An Optimization Playground for Precision and Number Representation Tuning

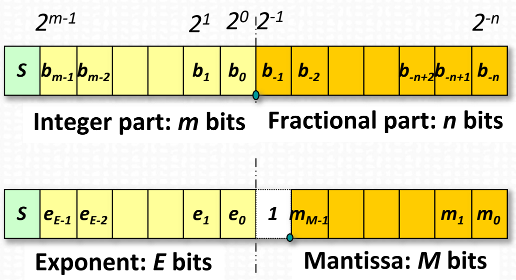
Olivier Sentieys

Univ. Rennes, Inria

with Van-Phu Ha, Tomofumi Yuki, Daniel Ménard and others

Computer Arithmetic

- At the core of computing we find **number representations** (e.g., real) and **arithmetic operations** (e.g., +, ×, ÷, √)
- Numeric formats
  - – Fixed-Point (integer)
  - – Floating-Point



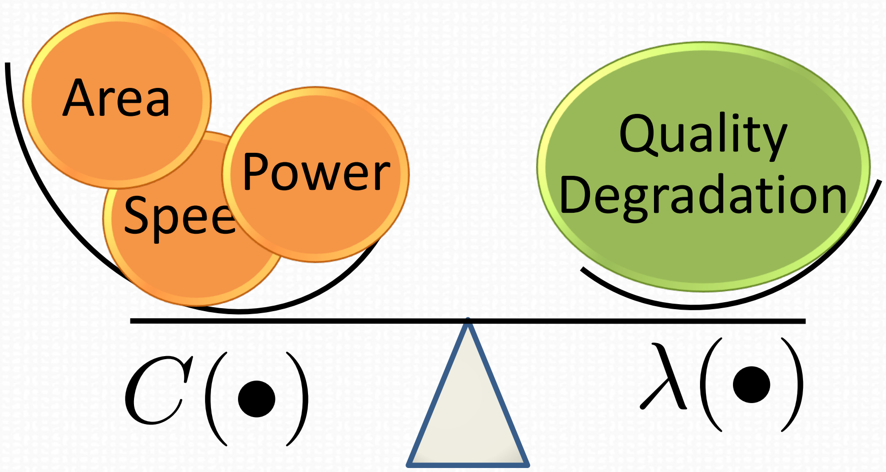
- Energy, delay, and area vary a lot between numeric formats and word-length

	Addition	Multiplication
8-bit integer	0.03pJ / 36μm²	0.2pJ / 282μm²
32-bit float	0.9pJ / 4184μm²	3.7pJ / 7700μm²

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}

➡ $O(B^N)$

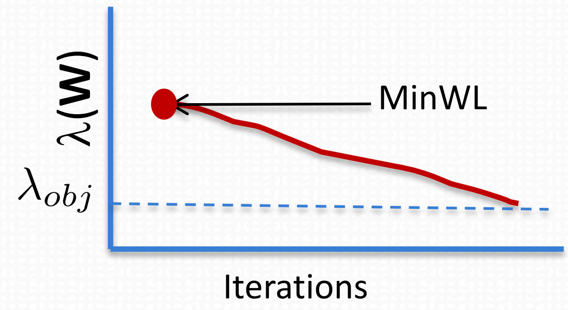
N: number of variables
B: number of bits to explore per variable

Automatic Precision Tuning

- Multi-variable word-length optimization

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

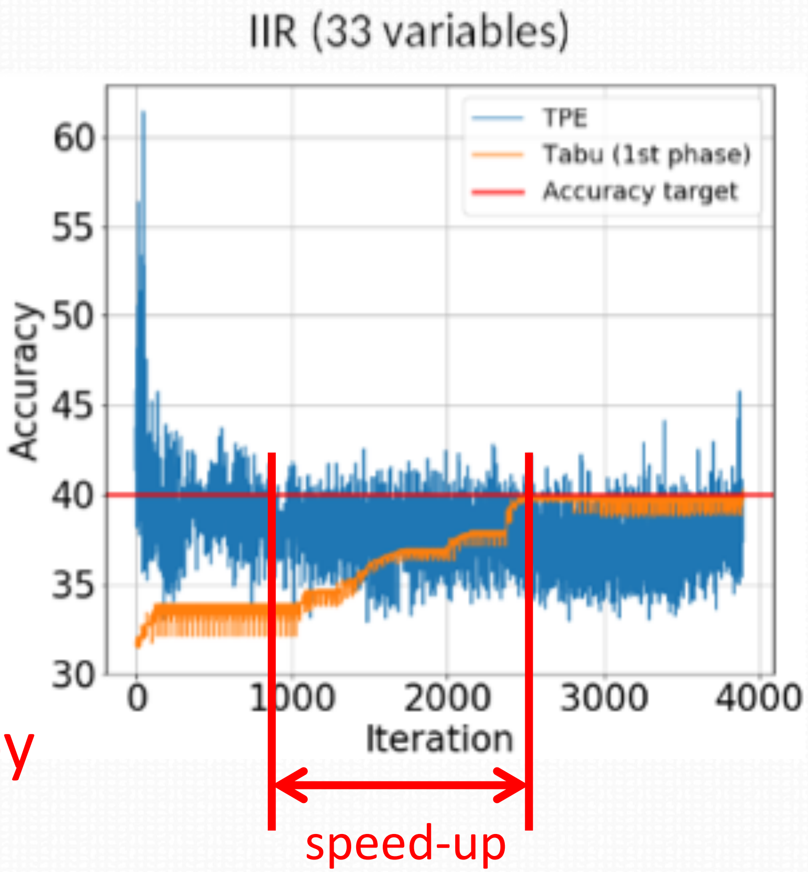
- Known to be non-convex and NP-hard
- Optimized using heuristic rules, iterative optimization process, stochastic approaches



$\lambda(\boldsymbol{w})$: 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}

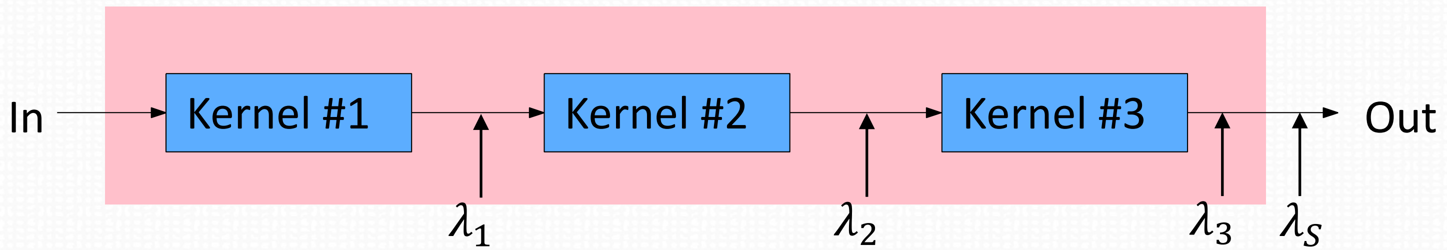
Speeding-up Global Search

- 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

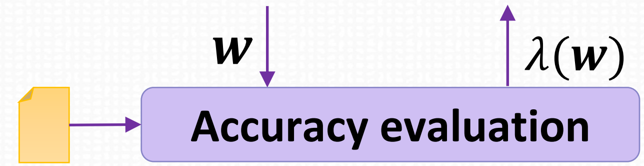


Multi-kernel approach

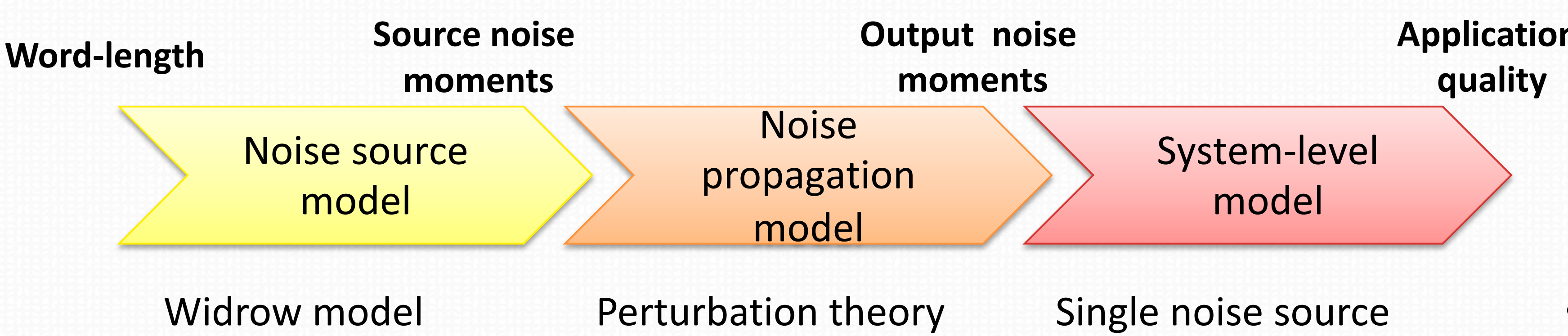
- Key idea: construct models that express
  - impact of **noise budgets** to Cost and Accuracy
  - relation among **noise budgets**
- Significantly reduce exploration time and improve the quality of the solutions for large applications

[DATE'20]

Accuracy Evaluation

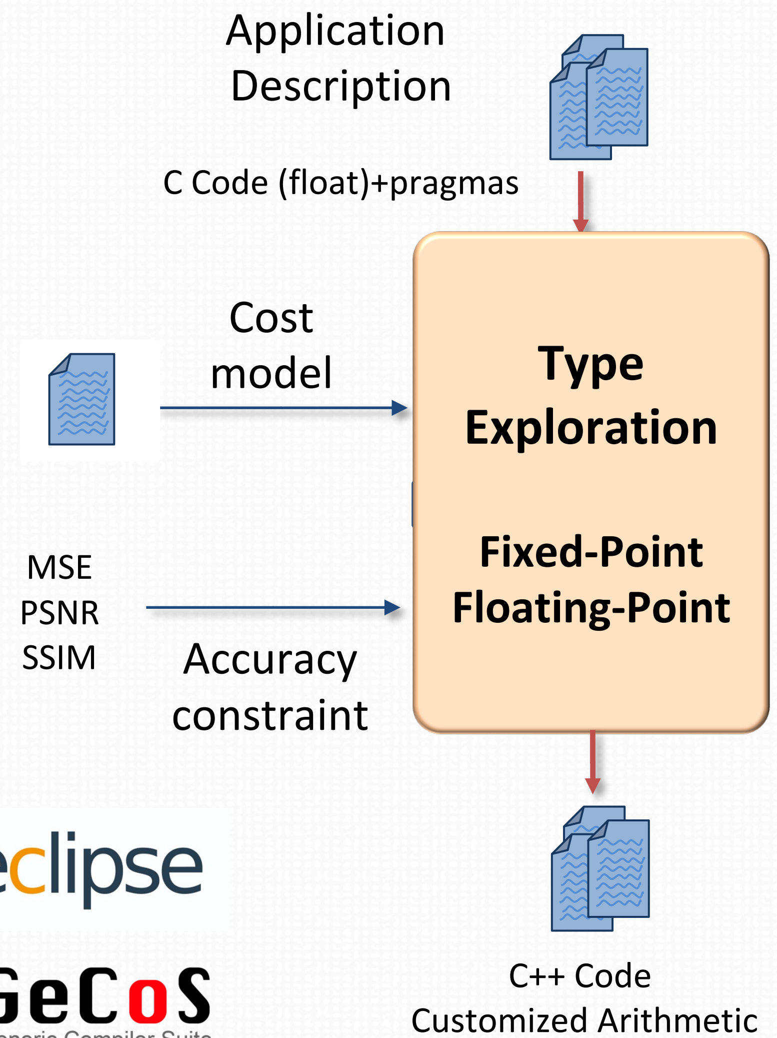


- One of the most time consuming tasks during precision tuning
- Models for quantization effect analysis
  - Analytical accuracy evaluation
  - System-level estimation [ICCAD'14, DATE'16]
  - Speeding-up simulations [DATE'20, ICCAD'14]



TypEx: A Framework for Type Exploration

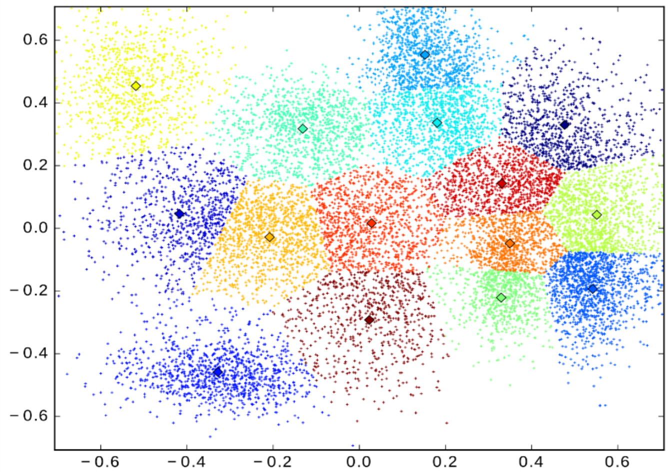
- Source-to-source
  - C code in float to C code using custom arithmetic
- Word-length optimisation
  - fixed or float



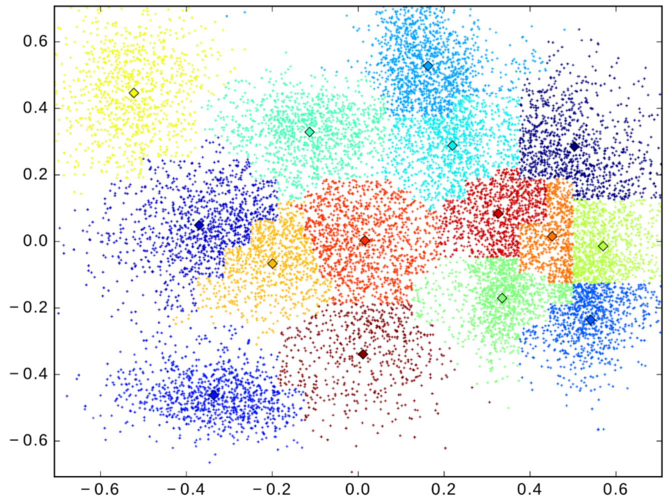
Custom Floating-Point

- Slower increase of errors for floating-point
  - e.g., 8-bit float is still effective for K-means clustering [SiPS'17]
- Difference in cost/energy between float/fixed is small for low-precision operators

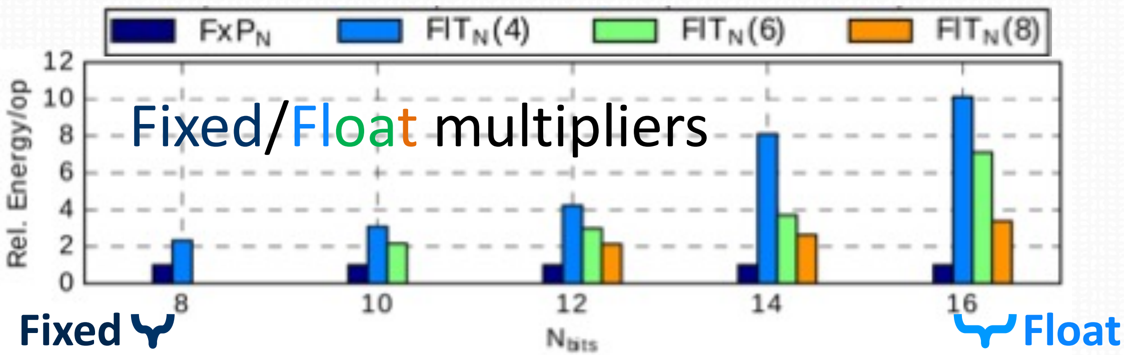
Approximate K-Means Clustering



Reference: double



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



Custom Floating-Point

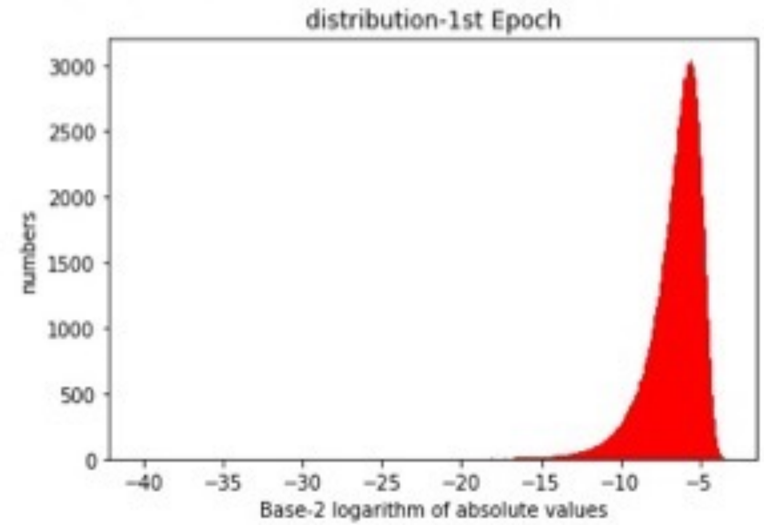
- `ct_float`: a Custom Floating-Point C++ Library `https://gitlab.inria.fr/sentieys/ctfloat`
  - Synthesizable (with HLS) library
  - Templated C++ class
    `ct_float`<*e*,*m*,*r*>
    - Exponent width e (int)
    - Mantissa width m (int)
    - Rounding method r
    - Bias b

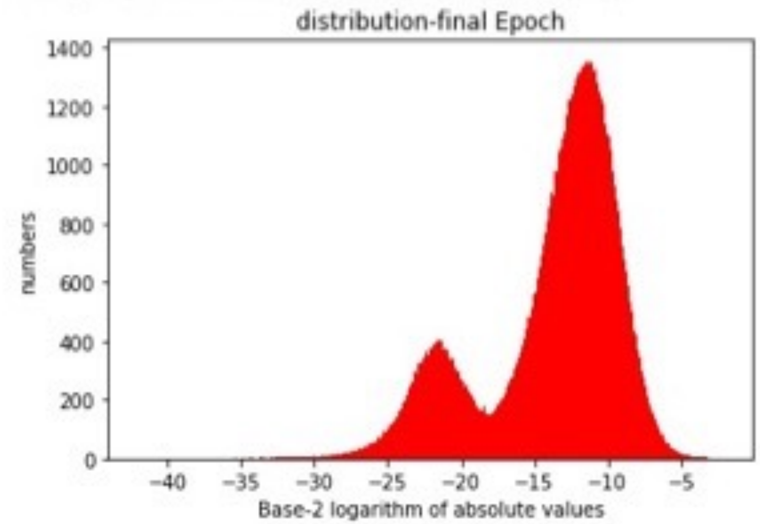
```
ct_float<8,12,CT_RD> x,y,z;
x = 1.5565e-2;
z = x + y;
```

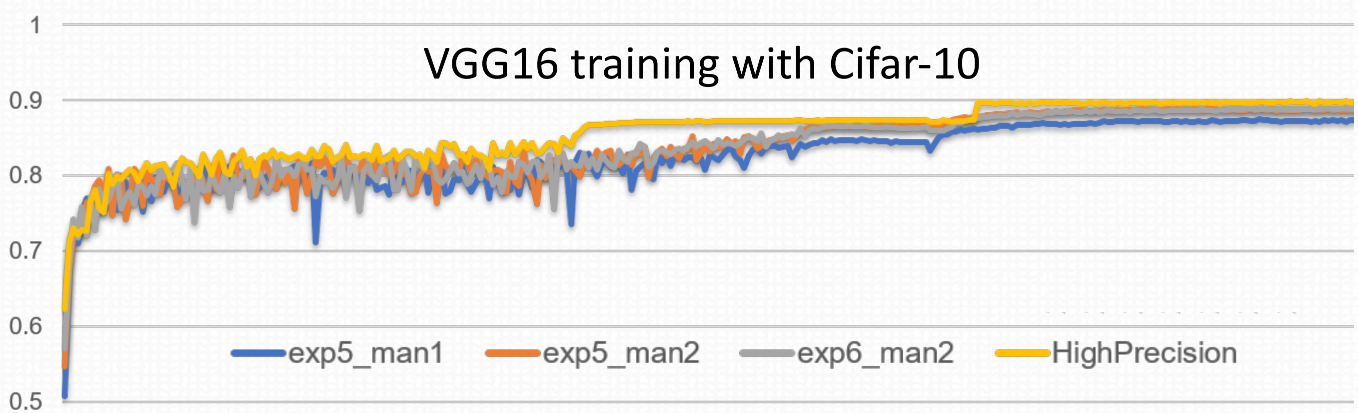
- Many possible design points
  - latency constraints, rounding modes, etc.

Work in Progress

- Low-Precision Training of DNNs







Open Issues

- WLO is still a difficult problem for large applications
  - Mainly limited by simulation time to evaluate $\lambda(\boldsymbol{w})$, analytical evaluation still limited
- Evaluating cost $C(\boldsymbol{w})$ is also an important (and less studied) issue
  - Resource sharing, complexity related to one w_i
- Automatizing the choice between (or combining) float and fixed is a challenge
  - Towards an automatic optimizing compiler framework
    - considering both float and fixed representations
    - combining WL/Number Representation optim.

More Open Issues

- Compiler level: identify candidate computation kernels for approximations
- Hardware-level: build a precision-reconfigurable acceleration platform, especially suitable for training ML/DL
- Algorithmic-level: build analytical (maths) models of accuracy loss in DNN to avoid long simulations