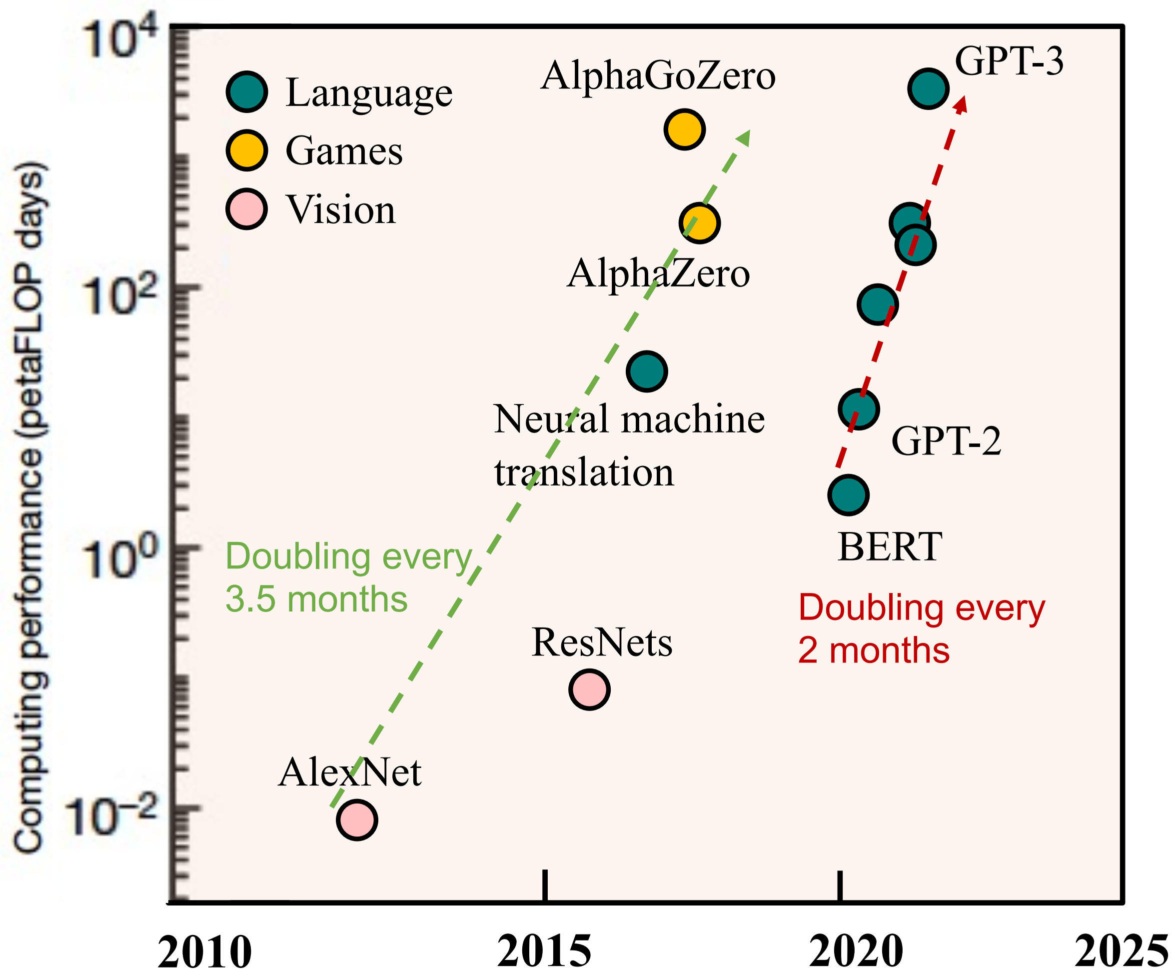
MINT: Multiplier-less INTeger Quantization for Energy Efficient Spiking Neural Networks

Ruokai Yin, Yuhang Li, Abhishek Moitra, Priyadarshini Panda

Department of Electrical Engineering, Yale University, USA

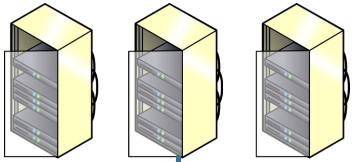
Email: ruokai.yin@yale.edu

Neural Networks at GPU era



Two Ways of Processing Neural Networks

Speed ↑
Cost ↑



Speed ↓
Cost ↓

Two Ways of Processing Neural Networks

What Spiking Neural Networks (SNNs) are targeting at.

Speed ↓
Cost ↓

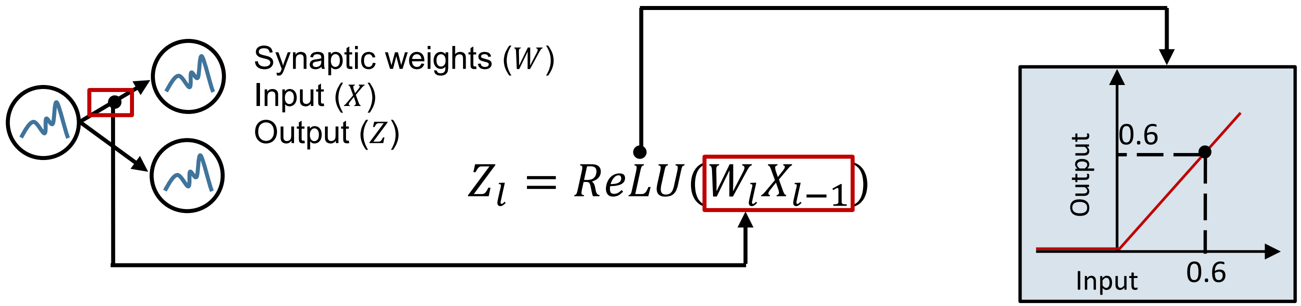
Two Ways of Processing Neural Networks

Speed ↓
Cost ↓

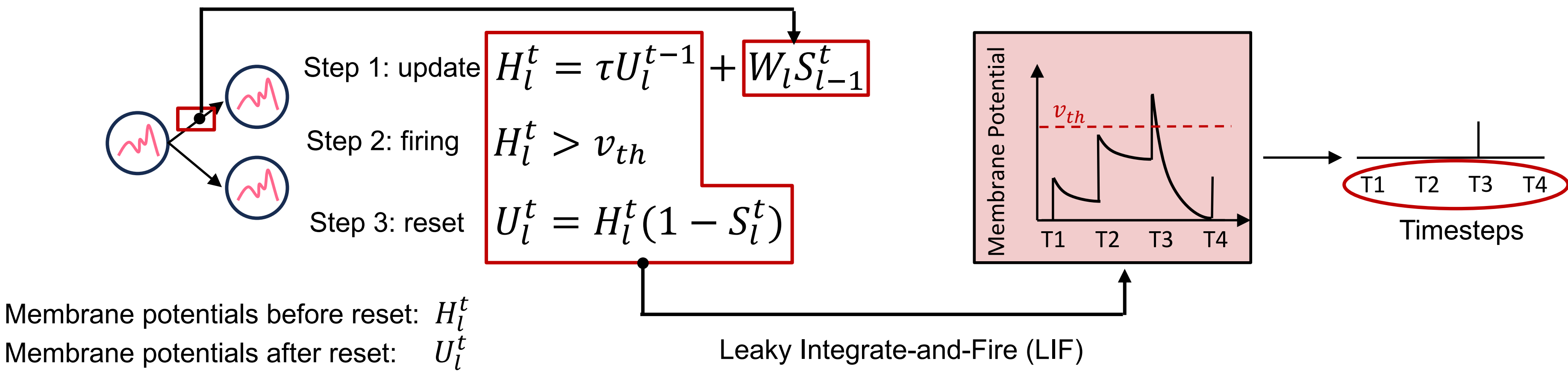
What **Spiking Neural Networks** (**SNNs**) are targeting.

Preliminary of Spiking Neural Networks

Artificial Neural Networks (ANNs)

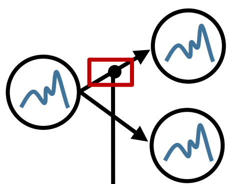


Spiking Neural Networks (SNNs)



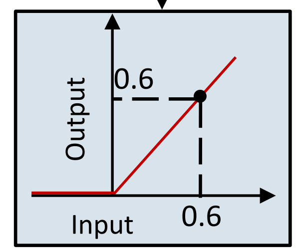
Preliminary of Spiking Neural Networks

Artificial Neural Networks (ANNs)

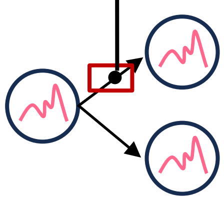


Synaptic weights (W)
Input (X)
Output (Z)

$$Z_l = ReLU(\boxed{W_l X_{l-1}})$$



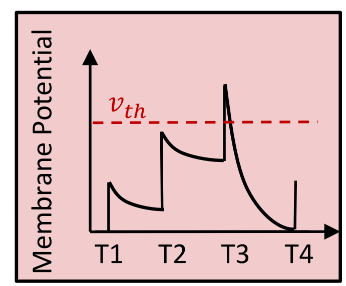
Spiking Neural Networks (SNNs)



Step 1: update $H_l^t = \tau U_l^{t-1} + \boxed{W_l S_{l-1}^t}$

Step 2: firing $H_l^t > v_{th}$

Step 3: reset $U_l^t = H_l^t(1 - S_l^t)$



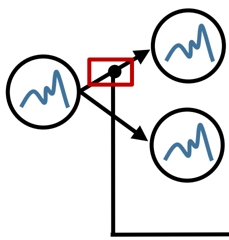
Timesteps

Leaky Integrate-and-Fire (LIF)

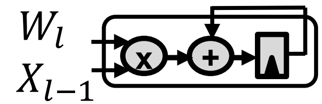
Membrane potentials before reset: H_l^t
Membrane potentials after reset: $\boldsymbol{U_l^t}$

Benefits of SNNs (Hardware)

Artificial Neural Networks (ANNs)

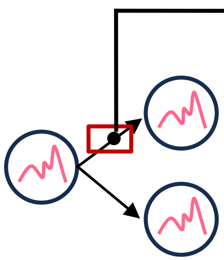


$$Z_l = ReLU(\boxed{W_l X_{l-1}})$$



Require multipliers (~32x power compared to adder)

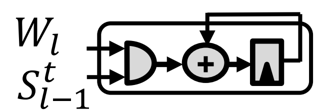
Spiking Neural Networks (SNNs)



$$H_l^t = \tau U_l^{t-1} + \boxed{W_l S_{l-1}^t}$$

$$H_l^t > v_{th}$$

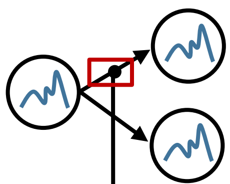
$$U_l^t = H_l^t(1 - S_l^t)$$



Does not require multipliers

Benefits of SNNs (Hardware)

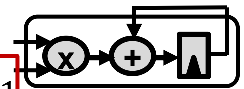
Artificial Neural Networks (ANNs)



$$Z_l = ReLU(\boxed{W_l X_{l-1}})$$

W_l

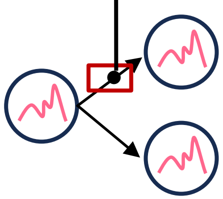
$\boxed{X_{l-1}}$



Require multipliers

Multi-bits value
Higher storage

Spiking Neural Networks (SNNs)



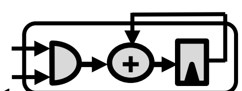
$$H_l^t = \tau U_l^{t-1} + \boxed{W_l S_{l-1}^t}$$

$$H_l^t > v_{th}$$

$$U_l^t = H_l^t(1 - S_l^t)$$

W_l

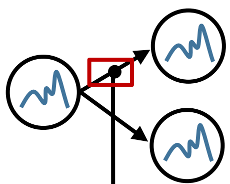
S_{l-1}^t



Does not require multipliers

Benefits of SNNs (Hardware)

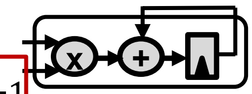
Artificial Neural Networks (ANNs)



$$Z_l = ReLU(\boxed{W_l X_{l-1}})$$

W_l

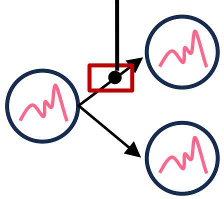
$\boxed{X_{l-1}}$



Require multipliers

Multi-bits value
Higher storage

Spiking Neural Networks (SNNs)



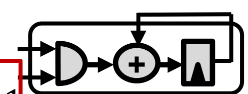
$$H_l^t = \tau U_l^{t-1} + \boxed{W_l S_{l-1}^t}$$

$$H_l^t > v_{th}$$

$$U_l^t = H_l^t (1 - S_l^t)$$

W_l

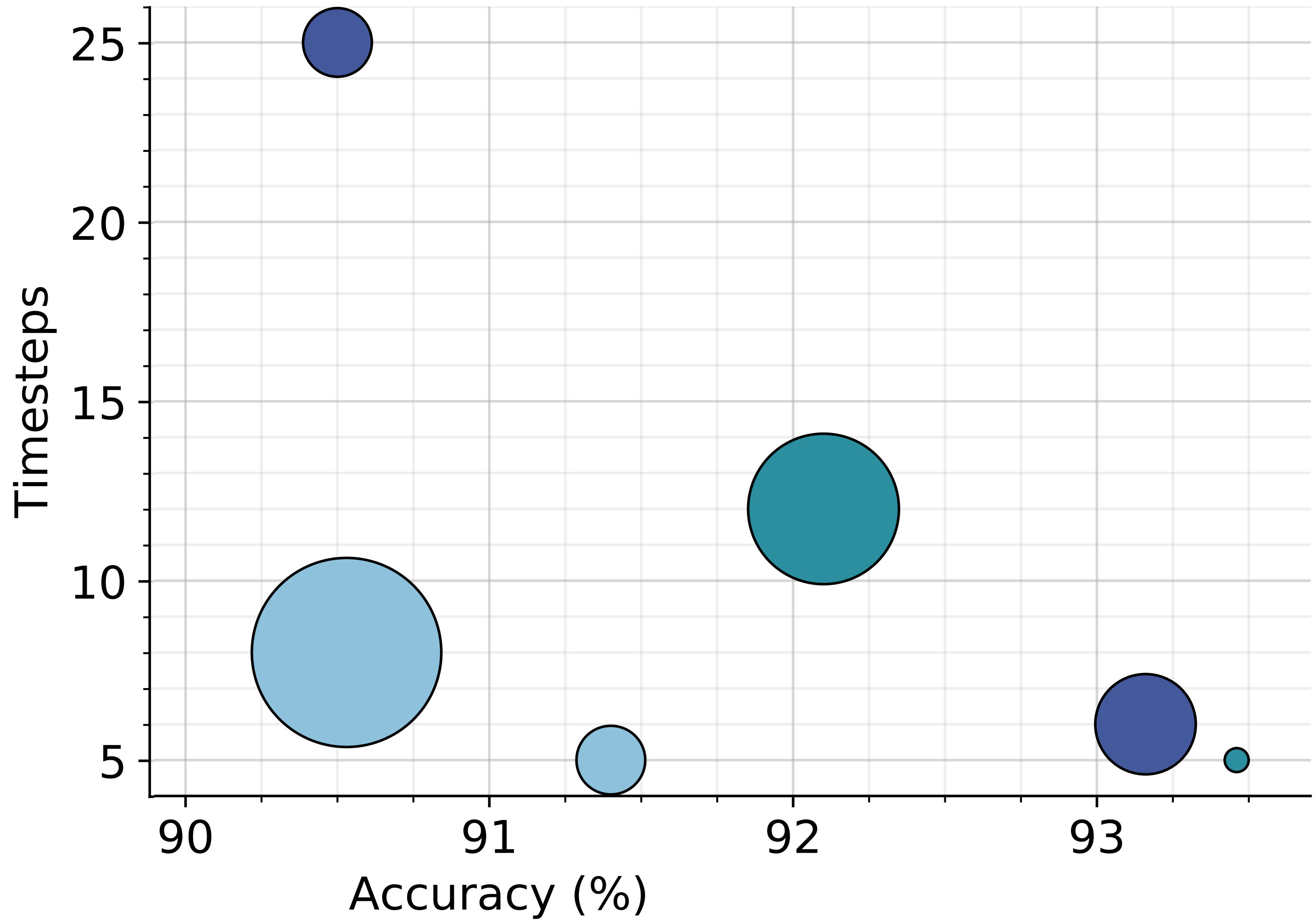
$\boxed{S_{l-1}^t}$



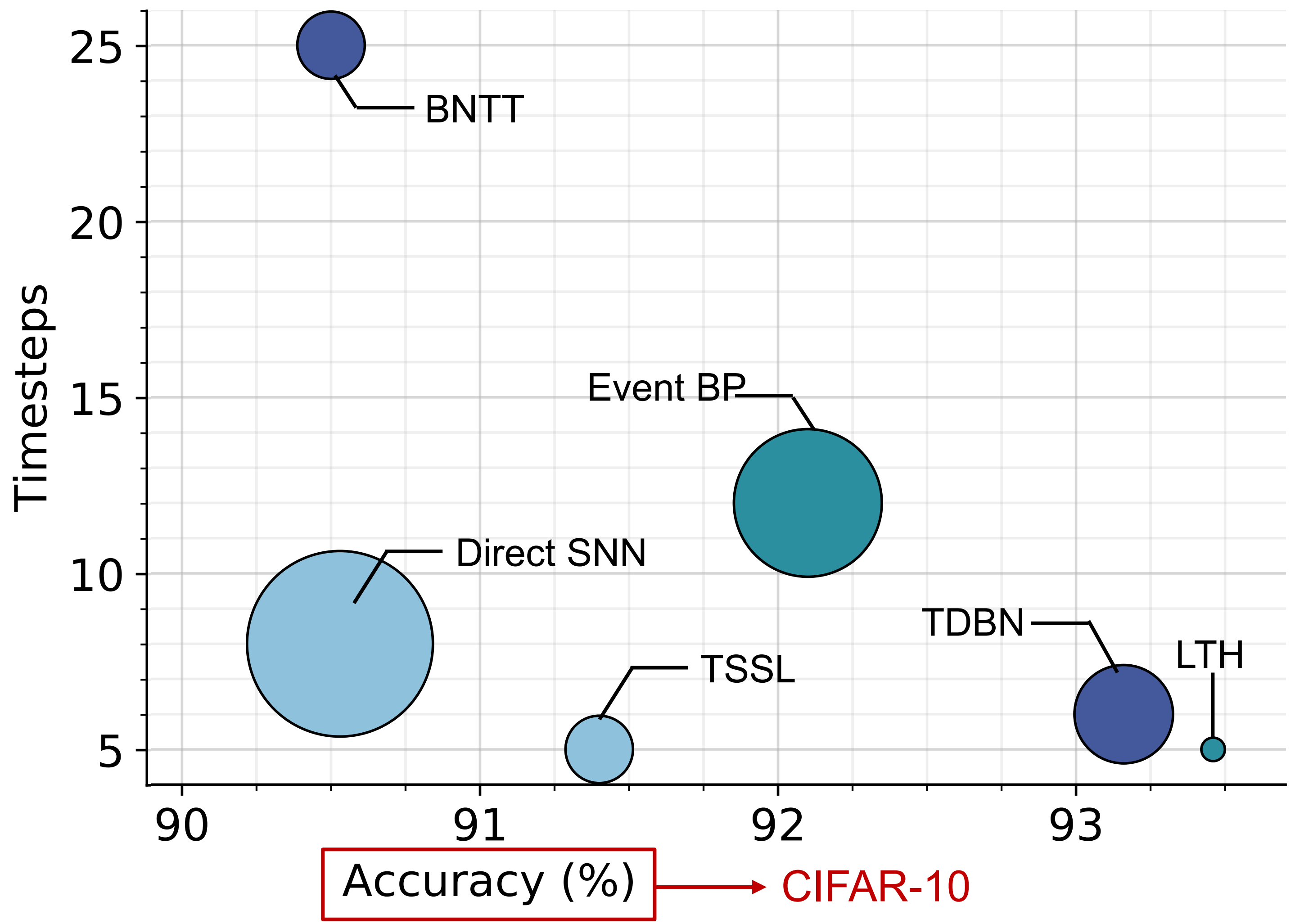
Does not require multipliers

Single-bit spike
Lower storage

Model Size of SNNs

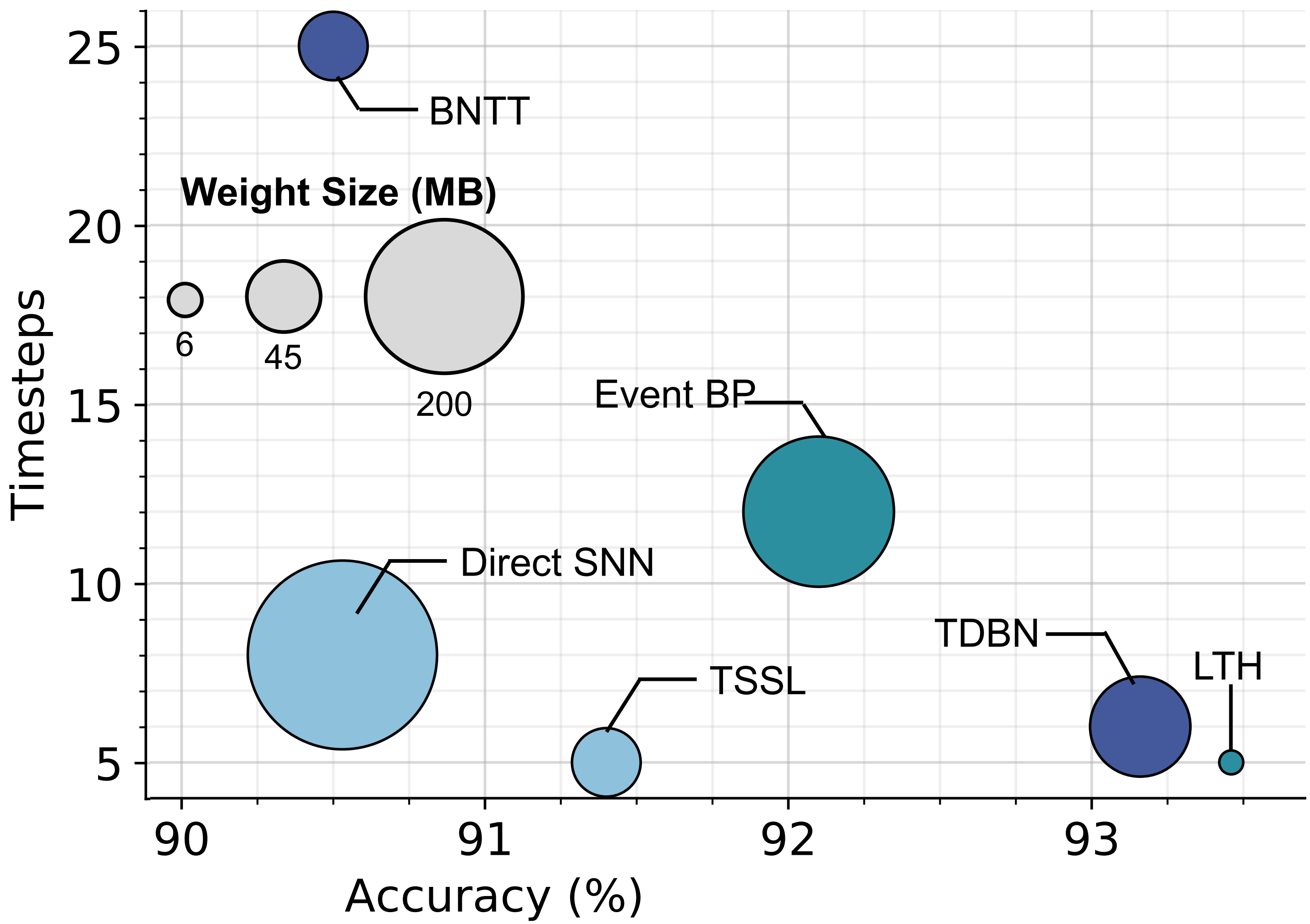


Model Size of SNNs



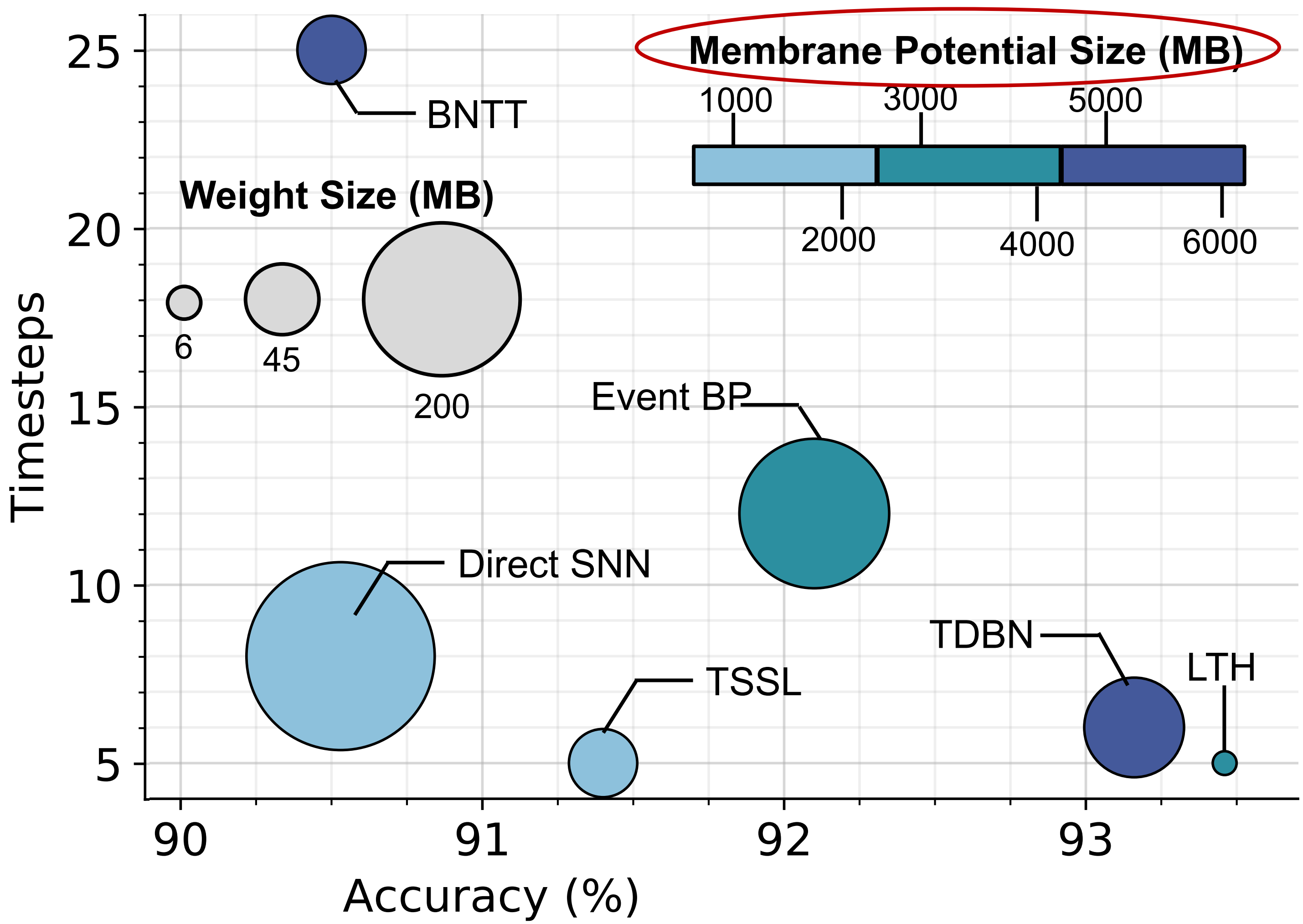
References for those works are in the appendix.

Model Size of SNNs



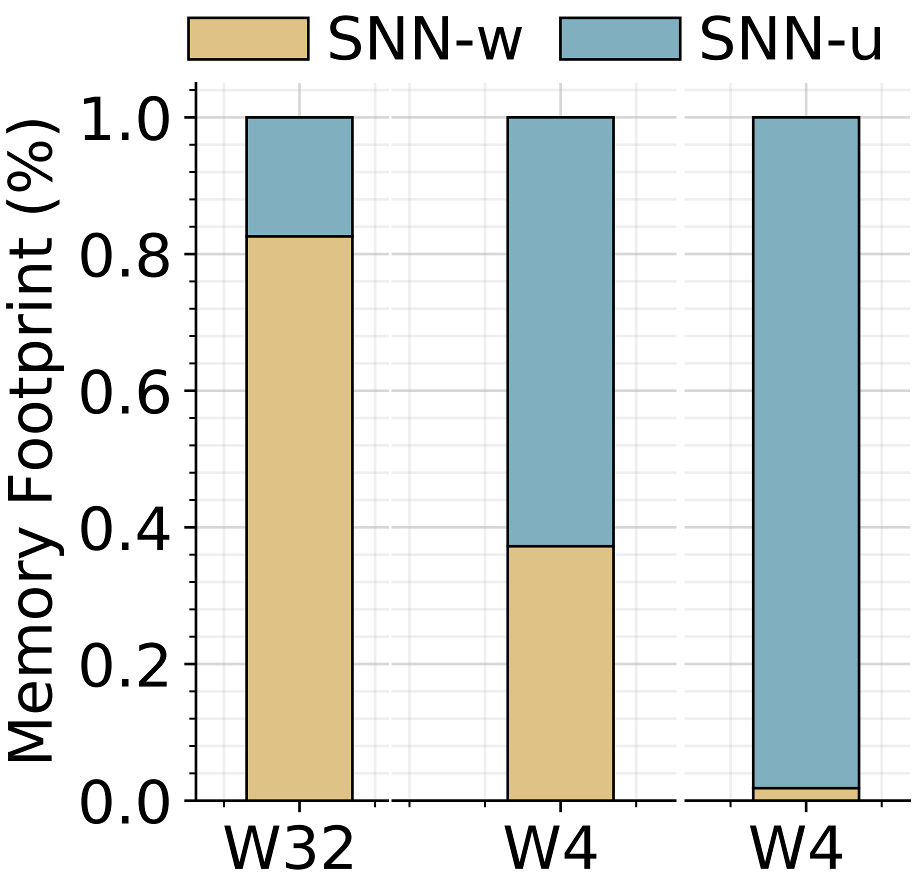
References for those works are in the appendix.

Model Size of SNNs



References for those works are in the appendix.

Size of Membrane Potential (U) in SNNs



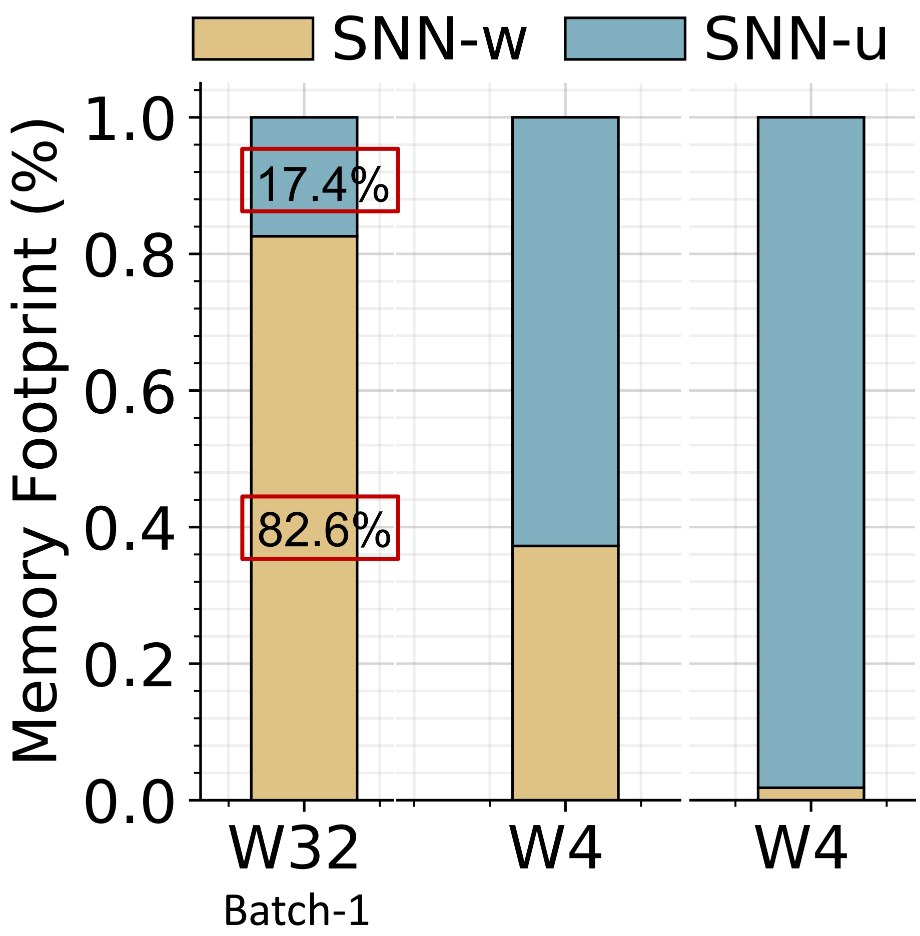
Network Arch: *VGG-9*

\# of Timesteps: *4*

Precision of U: *32-bit*

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

Size of Membrane Potential (U) in SNNs



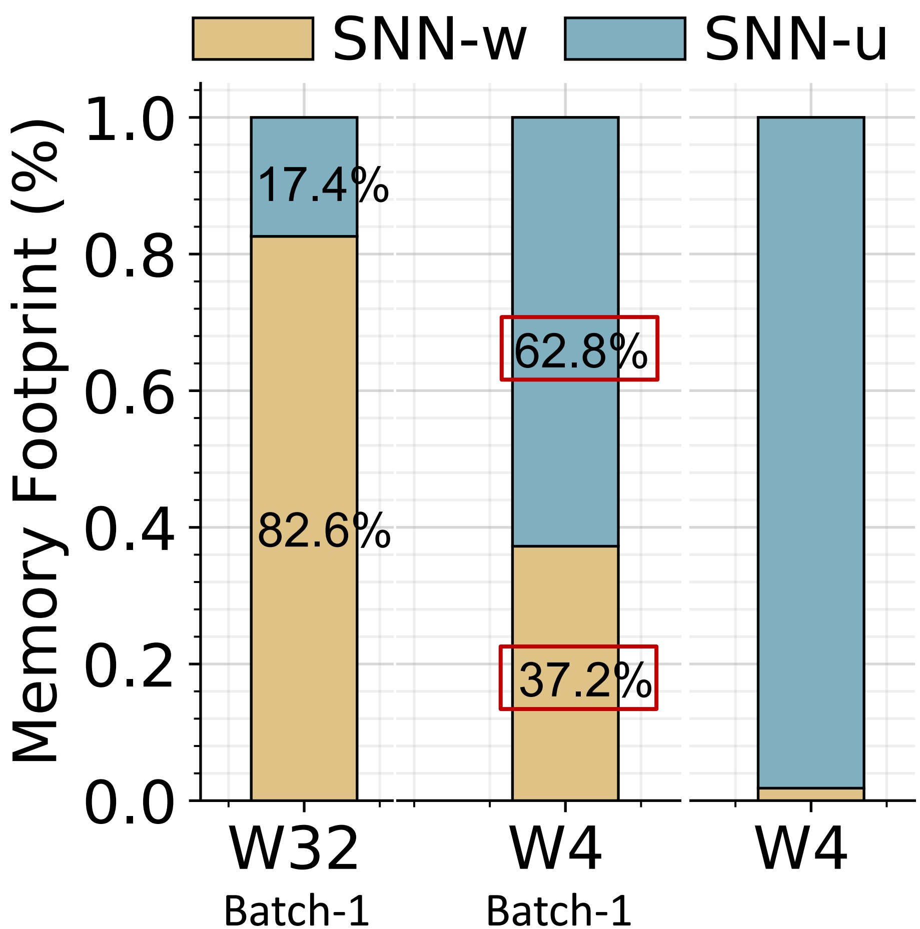
Network Arch: *VGG-9*

\# of Timesteps: *4*

Precision of U: *32-bit*

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

Size of Membrane Potential (U) in SNNs



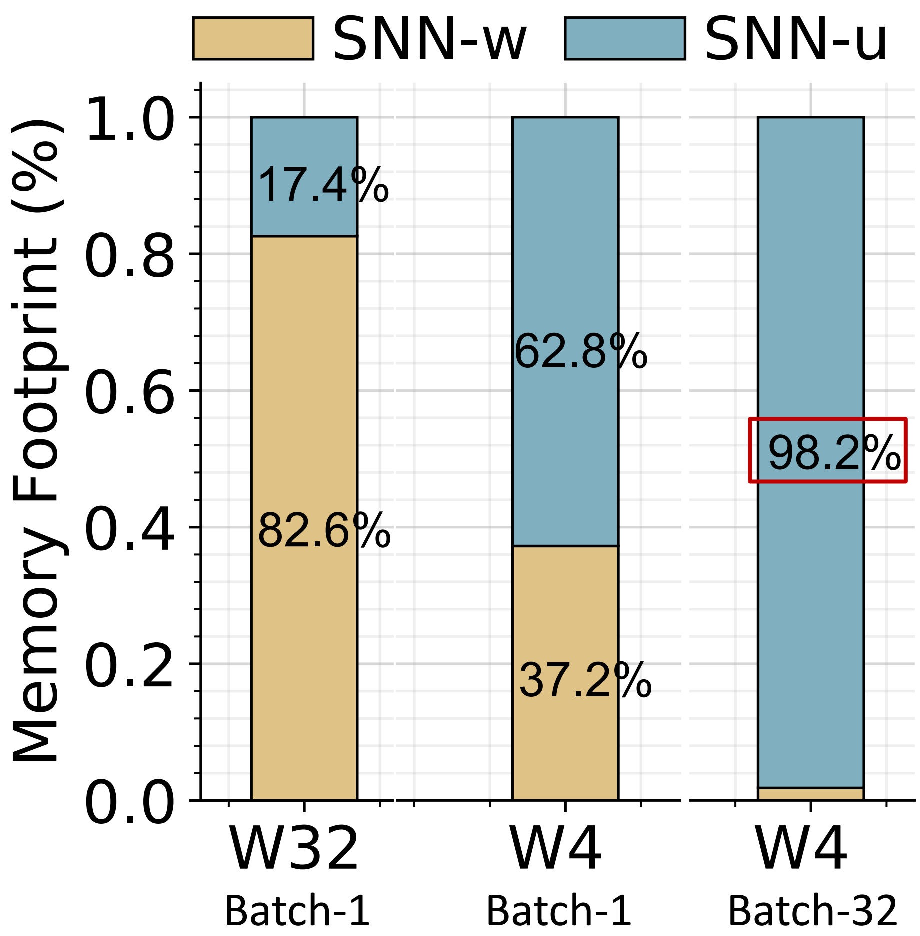
Network Arch: *VGG-9*

# of Timesteps: *4*

Precision of U: *32-bit*

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

Size of Membrane Potential (U) in SNNs



Network Arch: *VGG-9*

# of Timesteps: *4*

Precision of U: *32-bit*

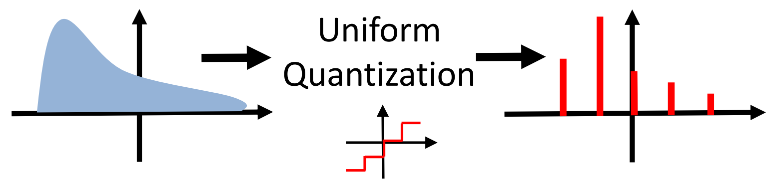
Key Observations:

1. size of weight ↓ portion of U ↑
2. batch size ↑ size of U ↑

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

Naïve Solution: Quantization!

- Uniform Quantization:
  - An affine mapping between low-precision integer vectors $\hat{q}$ and high-precision floating point vectors r.



Low-precision integer

$$r = \boxed{\alpha}\left(\boxed{\hat{q}} - Z\right) \xrightarrow{\text{simplify}} r = \alpha \cdot \hat{q}$$

Scaling factor: 32-bit floating point

- 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.

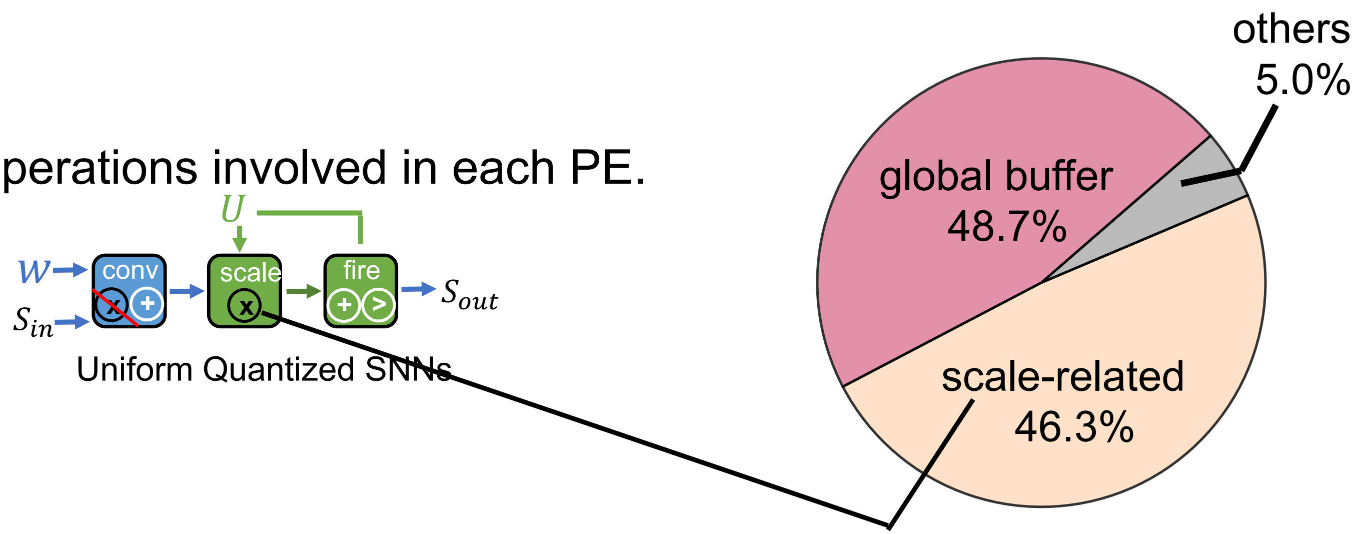
Step 1: update	$H_l^t = \tau U_l^{t-1} + W_l S_{l-1}^t$	⇒	$H_l^t = \alpha_2 \tau \widehat{U}_l^{t-1} + \alpha_1 \widehat{W}_l S_{l-1}^t$
Step 2: firing	$H_l^t > v_{th}$		$H_l^t > v_{th}$
Step 3: reset	$U_l^t = H_l^t (1 - S_l^t)$		$\alpha_3 \widehat{U}_l^t = H_l^t (1 - S_l^t)$

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.

The operations involved in each PE.



Area breakdown of a 4-bit SpinalFlow

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

Our methods: Review of LIF Equations

- 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 $H_l^t > v_{th}$

Step 3: reset $\alpha_3 \widehat{U}_l^t = H_l^t (1 - S_l^t)$

Assume no output spikes

$$\alpha_3 \widehat{U}_l^t = \alpha_2 \tau \widehat{U}_l^{t-1} + \alpha_1 \widehat{W}_l S_{l-1}^t$$

Our methods: Review of LIF Equations

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

Step 1: update $H_l^t = \alpha_2 \tau \hat{U}_l^{t-1} + \alpha_1 \widehat{W_l} S_{l-1}^t$

Step 2: firing $H_l^t > v_{th}$

Step 3: reset $\alpha_3 \hat{U}_l^t = H_l^t (1 - S_l^t)$

Assume no output spikes

$$\alpha_3 \hat{U}_l^t = \alpha_2 \tau \hat{U}_l^{t-1} + \alpha_1 \widehat{W_l} S_{l-1}^t$$

Our methods: Review of LIF Equations

- 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 $H_l^t > v_{th}$

Step 3: reset $\alpha_3 \widehat{U}_l^t = H_l^t (1 - S_l^t)$

$$\alpha_3 \widehat{U}_l^t = \alpha_2 \tau \widehat{U}_l^{t-1} + \alpha_1 \widehat{W}_l S_{l-1}^t$$

Assume no output spikes

$$\widehat{U}_l^t = \frac{\alpha_2}{\alpha_3} \tau \widehat{U}_l^{t-1} + \frac{\alpha_1}{\alpha_3} \widehat{W}_l S_{l-1}^t$$

How to get rid of those two multiplications?

How about adding the following constraints to the scaling factors?

$$\boxed{\alpha_1 = \alpha_2 = \alpha_3}$$

Our methods: Review of LIF Equations

- 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 $H_l^t > v_{th}$

Step 3: reset $\alpha_3 \widehat{U}_l^t = H_l^t (1 - S_l^t)$

Assume no output spikes

$$\alpha_3 \widehat{U}_l^t = \alpha_2 \tau \widehat{U}_l^{t-1} + \alpha_1 \widehat{W}_l S_{l-1}^t$$

$$\widehat{U}_l^t = \cancel{\frac{\alpha_2}{\alpha_3}} \tau \widehat{U}_l^{t-1} + \cancel{\frac{\alpha_1}{\alpha_3}} \widehat{W}_l S_{l-1}^t$$

Sharing the scaling factors between W and U

$$\widehat{U}_l^t = \tau \widehat{U}_l^{t-1} + \widehat{W}_l S_{l-1}^t$$

How to get rid of those two multiplications?

How about adding the following constraints to the scaling factors?

$$\boxed{\alpha_1 = \alpha_2 = \alpha_3}$$

Our methods: Review of LIF Equations

- 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 $H_l^t > v_{th}$ ⇒ $\alpha_2 \tau \widehat{U}_l^{t-1} + \alpha_1 \widehat{W_l} S_{l-1}^t > v_{th}$

Firing condition

Step 3: reset $\alpha_3 \widehat{U}_l^t = H_l^t (1 - S_l^t)$

Remember the shared scaling factors we just discussed.

$$\boxed{\alpha_1 = \alpha_2 = \alpha_3 = \alpha}$$

Our methods: Review of LIF Equations

- 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 $H_l^t > v_{th}$ $\Rightarrow$ $\alpha_2 \tau \widehat{U}_l^{t-1} + \alpha_1 \widehat{W}_l S_{l-1}^t > v_{th}$

Firing condition

Step 3: reset $\alpha_3 \widehat{U}_l^t = H_l^t (1 - S_l^t)$

Remember the shared scaling factors we just discussed.

$$\boxed{\alpha_1 = \alpha_2 = \alpha_3 = \alpha}$$

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

Our methods: Review of LIF Equations

- 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 $H_l^t > v_{th}$ $\Rightarrow$ $\alpha_2 \tau \widehat{U}_l^{t-1} + \alpha_1 \widehat{W}_l S_{l-1}^t > v_{th}$

Firing condition

Step 3: reset $\alpha_3 \widehat{U}_l^t = H_l^t (1 - S_l^t)$

Remember the shared scaling factors we just discussed.

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

$$\alpha_2 \tau \widehat{U}_l^{t-1} + \alpha_1 \widehat{W}_l S_{l-1}^t > v_{th} \Rightarrow \underbrace{\boxed{\tau \widehat{U}_l^{t-1} + \widehat{W}_l S_{l-1}^t}}_{\text{integers}} > \frac{v_{th}}{\alpha}$$

Our methods: Review of LIF Equations

- 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 $H_l^t > v_{th}$ $\Rightarrow$ $\alpha_2 \tau \widehat{U}_l^{t-1} + \alpha_1 \widehat{W}_l S_{l-1}^t > v_{th}$

Firing condition

Step 3: reset $\alpha_3 \widehat{U}_l^t = H_l^t (1 - S_l^t)$

Remember the shared scaling factors we just discussed.

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

integers

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

$$\tau \widehat{U}_l^{t-1} + \widehat{W}_l S_{l-1}^t \geq [\frac{v_{th}}{\alpha}]$$

Our methods: Review of LIF Equations

- 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 $H_l^t > v_{th}$ $\Rightarrow$ $\alpha_2 \tau \widehat{U}_l^{t-1} + \alpha_1 \widehat{W}_l S_{l-1}^t > v_{th}$

Firing condition

Step 3: reset $\alpha_3 \widehat{U}_l^t = H_l^t (1 - S_l^t)$

Remember the shared scaling factors we just discussed.

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

integers

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

integers

$$\boxed{\tau \widehat{U}_l^{t-1} + \widehat{W}_l S_{l-1}^t} \geq \boxed{\left\lceil \frac{v_{th}}{\alpha} \right\rceil}$$

integer

Equivalent of scaling the firing threshold!

MINT Quantization

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

$$H_l^t \leftarrow \widehat{W_l} S_{l-1}^t + \widehat{U}_l^{t-1} \gg 1$$

$$if\ H_l^t \geq \left\lceil \frac{v_{th}}{\alpha} \right\rceil then$$

$$S_l^t \leftarrow 1$$

$$\widehat{U}_l^t \leftarrow 0$$

$$else$$

$$S_l^t \leftarrow 0$$

$$\widehat{U}_l^t \leftarrow H_l^t$$

$$endif$$

Low Storage of weights

Low Storage of potentials

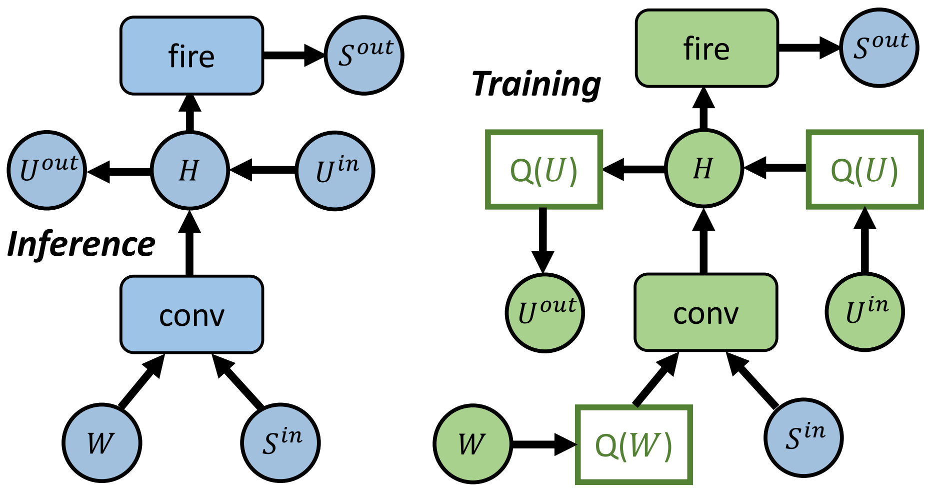
No Multipliers

Fully Integer

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

Training of MINT Quantized SNNs

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

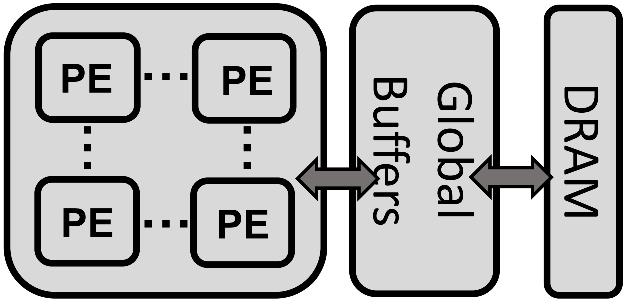


- We further made the shared quantization scaling factor α an learned parameter, which significantly improved the accuracy.

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

Projection on SNN Accelerators

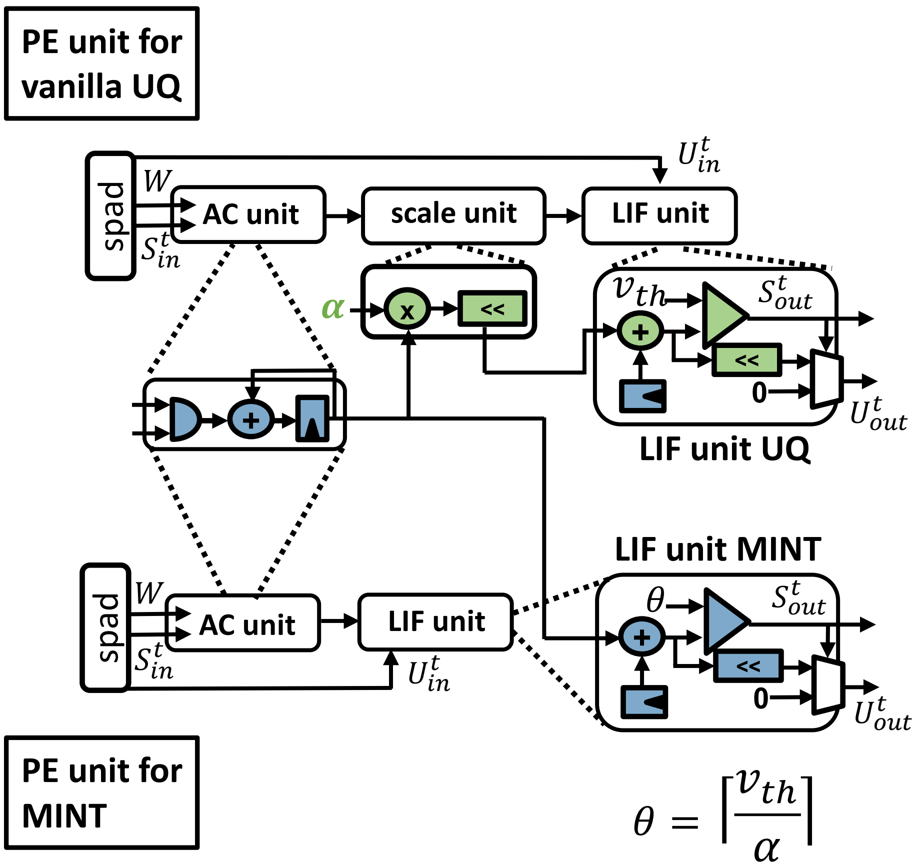
- Processing element (PE) differences between MINT and vanilla Uniform Quantization (UQ) inside the SNN accelerator systems.

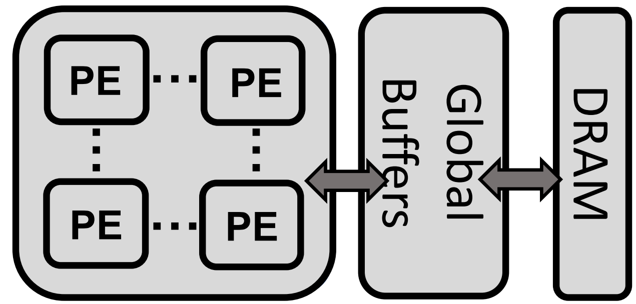


Technology	CMOS32
PE array size	128
Global buffer size	144 KB
Spad size	1 KB

Projection on SNN Accelerators

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





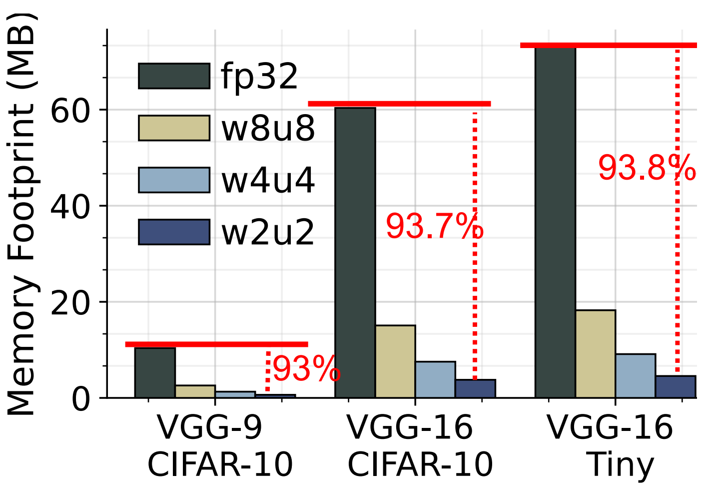
Technology	CMOS32
PE array size	128
Global buffer size	144 KB
Spad size	1 KB

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

Experimental Results

- 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

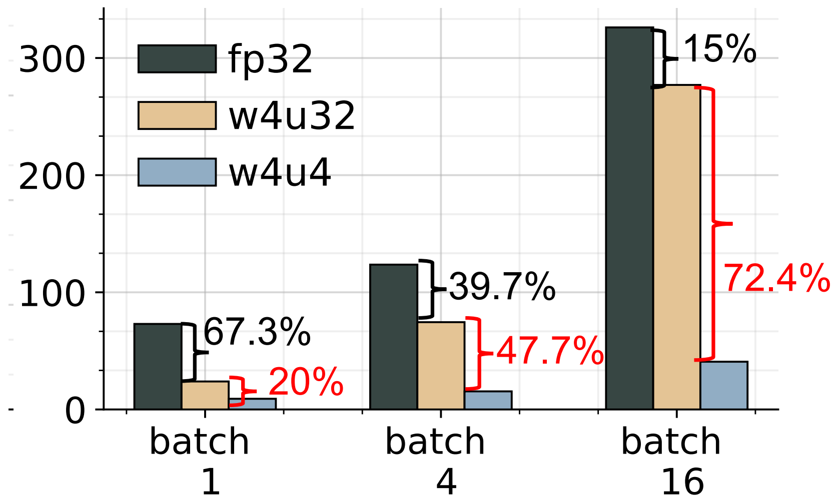


Yin R, et al. MINT: Multiplier-less Integer Quantization for Spiking Neural Networks[J]. arXiv preprint arXiv:2305.09850, 2023.

Experimental Results

- 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.

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

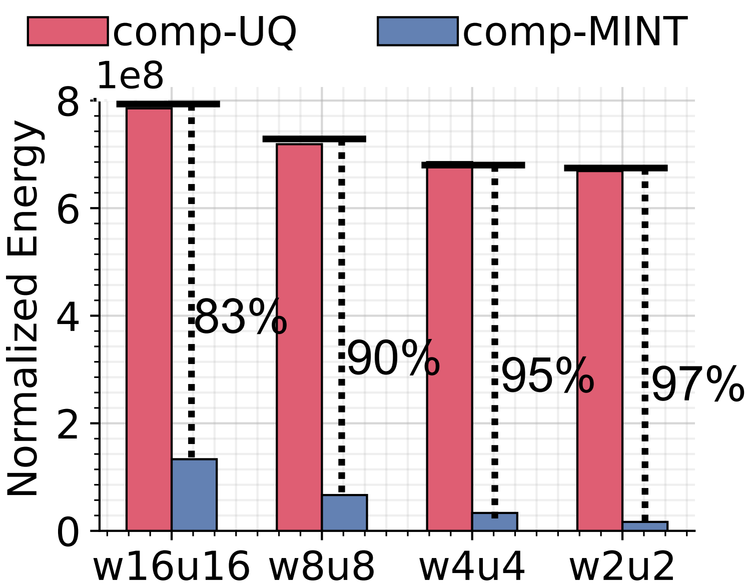


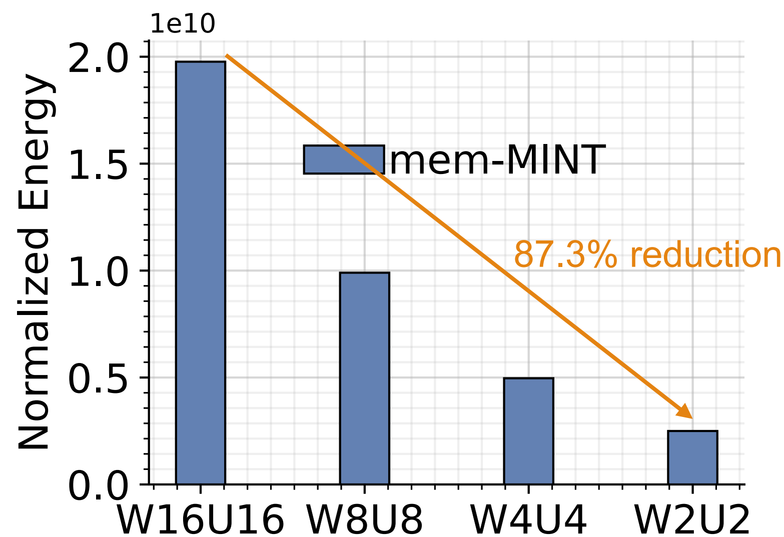
Yin R, et al. MINT: Multiplier-less Integer Quantization for Spiking Neural Networks[J]. arXiv preprint arXiv:2305.09850, 2023.

Experimental Results

- 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

Experimental Results

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

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

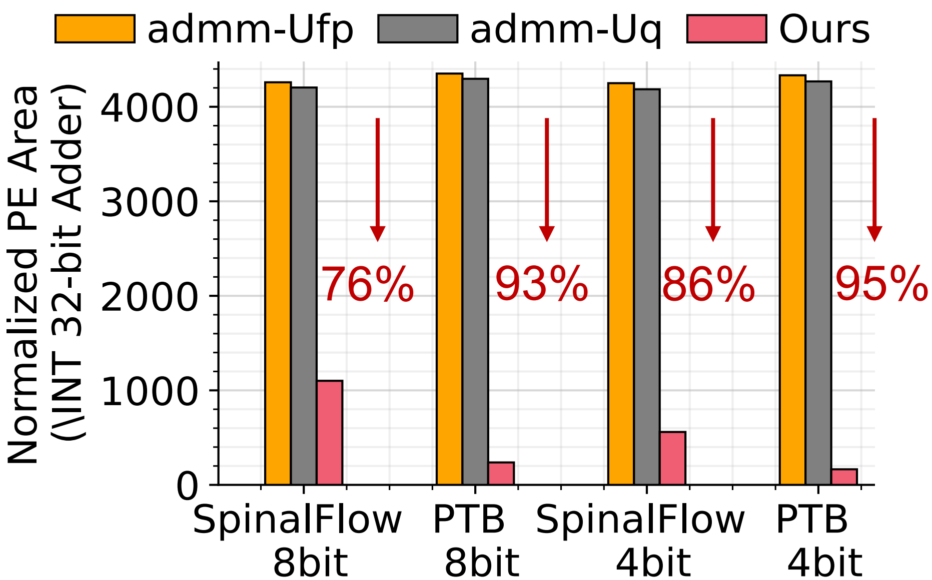
Deng, Lei, et al. "Comprehensive snn compression using admm optimization and activity regularization." (TNNLS). IEEE, 2021.

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.

Experimental Results

- 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.

Thank you!

Code is available at: https://github.com/RuokaiYin/MINT_Quantization/

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?

References

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.