Simulate-the-hardware: Training Accurate Binarized Neural Networks for Low-Precision Neural Accelerators

Jiajun Li, Ying Wang, Bosheng Liu, Yinhe Han, and Xiaowei Li

Cyber Computing Laboratory,
State Key Laboratory of Computer Architecture,
Institute of Computing Technology,
Chinese Academy of Sciences

ASP-DAC' 2019, Tokyo, Japan
January 2018

Outline

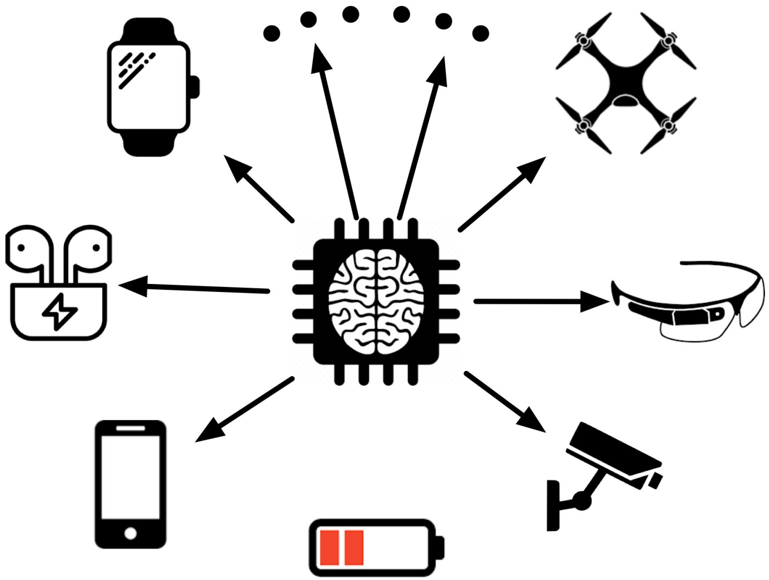
- Background and Motivations
  - Introduction: Binarized Neural Networks (BNNs)
  - Problems: overflow and deviation
- Our woks: Simulate-the-hardware
  - Overflow Containing
  - Overflow/Rounding Simulating
  - BNNs Training
- Experimental Results
- Conclusions

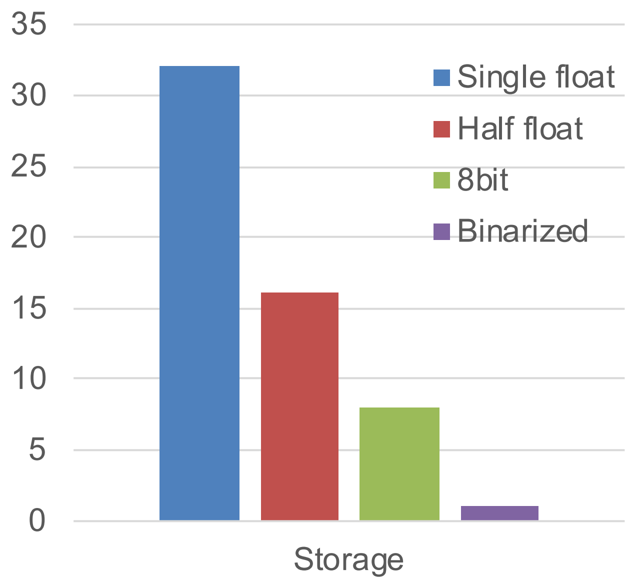
The part one

BACKGROUND & MOTIVATIONS

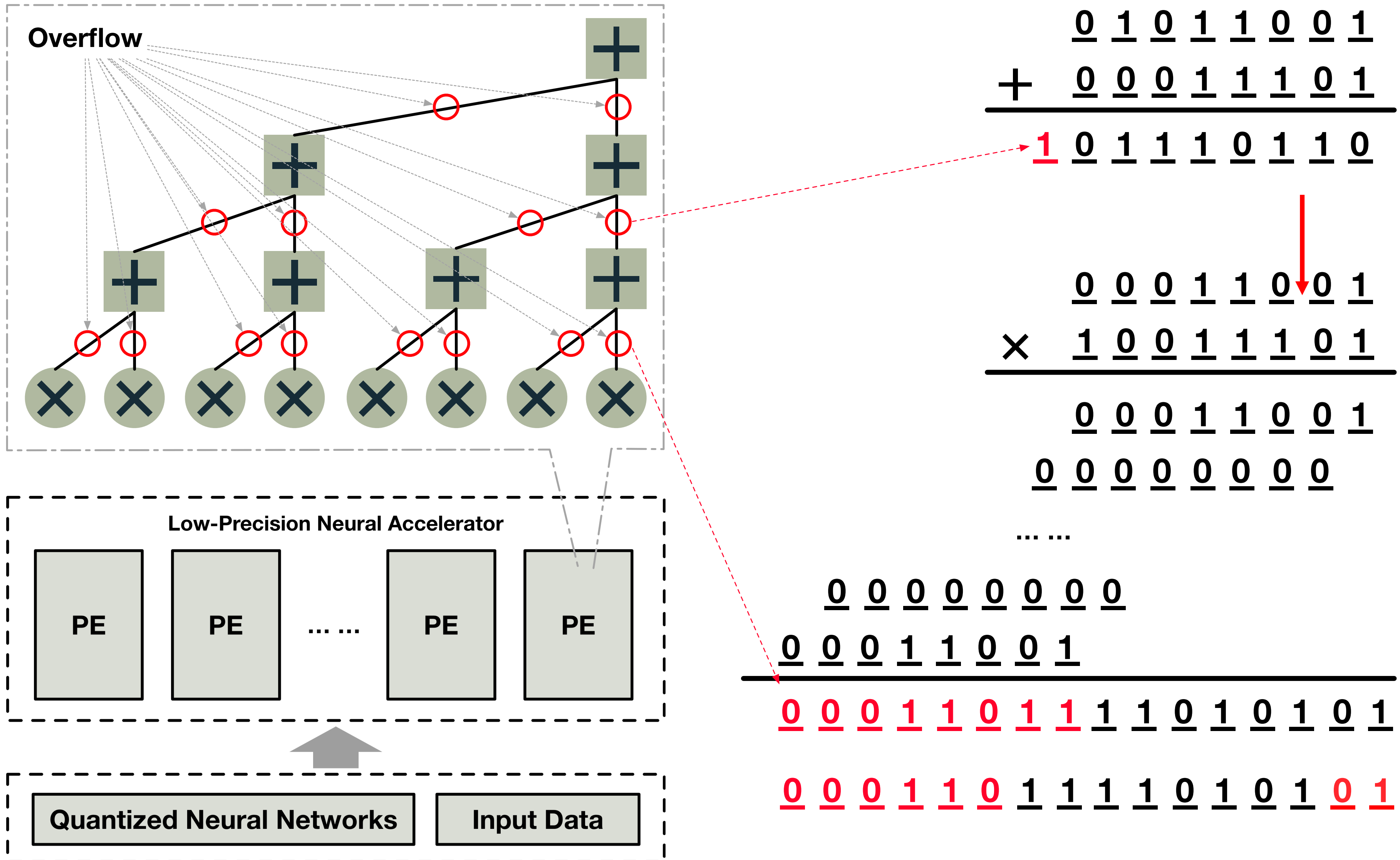
Background and Motivations

- Advantages of Binarized Neural Networks (BNNs)
  - Reduce storage requirement
  - Multiply-free computation
    - Increase calculating speed
    - Reduce chip area

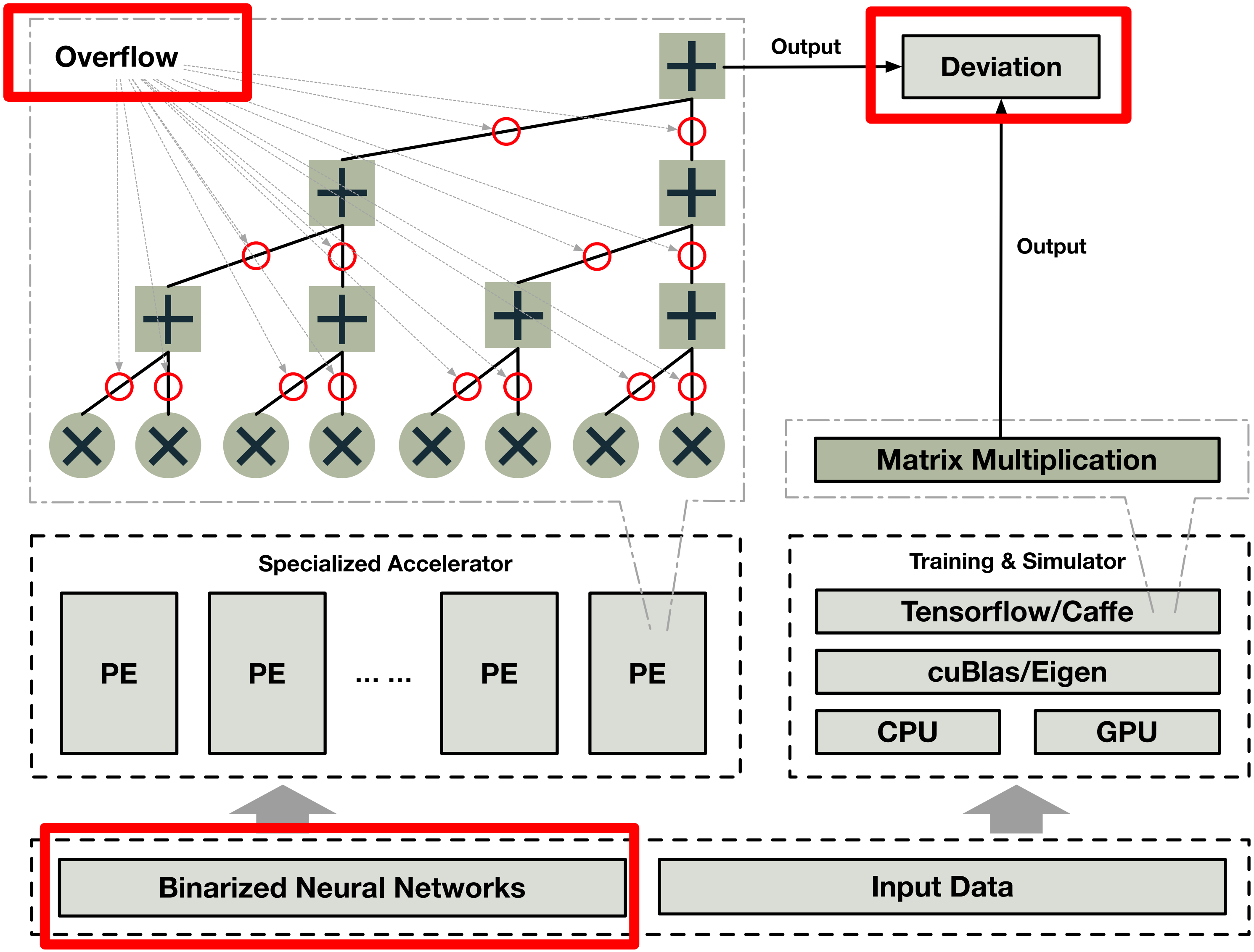




Problems: Overflow

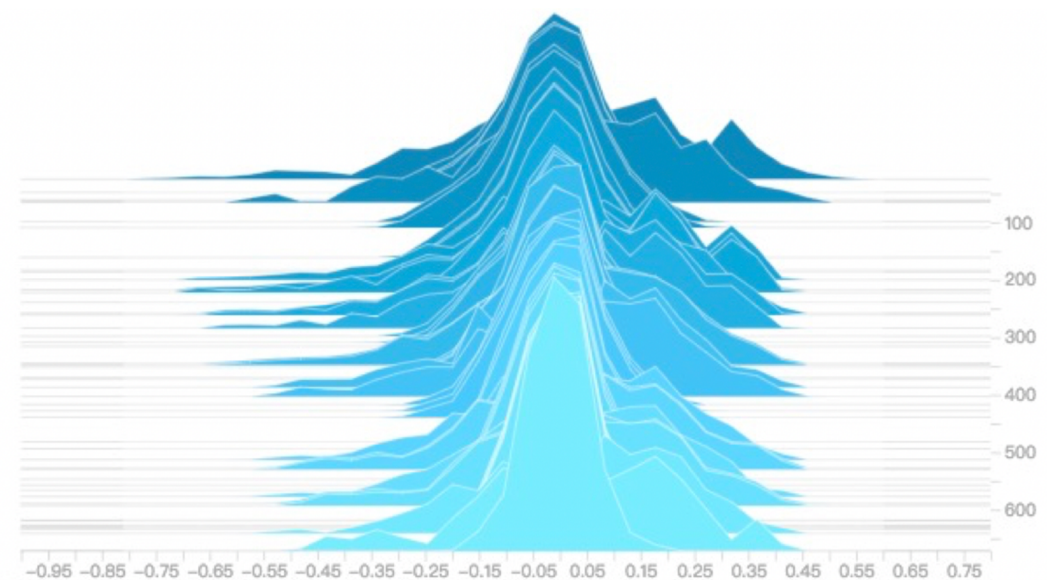


Problems: Deviation



Problems: Deviation

- Relation of overflow/rounding and deviation
  - Overflow
    - Limited-precision
    - Save energy and storage
    - Drop accuracy
  - Deviation
    - caused by overflow
    - must be eliminated
    - simulate the overflow



The part two

OUR WORKS

Our works

- Overflow containment
  - Hardware-friendly normalization layer that is at least 1.5X faster than Batch Normalization
  - Aggregated convolutional operation

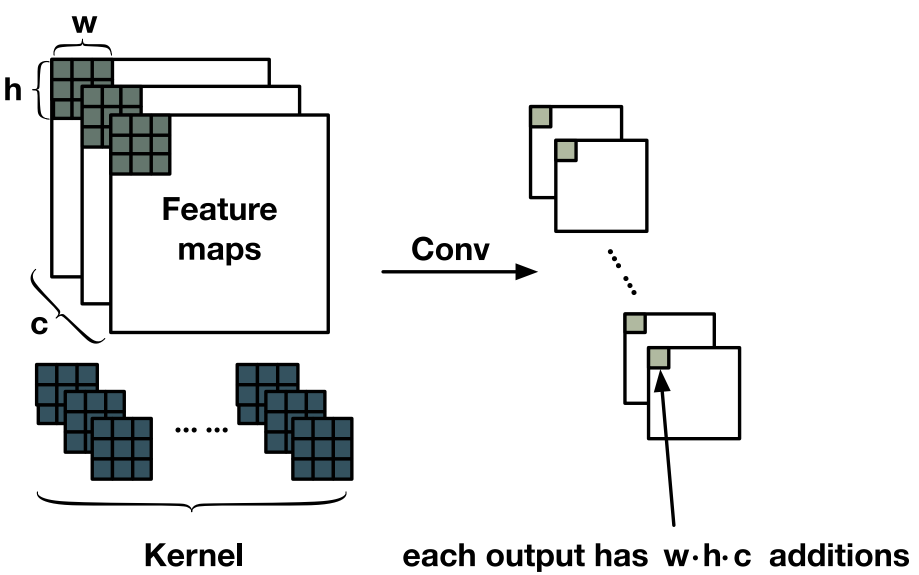
- • Overflow/rounding simulation
  - Simulate the overflow/rounding with GPU
    - 100X faster than the vanilla method
    - 80.8% slower than the original method
  - A new regularization term
    - The accuracy of our method is 12% higher compared with the past method.

Overflow Containing

- Normalization Layer Design
  - For containing overflow
    - Division
    - Support the Aggregated Convolutional Operation
  - For efficiency
    - The sigma equals 2 to the power of n

$$Norm(X) = \frac{X - \mu}{\sigma'},$$

$$\sigma' = 2^n \text{ where n is natural number}$$



Overflow Containing

- Aggregated Convolutional Operation

$$ReLU(x) = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases} \qquad HardTanh(x) = \begin{cases} 1, & x \geq 1 \\ x, & -1 \leq x < 1 \\ -1, & x < -1 \end{cases}$$

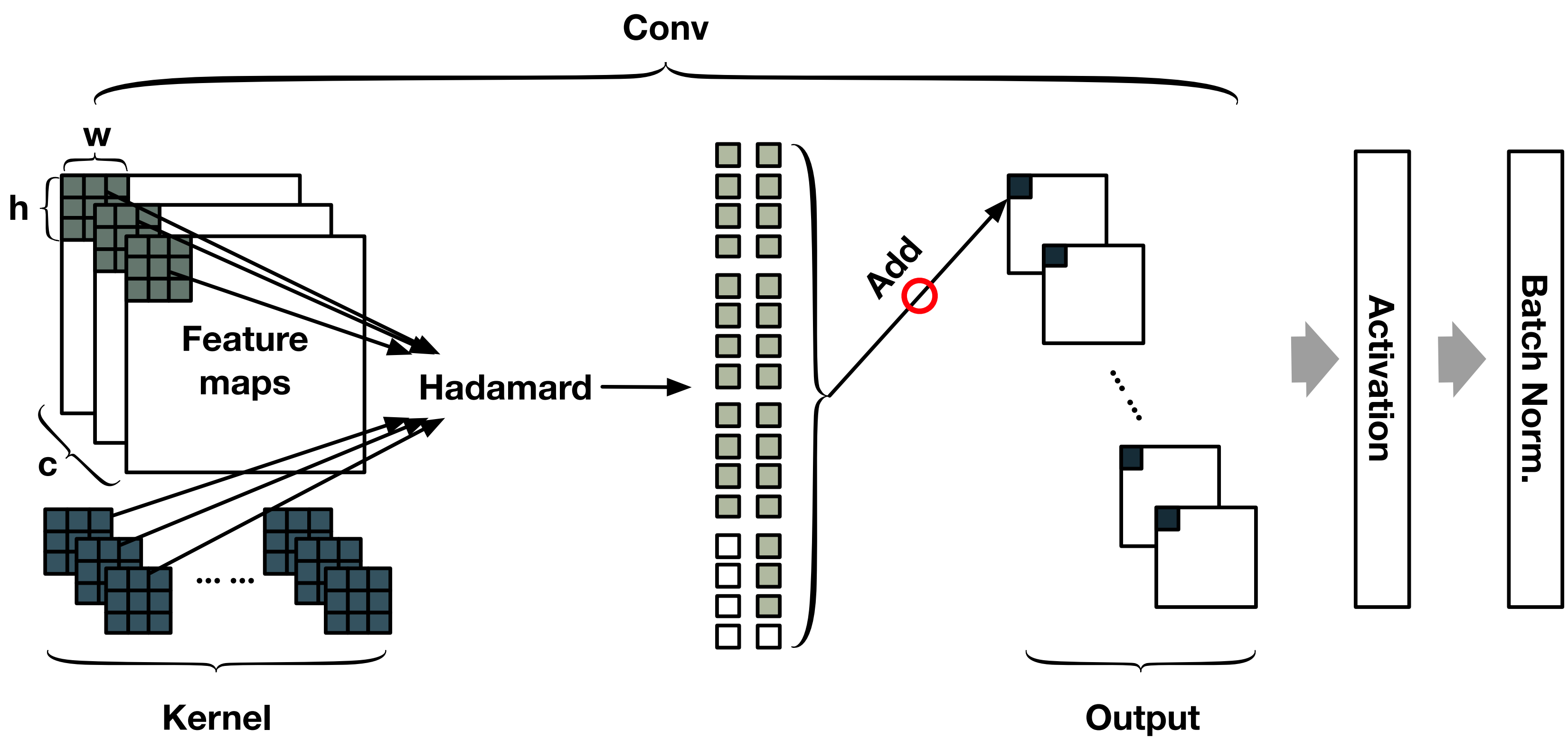
$$GP = \sum_{g}^{G} \frac{1}{\sigma'} \sum_{i} \sum_{j} W'_{ij} x'_{ij}$$

$$AC_{ReLU}(x) = \begin{cases} GP - \frac{\mu}{\sigma'}, & GP \geq 0 \\ 0, & otherwise \end{cases}$$

$$AC_{HT}(x) = \begin{cases} \frac{1}{\sigma'} - \frac{\mu}{\sigma'}, & GP \geq \frac{1}{\sigma'} \\ GP - \frac{\mu}{\sigma'}, & -\frac{1}{\sigma'} \leq GP < \frac{1}{\sigma'} \\ -\frac{1}{\sigma'} - \frac{\mu}{\sigma'}, & \frac{1}{\sigma'} < GP \end{cases}$$

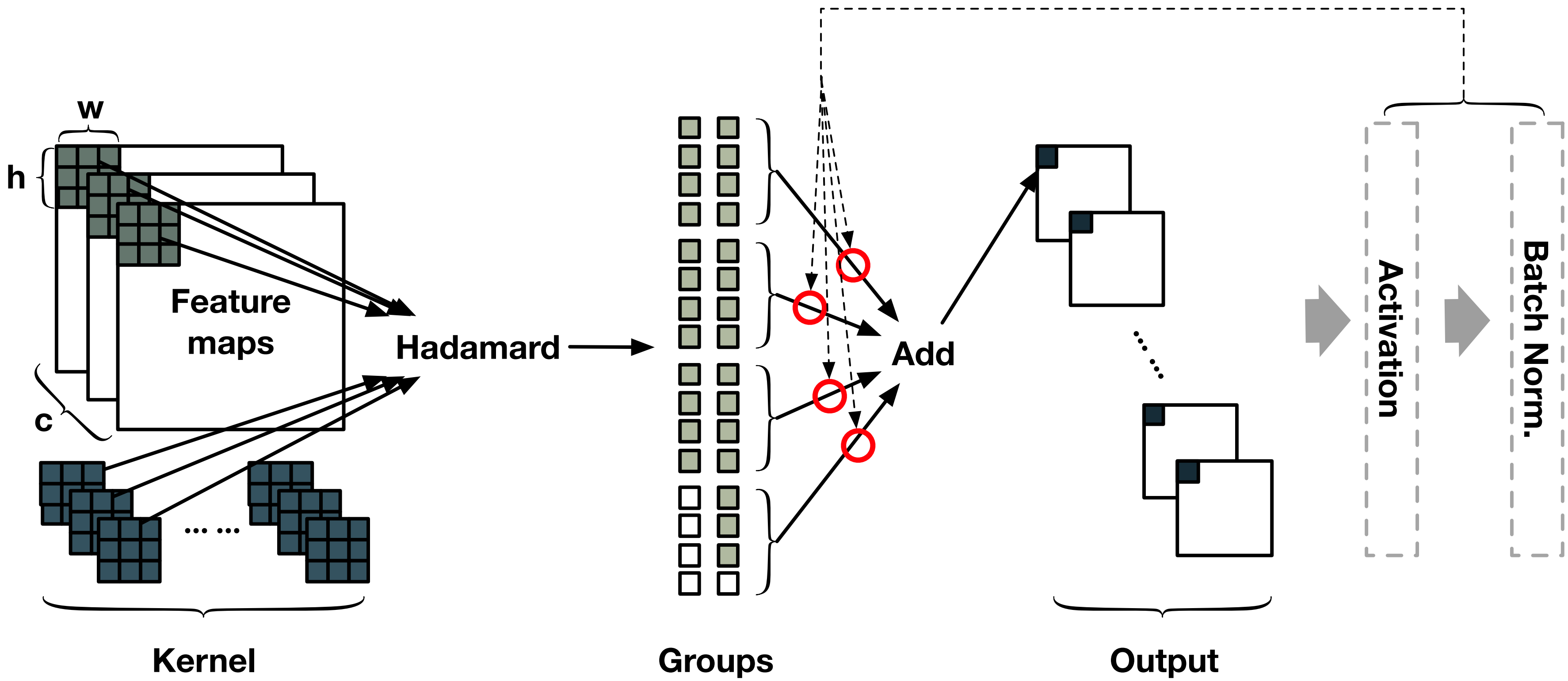
Overflow Containing

- Aggregated Convolutional Operation



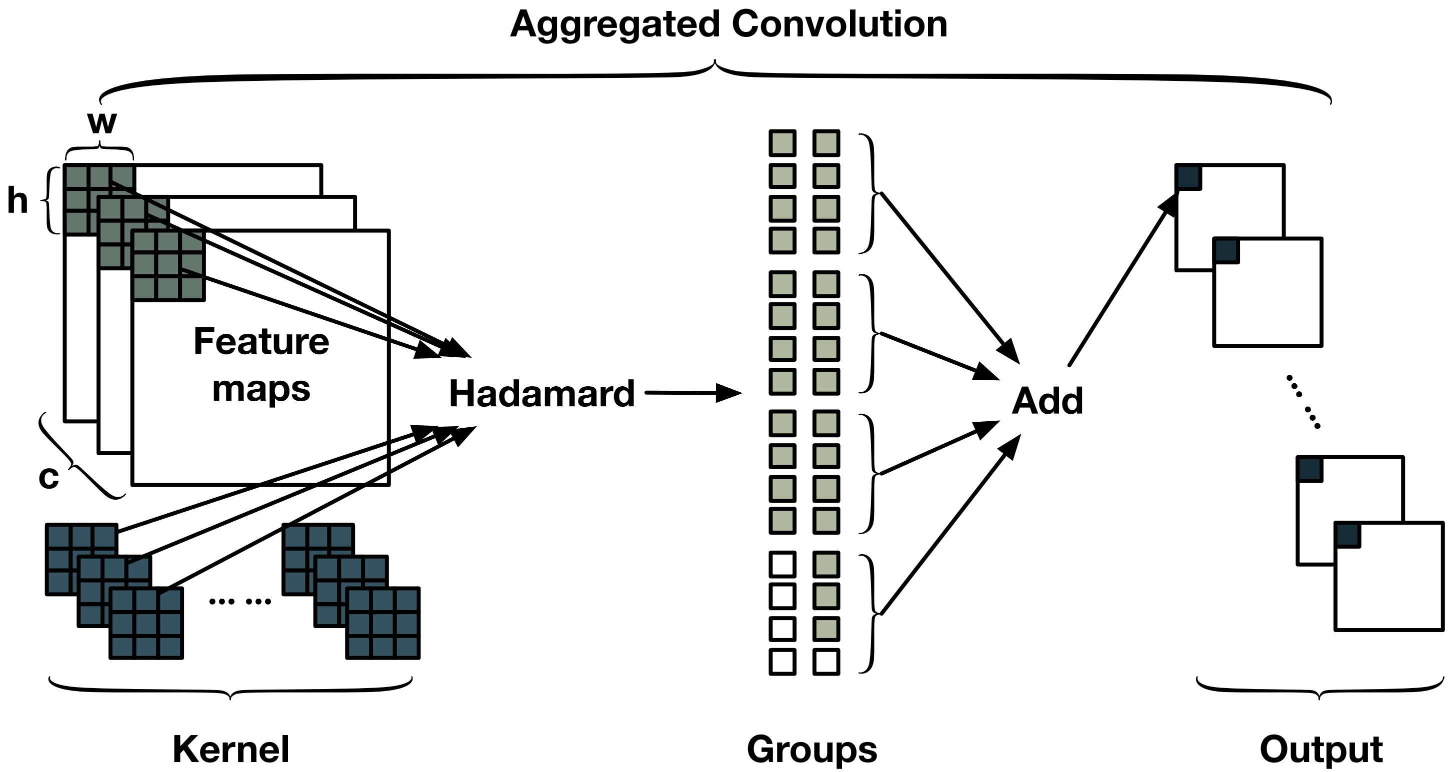
Overflow Containing

- Aggregated Convolutional Operation



Overflow Containing

- Aggregated Convolutional Operation

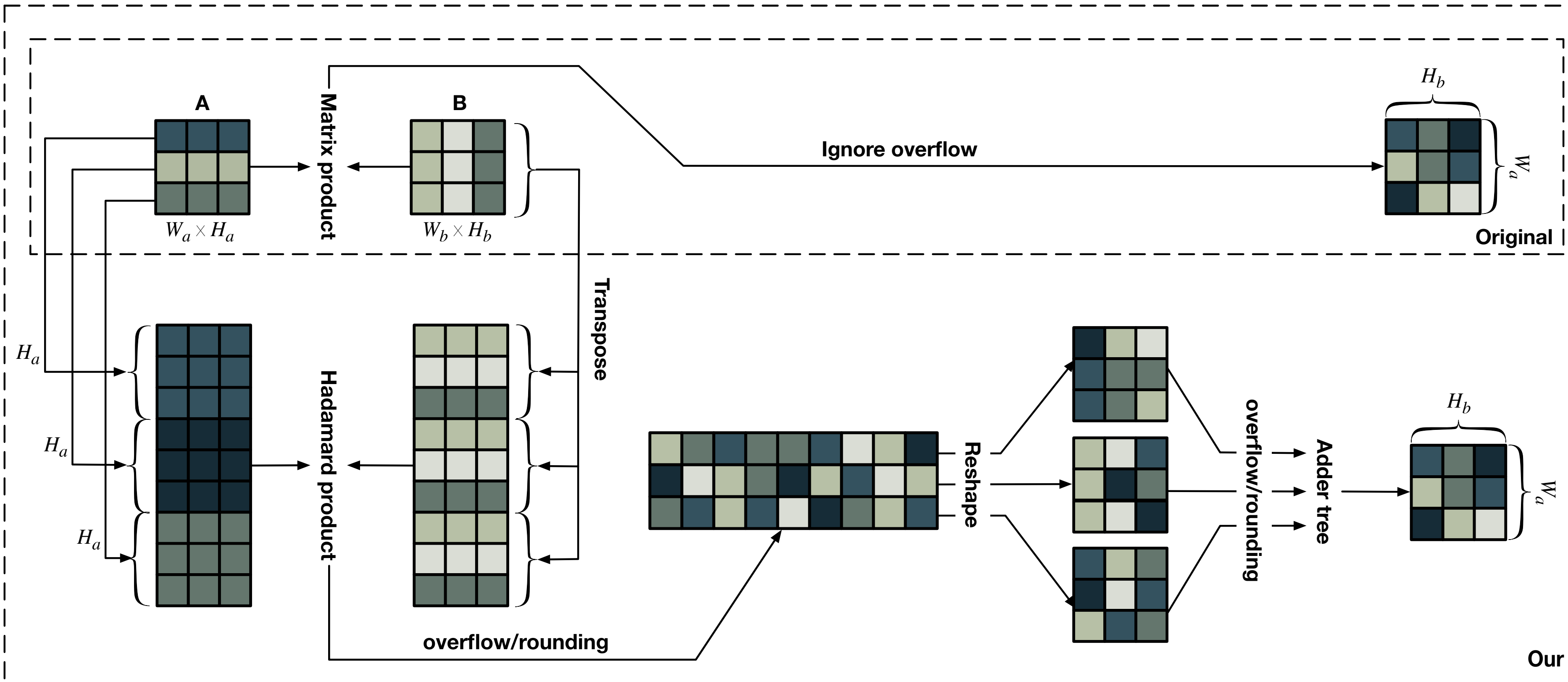


Overflow Containing

- Aggregated Convolutional Operation
  - Suppose that σ' = 4, μ = 0, G = 4 and the convolutional layer uses an 8-bit adder that the output range is [-128, 127].
  - Floating-point
    - Conv(X) = 512, ReLU(512) = 512, and Norm(512) = 128
  - Original Convolution with 8-bit adder
    - Conv(X) = 127, ReLU(127) = 127, and Norm(127) = 31
  - Aggregated Convolution with 8-bit adder
    - AConv(X) = 127/4+127/4+127/4+127/4 = 31+31+31+31 = 124

Overflow/Rounding Simulating

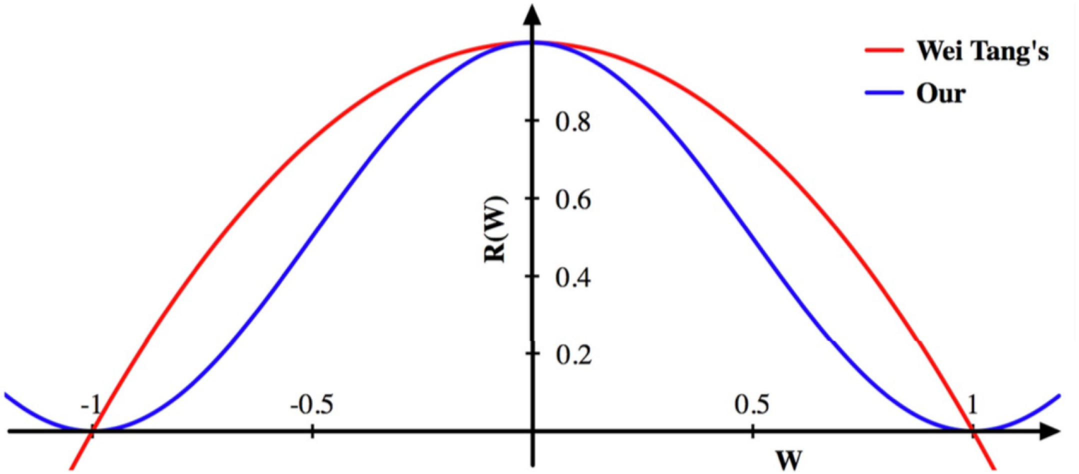
- Simulating Overflow/Rounding
  - Pytorch, Caffe, and Tensorflow
  - Illustrate with a figure



Regularization Term

$$J(W,b) = Loss(W,b) + \lambda \sum_{l=1}^{L} \sum_{i=1}^{N_l} \sum_{j=1}^{M_l} (1 - (W_{ij}^{(l)})^2)$$

$$J(W,b) = Loss(W,b) + \lambda \sum_{l=1}^{L} \sum_{i=1}^{N_l} \sum_{j=1}^{M_l} (cos(\pi W_{ij}^{(l)}) + 1)$$



The part three

EXPERIMENTAL RESULTS

Experimental Results

- Normalization Layer
  - Drop a little of accuracy
  - Improves tremendous computational speed
  - Hardware-friendly
  - Simpler architecture design

Table 1: Quantized Batch Normalization vs. Batch Normalization.

	Batch Norm.	Our
Accuracy	82.32%	72.30%
FPS	4060.45	6097.35

Experimental Results

- Regularization Term
  - AlexNet of BNN
  - Use the mentioned normalization layer

Table 2: The comparison of two regularization terms.

	Wei Tang et al.	Our
Accuracy	64.55%	72.30%

Experimental Results

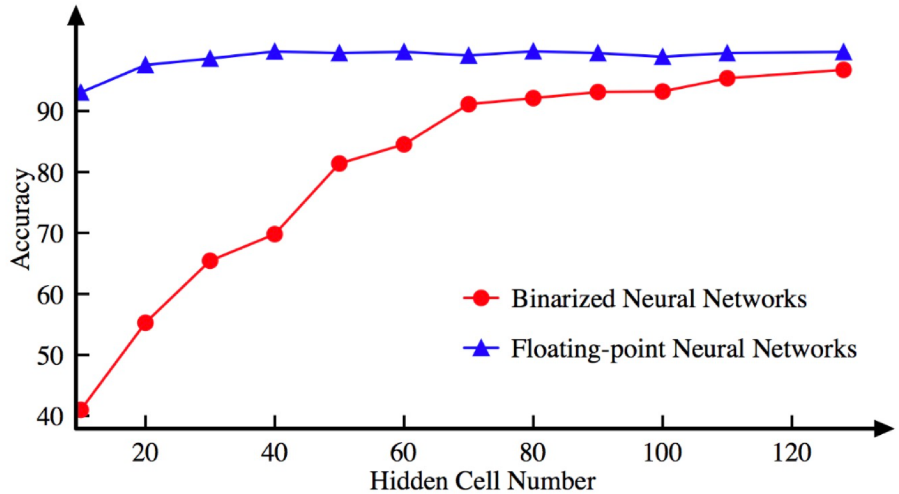
- Simulating Overflow/Rounding
  - Batch size is 128
  - Our method is 80.75% slower than the original method
  - The original almost cannot simulate overflow/rounding
  - Our method is about 100X faster than the vanilla method

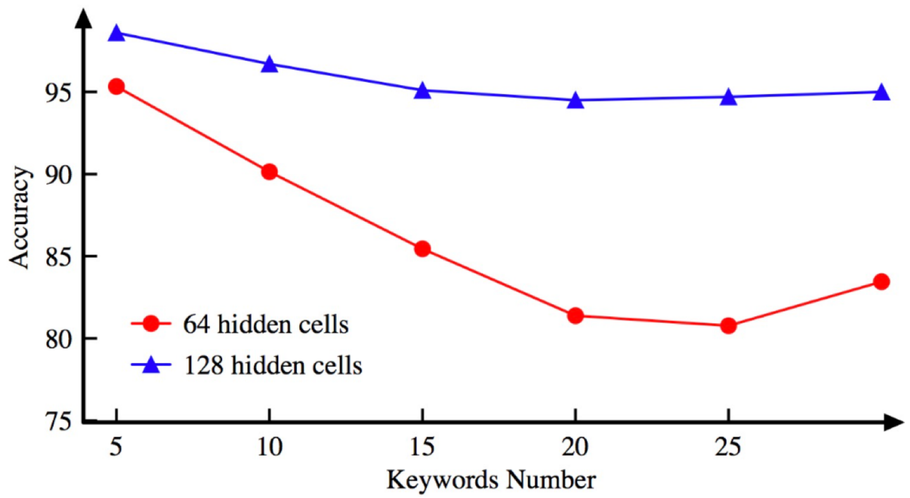
Table 4: The time-consuming of matrix product.

	Original	Vanilla	Our
Time consuming (s/batch)	1.6	840.5	8.3

Experimental Results

- Overall Evaluation
  - Keywords Spotting with BNN-LSTM
  - Dataset from Google AI
  - The affect of hidden cell number
  - The affect of keyword number





Summary

- Our Work
  - A series of the methods to contain the overflow, simulate the overflow/rounding, and train accurate BNNs for low-precision neural accelerators.
- Authors' Hope
  - The work can inspire the intelligent specialized accelerators to achieve better performance.
- My Vision
  - Let edge devices be smarter
- Acknowledgment
  - Thanks for your coming and the anonymous referees for their valuable comments and helpful suggestions.

Thank you

Q&A