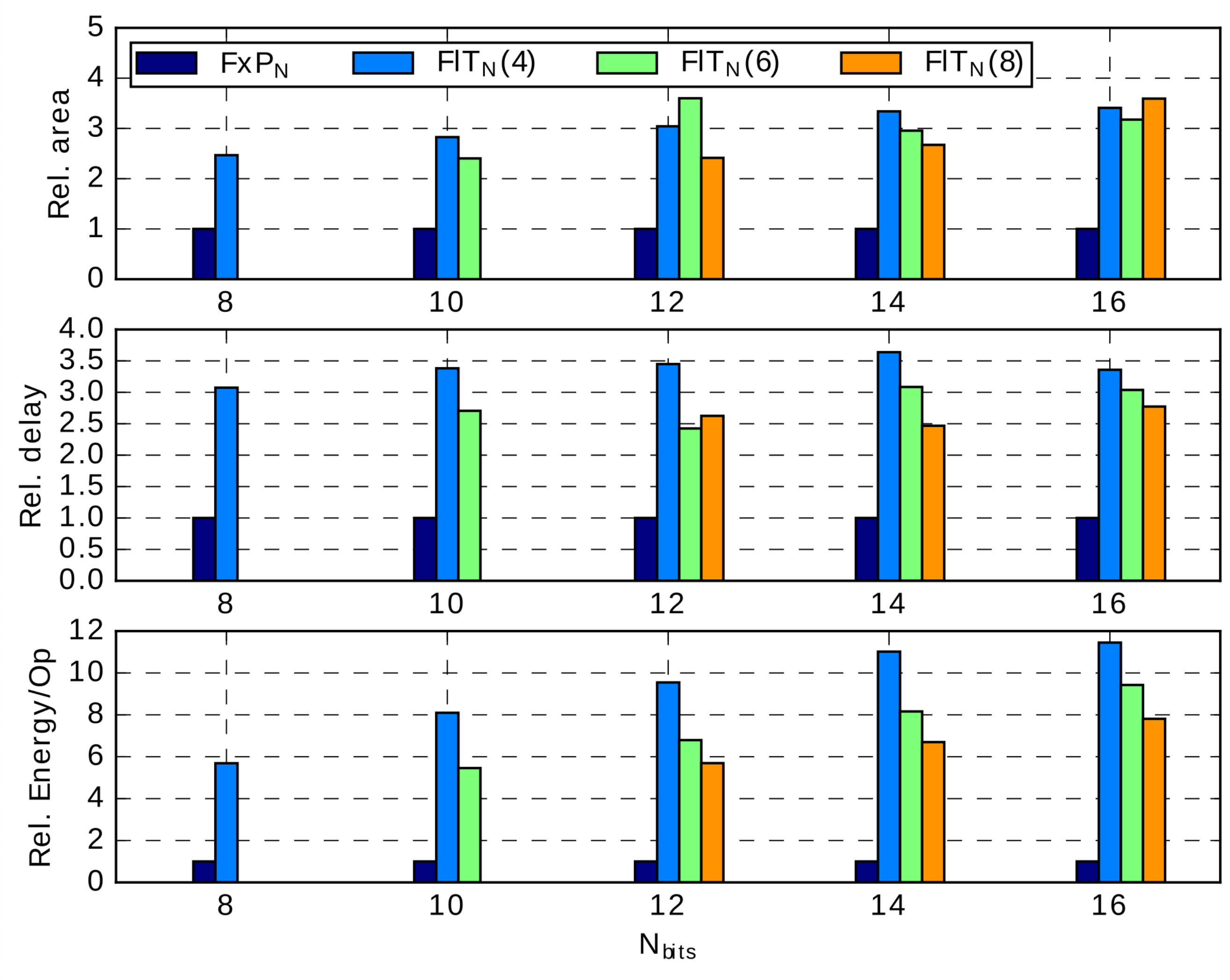
$1.m_z = \{1.0; 1.5; 1.5 \text{ or } 1.0\}$

s_z 1 $m_z = m_x$ **OR/XOR** m_y

- 5-bit adder and 3 gates!

FxP vs. FlP: Adders

- FxP_N
  - *N*-bit Fixed-Point
- $FlT_N(E)$
  - *N*-bit Float
  - Exponent *E* bits
- FxP adders are always smaller, faster, less energy

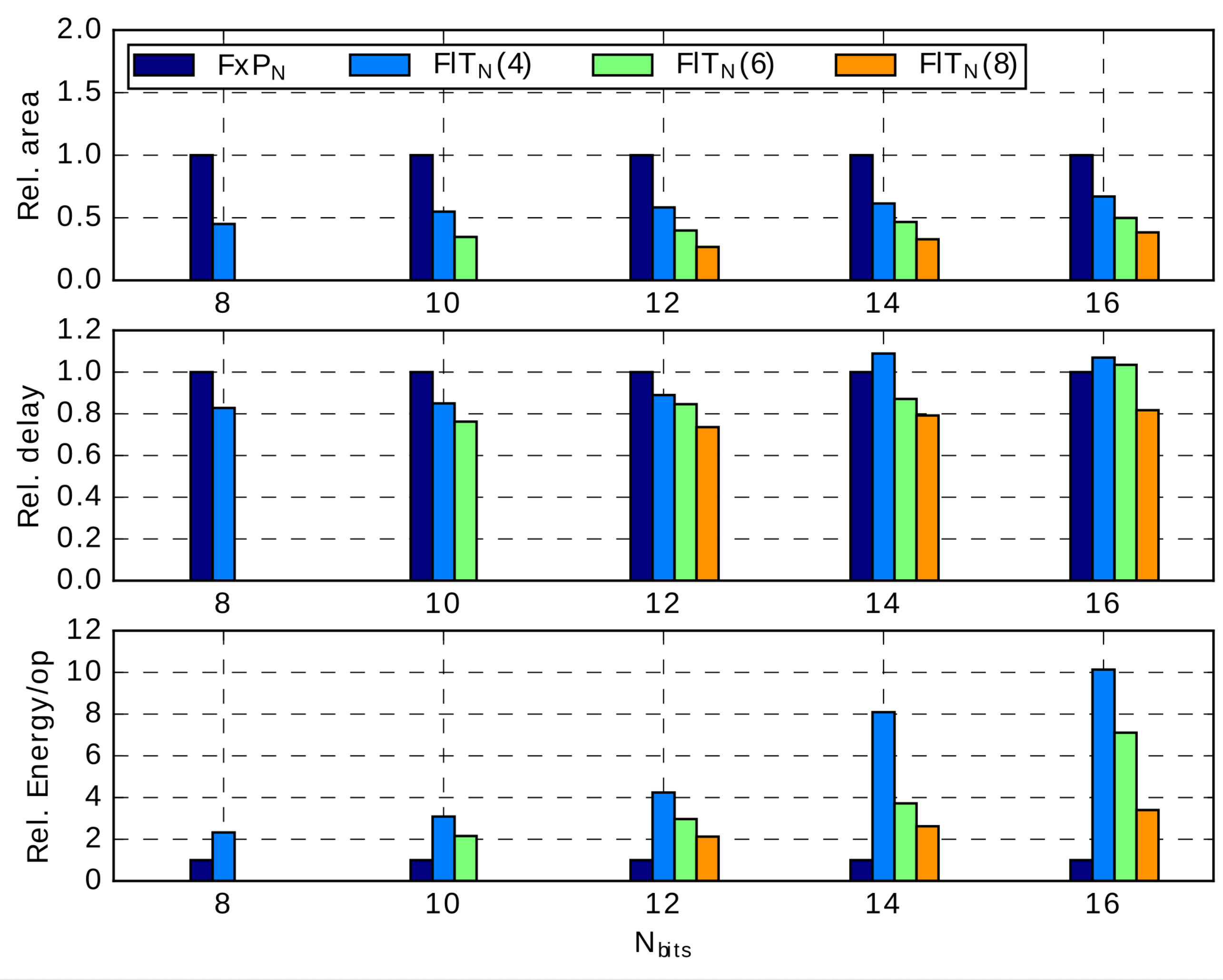


28nm FDSOI technology, Catapult (HLS), Design Compiler, PrimeTime

FxP vs. FlP: Multipliers

- FxP_N
  - Fixed-Point
  - *N* bits
- $FlT_N(E)$
  - Floating-Point
  - *N* bits
  - Exponent *E* bits

- FlP multipliers are smaller, faster, but consume more energy



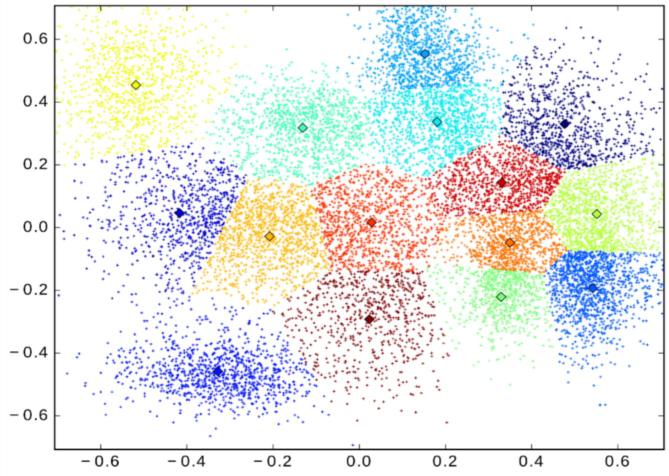
28nm FDSOI technology, Catapult (HLS), Design Compiler, PrimeTime

Custom Floating-Point

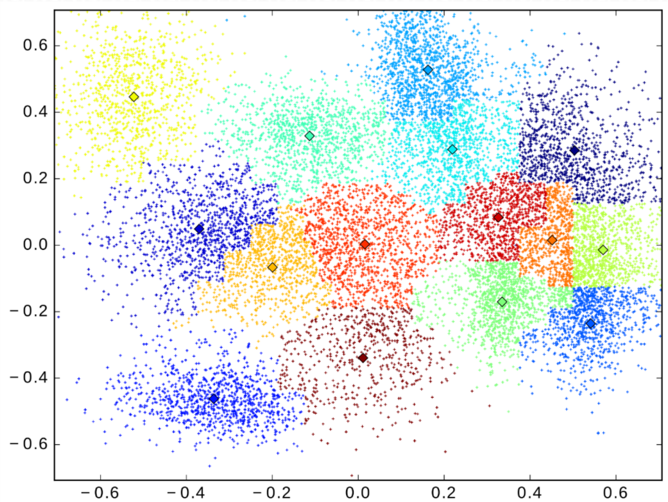
- Difference in cost/energy between float/fixed is smaller for low-precision operators

- Slower increase of errors for floating-point
  - e.g., 8-bit float is still effective for K-means clustering
    [SiPS'17]

Approximate K-Means Clustering



Reference: double



Floating-Point: ct_float_8
5-bit exponent
3-bit mantissa

Custom Floating-Point

- `ct_float`: a Custom Floating-Point C++ Library `https://gitlab.inria.fr/sentieys/ctfloat`
  - Synthesizable (with HLS) library
  - Templated C++ class

    `ct_float<`*e*,*m*,*r*`>`

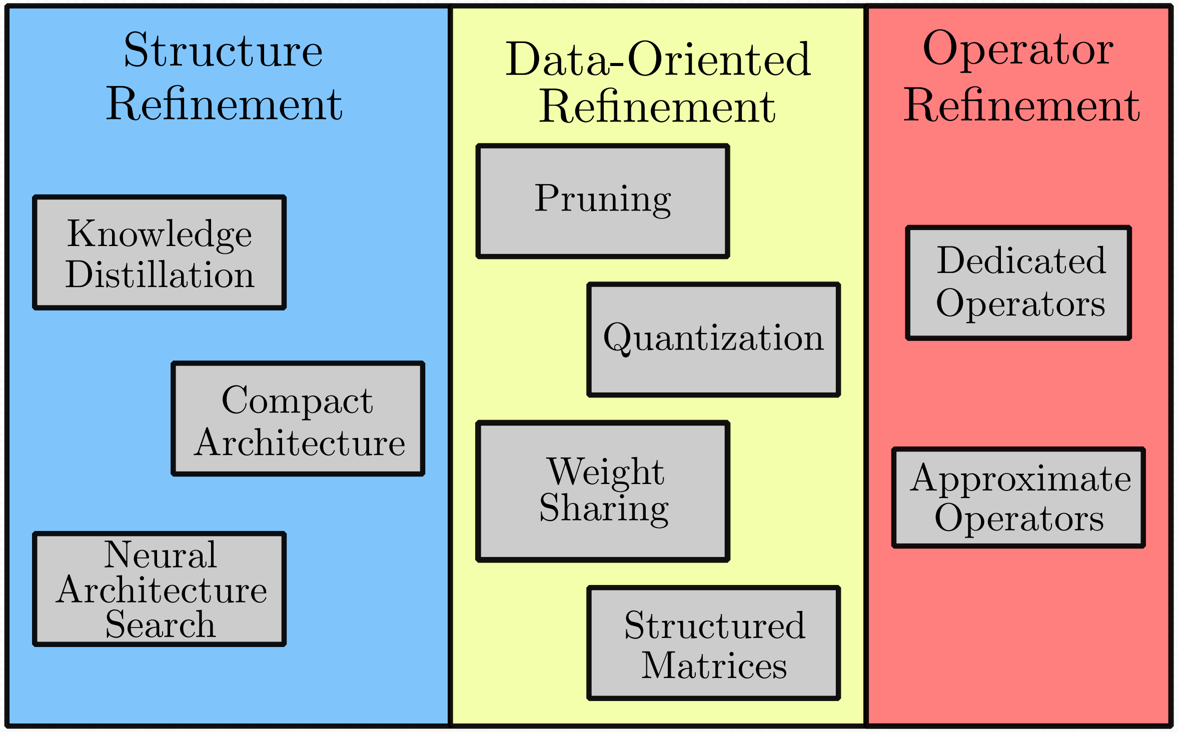
    - Exponent width e (int)
    - Mantissa width m (int)
    - Rounding method r
    - Bias b

```
ct_float<8,12,CT_RD> x,y,z;
x = 1.5565e-2;
z = x + y;
```

- Many possible design points
  - latency constraints, rounding modes, etc.

How does this apply to DNNs?

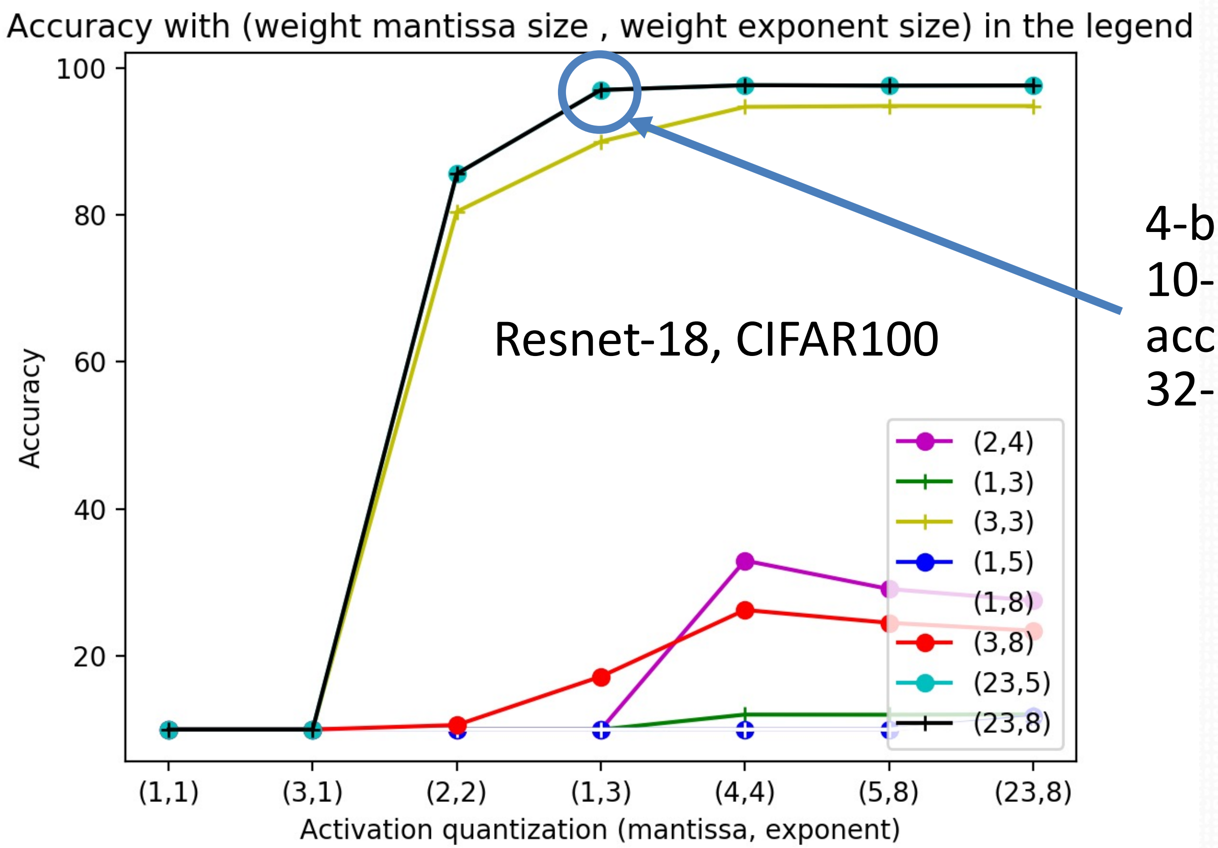
Approximate DNNs



- Float
  - half-precision
  - Bfloat16
- Fixed-point
  - INT8
- Block floating-point
- BNN/TNN

Approximate DNNs: Low-Precision

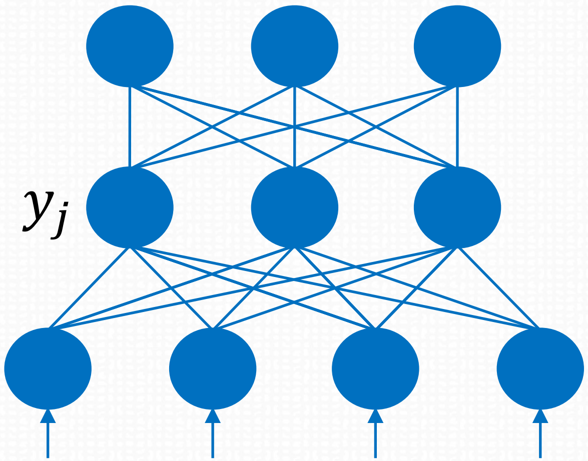
- Not only Weights, but also Activations, Per-Layer Quantization, etc.



4-bit activations and 10-bit weights keeps accuracy near (98.4%) 32-bit float reference

What is still difficult: learning

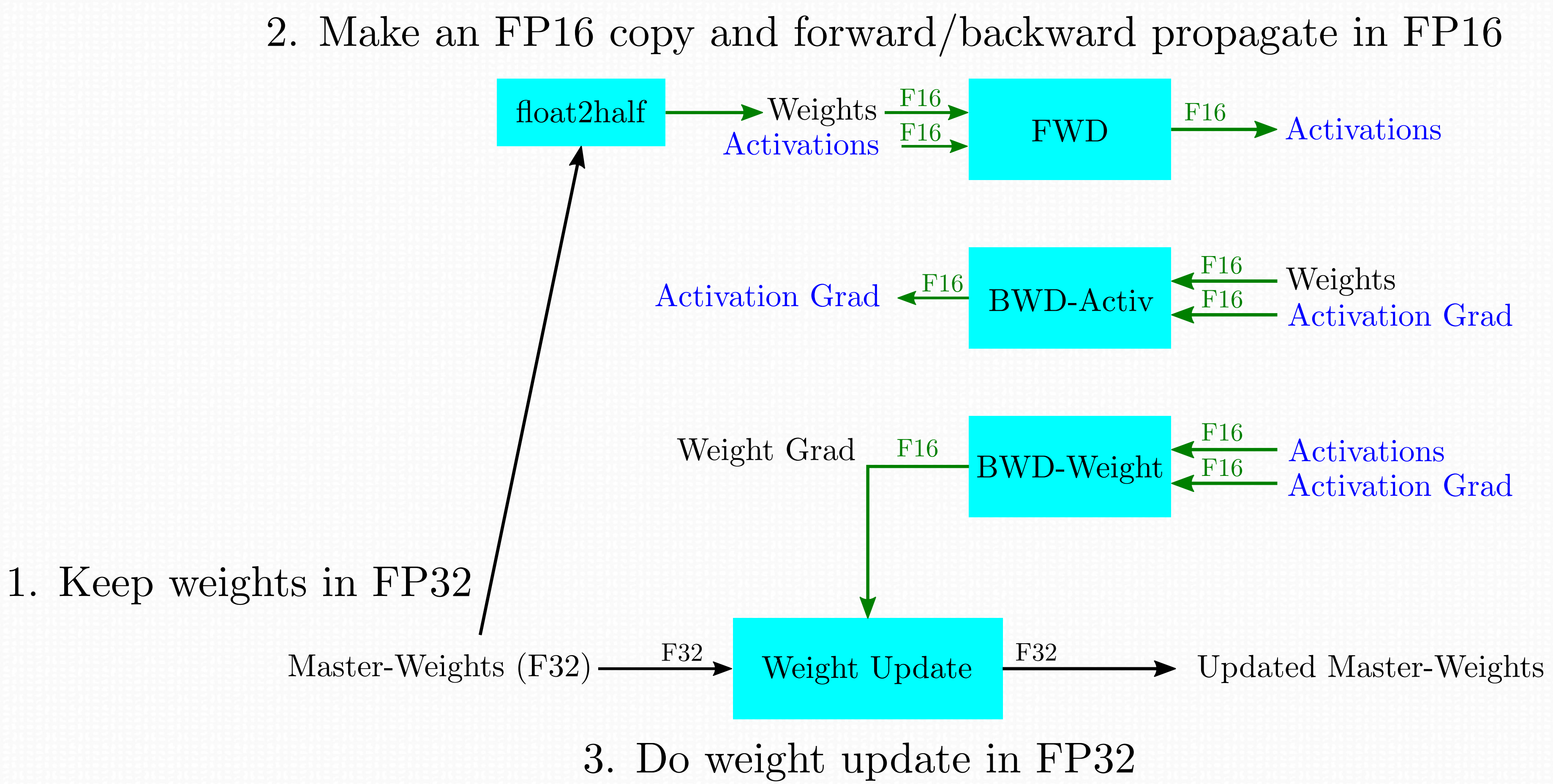
- Learning: gradient descent and backpropagation



$$w_{ij}^{t} = w_{ij}^{t-1} - \alpha \frac{\partial \ell}{\partial w_{ij}^{t-1}}$$

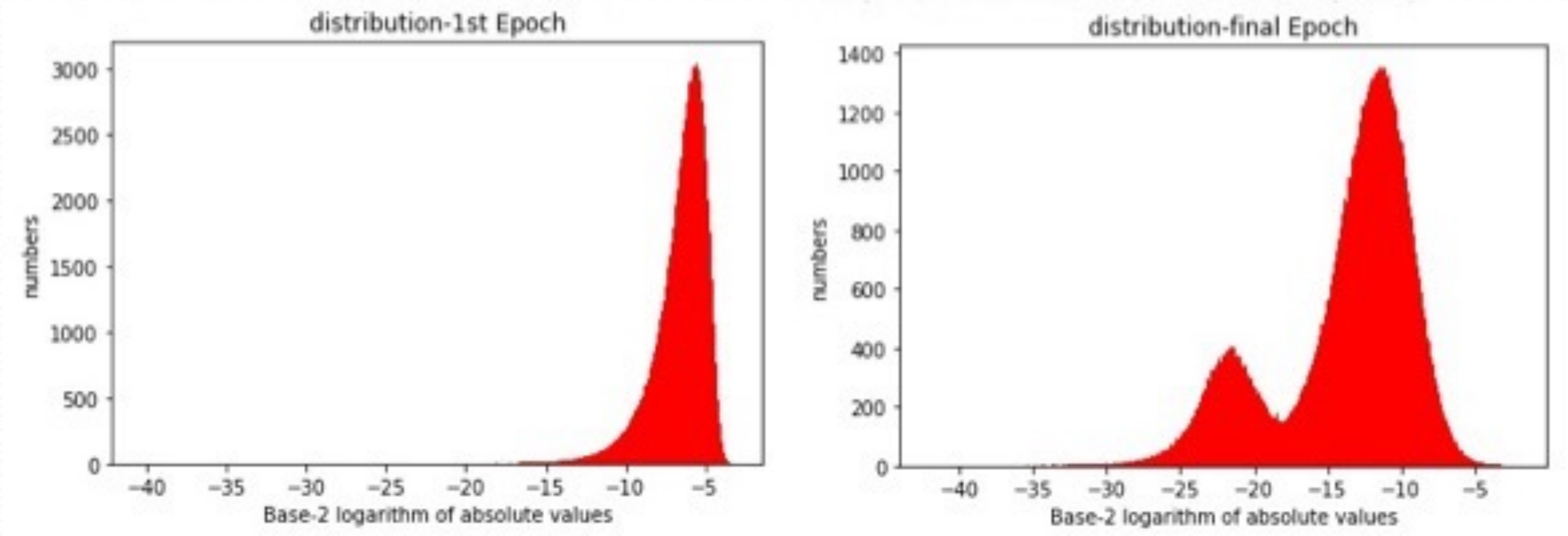
- This is very expensive to compute, even in HW
  - Approximating and accelerating learning is much more difficult

Mixed-Precision Training

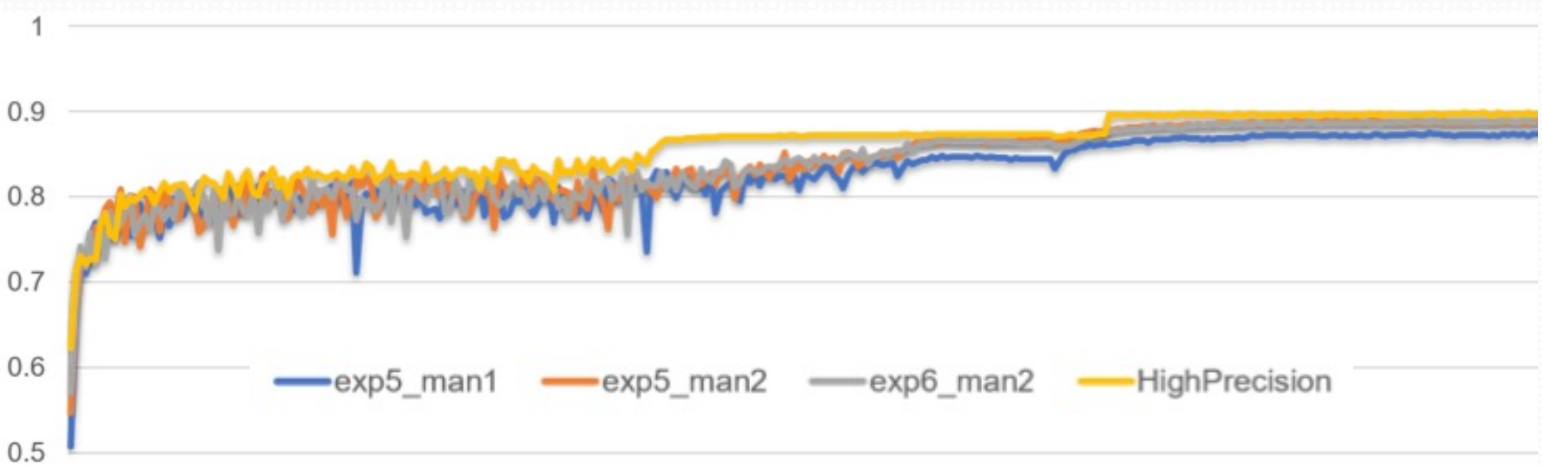


[NVIDIA, Mixed precision training, 2018]

Low-Precision Training of DNNs



VGG16 training with Cifar-10

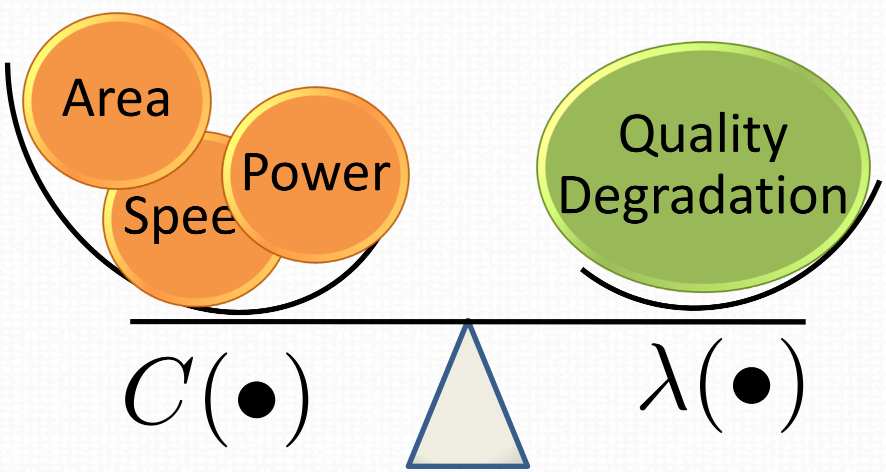


Can we Tune Precision Automatically?

Automatic Precision Tuning

[also Word-Length Optimization (WLO)]

- Optimization process that
  – determines the number of bits for each data
  – minimizing a cost function *C*
  – constrained by (application) quality degradation λ
    - e.g., noise power, SSIM, abs. error



Automatic Precision Tuning

Multiple Word Length

Variable	Integer Word Length	Fractional Word Length	
x_0	1 1 1 0 1 1 1	0 1 0 0 1 0 1 1 0	w_0
x_1	1 0 1 0 1 0 1	1 1 0 1 1 0 1 1 1	w_1
x_2	1 0 1 0 0 1 1	0 1 0 1 1 1 1 0 1	w_2
…	…	…	…
x_k	1 0 1 1 1 1 1	1 0 0 1 1 0 0 1 1	w_k
…	…	…	…
x_{N-1}	1 1 1 1 0 1 0	0 1 1 1 1 0 0 1 1	w_{N-1}

$\Rightarrow O(B^N)$

Fixed-Point Arithmetic

N: number of variables

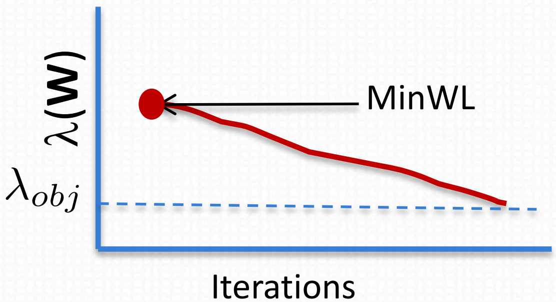
B: number of bits to explore per variable

Automatic Precision Tuning

- Multi-variable word-length optimization

$$\min\left(C(\mathbf{w})\right) \quad \text{subject to} \quad \lambda(\mathbf{w}) \leq \lambda_{obj}$$

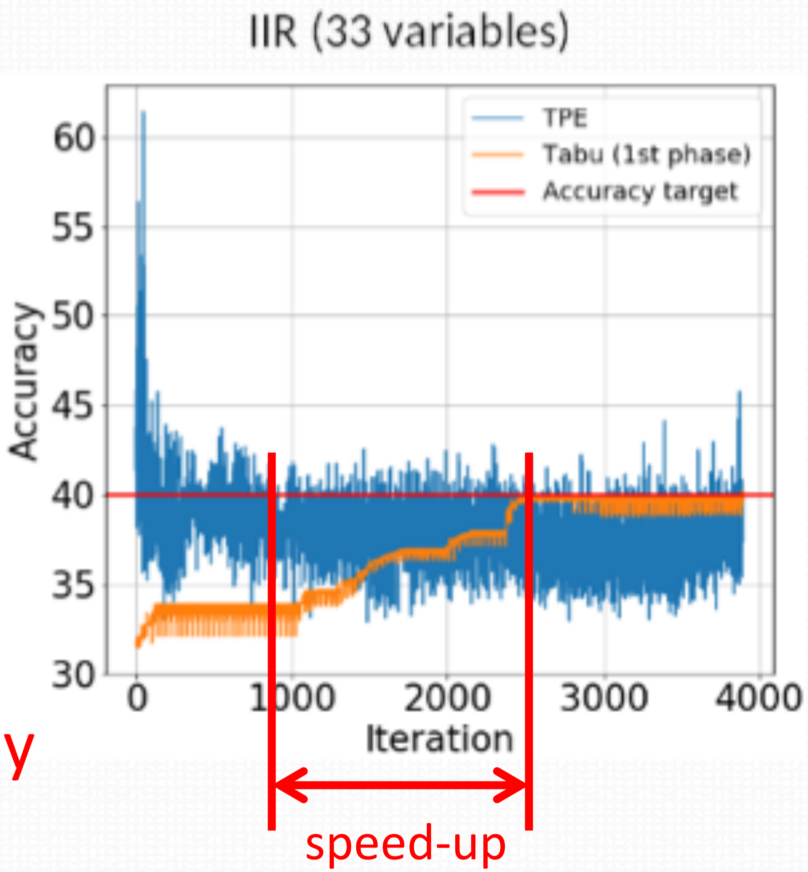
- Known to be non-convex and NP-hard
- Optimized using heuristic rules, iterative optimization process, stochastic approaches



$\lambda(\boldsymbol{w})$: accuracy degradation of solution $\boldsymbol{w}$
$C(\boldsymbol{w})$: cost of solution $\boldsymbol{w}$
Data word lengths: $\boldsymbol{w}=\{w_0, w_1, \ldots, w_{N-1}\}$
Maximum degradation: λ_{obj}

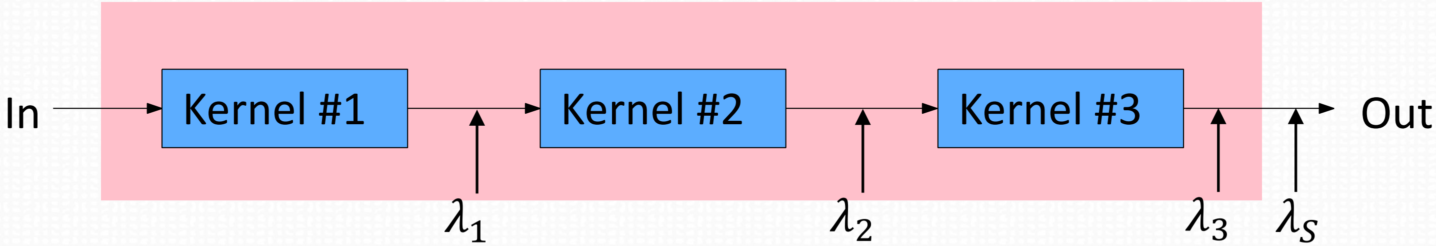
Speeding-up Global Search

- Combine Bayesian Optimization and Local Search
  - Bayesian Optimization for narrowing down solution space
  - Fine-tuning with local search
- Transition point based on statistical metrics
  - word-lengths (WL) are distributed with low variance
    - e.g., with less than 1 bit
- Optimization time is reduced by 50-80% w.r.t. best algorithm with similar cost



Scaling the WLO Procedure

- Large system sizes present enormous complexity
  - Too many variables for global optimization

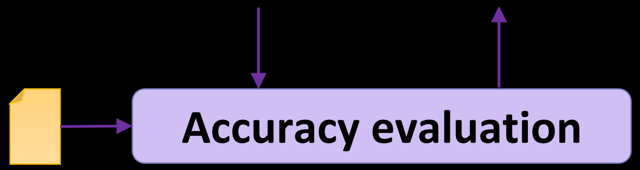


Multi-kernel approach

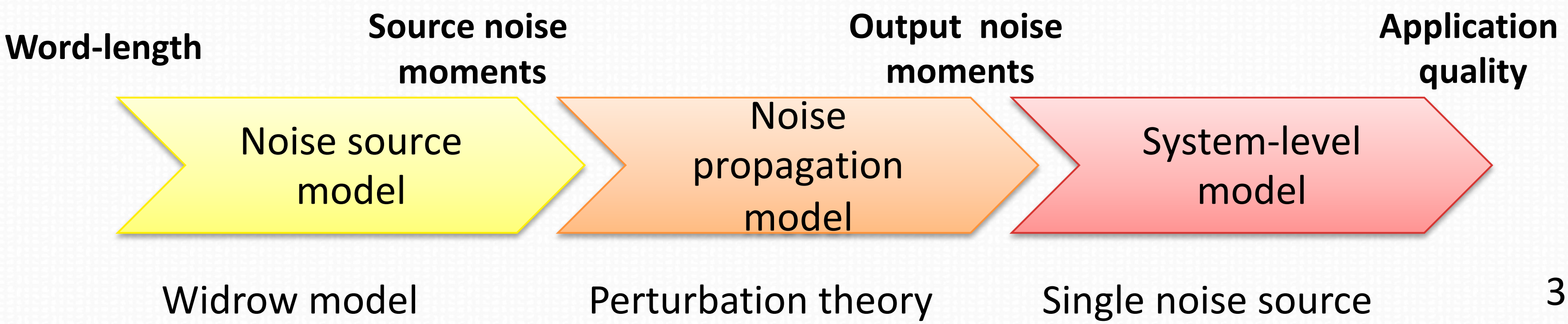
- Key idea: construct models that express
  - impact of **noise budgets** to Cost and Accuracy
  - relation among **noise budgets**
- Significantly reduce exploration time and improve the quality of the solutions for large applications

[DATE'20]

Accuracy Evaluation

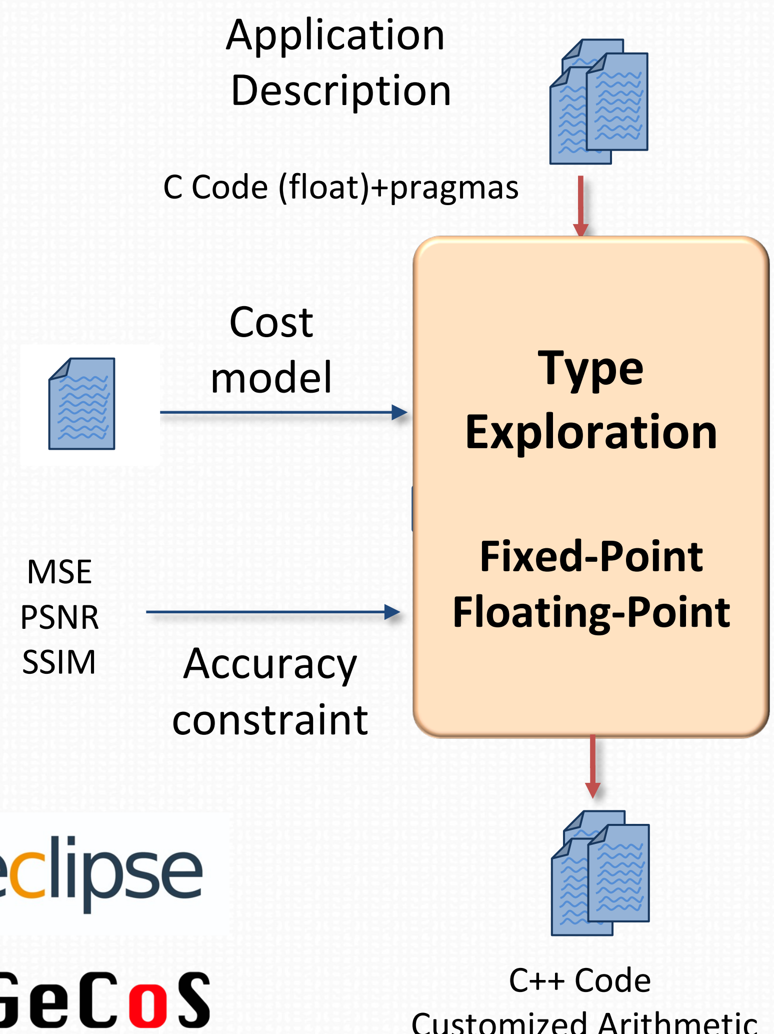


- One of the most time consuming tasks during precision tuning
- Models for quantization effect analysis
  - Analytical accuracy evaluation
  - System-level estimation [ICCAD'14, DATE'16]
  - Speeding-up simulations [DATE'20, ICCAD'14]

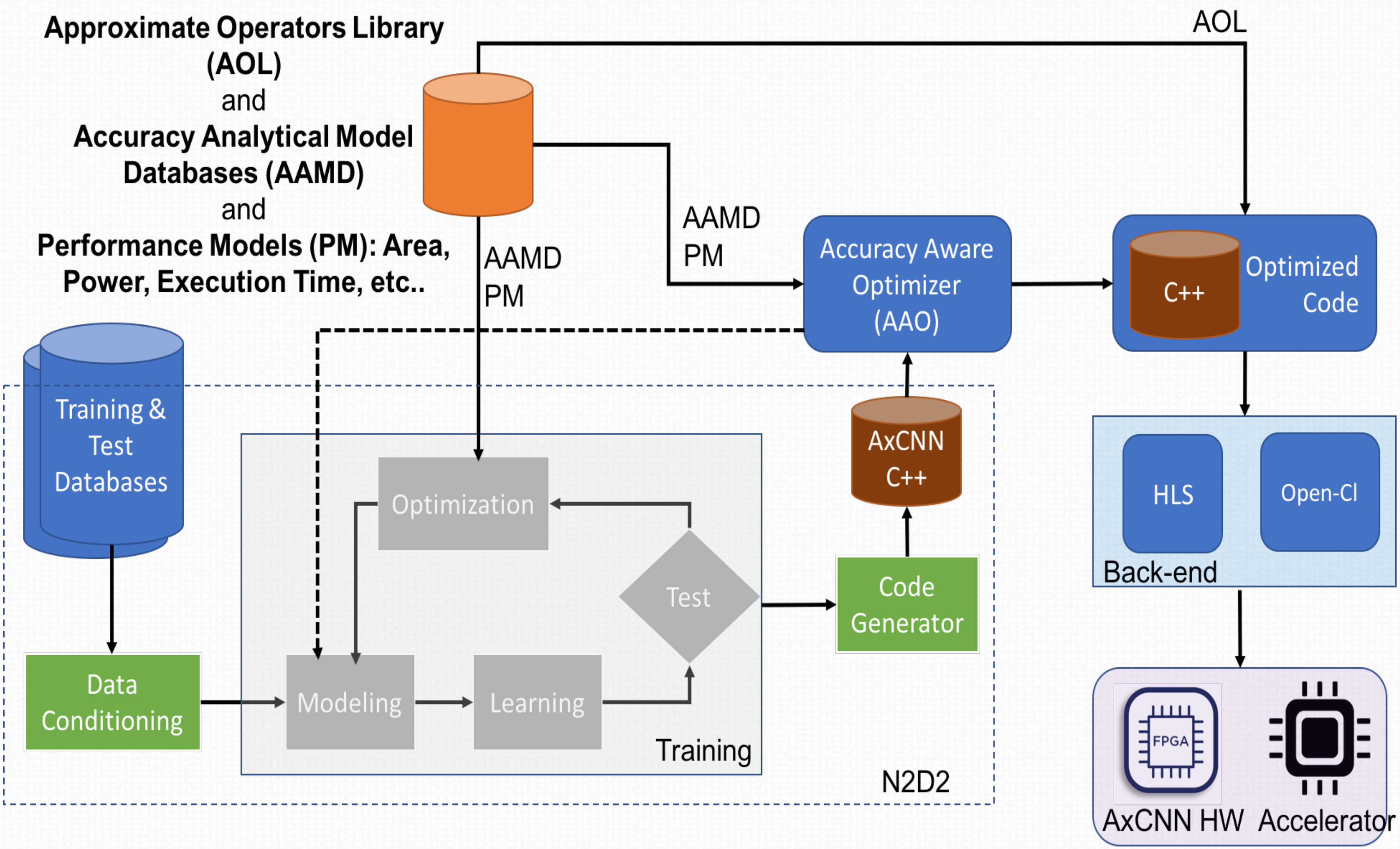


TypEx: A Framework for Type Exploration

- Source-to-source
  - C code in float to C code using custom arithmetic
- Word-length optimisation
  - fixed or float



Accuracy and Hw Aware Exploration



Conclusions

- Most applications tolerate imprecision
- Playing with precision is an effective way to save energy consumption
  - Number representations, low-precision
  - Not only computation, but also memory and transfers
  - Run-time accuracy adaptation would increase energy efficiency even further
- Low-Precision Training and Inference

Open Issues

- Exploring number representations and word-length is a difficult problem for large applications
  - Mainly limited by simulation time to evaluate accuracy
  - Automatizing the choice between (or combining) float and fixed is a challenge
    - Towards an automatic optimizing compiler framework
  - Domain-specific knowledge is a key
- Evaluating cost is also an important (and less studied) issue
  - e.g., #weights alone is not a good metric
  - e.g., unstructured pruning reduces performance
  - Hardware-aware pruning/quantization requires a good cost model