FASTSNN: ACCURATE POST-TRAINING COMPRESSION FOR ACCELERATING SPIKING NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Spiking neural networks (SNNs) have shown promising performance on a wide range of perception tasks, making them attractive for edge deployment. However, these powerful SNNs require massive memory and computational resources, often exceeding the tight budgets of edge devices. Model compression is therefore essential, and post-training compression (PTC), which efficiently compresses a pretrained model without retraining, is particularly appealing. Existing PTC methods for SNNs minimize the local reconstruction error of each layer's signals, but this objective overlooks the discrete and temporal nature of spike transmission. In SNNs, a weight perturbation introduced by compression alters the transmitted signal only through spike flips, i.e., a spike is either inserted or deleted. We find that model performance after PTC depends less on how many flips perturbations cause than on whether flips are balanced. Building on this flip-balance perspective, we propose FastSNN, a PTC framework that jointly accounts for signal reconstruction and flip balance during SNN compression. By modeling the relation between weight perturbations and spike flips in a unified form, FastSNN applies to a broad range of compression types, including pruning and quantization. Across multiple datasets and SNN models, FastSNN outperforms the state-of-the-art method by 9.09% on average under 87.5% unstructured pruning and delivers up to a 4.2× inference speedup on a prototype neuromorphic chip.

1 INTRODUCTION

Spiking neural networks (SNNs) have shown promising performance in wireless sensing (Lu et al., 2026; Yu et al., 2026a), object detection (Luo et al., 2024), and speech recognition (Wang et al., 2026a). The feasibility of deploying high-performance SNNs on mobile and edge devices (Yu et al., 2026b; Davies et al., 2018) further motivates their deployment for these applications. However, these devices operate under tight memory and power budgets, and many applications also require low-latency inference. In contrast, the SNNs (Luo et al., 2024; Zhou et al., 2022) that reach this level of performance have grown deeper and wider with increased memory access and computation. The resulting gap between model cost and device capacity makes direct deployment difficult.

Model compression provides an effective means of meeting these deployment requirements by reducing model size and computational cost while preserving model performance. Many existing SNN compression approaches are integrated into training (Deng et al., 2021; Ye et al., 2025; Shi et al., 2026; She et al., 2026), requiring access to the training pipeline and additional optimization for different compression configurations. More importantly, such training is computationally expensive and time-consuming. Since every new compression target requires another full training run, these methods cannot keep up with the requirements of quick and massive deployment on various devices.

Post-training compression (PTC) offers an effective alternative: it compresses a pretrained model using a small calibration set, without retraining. A common approach is layerwise reconstruction (Frantar & Alistarh, 2022), which compresses the network one layer at a time, achieving great success (Frantar & Alistarh, 2022; Frantar et al., 2022) in compressing conventional deep neural networks (DNNs). Recent work has begun to bring layerwise PTC to SNNs (Li et al., 2025; Shi et al., 2025). Methods following this paradigm optimize compressed weights by minimizing local

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

membrane-potential reconstruction loss. However, directly applying such layerwise PTC to SNNs introduces a mismatch between the reconstruction objective and discrete spikes (Shi et al., 2025).

Specifically, SNNs transmit discrete spikes between layers, so weight perturbations from compression affect these signals in the form of spike flips: insertions ($0 \to 1$) or deletions ($1 \to 0$). Ideally, compression should preserve every spike to keep transmitted signals lossless. However, when applying those PTC approaches (Frantar & Alistarh, 2022; Shi et al., 2025) to SNNs, even a tiny reconstruction error can cause a flip when the original membrane potential lies close to the firing threshold. These flips can also propagate through subsequent layers, making complete spike preservation increasingly difficult. Moreover, the results in Figure 1(a) suggest that the number of flips alone is insufficient to explain the observed performance gap. Actually, not every flip needs to be eliminated to preserve the final prediction. Consider, for instance, an SNN with integrate-and-fire (IF) neurons that averages its final output over time. Deleting a spike at one timestep and inserting another at another timestep changes a neuron's spike timing but preserves its total count. If this count is preserved up to the final layer, the final output, i.e., the temporally averaged representation, remains exactly unchanged. These observations motivate us to focus on flip balance: when flips are unavoidable, the objective should favor insertions and deletions whose effects cancel out over time.

Based on these observations, we introduce FastSNN, a unified PTC framework for SNNs. FastSNN seeks to balance the effects of flips rather than eliminate every flip. However, balance within an intermediate layer does not necessarily translate into balance at the output, as its downstream effects depend on neuronal dynamics and model architecture. Notably, for models with a temporally averaged readout, preserving spike counts of the last layer provides a clear target: the model output remains unchanged. We therefore use the sensitivity to the final balance loss to guide reconstruction in intermediate layers, encouraging perturbations whose effects cancel before reaching the output. Our contributions are summarized as follows:

- We introduce a flip-balance perspective on PTC for SNNs. Our analysis and experiments show that the number of flips alone does not explain the damage caused by compression, and flip balance plays a crucial role, which motivates us to treat balance as an objective along with minimizing the reconstruction error for better model performance.
- We propose FastSNN, a unified PTC framework for SNNs whose objective combines local reconstruction with flip balance guidance. Applied through a layerwise solver, this guidance keeps the flips of each layer more balanced, and thereby better preserves spike counts within transmitted signals.
- Extensive experiments on multiple datasets and SNN architectures demonstrate substantial accuracy improvements under aggressive compression compared to other PTC methods. For instance, FastSNN achieves average accuracy gains of 7.45% and 9.09% over state-of-the-art methods under 2:8 structured pruning and unstructured pruning with 87.5% weight sparsity, respectively.

2 Related Work

Post-Training Model Compression. PTC modifies pretrained models without retraining, typically using a small calibration set. Optimal Brain Compression (OBC) (Frantar & Alistarh, 2022) unifies pruning and quantization through layerwise reconstruction using second-order Hessian information. SparseGPT (Frantar & Alistarh, 2023) scales this approach to large language models (LLMs) and supports N:M sparsity, while ALPS (Meng et al., 2024a) uses ADMM for high-sparsity pruning. Wanda (Sun et al., 2024) prunes using weight magnitudes and activation norms, whereas OSSCAR (Meng et al., 2024b) formulates structured pruning as combinatorial optimization. For quantization, AdaRound (Hubara et al., 2021), BRECQ (Li et al., 2021), and QDrop (Wei et al., 2022) develop reconstruction-based methods for vision models. GPTQ (Frantar et al., 2022) adapts OBC framework to quantize LLMs, RTN (Dettmers & Zettlemoyer, 2023) provides a calibration-free baseline, and AWQ (Lin et al., 2024) uses activation-aware channel scaling. GuidedQuant (Kim et al., 2025) uses end-loss Fisher information for the layerwise reconstruction. However, applied directly to SNNs, they account for neither the discreteness of spikes nor the dependence among a layer's outputs at different timesteps.

SNN Compression. Most SNN compression methods operate during training. ADMM-based compression (Deng et al., 2021) jointly compresses weights and spiking activity, while gradient rewiring (Chen et al., 2021), dendritic-spine state transitions (Chen et al., 2022), and ESL-

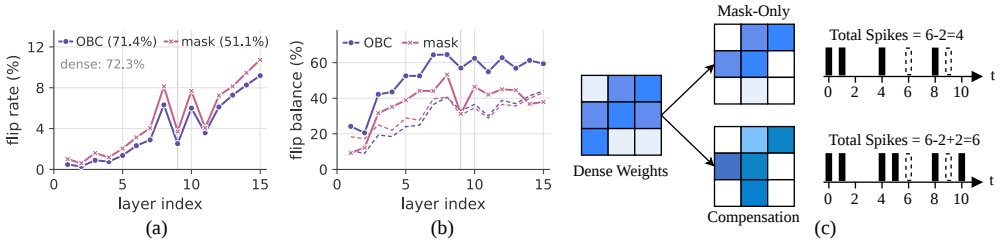


Figure 1: Spike flip profile of SEW-ResNet-DVS with LIF neurons under 2:4 pruning on CIFAR10-DVS. **(a)** Layerwise flip rates. **(b)** Temporal flip balance. **(c)** An illustrative spike train: mask-only pruning deletes two of six spikes; compensation introduces two spikes, restoring the count despite increasing the total number of flips.

SNNs (Shen et al., 2023) learn sparse connectivity. Deployment-oriented approaches include probability-based N:M pruning (Ye et al., 2025), joint weight and spike sparsification in UniSparse (Shi et al., 2026), and channel pruning guided by synaptic-operation cost in Spik4Lite (She et al., 2026). Quantization likewise relies primarily on training. MINT (Yin et al., 2024) shares quantization scales between weights and membrane potentials for multiplier-free inference, Q-SNNs (Wei et al., 2024) quantize both to low bit-widths, and QP-SNN (Wei et al., 2025) jointly performs quantization and pruning.

Post-training SNN compression remains less explored. NeuronQuant (Li et al., 2025) calibrates quantized SNNs without retraining, while Spiking Brain Compression (SBC) (Shi et al., 2025) adapts OBC using a surrogate membrane-potential Hessian that incorporates temporal dynamics. Although both methods tailor their objectives to spiking neurons, they still minimize a local reconstruction error and do not account for the temporal balance of signed spike flips.

3 Methodology

3.1 Background and Problem Definition

SNNs transmit information across layers through sparse spikes. Considering the leaky integrate-and-fire (LIF) neuron, whose neuronal dynamics evolves as:

$$V[t] = \Big(1 - \frac{1}{\tau}\Big) U[t-1] + \frac{1}{\tau} \mathbf{W} S[t], \tag{1}$$

$$O[t] = \Theta(V[t] - V_{\text{th}}), \tag{2}$$

$$U[t] = V[t](1 - O[t]) + O[t] * V_{\text{reset}}, \tag{3}$$

where $S[t]$ and $O[t]$ are the input and output spikes, and $V[t]$ and $U[t]$ are membrane potentials before and after reset. Here, τ is the time constant, $\mathbf{W}$ contains synaptic weights, Θ is the Heaviside step function, and V_{th} and V_{reset} are the firing threshold and reset potential.

We consider SNNs that form their outputs by averaging over the temporal dimension:

$$\mathbf{z} = \mathbf{W}_{\text{o}} \frac{1}{T} \sum_{t=1}^{T} \mathbf{O}_L[t], \tag{4}$$

where $\mathbf{O}_L[t]$ denotes the final spiking-layer output and $\mathbf{W}_{\text{o}}$ is the output projection. Our scope is defined by this temporal aggregation structure, which is a common setting in modern SNNs (Zhou et al., 2022; Fang et al., 2021). Other models (Zhu et al., 2023; Shen et al., 2025) that use sequence positions as timesteps without this readout structure fall outside our discussion.

Following OBC (Frantar & Alistarh, 2022), we formulate PTC for SNNs as:

$$\min_{\widehat{\mathbf{W}}_l \in \mathcal{C}_l} \mathbb{E}_{\mathbf{S}_l} \mathcal{L}\Big(f_l(\mathbf{S}_l, \mathbf{W}_l), f_l(\mathbf{S}_l, \widehat{\mathbf{W}}_l) \Big), \tag{5}$$

where f_l includes synaptic integration and neuronal dynamics, $\mathbf{S}_l \in \mathbb{R}^{T \times d_{\text{col}} \times P}$ is the input spike tensor of layer l, $\mathbf{W}_l \in \mathbb{R}^{d_{\text{row}} \times d_{\text{col}}}$ is the pretrained weights, and the constraint $\mathcal{C}_l$ on compressed

weights $\widehat{\mathbf{W}}_l$ denotes customized compression patterns. To account for temporal dynamics, the van Rossum distance (VRD) (van Rossum, 2001) filters spike trains for accurate loss measurement:

$$\mathcal{L} = \left\| \mathbf{M}(\mathbf{O}_l - \widehat{\mathbf{O}}_l) \right\|_F^2, \qquad M_{ij} = \frac{1}{\tau}\Big(1 - \frac{1}{\tau}\Big)^{i-j} \mathbb{1}[i \geq j], \tag{6}$$

where $\mathbf{M}$ acts independently along the T dimension of each spike train. For IF neurons ($\tau \to \infty$), the constant factor $1/\tau$ is dropped, so that $\mathbf{M}$ reduces to the lower-triangular all-ones matrix. However, this spike-level objective remains non-quadratic. We instead adopt a quadratic membrane-level surrogate. Let $\widetilde{\mathbf{S}}_l \in \mathbb{R}^{d_{\text{col}} \times TP}$ denote S_l filtered by $\mathbf{M}$, $\mathbf{\Delta} = \widehat{\mathbf{W}} - \mathbf{W}$ denote the weight perturbations, the surrogate is

$$\mathcal{L}_{\text{mem}}(\mathbf{\Delta}) = \left\| \mathbf{\Delta}\widetilde{\mathbf{S}}_l \right\|_F^2, \qquad \mathbf{H} = 2\widetilde{\mathbf{S}}_l\widetilde{\mathbf{S}}_l^\top, \tag{7}$$

where $\mathbf{H}$ is the Hessian matrix of $\mathcal{L}_{\text{mem}}$. The ExactOBS (Frantar & Alistarh, 2022) solver then greedily removes weights with compensation as follows:

$$p^\star = \arg\min_p \frac{w_p^2}{[\mathbf{H}^{-1}]_{pp}}, \qquad \mathbf{w} \leftarrow \mathbf{w} - \frac{w_{p^\star}}{[\mathbf{H}^{-1}]_{p^\star p^\star}} [\mathbf{H}^{-1}]_{p^\star,:}. \tag{8}$$

However, such an objective cannot directly measure changes in transmitted spikes, where

$$\widehat{O}[t] - O[t] = \Theta(\widehat{V}[t] - V_{\text{th}}) - \Theta(V[t] - V_{\text{th}}) \in \{-1, 0, 1\}. \tag{9}$$

Even a very small difference between $\widehat{V}[t]$ and $V[t]$ can cause a full spike flip near the firing threshold (Wang et al., 2026b). The membrane potential error therefore says little about the spike error. This motivates examining the structure of these flips rather than the local reconstruction error alone.

3.2 Key Motivation

Flip Balance. We first define two layerwise statistics of compression-induced spike flips. For a given input, let k_i be the number of spikes that neuron i emits in the dense model over T timesteps, and let s_i^- and s_i^+ be the numbers of spikes deleted from and inserted into its spike train after compression. The *flip rate* R_l is the number of flips relative to the number of dense spikes, and the *temporal flip balance* B_l is the fraction of flips that can be paired with a flip of opposite polarity in the same neuron:

$$R_l = \frac{\sum_{i \in l}(s_i^- + s_i^+)}{\sum_{i \in l} k_i}, \qquad B_l = \frac{2\sum_{i \in l} \min(s_i^-, s_i^+)}{\sum_{i \in l}(s_i^- + s_i^+)}. \tag{10}$$

Each such pair leaves the neuron's spike count unchanged, as illustrated in Figure 1(c). To explore their effects on model performance, we compare standard OBC with mask-only pruning on SEW-ResNet (Fang et al., 2021) with LIF neurons, pretrained on CIFAR10-DVS (Li et al., 2017). Both use the same 2:4 pruning mask, but only OBC applies the compensation. Figure 1(a) shows that the two methods follow similar layerwise R_l trends. Their accuracies nevertheless differ substantially: OBC retains 71.3%, close to the dense model's 72.3%, whereas mask-only pruning drops to 51.1%. This gap motivates focusing on the structure of the flips, beyond their number.

Figure 1(b) reports B_l of both methods. Because some insertions and deletions pair up purely by chance, B_l of each method is shown with a reference (dashed line): the balance expected when every neuron keeps its flip count but the flips fall on random positions among the $T = 16$ timesteps. OBC exceeds its reference in all layers, illustrating that more inserted and deleted spikes pair up within neurons than chance would give. Mask-only pruning stays near its reference in the middle layers and falls below it in several early and late layers. Since a balanced pair leaves the neuron's spike count unchanged, such flips disturb the downstream layers less, which is consistent with OBC retaining its accuracy while mask-only pruning does not.

Motivation. These observations suggest taking B_l into account when compression-induced flips cannot be avoided. At the final spiking layer, preserving each neuron's spike count exactly preserves the temporally averaged readout in Eq. 4. However, balance may not survive the subsequent neuronal dynamics in intermediate layers, and residual errors can accumulate across layers, as observed in DNN compression (Ma et al., 2023; Arai & Ichikawa, 2026) and suggested by the increasing flip rates in Figure 1(a). Local reconstruction alone does not fully capture these downstream effects. We therefore propose to use the sensitivity to the final balance loss to guide layerwise reconstruction, which accounts for perturbations whose approximate effects on the final balance cancel.

3.3 FastSNN Framework

End-Balance Guided Compression. Although balancing the two types of flips preserves the spike count of a neuron, it does not guarantee balance in subsequent layers. Changes in spike timing can alter downstream membrane trajectories of leaky neurons, thus affecting downstream output spikes. Moreover, deriving this propagation explicitly is difficult because of the nonlinear and non-differentiable spiking dynamics. We therefore avoid enforcing balance independently through the forward path and instead use the exact balance condition at the network output to guide compression.

For the readout in Eq. 4, given the sample label y and final loss ℓ (e.g., cross-entropy loss), the terminal imbalance and the resulting end-loss variation $\mathcal{E}$ are:

$$\mathbf{b} = \frac{1}{T}\sum_{t=1}^{T}\left(\widehat{\mathbf{O}}_L[t] - \mathbf{O}_L[t]\right), \quad \delta\mathbf{z} = \widehat{\mathbf{z}} - \mathbf{z} = \mathbf{W}_{\mathrm{o}}\mathbf{b}, \quad \mathcal{E} = \ell(\widehat{\mathbf{z}}, y) - \ell(\mathbf{z}, y). \tag{11}$$

Thus, $\mathbf{b} = \mathbf{0}$ guarantees identical final results and $\mathcal{E} = 0$. Since $\ell(\widehat{\mathbf{z}}, y) = \ell(\mathbf{z} + \mathbf{W}_{\mathrm{o}}\mathbf{b}, y)$ depends on the compressed model only through the terminal imbalance $\mathbf{b}$, we take ℓ itself as the final balance loss. This exact endpoint allows backward propagation to identify intermediate perturbations whose effects cancel before reaching the output.

Assume a calibration set of N samples. Following BPTT (Wu et al., 2018) for backpropagation in SNN, let $\mathbf{S}_n[t]$ be the input spike at t for sample n and let $g_{n,t} = \partial\ell(\mathbf{z}_n, y_n)/\partial I_n[t]$ denote the gradient of its synaptic input I at t. We construct the end-loss sensitivity $\mathbf{u}_n$ with respect to weights:

$$\mathbf{u}_n = \sum_{t=1}^{T} g_{n,t}\mathbf{S}_n[t]^\top, \quad \mathbf{F} = \sum_{n=1}^{N}\mathbf{u}_n^\top\mathbf{u}_n, \tag{12}$$

where the Fisher approximation $\mathbf{F}$ measures balance changes predicted by backpropagation. Although our final goal is to balance flips at the last spiking layer, utilizing $\mathbf{F}$ can enable opposite-signed flips to balance in intermediate layers. Appendix A.2 derives this property in detail.

We incorporate this balance guidance into the solver pipeline through the trace-normalized Hessian

$$\mathbf{H}(\alpha) = (1-\alpha)\mathbf{H} + \alpha\cdot\kappa\mathbf{F}, \quad \kappa = \frac{\mathrm{tr}(\mathbf{H})}{\mathrm{tr}(\mathbf{F})}, \quad \mathbf{G} = \kappa\mathbf{F}, \tag{13}$$

while substituting $\mathbf{H}(\alpha)$ for $\mathbf{H}$ in Eq. 8 retains the solver without any incompatibility.

Adaptive choice of α. We combine $\mathbf{H}$ and $\mathbf{F}$ because $\mathbf{H}$ provides a stable reference for local reconstruction, whereas $\mathbf{F}$ captures flip balance but is subject to calibration noise. Let $\mathbf{\Sigma} = \mathbb{E}[\mathbf{G}]$ denote the expectation of $\mathbf{G}$ over the draw of the calibration set. Then $\mathbf{H}(\alpha)$ is an estimator of $\mathbf{\Sigma}$ that shrinks the noisy $\mathbf{G}$ towards the stable $\mathbf{H}$, and its mean squared error splits into a bias term contributed by $\mathbf{H}$ and a variance term contributed by $\mathbf{G}$:

$$\begin{aligned} \mathrm{MSE}(\alpha) = \mathbb{E}\left\|\mathbf{H}(\alpha) - \mathbf{\Sigma}\right\|_F^2 = (1-\alpha)^2\beta + \alpha^2\sigma^2, \\ \beta = \left\|\mathbf{H} - \mathbf{\Sigma}\right\|_F^2, \qquad \sigma^2 = \mathbb{E}\left\|\mathbf{G} - \mathbf{\Sigma}\right\|_F^2, \end{aligned} \tag{14}$$

which is minimized at $\alpha^\star = \beta/(\beta + \sigma^2)$. Hence α moves towards $\mathbf{F}$ when $\mathbf{H}$ is far from the end-balance loss (large β) and towards $\mathbf{H}$ when the Fisher estimate is noisy (large σ^2).

Neither β nor σ^2 is observable directly, but both can be estimated from the calibration set. We split it into two disjoint halves, build $\mathbf{F}^{(1)}$ and $\mathbf{F}^{(2)}$ on each, and deploy their sum $\mathbf{F} = \mathbf{F}^{(1)} + \mathbf{F}^{(2)}$. The two halves share the same expectation and carry independent noise, so their difference isolates the sampling noise, whereas $\|\mathbf{H} - \mathbf{G}\|_F^2$ measures bias plus noise:

$$\widehat{\sigma}^2 = \kappa^2\left\|\mathbf{F}^{(1)} - \mathbf{F}^{(2)}\right\|_F^2, \qquad \widehat{\beta} = \left[\left\|\mathbf{H} - \mathbf{G}\right\|_F^2 - \widehat{\sigma}^2\right]_+, \qquad \widehat{\alpha} = \frac{\widehat{\beta}}{\widehat{\beta} + \widehat{\sigma}^2}, \tag{15}$$

where $\widehat{\sigma}^2$ and $\|\mathbf{H} - \mathbf{G}\|_F^2$ are unbiased estimators of σ^2 and $\beta + \sigma^2$. Appendix A.3 gives the derivation.

Compression Framework. Algorithm 1 summarizes FastSNN as a PTC pipeline. Given a trained SNN with pretrained weights $\mathbf{W} = \{\mathbf{W}_l\}_{l=1}^{L}$ and a labeled calibration set $\mathcal{D}$ of N samples, FastSNN first collects the layerwise Hessian and the end-loss Fisher

from the dense model, constructs $\mathbf{H}(\alpha)$, and passes the resulting metric to the ExactOBS solver. Following OBC, we represent both linear and convolutional layers as linear operators, $\mathbf{W}_l \in \mathbb{R}^{d_{\text{row}} \times d_{\text{col}}}$, where convolutions can be unfolded into linear ones.

Step 1: Forward Hessian collection. A gradient-free forward pass records $\{\mathbf{S}_{l,n}\}$ for every compressed layer. The VRD matrix $\mathbf{M}$ is applied along the temporal dimension of each $\mathbf{S}_{l,n}$, yielding $\widetilde{\mathbf{S}}_{l,n} \in \mathbb{R}^{d_{\text{col}} \times TP}$. Following Eq. 7 and the row-wise solver in ExactOBS, the layerwise Hessian is then accumulated as

$$\mathbf{H}_l = \frac{2}{N_s} \sum_{n=1}^{N} \widetilde{\mathbf{S}}_{l,n} \widetilde{\mathbf{S}}_{l,n}^{\top}, \tag{16}$$

where $N_s = T \cdot N \cdot P$. The resulting $\mathbf{H}_l \in \mathbb{R}^{d_{\text{col}} \times d_{\text{col}}}$ is shared by all output rows.

Algorithm 1 FastSNN compression pipeline

for $l = 1, 2, \ldots, L$ **do**
  $\mathbf{H}_l \leftarrow \frac{2}{N_s} \Sigma_{n \in \mathcal{D}} \widetilde{\mathbf{S}}_{l,n} \widetilde{\mathbf{S}}_{l,n}^{\top}$,
  for $r = 1, \ldots, d_{\text{row}}^l$ **do**
    $\mathbf{u}_{l,n,r} \leftarrow \nabla_{\mathbf{w}_{l,r}} \ell$,
    $\mathbf{F}_{l,r}^{(k)} \leftarrow \Sigma_{n \in \mathcal{D}^{(k)}} \mathbf{u}_{l,n,r}^{\top} \mathbf{u}_{l,n,r}, \quad k \in \{1, 2\}$
    $\mathbf{F}_{l,r} \leftarrow \mathbf{F}_{l,r}^{(1)} + \mathbf{F}_{l,r}^{(2)}$, $\kappa_{l,r} \leftarrow \text{tr}(\mathbf{H}_l) / \text{tr}(\mathbf{F}_{l,r})$
    Obtain $\widehat{\alpha}_{l,r}$ using Eq. 15.
  end for
  $\widehat{\alpha}_l = \frac{1}{d_{\text{row}}} \sum_{r=1}^{d_{\text{row}}} \widehat{\alpha}_{l,r}$
  for $r = 1, \ldots, d_{\text{row}}^l$ **do**
    $\mathbf{H}_{l,r}(\widehat{\alpha}_l) \leftarrow (1 - \widehat{\alpha}_l)\mathbf{H}_l + \widehat{\alpha}_l \cdot \kappa_{l,r} \mathbf{F}_{l,r}$
    $\widehat{\mathbf{W}}_{l,r} \leftarrow \mathbf{ExactOBS}_{\mathcal{C}}(\mathbf{W}_{l,r}, \mathbf{H}_{l,r}(\widehat{\alpha}_l))$
  end for
end for

Step 2: Backward Fisher collection. A second forward pass followed by surrogate backpropagation collects the end-loss Fisher information matrix. Let $\mathbf{g}_{l,n,r,t}$ denote the loss gradient with respect to the output of row r of the synaptic input $I_{l,n}[t]$ at timestep t. Then, following Eq. 12, the Fisher block $\mathbf{F}_{l,r}$ with respect to $\mathbf{W}_{r,:}$ is:

$$\mathbf{u}_{l,n,r} = \sum_{t=1}^{T} \mathbf{g}_{l,n,r,t} \mathbf{S}_{l,n}[t]^{\top}, \quad \mathbf{F}_{l,r} = \sum_{n=1}^{N} \mathbf{u}_{l,n,r}^{\top} \mathbf{u}_{l,n,r}. \tag{17}$$

Since $\mathbf{H}_l$ is shared by all output rows while $\mathbf{F}_{l,r}$ is row-specific, Eq. 15 gives a row-specific $\widehat{\alpha}_{l,r}$. Rather than using it directly, we average the estimates within a layer and assign $\widehat{\alpha}_l$ to all rows:

$$\widehat{\alpha}_l = \frac{1}{d_{\text{row}}} \sum_{r=1}^{d_{\text{row}}} \widehat{\alpha}_{l,r}, \quad \mathbf{H}_{l,r}(\widehat{\alpha}_l) = (1 - \widehat{\alpha}_l)\mathbf{H}_l + \widehat{\alpha}_l \cdot \kappa_{l,r} \mathbf{F}_{l,r}. \tag{18}$$

Notably, $\mathbf{H}_{l,r}(\widehat{\alpha}_l)$ are independent across rows, so the ExactOBS solver can process rows in parallel.

Compression Types. For N:M pruning, we use the group-constrained ExactOBS elimination rule. Unstructured pruning uses the same solver with one sparsity budget for the entire output row. For b-bit quantization, we follow the fixed-order GPTQ update (Frantar et al., 2022) under $\mathbf{H}(\alpha)$.

Complexity. We analyze one layer and omit the layer index. Relative to OBC, whose per-layer cost is dominated by the $\mathcal{O}(d_{\text{row}} d_{\text{col}}^3)$ ExactOBS solve, FastSNN adds Fisher statistics that require one surrogate backward pass over the calibration set, following Eq. 17, which together cost $\mathcal{O}\big(N d_{\text{row}} d_{\text{col}} (TP + d_{\text{col}})\big)$. For pruning, the per-row $\mathbf{H}_r(\alpha)$ leaves the ExactOBS order unchanged, since the extra factorization per row costs $\mathcal{O}(d_{\text{col}}^3)$, the same as the rank-one updates for that row. For quantization, the fixed-order GPTQ update can no longer share one factorization across rows, so its cost increases from $\mathcal{O}(d_{\text{col}}^3 + d_{\text{row}} d_{\text{col}}^2)$ to $\mathcal{O}(d_{\text{row}} d_{\text{col}}^3)$. The per-row Fisher statistics occupy $\mathcal{O}(d_{\text{row}} d_{\text{col}}^2)$ memory, while processing d'_{row} rows at a time reduces the memory cost to $\mathcal{O}(d'_{\text{row}} d_{\text{col}}^2)$ without changing the arithmetic cost. Runtime measurements in Appendix A.4 show that our framework remains computationally practical.

4 Experiments

Experimental Setup. We evaluate FastSNN under several standard compression settings: unstructured pruning, 2:4, 4:8, and 2:8 pruning, uniform weight quantization, and joint pruning and quantization for deployment on different accelerators. We test ResNet-series (Fang et al., 2021; Hu et al., 2024) and Transformer-series (Fang et al., 2026; Zhou et al., 2024; Shi et al., 2024) models on ImageNet (Deng et al., 2009) and CIFAR-100 (Krizhevsky et al., 2009) for image classification and

Table 1: Overall post-training *pruning* results of accuracy (%): (a) N:M structured sparsity, (b) unstructured sparsity. Best results are in **bold**, second best are underlined. More results can be found in Appendix A.6. Note: following prior work (Zhou et al., 2022; Deng et al., 2022; Fang et al., 2021), we use a 9000/1000 train-test split for CIFAR10-DVS. Therefore, the accuracy of CIFAR10-DVS can only be reported to one decimal place.

(a) N:M structured sparsity

		Wanda			OSSCAR			SBC			**Ours**		
Model	Dense	2:4	4:8	2:8	2:4	4:8	2:8	2:4	4:8	2:8	2:4	4:8	2:8
ImageNet													
MS-ResNet-18	63.04	54.31	57.15	22.68	59.33	60.71	43.88	60.80	61.91	40.97	**62.44**	**62.53**	**58.27**
SEW-ResNet-34	66.83	56.57	60.39	21.97	61.88	63.71	41.22	64.88	65.45	49.81	**65.55**	**65.94**	**59.94**
SpikingResFormer-S	75.88	69.04	70.89	42.64	72.70	73.62	57.99	72.82	**73.81**	54.33	**72.89**	73.80	**60.84**
MaxFormer-10-768	82.21	79.11	79.97	66.41	**80.15**	**80.67**	70.82	79.99	80.62	65.57	80.01	**80.67**	**72.05**
CIFAR-100													
MS-ResNet-56	64.60	52.69	57.22	21.58	58.12	60.38	36.19	60.53	62.02	33.26	**63.05**	**63.99**	**51.84**
SEW-ResNet-56	64.82	57.36	60.13	28.55	59.85	61.36	39.15	62.59	63.75	40.33	**63.95**	**64.67**	**58.99**
MaxFormer	75.78	73.43	73.82	65.23	74.06	74.67	70.25	74.35	**75.17**	68.93	**74.47**	74.66	**71.03**
CIFAR10-DVS													
MS-ResNet-DVS	68.30	54.20	57.80	26.00	59.90	61.90	42.90	60.60	62.30	47.70	**62.10**	**63.50**	**56.80**
SEW-ResNet-DVS	72.30	71.10	69.10	66.90	71.20	72.40	69.90	71.40	71.60	68.30	**72.40**	**72.80**	**71.20**
SpikingResFormer-Ti	80.70	73.40	76.10	62.40	76.10	**77.80**	71.50	**77.40**	77.20	73.70	76.20	76.90	**74.10**

(b) Unstructured sparsity

		Wanda			OSSCAR			SBC			**Ours**		
Model	Dense	50%	75%	87.5%	50%	75%	87.5%	50%	75%	87.5%	50%	75%	87.5%
ImageNet													
MS-ResNet-18	63.04	59.94	37.51	7.00	62.50	55.43	32.85	62.71	54.40	19.60	**62.81**	**60.86**	**49.40**
SEW-ResNet-34	66.83	63.18	39.67	8.25	65.94	57.45	31.76	66.28	59.43	27.15	**66.64**	**63.52**	**49.34**
SpikingResFormer-S	75.88	73.14	53.64	13.05	74.63	64.67	31.53	**74.96**	65.08	22.20	74.70	**67.47**	**42.20**
MaxFormer-10-768	82.21	80.70	70.94	37.24	81.13	74.15	47.01	**81.47**	74.05	30.73	80.91	**75.84**	**55.30**
CIFAR-100													
MS-ResNet-56	64.60	60.11	33.83	9.25	63.48	52.37	24.32	63.12	45.71	6.88	**63.93**	**58.83**	**34.21**
SEW-ResNet-56	64.82	61.91	40.40	10.66	64.55	55.86	33.16	64.70	53.22	9.33	**64.87**	**61.84**	**45.92**
MaxFormer	75.78	74.47	68.21	51.39	75.04	72.00	64.83	**75.49**	72.08	59.61	74.69	**72.83**	**65.55**
CIFAR10-DVS													
MS-ResNet-DVS	68.30	60.50	37.10	10.20	65.00	57.80	37.80	61.90	52.00	28.10	**65.50**	**60.20**	**45.80**
SEW-ResNet-DVS	72.30	71.40	67.50	57.40	71.90	71.80	66.20	71.20	71.10	63.30	**72.20**	**72.40**	**70.70**
SpikingResFormer-Ti	80.70	77.20	66.00	42.90	**78.10**	74.40	66.10	77.30	74.10	69.60	77.20	**75.00**	**71.50**

on CIFAR10-DVS (Li et al., 2017) for neuromorphic recognition. We also test EMS-YOLO (Su et al., 2023) and SpikeYOLO (Luo et al., 2024) on COCO (Lin et al., 2014) for object detection, and SpikingVGG16 on UT-HAR (Yang et al., 2023) for wireless sensing recognition. For pruning, we compare with SBC (Shi et al., 2025) (an SNN-specific OBC method), Wanda (Sun et al., 2024), and OSSCAR (Meng et al., 2024b). For weight-only quantization, we compare with round-to-nearest (RTN), GPTQ (Frantar et al., 2022), NeuronQuant (Li et al., 2025), and SBC with the OBQ solver. All methods use the same calibration data, a small random subset of each training set (about 0.1% of ImageNet), along with the same compressed layers and BN recalibration protocol. All experiments are conducted on an NVIDIA RTX 4090 GPU. More details on reproducibility are in Appendix A.4.

4.1 Comparison with the state-of-the-art methods

Pruning. Table 1 summarizes the results for N:M structured pruning and unstructured pruning. FastSNN is competitive with existing methods at moderate sparsity, while its advantages become more pronounced under stronger compression. Across the configurations in Table 1(a), its average accuracy gain over the strongest baseline for each configuration increases from 0.75% and 0.42% at 2:4 and 4:8 sparsity to 7.45% at 2:8 sparsity. A similar trend appears in unstructured pruning: the average gain increases from 3.04% at 75% sparsity to 9.09% at 87.5% sparsity. To figure out why FastSNN benefits more from aggressive pruning, we prune SEW-ResNet-DVS on CIFAR10-

Table 2: Overall post-training weight *quantization* results of accuracy (%). Best results are in **bold**, second best are underlined. More results can be found in Appendix A.6.

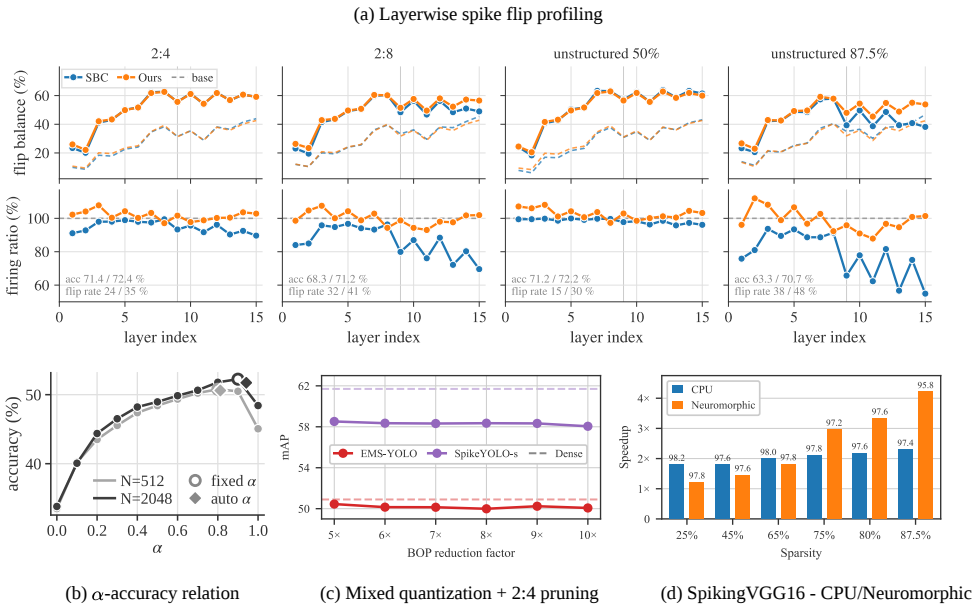
		RTN			GPTQ			NeuronQuant			SBC			**Ours**		
Model	Dense	4-bit	3-bit	2-bit	4-bit	3-bit	2-bit	4-bit	3-bit	2-bit	4-bit	3-bit	2-bit	4-bit	3-bit	2-bit
ImageNet																
MS-ResNet-18	63.04	61.20	50.98	35.17	62.60	59.47	53.02	60.64	40.76	2.26	62.53	60.73	37.68	**62.91**	**61.60**	**55.90**
SEW-ResNet-34	66.83	65.14	55.69	23.14	66.20	63.72	56.99	64.67	44.09	0.67	66.34	65.19	53.99	**66.65**	**65.75**	**58.84**
SpikingResFormer-L	78.44	76.32	69.51	66.06	**77.11**	74.32	**73.72**	76.29	67.18	23.97	76.78	75.61	73.66	76.86	**75.63**	73.00
MS-QKFormer-384	76.26	75.62	69.45	66.47	**76.06**	73.05	**70.49**	75.61	68.59	52.77	75.68	73.87	54.91	76.01	**74.00**	68.72
CIFAR-100																
MS-ResNet-32	61.75	61.19	57.37	49.59	61.97	60.49	55.26	61.07	56.25	46.27	61.98	60.20	47.30	**62.39**	**61.42**	**57.23**
SEW-ResNet-56	64.82	64.48	61.41	54.22	65.01	64.14	60.21	64.89	60.54	49.04	64.53	63.72	56.08	**65.53**	**64.47**	**61.41**
SpikingResFormer-Ti	83.56	82.03	72.44	69.26	82.42	77.61	75.25	81.37	68.34	50.84	82.34	**80.58**	**76.51**	**82.43**	80.25	75.96
MS-QKFormer	74.49	73.70	69.15	64.57	73.47	72.35	**70.47**	73.54	68.47	55.99	**74.50**	**73.59**	67.03	73.76	72.71	69.27
CIFAR10-DVS																
MS-ResNet-DVS	68.30	65.80	61.70	54.00	66.20	**65.60**	61.10	**66.30**	59.00	53.80	66.00	62.80	55.20	**66.30**	64.90	**61.20**
SEW-ResNet-DVS	72.30	72.00	70.40	68.10	**72.30**	72.00	69.10	72.10	62.00	62.30	**72.30**	71.50	69.60	72.10	**72.50**	**72.20**
SpikingResFormer-Ti	80.70	77.20	70.60	67.80	**77.60**	72.70	72.60	77.00	64.70	52.20	77.40	75.60	**74.10**	76.70	**76.90**	73.40
MaxFormer	78.70	76.10	75.10	75.40	76.10	**77.00**	**76.00**	76.70	76.90	75.70	75.90	76.10	73.40	**76.90**	74.60	75.90

DVS at 4 sparsity levels and profile its spike statistics (Figure 2(a)). Stronger pruning increases flip rates for both methods, yet SBC produces fewer flips despite its lower accuracy. At 87.5% sparsity, FastSNN achieves 70.7% accuracy versus 63.3% for SBC, despite a higher relative flip rate of 47.7% versus 38.5%. SBC also exhibits weaker deep-layer flip balance and greater suppression of firing activity. In the final profiled layer, FastSNN achieves 53.8% balance versus 38.2% for SBC and maintains the dense firing rate, whereas SBC retains only 54.9% of it. A similar pattern appears under 2:8 pruning, where the final-layer relative firing ratios are 101.9% for FastSNN and 69.6% for SBC. Consistent with our analysis in Section 3.2, higher accuracy accompanies better temporal flip balance and preservation of firing activity, despite more flips. As a result, FastSNN better preserves accuracy under aggressive compression by accounting for flip balance.

Quantization. Table 2 presents the results for uniform weight quantization. The gains are less uniform than those observed for high-sparsity pruning and depend on the architecture and bit-width. FastSNN is particularly effective for low-bit ResNet models. On ImageNet, it achieves 55.90% and 58.84% accuracy for MS-ResNet-18 and SEW-ResNet-34 at 2 bits, exceeding GPTQ by 2.88% and 1.85%, respectively. On CIFAR10-DVS, the 2-bit SEW-ResNet retains 72.20% accuracy, compared with 72.30% for the dense model. Results on Transformer-based architectures are broadly comparable to established quantizers. For instance, our 2-bit accuracy on ImageNet SpikingResFormer-L is 73%, compared with 73.72% for GPTQ. These results thus demonstrate the applicability of FastSNN beyond pruning, with the clearest accuracy benefits observed in low-bit ResNet settings.

Ablation Study. To examine whether $\widehat{\alpha}$ enables a good trade-off between the two terms of Eq. 13, we sweep a fixed $\alpha \in \{0, 0.1, \ldots, 1\}$ and compare it with using $\widehat{\alpha}$ on MS-ResNet-56 under 2:8 pruning on CIFAR-100, using $N = 512$ and $N = 2048$ (the setting used in our main experiments) calibration samples; all results are averaged over three seeds (Figure 2(b)). At $\alpha = 0$, FastSNN falls back to SBC, and accuracy falls to 33.80% and 33.50% for the two budgets: local reconstruction is insensitive to the calibration size but does not account for flip balance. At $\alpha = 1$, only $\mathbf{F}$ remains, which corresponds to a GuidedQuant-style variant of our framework. It reaches 48.42% with $N = 2048$ but drops to 45.06% with $N = 512$, showing that the pure balance guidance is sensitive to the noise of the Fisher estimate. FastSNN outperforms both methods by adaptively combining information from local reconstruction and flip balance, and the empirically best choices, $\alpha = 0.9$ for $N = 2048$ and $\alpha = 0.8$ for $N = 512$, reach 52.25% and 50.79%. Without any search, our framework selects $\alpha \approx 0.94$ and 0.81 and attains close accuracies of 51.73% and 50.61%, respectively. $\widehat{\alpha}$ also decreases when fewer samples are available, following the trend predicted in Eq. 14. These results indicate that FastSNN benefits from both local reconstruction and flip balance guidance, and that the split-half estimate recovers a near-optimal weighting between them without a grid search.

4.2 On-Device Evaluation



(b) α-accuracy relation (c) Mixed quantization + 2:4 pruning (d) SpikingVGG16 - CPU/Neuromorphic

Figure 2: (a) Layerwise spike profiles of SEW-ResNet-DVS on CIFAR10-DVS. (b) α-accuracy relation of MS-ResNet-56 on CIFAR-100, 2:8 pruning. (c) Mixed quantization and 2:4 pruning for various BOP reduction targets on NVIDIA Jetson AGX Orin platform. (d) Joint block-pruning and quantization for CPU/neuromorphic chip inference time speedups.

Table 3: Hardware deployment configurations.

Model	Dataset	Platform	Compression
EMS-YOLO SpikeYOLO-s	COCO	Jetson AGX Orin	Mixed quant+2:4 pruning
SpikingVGG16	UT-HAR	AMD Ryzen 9 CPU Darwin3	Block Sparsity

Now we validate our framework across different hardware platforms, as depicted in Table 3. NVIDIA GPUs equipped with Sparse Tensor Cores provide hardware acceleration for 2:4 structured sparsity (Pool, 2020). We therefore combine this sparsity pattern with mixed-precision weight quantization, achieving a $10\times$ BOP (number of bits times flops) reduction on COCO with mAP drops of 0.83% and 3.65% for EMS-YOLO and SpikeYOLO-s, respectively (Figure 2(c)). We further deploy SpikingVGG16 for UT-HAR on an AMD CPU and a neuromorphic chip, Darwin3 (Ma et al., 2024), using platform-specific block sparsity patterns detailed in Appendix A.5. Figure 2(d) shows that FastSNN maintains high accuracy under aggressive compression. At 87.5% sparsity, the CPU model achieves 97.4% accuracy, only 1.4% below the dense baseline of 98.8%, with a $2.3\times$ inference speedup. The neuromorphic chip reaches a $4.2\times$ speedup while retaining 95.8% accuracy. Speedup increases more strongly with sparsity on the neuromorphic chip than on the CPU. This trend is consistent with specialized support for spike processing, whereas software neuron updates and data movement may limit the CPU's gains from reduced computation. These results demonstrate that FastSNN can translate compression into practical inference gains across conventional and neuromorphic hardware.

5 Conclusions & Discussions

We present FastSNN, a unified post-training framework for SNN pruning and quantization. Our central finding is that compression quality is governed not by the number of spike flips alone, but by whether their effects balance over time. Based on this observation, FastSNN incorporates end-level flip balance into the OBC framework by combining local membrane reconstruction with task-aware end-loss sensitivity, yielding accurate compression across structured pruning, unstructured pruning, and weight quantization without retraining. These results establish spike-flip balance as a useful principle for understanding and optimizing compressed SNNs. Future work will extend this principle to larger and more complex architectures, particularly spiking large language models.

AI USE STATEMENT

In this work, generative AI tools were used only as assistants after the authors had conceived the research problem, the flip-balance perspective, the FastSNN method, and the experimental design; all AI-assisted content was produced under the authors' direction and control. Specifically, we used AI coding assistants (Claude Code) to help implement the compression pipeline and experiment scripts and to organize and reformat experimental data and results; the generated code was reviewed, tested, and verified by the authors. We used generative AI tools (GPT and Claude models) to check the notation and derivations in Section 3 and Appendices A.1–A.3 for consistency and algebraic errors, with all corrections verified by the authors. We have not used generative AI tools to generate synthetic datasets, develop the theoretical model, formulate mathematical claims, propose or refine hypotheses, design the experiments, or interpret results; translation and qualitative data analysis are not applicable to this work. Additionally, we used generative AI tools (GPT and Claude models) to draft and revise parts of the text, edit the paper for readability, and check grammar. We have reviewed all AI-assisted code, derivations, and text, and we take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Algorithm 1 specifies the complete FastSNN pipeline, and Appendices A.1–A.3 give full derivations of the surrogate Hessian, the end-loss guidance, and the adaptive mixing weight. Section 4 and Appendix A.4 list all models, datasets, compression settings, calibration budgets, the fixed random seed, and the BN recalibration procedure; Appendix A.4 also reports seed sensitivity and runtimes. Appendix A.5 describes the block-sparsity patterns and hardware used in the on-device evaluation. Source code will be released publicly upon acceptance.

REFERENCES

Yamato Arai and Yuma Ichikawa. Quantization error propagation: Revisiting layer-wise post-training quantization. *Advances in Neural Information Processing Systems*, 38:151916–151951, 2026.

Yanqi Chen, Zhaofei Yu, Wei Fang, Tiejun Huang, and Yonghong Tian. Pruning of deep spiking neural networks through gradient rewiring. *arXiv preprint arXiv:2105.04916*, 2021.

Yanqi Chen, Zhaofei Yu, Wei Fang, Zhengyu Ma, Tiejun Huang, and Yonghong Tian. State transition of dendritic spines improves learning of sparse spiking neural networks. In *International Conference on Machine Learning*, pp. 3701–3715. PMLR, 2022.

Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, et al. Loihi: A neuromorphic manycore processor with on-chip learning. *Ieee micro*, 38(1):82–99, 2018.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Lei Deng, Yujie Wu, Yifan Hu, Ling Liang, Guoqi Li, Xing Hu, Yufei Ding, Peng Li, and Yuan Xie. Comprehensive snn compression using admm optimization and activity regularization. *IEEE transactions on neural networks and learning systems*, 34(6):2791–2805, 2021.

Shikuang Deng, Yuhang Li, Shanghang Zhang, and Shi Gu. Temporal efficient training of spiking neural network via gradient re-weighting. *arXiv preprint arXiv:2202.11946*, 2022.

Tim Dettmers and Luke Zettlemoyer. The case for 4-bit precision: k-bit inference scaling laws. In *International Conference on Machine Learning*, pp. 7750–7774. PMLR, 2023.

Wei Fang, Zhaofei Yu, Yanqi Chen, Tiejun Huang, Timothée Masquelier, and Yonghong Tian. Deep residual learning in spiking neural networks. *Advances in neural information processing systems*, 34:21056–21069, 2021.

Yuetong Fang, Deming Zhou, Ziqing Wang, Hongwei Ren, Lusong Li, Renjing Xu, et al. Spiking neural networks need high-frequency information. *Advances in Neural Information Processing Systems*, 38:103404–103425, 2026.

Elias Frantar and Dan Alistarh. Optimal brain compression: A framework for accurate post-training quantization and pruning. *Advances in Neural Information Processing Systems*, 35:4475–4488, 2022.

Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International conference on machine learning*, pp. 10323–10337. PMLR, 2023.

Elias Frantar, Eldar Kurtic, and Dan Alistarh. M-fac: Efficient matrix-free approximations of second-order information. *Advances in Neural Information Processing Systems*, 34:14873–14886, 2021.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

Yifan Hu, Lei Deng, Yujie Wu, Man Yao, and Guoqi Li. Advancing spiking neural networks toward deep residual learning. *IEEE transactions on neural networks and learning systems*, 36(2):2353–2367, 2024.

Itay Hubara, Yury Nahshan, Yair Hanani, Ron Banner, and Daniel Soudry. Accurate post training quantization with small calibration sets. In *International conference on machine learning*, pp. 4466–4475. PMLR, 2021.

Jinuk Kim, Marwa El Halabi, Wonpyo Park, Clemens JS Schaefer, Deokjae Lee, Yeonhong Park, Jae W Lee, and Hyun Oh Song. Guidedquant: Large language model quantization via exploiting end loss guidance. *arXiv preprint arXiv:2505.07004*, 2025.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Haomin Li, Fangxin Liu, Zewen Sun, Zongwu Wang, Shiyuan Huang, Ning Yang, and Li Jiang. Neuronquant: Accurate and efficient post-training quantization for spiking neural networks. In *Proceedings of the 30th Asia and South Pacific Design Automation Conference*, pp. 734–740, 2025.

Hongmin Li, Hanchao Liu, Xiangyang Ji, Guoqi Li, and Luping Shi. Cifar10-dvs: an event-stream dataset for object classification. *Frontiers in neuroscience*, 11:244131, 2017.

Yuhang Li, Ruihao Gong, Xu Tan, Yang Yang, Peng Hu, Qi Zhang, Fengwei Yu, Wei Wang, and Shi Gu. Brecq: Pushing the limit of post-training quantization by block reconstruction. *arXiv preprint arXiv:2102.05426*, 2021.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. *Proceedings of machine learning and systems*, 6:87–100, 2024.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pp. 740–755. Springer, 2014.

Yisha Lu, Liwen Jing, Jiangmao Zheng, and Bowen Zhang. Spiking-aided neural architecture for efficient and robust wifi sensing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 24106–24114, 2026.

Xinhao Luo, Man Yao, Yuhong Chou, Bo Xu, and Guoqi Li. Integer-valued training and spike-driven inference spiking neural network for high-performance and energy-efficient object detection. In *European Conference on Computer Vision*, pp. 253–272. Springer, 2024.

De Ma, Xiaofei Jin, Shichun Sun, Yitao Li, Xundong Wu, Youneng Hu, Fangchao Yang, Huajin Tang, Xiaolei Zhu, Peng Lin, et al. Darwin3: a large-scale neuromorphic chip with a novel isa and on-chip learning. *National Science Review*, 11(5):nwae102, 2024.

Yuexiao Ma, Huixia Li, Xiawu Zheng, Xuefeng Xiao, Rui Wang, Shilei Wen, Xin Pan, Fei Chao, and Rongrong Ji. Solving oscillation problem in post-training quantization through a theoretical perspective. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7950–7959. IEEE, 2023.

Xiang Meng, Kayhan Behdin, Haoyue Wang, and Rahul Mazumder. Alps: Improved optimization for highly sparse one-shot pruning for large language models. *Advances in neural information processing systems*, 37:37594–37625, 2024a.

Xiang Meng, Shibal Ibrahim, Kayhan Behdin, Hussein Hazimeh, Natalia Ponomareva, and Rahul Mazumder. Osscar: One-shot structured pruning in vision and language models with combinatorial optimization. *arXiv preprint arXiv:2403.12983*, 2024b.

Jeff Pool. Accelerating sparsity in the nvidia ampere architecture. *GTC 2020*, 4, 2020.

Yongzhi She, Qihua Zhou, Yuhao Wang, Yaodong Huang, Laizhong Cui, and Jingcai Guo. Spik4lite: Refactoring neuromorphic sparsity for efficient spiking neural networks on commodity edge devices. In *Forty-third International Conference on Machine Learning*, 2026.

Jiangrong Shen, Qi Xu, Jian K Liu, Yueming Wang, Gang Pan, and Huajin Tang. Esl-snns: An evolutionary structure learning strategy for spiking neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 86–93, 2023.

Shuaijie Shen, Chao Wang, Renzhuo Huang, Yan Zhong, Qinghai Guo, Zhichao Lu, Jianguo Zhang, and Luziwei Leng. Spikingssms: Learning long sequences with sparse and parallel spiking state space models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 20380–20388, 2025.

Lianfeng Shi, Ao Li, and Benjamin Ward-Cherrier. Spiking brain compression: exploring one-shot post-training pruning and quantization for spiking neural networks. In *OPT 2025: Optimization for Machine Learning*, 2025.

Xinyu Shi, Zecheng Hao, and Zhaofei Yu. Spikingresformer: Bridging resnet and vision transformer in spiking neural networks. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5610–5619. IEEE, 2024.

Xinyu Shi, Tong Bu, and Zhaofei Yu. Unisparse: Combining weight pruning and spike sparsification in spiking neural networks. In *Forty-third International Conference on Machine Learning*, 2026.

Qiaoyi Su, Yuhong Chou, Yifan Hu, Jianing Li, Shijie Mei, Ziyang Zhang, and Guoqi Li. Deep directly-trained spiking neural networks for object detection. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6532–6542. IEEE, 2023.

Mingjie Sun, Zhuang Liu, Anna Bair, and Zico Kolter. A simple and effective pruning approach for large language models. In *International Conference on Learning Representations*, volume 2024, pp. 4942–4964, 2024.

Mark CW van Rossum. A novel spike distance. *Neural computation*, 13(4):751–763, 2001.

Jiaqi Wang, Liutao Yu, Xiongri Shen, Sihang Guo, Chenlin Zhou, Leilei Zhao, Yi Zhong, Zhiguo Zhang, and Zhengyu Ma. Spikcommander: A high-performance spiking transformer with multi-view learning for efficient speech command recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 2119–2127, 2026a.

Shuai Wang, Malu Zhang, Yulin Jiang, Dehao Zhang, Ammar Belatreche, Yu Liang, Yimeng Shan, Zijian Zhou, Yang Yang, and Haizhou Li. Robust spiking neural networks against adversarial attacks. *arXiv preprint arXiv:2602.20548*, 2026b.

Wenjie Wei, Yu Liang, Ammar Belatreche, Yichen Xiao, Honglin Cao, Zhenbang Ren, Guoqing Wang, Malu Zhang, and Yang Yang. Q-snns: Quantized spiking neural networks. In *Proceedings of the 32nd ACM international conference on multimedia*, pp. 8441–8450, 2024.

Wenjie Wei, Malu Zhang, Zijian Zhou, Ammar Belatreche, Yimeng Shan, Yu Liang, Honglin Cao, Jieyuan Zhang, and Yang Yang. Qp-snn: Quantized and pruned spiking neural networks. In *International Conference on Learning Representations*, volume 2025, pp. 96470–96495, 2025.

Xiuying Wei, Ruihao Gong, Yuhang Li, Xianglong Liu, and Fengwei Yu. Qdrop: Randomly dropping quantization for extremely low-bit post-training quantization. *arXiv preprint arXiv:2203.05740*, 2022.

Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, and Luping Shi. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in neuroscience*, 12:331, 2018.

Jianfei Yang, Xinyan Chen, Han Zou, Chris Xiaoxuan Lu, Dazhuo Wang, Sumei Sun, and Lihua Xie. Sensefi: A library and benchmark on deep-learning-empowered wifi human sensing. *Patterns*, 4 (3), 2023.

Shuhan Ye, Yi Yu, Qixin Zhang, Chenqi Kong, Qiangqiang Wu, Xudong Jiang, and Dacheng Tao. Sparse by rule: Probability-based n: M pruning for spiking neural networks. *arXiv preprint arXiv:2511.12097*, 2025.

Ruokai Yin, Yuhang Li, Abhishek Moitra, and Priyadarshini Panda. Mint: Multiplier-less integer quantization for energy efficient spiking neural networks. In *2024 29th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 830–835. IEEE, 2024.

Di Yu, Zhenyu Liao, Changze Lv, Wentao Tong, Linshan Jiang, Sijie Ji, Xin Du, Hailiang Zhao, Xiaoqing Zheng, and Shuiguang Deng. Frequency matching in spiking neural networks for mmwave sensing. *arXiv preprint arXiv:2605.09983*, 2026a.

Di Yu, Helin Zheng, Changze Lv, Xin Du, Linshan Jiang, Xiang Liu, Gang Pan, and Shuiguang Deng. Masi: Memory-adaptive inference framework for spiking neural networks on edge devices. In *Proceedings of the ACM Web Conference 2026*, pp. 5063–5074, 2026b.

Chenlin Zhou, Han Zhang, Zhaokun Zhou, Liutao Yu, Liwei Huang, Xiaopeng Fan, Li Yuan, Zhengyu Ma, Huihui Zhou, and Yonghong Tian. Qkformer: Hierarchical spiking transformer using qk attention. *Advances in Neural Information Processing Systems*, 37:13074–13098, 2024.

Zhaokun Zhou, Yuesheng Zhu, Chao He, Yaowei Wang, Shuicheng Yan, Yonghong Tian, and Li Yuan. Spikformer: When spiking neural network meets transformer. *arXiv preprint arXiv:2209.15425*, 2022.

Rui-Jie Zhu, Qihang Zhao, Guoqi Li, and Jason K Eshraghian. Spikegpt: Generative pre-trained language model with spiking neural networks. *arXiv preprint arXiv:2302.13939*, 2023.

A Appendix

A.1 Deriving Exact Hessian For Spiking Layer

Fix an output row $\mathbf{w} \in \mathbb{R}^{1\times d}$ and a calibration input $\mathbf{S} \in \mathbb{R}^{T\times d\times P}$. Let $\boldsymbol{\delta} = \widehat{\mathbf{w}} - \mathbf{w}$ and $\mathbf{o}(\mathbf{w}) \in \{0,1\}^{T\times P}$ denote the output spike train. The reconstruction objective is

$$\mathcal{L}_{\mathrm{spk}}(\boldsymbol{\delta}) = \left\|\mathbf{M}\big[\mathbf{o}(\mathbf{w}+\boldsymbol{\delta}) - \mathbf{o}(\mathbf{w})\big]\right\|_F^2, \qquad M_{ij} = \frac{1}{\tau}\Big(1-\frac{1}{\tau}\Big)^{i-j}\mathbb{1}[i \geq j].$$

Since $\det \mathbf{M} = \tau^{-T} \neq 0$,

$$\mathcal{L}_{\mathrm{spk}}(\boldsymbol{\delta}) = 0 \quad \Longleftrightarrow \quad \mathbf{o}(\mathbf{w}+\boldsymbol{\delta}) = \mathbf{o}(\mathbf{w}).$$

Replacing the recurrent spike Jacobian by a constant-surrogate Jacobian:

$$g(v) \equiv c > 0, \qquad g'(v) = 0, \qquad \frac{\partial \mathbf{o}}{\partial \mathbf{w}} \approx \mathbf{J}_{\mathrm{SMP}} = c\mathbf{S}.$$

The corresponding linearization gives

$$\mathcal{L}_{\mathrm{spk}}(\boldsymbol{\delta}) \approx \left\|\mathbf{M}\mathbf{J}_{\mathrm{SMP}}\boldsymbol{\delta}^\top\right\|_F^2 = c^2\left\|\mathbf{M}\mathbf{S}\boldsymbol{\delta}^\top\right\|_F^2,$$

where $\mathbf{S}\boldsymbol{\delta}^\top \in \mathbb{R}^{T\times P}$ contracts the d axis. Positive rescaling leaves the OBS update unchanged; set $c = 1$. With $\widetilde{\mathbf{S}} \in \mathbb{R}^{d\times TP}$ obtained by applying $\mathbf{M}$ along the temporal axis of $\mathbf{S}$ and merging the T and P axes,

$$\mathcal{L}_{\mathrm{mem}}(\boldsymbol{\delta}) := \|\boldsymbol{\delta}\widetilde{\mathbf{S}}\|_2^2 = \tfrac{1}{2}\boldsymbol{\delta}\mathbf{H}\boldsymbol{\delta}^\top,$$
$$\mathbf{H} = \nabla_{\boldsymbol{\delta}}^2\mathcal{L}_{\mathrm{mem}} = 2\widetilde{\mathbf{S}}\widetilde{\mathbf{S}}^\top \in \mathbb{R}^{d\times d}.$$

Calibration averaging therefore yields the shared row Hessian

$$\boxed{\mathbf{H} = \frac{2}{N_s}\sum_{n=1}^{N}\widetilde{\mathbf{S}}_n\widetilde{\mathbf{S}}_n^\top, \qquad N_s = TNP.}$$

This Hessian is exact for the quadratic surrogate membrane potential (SMP), which approximates the discrete spike reconstruction objective.

A.2 Derivation of End-Balance Guidance

For the fixed temporal-mean readout, define

$$\mathbf{b} := \frac{1}{T}\sum_{t=1}^{T}\big(\widehat{\mathbf{O}}_L[t] - \mathbf{O}_L[t]\big), \qquad \widehat{\mathbf{z}} - \mathbf{z} = \mathbf{W}_{\mathbf{o}}\mathbf{b},$$
$$\mathcal{E} := \ell(\widehat{\mathbf{z}}, y) - \ell(\mathbf{z}, y),$$
$$\mathbf{b} = \mathbf{0} \Longrightarrow \widehat{\mathbf{z}} = \mathbf{z} \Longrightarrow \mathcal{E} = 0.$$

Perturb a single layer and suppress its index. For row $\mathbf{w}_r \in \mathbb{R}^{1\times d_{\mathrm{col}}}$ and a fixed calibration input $\mathbf{S}_n \in \mathbb{R}^{T\times d_{\mathrm{col}}\times P}$ with temporal slices $\mathbf{S}_n[t] \in \mathbb{R}^{d_{\mathrm{col}}\times P}$, define

$$I_{n,r}[t] = \mathbf{w}_r\mathbf{S}_n[t], \qquad \widehat{I}_{n,r}[t] = \widehat{\mathbf{w}}_r\mathbf{S}_n[t],$$
$$\boldsymbol{\delta}_r := \widehat{\mathbf{w}}_r - \mathbf{w}_r, \quad \delta I_{n,r}[t] = \boldsymbol{\delta}_r\mathbf{S}_n[t] \in \mathbb{R}^{1\times P}.$$

All derivatives below are evaluated at the dense model using the surrogate derivatives adopted during pretraining. The chain rule and BPTT (Wu et al., 2018) give

$$g_{n,r}[t] := \frac{\partial\ell(\mathbf{z}_n, y_n)}{\partial I_{n,r}[t]} \in \mathbb{R}^{1\times P}, \qquad \mathbf{u}_{n,r} := \sum_{t=1}^{T} g_{n,r}[t]\,\mathbf{S}_n[t]^\top = \frac{\partial\ell(\mathbf{z}_n, y_n)}{\partial\mathbf{w}_r},$$

and hence the surrogate first-order approximation is:

$$\mathcal{E}_n \approx \sum_r\sum_{t=1}^{T} g_{n,r}[t]\,\delta I_{n,r}[t]^\top = \sum_r \mathbf{u}_{n,r}\boldsymbol{\delta}_r^\top.$$

Define the empirical Fisher block and the signed flip contributions by

$$\mathbf{F}_r := \sum_{n=1}^{N}\mathbf{u}_{n,r}^\top\mathbf{u}_{n,r} \succeq 0, \qquad a_{n,r}[t] := g_{n,r}[t]\,\delta I_{n,r}[t]^\top.$$

Then

$$\boldsymbol{\delta}_r\mathbf{F}_r\boldsymbol{\delta}_r^\top = \sum_n(\mathbf{u}_{n,r}\boldsymbol{\delta}_r^\top)^2 = \sum_n\left(\sum_{t=1}^{T}a_{n,r}[t]\right)^2$$
$$= \sum_{n,t}a_{n,r}[t]^2 + 2\sum_n\sum_{t<t'}a_{n,r}[t]\,a_{n,r}[t'].$$

Opposite-signed contributions therefore cancel temporally before squaring. Notably, the cancellation concerns the surrogate first-order final balance response and does not imply exact terminal

balance. Though, we still notice significant balance improvement in intermediate layers, indicating the effectiveness of our strategy.

Let $\mathbf{H}$ be the calibration-averaged SMP Hessian from Appendix A.1, with $\mathcal{L}_{\mathrm{mem}}(\boldsymbol{\delta}_r) := \frac{1}{2}\boldsymbol{\delta}_r \mathbf{H} \boldsymbol{\delta}_r^\top$. For $\mathrm{tr}(\mathbf{F}_r) > 0$ and $\alpha \in [0, 1]$, set

$$\kappa_r := \frac{\mathrm{tr}(\mathbf{H})}{\mathrm{tr}(\mathbf{F}_r)}, \qquad \mathbf{H}_r(\alpha) := (1-\alpha)\mathbf{H} + \alpha\kappa_r \mathbf{F}_r.$$

Consequently,

$$\mathrm{tr}(\kappa_r \mathbf{F}_r) = \mathrm{tr}(\mathbf{H}_r(\alpha)) = \mathrm{tr}(\mathbf{H}), \qquad \mathbf{H}_r(0) = \mathbf{H},$$
$$\tfrac{1}{2}\boldsymbol{\delta}_r \mathbf{H}_r(\alpha) \boldsymbol{\delta}_r^\top = (1-\alpha)\mathcal{L}_{\mathrm{mem}}(\boldsymbol{\delta}_r) + \frac{\alpha\kappa_r}{2} \sum_n (\mathbf{u}_{n,r} \boldsymbol{\delta}_r^\top)^2.$$

Thus α reweights membrane reconstruction and end-loss sensitivity at fixed total trace. If $\mathrm{tr}(\mathbf{F}_r) = 0$, retain $\mathbf{H}_r(\alpha) = \mathbf{H}$. Substituting $\mathbf{H}_r(\alpha)$ for $\mathbf{H}$ in the original OBS pruning and compensation updates requires no explicit propagation or matching of discrete spike flips.

A.3 Optimality and Split-Half Estimation

Fix a layer and row and suppress their indices. Denote

$$\mathbf{G} = \kappa \mathbf{F}, \qquad \boldsymbol{\Sigma} = \mathbb{E}[\mathbf{G}], \qquad \mathbf{H}(\alpha) = (1-\alpha)\mathbf{H} + \alpha\mathbf{G},$$
$$\mathbf{E} = \mathbf{G} - \boldsymbol{\Sigma}, \qquad \mathbb{E}[\mathbf{E}] = \mathbf{0}, \qquad \beta = \|\mathbf{H} - \boldsymbol{\Sigma}\|_F^2, \qquad \sigma^2 = \mathbb{E}\|\mathbf{E}\|_F^2.$$

Since $\mathbb{E}\langle \mathbf{H} - \boldsymbol{\Sigma}, \mathbf{E} \rangle_F = 0$,

$$\begin{aligned} \mathrm{MSE}(\alpha) &:= \mathbb{E}\|\mathbf{H}(\alpha) - \boldsymbol{\Sigma}\|_F^2 \\ &= \mathbb{E}\|(1-\alpha)(\mathbf{H} - \boldsymbol{\Sigma}) + \alpha\mathbf{E}\|_F^2 \\ &= (1-\alpha)^2\beta + \alpha^2\sigma^2. \end{aligned}$$

For $\beta + \sigma^2 > 0$,

$$\mathrm{MSE}'(\alpha) = 2[(\beta+\sigma^2)\alpha - \beta], \qquad \mathrm{MSE}''(\alpha) = 2(\beta+\sigma^2) > 0,$$
$$\boxed{\alpha^\star = \arg\min_{\alpha \in [0,1]} \mathrm{MSE}(\alpha) = \frac{\beta}{\beta+\sigma^2}.}$$

The minimizer is unique and, when $\beta > 0$ and $\sigma^2 > 0$,

$$\mathrm{MSE}(\alpha^\star) = \frac{\beta\sigma^2}{\beta+\sigma^2} < \min\{\beta, \sigma^2\} = \min\{\mathrm{MSE}(0), \mathrm{MSE}(1)\}.$$

The same MSE bounds the quadratic metric error uniformly:

$$\mathbb{E}\left[\sup_{\|\boldsymbol{\delta}\|_2 \le 1} \left|\boldsymbol{\delta}(\mathbf{H}(\alpha) - \boldsymbol{\Sigma})\boldsymbol{\delta}^\top\right|^2\right] \le \mathrm{MSE}(\alpha).$$

The observable discrepancy estimates bias plus variance:

$$Q := \|\mathbf{H} - \mathbf{G}\|_F^2, \qquad \mathbb{E}[Q] = \mathbb{E}\|\mathbf{H} - \boldsymbol{\Sigma} - \mathbf{E}\|_F^2 = \beta + \sigma^2.$$

A separate variance estimate is therefore required. Split the i.i.d. calibration samples into two equal, disjoint subsets and write:

$$\mathbf{F}^{(k)} := \sum_{n \in \mathcal{D}^{(k)}} \mathbf{u}_n^\top \mathbf{u}_n = \overline{\mathbf{F}} + \mathbf{E}_k, \qquad k \in \{1, 2\},$$
$$\mathbb{E}[\mathbf{E}_k] = \mathbf{0}, \qquad \mathbf{E}_1 \perp\!\!\!\perp \mathbf{E}_2, \qquad \mathbb{E}\|\mathbf{E}_k\|_F^2 = \sigma_h^2.$$

The deployed Fisher is their sum, giving

$$\mathbf{G} = \kappa(\mathbf{F}^{(1)} + \mathbf{F}^{(2)}), \qquad \boldsymbol{\Sigma} = 2\kappa\overline{\mathbf{F}}, \qquad \mathbf{E} = \kappa(\mathbf{E}_1 + \mathbf{E}_2),$$
$$\mathbf{F}^{(1)} - \mathbf{F}^{(2)} = \mathbf{E}_1 - \mathbf{E}_2,$$
$$\sigma^2 = \kappa^2 \mathbb{E}\|\mathbf{E}_1 + \mathbf{E}_2\|_F^2 = 2\kappa^2\sigma_h^2 = \kappa^2 \mathbb{E}\|\mathbf{F}^{(1)} - \mathbf{F}^{(2)}\|_F^2.$$

Equal halves cancel the common signal; independence eliminates the noise cross-covariance. Their sum retains all calibration samples without additional backward passes.

Consequently,

$$\widehat{\sigma}^2 := \kappa^2 \|\mathbf{F}^{(1)} - \mathbf{F}^{(2)}\|_F^2, \qquad \mathbb{E}[\widehat{\sigma}^2] = \sigma^2,$$

$$\widehat{\beta} := [Q - \widehat{\sigma}^2]_+, \qquad \boxed{\widehat{\alpha} = \frac{\widehat{\beta}}{\widehat{\beta} + \widehat{\sigma}^2} = \left[1 - \frac{\widehat{\sigma}^2}{Q}\right]_0^1.}$$

The ratio applies when $Q > 0$; $[x]_+ := \max\{x, 0\}$ and $[x]_0^1 := \min\{1, \max\{0, x\}\}$. If $\mathrm{tr}(\mathbf{F}) = 0$ or $Q = 0$, set $\widehat{\alpha} = 0$ and retain $\mathbf{H}$.

The estimate above is formed for each output row r of a layer, giving $\widehat{\alpha}_r$. The deployed mixing weight is the layer mean $\widehat{\alpha} = \frac{1}{d_{\mathrm{row}}} \sum_r \widehat{\alpha}_r$, used together with the row-specific κ_r and $\mathbf{F}_r$ in $\mathbf{H}_r(\widehat{\alpha})$. Rows with $\mathrm{tr}(\mathbf{F}_r) = 0$ or $Q_r \leq \widehat{\sigma}_r^2$ have $\widehat{\alpha}_r = 0$ and are included in the mean. The mean is taken over the row estimates; it is not obtained by pooling the row statistics into a single layer-level Q and $\widehat{\sigma}^2$ before forming the ratio. When the per-row Fisher statistics of a layer exceed the memory budget, rows are partitioned into groups that share one Fisher block, and the mean is taken over the group estimates with equal weight regardless of group size.

A.4 Detailed Statistics

Objectives & Models & Datasets. To demonstrate the effectiveness and generalization of our analysis and FastSNN framework, we consider several different standard compression settings. Concretely, we consider unstructured pruning for given sparsity, global 2:4, 4:8, and 2:8 pruning, as well as uniform weight quantization. Each layer is pruned to a given constraint. Additionally, we also study some practical tasks requiring joint pruning and quantization when deploying on different accelerators. We work with variants of the following models and tasks : ResNet-series (Fang et al., 2021; Hu et al., 2024) and Transformer-series models (Fang et al., 2026; Zhou et al., 2024; Shi et al., 2024) for image classification on ImageNet (Deng et al., 2009) and CIFAR-100 (Krizhevsky et al., 2009), as well as neuromorphic recognition on CIFAR10-DVS Li et al. (2017). EMS-YOLO (Su et al., 2023) and SpikeYOLO (Luo et al., 2024) for object detection on COCO (Lin et al., 2014) and SpikingVGG16 for wireless sensing recognition on UT-HAR (Yang et al., 2023), indicated in Table 4. Also, since compression behavior depends on the neuron dynamics of each model, we use the neuron types and hyperparameters of the original implementations and list them in Table 5 for reproducibility. For all models on ImageNet and COCO2017, and for SEW-ResNet-DVS on CIFAR10-DVS, we use the officially released pretrained checkpoints for evaluation. We train the remaining models on CIFAR-100 and CIFAR10-DVS with our own training pipeline for the rest evaluations, where hyper parameters of our training pipeline are shown in Table 6.

Experimental Setups. We use a small, randomly sampled subset of each training set for calibration. For instance, we use approximately 0.1% of the ImageNet training data. CIFAR and ImageNet samples use the standard crop and horizontal-flip augmentation, while no augmentation is applied to the other tasks. After compression, we reset and re-estimate BN running statistics using 64 batches of 64 calibration samples each. All methods use the same calibration data, compressed layers, and BN recalibration protocol. For pruning, we compare with SBC (Shi et al., 2025), Wanda (Sun et al., 2024), and OSSCAR (Meng et al., 2024b). For weight-only quantization, we compare with round-to-nearest (RTN), GPTQ (Frantar et al., 2022), NeuronQuant (Li et al., 2025), and SBC equipped with OBQ solver. In joint sparsity-quantization experiments, all methods first prune the model and then quantize its remaining weights while keeping the pruning mask fixed. For BOP (number of bits times flops)-constrained mixed-precision experiments, we adapt the FLOPs-constrained protocol of OBC (Frantar & Alistarh, 2022) by using BOPs as the resource measure and SPDY (Frantar et al., 2021) for per-layer configuration assignment. All experiments are conducted on an NVIDIA RTX 4090 GPU. For layers whose per-row Fisher statistics exceed the memory budget, rows are partitioned into groups sharing one Fisher block (Appendix A.3); all CIFAR experiments use exact per-row Fisher.

Calibration Settings. We randomly sample a fixed labeled subset from each training split using default seed 42. Unless otherwise stated, all models on the same dataset use the calibration budget

Table 4: Detailed experimental settings.

Datasets	Model	Compression Type
ImageNet	SEW-ResNet/MS-ResNet MaxFormer/QK-Former SpikingResFormer	N:M pruning Unstructured Pruning n-bit Quantization
CIFAR-100	SEW-ResNet/MS-ResNet MaxFormer/QK-Former SpikingResFormer	N:M pruning Unstructured Pruning n-bit Quantization
CIFAR10-DVS	SEW-ResNet/MS-ResNet MaxFormer/QK-Former SpikingResFormer	N:M pruning Unstructured Pruning n-bit Quantization
COCO2017	EMS-YOLO SpikeYOLO	Mixed quant + 2:4 pruning
UT-HAR	SpikingVGG16	Block Pruning

Table 5: Spiking-neuron settings of the evaluated models. † $V_{\text{th}} = 0.5$ for the attention LIF. ‡Integer spikes $S \in \{0, \ldots, 4\}$, no explicit threshold.

Model	Neuron	Time constant	V_{th}	T
SEW-ResNet (ImageNet / CIFAR-100)	IF	–	1.0	4
SEW-ResNet (CIFAR10-DVS)	LIF	$\tau = 2$	1.0	16
MS-ResNet	MS-LIF	$\tau = 4$	0.5	6
MaxFormer / MS-QKFormer	LIF	$\tau = 2$	$1.0^{\dagger}$	4
SpikingResformer	LIF	$\tau = 2$	1.0	4 / 6
EMS-YOLO	MS-LIF	$\tau = 4$	0.5	4
SpikeYOLO	I-LIF	$\tau = 4$	$D = 4^{\ddagger}$	4
SpikingVGG16	LIF	$\tau = 2$	1.0	16

in Table 7. CIFAR-100 and ImageNet inputs use random cropping, horizontal flipping, and dataset-specific normalization.

To account for activation distribution shifts after compression, we recalibrate the running statistics of the remaining batch-normalization (BN) layers. All weights and BN affine parameters remain fixed.

1. Take the first 64 minibatches from the cached calibration data, or all available minibatches if fewer. Discard the labels.
2. Keep the model in evaluation mode and enable training mode only for the remaining BN layers. Reset their running means, running variances, and batch counters to $\mathbf{0}$, $\mathbf{1}$, and 0, respectively.
3. With gradient recording disabled, process each minibatch through the compressed model. At the k-th update of BN layer ℓ, compute

$$\begin{aligned} \boldsymbol{\mu}_k^{\ell} &= (1-m)\boldsymbol{\mu}_{k-1}^{\ell} + m\widehat{\boldsymbol{\mu}}_k^{\ell}, \\ \mathbf{v}_k^{\ell} &= (1-m)\mathbf{v}_{k-1}^{\ell} + m\widehat{\mathbf{v}}_k^{\ell}, \qquad m = 0.1, \end{aligned} \tag{19}$$

where $\widehat{\boldsymbol{\mu}}_k^{\ell}$ and $\widehat{\mathbf{v}}_k^{\ell}$ are the channelwise batch mean and unbiased batch variance, respectively. The index k counts BN updates, including repeated invocations across time steps. Reset the neuronal states after each minibatch.

Table 6: Training settings of the models trained with our pipeline. SpikingResFormer is fine-tuned from its official ImageNet checkpoint, following its official transfer learning steps (Shi et al., 2024).

Datasets	Model	T	Epochs	Batch	Optimizer (learning rate)
CIFAR-100	SEW-ResNet	4	100	100	SGD (0.1)
	MS-ResNet	6	100	100	SGD (0.1)
	MaxFormer/QK-Former	4	400	64	AdamW (1.5e-3)
	SpikingResFormer	4	150	128	AdamW (1e-4)
CIFAR10-DVS	MS-ResNet	20	200	16	SGD (1e-3)
	MaxFormer	16	150	64	AdamW (5e-3)
	QK-Former	16	96	32	AdamW (5e-3)
	SpikingResFormer	10	150	128	AdamW (1e-4)
UT-HAR	SpikingVGG16	4	100	16	AdamW (5e-4)

Table 7: Calibration Statistics for three major datasets we evaluated.

Dataset	Labeled samples	Fraction
CIFAR-100	2048	4.096%
ImageNet	1024	0.080%
CIFAR10-DVS	1024	11.378%

4. Restore all modules to evaluation mode and retain the updated running statistics for inference.

We note that it is critical to apply the statistics correction already while computing the compressed means and variances in order to properly account for compounding distribution shifts.

Sensitivity to Random States. For a fixed pretrained model, randomness in FastSNN primarily arises from calibration-set sampling and data augmentation. The augmented inputs are cached and reused for curvature estimation and BN recalibration. Once these calibration batches and their order are fixed, the compression procedure introduces no additional calibration sampling or stochastic optimization. To assess seed sensitivity, we perform 4-bit quantization of SEW-ResNet-32 and 2:4 pruning of SEW-ResNet-110 on CIFAR-100, using 5 random seeds for each setting. Table 8 reports the mean and sample standard deviation of accuracy. The standard deviations are below

Table 8: Typical random variation of FastSNN results.

SEW-ResNet-32 (4-bit)	SEW-ResNet-110 (2:4)
62.09 ± 0.29	66.19 ± 0.25

0.30% in both settings, with observed peak-to-peak variations of 0.75% and 0.67%, respectively. These results indicate limited sensitivity to random seeds for the evaluated quantization and pruning configurations.

Timing Information. Table 9 compares the runtimes of RTN, NeuronQuant, GPTQ, SBC, and FastSNN for 4-bit weight quantization of SEW-ResNet-18 on ImageNet. All times are reported in seconds and include BN recalibration where performed. SBC uses folded BN without recalibration; the other methods retain BN and recalibrate its running statistics after compression. FastSNN completes quantization in 196 s, compared with 76.7 s for SBC and 57.1 s for GPTQ. RTN and NeuronQuant require 8.6 s and 11.1 s, respectively. Profiling the FastSNN run attributes 125.9 s to Fisher estimation (where backpropagation is included), 24.9 s to SMP Hessian collection, 1.6 s to computing the mixing coefficients, and 43.8 s to the compression solve and BN recalibration. Fisher estimation therefore accounts for approximately 64% of the runtime, while computing the mixing coefficients contributes less than 1%. The main optimization target in this run is thus Fisher estimation.

Table 9: Runtimes of post-training quantization methods on SEW-ResNet-18, in seconds (s).

Model	RTN	NeuronQuant	GPTQ	SBC	FastSNN
SEW-ResNet-18	8.6	11.1	57.1	76.7	196

Table 10 reports FastSNN runtimes for SEW-ResNet-18, MaxFormer-10-384, and SpikingResFormer-Ti on ImageNet, under 4-bit quantization, unstructured pruning at 50% sparsity, and 2:4 structured pruning. Quantization takes 196–375 s across the three models and is faster than either pruning configuration in every case. The two pruning configurations have similar runtimes within each model, differing by less than 1.2%. Pruning requires 2314–2319 s for SEW-ResNet-18, 724–732 s for MaxFormer-10-384, and 542–548 s for SpikingResFormer-Ti. The relative cost of quantization and pruning is therefore strongly model dependent, with the largest gap observed for SEW-ResNet-18.

Table 10: Runtimes of FastSNN for different models and compression type, in seconds.

Model	Quant	Unstr	2:4
SEW-ResNet-18	196	2319	2314
MaxFormer-10-384	375	724	732
SpikingResFormer-Ti	355	542	548

A.5 Details of Deployment

Hardware Features. The neuromorphic chip we use (Figure 3.), Darwin3 (Ma et al., 2024), comprises neuromorphic cores connected by an XY-mesh network-on-chip (NoC). Each core stores neuronal states and synaptic connections locally and provides hardware support for neuron updates. When a neuron fires, its outgoing connection table identifies the destination cores, which receive the spike as a packet and retrieve the corresponding postsynaptic targets and weights. Consequently, pruning can reduce local synaptic storage and processing, while eliminating all connections from a source neuron to a destination core can additionally reduce spike traffic. This organization is also found in established platforms such as Intel Loihi (Davies et al., 2018). Our pruning principle is therefore applicable in principle to Loihi, with the grouping and deployment mapping adapted to its connectivity format.

Sparsity Pattern. We evaluate CPU inference on an AMD Ryzen 9 7950X CPU using the block-sparsity pattern considered in OBC (Frantar & Alistarh, 2022), with a block size of 4. Each output row of the weight matrix is partitioned into contiguous 1×4 blocks, whose weights are jointly retained or pruned. For convolutional layers, blocks are defined after arranging the weights into the matrix layout used for deployment. This pattern preserves layer dimensions while allowing sparse inference kernels to skip zero blocks.

For Darwin3, we use a two-level sparsity pattern targeting both synaptic processing and communication. Consider a convolutional weight tensor $\mathbf{W} \in \mathbb{R}^{C_{\text{out}} \times C_{\text{in}} \times k_h \times k_w}$. We partition input and output channels into groups of $G = 8$ and $g = 16$, respectively. At the coarse level, each $g \times G \times k_h \times k_w$ block is retained or pruned jointly, removing all kernel connections between an input-channel group and an output-channel group. Each output group retains the same number of input groups to distribute the remaining connectivity evenly. These removals reduce synaptic work and can eliminate packet destinations when no connection to the corresponding core remains.

Within the surviving coarse blocks, fine pruning operates on $1 \times G \times 1 \times 1$ blocks. Thus, G adjacent input channels share a pruning decision for each output channel and kernel position. Their retained weights remain distinct, but the common connectivity pattern permits sharing postsynaptic index lists and reduces the number of synaptic entries processed. For fully connected layers, the corresponding fine blocks contain G consecutive input weights per output neuron. During deployment, pruned synaptic entries and empty routing destinations are removed. Dense and sparse variants use the same neuron-to-core placement, allowing our evaluations to isolate reductions in synaptic and communication workloads.

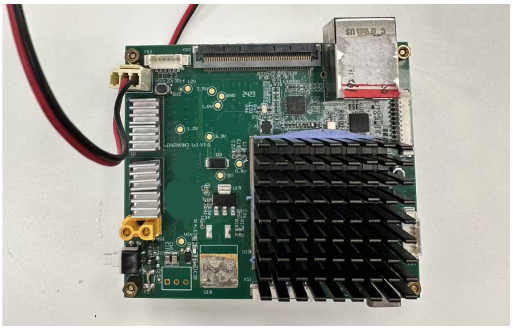


Figure 3: Darwin3 neuromorphic board.

A.6 More Experimental Results.

We compare FastSNN with training-based compression methods on CIFAR-100, CIFAR10-DVS, and DVS128 Gesture. Table 11 compares FastSNN with UniSparse (Shi et al., 2026) and Spik4Lite (She et al., 2026) at sparsity of 50%, 75%, and 87.5%. FastSNN uses unstructured weight pruning in these experiments.

Table 11: Accuracy (%) of UniSparse, Spik4Lite, and FastSNN on different sparsity.

	UniSparse			Spik4Lite			FastSNN		
Model	50%	75%	87.5%	50%	75%	87.5%	50%	75%	87.5%
CIFAR-100									
MS-ResNet-32	**64.26**	**62.51**	**59.48**	60.23	50.22	43.09	61.74	54.77	27.40
MS-ResNet-56	**67.22**	**65.75**	**64.00**	64.22	56.98	48.33	63.93	58.83	34.21
MS-ResNet-110	**64.15**	**62.66**	**61.41**	63.49	60.75	51.66	61.95	57.94	34.78
SEW-ResNet-32	**64.14**	**61.43**	**54.57**	58.76	50.49	43.05	61.71	57.79	40.59
SEW-ResNet-56	**65.80**	**62.21**	**57.22**	63.17	57.79	48.08	64.87	61.84	45.92
SEW-ResNet-110	66.07	**63.86**	**59.23**	66.58	63.39	55.30	**67.06**	63.51	46.93
SpikingResFormer-S	61.94	60.45	57.94	50.90	46.64	25.90	**83.66**	**81.16**	**71.29**
SpikingResFormer-Ti	60.92	59.09	55.48	44.16	47.25	33.23	**81.90**	**78.34**	**67.25**
MS-QKFormer	**75.97**	**75.41**	**74.04**	69.24	57.68	29.59	73.76	71.98	63.23
MaxFormer	**76.37**	**75.44**	**73.57**	70.26	63.88	46.89	74.69	72.83	65.55
CIFAR10-DVS									
MS-ResNet	**67.90**	**65.60**	**66.50**	66.80	61.70	50.20	65.50	60.20	45.80
SEW-ResNet	71.60	71.50	70.00	69.00	66.60	54.50	**72.20**	**72.40**	**70.70**
SpikingResFormer-Ti	68.90	67.50	67.60	60.70	60.20	49.80	**77.20**	**75.00**	**71.50**
SpikingResFormer-S	69.70	68.60	**69.00**	63.20	59.30	55.40	**74.00**	**71.40**	67.60
MaxFormer	74.30	71.90	69.80	74.50	72.70	70.20	**78.10**	**75.40**	**72.90**
MS-QKFormer	74.30	71.90	69.80	74.30	72.70	70.20	**76.20**	**75.50**	**73.10**
DVS128 Gesture									
MS-ResNet	**95.83**	**95.83**	**95.83**	72.22	72.57	66.67	76.39	77.08	71.53
SEW-ResNet	90.97	90.97	91.67	86.11	82.29	76.74	**95.49**	**95.49**	**93.75**
SpikingResFormer-Ti	**96.53**	**96.18**	**95.14**	86.81	84.72	77.43	88.89	86.81	81.25
SpikingResFormer-S	**94.44**	**95.49**	**96.18**	89.58	88.19	81.25	87.50	87.50	82.29
MaxFormer	**96.53**	**96.88**	**96.88**	94.79	93.75	89.93	95.83	95.83	94.44
MS-QKFormer	**96.53**	**96.88**	**96.88**	95.83	94.10	90.26	95.83	95.83	94.44

FastSNN achieves accuracy close to, and in some settings higher than, the training-based baselines without additional sparse training or fine-tuning. For SEW-ResNet-110 on CIFAR-100 at 75% sparsity, FastSNN reaches 63.51% accuracy, within 0.35% of UniSparse (63.86%) and slightly above Spik4Lite (63.39%). At the same sparsity on CIFAR10-DVS, MaxFormer achieves 75.40% with FastSNN, compared with 72.70% for Spik4Lite and 71.90% for UniSparse. These results demonstrate that calibration-based compression can preserve competitive accuracy without incorporating sparsity constraints into training, although larger gaps remain for some models at high sparsity.

Table 12 compares FastSNN with SpikeNM (Ye et al., 2025) under $2{:}4$, $4{:}8$, $2{:}8$, and $1{:}8$ patterns. Each comparison uses the same N:M pattern.

FastSNN also attains competitive accuracy under explicit N:M constraints. For MaxFormer on CIFAR10-DVS with $2{:}8$ pruning, FastSNN achieves 75.20%, within 0.80% of SpikeNM (76.00%). For SEW-ResNet on the same dataset under $2{:}4$ pruning, FastSNN reaches 72.40%, exceeding SpikeNM (71.30%) by 1.10%. The advantage is not uniform: SpikeNM remains stronger in several aggressive pruning settings, including some $1{:}8$ configurations. Nevertheless, these comparisons show that PTC can approach or exceed the accuracy of training-based N:M pruning in selected settings, providing a practical alternative when additional training or fine-tuning is unavailable.

Although FastSNN and the training-based methods reach comparable accuracy in many settings, their computational costs differ by orders of magnitude. Table 13 reports the GPU time needed to produce each compressed model. FastSNN compresses SEW-ResNet-56 on CIFAR-100 in under 3 minutes and the two CIFAR10-DVS Transformers in about 8 to 10 minutes, for both unstructured and $2{:}4$ sparsity. The training-based baselines take between 5.5 and 22.2 hours on the same models, so FastSNN is roughly $33\times$ to $220\times$ faster. This gap in compression cost, more than any difference in accuracy, is the main advantage of FastSNN over training-based compression.

Additional post-training results. Tables 14 and 15 extend Tables 1 and 2 to additional ResNet- and Transformer-series models, and Table 16 reports the extreme $1{:}8$ N:M pattern. We summarize each table by the accuracy gain of FastSNN over the strongest baseline in each setting.

Pruning. Table 14 follows the trend in Section 4. At moderate sparsity, FastSNN stays on par with the strongest baseline, with average gains of 0.32%, 0.03%, and -0.14% at $2{:}4$, $4{:}8$, and 50% sparsity over the 22 model–dataset settings. Under aggressive compression, the average gain rises to

Table 12: Accuracy (%) of SpikeNM and FastSNN under N:M pruning patterns.

	SpikeNM				FastSNN			
Model	2:4	4:8	2:8	1:8	2:4	4:8	2:8	1:8
CIFAR-100								
MS-ResNet-32	58.31	**61.21**	**58.75**	**47.07**	**59.83**	60.65	50.04	13.45
MS-ResNet-110	**67.53**	**65.93**	**61.71**	**56.16**	61.04	61.88	55.34	18.14
SEW-ResNet-32	**63.98**	**62.88**	**56.76**	**48.14**	60.71	61.79	55.23	25.18
SEW-ResNet-56	**66.00**	**66.42**	**62.36**	**53.99**	63.95	64.67	58.99	29.03
SEW-ResNet-110	**67.69**	**68.36**	**62.86**	**58.02**	65.94	66.54	60.82	30.19
SpikingResFormer-S	74.04	72.67	67.44	**65.45**	**82.94**	**83.47**	**78.07**	58.23
SpikingResFormer-Ti	71.11	70.45	66.04	**63.54**	**81.38**	**81.47**	**75.41**	50.84
MS-QKFormer	69.12	69.22	67.56	**68.31**	**73.47**	**73.40**	**70.31**	50.86
MaxFormer	70.86	70.33	69.64	**69.34**	**74.47**	**74.66**	**71.03**	57.54
CIFAR10-DVS								
MS-ResNet	**62.70**	63.10	**66.50**	**66.80**	62.10	**63.50**	56.80	36.30
SEW-ResNet	71.30	71.40	64.20	57.40	**72.40**	**72.80**	**71.20**	**66.00**
SpikingResFormer-Ti	69.80	68.80	66.10	64.50	**76.20**	**76.90**	**74.10**	**65.90**
SpikingResFormer-S	70.20	69.30	68.50	**64.70**	**74.50**	**73.80**	**71.60**	63.90
MaxFormer	**79.40**	**77.90**	**76.00**	**72.70**	75.90	75.70	75.20	68.90
MS-QKFormer	**79.40**	**77.90**	**76.00**	**72.70**	75.80	76.00	73.20	67.10
DVS128 Gesture								
MS-ResNet	70.83	69.09	**79.17**	**78.47**	**75.69**	**76.04**	72.22	68.40
SEW-ResNet	93.05	88.89	82.29	74.31	**96.18**	**95.14**	**95.83**	**93.06**
SpikingResFormer-Ti	**95.13**	**94.44**	**92.01**	**92.01**	88.19	86.46	81.94	70.83
SpikingResFormer-S	**95.48**	**95.83**	**94.79**	**90.28**	89.24	86.81	84.38	76.04
MaxFormer	**98.61**	**97.92**	**97.92**	**97.22**	95.14	96.18	95.14	94.10
MS-QKFormer	**98.61**	**97.92**	**97.92**	**97.22**	95.14	96.18	95.14	94.10

Table 13: GPU time cost to produce a compressed model.

Method (model)	Dataset	Unstr.	2:4
FastSNN (SEW-ResNet-56)	CIFAR-100	162 s	168 s
UniSparse (SEW-ResNet-56)	CIFAR-100	9.9 h	–
Spik4Lite (SEW-ResNet-56)	CIFAR-100	6.1 h	–
SpikeNM (SEW-ResNet-56)	CIFAR-100	–	6.2 h
FastSNN (MaxFormer-DVS)	CIFAR10-DVS	10 m10 s	10 m15 s
UniSparse (MaxFormer-DVS)	CIFAR10-DVS	17.2 h	–
Spik4Lite (MaxFormer-DVS)	CIFAR10-DVS	5.6 h	–
SpikeNM (MaxFormer-DVS)	CIFAR10-DVS	–	7.2 h
FastSNN (SpikingResFormer-Ti)	CIFAR10-DVS	8 m20 s	8 m25 s
UniSparse (SpikingResFormer-Ti)	CIFAR10-DVS	22.2 h	–
Spik4Lite (SpikingResFormer-Ti)	CIFAR10-DVS	5.5 h	–
SpikeNM (SpikingResFormer-Ti)	CIFAR10-DVS	–	16.3 h

6.48% at 2:8 and 7.98% at 87.5%, and FastSNN is best in 18 and 16 of the 22 settings, respectively. The improvement is largest for ResNet-series models, where FastSNN is best in all 10 settings at both levels, with average gains of 13.42% at 2:8 and 16.21% at 87.5%; for example, MS-ResNet-104 on ImageNet retains 51.13% at 87.5% sparsity, compared with 17.40% for the strongest baseline. On Transformer-series models, the average gains are smaller, 0.68% and 1.13%, and OSSCAR or SBC remains stronger for some models.

Quantization. At 4 bits, all methods stay close to the dense accuracy, and FastSNN is within 0.39% of the strongest baseline on average. At 3 bits, it is best in 10 of 20 settings and exceeds GPTQ in 18 of 20 settings, by 1.52% on average. At 2 bits, its advantage concentrates on ResNet-series models, where it is best in 9 of 10 settings with an average gain of 1.74%, reaching 62.58% for MS-ResNet-104 on ImageNet compared with 57.07% for GPTQ. For Transformer-series models, GPTQ or SBC with the OBQ solver is slightly better at low bit widths, by 1.02% on average at 2 bits.

Extreme 1:8 *sparsity.* Table 16 keeps only one of every eight weights. Most baselines collapse on ResNet-series models; for MS-ResNet-34 on ImageNet, the strongest baseline reaches 7.93%, whereas FastSNN retains 34.64%. FastSNN is best in all 16 ResNet settings, with an average gain

Table 14: Post-training *pruning* results: (a) N:M structured sparsity, (b) unstructured sparsity. Best results are in **bold**, second best are underlined.

(a) N:M sparsity

Model	Dense	Wanda			OSSCAR			SBC			**Ours**		
		2:4	4:8	2:8	2:4	4:8	2:8	2:4	4:8	2:8	2:4	4:8	2:8
ImageNet													
MS-ResNet-34	69.42	62.59	64.93	29.52	66.28	67.51	50.10	67.56	68.39	50.63	**69.03**	**69.14**	**65.12**
MS-ResNet-104	74.24	65.05	67.87	17.64	70.00	71.14	35.32	72.05	72.69	46.68	**72.84**	**73.26**	**66.50**
SEW-ResNet-18	62.82	53.06	56.27	21.45	57.99	59.54	39.54	60.76	61.39	46.60	**61.63**	**62.16**	**56.46**
SEW-ResNet-50	67.53	56.08	59.78	16.52	64.03	65.15	45.90	63.66	64.97	33.11	**64.95**	**65.46**	**52.26**
SEW-ResNet-101	68.57	58.43	61.67	18.55	65.07	66.10	44.61	63.83	65.85	27.68	**65.55**	**66.59**	**52.25**
SEW-ResNet-152	69.10	57.66	60.96	14.56	65.62	66.45	43.14	63.56	65.94	21.84	**65.85**	**66.66**	**50.68**
SpikingResFormer-Ti	74.26	64.67	67.56	36.59	69.31	70.58	53.01	**69.88**	**71.37**	48.33	69.26	70.63	**54.30**
SpikingResFormer-M	77.07	71.79	73.25	49.01	74.80	75.55	62.64	**74.89**	**75.86**	59.09	74.61	75.18	**64.57**
SpikingResFormer-L	78.44	73.60	74.99	52.26	76.16	76.64	65.66	**76.46**	**76.80**	**69.63**	76.01	76.63	67.86
MaxFormer-10-384	77.54	72.60	73.89	52.15	74.26	75.21	60.15	73.18	74.76	45.70	**74.51**	**75.27**	**63.20**
MaxFormer-10-512	78.68	74.99	76.24	58.85	76.49	77.21	65.30	76.67	**77.74**	54.65	**76.88**	77.45	**67.52**
MS-QKFormer-384	76.26	71.52	73.18	51.30	**73.29**	**74.29**	59.95	71.39	73.05	45.64	73.07	74.20	**60.47**
CIFAR-100													
MS-ResNet-32	61.75	51.21	54.30	20.81	55.34	57.50	32.64	56.71	59.28	30.50	**59.83**	**60.65**	**50.04**
MS-ResNet-110	61.75	54.15	56.53	26.85	58.04	58.65	39.29	59.53	60.87	42.94	**61.04**	**61.88**	**55.34**
SEW-ResNet-32	62.09	51.88	56.03	22.40	55.46	58.44	34.75	59.78	60.85	34.23	**60.71**	**61.79**	**55.23**
SEW-ResNet-110	67.45	60.30	62.50	34.51	62.33	64.25	42.57	65.07	65.78	37.62	**65.94**	**66.54**	**60.82**
SpikingResFormer-Ti	83.56	78.27	80.15	59.91	80.96	81.61	71.15	81.17	**81.94**	74.40	**81.38**	81.47	**75.41**
SpikingResFormer-S	84.42	80.73	82.13	67.44	82.45	83.27	75.48	**82.97**	**83.61**	**78.25**	82.94	83.47	78.07
MS-QKFormer	74.49	72.21	73.28	66.24	73.34	73.37	70.06	**73.76**	**73.89**	69.28	73.47	73.40	**70.31**
CIFAR10-DVS													
SpikingResFormer-S	74.30	69.40	70.90	52.10	**75.50**	73.40	67.50	74.10	**75.50**	70.10	74.50	73.80	**71.60**
MS-QKFormer	76.90	76.60	76.50	73.50	**76.70**	**76.80**	74.00	74.00	75.10	**74.50**	75.80	76.00	73.20
MaxFormer	78.70	75.40	76.10	73.00	75.80	76.00	**75.50**	**76.90**	**76.70**	73.40	75.90	75.70	75.20

(b) Unstructured sparsity

Model	Dense	Wanda			OSSCAR			SBC			**Ours**		
		50%	75%	87.5%	50%	75%	87.5%	50%	75%	87.5%	50%	75%	87.5%
ImageNet													
MS-ResNet-34	69.42	67.19	47.25	8.40	69.02	62.92	39.59	69.16	62.23	28.75	**69.18**	**67.52**	**55.95**
MS-ResNet-104	74.24	70.90	37.38	1.24	73.42	62.59	17.40	73.77	64.45	13.89	**73.84**	**70.14**	**51.13**
SEW-ResNet-18	62.82	59.35	36.50	8.04	61.95	53.11	30.52	62.15	54.95	27.34	**62.41**	**59.42**	**46.57**
SEW-ResNet-50	67.53	62.97	29.38	1.91	66.11	53.74	16.11	66.25	48.35	6.57	**66.38**	**58.01**	**32.70**
SEW-ResNet-101	68.57	64.18	30.43	1.69	67.24	54.68	11.61	**67.62**	46.24	3.32	67.42	**58.43**	**29.96**
SEW-ResNet-152	69.10	64.16	26.00	1.13	**67.74**	54.16	10.56	67.73	41.57	2.41	67.41	**57.97**	**29.66**
SpikingResFormer-Ti	74.26	70.16	48.03	9.55	71.87	60.56	25.46	**72.61**	60.77	20.07	71.79	**62.31**	**31.63**
SpikingResFormer-M	77.07	75.13	59.24	16.34	76.14	68.15	36.50	**76.31**	**71.12**	**50.94**	76.05	69.77	46.51
SpikingResFormer-L	78.44	75.74	57.18	12.59	76.84	66.54	32.59	**77.22**	**73.06**	**56.49**	76.91	72.25	50.43
MaxFormer-10-384	77.54	75.25	60.27	22.76	76.13	66.01	35.85	**76.22**	61.77	12.51	76.21	**68.43**	**44.32**
MaxFormer-10-512	78.68	77.32	65.17	29.34	77.97	69.84	43.14	**78.61**	68.30	21.79	77.98	**72.06**	**52.87**
MS-QKFormer-384	76.26	74.23	59.23	26.00	75.17	65.79	39.12	74.52	59.41	12.32	**75.18**	**66.07**	**39.89**
CIFAR-100													
MS-ResNet-32	61.75	57.76	30.84	6.54	60.61	48.32	19.67	60.29	41.74	9.37	**61.74**	**54.77**	**27.40**
MS-ResNet-110	61.75	58.66	36.09	9.46	61.25	52.14	25.00	61.62	51.08	11.56	**61.95**	**57.94**	**34.78**
SEW-ResNet-32	62.09	58.22	34.23	6.94	61.32	51.08	25.96	61.22	48.38	7.38	**61.71**	**57.79**	**40.59**
SEW-ResNet-110	67.45	64.51	45.89	14.13	66.79	59.08	37.17	66.55	52.29	9.59	**67.06**	**63.51**	**46.93**
SpikingResFormer-Ti	83.56	81.35	68.54	34.30	**82.59**	76.35	54.06	82.06	**78.73**	65.13	81.90	78.34	**67.25**
SpikingResFormer-S	84.42	83.11	73.45	42.48	83.75	79.39	59.19	**83.83**	80.61	69.57	83.66	**81.16**	**71.29**
MS-QKFormer	74.49	74.01	68.31	51.82	73.45	71.79	**64.65**	**74.24**	71.93	56.26	73.76	**71.98**	63.23
CIFAR10-DVS													
SpikingResFormer-S	74.30	74.70	63.00	39.70	75.30	70.90	61.40	**75.70**	**72.90**	**67.70**	74.00	71.40	67.60
MS-QKFormer	76.90	76.20	74.00	68.90	**76.70**	**75.70**	**75.10**	76.40	74.40	68.90	76.20	75.50	73.10
MaxFormer	78.70	77.60	73.10	68.00	77.00	**76.70**	**74.30**	76.40	73.50	68.20	**78.10**	75.40	72.90

Table 15: Post-training weight *quantization* results. Best results are in **bold**, second best are underlined.

		RTN			GPTQ			NeuronQuant			SBC			**Ours**		
Model	Dense	4-bit	3-bit	2-bit	4-bit	3-bit	2-bit	4-bit	3-bit	2-bit	4-bit	3-bit	2-bit	4-bit	3-bit	2-bit
ImageNet																
MS-ResNet-34	69.42	67.69	58.12	42.94	68.95	65.72	58.48	67.50	54.14	27.39	68.87	67.00	47.36	**69.05**	**67.59**	**60.22**
MS-ResNet-104	74.24	72.17	59.56	41.05	73.50	70.45	57.07	71.95	51.30	9.14	**73.88**	72.21	37.90	73.70	**72.46**	**62.58**
SEW-ResNet-18	62.82	61.02	48.53	21.37	62.32	59.08	51.46	59.77	37.32	0.38	**62.60**	61.12	47.12	62.57	**61.40**	**54.19**
SEW-ResNet-50	67.53	66.12	57.59	34.68	67.02	63.23	56.83	65.89	52.18	15.20	67.14	65.29	42.56	**67.25**	**65.53**	**57.82**
SEW-ResNet-101	68.57	67.38	59.46	37.24	68.06	64.55	57.70	66.93	54.81	12.24	**68.48**	**66.50**	44.34	68.13	66.28	**58.93**
SEW-ResNet-152	69.10	67.82	60.80	37.19	68.44	65.35	**58.29**	67.54	55.76	3.52	68.48	66.41	41.39	**68.61**	**66.97**	58.05
SpikingResFormer-Ti	74.26	70.51	55.56	48.60	71.76	63.50	60.61	70.37	48.00	24.89	**73.19**	**69.05**	**61.77**	72.16	67.87	60.41
SpikingResFormer-S	75.88	74.23	61.80	57.99	75.30	68.95	66.12	73.99	55.99	31.91	**75.35**	72.25	**67.79**	75.01	**72.37**	66.67
SpikingResFormer-M	77.07	75.71	65.62	62.34	76.40	71.93	70.80	75.63	61.54	23.68	**76.76**	**74.46**	**71.72**	76.32	74.28	70.28
MaxFormer-10-384	77.54	76.05	71.00	67.51	76.51	73.30	**71.36**	75.89	69.98	60.92	**77.02**	**74.93**	52.99	76.56	74.40	69.98
MaxFormer-10-512	78.68	78.07	73.15	70.97	78.37	75.96	**74.07**	78.06	73.08	66.56	**79.21**	**77.69**	62.20	78.51	76.99	73.13
MaxFormer-10-768	82.21	81.09	78.13	76.17	81.42	79.78	**78.64**	81.06	77.33	71.29	**81.80**	**80.64**	67.40	81.30	80.28	77.81
CIFAR-100																
MS-ResNet-56	64.60	63.92	60.39	53.55	**64.74**	63.11	57.64	63.97	59.60	49.19	64.71	62.99	54.87	64.59	**63.98**	**59.71**
MS-ResNet-110	61.75	61.57	57.70	51.52	61.91	61.12	57.01	61.09	57.96	46.39	61.78	60.44	52.29	**62.11**	**61.81**	**57.40**
SEW-ResNet-32	62.09	61.33	57.59	50.05	62.20	60.56	56.21	61.88	56.49	44.00	62.18	61.32	53.38	**62.43**	**61.52**	**58.10**
SEW-ResNet-110	67.45	66.70	64.44	57.07	**67.45**	66.06	62.37	66.60	63.15	48.05	67.09	65.78	58.55	67.43	**66.36**	**63.44**
SpikingResFormer-S	84.42	83.50	76.05	72.67	**84.20**	80.35	78.62	83.20	72.64	60.01	83.70	**82.29**	**79.10**	84.03	82.09	78.73
MaxFormer	75.78	74.49	71.58	67.51	75.22	74.29	**72.35**	74.60	70.93	62.94	**75.82**	**75.11**	70.17	74.66	74.00	70.67
CIFAR10-DVS																
SpikingResFormer-S	74.30	73.50	61.60	61.80	**74.90**	69.80	70.90	73.90	58.30	45.40	74.70	**72.60**	61.30	73.80	71.50	**71.60**
MS-QKFormer	76.90	76.20	76.40	74.20	76.30	75.80	**76.90**	**76.70**	**76.70**	74.90	76.50	75.80	73.10	74.70	75.70	75.10

of 15.92% and a maximum of 26.71%. On Transformer-series models it is best in 11 of 16 settings, while SBC or OSSCAR remains stronger for SpikingResFormer-M and -L on ImageNet, MS-QKFormer on CIFAR-100 and CIFAR10-DVS, and MaxFormer on CIFAR10-DVS. Since accuracy at 1:8 remains well below that of the dense models, we treat this pattern as a stress test rather than a deployment setting. Overall, these results agree with Section 4: the advantage of FastSNN grows with compression strength and is most pronounced on ResNet-series models.

Table 16: Extreme structured sparsity: 1:8 pruning results. Best results are in **bold**, second best are underlined.

Model	Dense	Wanda	OSSCAR	SBC	**Ours**
ImageNet					
MS-ResNet-18	63.04	1.04	7.77	1.55	**29.75**
MS-ResNet-34	69.42	1.10	7.93	3.21	**34.64**
MS-ResNet-104	74.24	0.35	1.27	0.57	**22.18**
SEW-ResNet-18	62.82	1.33	7.43	5.70	**30.33**
SEW-ResNet-34	66.83	1.25	5.64	3.84	**31.57**
SEW-ResNet-50	67.53	0.53	5.07	0.96	**15.48**
SEW-ResNet-101	68.57	0.48	2.18	0.49	**11.72**
SEW-ResNet-152	69.10	0.33	1.02	0.50	**9.46**
SpikingResFormer-Ti	74.26	1.79	8.69	2.00	**11.01**
SpikingResFormer-S	75.88	2.32	9.54	1.76	**18.64**
SpikingResFormer-M	77.07	3.72	15.17	**29.27**	25.56
SpikingResFormer-L	78.44	2.98	18.95	**37.07**	30.85
MaxFormer-10-384	77.54	9.55	20.75	0.98	**26.29**
MaxFormer-10-512	78.68	14.20	28.72	2.09	**37.21**
MaxFormer-10-768	82.21	21.86	32.27	5.49	**38.90**
MS-QKFormer-384	76.26	13.73	24.16	1.33	**25.96**
CIFAR-100					
MS-ResNet-32	61.75	5.53	5.53	5.39	**13.45**
MS-ResNet-56	64.60	4.83	9.03	3.65	**18.88**
MS-ResNet-110	61.75	5.30	9.13	5.75	**18.14**
SEW-ResNet-32	62.09	3.74	8.15	2.69	**25.18**
SEW-ResNet-56	64.82	3.73	7.42	2.71	**29.03**
SEW-ResNet-110	67.45	5.54	7.27	3.79	**30.19**
SpikingResFormer-Ti	83.56	17.10	36.14	50.02	**50.84**
SpikingResFormer-S	84.42	23.36	42.12	54.22	**58.23**
MaxFormer	75.78	43.64	56.30	44.77	**57.54**
MS-QKFormer	74.49	43.67	**57.25**	28.66	50.86
CIFAR10-DVS					
MS-ResNet	68.30	12.70	21.40	22.20	**36.30**
SEW-ResNet	72.30	44.60	60.60	56.90	**66.00**
SpikingResFormer-Ti	80.70	32.50	56.60	65.60	**65.90**
SpikingResFormer-S	74.30	26.50	52.00	61.50	**63.90**
MaxFormer	78.70	62.30	**71.20**	58.80	68.90
MS-QKFormer	76.90	66.60	**70.00**	64.00	67.10