Adaptive Fission: Post-training Encoding for Low-latency Spike Neural Networks

Yizhou Jiang[1] **Feng Chen**[1] **Yihan Li**[1] **Yuqian Liu**[1]

Haichuan Gao[2] **Tianren Zhang**[1*] **Ying Fang**[3*]

[1]Department of Automation, Tsinghua University, Beijing, China
[2]Beijing QianJue Technology Co., Ltd.
[3]College of Computer and Cyber Security, Fujian Normal University, Fuzhou, China
{jiangyz20, lyh19, liuyuqian21}@mails.tsinghua.edu.cn
haichuan.gao@qj-robots.com, chenfeng@mail.tsinghua.edu.cn
trzhang@mail.tsinghua.edu.cn, fy20@fjnu.edu.cn

Abstract

Spiking Neural Networks (SNNs) often rely on rate coding, where high-precision inference depends on long time-steps, leading to significant latency and energy cost—especially for ANN-to-SNN conversions. To address this, we propose Adaptive Fission, a post-training encoding technique that selectively splits high-sensitivity neurons into groups with varying scales and weights. This enables neuron-specific, on-demand precision and threshold allocation while introducing minimal spatial overhead. As a generalized form of population coding, it seamlessly applies to a wide range of pretrained SNN architectures without requiring additional training or fine-tuning. Experiments on neuromorphic hardware demonstrate up to 80% reductions in latency and power consumption without degrading accuracy.

1 Introduction

Unlike traditional artificial neural networks (ANNs) that use floating-point activations, spiking neural networks (SNNs) encode activations as binary spike sequences over time, promising energy-efficient intelligence on neuromorphic hardware [31, 5]. However, they still lag behind ANNs in precision and performance due to inefficient rate-based encoding. While a T-bit quantized ANN represents 2^T distinct values, an SNN needs T time-steps with 1-bit outputs per step to encode only $T+1$ discrete states, as illustrated in Fig. 1. This makes high-precision inference slow and energy-intensive on complex tasks such as generation [44, 20].

Inspired by training-aware quantization [9, 51], recent efforts aim to train SNNs from scratch with variable bit-lengths or time-steps [45, 37], thereby reducing reliance on temporal precision. However, in many real-world cases such as open-vocabulary detection, SNNs are converted from large, pretrained ANNs (e.g., applications of CLIP [35, 24, 50, 48]), where retraining is infeasible due to proprietary data or computational cost. To address this, existing post-training methods either modify spike bits [42, 41] or increase firing steps locally [27], but yield only modest gains. Moreover, most neuromorphic chips lack support for multi-bit, multi-threshold neurons or dynamic time-steps. Therefore, a hardware-friendly post-training solution for low-latency deployment remains critically needed.

*Corresponding authors.
[1]Code is available at: https://github.com/JiangYizhou16/Adaptive-Fission

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

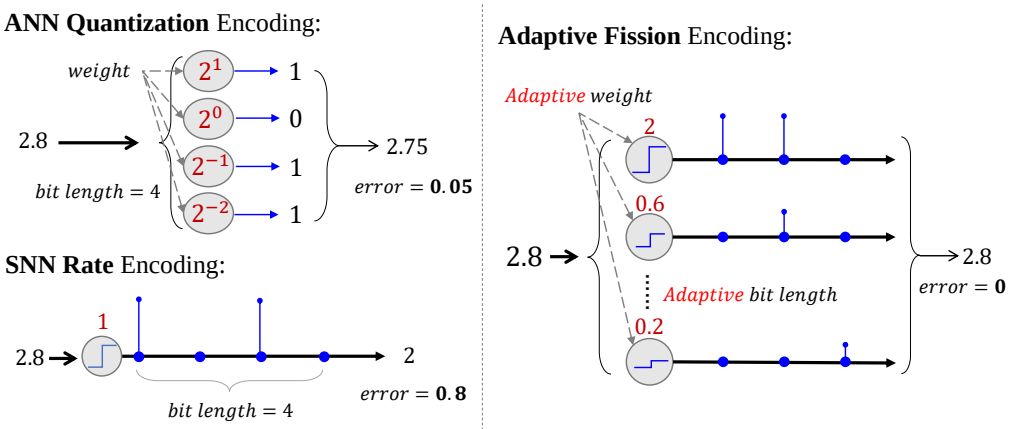


Figure 1: An overview of encoding methods. *Adaptive Fission* combines the sparsity of SNN rate encoding with the efficiency of ANN quantization, enabling variable bit-lengths and weights.

As a classical yet underutilized alternative, population coding distributes representation across groups of neurons, replacing temporal integration with spatial parallelism and thus reducing latency. Prior works [40, 30, 24] typically adopt static group structures with fixed scales, weights, and thresholds, resulting in substantial memory overhead and hindering practical deployment. In contrast, we propose Adaptive Fission, a flexible, generalized population-coding strategy that selectively splits high-sensitivity neurons into dynamically sized groups and allocates precision and thresholds on demand. This neuron-wise bit allocation enables mixed-precision inference while remaining compatible with binary neuron models. Our analysis shows it approaches the theoretical capacity of population coding with only moderate spatial overhead.

To implement Adaptive Fission, we design a two-stage iterative pipeline to address two key challenges: identifying which neurons should undergo fission and determining the scales and thresholds within each neuron group. In the first stage, we estimate and rank neuron sensitivity by jointly analyzing gradients and residual membrane potentials, prioritizing those with the greatest impact. This targeted reallocation achieves capacity comparable to fixed-length population encoding with only 20% additional neurons. In the second stage, we derive an optimality condition for assigning thresholds to the newly created sub-neurons, based on the cumulative distribution of membrane potentials. Although analytically intractable, we approximate the solution using Monte Carlo sampling combined with Newton's method, yielding minimal encoding error.

By enhancing representational precision under strict latency constraints, Adaptive Fission enables faster, more energy-efficient inference. As a post-training technique, it is compatible with both ANN-to-SNN conversions and directly trained SNNs, and can be combined with quantization and pruning to offset the modest increase in neuron count. Deployments on a Lynxi HP201 neuromorphic chip [34] demonstrate up to 80% reductions in latency and energy consumption without compromising accuracy. Beyond classification, Adaptive Fission also supports generative tasks like image synthesis, highlighting its potential for high-throughput neuromorphic computing.

Our key contributions are as follows:

1. We propose Adaptive Fission, a post-training population-coding method that assigns variable bit lengths and weights, with theoretical guarantees of exponential error reduction.
2. We introduce a practical, hardware-aware encoding pipeline for parallel SNN acceleration, compatible with both ANN-to-SNN conversion and directly trained SNNs.
3. Evaluation on a neuromorphic platform demonstrates up to 80% reductions in latency and power consumption while maintaining competitive accuracy.

2 Related Work

2.1 Spike Encoding

Rate coding [1, 11] remains the dominant strategy in SNNs, where all spikes contribute equally with temporal information discarded. Despite its simplicity and robustness, it suffers from low information density. Variants such as signed or ternary spikes [42, 14] and variable spike amplitudes

or thresholds [15, 47, 8, 41] enhance expressivity but often compromise hardware compatibility. Alternatively, population coding leverages multiple neurons to jointly represent scalar values. While promising, existing works typically employ fixed-scale ensembles [45, 37] or rely on hand-crafted heuristics for neuron substitution [24], limiting precision scalability. Besides, temporal coding [12, 39, 32, 22] uses spike timing rather than frequency to improve encoding density, but it currently underperforms rate coding. Our method can be viewed as a generalized framework of population coding with adaptive, hardware-aware support.

2.2 Latency Reduction in SNNs

Reducing the number of simulation steps is critical for latency and energy-efficiency. Existing efforts can be categorized into three main types: **(1) Post-training conversions** introduce burst spikes [29] or negative spikes [42], but still typically require > 32 time-steps to match ANN-level accuracy. **(2) Quantization-aware conversions** emulate low-bit activations via modified activation functions [17, 4], or employ pre-charged neurons [23, 3] to reduce inter-layer delays, bringing time-steps down to 8–16 but necessitating training from scratch. **(3) Direct training approaches** treat SNNs as recurrent networks and use surrogate gradients [33, 52] or synaptic plasticity [25, 2], with recent work incorporating variable-bit activations [43, 37]. Despite achieving 2–8 time-steps, they suffer from training instability and scalability issues in larger networks [46]. Our method complements these approaches by offering a post-training optimization without retraining or architectural modifications, and is fully compatible with both converted and directly trained SNNs.

3 Preliminaries

As our method operates at the neuron level, we first formulate the model from the perspective of individual neurons.

ANN models: In a standard feedforward ANN with ReLU activation, each neuron receives input from presynaptic activations a_{pre}^j via synaptic weights w^j. The total input q_{ann} and activation a are:

$$q_{ann} = \sum_j w^j a_{pre}^j, \quad a = \max\{q_{ann}, 0\}. \tag{1}$$

SNN models with weighted spikes: We focus on soft-reset Integrate-and-Fire (IF) neurons [36, 16] and adopt the weighted-spike formulation commonly used in ANN–SNN conversion [28]. Let θ be the firing threshold, and $s_{pre}^j(t)$ the presynaptic spike at time-step t. The total input is $q_{snn}(t) = \sum_j w^j s_{pre}^j(t)$, and membrane dynamics are defined as:

$$\text{Charging: } v_{temp}(t) = v(t-1) + q_{snn}(t), \tag{2}$$

$$\text{Firing: } s(t) = \theta \cdot H\big(v_{temp}(t) - \theta\big), \tag{3}$$

$$v(t) = v_{temp}(t) - s(t), \tag{4}$$

where $v(t)$ is the membrane potential after step t, and $H(\cdot)$ the Heaviside function. Here $s(t)$ is not binary $\{0, 1\}$ but weighted $\{0, \theta\}$. To support binary-only hardware, we absorb θ into synaptic weights:

$$w := \theta w, \quad s(t) := s(t)/\theta = H\big(v_{temp}(t) - \theta\big) \in \{0, 1\}. \tag{5}$$

For simplicity, we continue using the $\{0, \theta\}$ notation, where θ also denotes the effective output weight.

SNN and Quantized ANNs: In rate coding, a neuron's activation intensity is represented by its average firing rate over T time-steps, i.e., $s = \frac{\sum_{t=1}^{T} s(t)}{T}$. Assuming $\sum_t q_{snn}(t) = q_{ann} = a$, the SNN activation approximates a T-level quantized ANN:

$$s = \frac{a - v(T)}{T} = \frac{\theta \cdot clip\left(\left\lfloor \frac{a}{\theta} \right\rfloor, 0, T\right)}{T}. \tag{6}$$

Thus, the firing rate approximates the quantized ANN activation, with quantization error stemming from θ and reflected in the residual potential $v(T)$. Increasing T distributes activation across more steps, allowing a smaller threshold $\theta \approx \max(a)/T$ for finer quantization.

4 Population Encoding Models

Most SNNs assign a single binary neuron to each activation, limiting precision under temporal constraints. We propose a generalized population coding framework where multiple binary neurons jointly encode a single activation, enabling finer quantization while preserving hardware-friendly binarization. Unlike prior methods that require retraining or weight updates, our approach is entirely post-training and model-agnostic, making it applicable to both converted and directly trained SNNs.

4.1 Static Models and Error Analysis

To understand the precision limits of multi-neuron representations, we first isolate static encoding behavior, independent of spike timing.

Definition 4.1 (Encoding Error)**.** Consider a group of k neurons with distinct thresholds $\theta_i \in \mathbb{R}^+$. Let n_i denote the spike count of neuron i within T time-steps. The set of representable values is:

$$\mathcal{S} = \left\{ 0, \sum_{i=1}^{k} \theta_i n_i + b \,\middle|\, n_i \in \{0, 1, \dots, T\} \right\}, \tag{7}$$

where b is constant bias. Given an input distribution $P(q)$ with $q \geq 0$, the expected encoding error is:

$$r = \mathbb{E}_q \left[\min_{s \in \mathcal{S}, s \leq q} (q - s) \right]. \tag{8}$$

In this context, a conventional IF neuron is a simplified case with $k = 1$. For a fixed θ, the optimal firing count is $n = clip\big(\lfloor \frac{q-b}{\theta} \rfloor, 0, T\big)$. Assuming a uniform feature distribution, the expected encoding error can be directly computed, aligning with previous results [7].

Proposition 4.2 (Optimal Error of Single Neuron)**.** *If $q \sim U(0, q_{\max})$, the minimal error of a single neuron is given by $r = \frac{q_{\max}}{2(T+1)}$ with the threshold and bias $\theta = 2r$ and $b = r$.*

Extending to multi-neuron encoding (Def. 4.1), the absence of fixed firing patterns across neurons prevents direct computation of individual spike counts n_i. Yet under the uniform assumption, an optimal configuration can still be derived without explicit firing rules.

Proposition 4.3 (Optimal Error of Neuron Groups)**.** *For a k-neuron group encoding $q \sim U(0, q_{\max})$, the optimal error r, threshold θ_i and bias b are given by:*

$$r = \frac{q_{\max}}{2(T+1)^k},\ \theta_i = \frac{q_{\max}}{(T+1)^i},\ b = r. \tag{9}$$

As the number of neurons increases, the error decreases exponentially, yielding far substantial precision gains compared to the linear improvement from increasing time-steps T. This result, however, captures only aggregate encoding after all T steps, providing an optimal hindsight solution while ignoring spike dynamics.

4.2 Temporal Dynamic Encoding Models

Previous encoding analyses typically assume full knowledge of the input q over all time-steps. In contrast, practical SNN inference operates under an online regime, where input is incrementally accumulated and spikes are emitted as soon as the membrane potential crosses the threshold. This scenario can be viewed as a special case of Proposition 4.3 with $T = 1$.

Proposition 4.4 (Optimal Error with Temporal Dynamics)**.** *For a k-neuron group, assume the residual membrane potential at the final step follows $P\big(v_{temp}(T)\big) \in U(0, \frac{q_{\max}}{T})$. The optimal error r, threshold θ_i and bias b are given by*

$$r = \frac{q_{\max}}{T \cdot 2^{k+1}}, \theta_i = \frac{q_{\max}}{T \cdot 2^i}, b = r. \tag{10}$$

This proposition implies that exponentially increasing precision (2^{k+1} levels) can still be achieved under online temporal encoding by assigning thresholds as powers of two. Moreover, it provides

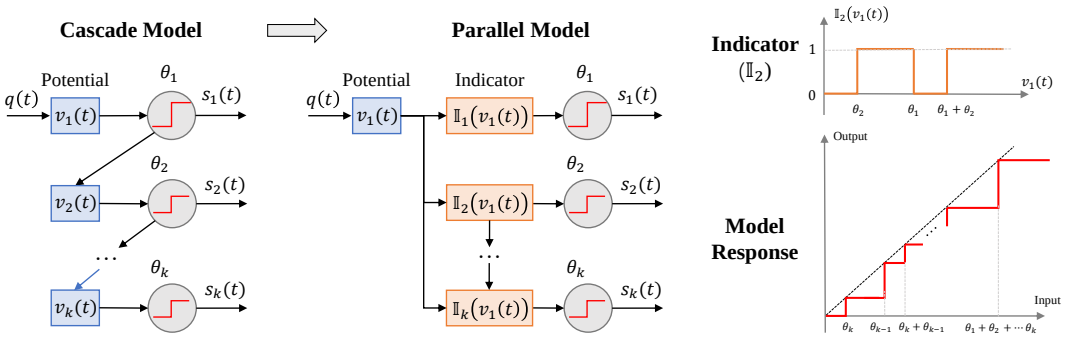


Figure 2: Dynamics of our proposed neuron groups. Neurons are arranged in a hierarchical structure based on descending thresholds, and can be parallelized by pre-compiled indicators. Their response of input spikes forms a multi-level step function as an uneven quantization.

theoretical support for recent neuron-grouping heuristics [30]. However, these results critically rely on the assumption of a uniformly distributed residual potential, which rarely holds in practice, where long-tailed or multi-modal distributions are more common.

To generalize the analysis to arbitrary input distributions, a more explicit mathematical formulation of a neuron group's temporal dynamics is necessary. While multiple formulations are possible, we adopt a descending-threshold population model (Fig. 2), where the residual input is successively propagated to neurons with lower thresholds after higher-threshold neurons fire. Although this formulation is inherently sequential, it admits an equivalent parallel implementation that preserves efficiency.

Definition 4.5 (Spiking Neuron Group Model)**.** Consider a group of k neurons with descending thresholds $\theta_1 \geq \theta_2 \geq \cdots \geq \theta_k$. At time-step t, let $q(t)$ denote the total synaptic input, $v_i(t)$ the residual potential of neuron i, and $s_i(t)$ its weighted output spike.

Initialize the first neuron's potential as $v_1(1) = q(1)$. Then, for all $t = 1, \ldots, T$ and $i = 1, \ldots, k$:

$$\text{Firing:} \quad s_i(t) = \theta_i \cdot H\left(v_i(t) - \theta_i\right), \tag{11}$$

$$\text{Transfer:} \quad v_{i+1}(t) = v_i(t) - s_i(t), \tag{12}$$

$$\text{Charging:} \quad v_1(t+1) = v_k(t) - s_k(t) + q(t+1). \tag{13}$$

The total output $s(t)$ is the sum of individual weighted spikes:

$$s(t) = \sum_{i=1}^{k} s_i(t). \tag{14}$$

Although the above dynamics is presented as a sequential cascade—where higher-threshold neurons fire first and residual potential is propagated downward—the computation can equivalently be realized in a fully parallel manner. Specifically, each neuron's firing condition depends solely on the total input and the predefined threshold set, eliminating the need for explicit synchronization with upstream firing events. This property enables the cascade to be "precompiled" into lightweight parallel comparison logic. Formally, once the thresholds θ_i are fixed, all spike firings can be expressed as binary indicator functions $\mathbb{I}_i$ composed of Heaviside terms:

$$s_i(t) := \theta_i \cdot \mathbb{I}_i(v_1(t)) = \theta_i \cdot \sum_{m_k \in \{0,1\}} \left\{ H\Big(v_1(t) - \sum_{j=1}^{i-1} m_j\theta_j\Big) - H\Big(v_1(t) - \sum_{j=1}^{i-1} m_j\theta_j - \theta_i\Big) \right\} \tag{15}$$

This representation requires only comparisons against fixed thresholds and thus introduces negligible overhead on neuromorphic hardware. The resulting hierarchical model forms the foundation for adaptive neuron encoding. By tuning the number of neurons k and the threshold set θ_i for each activation, precision can be flexibly allocated without retraining or modifying weight values. In the following section, we demonstrate how to determine the optimal configuration of this structure for each neuron based on post-training statistics.

5 Methodology: Adaptive Fission Encoding

5.1 Overview

In practical networks, individual neurons exhibit diverse activation distributions and precision requirements, making it intractable to derive optimal bit-lengths or threshold assignments analytically.

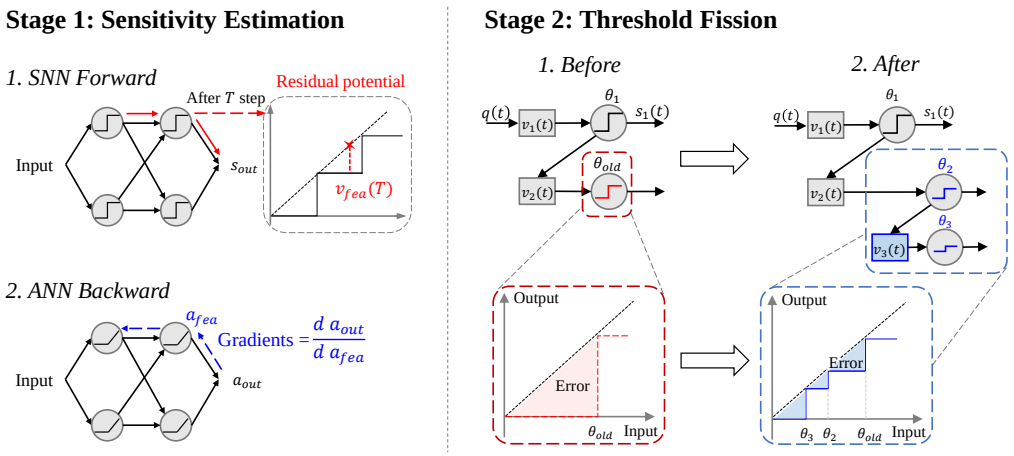


Figure 3: Overview of the two iterative stages in Fission Encoding pipeline. Sensitivity Estimation identifies neurons with high impact using residual potentials and gradients. Threshold Fission splits selected neurons and optimizes their thresholds based on their activation distribution. The process progressively enhances precision while preserving binary spike constraints.

To address this, we adopt a two-stage iterative pipeline, termed Adaptive Fission, as illustrated in Fig.3. Analogous to post-training calibration in ANN quantization [21], the first stage employs a small calibration set to estimate each neuron's precision demand based on its contribution to the output error. This calibration procedure is entirely training-free, as it involves no weight updates and only collects activation statistics, thereby requiring a small number of samples and incurring negligible computational overhead. In the second stage, we selectively allocate additional neurons and adjust their thresholds to minimize the expected residual potential. The algorithm progressively splits the most influential neurons into two units with reduced weights in each iteration. This neuron "fission" mechanism achieves a principled balance between precision and resource consumption while maintaining the original network's accuracy.

5.2 Stage 1: Sensitivity Estimation and Selection

To prevent excessive scale increase, we consider the inherent redundancy in deep networks, and focus precision enhancement on neurons with the most significant impact on output accuracy, introducing a "sensitivity" metric to quantify such impact. Specifically, its estimation involves two propagation passes. In the first, a standard SNN inference records the residual potential $v_{fea}(T)$ after T steps. In the second pass, we temporarily substitute spiking neurons with ReLUs to enable ANN-like gradient backpropagation. Denoting the continuous activation of a neuron as a_{fea}, the sensitivity σ_{fea} can be approximated by a first-order Taylor expansion:

$$\sigma_{fea} = \mathbb{E}\|a_{out}(a_{fea}) - a_{out}(a_{fea} - v(T))\|_2 \approx \mathbb{E}\|\frac{da_{out}}{da_{fea}} \cdot v_{fea}(T)\|_2, \tag{16}$$

where the expectation is taken over calibration samples. Intuitively, this measures how much the residual potential at each activation neuron affects the final output. Neurons are ranked by σ_{fea}, and only a top fraction (fission rate) are selected for further enhancement.

5.3 Stage 2: Optimal Threshold Fission

For each neuron group selected for fission, we add one extra neuron to redistribute thresholds and achieve finer-grained representation. Unlike prior analysis, we estimate the potential distribution directly from calibration data. Because residual potential is mainly determined by the smallest threshold, fission is applied to the last neuron in the group with threshold θ_{old}. We split it into two with thresholds $\theta_k > \theta_{k+1}$ such that $\theta_k + \theta_{k+1} = \theta_{old}$, preserving total contribution. Given the residual potential $v = v_k(T)$, the improved quantization reduces residual error as:

$$\Delta v = (\theta_k - \theta_{k+1})H(v - \theta_k) + \theta_{k+1}H(v - \theta_{k+1}), \tag{17}$$

and the objective is therefore to determine θ_k that maximizes the expected error reduction, $\mathbb{E}[\Delta v]$.

Since Δv is non-differentiable, closed-form solutions are infeasible. Instead, we derive the optimal thresholds using the cumulative distribution of potential.

Theorem 5.1 (Optimal Fission Threshold)**.** *Let $F(\cdot)$ denote the cumulative distribution function of residual potential v. The optimal threshold θ_k is given by solution of equation:*

$$2F(\theta_k) - F(\theta_{old} - \theta_k) - 1 = 0, \tag{18}$$

and the maximum expected reduction in residual potential is:

$$\mathbb{E}[\Delta v] = \theta_{old} \cdot (1 - F(\theta_k)), \tag{19}$$

which is bounded by:

$$\frac{1}{2} \leq \frac{\mathbb{E}\left[\Delta v\right]}{\mathbb{E}\left[v\right]} \leq 1 \tag{20}$$

Remark 5.2. Detailed proof is in the Appendix.A.3 . We first substituting the expectation of Δv with an integral over the range of v, then transforming it into the cumulative distribution function $F(v)$:

$$\begin{aligned}\mathbb{E}[\Delta v] = &\int_{\theta_k}^{\theta_{old}} (\theta_k - \theta_{k+1}) p(v) dv + \int_{\theta_{k+1}}^{\theta_{old}} \theta_{k+1} p(v) dv \\ =&\theta_k \left[1 - F(\theta_k)\right] - (\theta_{old} - \theta_k) \cdot \left[F(\theta_k) - F(\theta_{old} - \theta_k)\right].\end{aligned}$$

The extremum of $\theta_k = \arg\max_{\theta_k} \mathbb{E}[\Delta v]$ can be obtained by differentiation. As $F(v)$ is estimated from discrete samples, its derivative $p(v)$ can be neglected, yielding:

$$\frac{d\mathbb{E}[\Delta v]}{d\theta} = [2F(\theta_k) - F(\theta_{old} - \theta_k) - 1] = 0, \tag{21}$$

which provides the optimal solutions in Eq.18.

This result implies that each additional neuron introduced by fission encoding can reduce the quantization error by at least half, even under the worst-case residual distribution. Consequently, the precision exhibits exponential scaling with a base greater than 2, surpassing the efficiency of uniform quantization and demonstrating even stronger empirical performance.

For practical numerical implementation, the search for θ_k is conducted using a Monte Carlo estimation of $F(v)$ collected from the calibration set. Owing to the piecewise nature of $F(v)$, we first initialize θ_k using a coarse grid search, followed by refinement via the Newton iteration method:

$$\theta_k := \theta_k - \frac{2F(\theta_k) - F(\theta_{k+1}) - 1}{2p(\theta_k) + p(\theta_{k+1})}, \tag{22}$$

where $p(\theta_k)$ is estimated by sampling within small intervals δ around θ_k, i.e.,

$$F(\theta)' = p(\theta) \approx \frac{1}{2\delta} \sum_v I(\theta - \delta \leq v \leq \theta + \delta). \tag{23}$$

To further reduce quantization bias caused by Heaviside activation, we adopt a rounding strategy similar to [4] for the last neuron in each group. Specifically, we apply a linear compensation to the input and modify the step function to be symmetric around the threshold:

$$q(t) \leftarrow q(t) + \frac{0.5}{T}\theta_{k+1}, \tag{24}$$

thereby compensating for the flooring bias in spike generation.

6 Experiments

6.1 Implementations

Fission Encoding is a post-training strategy broadly compatible with diverse SNN architectures. It leverages high-accuracy training with long time-steps while enabling low-latency inference at deployment. We evaluate it across representative paradigms, including continuous-activation conversion Calib [28], quantized conversion QCFS, SRP [4, 17], direct training for spike CNNs TEBN [10], post-training conversion of ANN Transformers STA [24], and spike-based Transformers Spike-driven [49].

For classification, we conduct experiments on CIFAR-100 [26] and ImageNet [6] using ResNet20 [18], VGG16 [38], and a spike-based Transformer [49]. Converted models use $T = 32$ time-steps, while

Table 1: Performance comparison before/after Fission. Mem., Time and Energy per epoch are measured directly on-chip. All accuracy values reflect the best performance at evaluated time-step.

Method	Time Step	Fission Rate	Accu. (%)	Mem. (GB)	Time/ Epoch(s)	Energy (Wh)
CIFAR-100 & ResNet20						
Calib. (Conversion)	32	No	*76.32*	0.85	56.8	0.64
	16		64.48		29.1	0.37
	8		32.10		16.3	0.22
	4		20.43		10.0	0.14
+ Fission	16	0.10	76.20	0.88	31.4	0.42
	8	0.78	73.28	1.17	19.5	0.27
	4	1.35	73.11	1.31	12.0	0.18
	2	3.70	70.50	2.25	**6.4**	**0.11**
QCFS. (Quantization)	32	No	*76.78*	0.81	59.6	0.70
	16		67.54		30.2	0.36
	8		62.60		16.2	0.21
	4		55.25		9.7	0.12
+ Fission	16	0.16	*74.61*	0.84	31.8	0.39
	8	0.69	72.79	1.09	16.4	0.22
	4	1.17	72.15	1.22	10.1	0.18
	2	2.93	69.43	1.83	**5.9**	**0.13**
SRP. $\tau = 4$ (Quantization)	32	No	65.50	0.80	55.1	0.57
	16		64.71		28.3	0.35
	8		62.94		15.7	0.18
	4		59.34		8.0	0.11
+ Fission	16	0.28	**68.72**	0.86	27.8	0.39
	8	1.02	66.41	1.13	14.4	0.22
	4	2.71	65.39	1.65	9.1	0.16
TEBN. (Training)	4	No	*76.13*	0.83	9.8	0.18
	2		56.04		5.0	0.11
+ Fission	2	1.47	73.46	1.18	5.3	0.14
	1	1.95	69.20	1.50	**3.2**	**0.09**
CIFAR-100 & ViT-B/32 (ANN Transformer Conversion)						
STA. (Conversion)	64	No	**85.25**	8.71	128.3	1.72
	32		84.15		70.5	1.03
+ Fission	16	0.53	82.70	10.4	39.2	0.77
	8	1.15	80.28	15.7	24.3	0.42

Method	Time Step	Fission Rate	Accu. (%)	Mem. (GB)	Time/ Epoch(s)	Energy (Wh)
ImageNet & VGG16						
Calib. (Conversion)	32	No	*63.64*	3.71	604	7.55
	16		55.79		302	4.30
	8		28.10		154	2.37
+ Fission	16	0.15	61.94	3.93	324	4.75
	8	0.87	60.79	5.28	169	2.61
	4	1.57	58.31	6.21	**88**	**1.43**
QCFS. (Quantization)	32	No	*68.10*	3.54	627	7.82
	16		50.97		324	4.49
+ Fission	16	0.23	67.32	3.87	332	4.61
	8	0.91	64.51	5.01	157	2.91
	4	1.80	62.87	6.92	**93**	**1.53**
TEBN.	4	No	*69.03*	3.62	78	1.16
+ Fission	2	0.39	67.85	4.19	47	0.84
	1	1.31	64.19	5.77	**28**	**0.69**
ImageNet & ResNet34						
Calib. (Conversion)	32	No	64.82	2.68	348	3.02
	16		56.21		189	1.73
	8		31.75		110	0.95
+ Fission	16	0.17	**66.94**	2.81	214	2.01
	8	0.96	63.75	4.60	135	1.38
	4	1.72	60.43	5.57	**55.2**	**0.61**
QCFS. (Quantization)	32	No	**69.37**	2.48	372	3.29
	16		59.35		228	2.08
+ Fission	16	0.28	67.93	2.96	240	2.41
	8	1.04	66.51	4.38	156	1.54
	4	1.95	62.41	5.05	**81.3**	**0.95**
ImageNet & Spike-driven Transformer 8-384						
Spike-driven.	4	No	*72.28*	4.57	192	3.49
+ Fission	2	0.54	72.41	5.54	98	1.91
	1	0.98	70.29	6.49	**60**	**1.02**

directly trained models use $T = 4$. Fission Encoding is applied post-training with 2,000 calibration samples for CIFAR-100 and 20,000 for ImageNet. For image generation on CIFAR-10, we convert a 4-layer spike CNN generator (WGAN-GP [13]) and a U-Net diffusion model [19], both configured with $T = 64$ and calibrated on the full 60,000-sample dataset. The output layer remains in non-spiking RGB format. All models are deployed on a Lynxi HP201 neuromorphic accelerator (16GB), a commercial successor to Tianjic [34], for real-time measurement of memory usage, latency, and energy consumption. More details see Sec. E.

6.2 Main Results on Image Classification

Table 1 summarizes classification results with Fission Encoding across various time-steps, training strategies, and architectures. We report top-1 accuracy, on-chip memory, inference latency per epoch, and energy usage, all measured on the Lynxi HP201 accelerator with a batch size of 32. Fission Encoding consistently improves the accuracy–efficiency trade-off. By reallocating precision spatially instead of extending spike windows temporally, it reduces inference time-steps by up to $8\times$ while maintaining competitive accuracy. For example, decreasing T from 32 to 4 results in less than a 5% accuracy drop, up to 83% energy savings, and only a $1.6\times$ memory increase.

This trade-off is particularly advantageous for neuromorphic SNN deployments, where memory budgets are typically modest. The additional memory introduced by fission remains well below the 16 GB on-chip capacity of the HP201 accelerator, obviating the need for off-chip storage or compres-

sion. Furthermore, the increased neuron count has negligible impact on per-cycle power, while the substantial reduction in time-steps markedly lowers both latency and total energy consumption.

Figure 4 illustrates how accuracy and power consumption scale with time-steps on ResNet20. At moderate T, only a small fraction of fission neurons is required to recover full accuracy. Under ultra-low latency constraints ($T = 1$–2), additional neurons significantly improve precision, maintaining competitive performance with minimal spatial overhead. These results demonstrate that Fission Encoding achieves exponential efficiency gains while keeping spatial complexity bounded. A more detailed analysis of memory and computational overhead is provided in Appendix C.

6.3 Main Results on Generation

We further evaluate Fission Encoding on image generation tasks, which demand higher activation precision due to richer decoding information flow. Conventional ANN-based generators typically require at least 6-bit precision (corresponding to 64 time-steps in SNNs), making low-latency inference challenging.

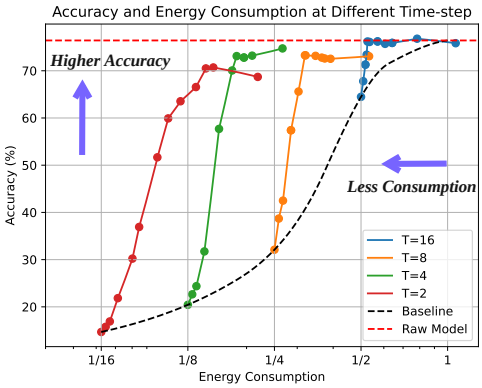


Figure 4: Accuracy and energy consumption across time-steps. Each point on the same curve denotes a different fission rate.

We convert two pretrained models—WGAN-GP [13] and DDPM [19]—into SNNs using post-training calibration [28] and apply Fission Encoding with increasing fission rates as T decreases: $[10, 30, 60, 90, 120]$ for T=64 to T=4. Directly reducing T below 32 severely degrades image quality, whereas Fission Encoding recovers visually comparable results even at T=4, as confirmed by PSNR measurements against ANN outputs. Under low-latency settings, DDPM models outperform WGAN-GP, preserving consistency despite reduced detail. To our knowledge, no prior work reports comparable visual quality at $T \leq 8$, enabling practical generative inference on neuromorphic hardware.

6.4 Further Analysis of Algorithm

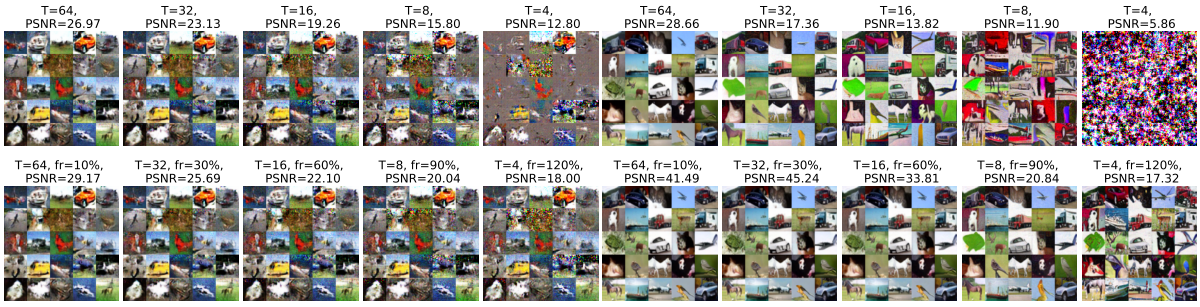
Fission Rate and Threshold. Fig.4 shows that accuracy initially improves rapidly with increasing fission rate and saturates once most layers achieve sufficient precision. At very low time-steps, the saturation point shifts due to the SNN unevenness effect [17], where excessive residual potential accumulates in late steps. Layer-wise analysis in Fig.6 reveals that deeper layers typically undergo more fission ("Fission 1/2" denotes activations split into two or three neurons). Fig.7 further illustrates that high-sensitivity neurons often exhibit long-tailed or multimodal activation distributions; optimal thresholds derived after fission (red lines) effectively capture these patterns.

Ablation and Sensitivity-Based Selection. We further examine the effect of Stage-1 sensitivity selection within the pipeline. Unlike prior coding schemes [24, 30] that uniformly allocate precision, our approach selectively applies fission to highly sensitive neurons. As shown in Fig. 8, disabling selection (i.e., fissioning all neurons) slightly increases peak accuracy ($< 2\%$) by including rare but important units. However, this significantly increases computational and memory overhead. With selection enabled, about 70% of neuron fission is avoided, yielding a better cost–accuracy trade-off.

Comparison with Baselines. We compare Fission Encoding with two population-coding methods: Spatial Approximation [24] and Group Neuron [30]. Even without sensitivity selection, our approach achieves 68.22% accuracy with 100% additional neurons, surpassing the 66.94% of [24]. When each method uses its optimal neuron budget, baseline accuracy plateaus at 72.85% with 400% overhead, while ours reaches 73.28% with only 78%. These results demonstrate that Fission Encoding delivers both higher accuracy and significantly better neuron efficiency.

6.5 Computational and Hardware Implications

Computational Complexity. Transitioning from cascade to parallel firing introduces more comparisons (CMPs) but significantly reduces additions (ADDs). For a group of n split neurons, achieving 2^n-level precision requires 2^n-1 CMPs and n ADDs with our approach, compared to 2^n-1 CMPs



(a) WGAN (b) DDPM

Figure 5: Image generation and PSNR for CIFAR-10 at varying inference time-steps. Top row: original SNN outputs. Bottom row: results with fission encoding applied.

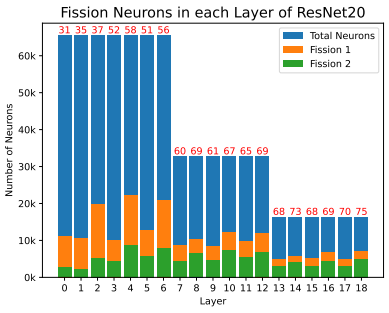


Figure 6: Neuron fission rounds across layers. Red numbers is the average fission rates per layer.

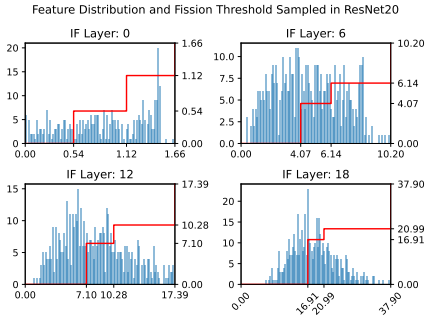


Figure 7: Activation value distributions. Red line reflects the fission thresholds and the response modes.

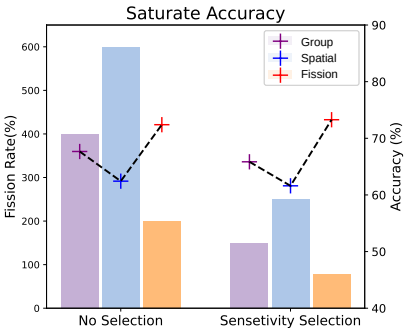


Figure 8: Comparison of accuracy and fission rate with related works and ablation of sensitivity selection.

and 2^n-1 ADDs in conventional parallel coding. While CMP complexity grows exponentially in both cases, ADD operations—typically more expensive—scale only linearly, making computation substantially more hardware-friendly. The slightly increased CMP cost is offset by exponentially fewer time-steps, yielding overall energy and latency benefits without hardware modifications. Further complexity analysis is provided in Appendix.C.

Compatibility with Pruning. Fission Encoding can be combined with unstructured pruning to further reduce memory overhead by removing neurons with low sensitivity and weak post-synaptic weights. While less effective on conventional GPUs, this sparsification provides clear gains on neuromorphic hardware: on ResNet20 with T=2, pruning reduces memory usage from 2.25 GB to 1.53 GB with less than 2% accuracy loss.

7 Conclusion and Discussion

Activation quantization under limited time-steps remains a critical obstacle to scaling SNNs toward complex tasks. Drawing inspiration from post-training quantization in ANNs, we reinterpret population coding from a hardware-aware perspective and propose Adaptive Fission, a post-training mechanism that redistributes precision across the network while preserving binary spike compatibility.

Our design is motivated by a simple observation: conventional ANN quantization imposes two unnecessary constraints—global fixed bitwidth and fixed bit weights (e.g., powers of two). By representing precision through multiple spiking sub-neurons, Adaptive Fission relaxes these constraints and flexibly allocates representational capacity where it is most needed. This enables efficient conversion of ANN backbones into SNNs without additional training or fine-tuning.

While broadly applicable across architectures and tasks, our framework shares limitations with existing ANN-to-SNN conversions. It is not directly suitable for event-driven datasets (e.g., CIFAR10-DVS), where input dynamics evolve with simulation time. Moreover, the temporal gains achieved by Adaptive Fission come at the cost of increased spatial complexity, which may raise resource demands in large-scale deployments. Despite these trade-offs, we believe that this precision-aware, post-training strategy represents a practical step toward bridging the gap between high-performance ANNs and efficient neuromorphic deployment.

Acknowledgements

This work was supported in part by the National Key Research and Development Program of China under STI 2030—Major Projects (No. 2021ZD0200300), in part by the National Key Research and Development Program of China (No. 2024YDLN0006), in part by the National Natural Science Foundation of China (Grant No. 62176133), in part by the Tsinghua University - Meituan Joint Institute for Digital Life under Agreement No. C0210322000380, in part by the Tsinghua-Fuzhou Data Technology Joint Research Institute (Project No. JIDT2024013), and in part by Qualcomm Technologies, Inc. under Statement of Work No.TSI617560.

References

[1] Edgar D Adrian. The impulses produced by sensory nerve endings: Part i. *The Journal of physiology*, 61(1):49, 1926.

[2] Mohammad Kazem Bahrami and Soheila Nazari. Digital design of a spatial-pow-stdp learning block with high accuracy utilizing pow cordic for large-scale image classifier spatiotemporal snn. *Scientific Reports*, 14(1):3388, 2024.

[3] Tong Bu, Jianhao Ding, Zhaofei Yu, and Tiejun Huang. Optimized potential initialization for low-latency spiking neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(1):11–20, June 2022.

[4] Tong Bu, Wei Fang, Jianhao Ding, PengLin Dai, Zhaofei Yu, and Tiejun Huang. Optimal ann-snn conversion for high-accuracy and ultra-low-latency spiking neural networks, March 2023.

[5] 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.

[6] 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*, pages 248–255. Ieee, 2009.

[7] Shikuang Deng and Shi Gu. Optimal conversion of conventional artificial neural networks to spiking neural networks, February 2021.

[8] Jianchuan Ding, Bo Dong, Felix Heide, Yufei Ding, Yunduo Zhou, Baocai Yin, and Xin Yang. Biologically inspired dynamic thresholds for spiking neural networks. *Advances in Neural Information Processing Systems*, 35:6090–6103, December 2022.

[9] Zhen Dong, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Hawq: Hessian aware quantization of neural networks with mixed-precision. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 293–302, 2019.

[10] Chaoteng Duan, Jianhao Ding, Shiyan Chen, Zhaofei Yu, and Tiejun Huang. Temporal effective batch normalization in spiking neural networks. *Advances in Neural Information Processing Systems*, 35:34377–34390, December 2022.

[11] Roger M. Enoka and Jacques Duchateau. Rate coding and the control of muscle force. *Cold Spring Harbor Perspectives in Medicine*, 7(10):a029702, January 2017.

[12] Tim Gollisch and Markus Meister. Rapid neural coding in the retina with relative spike latencies. *science*, 319(5866):1108–1111, 2008.

[13] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. *Advances in neural information processing systems*, 30, 2017.

[14] Yufei Guo, Yuanpei Chen, Xiaode Liu, Weihang Peng, Yuhan Zhang, Xuhui Huang, and Zhe Ma. Ternary spike: Learning ternary spikes for spiking neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 12244–12252, 2024.

[15] Yufei Guo, Liwen Zhang, Yuanpei Chen, Xinyi Tong, Xiaode Liu, YingLei Wang, Xuhui Huang, and Zhe Ma. Real spike: Learning real-valued spikes for spiking neural networks. In *European Conference on Computer Vision*, pages 52–68. Springer, 2022.

[16] Bing Han, Gopalakrishnan Srinivasan, and Kaushik Roy. Rmp-snn: Residual membrane potential neuron for enabling deeper high-accuracy and low-latency spiking neural network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13558–13567, 2020.

[17] Zecheng Hao, Tong Bu, Jianhao Ding, Tiejun Huang, and Zhaofei Yu. Reducing ann-snn conversion error through residual membrane potential. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(1):11–21, June 2023.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[19] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[20] Yangfan Hu, Qian Zheng, Xudong Jiang, and Gang Pan. Fast-snn: Fast spiking neural network by converting quantized ann. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(12):14546–14562, December 2023.

[21] 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*, pages 4466–4475. PMLR, 2021.

[22] Sangwoo Hwang, Seunghyun Lee, Dahoon Park, Donghun Lee, and Jaeha Kung. Spikedattention: Training-free and fully spike-driven transformer-to-snn conversion with winner-oriented spike shift for softmax operation. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[23] Sungmin Hwang, Jeesoo Chang, Kyungchul Park, Junsu Yu, and Jong-Ho Lee. Low-latency spiking neural networks using pre-charged membrane potential and delayed evaluation. *Frontiers in Neuroscience*, 15, February 2021.

[24] Yizhou Jiang, Kunlin Hu, Tianren Zhang, Haichuan Gao, Yuqian Liu, Ying Fang, and Feng Chen. Spatio-temporal approximation: A training-free snn conversion for transformers. In *The Twelfth International Conference on Learning Representations*, October 2023.

[25] Saeed Reza Kheradpisheh, Mohammad Ganjtabesh, Simon J Thorpe, and Timothée Masquelier. Stdp-based spiking deep convolutional neural networks for object recognition. *Neural Networks*, 99:56–67, 2018.

[26] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical Report Tech Report, University of Toronto, 2009.

[27] Yang Li and Yi Zeng. Efficient and accurate conversion of spiking neural network with burst spikes, May 2022.

[28] Yuhang Li, Shikuang Deng, Xin Dong, Ruihao Gong, and Shi Gu. A free lunch from ann: Towards efficient, accurate spiking neural networks calibration. In *Proceedings of the 38th International Conference on Machine Learning*, pages 6316–6325. PMLR, July 2021.

[29] Yuhang Li, Shikuang Deng, Xin Dong, and Shi Gu. Converting artificial neural networks to spiking neural networks via parameter calibration, May 2022.

[30] Liuzhenghao Lv, Wei Fang, Li Yuan, and Yonghong Tian. Optimal ann-snn conversion with group neurons, February 2024.

[31] Paul A Merolla, John V Arthur, Rodrigo Alvarez-Icaza, Andrew S Cassidy, Jun Sawada, Filipp Akopyan, Bryan L Jackson, Nabil Imam, Chen Guo, Yutaka Nakamura, et al. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6197):668–673, 2014.

[32] Marcelo A Montemurro, Malte J Rasch, Yusuke Murayama, Nikos K Logothetis, and Stefano Panzeri. Phase-of-firing coding of natural visual stimuli in primary visual cortex. *Current biology*, 18(5):375–380, 2008.

[33] Emre O Neftci, Hesham Mostafa, and Friedemann Zenke. Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks. *IEEE Signal Processing Magazine*, 36(6):51–63, 2019.

[34] Jing Pei, Lei Deng, Sen Song, Mingguo Zhao, Youhui Zhang, Shuang Wu, Guanrui Wang, Zhe Zou, Zhenzhi Wu, Wei He, et al. Towards artificial general intelligence with hybrid tianjic chip architecture. *Nature*, 572(7767):106–111, 2019.

[35] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021.

[36] Bodo Rueckauer, Iulia-Alexandra Lungu, Yuhuang Hu, Michael Pfeiffer, and Shih-Chii Liu. Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Frontiers in Neuroscience*, 11, 2017.

[37] Guobin Shen, Dongcheng Zhao, Tenglong Li, Jindong Li, and Yi Zeng. Are conventional snns really efficient? a perspective from network quantization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 27538–27547, 2024.

[38] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[39] Xiaotian Song, Andy Song, Rong Xiao, and Yanan Sun. One-step spiking transformer with a linear complexity. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, pages 3142–3150, 2024.

[40] Guangzhi Tang, Neelesh Kumar, Raymond Yoo, and Konstantinos Michmizos. Deep reinforcement learning with population-coded spiking neural network for continuous control. In *Conference on Robot Learning*, pages 2016–2029. PMLR, 2021.

[41] Xiaoting Wang, Yanxiang Zhang, and Yongzhe Zhang. Mt-snn: Enhance spiking neural network with multiple thresholds, March 2023.

[42] Yuchen Wang, Malu Zhang, Yi Chen, and Hong Qu. Signed neuron with memory: Towards simple, accurate and high-efficient ann-snn conversion. In *Thirty-First International Joint Conference on Artificial Intelligence*, volume 3, pages 2501–2508, July 2022.

[43] 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*, pages 8441–8450, 2024.

[44] Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In *Proceedings of the 40th International Conference on Machine Learning*, pages 38087–38099. PMLR, July 2023.

[45] Yongjun Xiao, Xianlong Tian, Yongqi Ding, Pei He, Mengmeng Jing, and Lin Zuo. Multi-bit mechanism: A novel information transmission paradigm for spiking neural networks. *arXiv preprint arXiv:2407.05739*, 2024.

[46] Xingrun Xing, Boyan Gao, Zheng Zhang, David A Clifton, Shitao Xiao, Li Du, Guoqi Li, and Jiajun Zhang. Spikellm: Scaling up spiking neural network to large language models via saliency-based spiking. *arXiv preprint arXiv:2407.04752*, 2024.

[47] Changqing Xu, Yi Liu, Dongdong Chen, and Yintang Yang. Direct training via backpropagation for ultra-low-latency spiking neural networks with multi-threshold. *Symmetry*, 14(9):1933, September 2022.

[48] Tianrun Xu, Haoda Jing, Ye Li, Yuquan Wei, Jun Feng, Guanyu Chen, Haichuan Gao, Tianren Zhang, and Feng Chen. Defacto: Counterfactual thinking with images for enforcing evidence-grounded and faithful reasoning, 2025.

[49] Man Yao, Jiakui Hu, Zhaokun Zhou, Li Yuan, Yonghong Tian, Bo Xu, and Guoqi Li. Spike-driven transformer. *Advances in neural information processing systems*, 36, 2024.

[50] Tianren Zhang, Chujie Zhao, Guanyu Chen, Yizhou Jiang, and Feng Chen. Feature contamination: Neural networks learn uncorrelated features and fail to generalize. In *International Conference on Machine Learning*, pages 60446–60495. PMLR, 2024.

[51] Zhaoyang Zhang, Wenqi Shao, Jinwei Gu, Xiaogang Wang, and Ping Luo. Differentiable dynamic quantization with mixed precision and adaptive resolution. In *International Conference on Machine Learning*, pages 12546–12556. PMLR, 2021.

[52] Yaoyu Zhu, Zhaofei Yu, Wei Fang, Xiaodong Xie, Tiejun Huang, and Timothée Masquelier. Training spiking neural networks with event-driven backpropagation. *Advances in Neural Information Processing Systems*, 35:30528–30541, 2022.

NeurIPS Paper Checklist

1. **Claims**
   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?
   Answer: [Yes]
   Justification: Our main claim involves a post-training optimization pipeline for Spiking Neural Networks (SNNs), which results in significant reductions in inference time and overall power consumption. These improvements are validated through experiments.
   Guidelines:
   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.
2. **Limitations**
   Question: Does the paper discuss the limitations of the work performed by the authors?
   Answer: [Yes]
   Justification: The main limitation of our method is that the reduction in inference time and power consumption comes at the cost of increased spatial complexity. This trade-off is discussed in the Conclusion and Discussion section.
   Guidelines:
   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.
3. **Theory assumptions and proofs**
   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?
   Answer: [Yes]
   Justification: Our propositions and theorems are accompanied by corresponding proofs. Some of these proofs are provided in the supplementary materials, while a brief remark outlining the main idea is included in the main paper.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Details of our algorithm and model design are included in the appendix under the Guidelines and Experiment Settings sections.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We provide the corresponding code in the supplementary materials and plan to release it after the paper is accepted.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.

- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**
   Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?
   Answer: [Yes]
   Justification: Training details are provided within the submitted code, and experimental settings are included in the appendix.
   Guidelines:
   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.
7. **Experiment statistical significance**
   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?
   Answer: [Yes]
   Justification: An analysis of randomness in our experiments is provided in Table.3 in the Appendix.
   Guidelines:
   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.
8. **Experiments compute resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?
Answer: [Yes]
Justification: We report inference time and system energy consumption directly in Table.1 of the main paper, as these metrics are central to our contribution.
Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**
Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?
Answer: [NA]
Justification: No ethical issues are involved.
Guidelines:
   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).
10. **Broader impacts**
Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?
Answer: [NA]
Justification: No significant or foreseeable social impact is associated with our work.
Guidelines:
   - The answer NA means that there is no societal impact of the work performed.
   - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
   - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
   - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
   - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
   - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).
11. **Safeguards**
Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?
Answer: [NA]

Justification: Our work does not involve specific risks.
Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**
Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?
Answer: [NA]
Justification: Our work does not involve such models or datasets.
Guidelines:
    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.
13. **New assets**
Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?
Answer: [NA]
Justification: No new assets are introduced in this work.
Guidelines:
    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.
14. **Crowdsourcing and research with human subjects**
Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?
Answer: [NA]
Justification: No human subjects are involved in this study.
Guidelines:
    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: No human subjects are involved in this study.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: LLMs are only used for writing and language polishing.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

A Detailed Proofs and Explanations

This section provides proofs and explanations for theorems and propositions in Section.3, 4 and 5.

A.1 Explanations of Equations 5 and 6

We provide additional clarification on the motivation and interpretation of the two key equations used in ANN-to-SNN conversion preliminaries.

Equation 5: Weight Normalization for Hardware Compatibility. In the weighted-spike formulation introduced in Eq. 3, the spike output $s(t)$ takes values in $\{0, \theta\}$ instead of the binary set $\{0, 1\}$ assumed in standard IF neuron models. This discrepancy causes a mismatch when deploying SNNs on neuromorphic hardware or when aligning their computation with standard ANN formulations.

A common solution in training-free ANN-to-SNN conversion [28, 42] is to absorb the threshold scaling factor into the synaptic weights. For example, when a presynaptic neuron emits a spike with magnitude $V_{th}^{(\ell-1)}$ and the postsynaptic neuron fires if its membrane potential exceeds $V_{th}^{(\ell)}$, the firing condition is

$$V_{th}^{(\ell-1)}\mathbf{W}^{(\ell)} > V_{th}^{(\ell)}. \tag{25}$$

This inequality is equivalent to

$$\frac{V_{th}^{(\ell-1)}\mathbf{W}^{(\ell)}}{{V_{th}}^{(\ell)}} > 1, \tag{26}$$

indicating that the threshold $V_{th}^{(\ell-1)}$ can be absorbed into the synaptic weight, thereby fixing the threshold to 1 without changing the neuron dynamics.

Following the same principle, Eq. 5 performs a linear rescaling:

$$w := \theta w, \quad s(t) := \frac{s(t)}{\theta} = H(v(t) - \theta) \in \{0, 1\}, \tag{27}$$

which normalizes the spike outputs and ensures computational compatibility with hardware platforms that support only binary spikes. Conceptually, this step aligns the weighted-spike model with a standard $\{0, 1\}$ IF neuron model, while preserving equivalence in input-output behavior.

Equation 6: Firing-Rate Approximation as Quantized Activation. Equation 6 provides an interpretation of the firing rate of a spiking neuron in terms of a quantized ANN activation. In rate-coding SNNs, the firing rate over T time-steps,

$$s = \frac{1}{T}\sum_{t=1}^{T} s(t), \tag{28}$$

represents the activation intensity. Assuming that the total integrated input matches the ANN pre-activation a, i.e., $\sum_t q_{\text{snn}}(t) = q_{\text{ann}} = a$, the firing rate can be rewritten as:

$$s = \frac{a - v(T)}{T} = \frac{\theta \cdot \text{clip}\left(\left\lfloor \frac{a}{\theta} \right\rfloor, 0, T\right)}{T}. \tag{29}$$

This expression shows that the firing rate effectively corresponds to a T-level quantized version of the ReLU activation a. The term $\lfloor a/\theta \rfloor$ represents the discrete spike counts (bounded by T), while the residual membrane potential $v(T)$ reflects the quantization error. Such a formulation is widely used in quantization-based conversion approaches [28, 7], where the temporal accumulation of spikes is treated as a discretized approximation of continuous activations.

Furthermore, Eq. 6 reveals a useful design principle: since the quantization granularity is determined by θ, a larger simulation window T allows for a smaller threshold $\theta \approx \max(a)/T$, thereby enabling finer-grained representation of neuronal activation with minimal quantization error.

A.2 Additional Proof for Population Encoding Models

Proof for Prop.4.2. This is a special case of Prop.4.3 when the bit-length of a neuron group $k = 1$. See the proof of Prop.4.3 □

Proof for Prop.4.3.

Define $\left\{ s_{(1)}, \ldots, s_{(m)} \middle| s_{(i)} \in \mathcal{S}, s_{(1)} \leq \cdots \leq s_{(m)} \right\}$, where $m = (T+1)^k$ as an ordered sequence of all possible spike combinations. Consider $q_{(i)} \in \left[s_{(i-1)}, s_{(i+1)} \right)$. The errors within this interval can be calculated as:

$$\begin{aligned} r_{(i)} &= \int_{s_{(i-1)}}^{s_{(i)}} (q - s_{(i-1)}) dq + \int_{s_{(i)}}^{s_{(i+1)}} (q - s_{(i)}) dq \\ &= \left[s_{(i)} - \frac{1}{2}(s_{(i-1)} + s_{(i+1)}) \right]^2 + \frac{1}{4} \left[s_{(i-1)} - s_{(i+1)} \right]^2 \\ &\geq \frac{1}{4} \left[s_{(i+1)} - s_{(i-1)} \right]^2, \end{aligned}$$

where the equation is taken when $s_{(i)} = \frac{1}{2}(s_{(i-1)} + s_{(i+1)})$. Define $s_{(0)} = 0, s_{(m+1)} = q_{\max}$ as boundary conditions,

$$\begin{aligned} r =& \frac{1}{2q_{\max}} \left(\sum_{i=1}^{m} r_{(i)} + \frac{1}{2}(s_{(1)} - s_{(0)})^2 + \frac{1}{2}(s_{(m+1)} - s_{(m)})^2 \right) \\ \geq& \frac{1}{8q_{\max}} [2(s_{(1)} - s_{(0)})^2 + (s_{(2)} - s_{(0)})^2 + \cdots + (s_{(m+1)} - s_{(m-1)})^2 + 2(s_{(m+1)} - s_{(m)})^2] \\ =& \frac{1}{8q_{\max}} \cdot \frac{4(s_{(m+1)} - s_{(0)})^2}{(\frac{1}{2} + 1 + \cdot + \frac{1}{2})} \\ =& \frac{1}{8q_{\max}} \cdot \frac{4q_{\max}^2}{(T+1)^k} \\ =& \frac{q_{\max}}{2(T+1)^k} \end{aligned}$$

where the equation is taken when $s_{(i)} = \frac{1}{2}(s_{(i-1)} + s_{(i+1)})$ holds for all $i = 1, \ldots, m$. Thus, we have

$$\begin{aligned} b = s_1 &= \frac{q_{\max}}{2(T+1)^k}, \\ \theta_i &= \frac{q_{\max}}{(T+1)^i} \end{aligned}$$

□

Proof for Prop.4.4. Consider multiple neurons operating at the final single step, setting $T := 1$ and $q := v_{temp}(T)$. Substitute these into Eq.9 to obtain the above result. □

A.3 Proof for Theorem.5.1

Proof. For clarity, we let $\theta_{old} = 1$ and denote $\theta_{k+1} = \theta$, consider $v \in [0, \theta_k')$ in most cases. The reduction caused by fission is given by:

$$\Delta v = v - v' = \begin{cases} 0 & 0 \leq v < \theta \\ \theta & \theta \leq v < 1 - \theta \\ 1 - \theta & 1 - \theta \leq v < 1 \end{cases}$$

Using the CDF to represent the piece-wise function, we can rewrite the error reduction on sample batch. The optimization target is defined as

$$\begin{aligned}
&\mathbb{E}[\Delta v] = \int \Delta v \cdot p(v) dv \\
=&\theta \int I_{\theta \le v < 1-\theta} \cdot p(v) dv - (1-\theta) \int I_{1-\theta \le v < 1} \cdot p(v) dv \\
=&\theta \int_{\theta}^{1-\theta} p(v) dv - (1-\theta) \int_{1-\theta}^{1} p(v) dv \\
=&\left[F(1-\theta) - F(\theta)\right] \theta + \left[1 - F(1-\theta)\right] (1-\theta).
\end{aligned}$$

Given $\theta = \arg\max_\theta \mathbb{E}[\Delta v]$, the extremum can be obtained by derivation. At the maximum point, we have:

$$\frac{d\mathbb{E}[\Delta v]}{d\theta} = [2F(1-\theta) - F(\theta) - 1] - \theta \cdot [2p(1-\theta) + p(\theta)] + p(1-\theta) = 0,$$

where p is the probability density function. Considering that in actual sampling, the distribution of v is approximately discrete, we assume $p(\theta) = p(1-\theta) = 0$. Thus we have:

$$\begin{aligned}
2F(1-\theta) - F(\theta) - 1 &= 0 \\
\mathbb{E}\Delta v = 1 - F(1-\theta) &= F(1-\theta) - F(\theta)
\end{aligned}$$

yielding the result in Eq.18. We can further estimate $\mathbb{E}v$ by piece-wise scaling

$$\begin{aligned}
0 \le \int_0^{\theta} v dv &\le \theta F(\theta), \\
\theta \left[F(1-\theta) - F(\theta)\right] \le \int_{\theta}^{1-\theta} v dv &\le (1-\theta)[F(1-\theta) - F(\theta)], \\
(1-\theta)(1 - F(1-\theta)) \le \int_{1-\theta}^{1} v dv &\le 1 - F(1-\theta).
\end{aligned}$$

Substituting this to $\mathbb{E}[v]$ and $\mathbb{E}[\Delta v] = 1 - F(1-\theta)$ back to $\mathbb{E}v$ can derive the range for $\mathbb{E}[\Delta v]$. □

Algorithm 1 Overall Algorithm

Input: Original SNN; time-step T

```
Set fission rate f, required accuracy p%.
Collect a batch of input data x^(j)
while Validation accuracy < p% do
  for each feature dimension fea do
    Conduct SNN forward propagation for T steps to calculate residual potential v_fea(T).
    Conduct ANN forward and backward propagation to calculate gradient da_out/da_fea.
  end for
  Calculate sensitivity σ_l with v_fea(T) and da_out/da_fea. (cf. Eq.16)
  Determine threshold σ_th as the f-percentile of all σ_l
  for each layer l = 1, 2, ..., m in the SNN do
    if sensitivity of i-th neuron σ_li ≥ σ_th then
      Perform fission encoding (cf. Theorem.5.1) and obtain new threshold sets θ_li of post-fission
      neuron group
    end if
  end for
end while
```

Output: Threshold sets $\boldsymbol{\theta}_{li}$ for all fission neurons

B Guidelines for the Overall Pipeline

We first provide the algorithm for our pipeline described in Section.5. The fission rate f should be dynamically adjusted based on the network architecture and the desired reduction in time-steps. For modest reductions in time-steps (e.g., halving the time-steps), setting $f \approx 15\%$ is typically effective. For more substantial reductions, f can be incrementally increased by 15%, and further tuning may be required based on empirical performance. In the case of an ANN-to-SNN conversion with 32 time-steps, where the inference requires at least 8 time-steps (i.e., reducing to one-quarter of the original time-steps), a higher threshold of $f \approx 80\%$ is usually sufficient. This can often be achieved with just one round of fission encoding. However, for more aggressive reductions, where fewer time-steps are required (e.g., reducing to 2 or 4 steps), the fission rate f should be lowered to around 35%, and multiple rounds (typically 2 to 4 iterations) of fission encoding may be necessary to ensure performance convergence. The fission rate should be continuously adjusted based on the specific network architecture and the corresponding trade-offs between computational efficiency and accuracy. Empirical evaluation is recommended for fine-tuning these parameters in different settings.

C Spatial Complexity Analysis

We analyze the spatial and computational implications of neuron fission from four complementary perspectives:

Memory Overhead. Consider a fully connected layer $Linear(n, n)$ with n thresholds, n membrane potentials, and n^2 synaptic weights. After fission with an average rate k, each original neuron is split into $k + 1$ neurons. These fissioned neurons *share the same input* (membrane potential) but have *independent outputs*, which determines how overhead arises:

- *Thresholds:* Increase from n to $n(k + 1)$, one per fissioned neuron.
- *Membrane Potentials:* Theoretically remain n since inputs are shared, but in practice duplicated to $n(k + 1)$ for matrix computation efficiency.
- *Synapses:* Each new neuron contributes n additional outgoing synapses, resulting in an increase of $nk \times n$ weights.

Thus, memory overhead grows **linearly** with the fission rate k across all components. In deployment, compiler optimizations such as in-place computation and buffer reuse typically reduce actual memory usage below this theoretical bound.

Computation Complexity (CMPs and ADDs). The transition from cascade to parallel firing indicators introduces additional comparisons but dramatically reduces additions. For a group of n split neurons, the parallel model requires 2^{i-1} comparisons (CMPs) and one addition (ADD) for the i-th neuron, leading to a total of $2^n - 1$ CMPs and n ADDs for 2^n-level precision. In contrast, achieving the same precision with a conventional parallel scheme would require $2^n - 1$ neurons, each with one CMP and ADD, resulting in $2^n - 1$ CMPs and $2^n - 1$ ADDs overall. While CMPs grow exponentially in both cases, ADDs — typically more expensive — grow only linearly in our method. Although a cascade implementation would reduce comparisons to n, its sequential nature hinders parallel execution and is thus unsuitable for hardware deployment.

Hardware Implications. Threshold comparisons align well with neuromorphic architectures, where CMP circuits are often implemented as dedicated units due to inherent spiking dynamics. Comparators typically consume 3–5$\times$ less energy and have lower delay and area than adders. As a result, the CMP-dominant computation pattern introduced by Adaptive Fission is well matched to hardware execution. Moreover, per-step complexity increases are offset by exponential reductions in time-steps, yielding overall computational efficiency gains.

Power and On-Chip Efficiency. Measured power profiles show that a significant portion of neuromorphic ASIC energy is consumed by I/O and memory access rather than arithmetic computation. Consequently, the growth in CMP operations has limited impact on total power consumption, while reduced time-steps translate into substantial energy savings. This trade-off — slightly higher per-step complexity for significantly improved temporal efficiency — is generally advantageous for neuromorphic deployment.

Hardware Compatibility. All neurons, including fissioned ones, are implemented as modular IF units with segmented step functions expressed via CMP-based firing logic. Spikes remain binary

$\{0, \theta_i\}$ and are normalized to $\{0, 1\}$ during deployment. Hardware experiments on our prototype neuromorphic chip confirm that Adaptive Fission integrates seamlessly without requiring custom logic paths.

D Discussion on spike-based ViTs and LLMs

Our method is applicable to spike-based vision transformers, as demonstrated with the Spike-driven Transformer in the main paper. However, many existing models still include floating-point operations (e.g., Softmax), limiting full compatibility with neuromorphic hardware. We believe that future spike-based transformers should avoid such nonlinearities to enable efficient deployment. We further tested our method on the Spatio-temporal Approximation model [24], a floating-point-free ViT-B/32 backbone from CLIP, used for zero-shot classification. Results on both GPU and the HP201 ASIC are shown in Table 2. While basic operations executed successfully, limited compiler support led to low execution efficiency, preventing meaningful power measurements.

Method	Time-Step	Fission Rate	Accu. (%)
CIFAR-100 & CLIP-pretrained ViT-B/32 Zero-shot			
STA	64	No	85.25
	32		84.15
+ Fission	16	0.53	82.90
	8	1.15	81.48

Table 2: Accuracy comparison of pretrained ViT-B/32 by CLIP for Zero-shot classification.

Applying our method to large language models (LLMs) faces additional challenges:

Attention precision: Attention scores typically require at least 8-bit precision, translating to long inference durations in SNNs. Adaptive Fission can help alleviate this, though memory costs may become more noticeable.

Error accumulation: The scale of LLMs amplifies small precision errors. More refined fission rate allocation strategies may be needed across network depth to balance overhead and accuracy.

E Experiment Settings

E.1 Hardware Resources

All our experiments were conducted on a workstation with Intel-13900KS, 32GB memory with a single 4090 GPU and a Lynxi HP201 neuromorphic accelerator. As a commercial version of Tianjic chip [34], HP201 adopts a many-core decentralized architecture, commonly utilized in neuromorphic chips, and integrates 60 configurable functional cores, 16 GB of memory, and image encoding/decoding units. This design allows it to support quantized ANN/SNN models of up to approximately 13 billion parameters. The chip is packaged in a PCIe form factor with a peak power consumption of 55 W. The provided LynBIDL library enables the compilation of SNN models based on PyTorch. However, since the underlying interfaces are not open, we are unable to directly measure the power consumption of individual cores or the number of MAC/AC operations during runtime. Instead, we record the total energy consumption using power readings from the sensors. While this method inevitably introduces interference (including power consumption from peripheral circuits, memory, and I/O interfaces), it reflects a more practical scenario. In such a case, reducing inference time significantly contributes to lowering power consumption, as the energy usage of peripheral circuits is largely determined by runtime duration.

Notably, while fission encoding can increase the number of neurons for single activation value, it does not alter the network's original parameters. As a result, the memory consumption typically increases by less than twice that of the baseline model.

E.2 Implementation of Classification

The baseline models and network architectures used in fission are directly sourced from the original codebase, including the Calibration [28] and QCFS [4] conversion methods, and spike-driven transformers [49]. We slightly modify the TEBN [10] hyperparameters to suit ResNet20 and VGG16, as the original experiments were conducted on ResNet19 and VGG11. Details can be found in the code.

E.3 Implementation of Image Generation

For the image generation model, we employ a 4-layer convolutional architecture based on the improved Wasserstein GAN (WGAN-GP) [13]. The architecture is structured as follows:

- ConvTranspose2d(100, 1024), BatchNorm2d, ReLU,
- ConvTranspose2d(1024, 512), BatchNorm2d, ReLU,
- ConvTranspose2d(512, 256), BatchNorm2d, ReLU,
- ConvTranspose2d(256, 3)

All convolutional transpose layers use a kernel size of 4. For the first layer, the stride is set to 1 and padding to 0, while for the remaining layers, the stride is 2 and padding is 1. This model is trained on the CIFAR-10 dataset using the WGAN-GP framework [13]. After training, the model is converted to a spiking neural network (SNN) representation by discretizing the output into 64 time-steps using the Calibration (Calib.) method.

It is important to note that direct training of SNNs for image generation tasks, especially with lower quantization precision, typically fails to achieve satisfactory performance, as demonstrated by the difficulty in generating high-quality images. Consequently, methods such as fission encoding are employed to improve inference performance in this context.

For the diffusion model, we follow the implementation described by Ho et al. [19]. The model consists of 4 residual blocks with a channel configuration of [2, 4, 4, 2], and uses the Swish activation function instead of ReLU. This prevents a direct conversion of the model to an Integrate-and-Fire (IF) spiking neuron network. To address this, we introduce an IF layer after each Swish activation, thereby discretizing the continuous output into spike events.

The diffusion model is trained on CIFAR-10 with 500 iterations for image generation. After training, the model is also converted to 64 time-steps using Calib. However, it is worth noting that the ANN-based diffusion model requires 500 iterations for each image generation, which results in a corresponding SNN model requiring 64 $\times$ 500 = 32,000 iterations. This computational cost is prohibitively high for real-time inference and necessitates the use of fission encoding to reduce latency.

For evaluating experimental results, we used the Peak Signal-to-Noise Ratio (PSNR) to measure the difference between the generated images and those from the original ANN model. The PSNR is calculated as follows:

$$PSNR = 10 \cdot \log_{10} \left(\frac{MAX^2}{MSE} \right), \tag{30}$$

where MAX represents the maximum possible pixel value in the image data. For 8-bit images, $MAX = 255$. The Mean Squared Error (MSE) is defined as the average of the squared differences between the pixel values of the generated image and the reference image, and is calculated as follows:

$$MSE = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} [I(i,j) - K(i,j)]^2, \tag{31}$$

where $I(i,j)$ and $K(i,j)$ represent the pixel values at position (i,j) in the generated image and the reference image, respectively, and m and n denote the height and width of the images.

E.4 Random error and Reproducibility

Fission Encoding employs a small batch of examples for Sensitivity detection and estimates each neuron's cumulative density function, which introduces some variability. By allocating an additional validation set, we can assess the optimal execution, which is then applied to the test set. In most of our experiments, we selected the best solution from 10 runs on the validation set. Table 3 displays the best and worst cases under these conditions, highlighting relatively significant performance variability.

Time-step	10%	20%	40%	80%	160%	320%
T=2	12.14-15.77	12.02-16.91	17.76-21.84	25.14-30.22	48.79-51.67	60.43-66.55
T=4	14.06-22.65	18.10-24.38	27.69-31.74	50.67-57.69	67.35-72.80	70.44-74.73
T=8	35.51-38.70	36.32-42.53	52.69-57.40	68.11-73.26	69.32-72.54	70.68-73.07
T=16	68.65-71.43	72.74-76.10	73.31-76.21	73.55-75.90	72.42-76.79	74.51-75.84

Table 3: The best and worst accuracy range of fission rate at different time-steps within 10 runs.

F Raw Data for Figures

In this section, we present the raw data corresponding to the figures in the main paper. The data points in Fig.4 are provided by Table 1 along with the supplementary Table 4 here. Table 5 corresponds to Fig.8.

Time-step	10%	20%	40%	80%	160%	320%
T=2	15.77	16.91	21.84	30.22	51.67	66.55
T=4	22.65	24.38	31.74	57.69	72.80	74.73
T=8	38.70	42.53	57.40	73.26	72.54	73.07
T=16	71.43	76.10	76.21	75.90	76.79	75.84

Table 4: Accuracy comparison of fission rate at different time-steps. The raw model is ResNet20 trained with 32 time-steps on CIFAR-100 with calibration.

Method	Fission (%)	Accuracy
Group [30]	100	66.94
	400	71.67
Sensitivity + Group	100	69.39
	150	72.85
Spatial [24]	100	52.51
	600	66.43
Sensitivity + Spatial	100	63.37
	250	68.61
Fission (Ours)	100	**68.22**
	200	72.40
Sensitivity + Fission	100	**72.79**
	78	73.28

Table 5: Ablation and Comparison of different neuron selection and group combining methods.

G Experimental Visualization and Analysis

This section provides a detailed analysis of the various components of the fission encoding method, including demonstrations of its effects on neurons within actual networks, the distribution of neurons across different layers after fission, and the distribution of sensitivity. Additionally, we provide more results of image generation tasks in a high-resolution format.

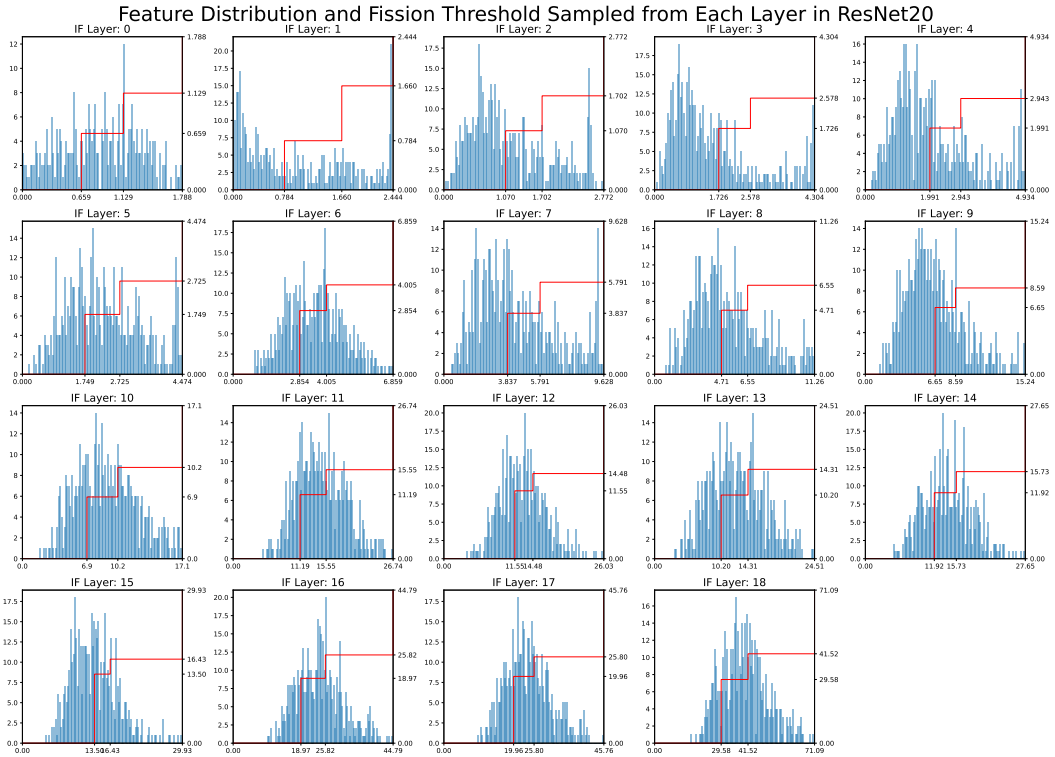


Figure 9: Feature distribution and their fission threshold in ResNet20. The feature distributions presented are randomly sampled from the more sensitive neurons across various layers of the network. The horizontal axis represents the feature distribution, the left vertical axis (associated with the bar chart) indicates the distribution density, while the right vertical axis (corresponding to the red line) reflects the fission threshold and the activation function resulting from the combination of neurons post-fission.

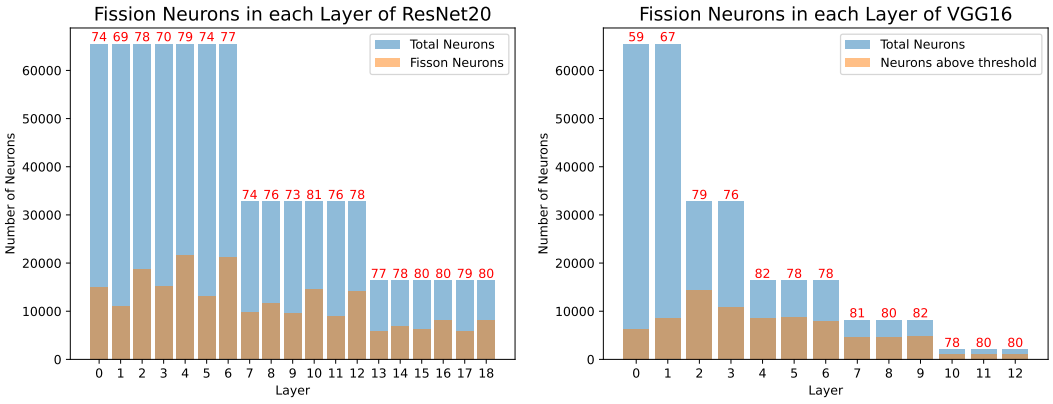


Figure 10: Distribution of Fission Neurons on all ReLU activation layers in ResNet20 and VGG16, when 30% of neurons undergo fission in a single round. The distribution shows that although the number of activation values significantly decreases in the network, the number of fission neurons only slightly reduces, while their proportion increases in deeper layers.

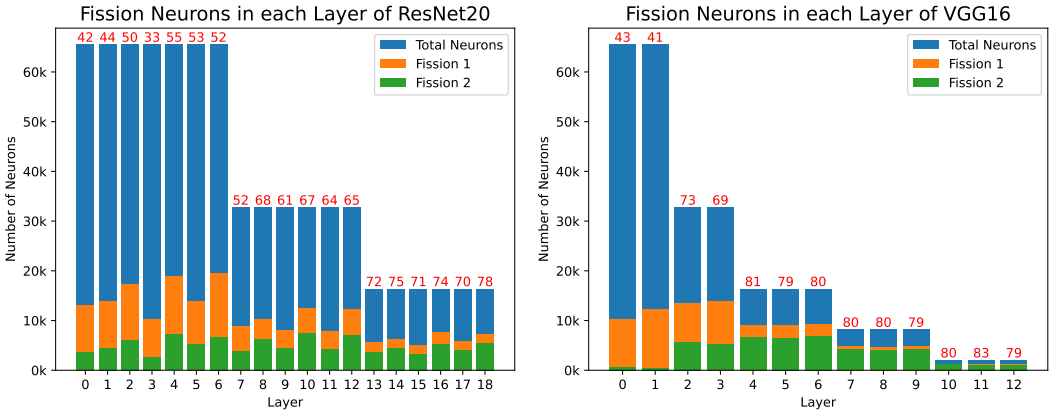


Figure 11: Distribution of Fission Neurons across ReLU activation layers in ResNet20 and VGG16 after two rounds of fission. The data indicates that the proportion of neurons undergoing multiple fission rounds is higher in the deeper layers, suggesting an increasing demand for feature precision in the deeper network layers.

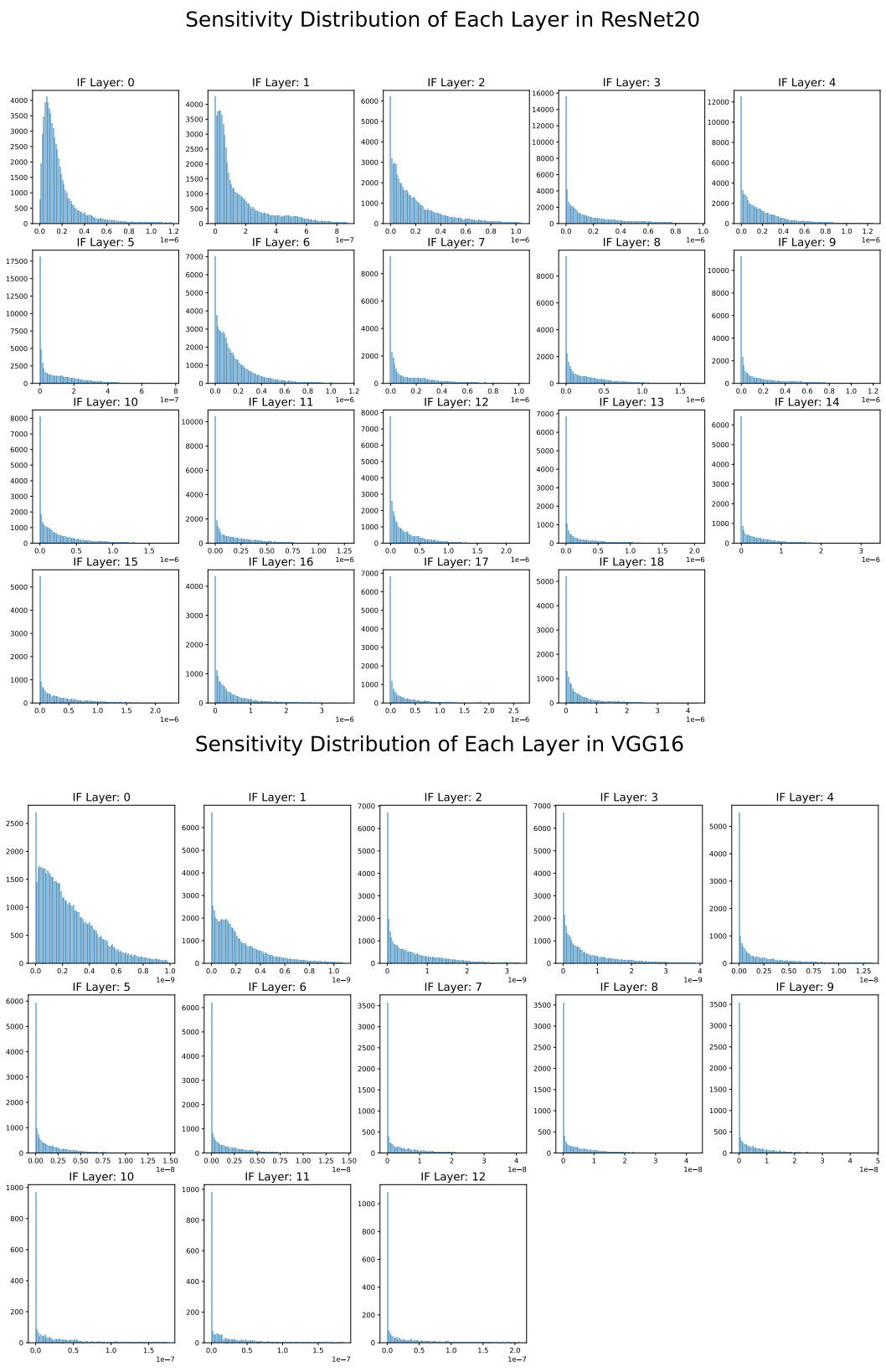


Figure 12: Sensitivity in ResNet20 and VGG16. The distributions is characterized by a long-tail distribution. The majority of neurons have a sensitivity of zero, indicating that fission is unnecessary for them. However, due to variations in sample sampling, there may be some random error in this distribution.

Figure 13: Image generation results of WGAN and PSNR for CIFAR-10 at different inference time-steps.

Figure 14: Image generation results of diffusion U-Net and PSNR for CIFAR-10 at different inference time-steps.