

## 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

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

This inequality is equivalent to

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

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:

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

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,

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

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:

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

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  $\theta$ , 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  $\square$

*Proof for Prop.4.3.*

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

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

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,

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

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

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

$\square$

*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.  $\square$

## 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

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

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

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

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  $\mathbb{E}[\Delta v]$ .  $\square$

---

#### 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  $\frac{da_{out}}{da_{fea}}$ .

**end for**

Calculate sensitivity  $\sigma_l$  with  $v_{fea}(T)$  and  $\frac{da_{out}}{da_{fea}}$ . (cf. Eq.16)

Determine threshold  $\sigma_{th}$  as the  $f$ -percentile of all  $\sigma_l$

**for** each layer  $l = 1, 2, \dots, m$  in the SNN **do**

**if** sensitivity of  $i$ -th neuron  $\sigma_{li} \geq \sigma_{th}$  **then**

      Perform fission encoding (cf. Theorem.5.1) and obtain new threshold sets  $\theta_{li}$  of post-fission neuron group

**end if**

**end for**

**end while**

**Output:** Threshold sets  $\theta_{li}$  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  $3\text{--}5\times$  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
-----			
	16	0.53	82.90
<b>+ Fission</b>	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 [48]. 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:

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

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:

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

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	<b>68.22</b>
	<b>200</b>	72.40
Sensitivity + Fission	100	<b>72.79</b>
	<b>78</b>	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.

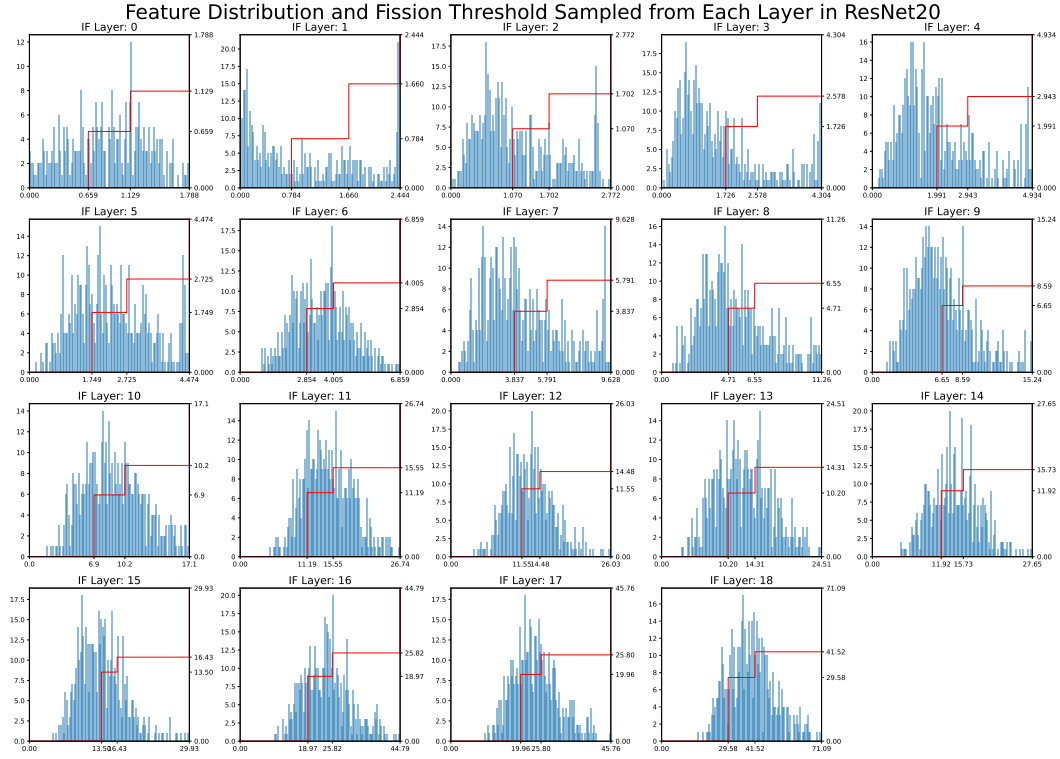


Figure 9: Feature distribution and their fission threshold in ResNet20. The feature distributions presented are randomly sampled from the more sensitive neurons across various layers of the network. The horizontal axis represents the feature distribution, the left vertical axis (associated with the bar 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.

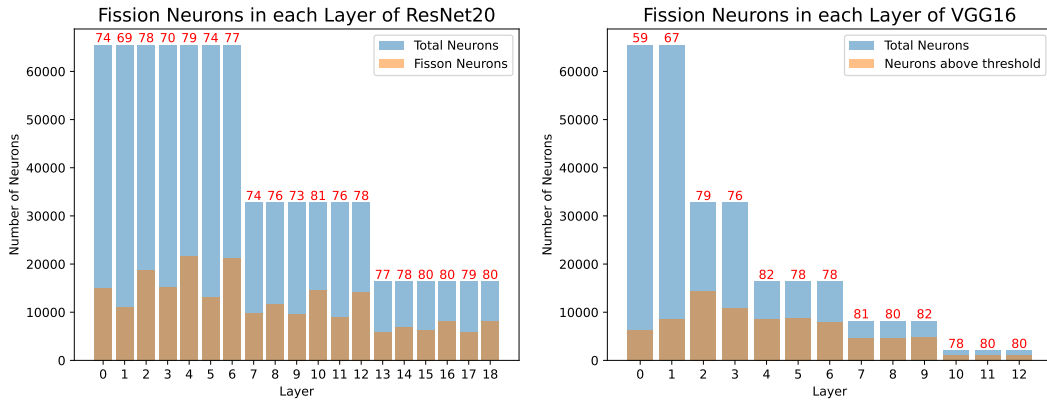


Figure 10: Distribution of Fission Neurons on all ReLU activation layers in ResNet20 and VGG16, when 30% of neurons undergo fission in a single round. The distribution shows that although the number of activation values significantly decreases in the network, the number of fission neurons only slightly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



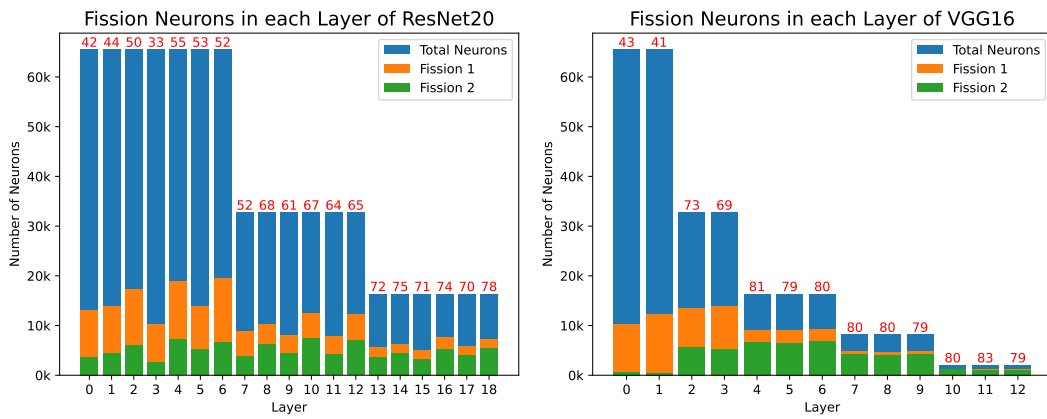
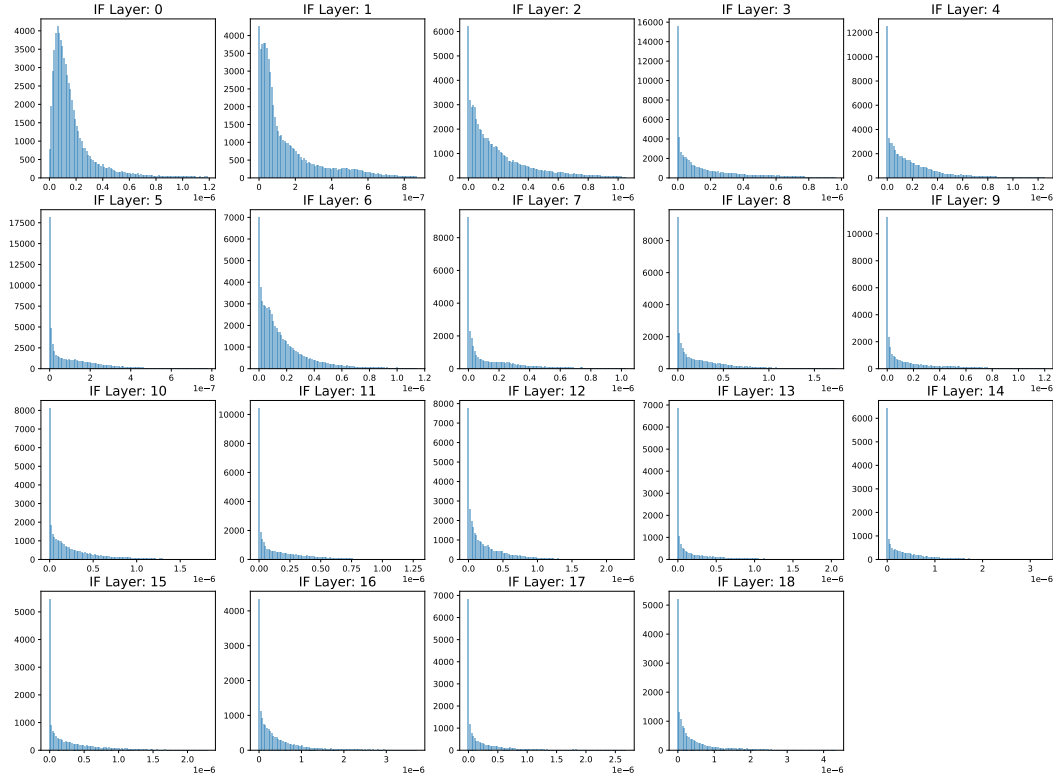


Figure 11: Distribution of Fission Neurons across ReLU activation layers in ResNet20 and VGG16 after two rounds of fission. The data indicates that the proportion of neurons undergoing multiple fission rounds is higher in the deeper layers, suggesting an increasing demand for feature precision in the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

### Sensitivity Distribution of Each Layer in ResNet20



### Sensitivity Distribution of Each Layer in VGG16

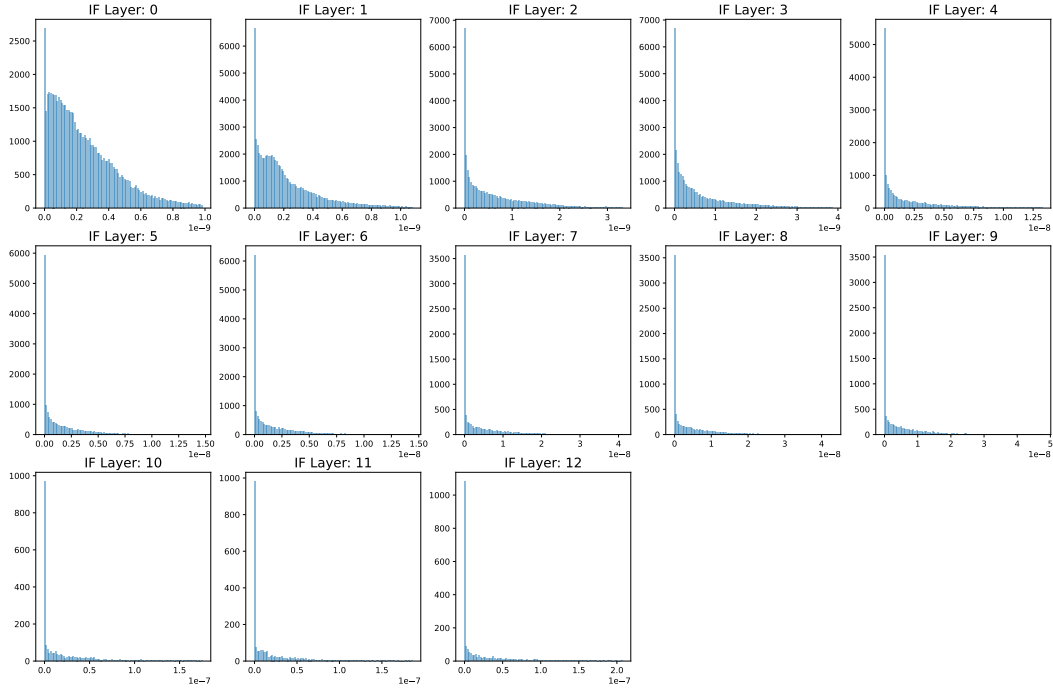


Figure 12: Sensitivity in ResNet20 and VGG16. The distributions is characterized by a long-tail distribution. The majority of neurons have a sensitivity of zero, indicating that fission is unnecessary for them. However, due to variations in sample sampling, there may be some random error in this distribution.



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.