# MINT: Multiplier-less INTeger Quantization for Energy Efficient Spiking Neural Networks

Ruokai Yin, Yuhang Li, Abhishek Moitra, Priyadarshini Panda Department of Electrical Engineering, Yale University, USA Email: <u>ruokai.yin@yale.edu</u>





#### Neural Networks at GPU era



## Two Ways of Processing Neural Networks



Speed ↑ Cost ↑





Speed ↓ Cost ↓



#### **Two Ways of Processing Neural Networks**



#### **Two Ways of Processing Neural Networks**



# **Preliminary of Spiking Neural Networks**

Artificial Neural Networks (ANNs)





# **Preliminary of Spiking Neural Networks**

Artificial Neural Networks (ANNs)





## **Benefits of SNNs (Hardware)**

Artificial Neural Networks (ANNs)





Require multipliers (~32x power compared to adder)



## **Benefits of SNNs (Hardware)**

Artificial Neural Networks (ANNs)





## **Benefits of SNNs (Hardware)**

Artificial Neural Networks (ANNs)









References for those works are in the appendix.



References for those works are in the appendix.



References for those works are in the appendix.



Network Arch: *VGG-9* # of Timesteps: *4* Precision of *U*: *32-bit* 



Network Arch: *VGG-9* # of Timesteps: *4* Precision of *U*: *32-bit* 



Network Arch: *VGG-9* # of Timesteps: *4* Precision of *U*: *32-bit* 



Network Arch: *VGG-9* # of Timesteps: *4* Precision of *U*: *32-bit* 

#### **Key Observations:**

1. size of weight  $\downarrow$  portion of  $U \uparrow$ 

2. batch size  $\uparrow$  size of  $U \uparrow$ 

#### Naïve Solution: Quantization!

- Uniform Quantization:
  - An affine mapping between low-precision integer vectors q
    and high-precision floating point vectors r.



• An example for a layer in ANN:

Input: X Weight: W Output: Z

Without quantization: Z = ReLU(WX) With weight quantization:  $Z = ReLU(\alpha(\widehat{W}X))$ 

## **Apply Quantization to SNNs**

• Let's review the equations for SNNs again:

Step 1: update	$H_l^t = \tau U_l^{t-1} + W_l S_{l-1}^t$
Step 2: firing	$H_l^t > v_{th}$
Step 3: reset	$U_l^t = H_l^t (1 - S_l^t)$

• The goal is to quantize both the weights and membrane potentials.

#### **Apply Quantization to SNNs**

• Let's review the equations for SNNs again:

Step 1: update	$H_l^t = \tau U_l^{t-1} + W_l S_{l-1}^t$
Step 2: firing	$H_l^t > v_{th}$
Step 3: reset	$U_l^t = H_l^t (1 - S_l^t)$

• The goal is to quantize both the weights and membrane potentials.

Step 1: update
$$H_l^t = \tau U_l^{t-1} + W_l S_{l-1}^t$$
 $H_l^t = \alpha_2 \tau \widehat{U}_l^{t-1} + \alpha_1 \widehat{W}_l S_{l-1}^t$ Step 2: firing $H_l^t > v_{th}$  $H_l^t > v_{th}$ Step 3: reset $U_l^t = H_l^t (1 - S_l^t)$  $\alpha_3 \widehat{U}_l^t = H_l^t (1 - S_l^t)$ 

#### **Apply Quantization to SNNs**

• Let's review the equations for SNNs again:

Step 1: update	$H_l^t = \tau U_l^{t-1} + W_l S_{l-1}^t$
Step 2: firing	$H_l^t > v_{th}$
Step 3: reset	$U_l^t = H_l^t (1 - S_l^t)$

• The goal is to quantize both the weights and membrane potentials.

Step 1: update
$$H_l^t = \tau U_l^{t-1} + W_l S_{l-1}^t$$
 $H_l^t = \alpha_2 \widehat{U}_l^{t-1} + \alpha_1 \widehat{W}_l S_{l-1}^t$ Step 2: firing $H_l^t > v_{th}$  $H_l^t > v_{th}$ Step 3: reset $U_l^t = H_l^t (1 - S_l^t)$  $\alpha_3 \widehat{U}_l^t = H_l^t (1 - S_l^t)$ 

Scaling factors need multiplications!

## **Cost of Scaling Factors**

- To support the scaling factors, we physically need a 32-bit multiplier.
- Output-stationary systolic array is usually adopted for accelerating SNNs. For the quantization scaling, each processing element (PE) requires a 32-bit multiplier.



Narayanan, S, et al. "SpinalFlow: An architecture and dataflow tailored for spiking neural networks." (*ISCA*). IEEE, 2020.

• We first look at the LIF equations for updating the membrane potential after reset.

Step 1: update
$$H_l^t = \alpha_2 \tau \widehat{U}_l^{t-1} + \alpha_1 \widehat{W}_l S_{l-1}^t$$
Step 2: firing $H_l^t > v_{th}$  $\square \qquad \alpha_3 \widehat{U}_l^t = \alpha_2 \tau \widehat{U}_l^{t-1} + \alpha_1 \widehat{W}_l S_{l-1}^t$ Step 3: reset $\alpha_3 \widehat{U}_l^t = H_l^t (1 - S_l^t)$ Assume no  
output spikes

• We first look at the LIF equations for updating the membrane potential after reset.

Step 1: update
$$H_l^t = \alpha_2 \tau \widehat{U}_l^{t-1} + \alpha_1 \widehat{W}_l S_{l-1}^t$$
Step 2: firing $H_l^t > v_{th}$  $\square \qquad \alpha_3 \widehat{U}_l^t = \alpha_2 \tau \widehat{U}_l^{t-1} + \alpha_1 \widehat{W}_l S_{l-1}^t$ Step 3: reset $\alpha_3 \widehat{U}_l^t = H_l^t (1 - S_l^t)$ Assume no  
output spikes

 We first look at the LIF equations for updating the membrane potential after reset.



How to get rid of those two multiplications?

How about adding the following constraints to the scaling factors?

$$\alpha_1 = \alpha_2 = \alpha_3$$

 We first look at the LIF equations for updating the membrane potential after reset.



How about adding the following constraints to the scaling factors?

 $\alpha_1 = \alpha_2 = \alpha_3$ 

• We then look at the LIF equations for firing output spikes.

Step 1: update
$$H_l^t = \alpha_2 \tau \widehat{U}_l^{t-1} + \alpha_1 \widehat{W}_l S_{l-1}^t$$
Step 2: firing $H_l^t > v_{th}$  $intherefore a > v_{th}$ Step 3: reset $\alpha_3 \widehat{U}_l^t = H_l^t (1 - S_l^t)$ Firing condition

$$\alpha_1 = \alpha_2 = \alpha_3 = \alpha$$

• We then look at the LIF equations for firing output spikes.

Step 1: update
$$H_l^t = \alpha_2 \tau \widehat{U}_l^{t-1} + \alpha_1 \widehat{W}_l S_{l-1}^t$$
Step 2: firing $H_l^t > v_{th}$  $inthered in item initial initial$ 

$$\alpha_1 = \alpha_2 = \alpha_3 = \alpha$$

$$\alpha_2 \tau \widehat{U}_l^{t-1} + \alpha_1 \widehat{W}_l S_{l-1}^t > v_{th} \quad \Longrightarrow \quad \tau \widehat{U}_l^{t-1} + \widehat{W}_l S_{l-1}^t > \frac{v_{th}}{\alpha}$$

• We then look at the LIF equations for firing output spikes.

Step 1: update
$$H_l^t = \alpha_2 \tau \widehat{U}_l^{t-1} + \alpha_1 \widehat{W}_l S_{l-1}^t$$
Step 2: firing $H_l^t > v_{th}$  $inthered in item initial initial$ 

$$\alpha_1 = \alpha_2 = \alpha_3 = \alpha$$

$$\alpha_{2}\tau\widehat{U}_{l}^{t-1} + \alpha_{1}\widehat{W}_{l}S_{l-1}^{t} > v_{th} \quad \Longrightarrow \quad \overline{\tau\widehat{U}_{l}^{t-1} + \widehat{W}_{l}S_{l-1}^{t}} > \frac{v_{th}}{\alpha}$$

• We then look at the LIF equations for firing output spikes.

Step 1: update
$$H_l^t = \alpha_2 \tau \widehat{U}_l^{t-1} + \alpha_1 \widehat{W}_l S_{l-1}^t$$
Step 2: firing $H_l^t > v_{th}$  $inthered in item initial initial$ 

$$\alpha_1 = \alpha_2 = \alpha_3 = \alpha$$

• We then look at the LIF equations for firing output spikes.

Step 1: update
$$H_l^t = \alpha_2 \tau \widehat{U}_l^{t-1} + \alpha_1 \widehat{W}_l S_{l-1}^t$$
Step 2: firing $H_l^t > v_{th}$  $intherefore a > v_{th}$ Step 3: reset $\alpha_3 \widehat{U}_l^t = H_l^t (1 - S_l^t)$ Firing condition

Remember the shared scaling factors we just discussed.

$$\alpha_1 = \alpha_2 = \alpha_3 = \alpha$$

Equivalent of scaling the firing threshold!

#### **MINT Quantization**

• The forward algorithm for our proposed MINT (Multiplierless INTeger) quantization is formalized as below:

$$\begin{split} H_{l}^{t} \leftarrow \widehat{W}_{l}S_{l-1}^{t} + \widehat{U}_{l}^{t-1} \gg 1 \\ & if \ H_{l}^{t} \geq \left\lceil \frac{v_{th}}{\alpha} \right\rceil then \\ & S_{l}^{t} \leftarrow 1 \\ & \widehat{U}_{l}^{t} \leftarrow 0 \\ & else \\ & S_{l}^{t} \leftarrow 0 \\ & \widehat{U}_{l}^{t} \leftarrow H_{l}^{t} \\ & endif \end{split}$$

Low Storage of weights Low Storage of potentials No Multipliers Fully Integer

## Training of MINT Quantized SNNs

• We utilized Quantization Aware Training (QAT) to train our SNNs based on Backpropagation-through-Time (BPTT).



• We further made the shared quantization scaling factor  $\alpha$  an learned parameter, which significantly improved the accuracy.

#### **Projection on SNN Accelerators**

 Processing element (PE) differences between MINT and vanilla Uniform Quantization (UQ) inside the SNN accelerator systems.



Technology	CMOS32	
PE array size	128	
Global buffer size	144 KB	
Spad size	1 KB	

#### **Projection on SNN Accelerators**

• Processing element (PE) differences between MINT and vanilla Uniform Quantization (UQ) inside the SNN accelerator systems.



- Datasets (Networks): CIFAR-10 and Tiny-ImageNet (VGG9 and VGG16).
- Various Precision (*W*-*U*): 8-8, 4-4, 2-2.
- At iso-accuracy with the full-precision models: MINT has less total memory footprint : ~93% total memory footprint reduction with 2-2 precision

Dataset	VGG-9	Acc. (%)	VGG-16	Acc. (%)
CIFAR10	fp32	88.03	fp32	91.15
	w8u8	87.48	w8u8	90.72
	w4u4	87.37	w4u4	90.65
	w2u2	87.47	w2u2	90.56
TinyImage	fp32	46.38	fp32	48.32
	w8u8	45.30	w8u8	50.18
	w4u4	45.02	w4u4	49.36
	w2u2	44.95	w2u2	48.60



- Datasets (Networks): CIFAR-10 and Tiny-ImageNet (VGG9 and VGG16).
- Various Precision (*W*-*U*): 8-8, 4-4, 2-2.
- The importance of quantizing membrane potential gets more significant when the batch size increased.



- Simulated energy comparison between the MINT and vanilla UQ across different precision.
- Simulated memory movement energy cost of MINT across different precision.



Yin, Ruokai, et al. "Sata: Sparsity-aware training accelerator for spiking neural networks." *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2022). https://github.com/RuokaiYin/SATA\_Sim

• Compare with other SOTA SNN quantization work, MINT achieves much smaller total memory footprint at iso-accuracy.

Method	Precision	Accuracy (%)	Mini	Memory
(CIFAR-10)	(W / U)	Top-1	Batches	Footprint (MB)
STBP-Quant	8 / 14	86.65	50	353.79
MINT (Ours)	8 / 8	<b>88.25</b>	50	<b>95.41</b>
ST-Quant	5 / 32	<b>88.6</b>	32	751.04
MINT (Ours)	5 / 5	88.04	32	<b>59.62</b>
ADMM-Quant	4 / 32	<b>89.4</b>	50	1279.66
STBP-Quant	4 / 10	84.99	50	248.39
<b>MINT (Ours)</b>	4 / 4	88.12	50	<b>47.71</b>
ADMM-Quant	2 / 32	<b>89.23</b>	50	1264.85
STBP-Quant	2 / 8	33.53	50	195.68
<b>MINT (Ours</b> )	2 / 2	88.39	50	<b>23.85</b>

Deng, Lei, et al. "Comprehensive snn compression using admm optimization and activity regularization." (TNNLS). IEEE, 2021.

Tan, Pai-Yu, and Cheng-Wen Wu. "A Low-Bitwidth Integer-STBP Algorithm for Efficient Training and Inference of Spiking Neural Networks." Proceedings of the 28th Asia and South Pacific Design Automation Conference. 2023.

Chowdhury, Sayeed Shafayet, Isha Garg, and Kaushik Roy. "Spatio-temporal pruning and quantization for low-latency spiking neural networks." 2021 International Joint Conference on Neural Networks (IJCNN). IEEE, 2021.

- MINT is agnostic to the underlying hardware used for deployment.
- Compared to other SOTA SNN quantization work that requires scaling factors, MINT achieves a much smaller circuit area on other existing SNN accelerator systems.



Narayanan, S, et al. "SpinalFlow: An architecture and dataflow tailored for spiking neural networks." *(ISCA).* IEEE, 2020. Lee, J, et al. "Parallel time batching: Systolic-array acceleration of sparse spiking neural computation." (HPCA). IEEE, 2022. Deng, Lei, et al. "Comprehensive snn compression using admm optimization and activity regularization." (TNNLS). IEEE, 2021.



Code is available at: <a href="https://github.com/RuokaiYin/MINT\_Quantization/">https://github.com/RuokaiYin/MINT\_Quantization/</a>

#### Please cite this work if found interesting:

Yin R, et al. MINT: Multiplier-less Integer Quantization for Spiking Neural Networks[J]. arXiv preprint arXiv:2305.09850, 2023.

Q & A?



# Appendix

BNTT :

Kim, Youngeun, and Priyadarshini Panda. "Revisiting batch normalization for training low-latency deep spiking neural networks from scratch." Frontiers in neuroscience 15 (2021): 773954. Direct SNN:

Wu, Yujie, et al. "Direct training for spiking neural networks: Faster, larger, better." Proceedings of the AAAI conference on artificial intelligence. Vol. 33. No. 01. 2019.

Event BP:

Zhu, Yaoyu, et al. "Training spiking neural networks with event-driven backpropagation." Advances in Neural Information Processing Systems 35 (2022): 30528-30541.

TSSL:

Zhang, Wenrui, and Peng Li. "Temporal spike sequence learning via backpropagation for deep spiking neural networks." Advances in Neural Information Processing Systems 33 (2020): 12022-12033.

#### TDBN:

Zheng, Hanle, et al. "Going deeper with directly-trained larger spiking neural

networks." Proceedings of the AAAI conference on artificial intelligence. Vol. 35. No. 12. 2021. LTH:

Kim, Youngeun, et al. "Exploring lottery ticket hypothesis in spiking neural networks." European Conference on Computer Vision. Cham: Springer Nature Switzerland, 2022.