A Low-Bitwidth Integer-STBP Algorithm for Efficient Training and Inference of Spiking Neural Network

Pai-Yu Tan and Cheng-Wen Wu

Department of Electrical Engineering, NTHU, Hsinchu, Taiwan

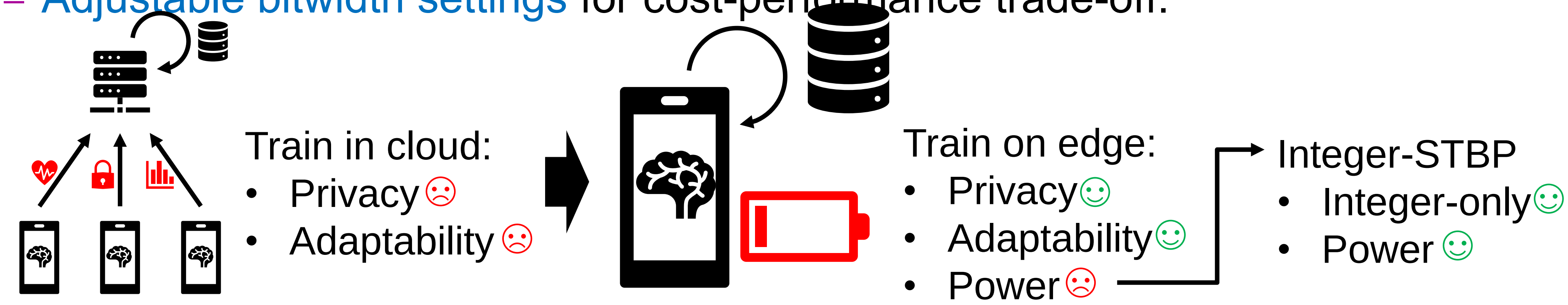
Presenter: Pai-Yu Tan

Outline

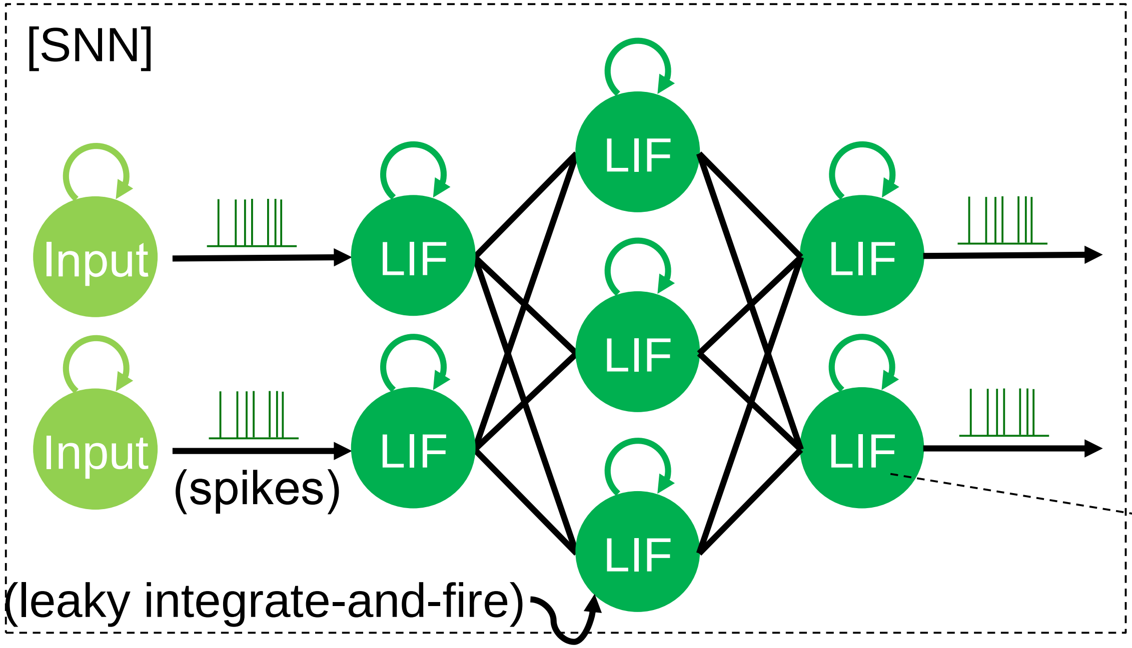
- Introduction & Motivation
- SNN & STBP algorithm
- Proposed Low-Bitwidth Integer-STBP Algorithm
- Experimental Results
- Conclusion and Future Work

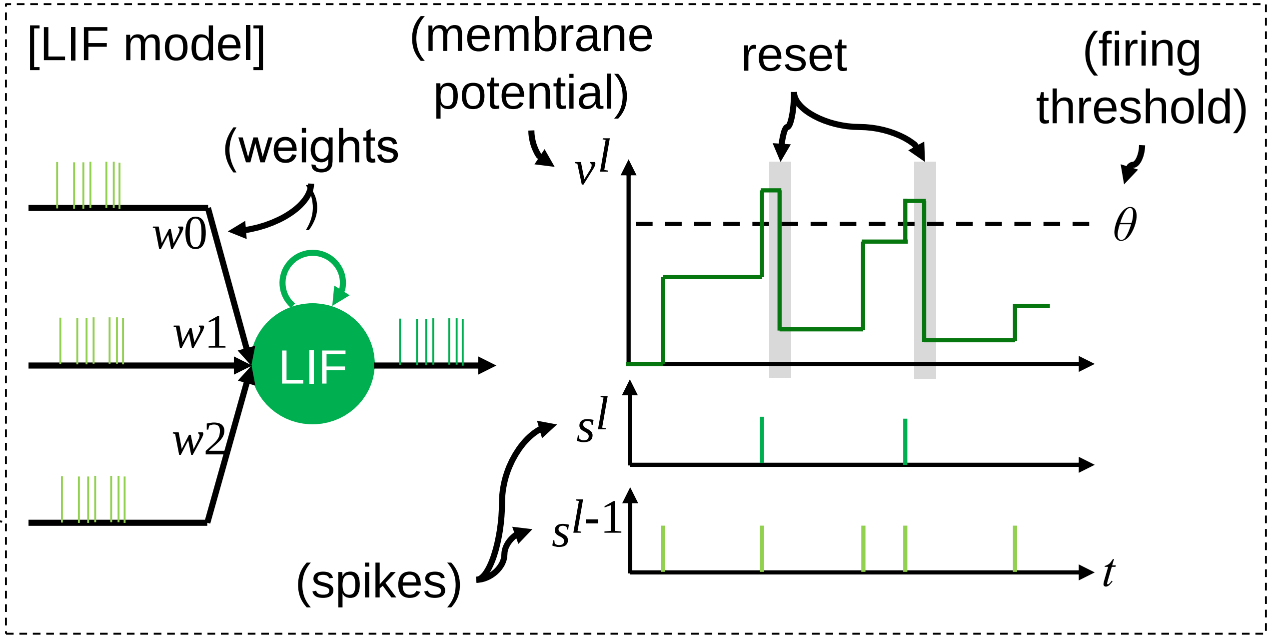
Introduction & Motivation

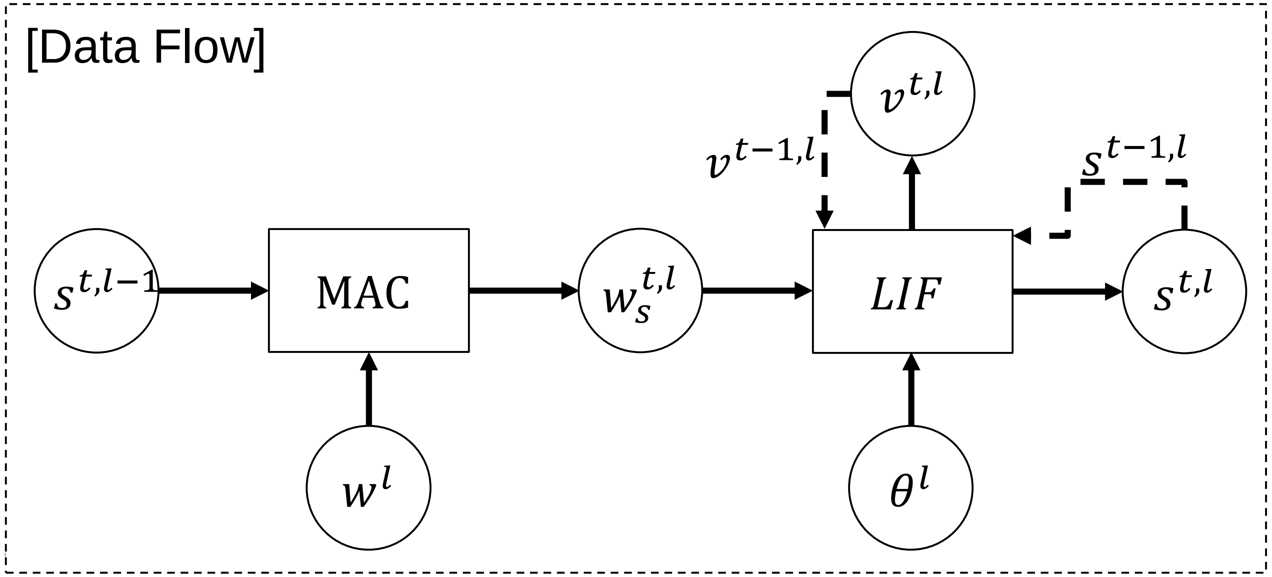
- Spiking neural network (SNN) enables energy-efficient neuromorphic hardware.
- Inference-only devices cannot adapt to diverse environment.
  – Need to train on end-point devices with local dataset.
- Training requires high-performance computing power.
  – Not suitable for edge devices with constrained power budgets.
- A low-bitwidth Integer-STBP algorithm is proposed.
  – Training with only integer operations.
  – Adjustable bitwidth settings for cost-performance trade-off.



Spiking Neural Network (SNN)





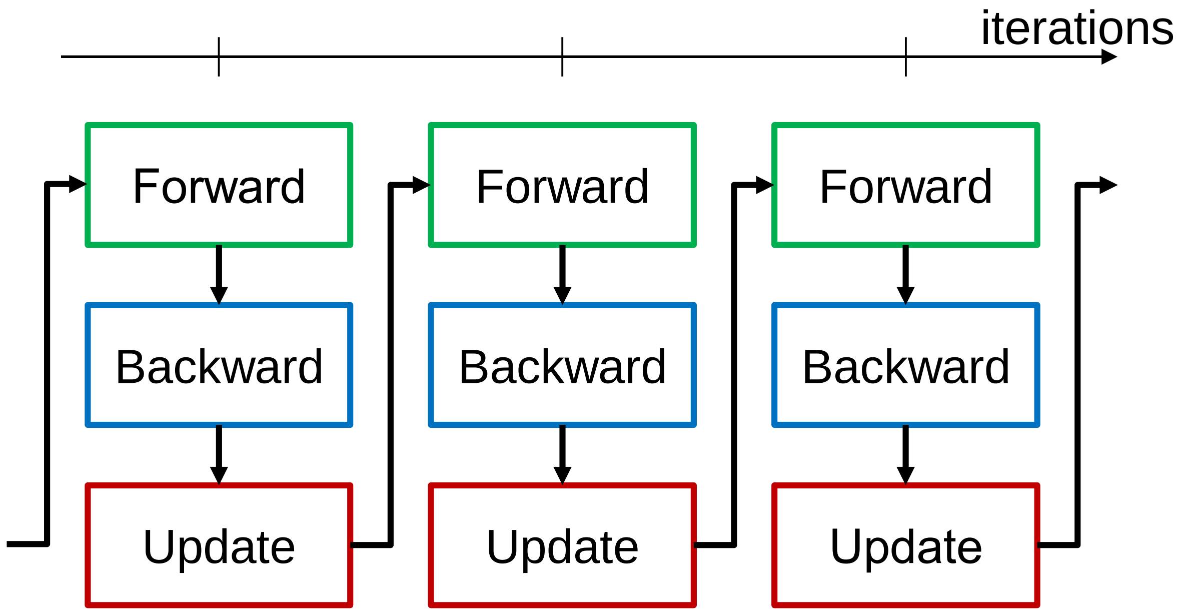


[Equations]

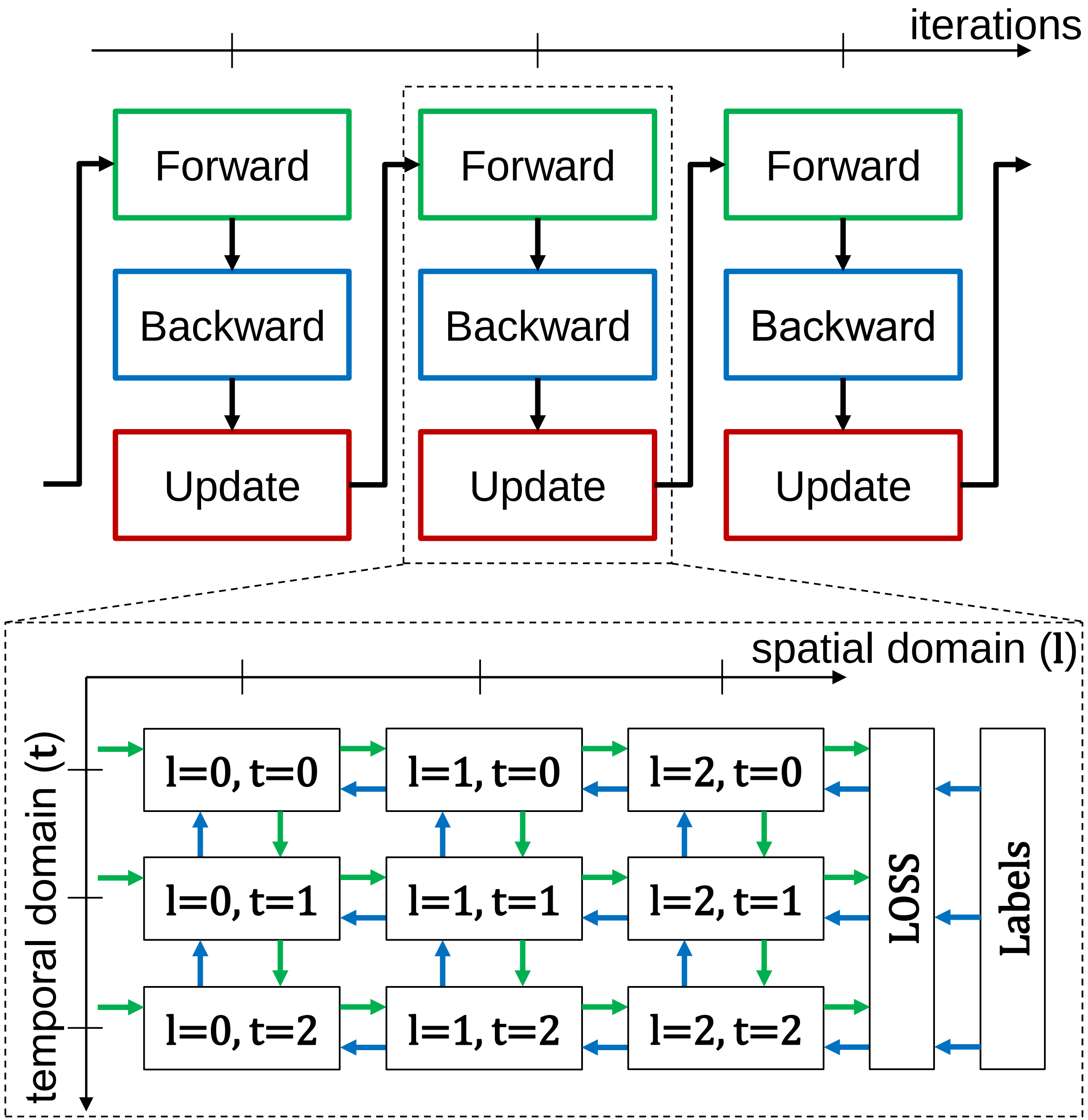
MAC
- $w_s^{t,l} = \sum_{i=1}^{M^{l-1}} s_i^{t,l-1} w_i^l$

LIF
- $v^{t,l} = v^{t-1,l} - s^{t-1,l}\theta^l + w_s^{t,l}$
- $s^{t,l} = \begin{cases} 1 & \text{, if } v^{t,l} \geq \theta^l \\ 0 & \text{, otherwise} \end{cases}$

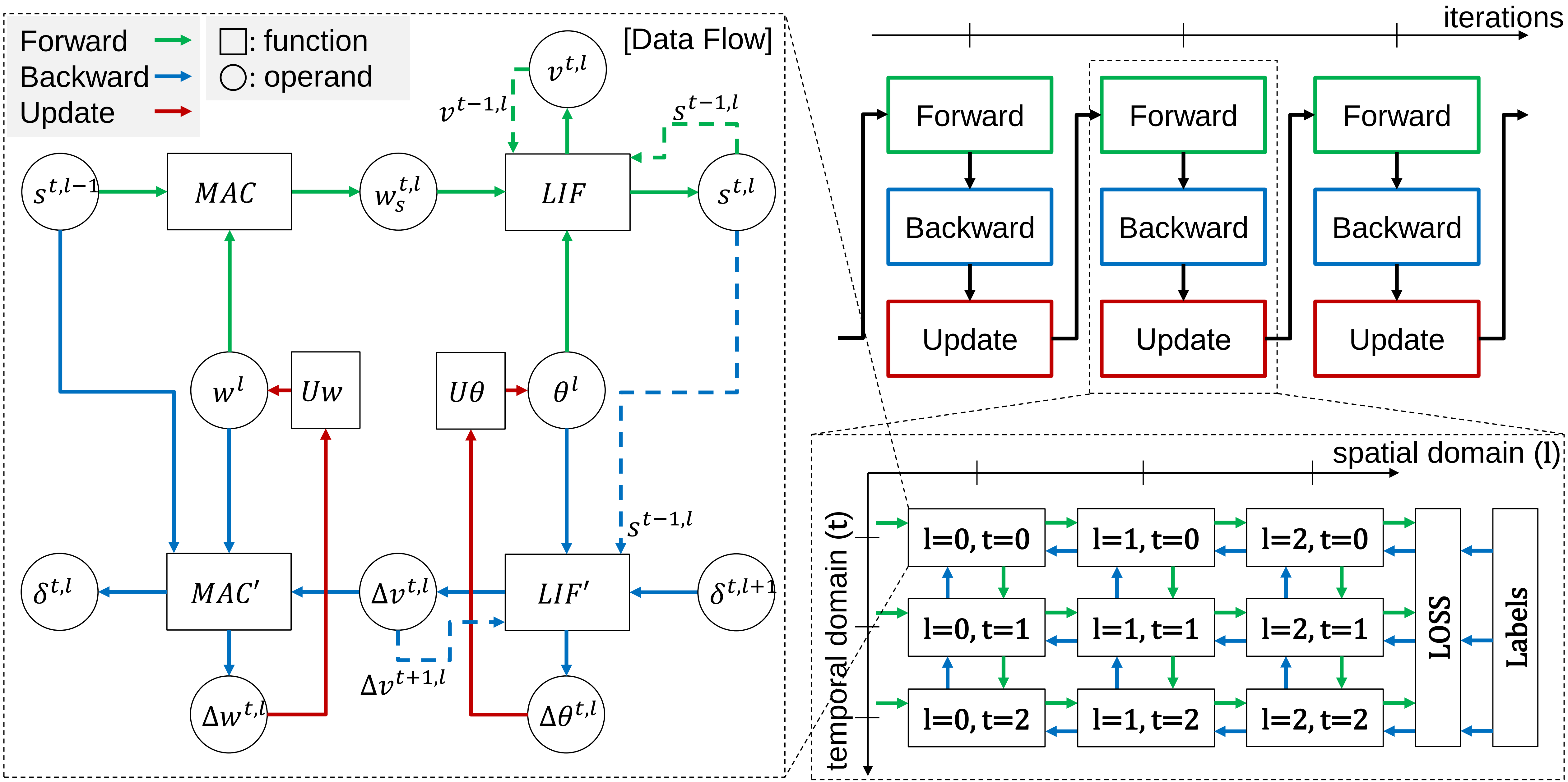
Spatio-Temporal Back-Propagation (STBP)



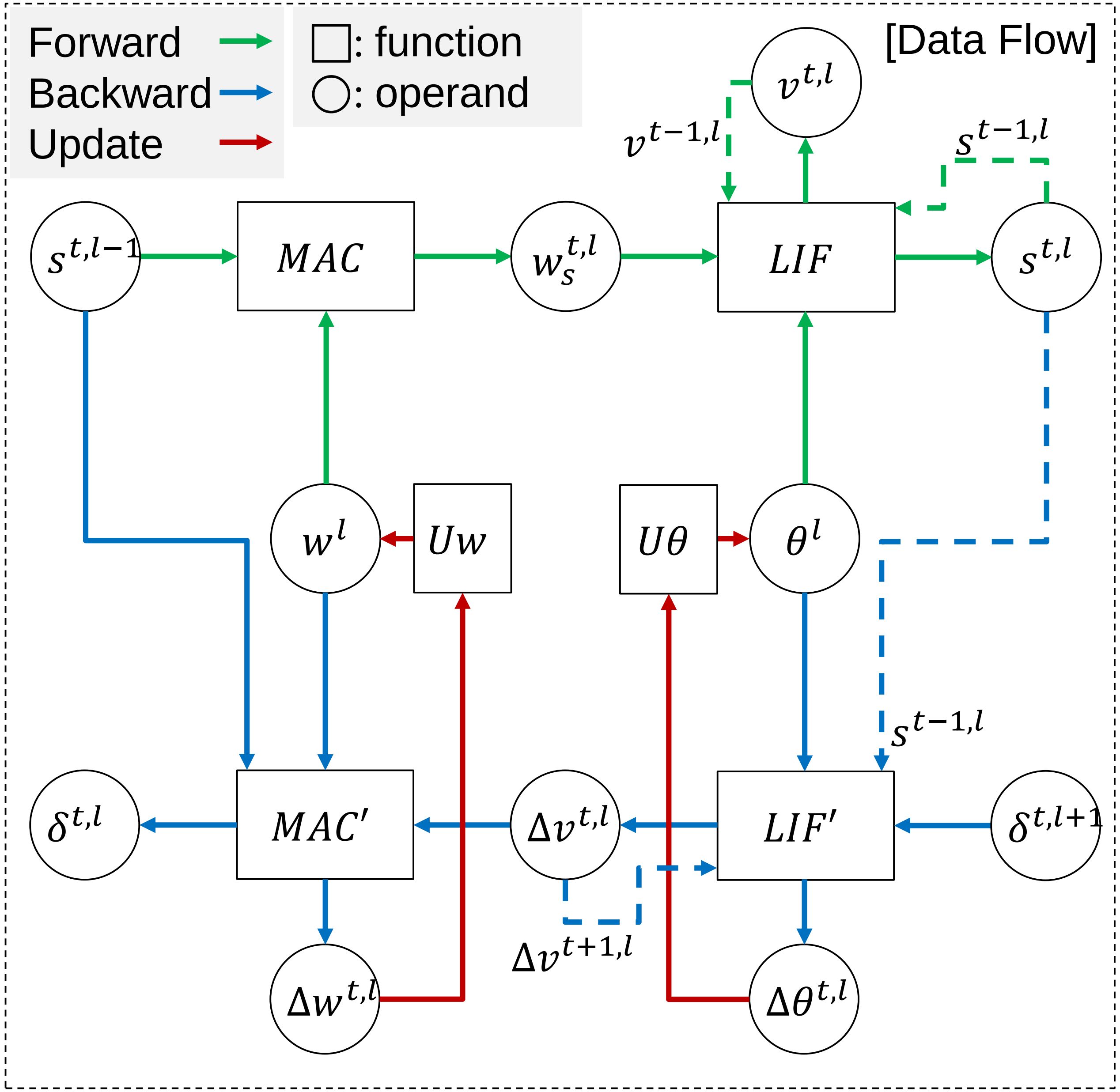
Spatio-Temporal Back-Propagation (STBP)



Spatio-Temporal Back-Propagation (STBP)



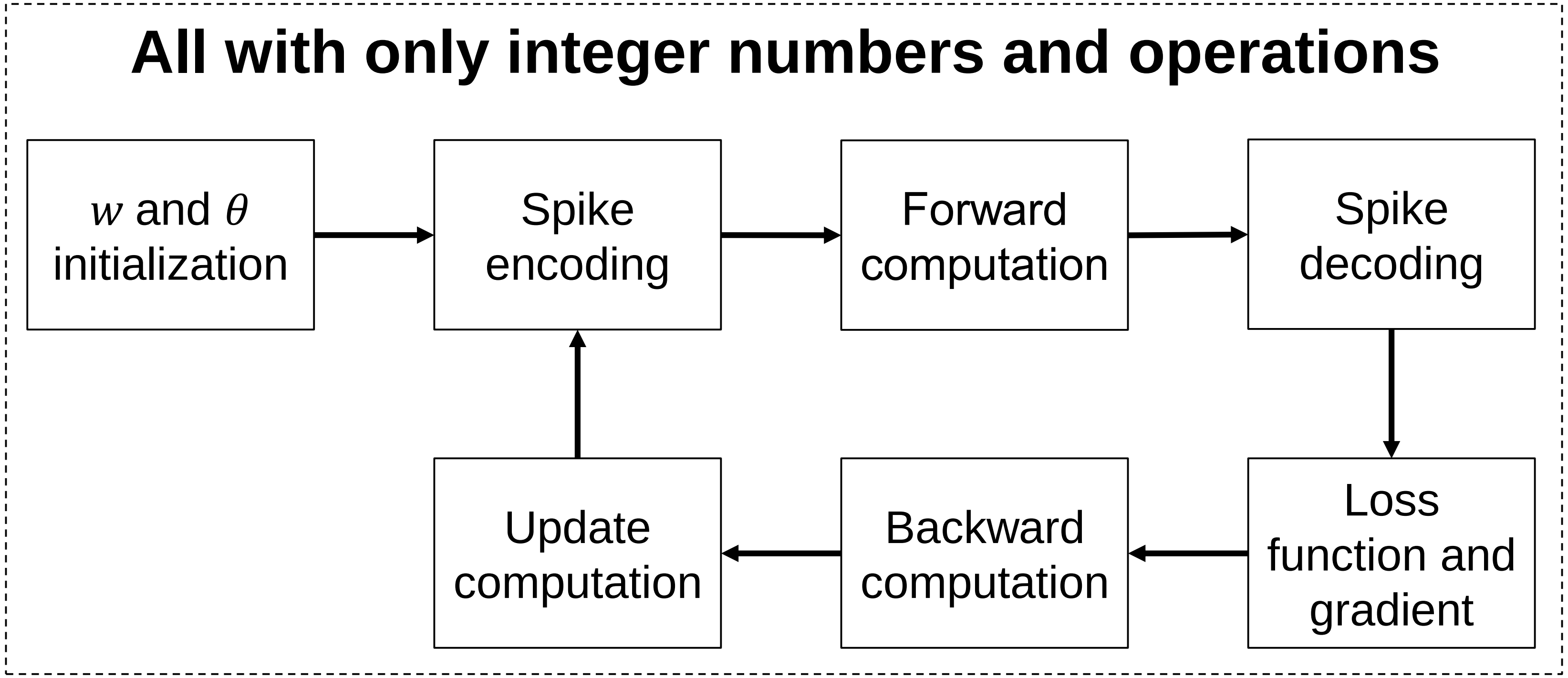
High-Precision STBP Algorithm



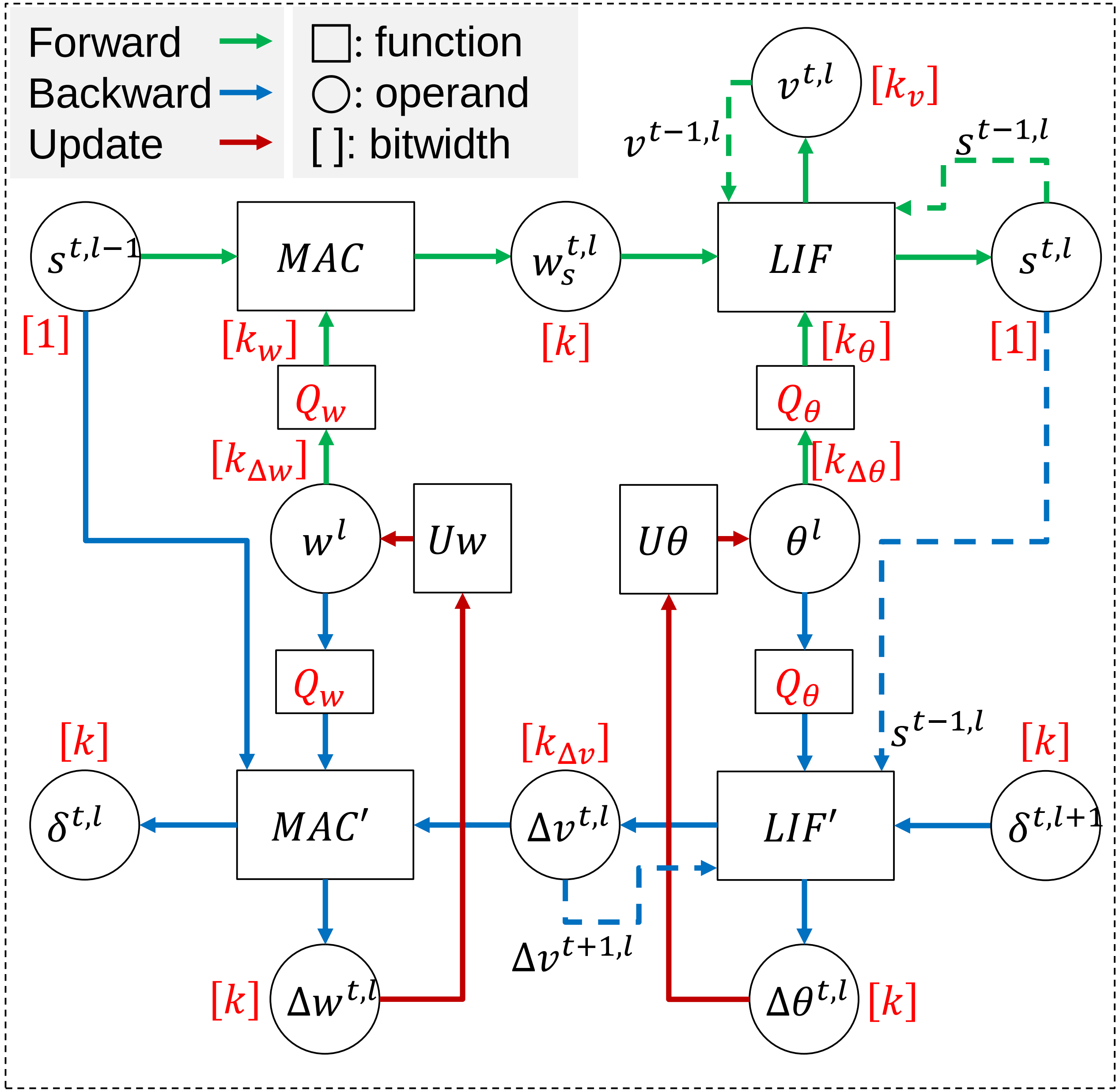
- Operands are floating-point numbers.
- Functions require floating-point operations.
- STBP requires high-precision numbers and operations.

Proposed Integer-STBP Algorithm

- An initialization method for integer weights (w) and firing thresholds (θ)
- Forward, backward, and update computations with only integer operations
- Spike encoding and decoding schemes for effective integer training
- Loss function and gradient with only integer operations



Overview of Integer-Based Data Flow

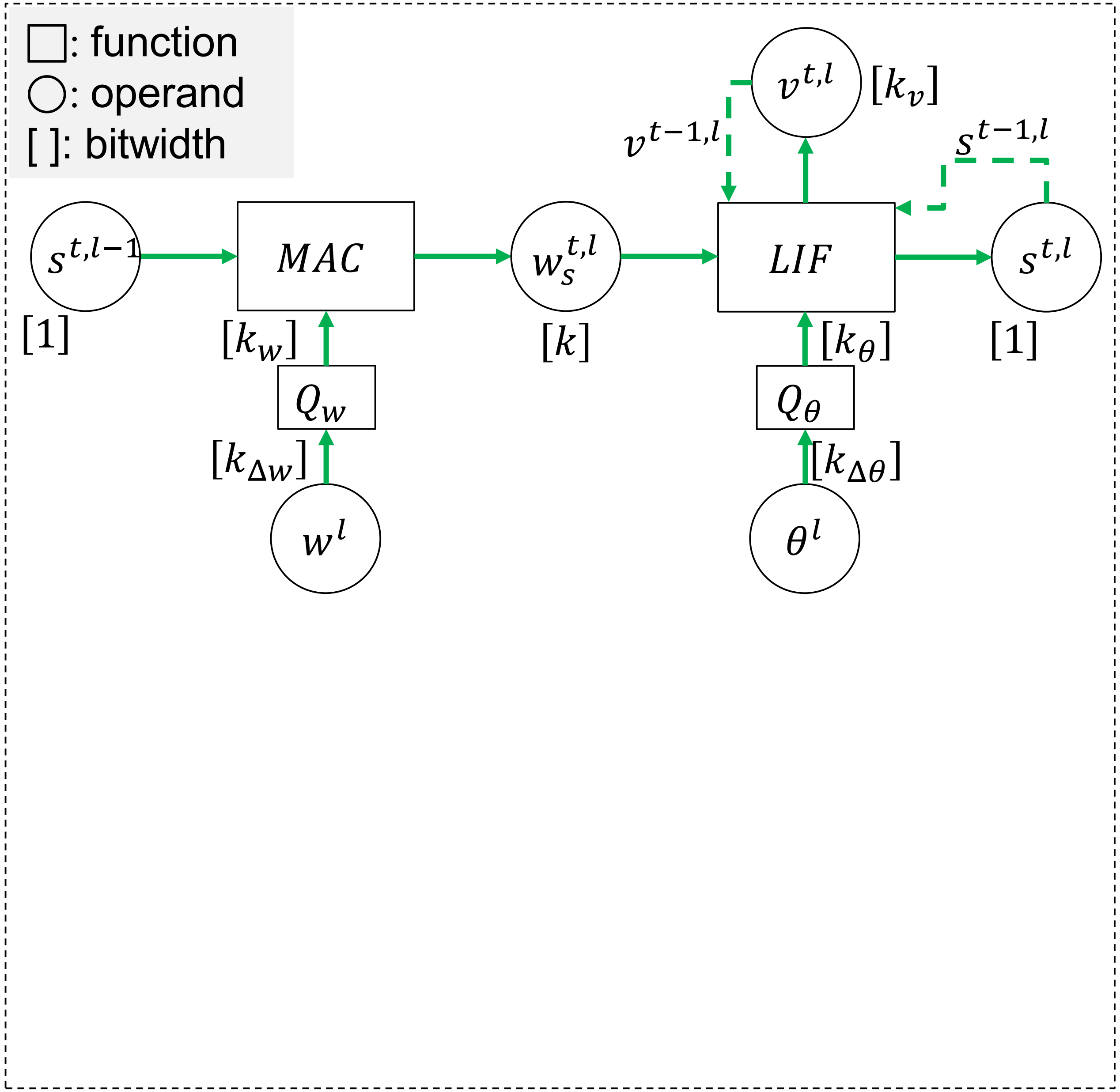


- Operands are with different bitwidths.

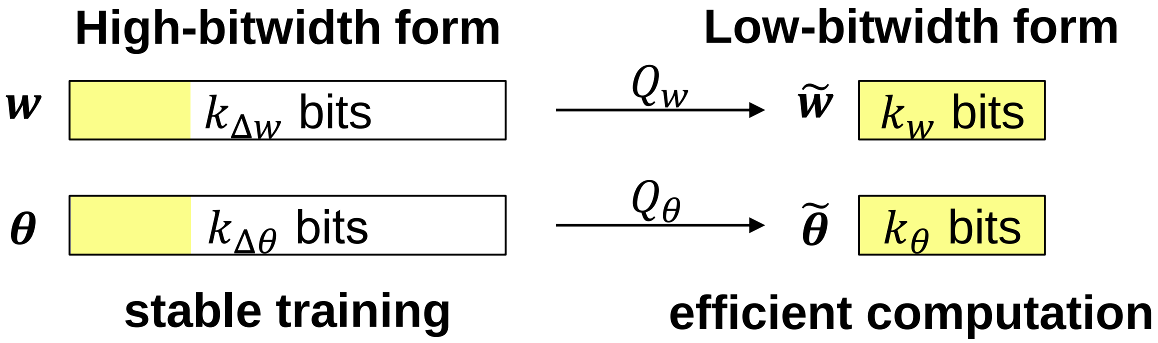
operand	w		θ		v	Δv	other
bitwidth	k_w	$k_{\Delta w}$	k_θ	$k_{\Delta\theta}$	k_v	$k_{\Delta v}$	k

- w and θ are with two different bitwidths.
  - High-bitwidth ($k_{\Delta w}, k_{\Delta\theta}$) for stable training.
  - Low-bitwidth (k_w, k_θ) for efficient computation.
- Functions are with integer operations.
  - Addition, subtraction, multiplication, bit-shift.

Integer-Based Forward Computation

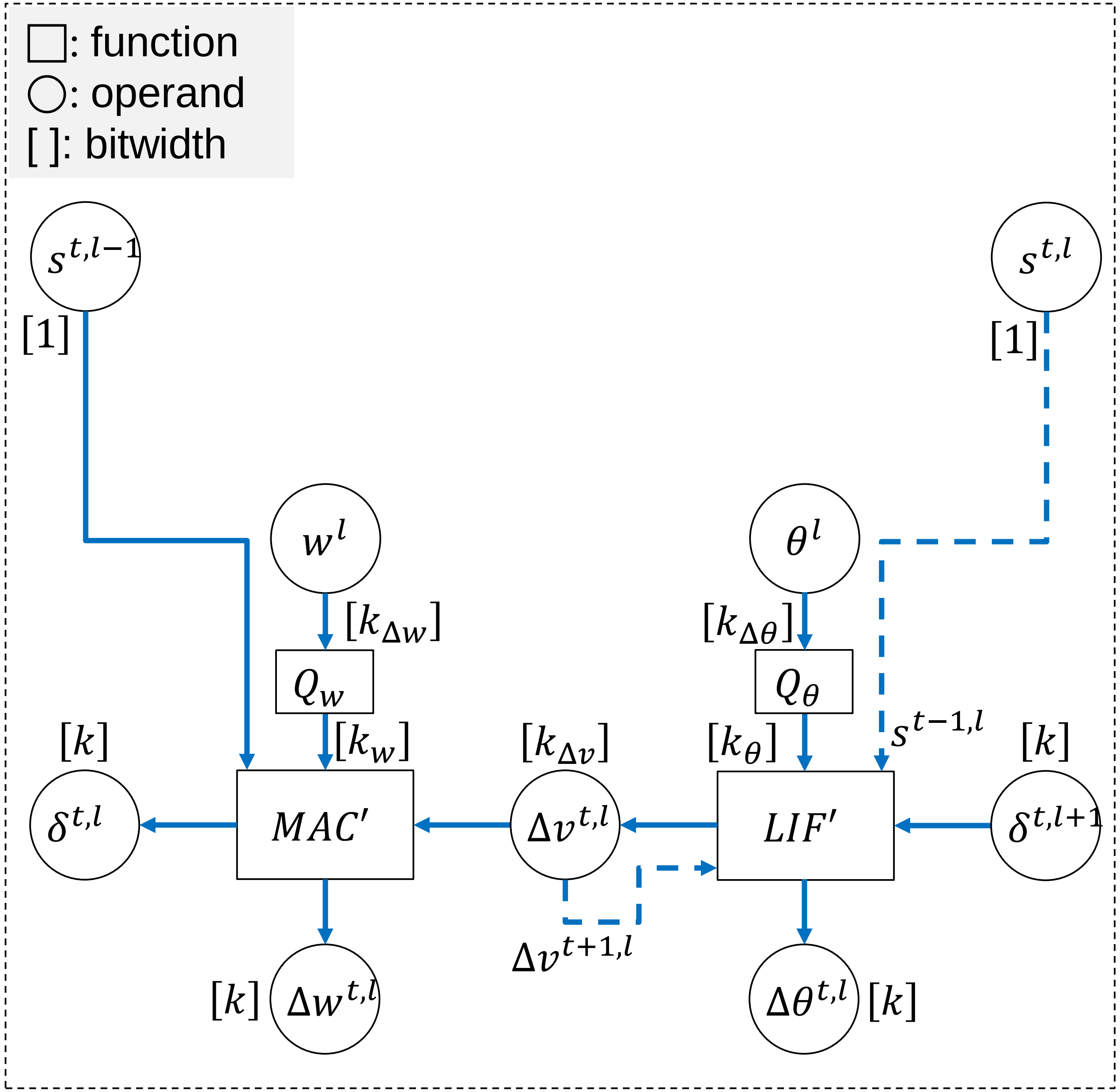


- Q_w and Q_θ convert low to high bitwidth forms for w and θ
  - Take left-most bits



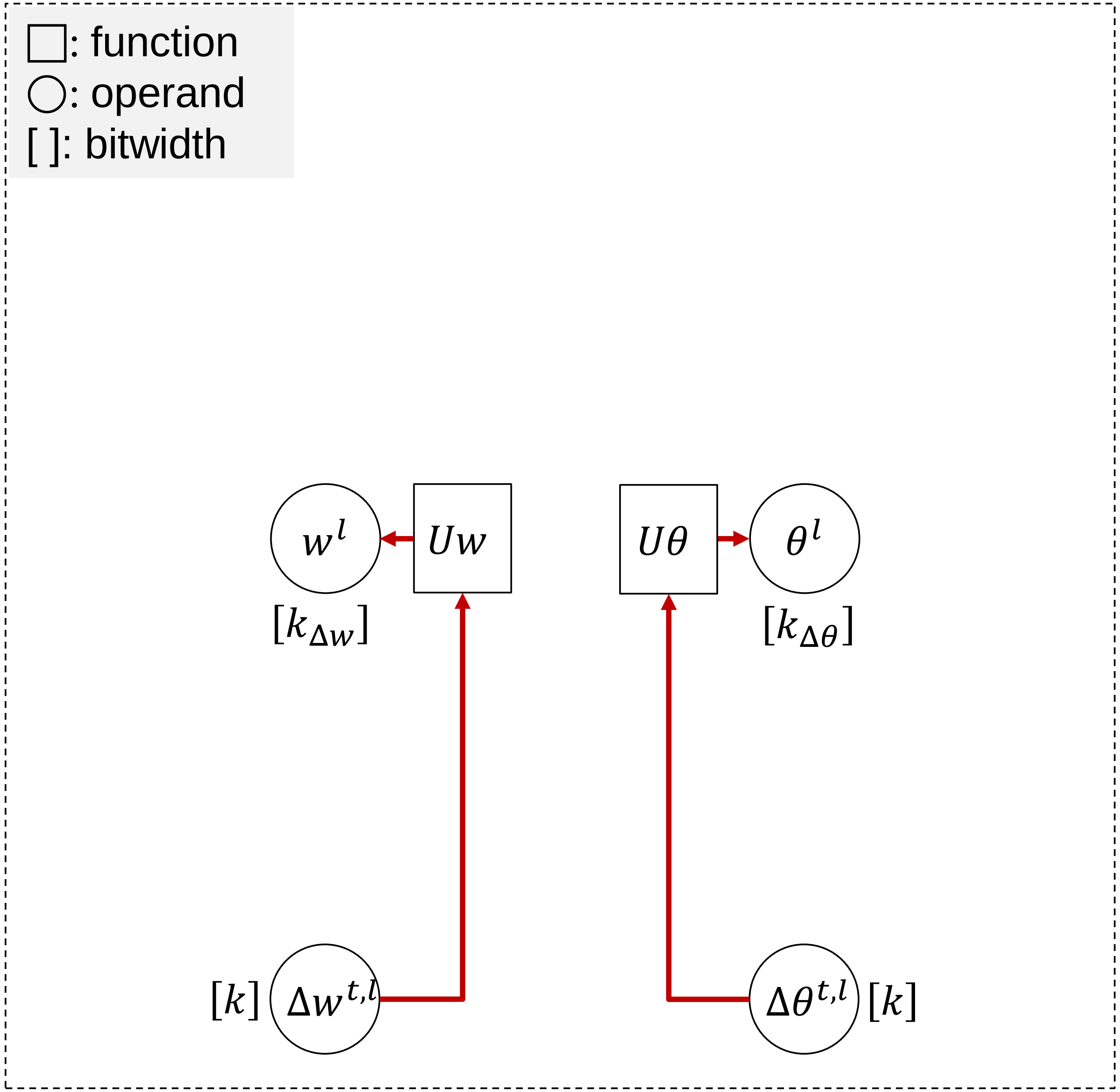
- MAC and LIF compute with $\tilde{w}$ and $\tilde{\theta}$.
  - $\tilde{w}$ and $\tilde{\theta}$ for efficient computation.
- LIF ensures $v \in [-2^{k_v-1} - 1, 2^{k_v-1} - 1]$.

Integer-Based Backward Computation



- MAC' and LIF' compute with $\tilde{w}$ and $\tilde{\theta}$.
  - $\tilde{w}$ and $\tilde{\theta}$ for efficient computation.
- A look-up table-based method computes $\delta^{t,l+1}/\tilde{\theta}^l$ in LIF' function.
  - $x \div y = x \times \frac{1}{y} = x \times \frac{2^k}{y} \div 2^k$
  - $\frac{2^k}{y}$ by a look-up table, i.e., $LUT(y) = \frac{2^k}{y}$.
  - $\div\, 2^k$ by the bit-shift operation, i.e., $\gg k$.
  - Division → LUT + multiply + bit-shift

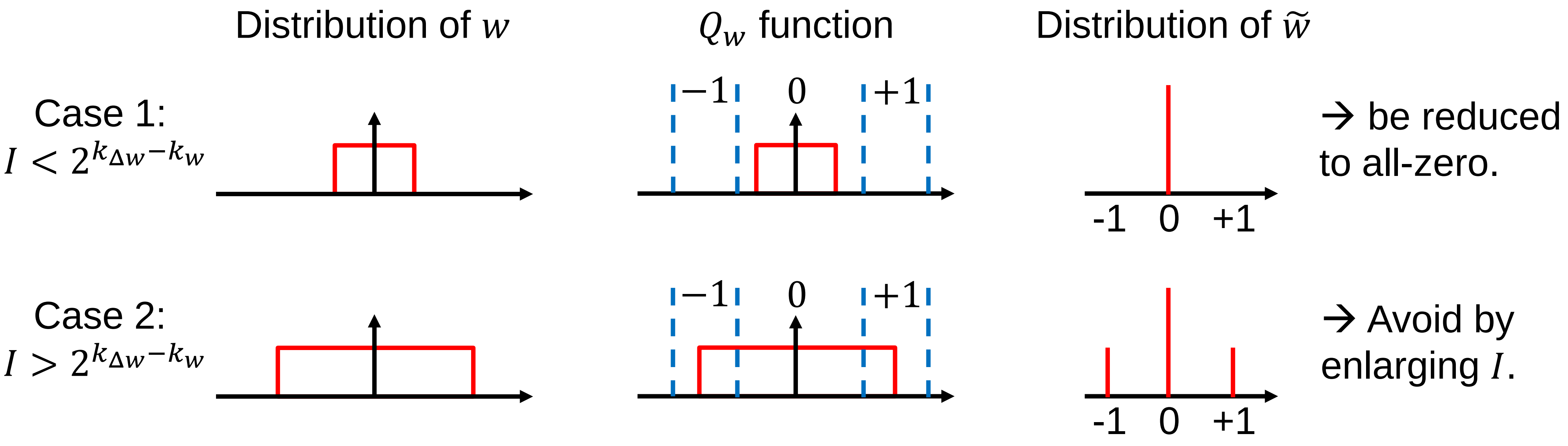
Integer-Based Update Computation



- Uw and $U\theta$ compute with w and θ.
  - High-bitwidth w and θ for stable training.
- Learning rate is set to 2^η
  - $\times 2^\eta$ by bit-shift operation, i.e., $\ll \eta$
- A value boundary (α) limits the range of Δw and $\Delta\theta$ for each update.
  - Low-bitwidth SNN is very sensitive to the variations after updating w and θ.

Initialization Method for Integer w and θ

- Modify Xavier initialization method to get initial integer w
  - Generate w from the uniform distribution $U(-I, +I)$.
- A constant (β) limits the range of $U(-I, +I)$, i.e., minimum $I = \beta \times 2^{k_{\Delta w} - k_w}$.
  - By setting $\beta > 1$, the low-bitwidth form ($\tilde{w}$) avoids being all-zero.
- When using minimum I, firing activity will be higher than expected.
  - Scale up θ accordingly.



Overall Training Implementation

- Spike encoding: spike-rate encoding
- Spike decoding: last-τ-tick decoding
  - Count the number of spikes in the last τ ticks
- Loss function: sum squared error (SSE)
  - Only integer operations for forward and backward.
- Enlarge input loss gradient by a constant γ
  - To avoid gradient vanishing after $\delta^{l+1}/\tilde{\theta}^{l}$.
- Optimizer: mini-batch gradient descent
  - No momentum optimizer, dropout, L2 regularization.

Spike-Rate Encoding

102 → 0 1 0 1 0

154 → 0 1 1 0 1

(value) (in spikes)

Last-τ-Tick Decoding

0 1 0 1 0 → 1

(τ=3)

0 1 1 0 1 → 2

(out spikes) (value)

Experimental Setup

- We use the network similar to that of the original STBP work.
  - Average pooling layer is merged into the preceding layer by a stride of 2.

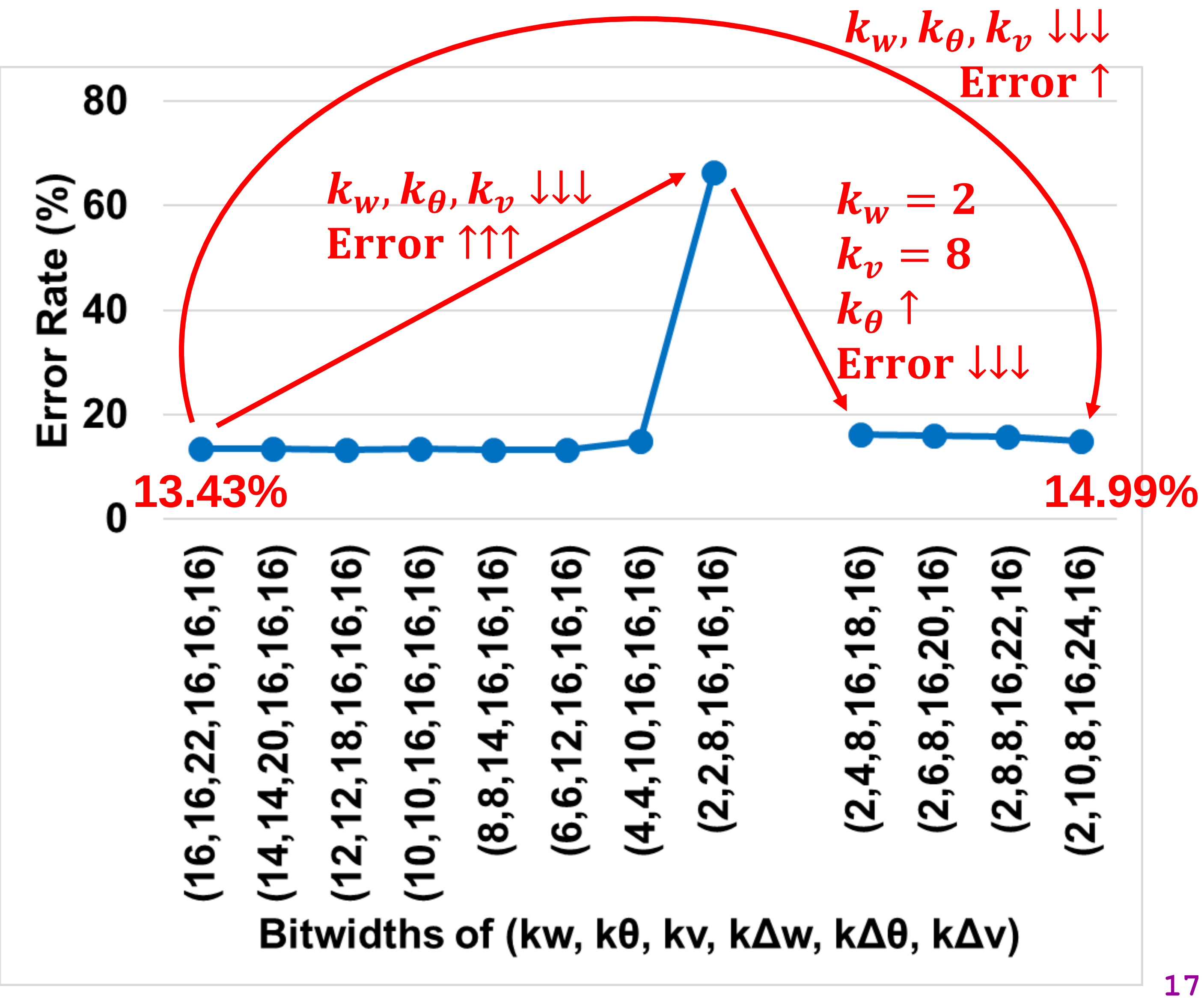
	Network Structure
in prev. work	Encoding–96C3–256C3–AP2–384C3–AP2–384C3–256C3–1024FC–1020FC–Decoding
in this work	Encoding–96C3–256C3S2–384C3S2–384C3–256C3–1024FC–1020FC–Decoding

- Evaluate on CIFAR10 dataset (50,000 training data and 10,000 testing data)
- Hyper-parameter settings if not specify:

batch size	total time steps	initialization constant (β)	Last-τ-tick decoding (τ)	θ for encoding neurons
100	8	1.5	5	256

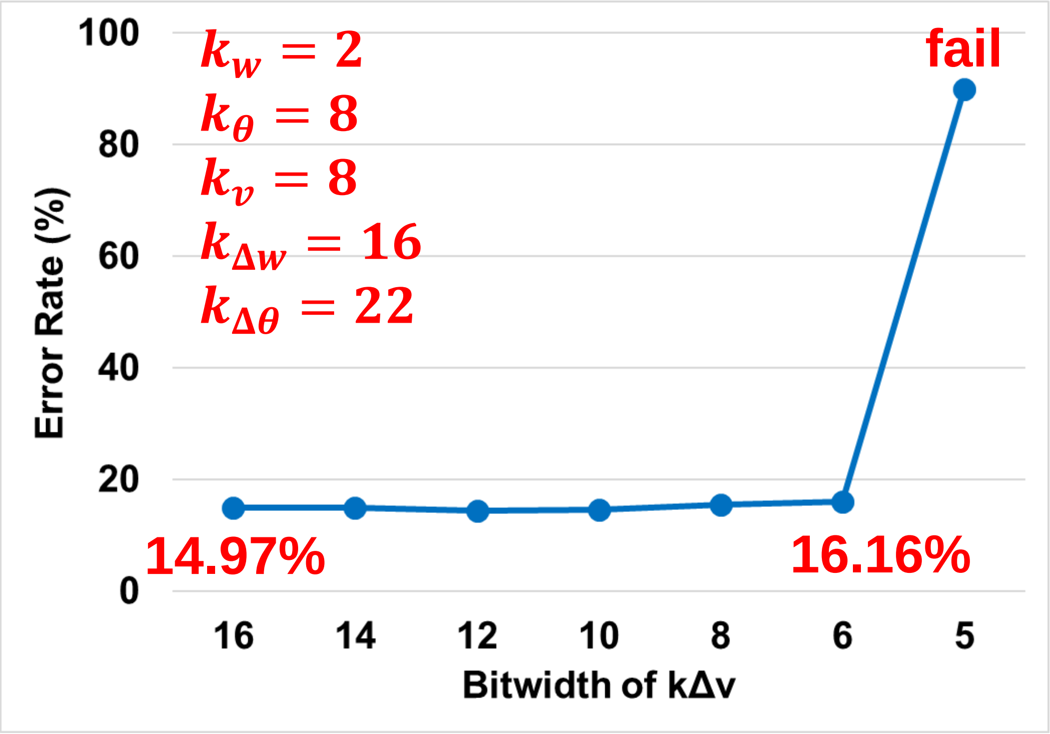
Evaluation of Bitwidths of k_w, k_θ, k_v

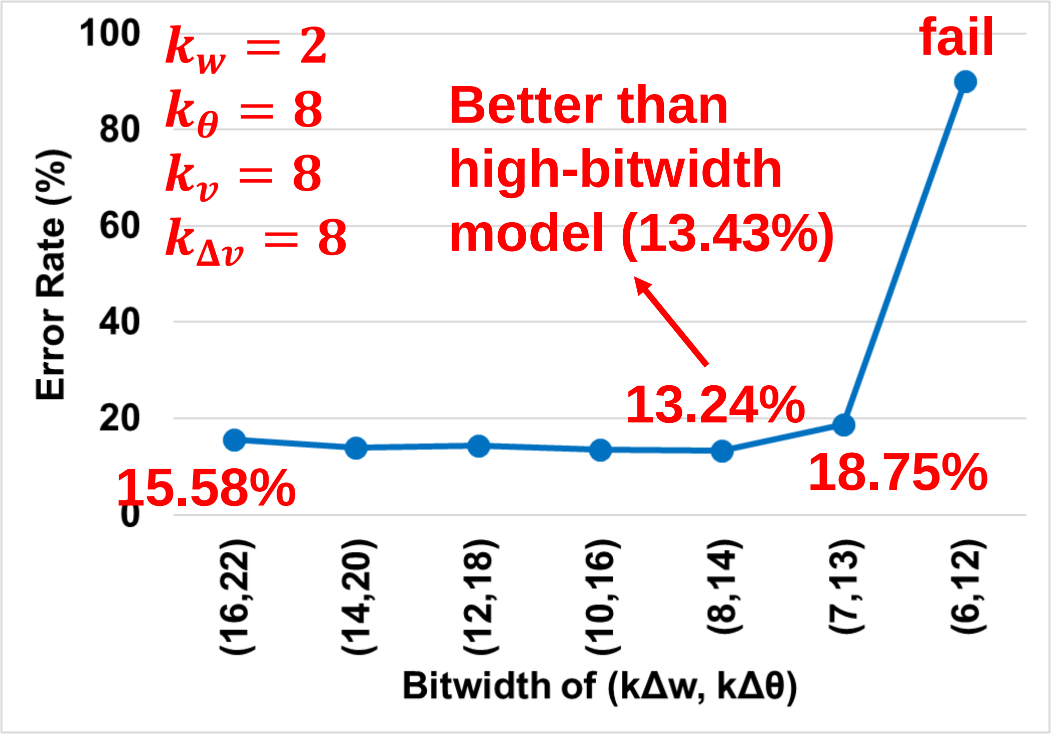
- k_w, k_θ, and k_v can be significantly reduced with negligible accuracy degradation.
  - k_w, k_θ, k_v = 2, 10, 8 bits.
  - $k_{\Delta w}$, $k_{\Delta\theta}$, $k_{\Delta v} \geq$ 16 bits.
- Low-bitwidth firing threshold (θ) cannot control the firing activity.
- Note: bitwidth setting should follows $k_{\Delta w} - k_w = k_{\Delta\theta} - k_\theta$
  - Ensure $\Delta w, \Delta\theta$ affect equally to w, θ



Evaluation of Bitwidths of $k_{\Delta w}, k_{\Delta\theta}, k_{\Delta v}$

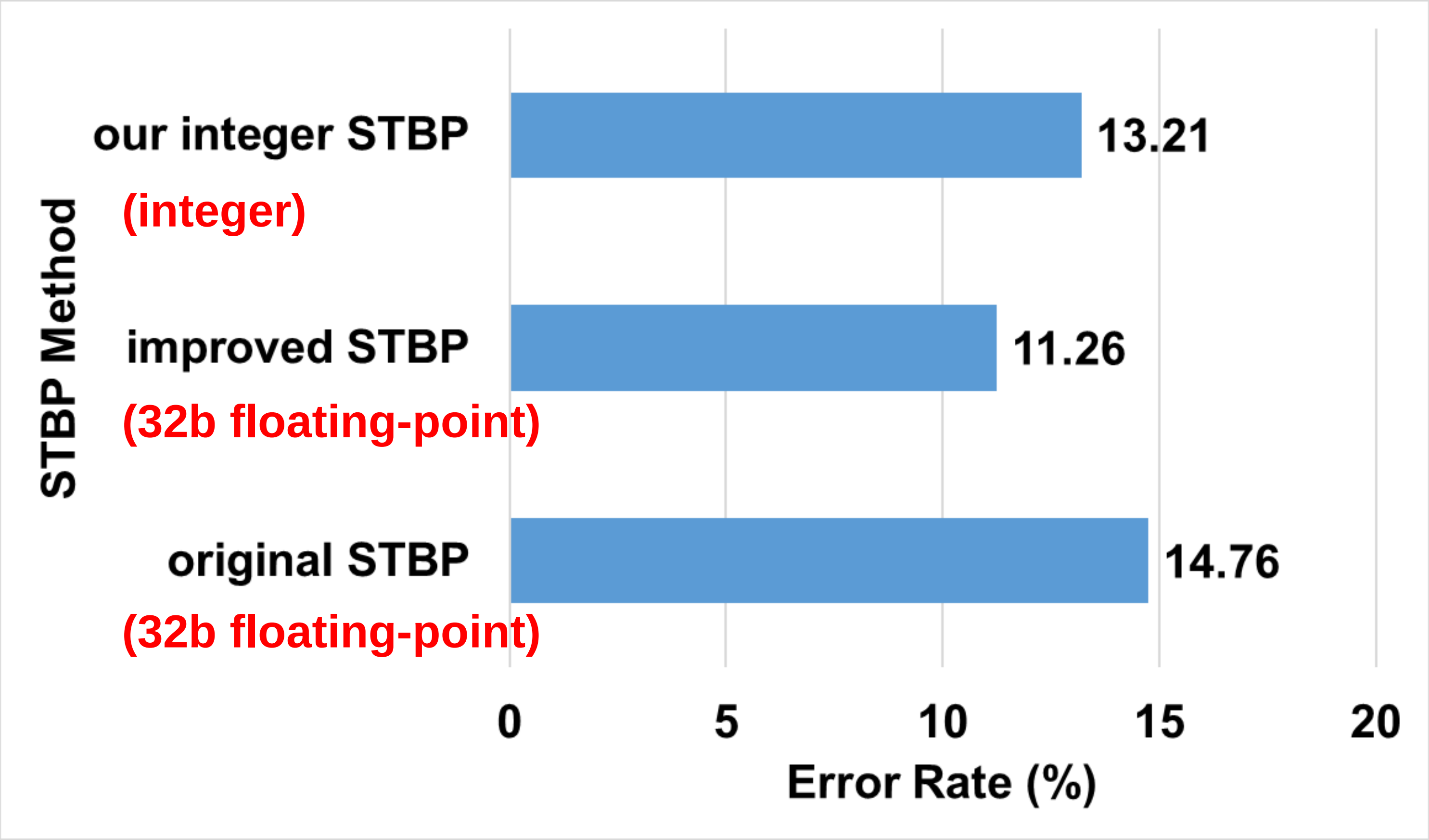
- If $k_{\Delta v}$ is too small → gradient vanishing after $\delta/\tilde{\theta}$
  - Least $k_{\Delta v} = 6\ bits$
- When $k_{\Delta w}, k_{\Delta\theta}$ become close to k_w, k_θ → w, θ update variations become large
  - Least $k_{\Delta w} = 7\ bits,\ k_{\Delta\theta} = 13\ bits\ (k_{\Delta w} - k_w = k_{\Delta\theta} - k_\theta = 5\ bits)$
- Limited precision acts as a type of regularization.





Comparison to Previous STBP Methods

- Accuracy: original floating-point < proposed integer < improved floating-point.
- Proposed low-bitwidth model achieves comparable accuracy.



Conclusion and Future Work

- An Integer-STBP algorithm is proposed for fully-integer training.
  - Training without floating-point operations.
- An extremely low-bitwidth integer model achieves comparable accuracy with the floating-point model.
  - Be more energy-efficient with negligible accuracy loss.
- Future work
  - Evaluating proposed approaches on larger SNNs and dataset.

THANKS FOR YOUR ATTENTION

Q&A