SPIKE-BY-SPIKE DIFFUSION: TRAJECTORY-LEVEL ANN-TO-SNN CONVERSION FOR ANYTIME TEXT-TO-IMAGE GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

ANN-to-SNN conversion provides a weight-preserving route to spike-based diffusion inference, but existing methods execute each denoising stage in a stage-major manner: all SNN time steps are completed before advancing to the next stage. This execution order largely removes the temporal flexibility of SNNs, since no terminal image is available until the final denoising stage. We introduce *Spike-by-Spike Diffusion*, a trajectory-level conversion framework that makes the complete sampling trajectory the unit of temporal refinement. Each global SNN step traverses all denoising stages and produces a terminal latent prefix that can be decoded immediately and further refined within the same run. To support this execution, we reformulate affine diffusion schedulers into incremental updates with one-shot stochastic injection, and introduce a step-sensitive multi-threshold neuron with layer–stage base calibration and stage-specific temporal threshold schedules for accurate early readout. Experiments on Stable Diffusion v1.5 with LCM-LoRA and SANA-Sprint show that our method produces high-quality terminal outputs substantially earlier than stage-major conversion and progressively approaches the ANN reference as computation increases.

1 INTRODUCTION

Text-to-image diffusion models generate high-quality images through repeated evaluations of a large denoising network (Rombach et al., 2022; Saharia et al., 2022). Few-step samplers reduce this cost by shortening the denoising trajectory through improved solvers or distillation (Lu et al., 2022; Salimans & Ho, 2022; Luo et al., 2023a). ANN-to-SNN conversion offers a complementary direction: it reuses pretrained weights while replacing continuous activations with temporally accumulated spike outputs, enabling spike-based inference without retraining the diffusion model (Diehl et al., 2015; Rueckauer et al., 2017).

A converted diffusion sampler therefore contains two computational axes: the denoising-stage axis $d \in \{1, \dots, D\}$ and the SNN time axis $t \in \{1, \dots, T\}$. Existing spiking diffusion methods organize these axes in a *stage-major* order (Cao et al., 2024; Fang et al., 2026): all T SNN steps are completed at one denoising stage before the sampler advances to the next. As illustrated in Figure 1, temporal accumulation is consequently local to the active denoiser evaluation. Intermediate SNN outputs refine only that stage, while a complete generation output becomes available only after reaching the final denoising stage. This execution order largely removes the temporal flexibility that makes SNN inference attractive: computation spent at intermediate SNN steps cannot be inspected as a terminal image, and increasing the SNN budget requires completing the preceding stages again.

We introduce *Spike-by-Spike Diffusion*, which makes the complete sampling trajectory, rather than an individual denoiser evaluation, the unit of SNN temporal refinement. At each global SNN time step, we advance all D denoising stages once in sampling order and propagate the resulting increments to the trajectory endpoint. Each traversal produces a terminal latent increment, which is accumulated into a terminal prefix $\hat{z}_0^{[t]}$. A complete image can therefore be read out after every SNN time step, while subsequent traversals continue refining the same generation without restarting inference.

Realizing this execution order requires incrementalizing the diffusion sampling procedure itself. A downstream denoising stage consumes a scheduler-transformed latent, whereas an intermediate SNN

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

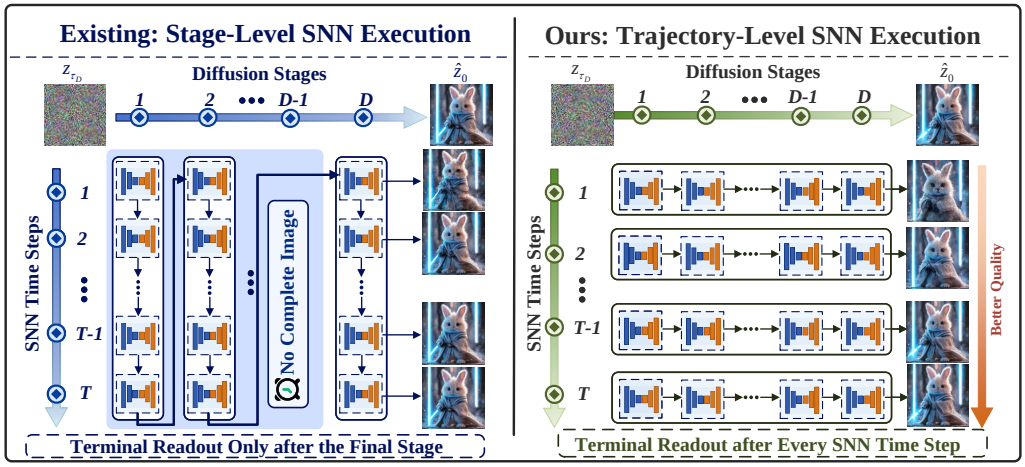


Figure 1: **Stage-major versus trajectory-level SNN execution.** Existing conversion accumulates SNN time within each denoising stage, whereas Spike-by-Spike Diffusion propagates every SNN step through the complete sampling trajectory and exposes a terminal readout after each traversal.

step provides only latent and model-output increments. We therefore derive an SNN-compatible scheduler for affine sampling transitions that propagates these increments directly across stages. Stochastic transition noise is injected once, and accumulation over SNN time recovers the original scheduler evaluated on the accumulated inputs. This enables all terminal prefixes to remain on the same sampling trajectory.

Early terminal readout is useful only when short SNN prefixes are sufficiently accurate. We observe that denoising stages exhibit different activation ranges, while different spatial frequencies develop at different rates over SNN time. Motivated by these observations, we introduce a step-sensitive multi-threshold neuron whose base thresholds are calibrated separately for each layer–stage pair and modulated by a stage-specific temporal schedule. This separates stage-dependent numerical calibration from temporal threshold adaptation, improving early-prefix fidelity while keeping all pretrained model weights fixed.

We evaluate Spike-by-Spike Diffusion on Stable Diffusion v1.5 with LCM-LoRA and SANA-Sprint across COCO and MJHQ-30K. Across both model families, our method provides high-quality terminal outputs substantially earlier than stage-major conversion and continues improving them within the same run. On four-stage SANA-Sprint, a $t = 4$ prefix already reaches an FID of 7.80 before the stage-major baseline exposes its first terminal output, while the final prefix reaches 6.96 versus 7.87 for the baseline. The first terminal readout uses only 14.85%–26.44% of the full-run energy across the evaluated configurations, leaving most of the computation budget available for progressive refinement.

Our contributions are as follows:

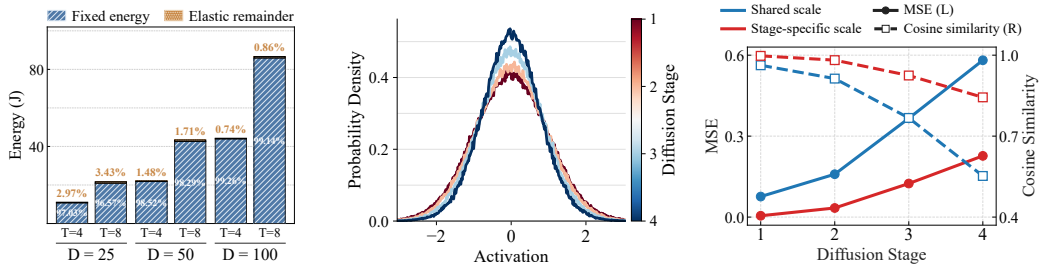
- We introduce a trajectory-level ANN-to-SNN conversion framework that makes the complete diffusion trajectory the unit of temporal accumulation, enabling terminal image readout and progressive refinement after every SNN time step.
- We derive an SNN-compatible incremental scheduler for affine diffusion transitions, allowing latent and model-output increments to propagate across denoising stages while preserving accumulation consistency with the original sampler.
- We develop a step-sensitive multi-threshold neuron with layer–stage base calibration and stage-specific temporal schedules, improving the fidelity of early SNN prefixes without updating pretrained model weights.
- Across Stable Diffusion v1.5 and SANA-Sprint, we demonstrate that trajectory-level conversion produces high-quality terminal outputs substantially earlier than stage-major execution and enables flexible trade-offs between generation quality and computation within a single generation run.

2 Related Work

ANN-to-SNN conversion. ANN-to-SNN conversion reuses pretrained weights and approximates ANN activations through temporal spike accumulation. Early methods relied on weight normalization and threshold balancing (Diehl et al., 2015; Rueckauer et al., 2017; Sengupta et al., 2019), while later work improves low-latency conversion through finite-step error analysis, activation mapping, threshold calibration, and membrane-potential initialization (Deng & Gu, 2021; Li et al., 2021; Liu et al., 2022a; Bu et al., 2022; Hao et al., 2023; Bu et al., 2023; Jiang et al., 2023). These methods primarily optimize conversion within an individual network evaluation, whereas our work reorganizes temporal accumulation across a sequence of denoiser evaluations.

Spiking generative and diffusion models. Spiking generation has been explored with fully spiking autoencoders (Kamata et al., 2022), discrete latent diffusion (Liu et al., 2024), and ANN-to-SNN conversion for diffusion models (Cao et al., 2024). Most closely related, SpikingDiT converts pretrained Diffusion Transformers using multi-threshold neurons and membrane-potential error feedback across denoising stages (Fang et al., 2026). Existing methods nevertheless retain stage-major execution, completing the SNN simulation of each denoising stage before advancing. Spike-by-Spike Diffusion instead propagates each SNN step through the full sampling trajectory, enabling terminal readout and progressive refinement throughout a single run.

Few-step diffusion sampling. Numerical samplers such as DDIM, PNDM, and DPM-Solver (Song et al., 2020; Liu et al., 2022b; Lu et al., 2022), together with progressive distillation and consistency-based methods (Salimans & Ho, 2022; Song et al., 2023; Luo et al., 2023a), reduce the number of denoising stages. Our method operates on a complementary axis: it restructures SNN temporal accumulation within the remaining denoising trajectory.



(a) First-readout energy (b) Stage-dependent ranges (c) Trajectory fidelity

Figure 2: **First-readout energy and stage-dependent scales. (a)** Stage-major execution exposes the first terminal output only after most energy is consumed. **(b)** Denoiser-output ranges vary across diffusion stages. **(c)** Stage-specific calibration better preserves the ANN trajectory.

3 Observations and Motivations

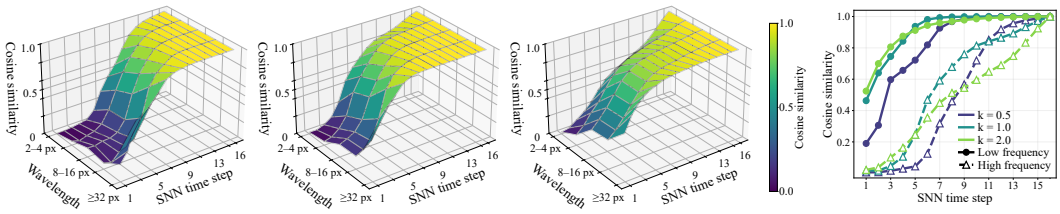
3.1 Stage-Major Execution Delays Terminal Readout

Stage-major execution completes all T SNN steps at one denoising stage before advancing to the next. Intermediate computation therefore refines only the active denoiser evaluation, and a terminal output becomes available only after reaching the final stage. For a trajectory with D denoising stages, the first terminal readout requires $(D-1)T+1$ of the DT one-step denoiser updates, leaving only $T-1$ updates for subsequent refinement. This window does not grow with D: longer trajectories increase the computation preceding the first readout without increasing the computation available afterward. Consistent with this structure, Figure 2(a) shows that the first terminal readout of stage-major SpikingDiT already consumes 96.57%–99.26% of the full-run energy across the profiled configurations. Appendix A.1 provides the profiling protocol and energy accounting.

Observation 1: Stage-major execution confines terminal-output refinement to the final denoising stage.

Insight 1: Make the complete sampling trajectory the unit of temporal refinement. Advancing all D stages once per global SNN step produces the first terminal estimate after D one-step denoiser

updates and another after every subsequent traversal, enabling stopping or continued refinement within the same run.



(a) Fixed $k = 0.5$ (b) Fixed $k = 1.0$ (c) Fixed $k = 2.0$ (d) Frequency alignment

Figure 3: **Thresholds shape temporal frequency development. (a–c)** Spectral alignment over SNN time for fixed $k \in \{0.5, 1.0, 2.0\}$. **(d)** Coarsest and finest bands across the three threshold settings.

3.2 Thresholds and Early-Readout Fidelity

Trajectory-level execution makes terminal outputs available early, shifting the challenge from readout availability to readout fidelity. We examine two factors that govern this fidelity: the numerical ranges encountered at different denoising stages and the effect of firing thresholds on temporal image development.

Numerical ranges vary across denoising stages. Although all stages share the same denoiser, their latent inputs and noise-level conditioning evolve along the sampling trajectory. For Stable Diffusion v1.5 with four-stage LCM-LoRA sampling, the ANN denoiser-output distribution progressively narrows across stages (Figure 2(b)). From Stage 1 to Stage 4, the standard deviation decreases from 0.983 to 0.788, while the 99th percentile of the absolute magnitude decreases from 2.535 to 2.088. Shared network weights therefore do not imply a shared activation range. We further compare converted trajectories using shared and stage-specific activation scales. Stage-specific scales consistently reduce latent MSE and increase cosine similarity to the ANN trajectory (Figure 2(c)), motivating calibration with stage-dependent activation statistics.

Activation ranges and firing thresholds. In ANN-to-SNN conversion, activation statistics provide the numerical reference for the spiking representation, while the firing threshold controls how this range is expressed at finite SNN horizons. We write $\theta = kx_{\max}$, where $x_{\max}$ is the calibrated activation range and k sets the threshold magnitude relative to that range. With n magnitude levels per sign, the available event magnitudes span from θ to $\theta/2^{n-1}$. Thus, $x_{\max}$ determines the numerical range, whereas k controls the corresponding range–resolution trade-off.

Thresholds shape temporal image development. Keeping the calibrated activation ranges fixed, we compare $k \in \{0.5, 1.0, 2.0\}$ under identical prompts, sampling settings, random draws, and threshold-level counts. Each k remains fixed throughout its $T = 16$ run. We decompose each decoded readout into spatial-frequency bands and measure its spectral alignment with the final output from the same run. Figure 3(a–c) reveals a consistent coarse-to-fine progression: coarse spatial components approach their final representations earlier, while finer components continue developing over later SNN steps. Panel (d) further shows that changing k shifts the timing of this progression. A threshold setting that favors early coarse-structure formation is not uniformly favored during later fine-detail refinement, indicating that a fixed threshold coefficient is suboptimal throughout temporal accumulation. Appendix A.2 provides the detailed protocol and analysis.

Observation 2: Denoising stages exhibit different activation ranges, and stage-specific scale calibration better preserves the ANN trajectory. With these ranges fixed, threshold selection changes the temporal development of coarse and fine image content.

Insight 2: Use stage-dependent activation statistics to anchor the numerical range, calibrate the base threshold coefficient relative to this range, and adapt the firing threshold over SNN time to support early structure formation and subsequent detail refinement. Since denoising stages operate on distinct trajectory states, both base calibration and temporal adaptation should be stage specific.

4 METHOD

4.1 OVERVIEW

Spike-by-Spike Diffusion makes the complete sampling trajectory the unit of SNN temporal refinement. Given a few-step sampler with D denoising stages and an SNN horizon T, each global SNN step advances all stages once in sampling order and produces a terminal latent increment. Accumulating these increments enables terminal image readout after every SNN step and continued refinement without restarting. To support this execution, we reformulate affine diffusion schedulers into incremental updates that propagate latent and model-output increments across stages. To improve early-readout fidelity, we further factorize the firing threshold into a stage-dependent activation range, a layer–stage base coefficient, and a stage-specific temporal coefficient, adapting the spiking representation across both denoising stage and SNN time.

4.2 TRAJECTORY-INTERLEAVED SNN EXECUTION

Let $d \in \{1, \dots, D\}$ index denoising stages with scheduler times τ_d, and $t \in \{1, \dots, T\}$ index SNN time. Stage-major execution completes all T SNN steps at each denoising stage before advancing to the next. We instead traverse all D stages in sampling order at each global SNN step, advancing each stage once while retaining a separate SNN state for every stage across traversals.

At stage d, let $\Delta z_{\tau_d}[t]$ denote the latent increment and $\Delta f_d[t]$ the model-output increment supplied to the scheduler, including any required parameterization conversion. The SNN-compatible scheduler in Section 4.3 maps these increments to $\Delta z_{\tau_{d+1}}[t]$ for $d < D$, or to the terminal increment $\Delta z_0[t]$ at $d = D$. Thus, each global SNN step propagates the current computation through the complete sampling trajectory.

We accumulate the terminal increments and decode the resulting prefix on demand:

$$\hat{z}_0^{[t]} = \hat{z}_0^{[t-1]} + \Delta z_0[t], \qquad \hat{z}_0^{[0]} = 0, \tag{1}$$

$$\hat{x}^{[t]} = \mathcal{D}_{\mathrm{VAE}}\left(\hat{z}_0^{[t]}\right), \qquad t = 1, \dots, T. \tag{2}$$

Inference can stop after any traversal or continue from the retained states without restarting. Algorithm 1 in Appendix B.5 gives the complete inference procedure.

4.3 SNN-COMPATIBLE DIFFUSION SCHEDULER

Trajectory-interleaved execution requires scheduler updates on increments rather than completed predictions. We consider ANN schedulers whose intermediate transitions ($d < D$) and terminal readout admit

$$z_{\tau_{d+1}}^{\mathrm{ANN}} = A_d z_{\tau_d}^{\mathrm{ANN}} + B_d f_d^{\mathrm{ANN}} + C_d \xi_d, \tag{3}$$

$$z_0^{\mathrm{ANN}} = A_D^{\mathrm{out}} z_{\tau_D}^{\mathrm{ANN}} + B_D^{\mathrm{out}} f_D^{\mathrm{ANN}}, \tag{4}$$

where f_d^{ANN} includes any required parameterization conversion. The coefficients are determined by the prescribed sampling schedule and remain fixed over SNN time. For stochastic transitions, $\xi_d \sim \mathcal{N}(0, I)$ denotes the transition noise.

Incremental updates. Using the same coefficients, we propagate the SNN-time increments as

$$\Delta z_{\tau_{d+1}}[t] = A_d \Delta z_{\tau_d}[t] + B_d \Delta f_d[t] + \mathbf{1}_{t=1} C_d \xi_d, \tag{5}$$

$$\Delta z_0[t] = A_D^{\mathrm{out}} \Delta z_{\tau_D}[t] + B_D^{\mathrm{out}} \Delta f_D[t]. \tag{6}$$

Each transition noise is sampled once per generation and injected only at $t = 1$. The initial latent is likewise supplied as $\Delta z_{\tau_1}[t] = \mathbf{1}_{t=1} z_{\mathrm{init}}$, so later SNN steps propagate only latent and model-output increments.

Accumulation consistency. Let $\hat{z}_{\tau_d}^{[t]} = \sum_{s=1}^{t} \Delta z_{\tau_d}[s]$ and $\hat{f}_d^{[t]} = \sum_{s=1}^{t} \Delta f_d[s]$. Accumulating the incremental scheduler through any prefix t gives

$$\begin{aligned} \hat{z}_{\tau_{d+1}}^{[t]} &= A_d \hat{z}_{\tau_d}^{[t]} + B_d \hat{f}_d^{[t]} + C_d \xi_d, \\ \hat{z}_0^{[t]} &= A_D^{\mathrm{out}} \hat{z}_{\tau_D}^{[t]} + B_D^{\mathrm{out}} \hat{f}_D^{[t]}. \end{aligned} \tag{7}$$

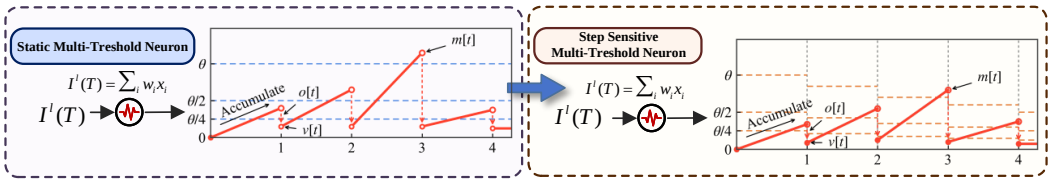


Figure 4: **Step-sensitive multi-threshold neuron.** Stage-aware base thresholds are modulated by $\alpha_d(t)$ over SNN time, while subtractive reset retains residual membrane information for subsequent correction.

Hence, temporal accumulation of the incremental scheduler recovers the original scheduler applied to the accumulated prefix. We instantiate this construction with the LCM scheduler used by LCM-LoRA (Luo et al., 2023a;b) and the TrigFlow-based sCM scheduler used by SANA-Sprint (Chen et al., 2025; Lu & Song, 2025). Appendix B.1 provides the equivalence proofs and scheduler-specific derivations.

4.4 Step-Sensitive Multi-Threshold Neuron

To improve early-readout fidelity, we introduce a multi-threshold neuron whose firing threshold adapts across both denoising stage and SNN time. For channel c of layer ℓ at stage d, we factorize the threshold as

$$\boxed{\theta_{\ell,d,c}[t] = \alpha_d(t)\, k^{\text{base}}_{\ell,d}\, x_{\max,\ell,d,c}} \tag{8}$$

where $x_{\max,\ell,d,c}$ is the channel-wise activation range at stage d, $k^{\text{base}}_{\ell,d}$ calibrates the base firing threshold relative to this range, and $\alpha_d(t)$ modulates the threshold over SNN time. Defining

$$\theta^{\text{base}}_{\ell,d,c} = k^{\text{base}}_{\ell,d} x_{\max,\ell,d,c}, \tag{9}$$

Eq. 8 becomes

$$\theta_{\ell,d,c}[t] = \alpha_d(t)\theta^{\text{base}}_{\ell,d,c}. \tag{10}$$

Suppressing indices ℓ, d, c for clarity, the neuron integrates the current input increment with the retained membrane residual:

$$m[t] = v[t-1] + x[t]. \tag{11}$$

With n magnitude levels per sign, the signed event levels and minimum magnitude are

$$\Lambda[t] = \left\{ \pm \frac{\theta[t]}{2^i} \;\middle|\; i = 0, \ldots, n-1 \right\}, \qquad \lambda^{\min}[t] = \frac{\theta[t]}{2^{n-1}}. \tag{12}$$

The emitted event is

$$o[t] = \begin{cases} 0, & |m[t]| < \lambda^{\min}[t], \\ \arg\min\limits_{\lambda \in \Lambda[t]} |m[t] - \lambda|, & \text{otherwise}, \end{cases} \tag{13}$$

followed by subtractive reset, $v[t] = m[t] - o[t]$. The residual $v[t]$ carries unexpressed information into subsequent SNN steps. Larger $\alpha_d(t)$ scales the available event magnitudes upward, whereas smaller values provide finer resolution for residual correction. Appendix B.2 gives the complete indexed dynamics and accumulation identity.

4.5 Layer–Stage Base Calibration

We first determine the stage-dependent base threshold $\theta^{\text{base}}_{\ell,d,c} = k^{\text{base}}_{\ell,d} x_{\max,\ell,d,c}$ in Eq. 8. For each layer–stage pair, $x_{\max,\ell,d,c}$ retains the channel-wise ANN activation range, while $k^{\text{base}}_{\ell,d}$ is shared across its channels and calibrates the threshold relative to that range.

For each candidate $k \in \mathcal{K}_{\text{base}}$, we run a local T-step multi-threshold simulation with thresholds $k x_{\max,\ell,d,c}$. Each ANN calibration activation is injected at $t = 1$ with zero input thereafter. We select

$$k^{\text{base}}_{\ell,d} = \arg\min_{k \in \mathcal{K}_{\text{base}}} \mathcal{L}^{\text{base}}_{\ell,d}(k), \tag{14}$$

where $\mathcal{L}_{\ell,d}^{\text{base}}(k)$ is the normalized reconstruction error between the accumulated neuron output at T and the corresponding ANN activation, measured after the subsequent weighted operation when applicable. Calibration is performed independently for each layer–stage pair. Appendix B.3 provides the activation statistics, local simulation, and full loss definition.

4.6 Temporal Sensitivity Calibration

With the calibrated base thresholds fixed, we optimize the remaining temporal factor $\alpha_d(t)$ in Eq. 8. For each denoising stage, we calibrate $\boldsymbol{\alpha}_d = [\alpha_d(1), \ldots, \alpha_d(T)]$ from the input increments encountered during trajectory-interleaved inference.

For each candidate schedule, we replay the recorded increments through the local multi-threshold neuron. Let $\mathcal{L}_{\ell,d}^{[t]}(\boldsymbol{\alpha}_d)$ denote the normalized reconstruction error between the accumulated neuron output and the corresponding activation prefix at SNN step t. We jointly optimize all T temporal coefficients using

$$\mathcal{J}_d(\boldsymbol{\alpha}_d) = \sum_{t=1}^{T} \omega_t \sum_{\ell \in \mathcal{L}_d} w_{\ell,d} \mathcal{L}_{\ell,d}^{[t]}(\boldsymbol{\alpha}_d) + \lambda_{\text{sm}} \sum_{t=1}^{T-1} \left(\log \alpha_d(t+1) - \log \alpha_d(t)\right)^2,$$

where $\mathcal{L}_d$ is the set of calibrated layers, $w_{\ell,d}$ weights layer-wise reconstruction errors, ω_t controls the importance of each SNN prefix, and λ_{sm} regularizes temporal smoothness.

Because the multi-threshold firing rule involves discrete level selection, we optimize $y_d(t) = \log \alpha_d(t)$ using covariance matrix adaptation evolution strategy (CMA-ES). The log parameterization ensures $\alpha_d(t) > 0$, and each denoising stage is optimized independently with T temporal variables. All pretrained network weights and calibrated base thresholds remain fixed. Appendix B.4 provides the complete replay objective and optimization procedure.

Algorithms 1 and 2 in Appendix B.5 summarize trajectory-interleaved inference and the two-stage calibration procedure.

5 Experiments

5.1 Experimental Setup

Models and data. We evaluate Stable Diffusion v1.5 (Rombach et al., 2022) with LCM-LoRA (Luo et al., 2023b) and `LCMScheduler` ($D = 4$, 512×512) on COCO2014-10K and COCO2017-5K, and SANA-Sprint-0.6B (Chen et al., 2025) with `SCMScheduler` ($D \in \{2, 4\}$, 1024×1024) on MJHQ-30K (Li et al., 2024a). The default SNN horizon is $T = 8$, with $n \in \{8, 10\}$ threshold levels per sign in the multi-threshold neuron.

Baselines. We compare with ANN counterparts and a stage-major SNN baseline adapted from Spiking DiTs (Fang et al., 2026). Quantized baselines include MixDQ (Zhao et al., 2024) for Stable Diffusion v1.5 and SVDQuant (Li et al., 2024b) for SANA-Sprint, both at W4A8 and W8A8. Within each model configuration, methods share prompts, random draws, sampling settings, text encoder, and VAE; SNN methods use the same n.

Metrics and implementation. We report FID, sFID, and CLIPScore. COCO FID/sFID follow the ADM protocol; MJHQ-30K FID uses the SANA evaluator. We measure computation by cumulative ACs and estimate energy from MAC and AC counts (Appendix A.1). Base thresholds and temporal schedules are calibrated with pretrained weights fixed. Experiments run on a single NVIDIA H20 (96 GiB).

5.2 Main Results

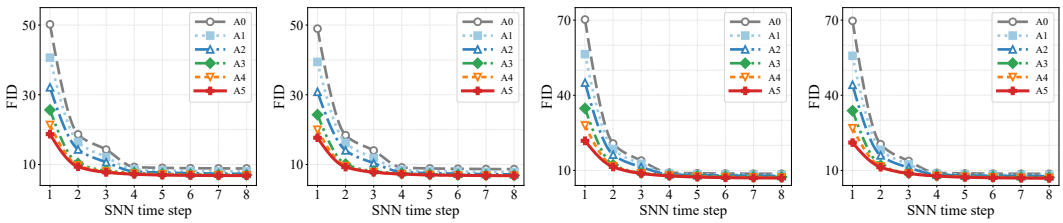
Tables 1 and 2 compare generation quality at six- and eight-step readouts, where T/t denotes the run horizon and reported readout step. SpikingDiT runs independently with $T = 6$ and $T = 8$, reporting the final output of each run. Our method instead performs a single $T = 8$ run, reading out at $t = 6$ and continuing to $t = 8$ without restarting.

Table 1: **Stable Diffusion v1.5 on COCO.**

Method \ Dataset	T/t	COCO2014-10K FID↓	COCO2014-10K sFID↓	COCO2014-10K CLIP↑	COCO2017-5K FID↓	COCO2017-5K sFID↓	COCO2017-5K CLIP↑
ANN	–	27.78	61.84	25.22	33.69	96.90	25.27
MixDQ-W4A8	–	40.25	76.57	24.36	47.18	104.49	24.42
MixDQ-W8A8	–	27.71	60.85	25.77	33.76	93.72	25.88
Threshold levels $n = 8$							
SpikingDiT	6/6	34.56	69.65	24.28	41.03	99.85	24.22
Ours	8/6	**26.93**	**56.67**	**24.84**	**32.79**	**89.15**	**24.89**
SpikingDiT	8/8	34.30	69.01	24.32	40.73	99.49	24.32
Ours	8/8	**26.75**	**59.98**	**25.13**	**32.77**	**92.96**	**25.20**
Threshold levels $n = 10$							
SpikingDiT	6/6	31.53	73.46	**24.96**	37.32	106.02	**25.06**
Ours	8/6	**27.89**	**57.96**	24.87	**34.04**	**89.87**	24.88
SpikingDiT	8/8	31.51	73.38	24.99	37.29	106.24	25.08
Ours	8/8	**27.02**	**58.99**	**25.12**	**32.88**	**92.05**	**25.19**

Table 2: **SANA-Sprint on MJHQ-30K.**

Method \ Setting	T/t	$D = 2$ FID↓	$D = 2$ sFID↓	$D = 2$ CLIP↑	$D = 4$ FID↓	$D = 4$ sFID↓	$D = 4$ CLIP↑
ANN	–	6.79	16.55	28.04	6.81	16.56	28.08
SVDQuant-W4A8	–	7.14	16.93	27.99	7.20	17.13	28.09
SVDQuant-W8A8	–	6.79	16.56	28.04	6.84	16.63	28.09
Threshold levels $n = 8$							
SpikingDiT	6/6	8.00	18.24	27.57	7.92	18.09	27.56
Ours	8/6	**6.88**	**16.68**	**28.01**	**7.10**	**16.93**	**28.06**
SpikingDiT	8/8	7.95	18.19	27.65	7.87	18.07	27.47
Ours	8/8	**6.82**	**16.59**	**28.02**	**6.96**	**16.79**	**28.07**
Threshold levels $n = 10$							
SpikingDiT	6/6	7.87	17.98	27.56	7.89	17.96	27.53
Ours	8/6	**6.89**	**16.68**	**28.02**	**7.05**	**16.85**	**28.05**
SpikingDiT	8/8	7.79	17.91	27.60	7.86	17.93	27.49
Ours	8/8	**6.80**	**16.59**	**28.03**	**6.87**	**16.69**	**28.08**



(a) $D = 2,\ n = 8$ (b) $D = 2,\ n = 10$ (c) $D = 4,\ n = 8$ (d) $D = 4,\ n = 10$

Figure 5: **Calibration ablations on SANA-Sprint.** FID versus SNN readout step on MJHQ-30K ($T = 8$). Each panel compares variants A0–A5 under a fixed (D, n) configuration.

Stable Diffusion v1.5 on COCO. Our method achieves lower FID and sFID than SpikingDiT across all reported settings. At $n = 8$ and $t = 8$, it reduces FID by 22.01% on COCO2014-10K and 19.54% on COCO2017-5K. For both threshold settings, final readouts also obtain lower measured FID and sFID than the ANN reference and MixDQ-W8A8 , with CLIP scores within 0.10 of the ANN references.

SANA-Sprint on MJHQ-30K. Our method improves FID, sFID, and CLIPScore over SpikingDiT across all reported settings. At $n = 10$ and $t = 8$, FID reaches 6.80 and 6.87 for $D = 2$ and $D = 4$, respectively, closely approaching the ANN (6.79, 6.81) and SVDQuant-W8A8 (6.79, 6.84) results. Across both models, our six-step readouts already achieve lower FID and sFID than SpikingDiT's complete $T = 8$ runs at matched n, while supporting continued refinement within the same run.

5.3 Ablation Studies

We evaluate the six variants in Table 3 on SANA-Sprint with $D \in \{2, 4\}$ and $n \in \{8, 10\}$. All use trajectory-interleaved execution with $T = 8$; Figure 5 reports FID across readout steps.

Stage-aware ranges and base calibration. Stage-specific activation ranges (A0→A1), calibrated layer-wise coefficients (A1→A2), and stage-specific coefficients (A2→A3) progressively improve early-readout FID across all four configurations. For $D = 4$ and $n = 8$, first-readout FID decreases from 70.24 (A0) to 56.42 (A1), 45.06 (A2), and 34.77 (A3).

Table 3: **Calibration variants.**

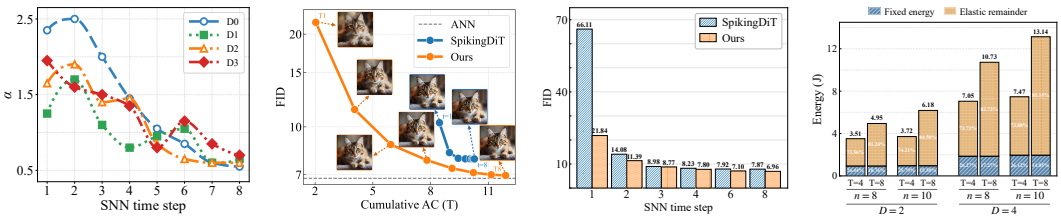
Variant	Activation range	Base coefficient	Temporal schedule
A0	$x^{\text{shared}}_{\max,\ell,c}$	0.5	1
A1	$x_{\max,\ell,d,c}$	0.5	1
A2	$x_{\max,\ell,d,c}$	k^{base}_{ℓ}	1
A3	$x_{\max,\ell,d,c}$	$k^{\text{base}}_{\ell,d}$	1
A4	$x_{\max,\ell,d,c}$	$k^{\text{base}}_{\ell,d}$	$\alpha(t)$
A5 (ours)	$x_{\max,\ell,d,c}$	$k^{\text{base}}_{\ell,d}$	$\alpha_d(t)$

Temporal sensitivity. For $D = 4$ and $n = 8$, a shared temporal schedule reduces first-readout FID from 34.77 (A3) to 27.95 (A4), while stage-specific schedules further reduce it to 21.84 (A5). The corresponding final FIDs are 7.37, 7.12, and 6.96. A5 achieves the lowest first-readout FID across all four configurations.

Calibrated temporal schedules. Figure 6(a) shows the calibrated $\alpha_d(t)$ for SANA-Sprint with $D = 4$, $n = 8$, and $T = 8$. The coefficients show an overall decreasing tendency over SNN time, while different denoising stages exhibit distinct local variations (Additional results in Appendix C.2).

5.4 Progressive Quality under Increasing Computation

Figure 6(b) shows FID versus cumulative ACs for single-run progressive generation with SANA-Sprint ($D = 4$, $T = 8$, $n = 8$). Our first terminal readout appears at approximately 2.0×10^{12} ACs with an FID of 21.84. By $t = 4$, FID drops to 7.80 at approximately 7.8×10^{12} ACs, already



(a) Temporal schedules (b) Quality–computation (c) Prescribed budgets (d) First-readout energy

Figure 6: **Temporal calibration and progressive inference. (a)** Calibrated stage-specific coefficients for SANA-Sprint ($D = 4$, $n = 8$, $T = 8$). **(b)** FID versus cumulative ACs within a single run. **(c)** FID at prescribed SNN budgets. **(d)** Energy consumed through the first terminal readout and remaining for refinement.

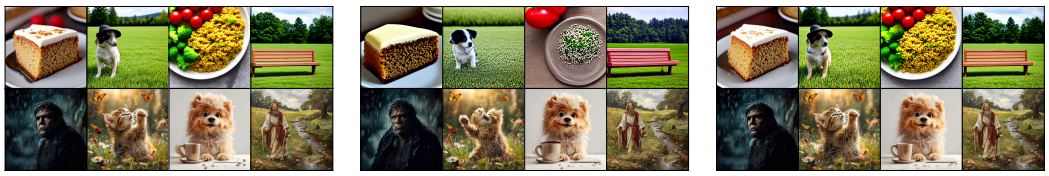
below SpikingDiT's final FID of 7.87 and before its first terminal readout becomes available at approximately 8.5×10^{12} ACs. Continuing the same run further reduces FID to 7.10 at $t = 6$ and 6.96 at $t = 8$, approaching the ANN reference of 6.81.

5.5 Generation Quality at Prescribed SNN Budgets

We compare generation quality at budgets $B \in \{1, 2, 3, 4, 6, 8\}$. SpikingDiT runs independently with $T = B$ at each budget, whereas our outputs are read at $t = B$ from a single $T = 8$ run. Figure 6(c) shows SANA-Sprint with $D = 4$ and $n = 8$. At $B = 1$, our method achieves an FID of 21.84, compared with 66.11 for SpikingDiT. By $B = 4$, our prefix reaches 7.80, already below SpikingDiT's complete $T = 8$ result of 7.87. Across $D \in \{2, 4\}$ and $n \in \{8, 10\}$, our method achieves lower FID in all 24 evaluated budget–configuration pairs. Complete results are provided in Appendix C.3.

5.6 Energy Budget for Progressive Refinement

Figure 6(d) reports the energy consumed through the first terminal readout under $D \in \{2, 4\}$, $T \in \{4, 8\}$, and $n \in \{8, 10\}$. Across all configurations, the first readout uses only 14.85%–26.44% of the full-run energy. For $T = 8$, 81.24%–85.15% of the budget remains available for subsequent refinement. This large post-readout budget is a direct consequence of trajectory-level execution: a complete terminal output is exposed after the first traversal, while the remaining traversals continue refining the same generation.



(a) ANN (b) SpikingDiT (c) Ours

Figure 7: **Qualitative comparison.**

5.7 Qualitative Results

Figure 7 compares ANN references with SNN outputs at $t = T = 8$. Across both models, our results more closely match the ANN references in object attributes, scene content, and composition than SpikingDiT. Appendix C.4 further visualizes single-run progressive refinement and additional final generations.

6 Conclusion

We introduced *Spike-by-Spike Diffusion*, a trajectory-level ANN-to-SNN conversion framework that enables terminal readout and progressive refinement after every SNN step. It combines an incremental diffusion scheduler with stage-aware and temporally adaptive threshold calibration. Experiments on Stable Diffusion v1.5 and SANA-Sprint show earlier high-quality outputs and better generation under flexible computation budgets.

AI USE STATEMENT

Generative AI tools were used to assist with manuscript organization, language editing, and discussion of research positioning. They were not used as a source of experimental evidence. The authors review and verify all AI-assisted text, mathematical claims, citations, code, and experimental results and take responsibility for the final content of the work.

REFERENCES

Tong Bu, Jianhao Ding, Zhaofei Yu, and Tiejun Huang. Optimized potential initialization for low-latency spiking neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pp. 11–20, 2022.

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. *arXiv preprint arXiv:2303.04347*, 2023.

Jiahang Cao, Hanzhong Guo, Ziqing Wang, Deming Zhou, Hao Cheng, Qiang Zhang, and Renjing Xu. Spiking diffusion models. *IEEE Transactions on Artificial Intelligence*, 6(1):132–143, 2024.

Junsong Chen, Shuchen Xue, Yuyang Zhao, Jincheng Yu, Sayak Paul, Junyu Chen, Han Cai, Song Han, and Enze Xie. Sana-sprint: One-step diffusion with continuous-time consistency distillation. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 16185–16195. IEEE, 2025.

Shikuang Deng and Shi Gu. Optimal conversion of conventional artificial neural networks to spiking neural networks. *arXiv preprint arXiv:2103.00476*, 2021.

Peter U Diehl, Daniel Neil, Jonathan Binas, Matthew Cook, Shih-Chii Liu, and Michael Pfeiffer. Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In *2015 International joint conference on neural networks (IJCNN)*, pp. 1–8. ieee, 2015.

Haoran Fang, Tianxing Man, Xingchen Li, Wanli Shi, Jinjie Fang, and Bin Gu. Trajectory-aware spiking dits conversion via membrane potential error-feedback. In *Forty-third International Conference on Machine Learning*, 2026.

Zecheng Hao, Tong Bu, Jianhao Ding, Tiejun Huang, and Zhaofei Yu. Reducing ann-snn conversion error through residual membrane potential. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pp. 11–21, 2023.

Haiyan Jiang, Srinivas Anumasa, Giulia De Masi, Huan Xiong, and Bin Gu. A unified optimization framework of ann-snn conversion: towards optimal mapping from activation values to firing rates. In *International Conference on Machine Learning*, pp. 14945–14974. PMLR, 2023.

Hiromichi Kamata, Yusuke Mukuta, and Tatsuya Harada. Fully spiking variational autoencoder. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pp. 7059–7067, 2022.

Daiqing Li, Aleks Kamko, Ehsan Akhgari, Ali Sabet, Linmiao Xu, and Suhail Doshi. Playground v2. 5: Three insights towards enhancing aesthetic quality in text-to-image generation. *arXiv preprint arXiv:2402.17245*, 2024a.

Muyang Li, Yujun Lin, Zhekai Zhang, Tianle Cai, Xiuyu Li, Junxian Guo, Enze Xie, Chenlin Meng, Jun-Yan Zhu, and Song Han. Svdquant: Absorbing outliers by low-rank components for 4-bit diffusion models. *arXiv preprint arXiv:2411.05007*, 2024b.

Yuhang Li, Shikuang Deng, Xin Dong, Ruihao Gong, and Shi Gu. A free lunch from ann: Towards efficient, accurate spiking neural networks calibration. In *International conference on machine learning*, pp. 6316–6325. PMLR, 2021.

Fangxin Liu, Wenbo Zhao, Yongbiao Chen, Zongwu Wang, and Li Jiang. Spikeconverter: An efficient conversion framework zipping the gap between artificial neural networks and spiking neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pp. 1692–1701, 2022a.

Luping Liu, Yi Ren, Zhijie Lin, and Zhou Zhao. Pseudo numerical methods for diffusion models on manifolds. *arXiv preprint arXiv:2202.09778*, 2022b.

Mingxuan Liu, Jie Gan, Rui Wen, Tao Li, Yongli Chen, and Hong Chen. Spiking-diffusion: Vector quantized discrete diffusion model with spiking neural networks. In *2024 5th International Conference on Computer, Big Data and Artificial Intelligence (ICCBD+ AI)*, pp. 627–631. IEEE, 2024.

Cheng Lu and Yang Song. Simplifying, stabilizing and scaling continuous-time consistency models. In *International Conference on Learning Representations*, volume 2025, pp. 50611–50649, 2025.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *Advances in neural information processing systems*, 35:5775–5787, 2022.

Simian Luo, Yiqin Tan, Longbo Huang, Jian Li, and Hang Zhao. Latent consistency models: Synthesizing high-resolution images with few-step inference. *arXiv preprint arXiv:2310.04378*, 2023a.

Simian Luo, Yiqin Tan, Suraj Patil, Daniel Gu, Patrick von Platen, Apolinário Passos, Longbo Huang, Jian Li, and Hang Zhao. LCM-LoRA: A universal stable-diffusion acceleration module. *arXiv preprint arXiv:2311.05556*, 2023b.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

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*, 11:682, 2017.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in neural information processing systems*, 35:36479–36494, 2022.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022.

Abhronil Sengupta, Yuting Ye, Robert Wang, Chiao Liu, and Kaushik Roy. Going deeper in spiking neural networks: Vgg and residual architectures. *Frontiers in neuroscience*, 13:95, 2019.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. 2023.

Man Yao, Jiakui Hu, Zhaokun Zhou, Li Yuan, Yonghong Tian, Bo Xu, and Guoqi Li. Spike-driven transformer. *Advances in neural information processing systems*, 36:64043–64058, 2023.

Tianchen Zhao, Xuefei Ning, Tongcheng Fang, Enshu Liu, Guyue Huang, Zinan Lin, Shengen Yan, Guohao Dai, and Yu Wang. Mixdq: Memory-efficient few-step text-to-image diffusion models with metric-decoupled mixed precision quantization. In *European Conference on Computer Vision*, pp. 285–302. Springer, 2024.

APPENDIX

A ADDITIONAL OBSERVATION DETAILS

A.1 PROFILING THE FIRST TERMINAL READOUT

This section details the profiling protocol and energy accounting underlying Figure 2(a).

Experimental configurations. We profile the stage-major baseline adapted from SpikingDiT (Fang et al., 2026) over all six combinations of $D \in \{25, 50, 100\}$ denoising stages and SNN horizons $T \in \{4, 8\}$, with $n = 8$ magnitude levels per sign. The denoising schedule and SNN horizon are fixed within each run.

Terminal-prefix readout. The first $D-1$ stages complete all T SNN steps before the final stage begins. Let z_D^{in} denote the final-stage input latent and $\widehat{p}_D^{[t]}$ the accumulated denoiser prediction after t SNN steps. We define the terminal prefix

$$\widehat{z}_0^{[t]} = \mathcal{R}_D\left(z_D^{\text{in}}, \widehat{p}_D^{[t]}; h_D, \xi_D\right), \qquad t = 1, \ldots, T, \tag{15}$$

where $\mathcal{R}_D$ is the terminal scheduler update, h_D denotes its incoming scheduler state, and ξ_D denotes any required transition randomness. All prefixes share the same z_D^{in}, h_D, and ξ_D. Each readout is evaluated independently and does not alter the ongoing SNN accumulation. The baseline's output normalization is kept fixed to the prescribed horizon T, so $\widehat{z}_0^{[t]}$ uses only the computation available through step t.

Energy accounting. We estimate energy from operation counts using $4.6\,\text{pJ}$ per multiply–accumulate (MAC) and $0.9\,\text{pJ}$ per accumulate (AC) operation (Yao et al., 2023). The cumulative energy through final-stage prefix t is

$$E(t) = [4.6 N_{\text{MAC}}(t) + 0.9 N_{\text{AC}}(t)] \times 10^{-12}\,\text{J}, \tag{16}$$

where $N_{\text{MAC}}(t)$ and $N_{\text{AC}}(t)$ include all T SNN steps in the first $D-1$ stages and the first t steps of the final stage. We count operations from patch embedding, dense and spiking linear layers, attention, and the associated temporal, residual, membrane-update, scaling, normalization, and activation computations.

We define

$$E_{\text{first}} = E(1), \qquad E_{\text{full}} = E(T), \qquad E_{\text{rem}} = E(T) - E(1), \tag{17}$$

and report

$$\rho_{\text{first}} = \frac{E_{\text{first}}}{E_{\text{full}}}, \qquad \rho_{\text{rem}} = \frac{E_{\text{rem}}}{E_{\text{full}}} = 1 - \rho_{\text{first}}. \tag{18}$$

These fractions correspond to the blue and orange segments in Figure 2(a), respectively. Across the six configurations, ρ_{first} ranges from 96.57% to 99.26%, leaving only 0.74%–3.43% of the full-run energy after the first terminal readout.

A.2 STAGE STATISTICS AND TEMPORAL SPECTRAL ANALYSIS

This section provides additional analysis underlying Observation 2 in Section 3.2, including stage-dependent activation statistics, the relation between activation ranges and firing thresholds, and the temporal spectral analysis used in Figure 3.

Stage-dependent activation statistics. Although the denoising stages share the same network weights, their latent inputs and noise-level conditioning vary along the sampling trajectory. For Stable Diffusion v1.5 with four-stage LCM-LoRA sampling, the ANN denoiser-output distribution progressively narrows across stages. From Stage 1 to Stage 4, the standard deviation decreases from 0.983 to 0.788, while the 99th percentile of the absolute magnitude decreases from 2.535 to 2.088 (Figure 2(b)).

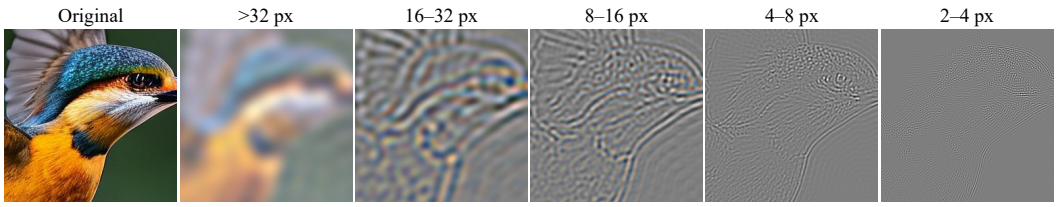


Figure 8: **Spatial content of the frequency bands.** Original bird crop and bandwise reconstructions for wavelengths > 32, 16–32, 8–16, 4–8, and 2–4 pixels, shown over the same spatial region. Coarse bands capture broad color and intensity variations, while finer bands emphasize contours and texture.

We further compare converted trajectories using shared and stage-specific activation scales against the ANN trajectory. As shown in Figure 2(c), stage-specific scales yield lower latent MSE and higher cosine similarity at every denoising stage. These results indicate that the numerical reference used by the spiking representation should reflect the activation range encountered at each stage.

Activation ranges and firing thresholds. Activation calibration and threshold selection play distinct roles in the multi-threshold representation. We parameterize the firing threshold as

$$\theta = k x_{\max}, \tag{19}$$

where $x_{\max}$ denotes the calibrated activation range and k determines the threshold magnitude relative to this range. With n magnitude levels per sign, the largest available event magnitude is θ, while the smallest is

$$\frac{\theta}{2^{n-1}}. \tag{20}$$

Thus, $x_{\max}$ defines the numerical reference of the representation, whereas k controls the range and resolution of the available event magnitudes relative to that reference.

Fixed-threshold temporal analysis. To isolate the effect of firing thresholds, we keep the calibrated activation ranges fixed and compare $k \in \{0.5, 1.0, 2.0\}$ with $T = 16$. The threshold-level count, prompt, sampling configuration, and random draws are shared across the three runs, and each coefficient remains fixed throughout its run.

At every SNN step, we decode the accumulated terminal latent and analyze the resulting RGB image in separate spatial-frequency bands. For each band, we measure spectral cosine similarity to the final $T = 16$ image from the same run. Each k setting therefore uses its own endpoint as the reference, so the analysis measures how different spatial components develop toward the final representation under that threshold setting.

Frequency decomposition. Let $I_t^{(k)} \in \mathbb{R}^{3\times512\times512}$ denote the decoded RGB readout at SNN step t under threshold coefficient k. We compute the two-dimensional Fourier transform independently for each channel:

$$F_t^{(k)}(c, f_x, f_y) = \mathrm{FFT2}\Big(I_t^{(k)}(c, \cdot, \cdot)\Big). \tag{21}$$

For frequencies f_x, f_y in cycles per pixel, the radial frequency is

$$r = \sqrt{f_x^2 + f_y^2}, \tag{22}$$

with corresponding wavelength $\lambda = 1/r$ for $r > 0$. We partition the spectrum into five radial bands:

Band	Wavelength (px)	Radial frequency r
Coarsest	> 32	$0 < r < 1/32$
Coarse	16–32	$1/32 \le r < 1/16$
Medium	8–16	$1/16 \le r < 1/8$
Fine	4–8	$1/8 \le r < 1/4$
Finest	2–4	$1/4 \le r \le 1/2$

All runs use the same frequency masks. We exclude the DC component ($r = 0$), which represents the channel means, and Fourier modes with $r > 1/2$.

Bandwise spectral alignment. Let Ω_b denote the set of frequency coordinates in band b. We concatenate the complex coefficients $F_t^{(k)}(c, \boldsymbol{f})$ across all three RGB channels and all $\boldsymbol{f} \in \Omega_b$ into $\mathbf{v}_{t,b}^{(k)}$. The spectral alignment at SNN step t is

$$C_b^{(k)}(t) = \frac{\mathrm{Re}\left[(\mathbf{v}_{T,b}^{(k)})^{\mathrm{H}} \mathbf{v}_{t,b}^{(k)}\right]}{\|\mathbf{v}_{T,b}^{(k)}\|_2 \|\mathbf{v}_{t,b}^{(k)}\|_2}, \tag{23}$$

where H denotes the conjugate transpose and $\mathbf{v}_{T,b}^{(k)}$ is obtained from the final readout of the same run. This joint RGB cosine incorporates both relative spectral amplitudes and phases within each band. We apply bandwise L_2 normalization and no additional min–max normalization.

Bandwise image reconstruction. To visualize the spatial content represented by each frequency band, we reconstruct an RGB image I using

$$I_b(c) = \mathrm{Re}\left\{\mathrm{IFFT2}\left[M_b \odot \mathrm{FFT2}(I(c))\right]\right\}, \tag{24}$$

where M_b is the binary mask corresponding to Ω_b and $\odot$ denotes elementwise multiplication. The filtering operation retains the complex Fourier coefficients within the selected band and therefore preserves their phases. Because the radial masks preserve conjugate symmetry, the inverse transform is real-valued up to numerical precision. The reconstructed bands have zero spatial mean before display adjustment.

Figure 8 illustrates the spatial interpretation of these bands. The coarsest components capture broad variations in color and intensity across the bird and background. Intermediate bands emphasize transitions around the head, eye, and beak, while the finest bands capture contours and feather texture.

Temporal alignment under different thresholds. Figure 3(a–c) shows the bandwise alignment trajectories for the three fixed threshold coefficients. Across the settings, coarse spatial components approach their final representations earlier, while finer components continue evolving over later SNN steps. This produces a coarse-to-fine temporal progression during SNN accumulation.

The threshold coefficient changes the timing of this progression. As summarized in Figure 3(d), larger k produces stronger coarse-band alignment during the first few SNN steps, but does not uniformly accelerate the finest band. In this example, $k = 1.0$ yields higher finest-band similarity during the middle of the run, whereas $k = 0.5$ becomes higher at later steps. The relative preference among threshold settings therefore changes with both spatial frequency and SNN time.

These observations motivate separating stage-dependent numerical calibration from temporal threshold modulation: activation statistics determine the numerical reference at each denoising stage, while the firing threshold can vary over SNN time to accommodate the changing requirements of early structure formation and later detail refinement.

B Additional Method Details

B.1 Scheduler Derivations and Equivalence

This section proves the accumulation consistency of the SNN-compatible scheduler in Section 4.3, establishes trajectory equivalence under denoiser-output correspondence, and derives the LCM and sCM scheduler instances.

B.1.1 Accumulation Consistency

The scheduler coefficients depend only on the prescribed sampling schedule and configuration and remain fixed over SNN time. Each transition noise ξ_d is sampled once per generation and injected only at $t = 1$. Define the accumulated latent and model-output prefixes as

$$\begin{aligned} \hat{z}_{\tau_d}^{[t]} &= \sum_{s=1}^{t} \Delta z_{\tau_d}[s], \\ \hat{f}_d^{[t]} &= \sum_{s=1}^{t} \Delta f_d[s], \end{aligned} \tag{25}$$

and let $\hat{z}_0^{[t]} = \sum_{s=1}^{t} \Delta z_0[s]$ denote the accumulated terminal latent.

Prefix-wise consistency. For any $t \in \{1, \dots, T\}$ and $d < D$, summing Eq. 5 gives

$$\begin{aligned}\hat{z}_{\tau_{d+1}}^{[t]} &= \sum_{s=1}^{t} \Delta z_{\tau_{d+1}}[s] \\ &= A_d \sum_{s=1}^{t} \Delta z_{\tau_d}[s] + B_d \sum_{s=1}^{t} \Delta f_d[s] + C_d \xi_d \sum_{s=1}^{t} \mathbf{1}_{s=1} \\ &= A_d \hat{z}_{\tau_d}^{[t]} + B_d \hat{f}_d^{[t]} + C_d \xi_d.\end{aligned}$$

The stochastic term appears exactly once because $\sum_{s=1}^{t} \mathbf{1}_{s=1} = 1$ for every $t \geq 1$. At the terminal stage, summing Eq. 6 gives

$$\begin{aligned}\hat{z}_0^{[t]} &= A_D^{\text{out}} \sum_{s=1}^{t} \Delta z_{\tau_D}[s] + B_D^{\text{out}} \sum_{s=1}^{t} \Delta f_D[s] \\ &= A_D^{\text{out}} \hat{z}_{\tau_D}^{[t]} + B_D^{\text{out}} \hat{f}_D^{[t]}.\end{aligned}$$

These are exactly the original scheduler transition and terminal readout evaluated on the accumulated inputs, establishing Eq. 7 for every SNN prefix.

Same-input scheduler equivalence. At the full horizon, suppose the accumulated inputs at stage d satisfy

$$\begin{aligned}\hat{z}_{\tau_d}^{[T]} &= z_{\tau_d}^{\text{ANN}}, \\ \hat{f}_d^{[T]} &= f_d^{\text{ANN}}.\end{aligned} \tag{26}$$

With identical scheduler coefficients and transition-noise realizations, the accumulated updates recover the ANN scheduler outputs:

$$\begin{aligned}\hat{z}_{\tau_{d+1}}^{[T]} &= z_{\tau_{d+1}}^{\text{ANN}}, \qquad d < D, \\ \hat{z}_0^{[T]} &= z_0^{\text{ANN}}, \qquad d = D.\end{aligned} \tag{27}$$

Thus, the incremental reformulation itself introduces no additional scheduler approximation.

B.1.2 Conditional Trajectory Equivalence

We next extend the stagewise scheduler identity to the complete sampling trajectory. Assume that the ANN and SNN computations share the same initial latent, scheduler coefficients, and transition-noise realizations, and that the converted denoiser satisfies

$$\hat{z}_{\tau_d}^{[T]} = z_{\tau_d}^{\text{ANN}} \quad \Longrightarrow \quad \hat{f}_d^{[T]} = f_d^{\text{ANN}}, \qquad d = 1, \dots, D. \tag{28}$$

Under this correspondence, equality of the accumulated and ANN latents follows by induction over denoising stages.

Initialization. The shared initial latent is injected only at the first SNN step:

$$\Delta z_{\tau_1}[t] = \mathbf{1}_{t=1} z_{\text{init}}. \tag{29}$$

Therefore,

$$\begin{aligned}\hat{z}_{\tau_1}^{[T]} &= \sum_{t=1}^{T} \Delta z_{\tau_1}[t] \\ &= z_{\text{init}} = z_{\tau_1}^{\text{ANN}}.\end{aligned} \tag{30}$$

Inductive step. Suppose $\hat{z}_{\tau_d}^{[T]} = z_{\tau_d}^{\mathrm{ANN}}$ for some $d < D$. Equation 28 then gives $\hat{f}_d^{[T]} = f_d^{\mathrm{ANN}}$. Applying the accumulation identity yields

$$\begin{aligned}\hat{z}_{\tau_{d+1}}^{[T]} &= A_d \hat{z}_{\tau_d}^{[T]} + B_d \hat{f}_d^{[T]} + C_d \xi_d \\ &= A_d z_{\tau_d}^{\mathrm{ANN}} + B_d f_d^{\mathrm{ANN}} + C_d \xi_d \\ &= z_{\tau_{d+1}}^{\mathrm{ANN}}.\end{aligned} \tag{31}$$

Hence, the correspondence propagates to the next denoising stage.

Terminal readout. By induction, $\hat{z}_{\tau_D}^{[T]} = z_{\tau_D}^{\mathrm{ANN}}$. Applying Eq. 28 at the final stage and then the terminal scheduler identity gives

$$\begin{aligned}\hat{z}_0^{[T]} &= A_D^{\mathrm{out}} \hat{z}_{\tau_D}^{[T]} + B_D^{\mathrm{out}} \hat{f}_D^{[T]} \\ &= A_D^{\mathrm{out}} z_{\tau_D}^{\mathrm{ANN}} + B_D^{\mathrm{out}} f_D^{\mathrm{ANN}} \\ &= z_0^{\mathrm{ANN}}.\end{aligned} \tag{32}$$

The scheduler transformation is therefore algebraically exact under the affine form; full ANN–SNN trajectory equivalence additionally requires the denoiser-output correspondence in Eq. 28.

B.1.3 Diffusion-Parameterized Consistency Scheduler: LCM

We consider the LCM scheduler used with LCM-LoRA on Stable Diffusion v1.5 (Luo et al., 2023a;b), using noise prediction with sample clipping and dynamic thresholding disabled.

ANN transition. For $f_d^{\mathrm{ANN}} = \epsilon_d^{\mathrm{ANN}}$, the scheduler first forms

$$\tilde{z}_{0,d}^{\mathrm{ANN}} = c_{\mathrm{skip}}(\tau_d) z_{\tau_d}^{\mathrm{ANN}} + c_{\mathrm{out}}(\tau_d) \frac{z_{\tau_d}^{\mathrm{ANN}} - \sqrt{1-\bar{\alpha}_{\tau_d}}\, \epsilon_d^{\mathrm{ANN}}}{\sqrt{\bar{\alpha}_{\tau_d}}}, \tag{33}$$

and, for $d < D$, advances to the next stage as

$$z_{\tau_{d+1}}^{\mathrm{ANN}} = \sqrt{\bar{\alpha}_{\tau_{d+1}}}\, \tilde{z}_{0,d}^{\mathrm{ANN}} + \sqrt{1-\bar{\alpha}_{\tau_{d+1}}}\, \xi_d. \tag{34}$$

Here $\bar{\alpha}_{\tau_d}$ is the cumulative diffusion signal-retention factor, while c_{skip} and c_{out} are the LCM boundary-condition coefficients.

Affine coefficients and incremental update. Substituting Eq. 33 into Eq. 34 and collecting the latent and prediction terms gives

$$\begin{aligned}A_d^{\mathrm{LCM}} &= \sqrt{\bar{\alpha}_{\tau_{d+1}}} \left(c_{\mathrm{skip}}(\tau_d) + \frac{c_{\mathrm{out}}(\tau_d)}{\sqrt{\bar{\alpha}_{\tau_d}}} \right), \\ B_d^{\mathrm{LCM}} &= -\sqrt{\bar{\alpha}_{\tau_{d+1}}}\, c_{\mathrm{out}}(\tau_d) \frac{\sqrt{1-\bar{\alpha}_{\tau_d}}}{\sqrt{\bar{\alpha}_{\tau_d}}}, \\ C_d^{\mathrm{LCM}} &= \sqrt{1-\bar{\alpha}_{\tau_{d+1}}}.\end{aligned} \tag{35}$$

The corresponding incremental transition is

$$\Delta z_{\tau_{d+1}}[t] = A_d^{\mathrm{LCM}} \Delta z_{\tau_d}[t] + B_d^{\mathrm{LCM}} \Delta \epsilon_d[t] + \mathbf{1}_{t=1} C_d^{\mathrm{LCM}} \xi_d. \tag{36}$$

Terminal readout. At stage D, the scheduler returns $z_0^{\mathrm{ANN}} = \tilde{z}_{0,D}^{\mathrm{ANN}}$ without further noise injection. Hence,

$$\begin{aligned}A_D^{\mathrm{out,LCM}} &= c_{\mathrm{skip}}(\tau_D) + \frac{c_{\mathrm{out}}(\tau_D)}{\sqrt{\bar{\alpha}_{\tau_D}}}, \\ B_D^{\mathrm{out,LCM}} &= -c_{\mathrm{out}}(\tau_D) \frac{\sqrt{1-\bar{\alpha}_{\tau_D}}}{\sqrt{\bar{\alpha}_{\tau_D}}}.\end{aligned} \tag{37}$$

Substituting these coefficients together with $\Delta f_D[t] = \Delta \epsilon_D[t]$ into Eq. 6 gives the incremental terminal readout. Both the intermediate transitions and terminal readout therefore satisfy the accumulation consistency established in Appendix B.1.1.

B.1.4 FLOW-PARAMETERIZED CONSISTENCY SCHEDULER: SCM

We consider the sCM scheduler used by SANA-Sprint (Chen et al., 2025), based on the TrigFlow parameterization for continuous-time consistency distillation (Lu & Song, 2025).

Input and model-output parameterization. Let τ_d denote TrigFlow time, with $\tau_{D+1} = 0$. The corresponding flow time and input-scaling factor are

$$\begin{aligned} \eta_d &= \frac{\sin(\tau_d)}{\cos(\tau_d) + \sin(\tau_d)}, \\ h_d &= \sqrt{\eta_d^2 + (1-\eta_d)^2}. \end{aligned} \tag{38}$$

For fixed latent scale $\sigma_{\text{data}} > 0$, the ANN receives

$$u_d^{\text{ANN}} = \frac{h_d}{\sigma_{\text{data}}} z_{\tau_d}^{\text{ANN}}, \tag{39}$$

and returns the raw flow prediction $w_d^{\text{ANN}} = \mathcal{F}_\theta(u_d^{\text{ANN}}, \eta_d)$. The model output supplied to the scheduler is

$$\begin{aligned} f_d^{\text{ANN}} = v_d^{\text{ANN}} &= \frac{\sigma_{\text{data}}}{h_d} \left((1-2\eta_d) u_d^{\text{ANN}} + h_d^2 w_d^{\text{ANN}} \right) \\ &= (1-2\eta_d) z_{\tau_d}^{\text{ANN}} + \sigma_{\text{data}} h_d w_d^{\text{ANN}}. \end{aligned} \tag{40}$$

Incremental parameterization conversion. Because η_d, h_d, and σ_{data} are fixed over SNN time, the corresponding increments are

$$\begin{aligned} \Delta u_d[t] &= \frac{h_d}{\sigma_{\text{data}}} \Delta z_{\tau_d}[t], \\ \Delta v_d[t] &= (1-2\eta_d) \Delta z_{\tau_d}[t] + \sigma_{\text{data}} h_d \Delta w_d[t]. \end{aligned} \tag{41}$$

Here $\Delta w_d[t]$ is the raw output increment of the converted denoiser, while $\Delta v_d[t] = \Delta f_d[t]$ is the increment supplied to the scheduler. Summing these expressions recovers the same linear input scaling and output conversion on the accumulated quantities.

ANN transition. Given v_d^{ANN}, the sCM scheduler forms

$$\tilde{z}_{0,d}^{\text{ANN}} = \cos(\tau_d) z_{\tau_d}^{\text{ANN}} - \sin(\tau_d) v_d^{\text{ANN}}, \tag{42}$$

and, for $d < D$, updates the latent as

$$z_{\tau_{d+1}}^{\text{ANN}} = \cos(\tau_{d+1}) \tilde{z}_{0,d}^{\text{ANN}} + \sin(\tau_{d+1}) \sigma_{\text{data}} \xi_d. \tag{43}$$

Affine coefficients and incremental update. Substituting Eq. 42 into Eq. 43 gives

$$\begin{aligned} A_d^{\text{sCM}} &= \cos(\tau_{d+1}) \cos(\tau_d), \\ B_d^{\text{sCM}} &= -\cos(\tau_{d+1}) \sin(\tau_d), \\ C_d^{\text{sCM}} &= \sigma_{\text{data}} \sin(\tau_{d+1}). \end{aligned} \tag{44}$$

The incremental transition is therefore

$$\Delta z_{\tau_{d+1}}[t] = A_d^{\text{sCM}} \Delta z_{\tau_d}[t] + B_d^{\text{sCM}} \Delta v_d[t] + \mathbf{1}_{t=1} C_d^{\text{sCM}} \xi_d. \tag{45}$$

Terminal readout. At the final stage, $\tau_{D+1} = 0$ removes the stochastic term, giving

$$\Delta z_0[t] = \cos(\tau_D) \Delta z_{\tau_D}[t] - \sin(\tau_D) \Delta v_D[t]. \tag{46}$$

Thus,

$$A_D^{\text{out,sCM}} = \cos(\tau_D), \qquad B_D^{\text{out,sCM}} = -\sin(\tau_D). \tag{47}$$

Together with the linear parameterization conversion, these coefficients establish accumulation consistency for both intermediate transitions and terminal readout.

Initialization and decoding. The shared initial latent is $z_{\text{init}} = \sigma_{\text{data}}\xi_{\text{init}}$, where $\xi_{\text{init}} \sim \mathcal{N}(0, I)$. Before decoding, the terminal latent is divided by σ_{data}. The decoder $\mathcal{D}_{\text{VAE}}$ in Eq. 2 includes this normalization and the autoencoder latent scaling.

Together, the LCM and sCM derivations show that the same affine incremental construction applies to both diffusion- and flow-parameterized consistency schedulers.

B.2 Step-Sensitive Neuron Dynamics

This section gives the full dynamics of the step-sensitive multi-threshold neuron in Section 4.4 and derives its accumulation identity.

Factorized threshold. For channel c of converted layer ℓ at denoising stage d, the firing threshold follows Eq. 8:

$$\theta_{\ell,d,c}[t] = \alpha_d(t)\, k^{\text{base}}_{\ell,d}\, x_{\max,\ell,d,c} = \alpha_d(t)\theta^{\text{base}}_{\ell,d,c}, \tag{48}$$

where $\theta^{\text{base}}_{\ell,d,c} = k^{\text{base}}_{\ell,d} x_{\max,\ell,d,c}$. The activation range $x_{\max,\ell,d,c}$ and base coefficient $k^{\text{base}}_{\ell,d}$ are fixed during inference, while $\alpha_d(t)$ modulates the threshold over SNN time.

Integration, firing, and reset. Let $x_{\ell,d,c}[t]$ denote the input increment at SNN step t and $v_{\ell,d,c}[t-1]$ the retained membrane residual, with all operations applied elementwise. The pre-firing membrane potential is

$$m_{\ell,d,c}[t] = v_{\ell,d,c}[t-1] + x_{\ell,d,c}[t]. \tag{49}$$

With n magnitude levels per sign, the signed event levels and minimum magnitude are

$$\begin{aligned} \Lambda_{\ell,d,c}[t] &= \left\{ \pm \frac{\theta_{\ell,d,c}[t]}{2^i} \;\middle|\; i = 0, \ldots, n-1 \right\}, \\ \lambda^{\min}_{\ell,d,c}[t] &= \frac{\theta_{\ell,d,c}[t]}{2^{n-1}}. \end{aligned} \tag{50}$$

The neuron remains silent when the membrane magnitude is below the minimum level; otherwise, it emits the closest signed level:

$$o_{\ell,d,c}[t] = \begin{cases} 0, & |m_{\ell,d,c}[t]| < \lambda^{\min}_{\ell,d,c}[t], \\ \arg\min\limits_{\lambda \in \Lambda_{\ell,d,c}[t]} |m_{\ell,d,c}[t] - \lambda|, & \text{otherwise.} \end{cases} \tag{51}$$

Subtractive reset updates the membrane state as

$$v_{\ell,d,c}[t] = m_{\ell,d,c}[t] - o_{\ell,d,c}[t]. \tag{52}$$

The residual is retained at the same denoising stage and carried into the next SNN step.

Accumulation identity. Combining Eqs. 49 and 52 gives

$$o_{\ell,d,c}[t] = x_{\ell,d,c}[t] + v_{\ell,d,c}[t-1] - v_{\ell,d,c}[t]. \tag{53}$$

Summing over the SNN horizon telescopes the membrane terms:

$$\begin{aligned} \sum_{t=1}^{T} o_{\ell,d,c}[t] &= \sum_{t=1}^{T} x_{\ell,d,c}[t] + \sum_{t=1}^{T} \left(v_{\ell,d,c}[t-1] - v_{\ell,d,c}[t] \right) \\ &= \sum_{t=1}^{T} x_{\ell,d,c}[t] + v_{\ell,d,c}[0] - v_{\ell,d,c}[T]. \end{aligned} \tag{54}$$

The same identity holds for every prefix. With zero initial membrane state,

$$\sum_{s=1}^{t} \left(x_{\ell,d,c}[s] - o_{\ell,d,c}[s] \right) = v_{\ell,d,c}[t], \qquad t = 1, \ldots, T. \tag{55}$$

Thus, the discrepancy between accumulated input increments and emitted events is retained in the membrane residual and remains available for subsequent correction.

This accumulation identity is independent of whether the threshold is constant or time varying, since it follows directly from integration and subtractive reset. With the calibrated base threshold fixed, $\alpha_d(t)$ changes the available event magnitudes over SNN time: larger values allow larger increments, while smaller values provide finer residual correction. The temporal schedule therefore controls how the accumulated input is expressed across SNN steps, while the membrane state preserves the unexpressed remainder.

B.3 Layer–Stage Base Calibration

This section details the activation-range estimation, local neuron simulation, and reconstruction objective used for the layer–stage base calibration in Section 4.5. This stage determines the two static factors $x_{\max,\ell,d,c}$ and $k^{\text{base}}_{\ell,d}$ in Eq. 8; temporal modulation is disabled during base calibration, i.e., $\alpha_d(t) \equiv 1$.

Activation ranges and candidate thresholds. Let $a^{(q)}_{\ell,d}$ denote the ANN activation at layer ℓ and denoising stage d for calibration sample $q \in \{1, \dots, N\}$. With ρ indexing all non-channel dimensions, we estimate the channel-wise activation range as

$$x_{\max,\ell,d,c} = \max_{q,\rho} \left| a^{(q)}_{\ell,d,c}(\rho) \right|. \tag{56}$$

For each candidate coefficient $k \in \mathcal{K}_{\text{base}}$, the corresponding candidate base threshold is

$$\theta^{\text{base}}_{\ell,d,c}(k) = k x_{\max,\ell,d,c}. \tag{57}$$

The activation range $x_{\max,\ell,d,c}$ remains channel specific, whereas k is shared across channels within each layer–stage pair. Thus, $x_{\max,\ell,d,c}$ provides the numerical reference and k adjusts the firing threshold relative to that range.

Local neuron simulation. Each candidate coefficient is evaluated using a local T-step multi-threshold simulation. For calibration sample q, the ANN activation is supplied as an impulse:

$$x^{\text{imp},(q)}_{\ell,d}[t] = \begin{cases} a^{(q)}_{\ell,d}, & t = 1, \\ 0, & t > 1. \end{cases} \tag{58}$$

The simulation follows the neuron dynamics in Appendix B.2 with the candidate threshold in Eq. 57 fixed throughout all T steps. Let $o^{\text{imp},(q)}_{\ell,d}[t; k]$ denote the emitted output at step t. The accumulated output at the full horizon is

$$Q^{\text{imp},(q),[T]}_{\ell,d}(k) = \sum_{t=1}^{T} o^{\text{imp},(q)}_{\ell,d}[t; k]. \tag{59}$$

All candidates are evaluated using the same calibration activations and simulation horizon.

Reconstruction objective. We evaluate reconstruction error after the subsequent weighted operation when applicable. For a convolutional or linear operation, let $g_\ell(x) = W_\ell x$, where W_ℓ denotes the fixed pretrained weights; otherwise, let $g_\ell(x) = x$. The normalized reconstruction loss is

$$\mathcal{L}^{\text{base}}_{\ell,d}(k) = \frac{\sum_{q=1}^{N} \left\| g_\ell\left(Q^{\text{imp},(q),[T]}_{\ell,d}(k)\right) - g_\ell\left(a^{(q)}_{\ell,d}\right) \right\|_2^2}{\sum_{q=1}^{N} \left\| g_\ell\left(a^{(q)}_{\ell,d}\right) \right\|_2^2 + \varepsilon}, \tag{60}$$

where $\varepsilon > 0$ prevents division by zero. This objective measures how accurately the accumulated multi-threshold output reconstructs the corresponding ANN activation in the space defined by g_ℓ.

Coefficient selection. For each layer–stage pair, we select

$$k^{\text{base}}_{\ell,d} = \arg\min_{k \in \mathcal{K}_{\text{base}}} \mathcal{L}^{\text{base}}_{\ell,d}(k), \tag{61}$$

which gives the calibrated base threshold

$$\theta^{\text{base}}_{\ell,d,c} = k^{\text{base}}_{\ell,d} x_{\max,\ell,d,c}. \tag{62}$$

Calibration is performed independently for every layer–stage pair. The resulting activation ranges, base coefficients, and base thresholds remain fixed during the subsequent temporal sensitivity calibration.

B.4 TEMPORAL SENSITIVITY CALIBRATION

This section details the recorded trajectory increments, prefix reconstruction objective, and stage-wise optimization used for temporal sensitivity calibration in Section 4.6. The activation ranges $x_{\max,\ell,d,c}$, base coefficients $k^{\text{base}}_{\ell,d}$, and resulting base thresholds $\theta^{\text{base}}_{\ell,d,c}$ are fixed throughout this procedure; temporal calibration optimizes only the remaining factor $\alpha_d(t)$ in Eq. 8.

Recorded increments and activation prefixes. After base calibration, we record the input increments encountered during trajectory-interleaved inference under the trace-recording configuration used for temporal calibration. For calibration sample $q \in \{1, \dots, N\}$, let $x^{(q)}_{\ell,d}[t]$ denote the input increment at layer ℓ, denoising stage d, and SNN step t. The corresponding accumulated activation prefix is

$$X^{(q),[t]}_{\ell,d} = \sum_{s=1}^{t} x^{(q)}_{\ell,d}[s]. \tag{63}$$

Unlike the impulse inputs used for base calibration, these sequences preserve the stepwise increments encountered along the trajectory. The same recorded sequences are reused when evaluating all candidate temporal schedules.

Candidate-schedule replay. For each denoising stage d, we calibrate a temporal sensitivity schedule

$$\boldsymbol{\alpha}_d = [\alpha_d(1), \dots, \alpha_d(T)], \qquad \alpha_d(t) > 0. \tag{64}$$

The schedule is shared across converted layers and channels within the same denoising stage, while their calibrated base thresholds remain fixed. During replay, the firing threshold is

$$\theta_{\ell,d,c}[t] = \alpha_d(t)\, k^{\text{base}}_{\ell,d}\, x_{\max,\ell,d,c} = \alpha_d(t)\theta^{\text{base}}_{\ell,d,c}. \tag{65}$$

For each candidate $\boldsymbol{\alpha}_d$, we replay the recorded input sequences through the neuron dynamics in Appendix B.2. Let $o^{(q)}_{\ell,d}[s; \boldsymbol{\alpha}_d]$ denote the emitted output at replay step s. Its accumulated output through prefix t is

$$Q^{(q),[t]}_{\ell,d}(\boldsymbol{\alpha}_d) = \sum_{s=1}^{t} o^{(q)}_{\ell,d}[s; \boldsymbol{\alpha}_d]. \tag{66}$$

Membrane residuals are retained across replay steps, and all candidate schedules are evaluated on the same recorded increments.

Prefix reconstruction loss. As in Appendix B.3, reconstruction error is measured after the subsequent weighted operation when applicable. We use $g_\ell(x) = W_\ell x$ for a convolutional or linear operation with fixed pretrained weights W_ℓ, and $g_\ell(x) = x$ otherwise. At SNN prefix t, the normalized reconstruction loss is

$$\mathcal{L}^{[t]}_{\ell,d}(\boldsymbol{\alpha}_d) = \frac{\sum_{q=1}^{N} \left\| g_\ell\left(Q^{(q),[t]}_{\ell,d}(\boldsymbol{\alpha}_d)\right) - g_\ell\left(X^{(q),[t]}_{\ell,d}\right) \right\|_2^2}{\sum_{q=1}^{N} \left\| g_\ell\left(X^{(q),[t]}_{\ell,d}\right) \right\|_2^2 + \varepsilon}, \tag{67}$$

where $\varepsilon > 0$ prevents division by zero. Thus, each prefix compares the accumulated neuron output with the recorded activation accumulated through the same SNN step.

Temporal calibration objective. For each denoising stage, we aggregate the prefix reconstruction losses over the calibrated layers $\mathcal{L}_d$ and all SNN steps, together with a temporal smoothness regularizer:

$$\mathcal{J}_d(\boldsymbol{\alpha}_d) = \sum_{t=1}^{T} \omega_t \sum_{\ell \in \mathcal{L}_d} w_{\ell,d} \mathcal{L}^{[t]}_{\ell,d}(\boldsymbol{\alpha}_d) + \lambda_{\text{sm}} \sum_{t=1}^{T-1} \left(\log \alpha_d(t+1) - \log \alpha_d(t)\right)^2. \tag{68}$$

The layer weights $w_{\ell,d}$ control the relative contributions of calibrated layers, while the prefix weights ω_t control their importance over SNN time. Larger weights on early prefixes emphasize short-prefix reconstruction, and λ_{sm} regularizes abrupt changes between adjacent temporal coefficients.

Algorithm 1 Trajectory-interleaved SNN inference

Input: converted denoiser, initial latent z_{init}, condition, D, T
Parameters: schedule $\{\tau_d\}$, scheduler coefficients, $\{x_{\max,\ell,d,c}\}$, $\{k_{\ell,d}^{\text{base}}\}$, $\{\boldsymbol{\alpha}_d\}_{d=1}^{D}$
Reset the retained SNN state $\mathcal{S}_d$ separately for each stage d
Sample and store $\xi_d \sim \mathcal{N}(0, I)$ independently for $d = 1, \ldots, D-1$
$\hat{z}_0^{[0]} \leftarrow 0$
for $t = 1, \ldots, T$ **do**
  $\Delta z_{\tau_1}[t] \leftarrow \mathbf{1}_{t=1} z_{\text{init}}$
  for $d = 1, \ldots, D$ **do**
    Set $\theta_{\ell,d,c}[t] \leftarrow \alpha_d(t) k_{\ell,d}^{\text{base}} x_{\max,\ell,d,c}$ for all converted layers ℓ and channels c
    Run one SNN denoiser step using $\Delta z_{\tau_d}[t]$, the condition, τ_d, $\mathcal{S}_d$, and the current thresholds
    Apply the required model-output parameterization conversion to obtain $\Delta f_d[t]$; retain the updated $\mathcal{S}_d$
    if $d < D$ **then**
      $\Delta z_{\tau_{d+1}}[t] \leftarrow A_d \Delta z_{\tau_d}[t] + B_d \Delta f_d[t] + \mathbf{1}_{t=1} C_d \xi_d$
    else
      $\Delta z_0[t] \leftarrow A_D^{\text{out}} \Delta z_{\tau_D}[t] + B_D^{\text{out}} \Delta f_D[t]$
    end if
  end for
  $\hat{z}_0^{[t]} \leftarrow \hat{z}_0^{[t-1]} + \Delta z_0[t]$
  Decode $\hat{x}^{[t]} = \mathcal{D}_{\text{VAE}}(\hat{z}_0^{[t]})$ if a readout is requested
end for
Output: terminal latent prefixes $\{\hat{z}_0^{[t]}\}_{t=1}^{T}$

Stage-wise optimization. Because the multi-threshold firing rule involves discrete level selection, we optimize the temporal schedule using covariance matrix adaptation evolution strategy (CMA-ES). We parameterize the search in the log domain:

$$\mathbf{y}_d = [y_d(1), \ldots, y_d(T)], \qquad y_d(t) = \log \alpha_d(t). \tag{69}$$

For each candidate $\mathbf{y}_d$, we set $\boldsymbol{\alpha}_d = \exp(\mathbf{y}_d)$ elementwise, replay the recorded increments, and evaluate $\mathcal{J}_d(\exp(\mathbf{y}_d))$. CMA-ES jointly optimizes all T coefficients within the stage, while the exponential parameterization guarantees $\alpha_d(t) > 0$.

Each denoising stage is calibrated independently. Let $\mathbf{y}_d^\star$ denote the optimized solution; the resulting temporal schedule is

$$\boldsymbol{\alpha}_d = \exp(\mathbf{y}_d^\star). \tag{70}$$

All pretrained network weights, activation ranges, base coefficients, and base thresholds remain fixed throughout temporal calibration.

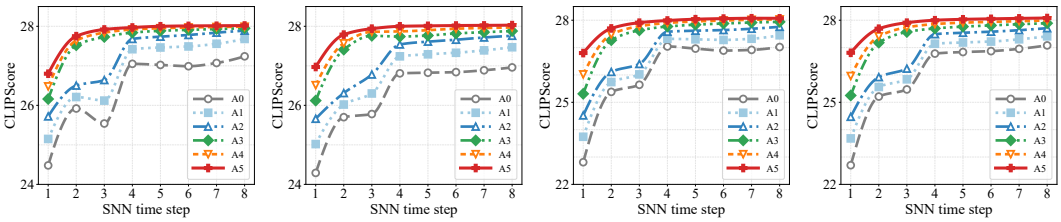
B.5 Inference and Calibration Procedures

Trajectory-interleaved inference. Algorithm 1 summarizes Spike-by-Spike Diffusion inference. At each global SNN step, all denoising stages are advanced once in sampling order while retaining a separate SNN state $\mathcal{S}_d$ for each stage. The firing thresholds follow Eq. 8, the initial latent and transition noise are injected only at $t = 1$, and the terminal latent is accumulated after every complete trajectory traversal.

Base and temporal calibration. Algorithm 2 summarizes the two-stage threshold calibration. Base calibration first estimates the channel-wise activation ranges and selects a base coefficient independently for each layer–stage pair. The resulting base thresholds are then fixed while trajectory-interleaved input increments are recorded. Temporal calibration subsequently optimizes one schedule $\boldsymbol{\alpha}_d$ for each denoising stage by replaying these recorded increments. All pretrained network weights remain fixed throughout.

Algorithm 2 Layer–stage base and temporal sensitivity calibration

Input: converted denoiser, ANN calibration activations, calibration samples
Parameters: $\mathcal{K}_{\text{base}}$, D, T, n, trace-recording configuration,
CMA-ES settings
Layer–stage base calibration:
for $d = 1, \dots, D$ and each converted layer ℓ **do**
Compute $x_{\max,\ell,d,c}$ for every channel c using Eq. 56
for each $k \in \mathcal{K}_{\text{base}}$ **do**
Evaluate $\mathcal{L}^{\text{base}}_{\ell,d}(k)$ by local T-step impulse simulation
with fixed threshold $k x_{\max,\ell,d,c}$
end for
$k^{\text{base}}_{\ell,d} \leftarrow \arg\min_{k \in \mathcal{K}_{\text{base}}} \mathcal{L}^{\text{base}}_{\ell,d}(k)$
Set $\theta^{\text{base}}_{\ell,d,c} \leftarrow k^{\text{base}}_{\ell,d} x_{\max,\ell,d,c}$ for every channel c
end for
Trajectory trace recording:
Run trajectory-interleaved inference using the calibrated base thresholds
under the prescribed trace-recording configuration
Record input increments $\{x^{(q)}_{\ell,d}[t]\}$ for all calibration samples,
stages, converted layers, and SNN steps
Stage-wise temporal calibration:
for $d = 1, \dots, D$ **do**
Initialize $\mathbf{y}_d = [y_d(1), \dots, y_d(T)]$, where $y_d(t) = \log \alpha_d(t)$
Replay the recorded increments for each candidate $\boldsymbol{\alpha}_d = \exp(\mathbf{y}_d)$
Evaluate $\mathcal{J}_d(\exp(\mathbf{y}_d))$ using Eq. 68
Use CMA-ES to obtain $\mathbf{y}^\star_d \leftarrow \arg\min_{\mathbf{y}_d} \mathcal{J}_d(\exp(\mathbf{y}_d))$
$\boldsymbol{\alpha}_d \leftarrow \exp(\mathbf{y}^\star_d)$
end for
Output: $\{x_{\max,\ell,d,c}\}$, $\{k^{\text{base}}_{\ell,d}\}$, $\{\theta^{\text{base}}_{\ell,d,c}\}$, and $\{\boldsymbol{\alpha}_d\}_{d=1}^{D}$



(a) $D = 2,\ n = 8$ (b) $D = 2,\ n = 10$ (c) $D = 4,\ n = 8$ (d) $D = 4,\ n = 10$

Figure 9: **CLIPScore ablations on SANA-Sprint.** CLIPScore versus SNN readout step on MJHQ-30K ($T = 8$) for variants A0–A5 under $D \in \{2, 4\}$ and $n \in \{8, 10\}$. Temporal calibration yields its largest gains at early prefixes, with the stage-specific schedule A5 providing the strongest early-readout alignment.

C Additional Experimental Results

C.1 Additional Ablation Results

We provide additional ablations on SANA-Sprint and Stable Diffusion v1.5 to complement the FID results in Section 5.3. The variant definitions A0–A5 follow Table 3: A0–A3 progressively introduce stage-specific activation ranges and base-threshold calibration, while A4 and A5 add shared and stage-specific temporal modulation, respectively.

SANA-Sprint: text–image alignment. Figure 9 reports CLIPScore over all eight SNN readout steps for the same four SANA-Sprint configurations used in the main-text FID ablation: $D \in \{2, 4\}$ and $n \in \{8, 10\}$. The CLIPScore trends are consistent with the FID results in Figure 5. Introducing stage-aware activation ranges and progressively calibrated base coefficients from A0 to A3 improves semantic alignment, with the largest differences appearing at short prefixes. Adding temporal modulation further improves early readouts: A5, which uses a separate temporal schedule $\alpha_d(t)$ for

Table 4: **Stable Diffusion v1.5 calibration ablations on COCO.** FID and CLIPScore for A0–A5 at $t \in \{4, 6, 8\}$ and $n \in \{8, 10\}$. Stage-aware range and base calibration provide large gains from A0 to A3, while temporal modulation in A4–A5 primarily improves short-prefix generation.

(a) COCO2014-10K

Variant	t	$n = 8$		$n = 10$	
		FID↓	CLIP↑	FID↓	CLIP↑
A0	4	207.15	15.21	224.37	16.15
	6	78.65	20.15	84.22	20.64
	8	44.16	22.16	45.63	22.47
A1	4	168.47	17.01	183.26	17.86
	6	59.33	22.15	62.15	22.57
	8	36.85	23.85	37.16	24.16
A2	4	136.27	18.73	151.30	19.25
	6	47.53	23.25	48.86	23.44
	8	32.50	24.53	32.16	24.67
A3	4	56.32	22.15	64.16	22.36
	6	28.74	24.47	30.15	24.58
	8	27.01	24.91	27.42	24.94
A4	4	44.16	23.01	51.27	23.16
	6	27.68	24.68	28.74	24.74
	8	26.81	25.07	27.08	25.05
A5	4	**39.04**	**23.58**	**46.06**	**23.49**
	6	**26.93**	**24.84**	**27.89**	**24.87**
	8	**26.75**	**25.13**	**27.02**	**25.12**

(b) COCO2017-5K

Variant	t	$n = 8$		$n = 10$	
		FID↓	CLIP↑	FID↓	CLIP↑
A0	4	214.64	15.15	231.55	16.08
	6	85.74	20.08	91.35	20.56
	8	51.26	22.07	52.74	22.40
A1	4	175.63	16.95	190.47	17.79
	6	66.47	22.07	69.18	22.51
	8	44.07	23.77	44.39	24.08
A2	4	143.47	18.54	158.66	19.24
	6	54.48	23.14	55.74	23.43
	8	38.93	24.52	38.52	24.73
A3	4	63.48	22.07	71.26	22.29
	6	34.89	24.41	36.27	24.51
	8	32.81	24.91	33.27	24.94
A4	4	51.26	22.95	58.36	23.08
	6	33.89	24.61	35.07	24.68
	8	**32.71**	25.09	32.95	25.07
A5	4	**44.82**	**23.53**	**52.02**	**23.47**
	6	**32.79**	**24.89**	**34.04**	**24.88**
	8	32.77	**25.20**	**32.88**	**25.19**

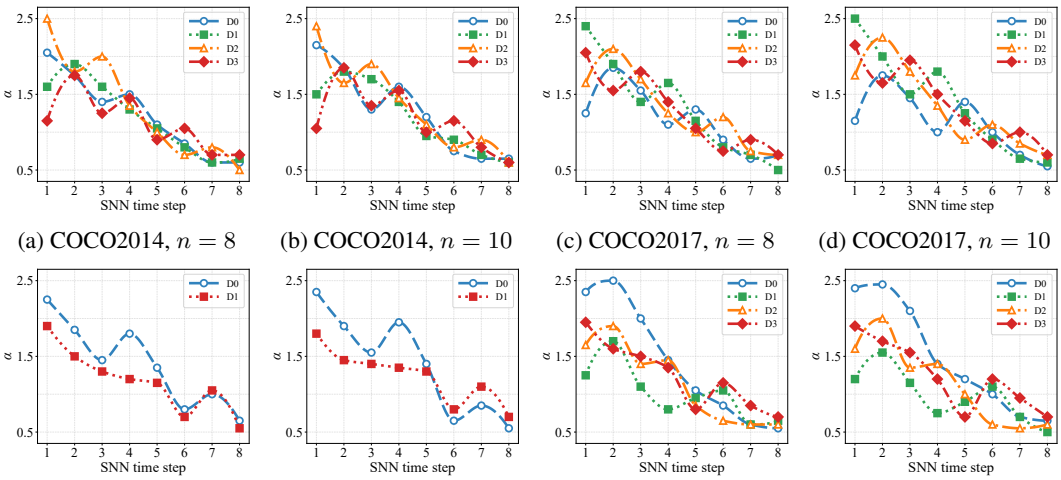
each denoising stage, achieves the strongest early-prefix CLIPScore across all four configurations. As SNN time increases, the gaps among the calibrated variants become smaller, indicating that temporal calibration primarily improves how quickly semantically meaningful content is established rather than only changing the final output. Together with the FID curves, these results show that the gains from calibration reflect both improved image distribution quality and improved text–image alignment.

Stable Diffusion: stage-aware range and base calibration. Table 4 reports the corresponding ablation on COCO2014-10K and COCO2017-5K. The effect of stage-aware calibration is especially pronounced at short prefixes. At $t = 4$ and $n = 8$, moving from the shared activation range in A0 to the fully stage-aware base calibration in A3 reduces FID from 207.15 to 56.32 on COCO2014 and from 214.64 to 63.48 on COCO2017; CLIPScore simultaneously increases from 15.21 to 22.15 and from 15.15 to 22.07, respectively. The same behavior holds for $n = 10$, where A0→A3 reduces FID from 224.37 to 64.16 on COCO2014 and from 231.55 to 71.26 on COCO2017, while increasing CLIPScore from 16.15 to 22.36 and from 16.08 to 22.29.

The intermediate variants further separate the contribution of each component. Replacing the shared activation range by stage-specific ranges (A0→A1) already produces a substantial improvement. Calibrating the base coefficient at the layer level (A1→A2) improves both FID and CLIPScore further, and making this coefficient stage specific (A2→A3) produces the largest additional reduction in early-prefix FID. For example, on COCO2014 with $n = 8$ and $t = 4$, FID progresses from 207.15 (A0) to 168.47 (A1), 136.27 (A2), and 56.32 (A3), while CLIPScore increases from 15.21 to 17.01, 18.73, and 22.15. This progression supports calibrating both the activation range and the base threshold relative to the denoising stage.

Temporal modulation improves short-prefix generation. Starting from the stage-specific base calibration in A3, A4 introduces a temporal schedule shared across denoising stages, whereas A5 calibrates a separate schedule for each stage. Their effect is again strongest at early readouts. On COCO2014 with $n = 8$ and $t = 4$, FID decreases from 56.32 (A3) to 44.16 (A4) and 39.04 (A5), while CLIPScore increases from 22.15 to 23.01 and 23.58. For $n = 10$, the corresponding FIDs are 64.16, 51.27, and 46.06, with CLIPScores of 22.36, 23.16, and 23.49.

The same pattern appears on COCO2017. At $t = 4$, A3→A4→A5 reduces FID from 63.48 to 51.26 and 44.82 for $n = 8$, and from 71.26 to 58.36 and 52.02 for $n = 10$. CLIPScore increases from 22.07 to 22.95 and 23.53 for $n = 8$, and from 22.29 to 23.08 and 23.47 for $n = 10$. These results



(a) COCO2014, $n = 8$ (b) COCO2014, $n = 10$ (c) COCO2017, $n = 8$ (d) COCO2017, $n = 10$

(e) $D = 2,\ n = 8$ (f) $D = 2,\ n = 10$ (g) $D = 4,\ n = 8$ (h) $D = 4,\ n = 10$

Figure 10: **Calibrated stage-specific temporal sensitivity schedules. (a–d)** Stable Diffusion v1.5 with four-stage LCM-LoRA sampling on COCO2014-10K and COCO2017-5K. **(e–h)** SANA-Sprint on MJHQ-30K with $D \in \{2, 4\}$. Each curve corresponds to one denoising stage. The schedules exhibit stage-dependent temporal structure, while paired $n = 8$ and $n = 10$ settings show similar profiles.

show that temporal modulation primarily accelerates the formation of useful early prefixes, while stage-specific schedules provide an additional gain over a single shared temporal profile.

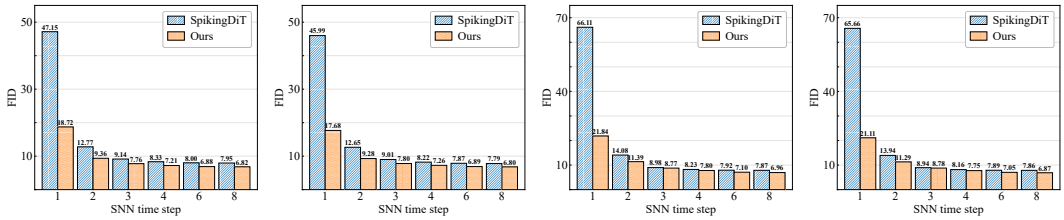
Convergence at later readouts. As temporal accumulation proceeds, the calibrated variants become increasingly close. At $t = 8$, A4 and A5 differ only slightly across both COCO datasets. For example, on COCO2017 with $n = 8$, A4 attains an FID of 32.71 and A5 32.77, while A5 obtains the higher CLIPScore (25.20 versus 25.09). On COCO2014 with the same setting, A5 gives FID 26.75 and CLIPScore 25.13, compared with 26.81 and 25.07 for A4. This convergence is consistent with the purpose of temporal sensitivity calibration: its largest benefit appears at short prefixes, where threshold allocation over SNN time has the strongest effect on the quality of an immediately available readout.

Overall, the ablations show a consistent hierarchy of effects. Stage-specific activation ranges establish an appropriate numerical reference for each denoising stage; layer–stage base coefficients improve the finite-horizon representation relative to those ranges; and stage-specific temporal schedules further improve how quickly high-quality and semantically aligned terminal outputs emerge during SNN accumulation.

C.2 Additional Temporal Sensitivity Schedules

Figure 10 extends the temporal-schedule visualization in Figure 6(a) to all evaluated Stable Diffusion v1.5 and SANA-Sprint configurations. The calibrated coefficients are clearly stage dependent: different denoising stages follow distinct temporal trajectories rather than collapsing to a common profile. Across configurations, $\alpha_d(t)$ shows an overall tendency to decrease as SNN accumulation proceeds, but individual stages contain local peaks, plateaus, or rebounds. Temporal sensitivity is therefore not captured well by a single monotonic decay or by a schedule shared across denoising stages.

Stage-specific temporal structure. For Stable Diffusion v1.5, panels (a–d) show separate schedules for the four LCM-LoRA denoising stages on both COCO datasets. Although the stages share the same denoiser weights, their calibrated temporal profiles differ in relative magnitude and in the locations of local turning points. The same behavior appears for SANA-Sprint. Panels (e–f) show distinct profiles for the two-stage sampler, while panels (g–h) show four separate schedules when the trajectory is expanded to $D = 4$. Panel (g) corresponds to the SANA-Sprint configuration visualized



(a) $D = 2,\ n = 8$ (b) $D = 2,\ n = 10$ (c) $D = 4,\ n = 8$ (d) $D = 4,\ n = 10$

Figure 11: **Generation quality at prescribed SNN budgets on SANA-Sprint.** FID at budgets $B \in \{1, 2, 3, 4, 6, 8\}$ on MJHQ-30K. SpikingDiT uses an independent run with horizon $T = B$ at each budget, whereas our method reports the $t = B$ prefix from a single $T = 8$ trajectory-interleaved run.

in the main text. These differences support calibrating $\alpha_d(t)$ independently for each denoising stage, consistent with the improvement from the shared temporal schedule A4 to the stage-specific schedule A5 in Section 5.3.

Consistency across threshold-level counts. For a fixed dataset and denoising configuration, changing the number of threshold levels from $n = 8$ to $n = 10$ preserves the main temporal structure of the calibrated schedules. The paired settings in panels (a,b), (c,d), (e,f), and (g,h) exhibit similar relative ordering among stages and similar major turning points over SNN time, even though the optimized coefficients are not identical. This behavior indicates that the temporal schedules primarily capture stage-dependent dynamics of trajectory-interleaved inference and remain consistent across the two evaluated threshold-level settings.

C.3 Generation Quality at Prescribed SNN Budgets

Figure 11 reports the complete SANA-Sprint comparison at prescribed SNN budgets $B \in \{1, 2, 3, 4, 6, 8\}$ for $D \in \{2, 4\}$ and $n \in \{8, 10\}$. At budget B, SpikingDiT is run independently with horizon $T = B$ and evaluated at its final output, whereas our method performs one $T = 8$ trajectory-interleaved run and reports the prefix at $t = B$. Both methods therefore use B SNN steps per denoising stage at the reported budget, but only our outputs at different budgets belong to the same progressively refined generation.

Two-stage sampling. For $D = 2$ and $n = 8$, our FID decreases from 18.72 at $B = 1$ to 9.36, 7.76, 7.21, 6.88, and 6.82 at $B = 2, 3, 4, 6, 8$, respectively. SpikingDiT obtains 47.15, 12.77, 9.14, 8.33, 8.00, and 7.95 at the same budgets. The largest difference occurs at $B = 1$, where trajectory-interleaved execution lowers FID by 28.43 points. The gap narrows as additional SNN steps accumulate but remains present through the full horizon.

The $n = 10$ setting exhibits the same behavior. Our FID progresses from 17.68 to 9.28, 7.80, 7.26, 6.89, and 6.80, compared with 45.99, 12.65, 9.01, 8.22, 7.87, and 7.79 for SpikingDiT. The corresponding first-budget improvement is 28.31 FID points. The close behavior of the $n = 8$ and $n = 10$ settings shows that the budget advantage is consistent across the two evaluated threshold-level counts.

Four-stage sampling. The early-budget difference becomes larger for the longer $D = 4$ trajectory. With $n = 8$, our method achieves an FID of 21.84 at $B = 1$, compared with 66.11 for SpikingDiT, a reduction of 44.27 points. At $B = 2$, the respective FIDs are 11.39 and 14.08. Our prefixes then improve to 8.77, 7.80, 7.10, and 6.96 at $B = 3, 4, 6, 8$, while SpikingDiT obtains 8.98, 8.23, 7.92, and 7.87.

For $n = 10$, our FID decreases from 21.11 at $B = 1$ to 11.29, 8.78, 7.75, 7.05, and 6.87, whereas SpikingDiT obtains 65.66, 13.94, 8.94, 8.16, 7.89, and 7.86. At the first budget, this corresponds to a 44.55-point reduction. Compared with $D = 2$, the substantially larger $B = 1$ gaps for $D = 4$ highlight the increasing cost of stage-major execution at short horizons as more denoising stages must be traversed before a terminal result is formed.

Early prefixes versus full baseline runs. A notable result is that the $B = 4$ prefix of our single $T = 8$ run already achieves lower FID than SpikingDiT's complete $T = 8$ run in all four configurations. For $(D, n) = (2, 8)$ and $(2, 10)$, our $B = 4$ FIDs are 7.21 and 7.26, compared with SpikingDiT's $B = 8$ FIDs of 7.95 and 7.79. For $(D, n) = (4, 8)$ and $(4, 10)$, the corresponding comparisons are 7.80 versus 7.87 and 7.75 versus 7.86. Thus, the progressively accumulated trajectory reaches the quality of the stage-major full-horizon result substantially before exhausting its own SNN horizon.

Overall trend. Across all 24 budget–configuration pairs, our method achieves lower FID than SpikingDiT. The advantage is largest at the shortest budgets, particularly for the longer $D = 4$ trajectory, and gradually narrows as both methods accumulate more SNN computation. Importantly, the comparison controls the number of SNN steps per denoising stage at every reported budget. The gains therefore extend beyond earlier terminal availability: trajectory-interleaved execution also produces higher-quality accumulated prefixes under the same prescribed SNN-step budget while retaining the ability to continue refining the same generation.

C.4 Additional Qualitative Results

We provide two complementary qualitative analyses. The first visualizes how a single generation evolves across SNN readout steps, while the second presents additional final outputs to illustrate the range of content produced by trajectory-interleaved SNN inference.

Single-run progressive refinement. Figure 12 shows the complete sequence of terminal readouts for ten SANA-Sprint examples. In these examples, the first few traversals already establish the primary subject, global layout, and dominant appearance of the image. Subsequent SNN steps preserve this overall structure while progressively refining object boundaries, facial and surface details, textures, lighting, and background content. The refinement is particularly visible in prompts containing detailed animal appearances, accessories, or structured scene elements, where later readouts sharpen local features without reconstructing the image from scratch.

The progression also illustrates that temporal refinement is not simply a sequence of independent generations. Once the main semantic structure is established, later readouts evolve from the retained SNN states of the same trajectory. Local details may continue to change, but the subject identity and global composition remain largely consistent across successive prefixes. This behavior directly reflects the trajectory-level execution of Spike-by-Spike Diffusion: each additional global SNN step contributes new computation to the same terminal generation.

Additional final generations. Figure 13 presents 54 additional final readouts from SANA-Sprint. The selected examples cover a broad range of visual content, including human subjects, animal portraits, natural scenes, close-up compositions, stylized characters, and fantasy or illustrative imagery. Across these examples, the converted SNN preserves detailed object appearance, coherent global composition, and diverse visual styles at the final readout.

Together with the ANN–SpikingDiT–Ours comparison in Figure 7, these visualizations provide two complementary views of the proposed method. Figure 12 illustrates how generation quality develops progressively within a single run, whereas Figure 13 shows the breadth and visual fidelity of the resulting final outputs. These qualitative observations are consistent with the quantitative prefix and final-readout results reported in the main text.

Prompt: a different wolf eyes, collage spectacular image,rich,stunning photo,beautiful,realistic,cyberpunk neon colors



Prompt: the animal Duck looks like the movie Star Wars, 8k



Prompt: a black lion in futuristic shades wearing neon sunglasses, in the style of dan mumford, 32k uhd, valentin rekunenko, lasar segall, mashup of styles, 1990s, solapunk



Prompt: image of pink octopus, in the sea taking out the tentacles



Prompt: a unique photorealistic illustration of a hooded humanoid wolf holding a lightsaber, hypercore style



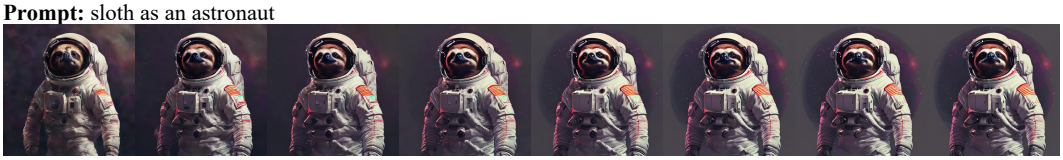
Prompt: Weimaraner synthwave, 80s sunset in background



Prompt: a photograph of a boarder collie dog wearing glasses behind a computer doing work at the office



Prompt: sloth as an astronaut



Prompt: photo of a lion splashing in the ocean, in the style of realistic hyperdetailed portraits, postapocalyptic imagery, ultra hd, characterized animals, moody color schemes, datamosh, strong facial expression



Figure 12: **Single-run progressive refinement on SANA-Sprint.** Each row corresponds to one prompt, and the eight columns show terminal readouts at $t = 1, \ldots, 8$ from the same $T = 8$ trajectory-interleaved run. These are successive prefixes of one generation rather than outputs from independently restarted runs.

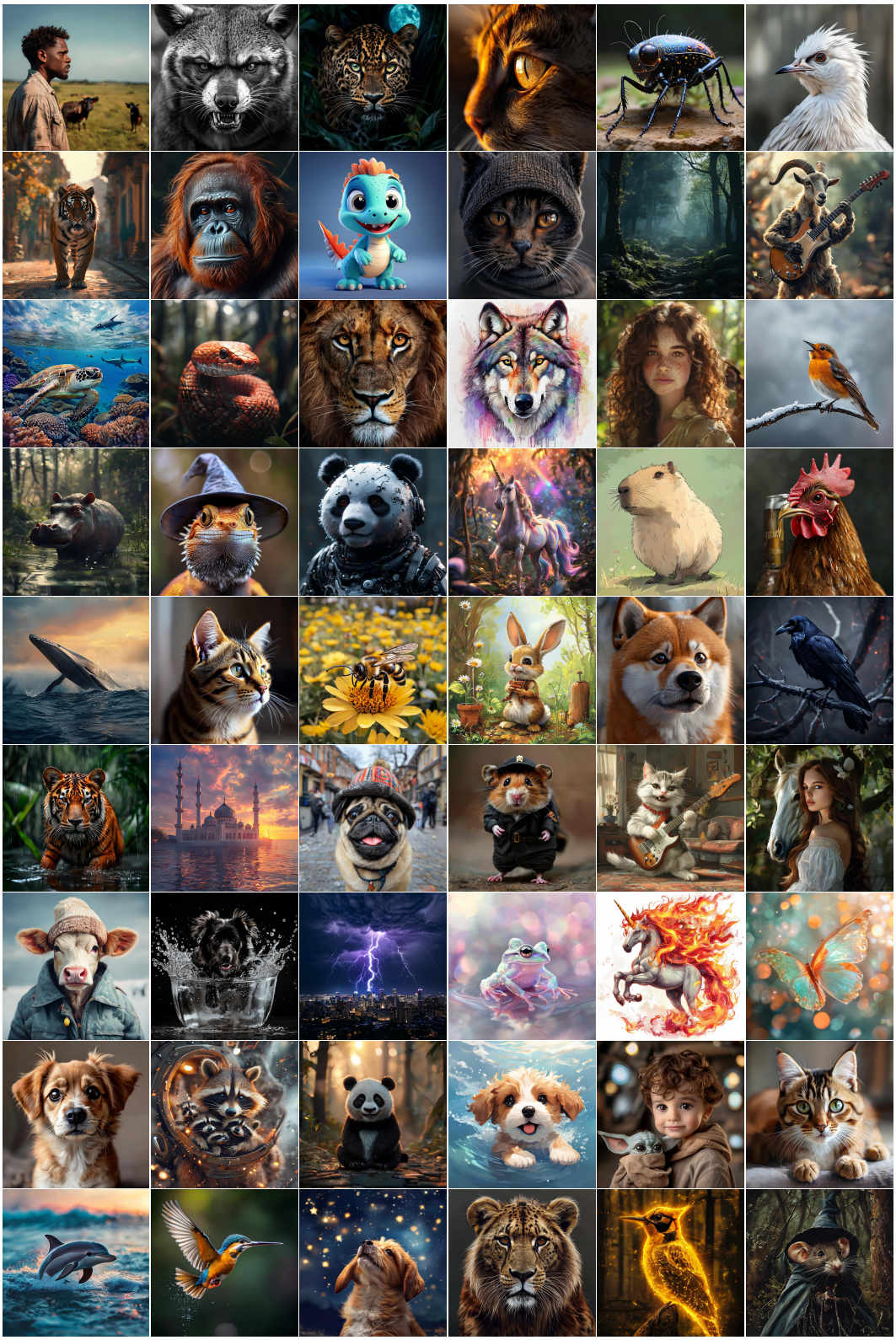


Figure 13: **Additional final generations on SANA-Sprint.** Fifty-four selected outputs from our method with $D = 4$, $n = 8$, and $t = T = 8$. The examples span different subjects, scenes, compositions, and visual styles.