Toward Low-Bit Neural Network Training Accelerator by Dynamic Group Accumulation

Speaker: **Yixiong Yang**, Dept. of Electronic Engineering, Tsinghua University
Authors: Yixiong Yang, Ruoyang Liu, Jinshan Yue, Wenyu Sun,
Huazhong Yang, and Yongpan Liu
Contact: ypliu@tsinghua.edu.cn

Outline

- **Background of the CNN Training**
- Challenge: Low-Bit Accumulation
- Dynamic Group Accumulation (DGA) Algorithm
- Efficient DGA Hardware Design
- Conclusion

Demand of the Training Accelerator

VGG
University of Oxford

AlphaGo

GPT3

Growing Need of Large Neural Network Model

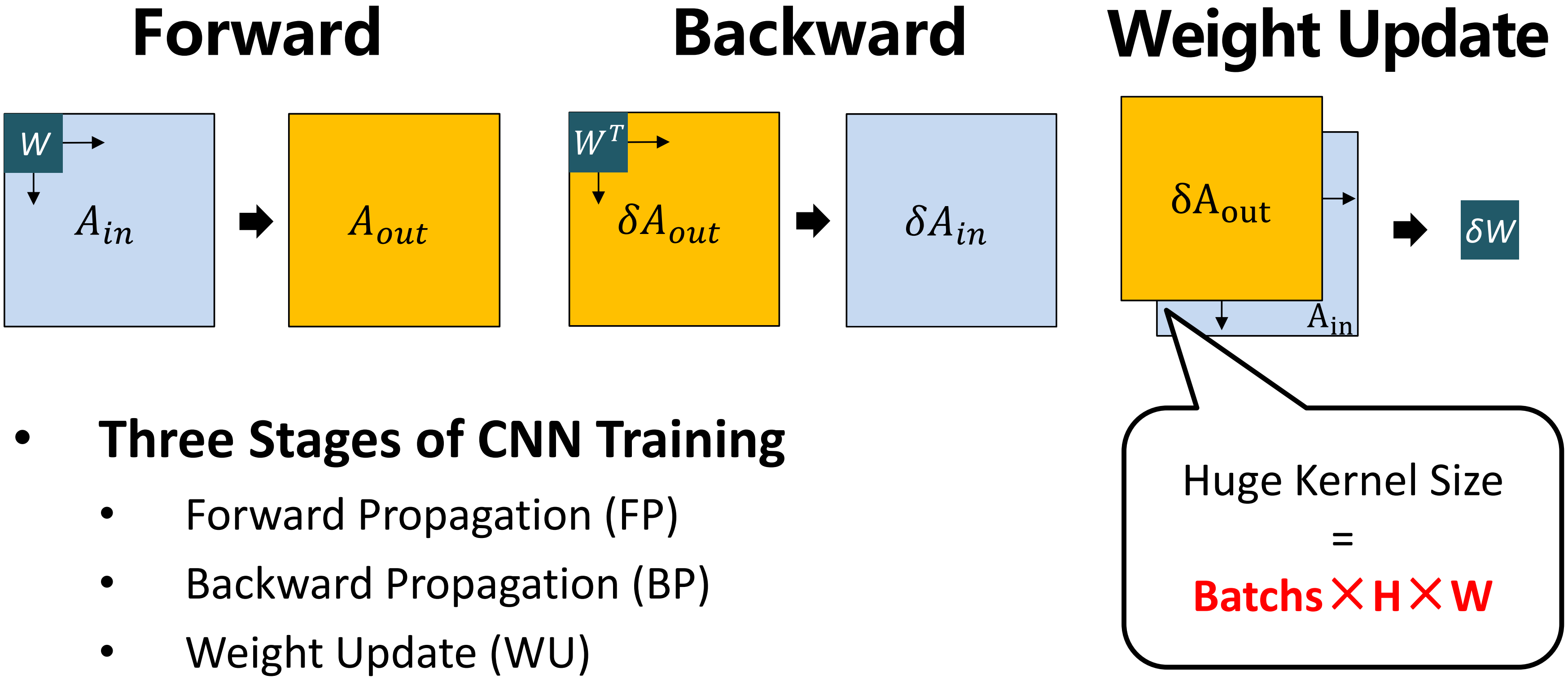
Higher Energy Efficiency of Training Accelerators

A-100, Nvidia

TPU, Google

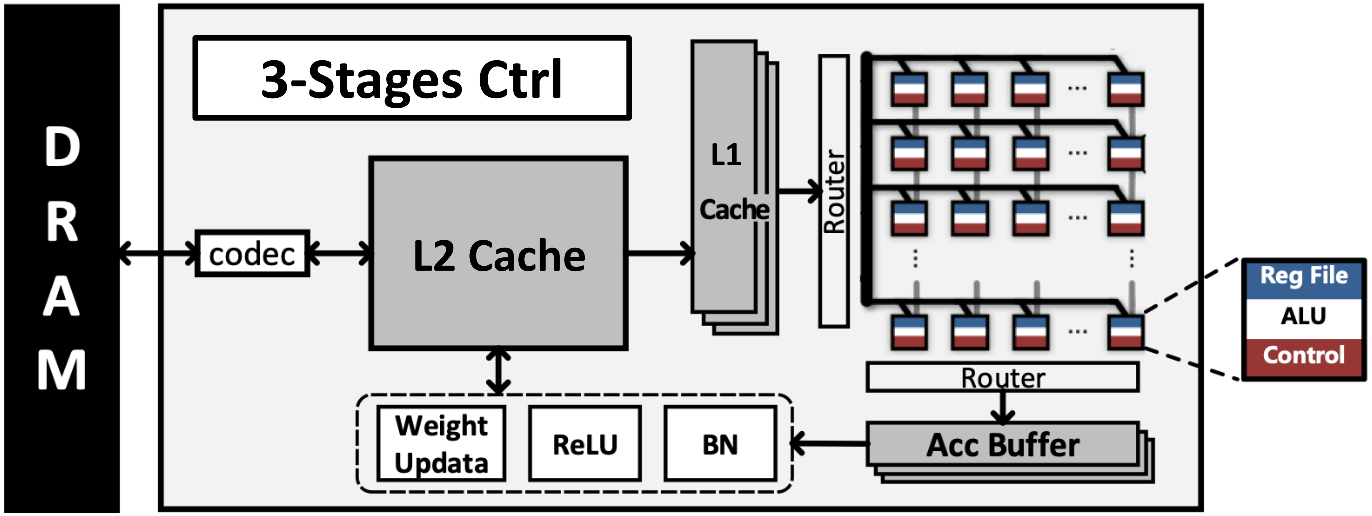
Ascend 910, Huawei

CNN Training Algorithm



- **Three Stages of CNN Training**
  - Forward Propagation (FP)
  - Backward Propagation (BP)
  - Weight Update (WU)
- **Training needs Higher Precision**
  - Uncertain data range in different training epochs
  - 10k-1M accumulations in the WU stage

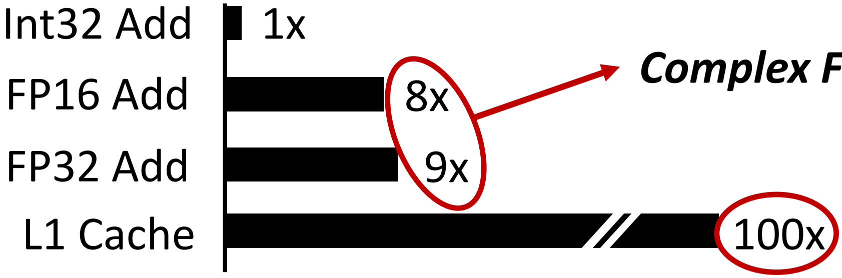
Architecture of Training Accelerator



High-Bit Float-Point Processing Unit

45nm CMOS Norm Energy (per bit)

M. Horowitz, ISSCC14 [5]



Complex Float-Point Adder

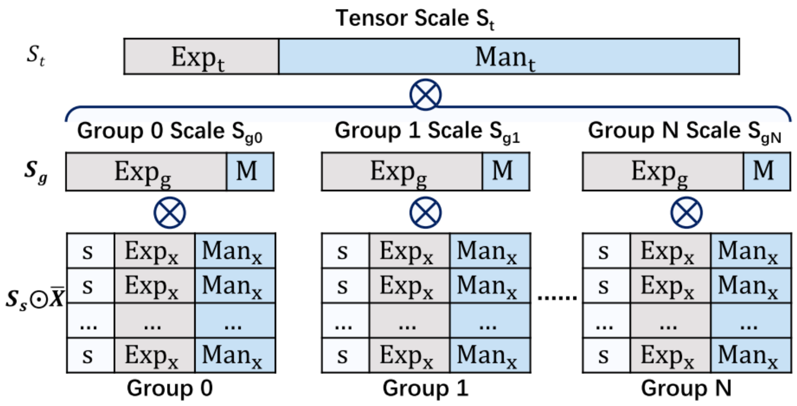
High-bit Accumulation Buffer Consume More Energy

Recent Works on Low-Bit Training

- **Methods to Achieve Low-bit Training**
  - Fine-grained Quantization
  - Clip the long-tailed data

Multi-Level Scaling

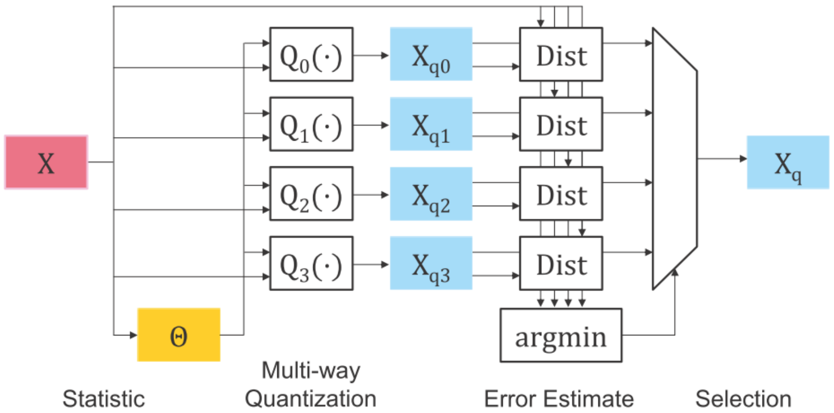
Zhong, et al. ArXiv:2006.02804



FP7(1-2-4) Train ImageNet with No Loss

Clipping Range Select

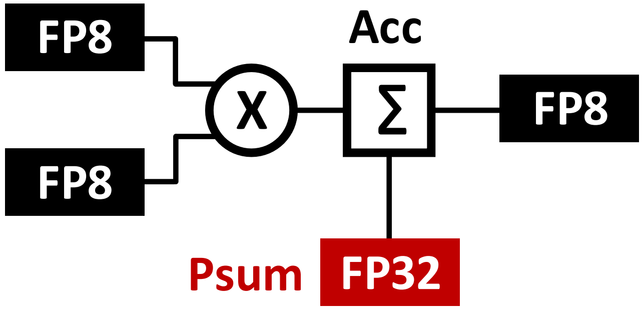
Cambrion-Q, ISCA 2021

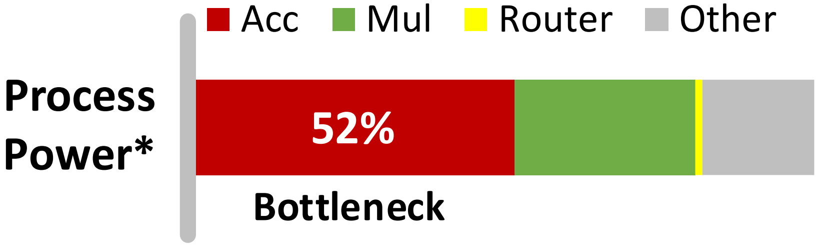


BFP8 Train ImageNet with No Loss

Recent Works on Low-Bit Training

- **Methods to Achieve Low-bit Training**
  - Fine-grained Quantization
  - Clip the long-tailed data
- **A Typical Quantization Setting** (J. Park, ISSCC2021)
  - Multiply: FP8 (1-4-3)
  - Accumulate: FP30 (1-6-23), similar to FP32 (1-8-23)





* Power consumption of the CNN accelerator [11].

The Bottleneck: High-Bit Accumulation

Outline

- Background of Low-Bit CNN Training
- **Challenge: Low-Bit Accumulation**
- Dynamic Group Accumulation (DGA) Algorithm
- Efficient DGA Hardware Design
- Conclusion

Challenge: Low-bit Accumulation

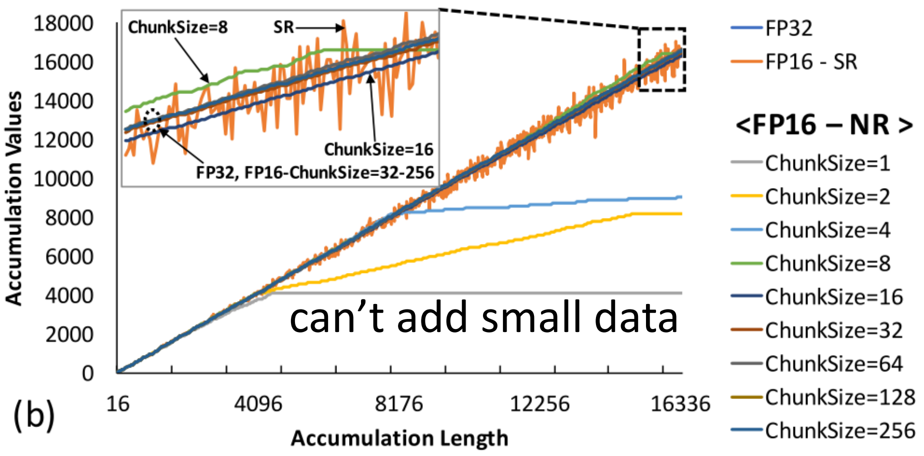
- **Float-Point Swamping Error**
  - Error increased with accumulation
- **Safe Accumulation Precision**
  - Commonly need FP32 in training [13]

Example of FP16 Accumulation

$$1024 + \underbrace{1 + 1 + \ldots + 1 + 1}_{\text{1024 data}} = 1024$$

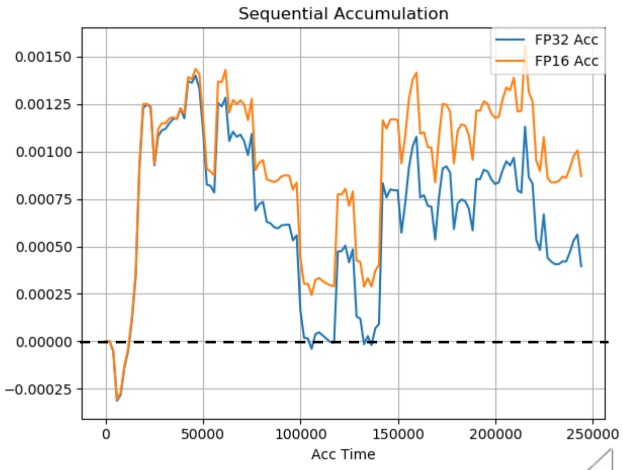
1-value is round-off in every addition

Numerical Simulation of FP16 Addtion [12]



can't add small data

FP16 Naïve Accumulation



Error Ratio: 130%

Challenge: Low-bit Accumulation

- **Solution：Group Accumulation**
  - Reduce error from 130% to 17%
- **Group Size Selection**
  - Static size Ng=16[7], Ng=24[13]
  - Can’t fit the changing data

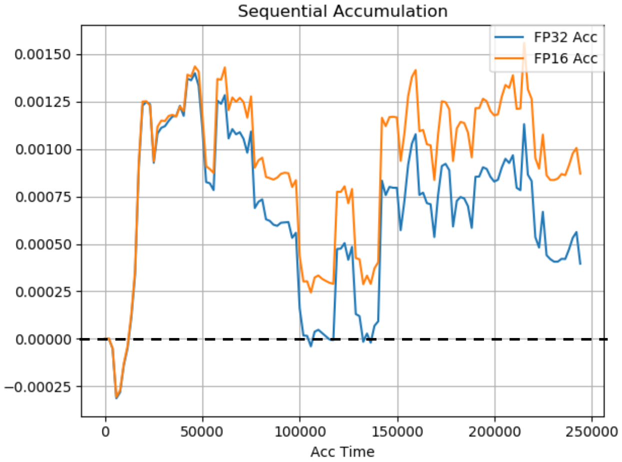
Example of Group Accumulation

$$1024 + \underbrace{(1 + \dots + 1)}_{\text{32 data}} + \underbrace{(4096 + \dots + 1)}_{\text{32 data}}$$

Should group *more data*

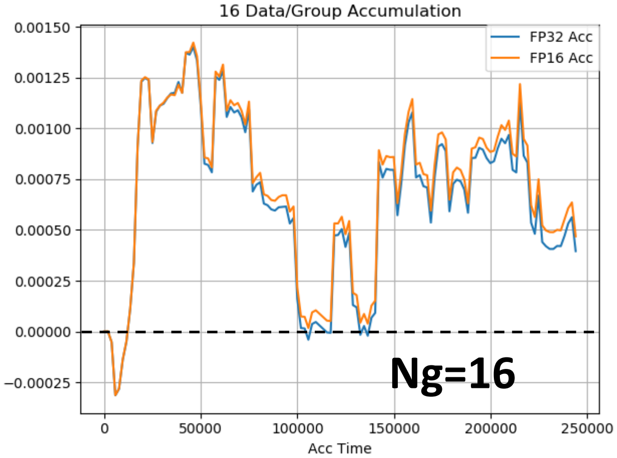
Should group *less data*

FP16 Naïve Accumulation



Error Ratio: 130%

FP16 Group Accumulation



Error Ratio：17%

Solution of This Work

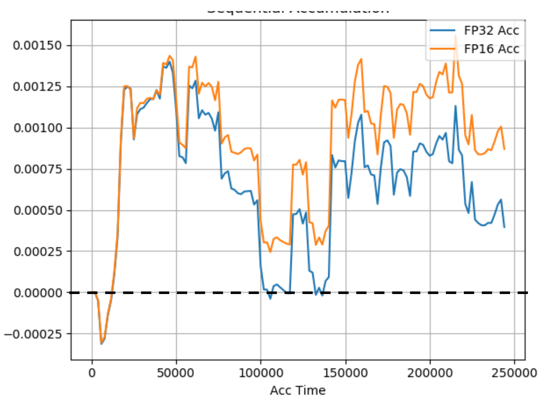
- **Dynamic Group Accumulation**
  - Reduce error from 17% to **0.25%**
- **Contributions of This Work**
  - The optimal DGA algorithm
  - The efficient DGA HW design

Dynamic Group Accumulation

$$1024 + \underbrace{(1 + \ldots + 1)}_{\text{1024 data}} + \underbrace{4096}_{\text{1 data}} + \underbrace{(8 + \ldots + 8)}_{\text{512 data}}$$

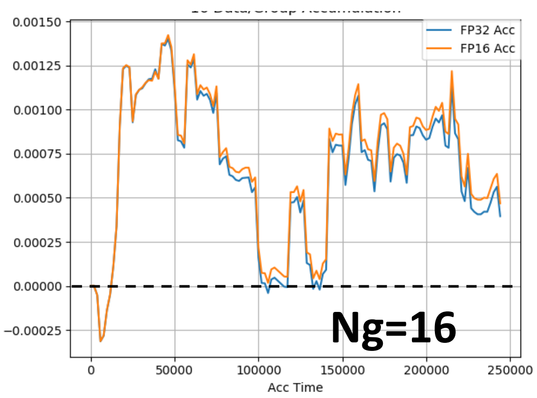
Adjust group size with changing data

FP16 Naïve Acc



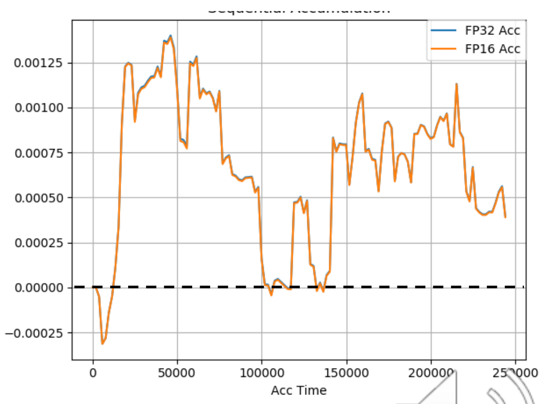
Error Ratio: 130%

FP16 Static Group Acc



Error Ratio： 17%

FP16 **Dynamic Group Acc**



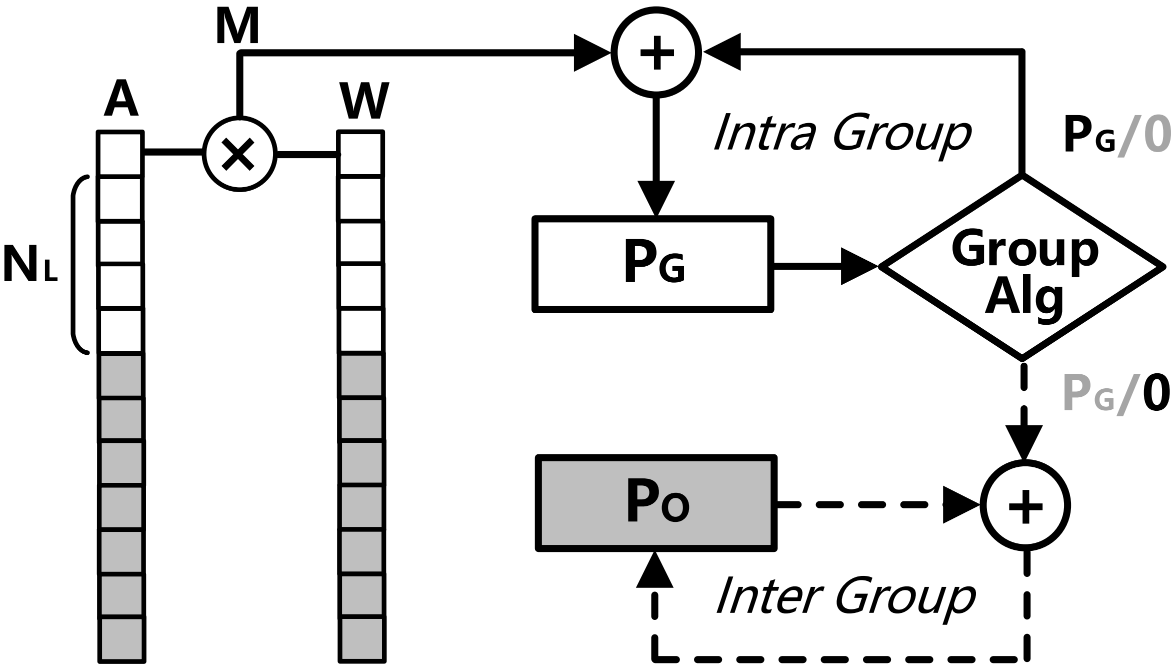
Error Ratio： 0.25%

Outline

- Background of Low-Bit CNN Training
- Challenge: Low-Bit Accumulation
- **Dynamic Group Accumulation (DGA) Algorithm**
- Efficient DGA Hardware Design
- Conclusion

Model Formulation of DGA

- **Two Stages of Accumulation**
  - Intra-Group Acc: M firstly accumulates in $\mathrm{P_G}$
  - Inter-Group Acc: Add $\mathrm{P_G}$ to $\mathrm{P_o}$ and clear $\mathrm{P_G}$
  - Group Algorithm: Decide intra- or inter-group accumulation



Basic Assumption & Optimal Result

- **Assumption 1： Independent & Uniform Error Distribution**

$$\overline{E^2_{acc}} = \frac{1}{N_g}(\overline{E^2_O} + \sum_{n=0}^{N_g-1} \overline{E^2_G(n)}) \qquad \overline{E^2_O} = \frac{(P_O * 2^{-m})^2}{12}, \quad \overline{E^2_G(n)} = \frac{(P_G * 2^{-m})^2}{12}$$

- **Assumption 2： Psum Result Changing Piecewise Linearly**
  - $P_G(n)$~n

- **Target Function & Optimial Condition**

$$\arg\min\ \overline{E^2_{acc}}(P_O,\ P_G,\ N_g) \rightarrow \boxed{\frac{2N_g}{3} P_G^2(N_g) = P_O^2}$$

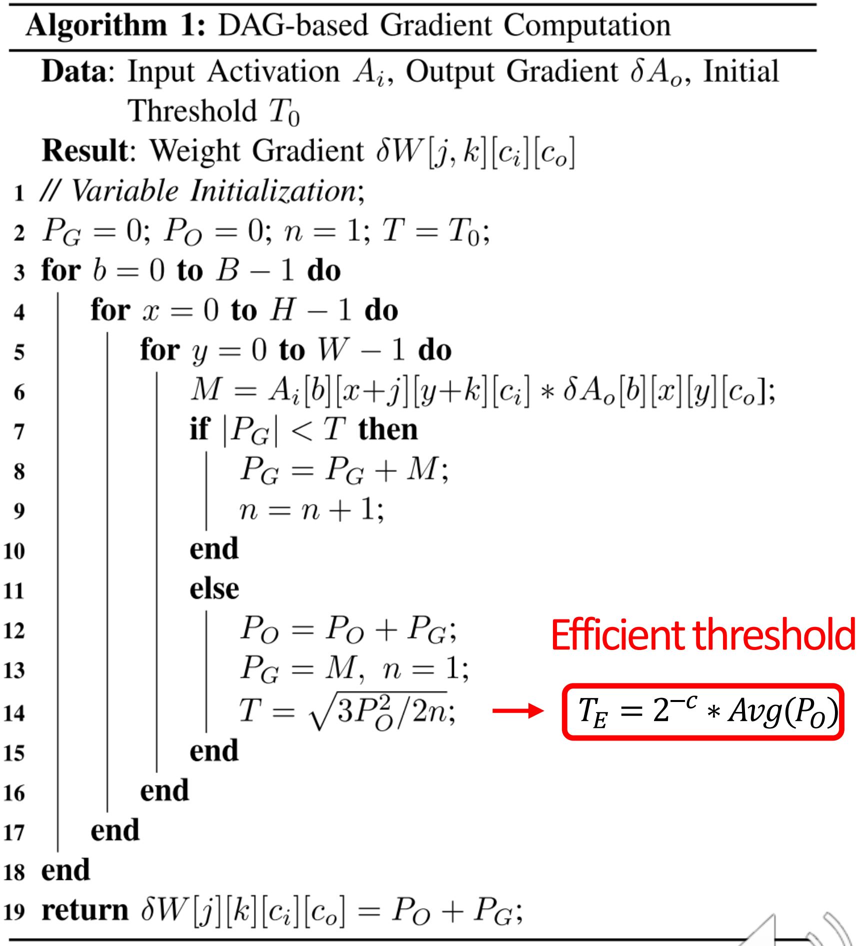
Efficient DGA Algorithm

- **Ideal DGA Algorithm**
  - Group threshold: $|P_G| > \sqrt{3{P_O}^2/2n}$

Problem 1: Threshold Calculation

Problem 2: Threshold Saving

- **Efficient DGA Algorithm**
  - Simplify threshold calculation
    - $|P_G| > T_E = 2^{-c} * P_O$
  - Saving threshold register usage
    - Sharing among weight kernel



Algorithm 1: DAG-based Gradient Computation

```
Data: Input Activation $A_i$, Output Gradient $\delta A_o$, Initial
      Threshold $T_0$
Result: Weight Gradient $\delta W[j,k][c_i][c_o]$
// Variable Initialization;
$P_G = 0$; $P_O = 0$; $n = 1$; $T = T_0$;
for $b = 0$ to $B - 1$ do
    for $x = 0$ to $H - 1$ do
        for $y = 0$ to $W - 1$ do
            $M = A_i[b][x+j][y+k][c_i] * \delta A_o[b][x][y][c_o]$;
            if $|P_G| < T$ then
                $P_G = P_G + M$;
                $n = n + 1$;
            end
            else
                $P_O = P_O + P_G$;
                $P_G = M,\ n = 1$;
                $T = \sqrt{3P_O^2/2n}$;
            end
        end
    end
end
return $\delta W[j][k][c_i][c_o] = P_O + P_G$;
```

Efficient threshold

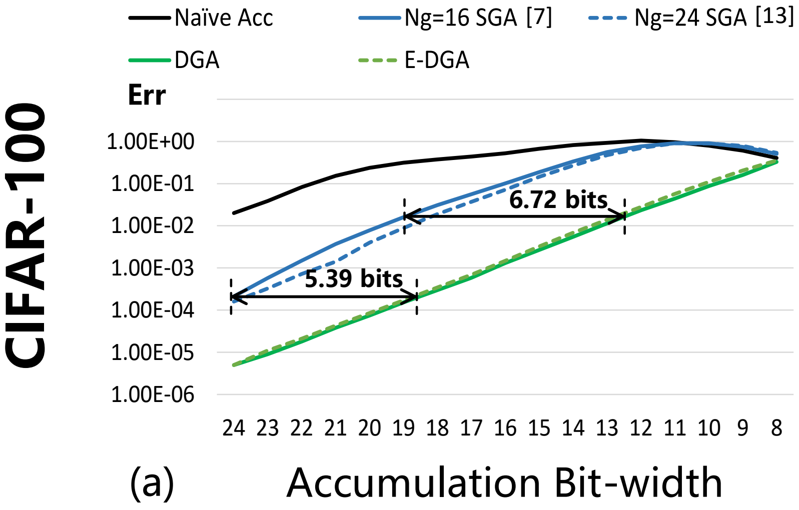
$\boxed{T_E = 2^{-c} * Avg(P_O)}$

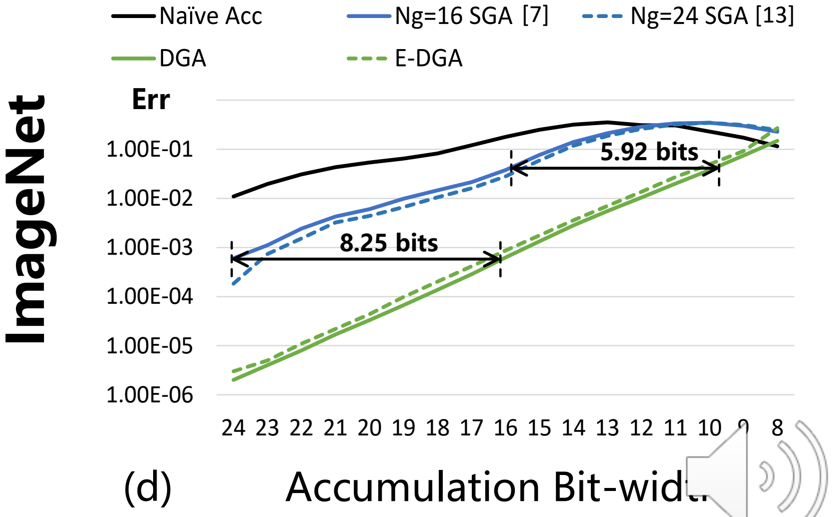
DGA Algorithm Performance

- **Experiment Setting**
  - CIFAR-100 dataset, ResNet-38 model
  - ImageNet dataset, ResNet-18 model
  - Training Batch Size = 256
- **Error Definition**

$$tan(\delta\theta) = \sqrt{\left(\frac{||\delta W_c||_2||\delta W_t||_2}{< \delta W_c,\ \delta W_t >}\right)^2 - 1}$$

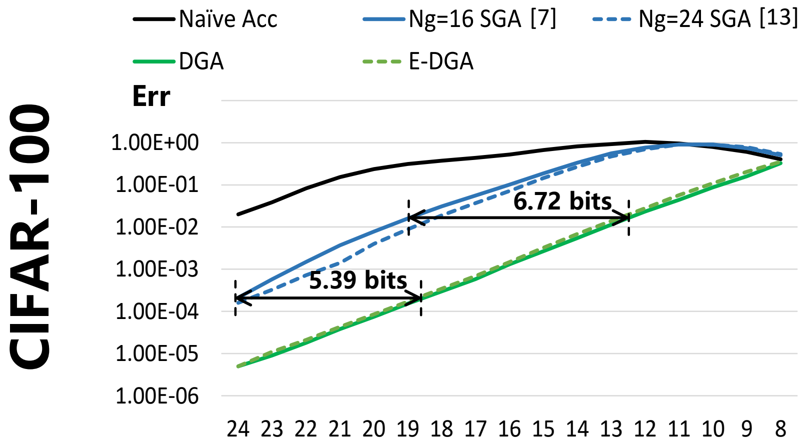
- **Calculation Error with Different Bit-width**
  - Ideal-DGA can **save 6.31/6.91 bits** compared to the SGA [7]
  - E-DGA only **loss 0.21/0.45 bits** than Ideal-DGA



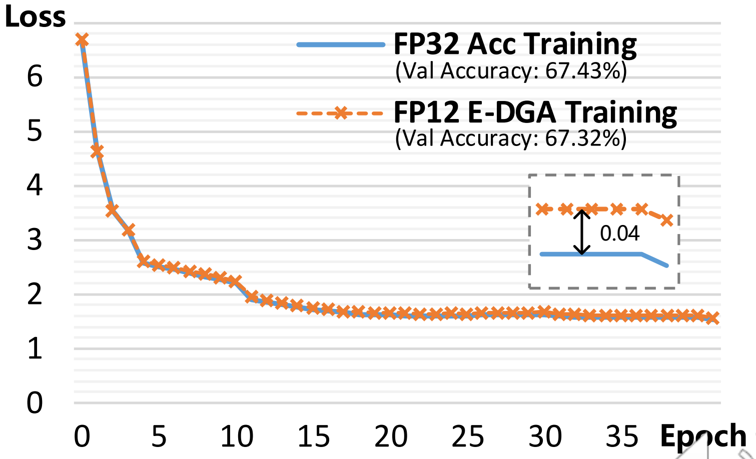


DGA Algorithm Performance

- **Experiment Setting**
  - CIFAR-100 dataset, ResNet-38 model
  - ImageNet dataset, ResNet-18 model
  - Training Batch Size = 256
- **Calculation Error with Different Bit-width**
  - Ideal-DGA can **save 6.31/6.91 bits** compared to the SGA []
  - E-DGA only **loss 0.21/0.45 bits** than Ideal-DGA



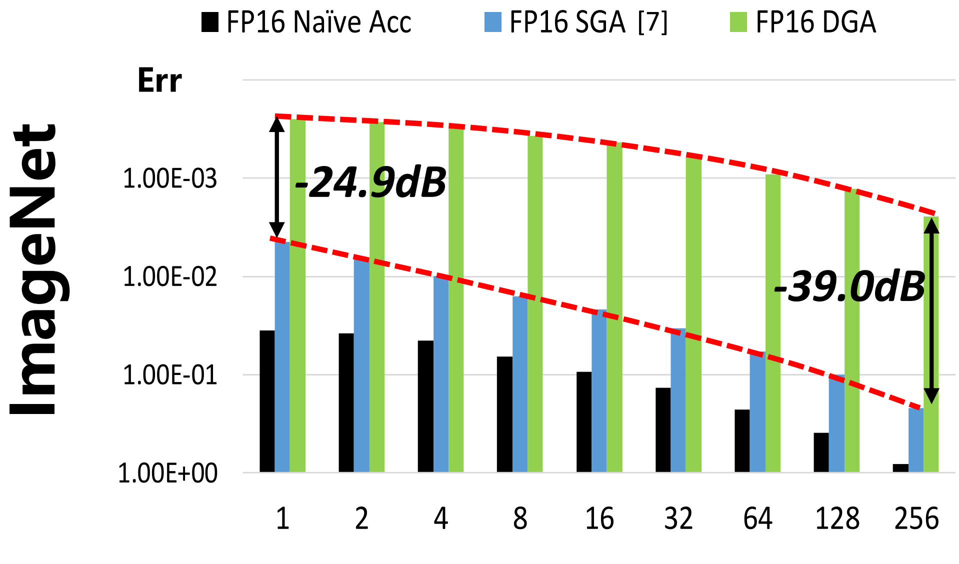
(a) Accumulation Bit-width



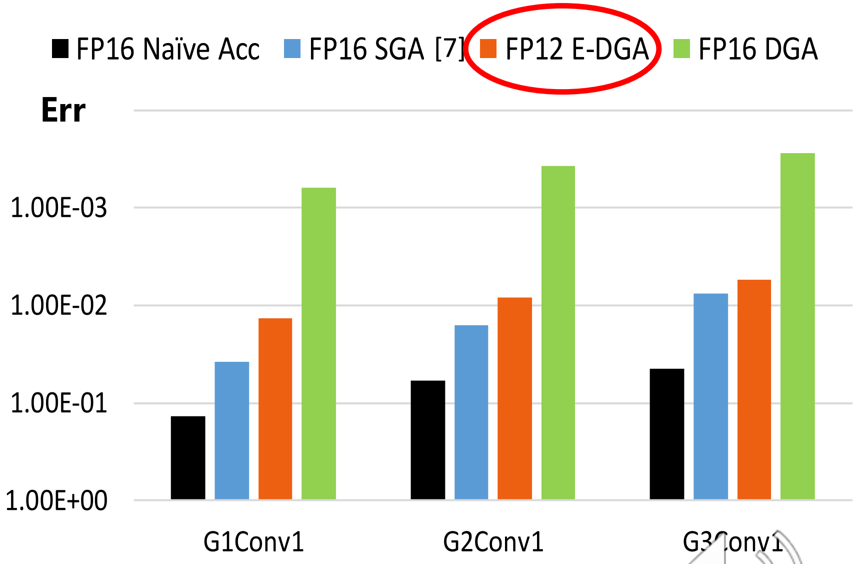
Loss Decent Curve

DGA Algorithm Performance

- **Discussion on Different Batch Size**
  - Batch Size = 1: DGA error is 17.6x smaller (-24.9dB) than SGA error
  - Batch Size = 256: DGA error is **89.1x** smaller (**-39.0dB**) than SGA error
- **Discussion on Different Layer**
  - **FP12** (1-5-6) E-DGA outperforms FP16 (1-6-9) SGA



(e) Batch Size



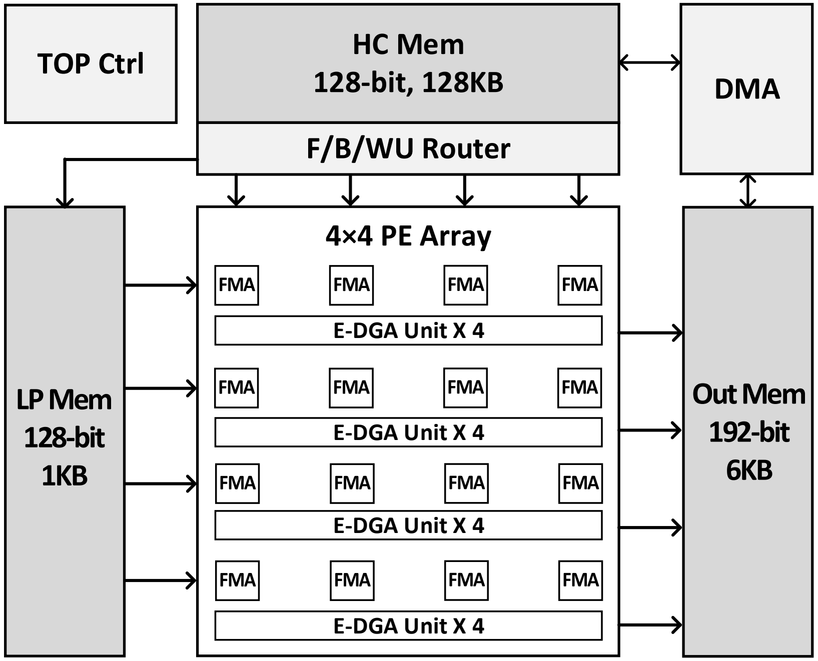
(f) Network Layers

Outline

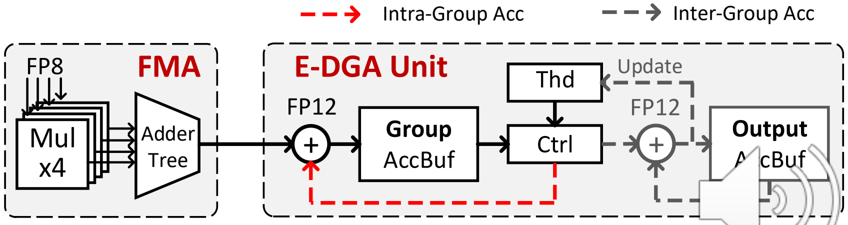
- Background of Low-Bit CNN Training
- Challenge: Low-Bit Accumulation
- Dynamic Group Accumulation (DGA) Algorithm
- **Efficient DGA Hardware Design**
- Conclusion

DGA-Based Training Architecture

- **Overall Architecture**
  - Tensor-Core like PE array + Memory hierarchy
  - F/B/WU Router: Route and transpose data for three stages
- **E-DGA Unit Design**
  - 8.8% area overhead for E-DGA function

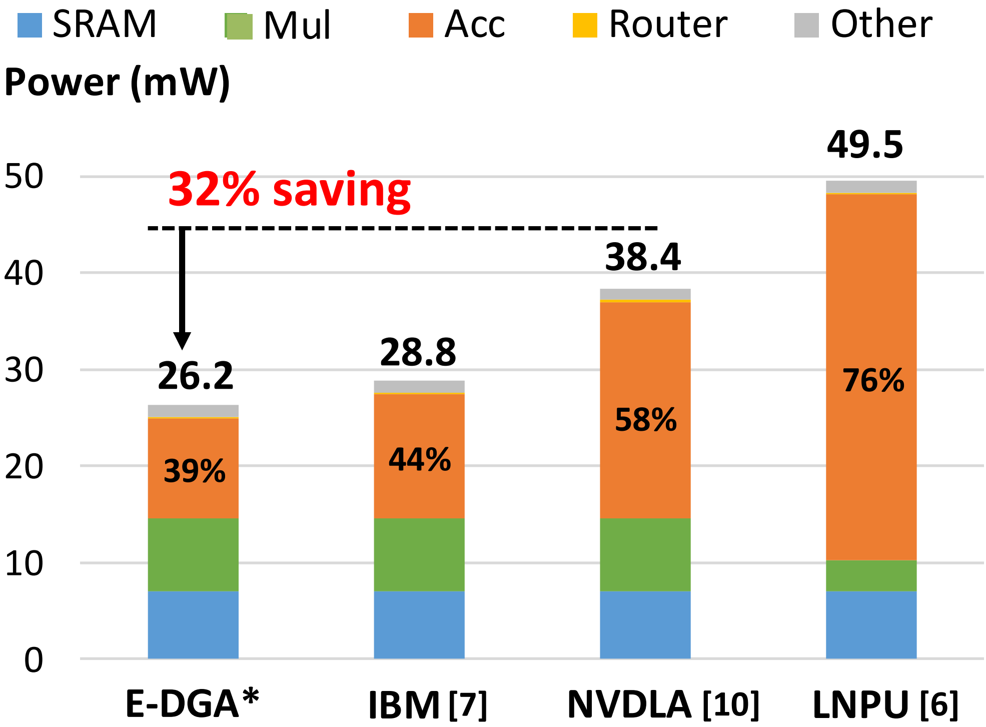


	Column Input	Row Input	PE Output
Forward	A_i	W	A_o
Backward	δA_o	W^T	δA_i
WU	A_i^T	δA_o^T	δW



Hardware Performance

- **Hardware Simulation Setting**
  - Implemented in Verilog
  - Synthesized result of TSMC 65-nm CMOS tech
- **Power Breakdown**
  - Four Schemes With Different Accumulation Units



Compared with IBM[7]

- Better precision
- 9.8% power saving

Compared with NVDLA [10]

- 0.1% accuracy Loss
- 32% power saving

Outline

- Background of Low-Bit CNN Training
- Challenge: Low-Bit Accumulation
- Dynamic Group Accumulation (DGA) Algorithm
- Efficient DGA Hardware Design
- **Conclusion**

Conclusion

- DGA Solution to Achieve Low-bit CNN Training
  - **An Optimized DGA Algorithm**
    - Giving two assumption and theoretical derivation
  - **Efficient DGA Hardware Design**
    - Support efficient DGA unit in the accelerator
  - **Comprehensive Analysis on DGA**
    - Multi experiment on both DGA algorithm and hardware

Reduce accumulation bit-width by 6 bits

and save 32% power consumption

Reference

[1] A. Krizhevsky, I. Sutskever, G. E. Hinton. “Imagenet classification with deep convolutional neural networks,” Advances in neural information processing systems, 2012, pp. 1097-1105

[2] J. Devlin, MW. Chang, K. Lee, et al. “Bert: Pre-training of deep bidirectional transformers for language understanding,” arXiv preprint, arXiv:1810.04805, 2018.[

3] D. Silver, A. Huang, C. J. Maddison, et al. “Mastering the game of Go with deep neural networks and tree search,” nature, 2016, 529(7587): 484-489.

[4] J. Zhuang, T. Tang, Y. Ding, et al. “AdaBelief Optimizer: Adapting Stepsizes by the Belief in Observed Gradients,” Proceedings of the 34nd International Conference on Neural Information Processing Systems, 2020, 33.

[5] U. Evci, T. Gale, J. Menick, et al. “Rigging the lottery: Making all tickets winners,” International Conference on Machine Learning, 2020, pp. 2943-2952.

[6] J. Lee, J. Lee, D.Han, et al. “LNPU: A 25.3 TFLOPS/W sparse deep-neural-network learning processor with fine-grained mixed precision of FP8-FP16,” 2019 IEEE International Solid-State Circuits Conference-(ISSCC), 2019, pp. 142-144.

[7] A. Agrawal, S. K. Lee, J. Silberman, et al. “A 7nm 4-Core AI Chip with 25.6 TFLOPS Hybrid FP8 Training, 102.4 TOPS INT4 Inference and Workload-Aware Throttling,” 2021 IEEE International Solid-State Circuits Conference (ISSCC), 2021, pp. 144-146.

[8] H. Jiang, S. Huang, X.Peng, et al. “A two-way SRAM array based accelerator for deep neural network on-chip training,” Design Automation Conference (DAC), 2020, pp. 1-3.

Reference

[9] R. Banner, I. Hubara, E. Hoffer, et al. “Scalable methods for 8-bit training of neural networks,” Proceedings of the 32nd International Conference on Neural Information Processing Systems, 2018, pp. 5151-5159.

[10] “Nvdla deep learning accelerator, , in http://nvdla.org.

[11] Z. Yuan, Y. Liu, J. Yue, et al. “STICKER: An Energy-Efficient Multi-Sparsity Compatible Accelerator for Convolutional Neural Networks in 65-nm CMOS,” IEEE Journal of Solid-State Circuits, 2020, pp. 465-477.

[12] N. Wang, J. Choi, D. Brand, et al. “Training deep neural networks with 8-bit floating point numbers,” Proceedings of the 32nd International Conference on Neural Information Processing Systems, 2018, pp. 7686-7695.

[13] J. Park, S. Lee, D. Jeon, et al. “A 40nm 4.81 TFLOPS/W 8b Floating-Point Training Processor for Non-Sparse Neural Networks Using Shared Exponent Bias and 24-Way Fused Multiply-Add Tree,” 2021 IEEE International Solid-State Circuits Conference (ISSCC), 2021, pp. 1-3.

Thanks a lot!

Welcome for questions!