DS-FLOW: DUAL-STREAM DYNAMICAL SYSTEM CORRECTION FOR FEW-STEP DIFFUSION SAMPLING

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion models have shown remarkable success in generative modeling, but maintaining high quality at extremely low numbers of function evaluations (NFE) remains challenging due to large truncation errors under coarse ODE discretization. Existing methods either rely on costly distillation or perform step-level correction that adjusts the sampling direction only once per interval, which limits fine-grained trajectory refinement. In this work, we propose DS-Flow, which embeds an electronic dynamical-system (DS) corrector within each sampling step, enabling continuous in-step trajectory refinement through multiple hardware-efficient micro-steps without additional denoiser calls. While a PCA subspace is the natural choice for keeping each DS update lightweight, our analysis reveals that online PCA captures only $\sim$70% of truncation error; the residual exhibits spatial structure that a low-dimensional subspace cannot adequately represent. This motivates a dual-stream design: a global DS with dense coupling corrects the dominant low-dimensional error, while a local DS with grid coupling captures spatial residuals directly in pixel space. Extensive experiments demonstrate that DS-Flow consistently outperforms existing methods. For example, on CIFAR-10, DS-Flow reduces the Euler baseline's FID from 49.66 to **7.95** at NFE=5 while the corrector adds only $\sim$0.1% parameters. On DS hardware, the corrector runs at $\sim$mW power and $\sim\mu$s latency, achieving $\sim 10^4\times$ energy reduction and 6–7$\times$ speedup per correction interval over GPU-based alternatives.

1 INTRODUCTION

Diffusion models (Sohl-Dickstein et al., 2015; Song & Ermon, 2019; Ho et al., 2020; Song et al., 2021b) have emerged as a powerful paradigm for generative modeling. They generate samples by numerically integrating an ordinary differential equation (ODE) driven by a learned denoiser, gradually transforming Gaussian noise into data. This iterative refinement process has enabled them to surpass GANs (Goodfellow et al., 2014) and VAEs (Kingma & Welling, 2014) in image (Dhariwal & Nichol, 2021; Rombach et al., 2022; Karras et al., 2022; Peebles & Xie, 2023; Esser et al., 2024) and video (Ho et al., 2022) generation domains. However, this comes at a substantial computational cost: each solver step requires evaluating a large denoising network (Dhariwal & Nichol, 2021; Peebles & Xie, 2023), and high-quality generation typically requires hundreds of such evaluations. Many practical applications, including real-time generation, on-device deployment, and interactive editing, demand high-quality diffusion sampling at NFE budgets of 10 or fewer, yet this remains a fundamental open challenge.

Existing acceleration strategies fall into three categories: model distillation (Salimans & Ho, 2022; Song et al., 2023; Luo et al., 2023; Yin et al., 2024), higher-order ODE solvers (Song et al., 2021a; Lu et al., 2022a;b; Zhang & Chen, 2023; Liu et al., 2022; Zhao et al., 2023), and lightweight learned corrections (Zheng et al., 2023; Zhou et al., 2024; Wang et al., 2025; Chen et al., 2024; Sabour et al., 2024) (Section 2.2 provides a detailed review). Despite their diversity, all training-free methods share a fundamental limitation: they perform step-level correction that adjusts the sampling direction only once per interval, and cannot steer the trajectory progressively within a step. This one-shot strategy is insufficient when step sizes grow large at low NFE, since a single correction cannot follow the substantial curvature of the true ODE trajectory within an interval.

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.

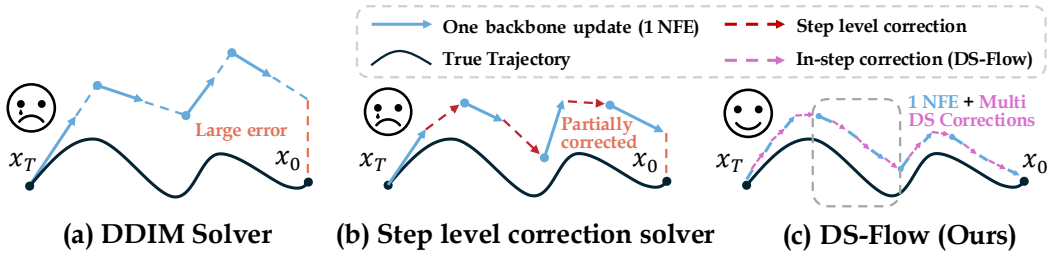


Figure 1: Sampling strategies. **(a)** DDIM Solver. **(b)** Step-level correction adds one correction per backbone update. **(c)** DS-Flow inserts multiple in-step DS corrections per backbone update.

To address this, we propose DS-Flow, which embeds an electronic dynamical-system (DS) corrector within each sampling step. A DS is a programmable analog circuit whose state evolves toward a learned equilibrium through natural annealing, with each annealing cycle running at near-zero cost in the analog domain. This makes DS a natural fit for in-step correction: both diffusion sampling and DS are governed by ODEs that drive a state toward a target, but DS performs each evolution at hardware speed. A DS corrector thus integrates seamlessly into the sampling loop (Wu et al., 2024; Liu et al., 2025b;c;a; Wu et al., 2025), performing multiple in-step corrections per solver interval (Figure 1). The remaining design question is what each micro-step should correct. To keep each DS update lightweight, we project corrections into a low-dimensional PCA subspace, following prior observations that sampling trajectories concentrate near a low-dimensional subspace (Wang et al., 2025; Zhou et al., 2024). However, our analysis reveals that online PCA leaves a substantial spatially structured residual, as only $\sim$70% of the truncation error fits within a compact subspace. This motivates a dual-stream design that combines low-dimensional correction in PCA subspace with spatial correction directly in pixel space. Our contributions can be summarized as follows:

- We propose DS-Flow, the first method to bridge electronic dynamical systems with diffusion sampling, transforming one-shot direction correction into continuous in-step refinement at zero additional NFE cost.
- Through truncation error analysis, we identify a spatially structured residual that motivates a dual-stream design combining PCA-subspace and pixel-space correction.
- We provide a hardware realization on a resistively coupled capacitor network with a hybrid GPU-DS deployment, where the denoiser runs on GPU while all micro-step corrections execute natively in the analog domain.
- Experiments across three benchmarks validate that DS-Flow consistently outperforms methods built on higher-order solvers, with negligible parameter overhead and substantial energy savings.

2 Background and Related Work

2.1 Diffusion ODE Sampling

Diffusion models (Ho et al., 2020; Song et al., 2021b; Nichol & Dhariwal, 2021) generate samples by solving a probability flow ODE that reverses a noise corruption process (Anderson, 1982). Under the EDM framework (Karras et al., 2022), this ODE takes the form:

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \frac{\mathbf{x} - D_\theta(\mathbf{x}, t)}{t}, \tag{1}$$

where $D_\theta(\mathbf{x}, t)$ is a pretrained denoiser. To draw a sample, we start from $\mathbf{x}_{t_N} \sim \mathcal{N}(\mathbf{0}, t_N^2\mathbf{I})$ and apply an ODE solver to numerically integrate Eq. (1) along a discrete time schedule $\{t_0{\approx}0 < t_1 < \cdots < t_N{=}T\}$. The simplest such solver is the first-order Euler solver:

$$\mathbf{x}_{i-1} = \mathbf{x}_i + (t_{i-1} - t_i)\,\mathbf{d}_i, \qquad \mathbf{d}_i = \frac{\mathbf{x}_i - D_\theta(\mathbf{x}_i, t_i)}{t_i}, \tag{2}$$

where $\mathbf{d}_i \in \mathbb{R}^D$ with $D{=}C{\times}H{\times}W$ is the denoising direction. Each evaluation of D_θ counts as one NFE. This Euler discretization incurs a per-step truncation error of $\mathcal{O}(h^2)$, where $h{=}|t_{i-1}{-}t_i|$ is the step size. At low NFE ($\leq$10), h becomes large, making truncation error the dominant source of quality degradation and the central target of this work.

2.2 Accelerating Diffusion Sampling

Dedicated ODE solvers. A first line of work accelerates diffusion sampling by designing better ODE solvers without additional training. The foundation is DDIM (Song et al., 2021a), which reformulates the reverse process as a deterministic ODE. Building on this, exponential integrators have proven particularly effective at exploiting the semi-linear ODE structure: DPM-Solver (Lu et al., 2022a) and DPM-Solver++ (Lu et al., 2022b) establish this approach, while DEIS (Zhang & Chen, 2023) further incorporates polynomial score approximation. Beyond exponential integrators, iPNDM (Liu et al., 2022; Zhang & Chen, 2023) adapts pseudo-linear multistep methods and UniPC (Zhao et al., 2023) unifies predictor-corrector schemes to arbitrary orders. While these solvers reduce the step count from hundreds to $\sim$20, they still degrade rapidly below 10 NFE as truncation error accumulates.

Model distillation. An alternative line of work avoids the truncation error problem entirely by training a student network to replicate many-step teacher outputs in fewer steps. Representative approaches include progressive student training (Salimans & Ho, 2022; Meng et al., 2023), self-consistency along the ODE trajectory (Song et al., 2023; Song & Dhariwal, 2024; Luo et al., 2023; Zheng et al., 2024), distribution matching (Yin et al., 2024), and learning straighter transport paths (Liu et al., 2023). While distillation can achieve high quality in very few steps, it requires retraining a comparably sized model for each architecture, incurring substantial computational cost and limiting flexibility.

Lightweight learned correction. To improve low-NFE quality without the cost of full distillation, a recent line of work attaches small learned modules to existing solvers. A common approach is to correct the sampling direction: DPM-Solver-v3 (Zheng et al., 2023) incorporates empirical model statistics into the solver coefficients, and AMED-Solver (Zhou et al., 2024) learns an optimal mean direction via an intermediate time point. Most relevant to our work, PAS (Wang et al., 2025) exploits the observation that sampling trajectories concentrate near a low-dimensional subspace and learns a few PCA-subspace coordinates for direction correction. Other works such as GITS (Chen et al., 2024) and AYS (Sabour et al., 2024) instead optimize the time discretization. Despite their efficiency, all these methods share a fundamental limitation: correction is applied only once per solver interval, leaving the trajectory unrefined within each step. This becomes critical at low NFE, where each step spans a large interval.

2.3 Electronic Dynamical Systems

Electronic dynamical systems (DS) have recently emerged as a physics-based computing paradigm that leverages the intrinsic physics of CMOS circuits such as resistors, capacitors, and switches to perform computation at the speed of electrons with milliwatt-level power consumption (Inagaki et al., 2016; Afoakwa et al., 2021; Mohseni et al., 2022; Wu et al., 2024). Unlike standard neural networks, where parameters are abstract numerical weights processed by digital arithmetic units, every DS parameter has a direct physical embodiment: coupling strengths J_{ij} are conductances of programmable resistors, node states V_i are voltages on nanoscale capacitors, and nonlinearities are implemented by clamping circuits (Liu et al., 2025b). Crucially, this design lets the physical circuit dynamics perform inference natively in the analog domain, rather than through sequential digital operations on GPUs (Wu et al., 2024; Liu et al., 2025b).

Formally, the system encodes the target function into an energy landscape governed by the Hamiltonian (Wu et al., 2024; Liu et al., 2025b):

$$H = -\sum_{i \neq j} J_{ij}\, V_i\, V_j \;+\; \sum_i r_i\, V_i^2, \tag{3}$$

where $V_i \in \mathbb{R}$ is the real-valued node state stored as a capacitor voltage, J_{ij} is the coupling strength between nodes i and j realized as a programmable resistor, and r_i is the self-reaction strength of a

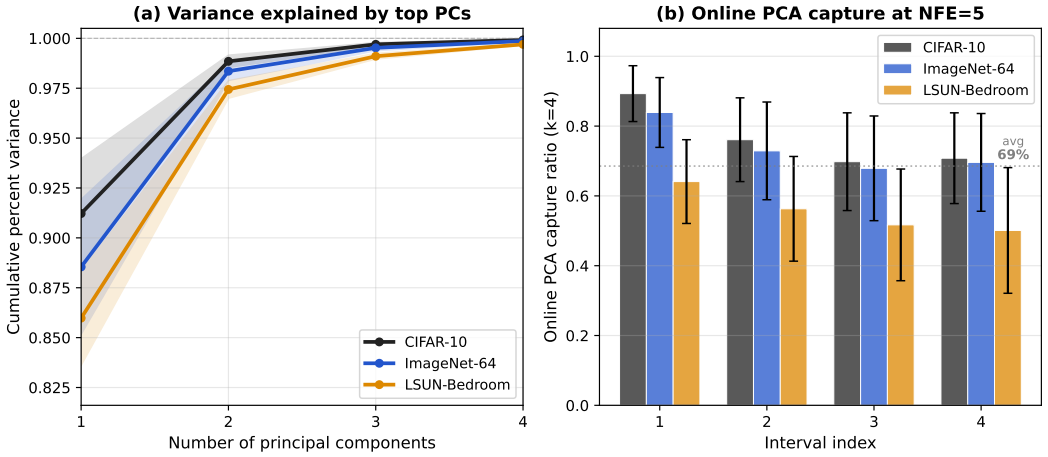


Figure 2: Truncation error analysis. **(a)** 4 PCs explain >99% of trajectory curvature variance. **(b)** Online PCA captures only 56–77% of truncation error at NFE=5, motivating the local DS stream.

variable resistor within each node. The node dynamics are designed as:

$$\frac{\mathrm{d}V_i}{\mathrm{d}\tau} = -\frac{\partial H}{\partial V_i} = \sum_{j \neq i} (J_{ij} + J_{ji})\, V_j \;-\; 2\, r_i\, V_i, \tag{4}$$

which guarantees that the energy monotonically decreases:

$$\frac{\mathrm{d}H}{\mathrm{d}\tau} = \sum_i \frac{\partial H}{\partial V_i}\,\frac{\mathrm{d}V_i}{\mathrm{d}\tau} = -\sum_i \left(\frac{\partial H}{\partial V_i}\right)^2 \leq 0. \tag{5}$$

Eq. (5) guarantees convergence to a stable equilibrium, a process called natural annealing (Wu et al., 2024). To use the system, the parameters J_{ij} and r_i are first trained so that the energy landscape encodes the target input-output relationship, with the desired outputs corresponding to low-energy states. At inference, a subset of nodes is clamped to the input values while the remaining nodes evolve freely according to Eq. (4); once the circuit settles, the equilibrium voltages of designated output nodes provide the prediction.

Building on this foundation, subsequent work has extended DS along multiple directions, including greater expressiveness, faster on-device training, and broader applications (Liu et al., 2025b;a;c). DS-Flow extends this line of work to generative model sampling, embedding a DS corrector within each diffusion sampling step and exploiting the shared ODE structure between diffusion sampling (Eq. 1) and DS dynamics (Eq. 4) for continuous trajectory refinement at zero additional NFE cost.

3 Method: DS-Flow

To overcome the step-level correction limitation identified in Section 2.2, DS-Flow embeds a dual-stream DS corrector within each sampling interval that continuously refines the trajectory through multiple micro-steps without additional denoiser calls.

At a high level (Figure 3), DS-Flow has two components: (1) an algorithmic component, a dual-stream design that decomposes truncation error into a low-dimensional component handled in a PCA subspace and a spatially structured residual handled in pixel space; and (2) a hardware component that executes both streams as analog annealing cycles on a programmable resistor-capacitor network. The algorithm is naturally aligned with the resistive-coupling structure of DS hardware, and the hardware in turn enables the many low-cost micro-step corrections that the algorithm leverages.

The rest of this section presents this design in three parts: motivation through the questions of when and what to correct (Section 3.1), the correction algorithm (Section 3.2), and the hardware realization (Section 3.3); training and sampling procedures are deferred to Appendix B.

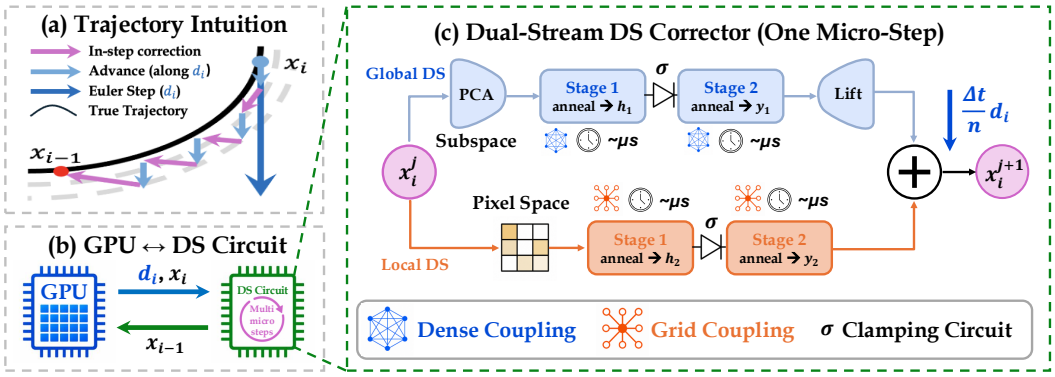


Figure 3: Overview of DS-Flow. **(a)** Micro-steps track the true trajectory within each interval. **(b)** GPU–DS interaction per interval. **(c)** One micro-step: global DS + local DS.

3.1 Motivation

When to correct: continuous in-step refinement. At low NFE ($\leq$10), each step covers a large interval where the true ODE trajectory curves substantially. A one-shot correction that adjusts the direction only once per interval cannot follow this curvature throughout the step. DS-Flow addresses this by performing continuous in-step refinement: within each interval, the DS corrector advances the state through many micro-step corrections, each adapted to the current state. This continuity is uniquely enabled by DS hardware, where each micro-step is a single analog annealing cycle ($\sim$ns latency, $\sim$mW power), so the cost of many corrections per interval remains negligible.

What to correct: dual-stream design. To keep each DS update lightweight, we want to concentrate the correction signal effectively. Prior work (Wang et al., 2025; Zhou et al., 2024) has shown that sampling trajectories concentrate near a low-dimensional subspace: PCA on per-sample trajectories confirms that k=4 components explain >99% of the curvature variance (Figure 2a). This motivates projecting corrections into a compact PCA subspace where even simple coupling can be effective. However, when the basis is estimated online from limited trajectory history, it captures only $\sim$70% of the truncation error on average (Figure 2b; details in Appendix C). The remaining $\sim$30% consists of spatially structured patterns that are not well-captured by a compact subspace. This motivates the dual-stream design: a global DS with dense coupling corrects the dominant low-dimensional component in PCA subspace, while a local DS with grid coupling captures the high-frequency residual directly in pixel space.

3.2 Dual-Stream DS Correction Algorithm

Correction framework. Within solver interval $[t_i, t_{i-1}]$, where i decreases from N to 1 along the time schedule of Section 2.1, DS-Flow divides the coarse Euler step into n micro-steps. Each micro-step advances along the denoising direction $\mathbf{d}_i = (\mathbf{x}_i - D_\theta(\mathbf{x}_i, t_i))/t_i$ produced by the diffusion denoiser, followed by a correction: both streams anneal to equilibrium, producing a fresh correction for the updated state. Let $\mathbf{x}_i^j$ denote the state at the j-th micro-step within interval i, with $\mathbf{x}_i^0 = \mathbf{x}_i$ and $\mathbf{x}_i^n = \mathbf{x}_{i-1}$. The corrected trajectory evolves as a continuous ODE:

$$\frac{\mathrm{d}\mathbf{x}_i^\tau}{\mathrm{d}\tau} = \Delta t \left(\mathbf{d}_i + \mathbf{\Delta}_\theta(\mathbf{x}_i^\tau, \tau, i)\right), \qquad \tau \in [0, 1], \tag{6}$$

where $\tau = (t - t_i)/\Delta t \in [0, 1]$ is the normalized in-interval time, $\Delta t = t_{i-1} - t_i$ is the (signed) solver step, and $\mathbf{\Delta}_\theta$ is the learned correction *direction*. The physical update direction is the bracket $(\mathbf{d}_i + \mathbf{\Delta}_\theta)$, a base direction plus a correction direction, both velocities, while the prefactor $\Delta t = \mathrm{d}t/\mathrm{d}\tau$ merely rescales real time onto the normalized clock τ, so the left-hand side is the physical direction scaled by Δt. Crucially, the ODE itself contains no n: the number of micro-steps enters only through the discretization below. When $\mathbf{\Delta}_\theta = \mathbf{0}$, integrating from τ=0 to τ=1 recovers the standard Euler

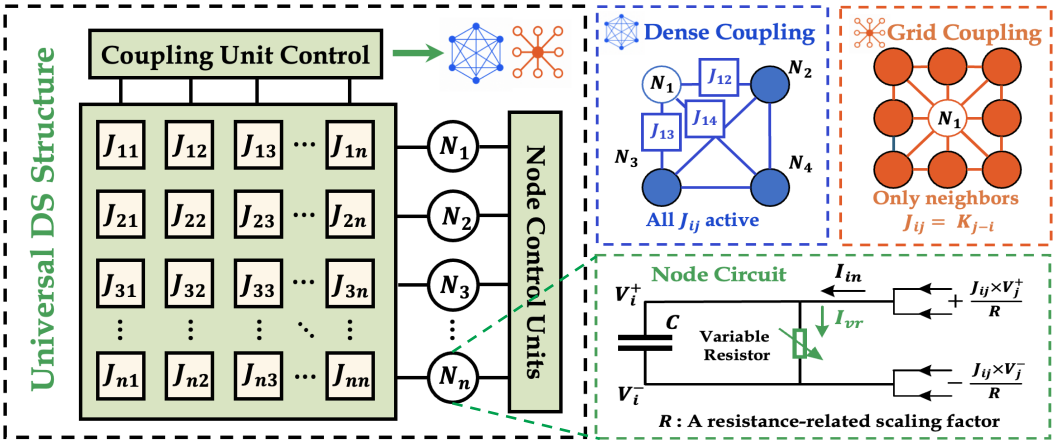


Figure 4: DS hardware architecture. **Left:** Universal DS structure with coupling units and node control. **Top right:** Dense coupling (global DS) vs. grid coupling (local DS, $J_{ij} = K_{j-i}$). **Bottom right:** Node circuit with capacitor C, coupling current I_{in}, and self-reaction current.

step $\mathbf{x}_{i-1} = \mathbf{x}_i + \Delta t\, \mathbf{d}_i$. The correction decomposes into two streams:

$$\boldsymbol{\Delta}_\theta(\mathbf{x}_i^\tau, \tau, i) = \underbrace{\mathbf{U}_i^\top \mathbf{y}_1^*}_{\text{global DS}} + \underbrace{\mathbf{y}_2^*}_{\text{local DS}}, \tag{7}$$

where $\mathbf{U}_i \in \mathbb{R}^{k\times D}$ (k=4) is an online PCA basis (defined below), $\mathbf{y}_1^* \in \mathbb{R}^k$ is the steady-state output of the global DS, and $\mathbf{y}_2^* \in \mathbb{R}^{C\times H\times W}$ is the steady-state output of the local DS. Discretizing Eq. (6) with n equal Euler micro-steps of the normalized clock ($\Delta\tau = 1/n$) yields the micro-step update:

$$\mathbf{x}_i^{j+1} = \mathbf{x}_i^j + \frac{\Delta t}{n}\Big(\mathbf{d}_i + \mathbf{U}_i^\top \mathbf{y}_1^*(\tilde{\mathbf{x}}_i^j) + \mathbf{y}_2^*(\hat{\mathbf{x}}_i^j)\Big), \qquad \tau_j = \tfrac{j}{n}, \quad j = 0, \ldots, n{-}1, \tag{8}$$

Here, $\tilde{\mathbf{x}}_i^j$ and $\hat{\mathbf{x}}_i^j$ denote the inputs to the global and local streams (defined in Eqs. (9) and (12)); both depend on the current state $\mathbf{x}_i^j$, so each annealing cycle produces a different correction adapted to the up-to-date state, the continuity advantage of DS-Flow over one-shot methods. All output couplings are zero-initialized, so training starts from the exact Euler solver. The total parameter count of both streams is $\sim$0.1% of the pretrained denoiser.

Global DS. The global stream operates in the k-dimensional PCA subspace. Following PAS (Wang et al., 2025), at each step i we construct a trajectory matrix from the direction history $\mathbf{X} = [\mathbf{d}_N, \ldots, \mathbf{d}_i]^\top \in \mathbb{R}^{(N-i+1)\times D}$, set the leading basis vector to $\mathbf{u}_1 = \mathbf{d}_i/\|\mathbf{d}_i\|$, and obtain the remaining vectors via Gram–Schmidt, yielding $\mathbf{U}_i = [\mathbf{u}_1, \ldots, \mathbf{u}_k]^\top \in \mathbb{R}^{k\times D}$. This estimation is per-sample and training-free. Whereas PAS applies a single corrected direction per step, DS-Flow uses $\mathbf{U}_i$ as projection and lift operators for a DS that continuously refines the state over multiple micro-steps. The input to the global DS concatenates projected features:

$$\tilde{\mathbf{x}}_i^j = \big[\, \mathbf{U}_i \mathbf{x}_i^j;\ \mathbf{U}_i \mathbf{d}_i;\ \mathbf{U}_i \hat{\mathbf{d}}_{i-1};\ \mathbf{e}_\tau;\ \mathbf{e}_i \,\big] \in \mathbb{R}^{d_{\text{in}}}, \tag{9}$$

where $\hat{\mathbf{d}}_{i-1}$ is a predicted next direction obtained by Adams–Bashforth extrapolation from $\{\mathbf{d}_N, \ldots, \mathbf{d}_i\}$, $\mathbf{e}_\tau$ is a sinusoidal encoding of $\tau_j = j/n$, and $\mathbf{e}_i$ is a learned interval embedding. The two-stage cascade evolves as:

$$\frac{\mathrm{d}\mathbf{h}_1}{\mathrm{d}\tau} = \mathbf{J}_1\, \tilde{\mathbf{x}} - r_h\, \mathbf{h}_1, \tag{10}$$

$$\frac{\mathrm{d}\mathbf{y}_1}{\mathrm{d}\tau} = \mathbf{J}_2\, \sigma(\mathbf{h}_1^*) - r_y\, \mathbf{y}_1, \tag{11}$$

where d_h is the hidden dimension; $\mathbf{J}_1 \in \mathbb{R}^{d_h\times d_{\text{in}}}$ and $\mathbf{J}_2 \in \mathbb{R}^{k\times d_h}$ are dense coupling parameters; r_h and r_y are self-reaction coefficients; and σ is a clamping nonlinearity applied between stages (Liu et al., 2025b). Stage 1 first settles to $\mathbf{h}_1^*$; the clamping circuit applies $\sigma(\mathbf{h}_1^*)$, which is then used as the input to Stage 2. The correction is lifted back to pixel space via $\mathbf{U}_i^\top \mathbf{y}_1^*$.

Local DS. The local stream operates directly in pixel space $\mathbb{R}^{C\times H\times W}$, targeting the spatially structured residual that the linear lift $\mathbf{U}_i^\top$ cannot represent. The input concatenates the current state, denoising direction, and a broadcasted interval embedding:

$$\hat{\mathbf{x}}_i^j = \left[\, \mathbf{x}_i^j;\ \mathbf{d}_i;\ \mathbf{e}_i \,\right] \in \mathbb{R}^{(2C+1)\times H\times W}. \tag{12}$$

The two-stage cascade evolves as:

$$\frac{\mathrm{d}\mathbf{h}_2}{\mathrm{d}\tau} = \mathbf{K}_1 * \hat{\mathbf{x}} - r_h\, \mathbf{h}_2, \tag{13}$$

$$\frac{\mathrm{d}\mathbf{y}_2}{\mathrm{d}\tau} = \mathbf{K}_2 * \sigma(\mathbf{h}_2^*) - r_y\, \mathbf{y}_2, \tag{14}$$

where $*$ denotes the spatial coupling operation (each node interacts with its 3×3 neighbors), and $\mathbf{K}_1$, $\mathbf{K}_2$ are shared grid coupling kernels (Wu et al., 2024; 2025); r_h and r_y here are stream-specific and learned independently of the global stream. This sparse, local connectivity captures spatially structured patterns such as textures, edges, and local contrast that the global stream cannot express. In practice, additional intermediate coupling stages with clamping nonlinearity are inserted between the input and output to increase capacity (Appendix F).

3.3 Hardware Realization

System architecture. DS-Flow operates as a hybrid GPU-DS system. At each solver interval, the GPU evaluates the denoiser to produce $\mathbf{d}_i$ (1 NFE), and the DS circuit autonomously performs the n in-step corrections in the analog domain. The trained coupling parameters $\{\mathbf{J}, \mathbf{K}\}$ are programmed into the resistive network once before sampling begins, and only a single DAC/ADC round-trip per interval is needed regardless of n. This separation places the computationally expensive denoiser on GPU while the lightweight but frequent micro-step corrections run on energy-efficient DS hardware.

DS circuit. The DS circuit is a programmable resistor-capacitor network (Figure 4), following the architecture established by BRIM (Afoakwa et al., 2021), NP-GL (Wu et al., 2024), and EADS (Liu et al., 2025b). As described in Section 2.3, each node stores its state as a voltage V_i on a capacitor C, while coupling units with conductance J_{ij} and self-reaction resistors with conductance r_i realize the coupling and self-reaction terms, respectively. By Kirchhoff's current law, the voltage evolves as:

$$C\,\frac{\mathrm{d}V_i}{\mathrm{d}\tau} = \underbrace{\sum\nolimits_j J_{ij}\, V_j}_{I_{\mathrm{in}}\text{: coupling}} - \underbrace{r_i\, V_i}_{I_{\mathrm{vr}}\text{: self-reaction}}, \tag{15}$$

which, after absorbing C into the time constant, recovers the node dynamics of Eq. (4). The two streams share the same node circuit but differ in coupling topology. In the global stream, dense coupling activates all J_{ij} as independently programmable parameters, realizing all-to-all connectivity. In the local stream, grid coupling restricts each node to its nearby neighbors with translation-invariant weights $J_{ij} = K_{j-i}$, where K is a small shared kernel and $j - i$ denotes the offset between two nodes; couplings beyond the kernel range are simply absent ($J_{ij} = 0$), drastically reducing the number of programmable resistors.

Inference flow. At each interval i, the GPU evaluates $D_\theta(\mathbf{x}_i, t_i)$ to obtain $\mathbf{d}_i$, computes the online PCA basis $\mathbf{U}_i$ via Gram–Schmidt, and transfers them to the DS circuit through DAC. The DS circuit then executes n micro-steps. At each micro-step j, both streams use the same input state $\mathbf{x}_i^j$ but are executed sequentially on hardware, with one annealing cycle each: (1) clamp input nodes to the current driving signal ($\tilde{\mathbf{x}}_i^j$ for global DS, $\hat{\mathbf{x}}_i^j$ for local DS); (2) let the circuit anneal to steady-state $\mathbf{V}^*$ ($\sim$ns per annealing cycle); (3) read out $\mathbf{V}^*$ as the correction $\mathbf{y}^*$. Because the state $\mathbf{x}$ is stored as capacitor voltages and all operations, namely bias current injection $\frac{\Delta t}{n}\mathbf{d}_i$, DS annealing $\mathbf{y}_1^*, \mathbf{y}_2^*$, and voltage summation, are performed natively in the analog domain, the entire n-step loop runs without intermediate digital conversion. After the final micro-step, a single ADC reads $\mathbf{x}_{i-1}$ back to the GPU for the next interval.

Table 1: FID↓ across three datasets at NFE $\in \{5, 6, 8, 10\}$. All methods use EDM pretrained models. †iPNDM base solver. ‡AFS (Analytical First Step). Best overall in **bold**; best without AFS underlined. DS-Flow uses only the first-order Euler solver.

	CIFAR-10 32×32				**ImageNet** 64×64				**LSUN Bedroom** 256×256			
Method	5	6	8	10	5	6	8	10	5	6	8	10
Training-free solvers												
DDIM	49.66	35.62	22.32	15.69	43.81	34.03	22.59	16.72	34.34	25.25	15.71	11.42
DPM-Solver++(3M)	24.97	11.99	4.54	3.00	25.49	15.06	7.84	5.67	23.15	12.28	7.44	5.71
DEIS-tAB3	14.39	9.40	5.55	4.09	14.75	12.57	6.84	5.34	16.31	11.75	7.00	5.18
UniPC	23.98	11.14	3.99	2.89	24.36	14.30	7.52	5.53	23.34	11.71	7.53	5.75
iPNDM	13.59	7.05	3.69	2.77	18.99	12.92	7.20	5.11	26.65	20.73	11.78	5.57
Learned correction												
GITS†	8.38	4.88	3.24	2.49	10.79	8.43	5.82	4.48	15.85	10.41	7.31	5.28
PAS†	13.61	7.47	3.87	2.84	23.33	12.89	7.27	5.32	15.48	10.24	6.67	5.14
AMED-Solver†	–	7.04	5.56	4.14	–	10.63	7.71	6.06	–	12.75	6.95	5.38
AMED-Plugin†‡	6.61	6.67	3.34	2.48	10.74	12.05	7.03	5.01	13.20	11.58	7.48	5.70
DS-Flow (Euler base solver)												
Ours	7.95	4.59	2.70	2.56	17.03	8.41	5.09	3.62	14.90	8.13	6.23	5.63
Ours-AFS‡	**4.55**	**3.23**	**2.57**	**2.39**	**10.67**	**7.60**	**4.33**	**3.48**	**12.14**	**7.11**	**5.42**	**4.99**

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Datasets and evaluation. We adopt the EDM framework (Karras et al., 2022) as a unified setup and evaluate DS-Flow on three standard benchmarks: **CIFAR-10** 32×32 (unconditional), **ImageNet** 64×64 (class-conditional, 1000 classes), and **LSUN Bedroom** 256×256 (unconditional), using the EDM polynomial time schedule (Appendix A). We report Fréchet Inception Distance (FID) (Heusel et al., 2017) over 50,000 generated samples relative to the official EDM reference statistics, at NFE counts of 5, 6, 8, and 10.

Baselines and DS-Flow variants. We benchmark against two families of baselines. Training-free ODE solvers include DDIM (Song et al., 2021a), DPM-Solver++(3M) (Lu et al., 2022b), DEIS-tAB3 (Zhang & Chen, 2023), UniPC (Zhao et al., 2023), and iPNDM (Liu et al., 2022; Zhang & Chen, 2023). Learned-correction methods include GITS (Chen et al., 2024), PAS (Wang et al., 2025), and AMED-Solver and AMED-Plugin (Zhou et al., 2024). All baselines share the same pretrained EDM denoisers for a controlled comparison. We evaluate two variants, **Ours** (no AFS) and **Ours-AFS** (with the Analytical First Step; Appendix A), both on the first-order Euler base solver. The corrector adds ∼58K parameters ($< 0.1\%$ of the denoiser); full training details are in Appendix B.

Hardware evaluation protocol. Our reported DS-hardware efficiency numbers for latency, power, and energy in Section 4.3 come from *circuit-level simulation* of the dynamical-system substrate, using Cadence mixed-signal simulation in a 45 nm process and following the pre-silicon evaluation protocol of prior DS hardware (Wu et al., 2024; Liu et al., 2025b). The simulation is also not idealized. It runs at 8-bit precision and injects device non-idealities such as mismatch and thermal noise at the typical-typical process corner and $25\,^{\circ}\mathrm{C}$, so the reported quality already reflects a realistic circuit. Appendix G reports the full bit-width and thermal-noise robustness sweeps. Timing and power constants are taken from prior DS hardware, namely the fabricated BRIM Ising machine (Afoakwa et al., 2021) and the simulated NP-GL and EADS architectures (Wu et al., 2024; Liu et al., 2025b). GPU baselines are measured on a single NVIDIA A100.

4.2 MAIN RESULTS

Table 1 reports FID across the three benchmarks at NFE $\in \{5, 6, 8, 10\}$. To isolate the contribution of our learned corrector from the orthogonal Analytical First Step (AFS) trick, we mark two best

results: **bold** indicates the overall best, while underline indicates the best among methods that do not use AFS.

Without AFS, DS-Flow attains the best FID in 9 of 12 configurations despite relying solely on the first-order Euler solver, whereas all learned-correction baselines build on the higher-order iPNDM. With AFS enabled, DS-Flow-AFS achieves the best FID across all 12 configurations, including a 31% FID reduction over the AFS-enabled AMED-Plugin on CIFAR-10 at NFE=5.

Qualitative comparisons at NFE=5 and the visual effect of the micro-step count n are shown in Appendix E.

We ablate two key design choices, with full tables in Appendix D. Removing the local stream costs up to 23% FID at NFE=5 but is negligible at NFE$\geq$8, and FID improves monotonically with the micro-step count n until it saturates near n=50, confirming that the gains stem from continuous in-step refinement rather than one-shot correction.

4.3 Hardware Efficiency

We evaluate the per-interval latency and energy consumption of the DS corrector against existing learned-correction methods. The DS corrector builds on the BRIM circuit architecture (Afoakwa et al., 2021) with $\sim$52K trainable coupling parameters operating at $\sim$130 mW; the GPU baselines (PAS, AMED, GITS) are measured on an NVIDIA A100 GPU using CUDA event timing averaged over 200 runs, with GPU power at $\sim$247 W.

Figure 5 shows that the DS corrector exhibits remarkable hardware efficiency. Specifically, it completes each interval in $\sim$1.4 μs, delivering a 6–7$\times$ latency speedup over PAS, AMED, and GITS, despite executing a far more complex correction with dual-stream, 50 micro-steps, and $\sim$58K parameters versus the baselines' $\sim$10 parameters. This $\sim$1.4 μs already includes data conversion, which occurs once per interval rather than per micro-step and therefore does not scale with n; Appendix G gives the full DAC, annealing, and ADC breakdown. Furthermore, operating at $\sim$130 mW, about 1,900$\times$ lower than the A100 GPU's $\sim$247 W, and combined with the latency advantage, the corrector yields a total energy reduction of $\sim 10^4\times$ per correction interval. Across an entire sampling run, the correction pipeline consumes only $\sim$0.9 μJ, negligible compared to the GPU denoiser energy budget of 28–110 J.

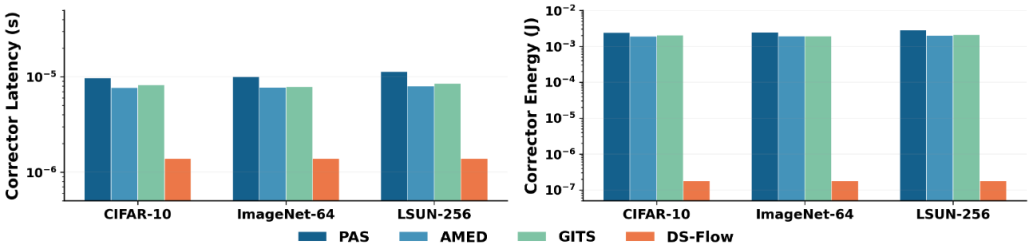


Figure 5: Per-interval corrector latency and energy. GPU baselines measured on A100; DS-Flow on DS hardware.

5 Conclusion

We introduced DS-Flow, the first method to bridge electronic dynamical systems with diffusion sampling. DS-Flow embeds a dual-stream DS corrector within each sampling step that performs continuous in-step refinement through multiple analog annealing cycles, replacing the one-shot correction of prior methods at zero additional NFE cost. Across three standard benchmarks at 5–10 NFE, DS-Flow attains the best FID in 9 of 12 settings among training-free solvers and learned-correction baselines while adding only $\sim$0.1% parameters; on DS hardware, the corrector achieves orders-of-magnitude energy reduction over GPU-based alternatives at near-real-time latency per interval. Promising directions for future work include extending DS-Flow to latent diffusion (Rombach et al., 2022) and flow matching (Lipman et al., 2023; Esser et al., 2024) frameworks, and scaling up DS hardware to higher node counts for larger images.

Ethics Statement

This work studies a sampling-acceleration technique for diffusion models and does not involve human subjects, private data, or personally identifiable information. All experiments use standard, publicly available benchmarks (CIFAR-10, ImageNet, and LSUN Bedroom) under their intended research use. DS-Flow accelerates the inference of existing pretrained generative models and does not introduce new generative capabilities; consequently, it inherits, but does not enlarge, the misuse risks of the underlying models, such as the generation of deepfakes, misinformation, or non-consensual synthetic content. We note the positive impact of lower-energy inference, and we encourage responsible deployment practices such as content-provenance markers and use-case-specific safeguards.

Reproducibility Statement

All generative-quality results in this paper are produced entirely in software and can be reproduced on standard GPUs. We build on the publicly available EDM framework and its released pretrained denoisers, and evaluate on three public benchmarks (CIFAR-10 32×32, ImageNet 64×64, and LSUN Bedroom 256×256), reporting FID over 50,000 samples against the official EDM reference statistics (Section 4.1). The dual-stream corrector architecture and the complete training and sampling procedures, namely teacher-trajectory generation, the L2+LPIPS objective, optimizer, schedule, and all hyperparameters, are specified in Appendix B with pseudocode, and the PCA analysis is detailed in Appendix C. All baselines use the same pretrained denoisers to ensure a controlled comparison. We will release our code and trained corrector parameters upon publication.

The hardware-efficiency results (latency, power, and energy in Section 4.3) are obtained from circuit-level modeling of the electronic dynamical-system (DS) substrate, following the established BRIM (Afoakwa et al., 2021), NP-GL (Wu et al., 2024), and EADS (Liu et al., 2025b) circuit architectures; given the reported device parameters, they are reproducible at the same circuit-simulation level. We emphasize that the DS corrector is evaluated in simulation and is *not* fabricated: physically reproducing the reported latency and energy would require manufacturing the corresponding DS chip, which is beyond software reproduction and is left to future work.

The Use of Large Language Models in This Work

Large language models were used in two ways under the authors' direction. First, to aid and polish the writing of the manuscript. Second, for retrieval and discovery, such as helping to find related work during the literature review. All AI-assisted text and any surfaced references were reviewed, verified, and edited by the authors. All reported measurements originate from executed experiments rather than from model estimates or predictions, and the authors take full responsibility for the experimental design, the verification of results, and all claims and artifacts in this paper.

References

Richard Afoakwa, Yiqiao Zhang, Uday Kumar Reddy Vengalam, Zeljko Ignjatovic, and Michael C. Huang. BRIM: Bistable resistively-coupled Ising machine. In *HPCA*, 2021.

Brian D. O. Anderson. Reverse-time diffusion equation models. *Stochastic Processes and Their Applications*, 12(3):313–326, 1982.

Defang Chen, Zhenyu Zhou, Can Wang, Chunhua Shen, and Siwei Lyu. On the trajectory regularity of ODE-based diffusion sampling. In *ICML*, 2024.

Prafulla Dhariwal and Alex Nichol. Diffusion models beat GANs on image synthesis. In *NeurIPS*, 2021.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *ICML*, 2024.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, 2014.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In *NeurIPS*, 2017.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020.

Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J. Fleet. Video diffusion models. In *NeurIPS*, 2022.

Takahiro Inagaki, Yoshitaka Haribara, Koji Igarashi, Tomohiro Sonobe, Shuhei Tamate, Toshimori Honjo, Alireza Marandi, Peter L McMahon, Takeshi Umeki, Koji Enbutsu, et al. A coherent Ising machine for 2000-node optimization problems. *Science*, 354(6312):603–606, 2016.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *NeurIPS*, 2022.

Diederik P. Kingma and Max Welling. Auto-encoding variational Bayes. In *ICLR*, 2014.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *ICLR*, 2023.

Chuan Liu, Ruibing Song, Chunshu Wu, Pouya Haghi, and Tong Geng. InstaTrain: Adaptive training via ultra-fast natural annealing within dynamical systems. In *ICLR*, 2025a.

Chuan Liu, Chunshu Wu, Ruibing Song, Ang Li, Ying Nian Wu, and Tong Geng. An expressive and self-adaptive dynamical system for efficient function learning. In *ICML*, 2025b.

Chuan Liu, Chunshu Wu, Ruibing Song, Guangyan Sun, Ying Nian Wu, Yousu Chen, Ang Li, and Tong Geng. DS-TIDE: Harnessing dynamical systems for efficient time-independent differential equation solving. In *MICRO*, 2025c.

Luping Liu, Yi Ren, Zhijie Lin, and Zhou Zhao. Pseudo numerical methods for diffusion models on manifolds. In *ICLR*, 2022.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *ICLR*, 2023.

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. In *NeurIPS*, 2022a.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. DPM-Solver++: Fast solver for guided sampling of diffusion probabilistic models. *arXiv preprint arXiv:2211.01095*, 2022b.

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*, 2023.

Chenlin Meng, Robin Rombach, Ruiqi Gao, Diederik Kingma, Stefano Ermon, Jonathan Ho, and Tim Salimans. On distillation of guided diffusion models. In *CVPR*, 2023.

Naeimeh Mohseni, Peter L. McMahon, and Tim Byrnes. Ising machines as hardware solvers of combinatorial optimization problems. *Nature Reviews Physics*, 4(6):363–379, 2022.

Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *ICML*, 2021.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *ICCV*, 2023.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022.

Amirmojtaba Sabour, Sanja Fidler, and Karsten Kreis. Align your steps: Optimizing sampling schedules in diffusion models. In *ICML*, 2024.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *ICLR*, 2022.

Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *ICML*, 2015.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *ICLR*, 2021a.

Yang Song and Prafulla Dhariwal. Improved techniques for training consistency models. In *ICLR*, 2024.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *NeurIPS*, 2019.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *ICLR*, 2021b.

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

Guangyi Wang, Wei Peng, Lijiang Li, Wenyu Chen, Yuren Cai, and Song-Zhi Su. Diffusion sampling correction via approximately 10 parameters. In *ICML*, 2025.

Chunshu Wu, Ruibing Song, Chuan Liu, Yunan Yang, Ang Li, Michael Huang, and Tong Geng. Extending power of nature from binary to real-valued graph learning in real world. In *ICLR*, 2024.

Chunshu Wu, Ruibing Song, Chuan Liu, Pouya Haghi, Ang Li, Michael Huang, and Tong Geng. DS-TPU: Dynamical system for on-device lifelong graph learning with nonlinear node interaction. In *ISCA*, 2025.

Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Frédo Durand, William T Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. In *CVPR*, 2024.

Qinsheng Zhang and Yongxin Chen. Fast sampling of diffusion models with exponential integrator. In *ICLR*, 2023.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

Wenliang Zhao, Lujia Bai, Yongming Rao, Jie Zhou, and Jiwen Lu. UniPC: A unified predictor-corrector framework for fast sampling of diffusion models. In *NeurIPS*, 2023.

Jianbin Zheng, Minghui Hu, Zhongyi Fan, Chaoyue Wang, Changxing Ding, Dacheng Tao, and Tat-Jen Cham. Trajectory consistency distillation. *arXiv preprint arXiv:2402.19159*, 2024.

Kaiwen Zheng, Cheng Lu, Jianfei Chen, and Jun Zhu. DPM-Solver-v3: Improved diffusion ODE solver with empirical model statistics. In *NeurIPS*, 2023.

Zhenyu Zhou, Defang Chen, Can Wang, and Chun Chen. Fast ODE-based sampling for diffusion models in around 5 steps. In *CVPR*, pp. 7777–7786, 2024.

A Additional Experimental Details

Time schedule. The temporal domain is segmented using the EDM polynomial schedule

$$t_n = \left(t_0^{1/\rho} + \frac{n}{N}\left(t_N^{1/\rho} - t_0^{1/\rho}\right)\right)^{\rho}, \tag{16}$$

with $t_0 = 0.002$, $t_N = 80$, $n \in [0, N]$, and $\rho = 7$.

Analytical first step (AFS). The **Ours-AFS** variant applies the Analytical First Step (Karras et al., 2022), which approximates $\mathbf{d}_N \approx \mathbf{x}_N/t_N$ to save one denoiser call. AFS is orthogonal to our corrector and is reported separately from the core **Ours** results.

B Training and Sampling

Teacher trajectory generation. For each dataset and NFE configuration, we generate teacher trajectories using a 50-step Euler solver from 5,000 random noise samples (CIFAR-10, ImageNet-64) or 1,000 samples (LSUN Bedroom). Within each student interval $[t_i, t_{i-1}]$, we record 10 evenly-spaced checkpoint states from the teacher trajectory, providing dense supervision for the n=50 micro-step corrector.

Training procedure. Both streams are trained jointly on GPU (Algorithm 1) to match the teacher states using an L2+LPIPS loss (Zhang et al., 2018):

$$\mathcal{L} = \frac{1}{|\mathcal{C}|}\sum_{c\in\mathcal{C}}\left[\left\|\mathbf{x}_c - \mathbf{x}_c^{\text{tea}}\right\|_2^2 + \lambda \cdot \text{LPIPS}\left(\mathbf{x}_c,\, \mathbf{x}_c^{\text{tea}}\right)\right], \tag{17}$$

where $\mathbf{x}_c$ and $\mathbf{x}_c^{\text{tea}}$ are predicted and teacher states at checkpoint c, and λ=0.1. We use Adam (lr=10^{-3}) with cosine annealing over 80 epochs and early stopping with patience 15. Gradients are accumulated across all intervals before a single optimizer step per batch.

Algorithm 1 DS-Flow Training. Offline on GPU; learns coupling parameters from teacher trajectories.

Require: D_θ, teacher states $\{\mathbf{x}_c^{\text{tea}}\}$, schedule $\{t_i\}_{i=0}^N$, micro-steps n
Ensure: Trained couplings $\{\mathbf{J}, \mathbf{K}\}$

Init output couplings to zero
for each batch $\mathbf{x}_N$ **do**
  $Q \leftarrow \{\mathbf{x}_N\}$
  for $i = N$ to 1 **do**
    $\mathbf{d}_i \leftarrow (\mathbf{x}_i - D_\theta(\mathbf{x}_i, t_i))/t_i$
    $\mathbf{U}_i \leftarrow \text{GramSchmidt}(Q \cup \{\mathbf{d}_i\})$
    for $j = 0$ to $n-1$ **do**
      $\mathbf{x}_i^{j+1} \leftarrow \mathbf{x}_i^j + \frac{\Delta t}{n}\left(\mathbf{d}_i + \mathbf{U}_i^\top \mathbf{y}_1^*(\tilde{\mathbf{x}}_i^j) + \mathbf{y}_2^*(\hat{\mathbf{x}}_i^j)\right)$
    end for
    $\mathbf{x}_{i-1} \leftarrow \mathbf{x}_i^n$; $Q \leftarrow Q \cup \{\mathbf{d}_i\}$
  end for
  Compute $\mathcal{L}$ at checkpoints (Eq. 17)
  Update $\{\mathbf{J}, \mathbf{K}\}$ with $\nabla\mathcal{L}$
end for

Algorithm 2 DS-Flow Sampling. GPU computes $\mathbf{d}_i$ (1 NFE); DS circuit executes n micro-steps per interval.

Require: D_θ, trained couplings $\{\mathbf{J}, \mathbf{K}\}$, schedule $\{t_i\}$, micro-steps n
Ensure: Generated sample $\mathbf{x}_0$

$\mathbf{x}_N \sim \mathcal{N}(\mathbf{0}, t_N^2\mathbf{I})$; $Q \leftarrow \{\mathbf{x}_N\}$
Program $\{\mathbf{J}, \mathbf{K}\}$ into resistive coupling
for $i = N$ to 1 **do**
  $\mathbf{d}_i \leftarrow (\mathbf{x}_i - D_\theta(\mathbf{x}_i, t_i))/t_i$ {1 NFE on GPU}
  $\mathbf{U}_i \leftarrow \text{GramSchmidt}(Q \cup \{\mathbf{d}_i\})$
  for $j = 0$ to $n-1$ **do**
    {on DS hardware}
    Clamp inputs; **anneal** global DS $\rightarrow$ $\mathbf{y}_1^*(\tilde{\mathbf{x}}_i^j)$
    Clamp inputs; **anneal** local DS $\rightarrow$ $\mathbf{y}_2^*(\hat{\mathbf{x}}_i^j)$
    $\mathbf{x}_i^{j+1} \leftarrow \mathbf{x}_i^j + \frac{\Delta t}{n}\left(\mathbf{d}_i + \mathbf{U}_i^\top \mathbf{y}_1^*(\tilde{\mathbf{x}}_i^j) + \mathbf{y}_2^*(\hat{\mathbf{x}}_i^j)\right)$
  end for
  $\mathbf{x}_{i-1} \leftarrow \mathbf{x}_i^n$; $Q \leftarrow Q \cup \{\mathbf{d}_i\}$
end for
return $\mathbf{x}_0$

Hyperparameters. Table 2 summarizes the training hyperparameters. The same architecture and hyperparameters are used across all datasets and configurations.

Table 2: Training hyperparameters.

Hyperparameter	Value
Teacher solver	Euler, 50 steps
Micro-steps n	50
PCA dimension k	4
Global DS hidden dim d_h	128
Local DS intermediate channels	32
Loss	L2 + 0.1 $\times$ LPIPS
Optimizer	Adam
Learning rate	10^{-3} with cosine annealing
Epochs	80 (early stopping, patience 15)
Time schedule	Polynomial (ρ=7)

Sampling. At inference, each interval evaluates D_θ once (1 NFE), estimates $\mathbf{U}_i$ via Gram–Schmidt on the direction history buffer Q, and runs n micro-steps (Eq. 8); the full procedure is given in Algorithm 2. Optionally, an analytical first step (AFS) (Karras et al., 2022) at the maximum noise level t_N computes $\mathbf{d}_N = \mathbf{x}_N / t_N$ at zero NFE cost, providing one extra correction interval within the same NFE budget. With AFS, a model trained for NFE=5 uses 6 intervals: 1 analytical step plus 5 denoiser calls.

C PCA Analysis Details

Panel (a): Trajectory dimensionality. To verify that sampling trajectories are low-dimensional, we generate 500 (CIFAR-10), 200 (ImageNet-64), and 100 (LSUN Bedroom) trajectories using a 100-NFE Heun solver (2nd-order, 50 actual steps). For each trajectory $\{\mathbf{x}_{t_0}, \ldots, \mathbf{x}_{t_N}\}$, we extract the primary direction $\mathbf{v} = (\mathbf{x}_{t_0} - \mathbf{x}_{t_N})/\|\mathbf{x}_{t_0} - \mathbf{x}_{t_N}\|$ and project all states onto the subspace perpendicular to $\mathbf{v}$: $\mathbf{x}_t^\perp = \mathbf{x}_t - (\mathbf{x}_t \cdot \mathbf{v})\,\mathbf{v}$. This removes the trivial scale variation (the dominant straight-line component from noise to data) and isolates the trajectory's curvature. PCA on the projected states $\{\mathbf{x}_t^\perp\}$ confirms that k=4 components explain >99% of the curvature variance across all three datasets (Figure 2a), following the methodology of AMED (Zhou et al., 2024) and GITS (Chen et al., 2024).

Panel (b): Online PCA capture ratio. To quantify how much truncation error the online PCA basis captures at inference, we run both a 5-step Euler solver (student) and a 100-NFE Heun solver (teacher) from the same initial noise. At each student interval i, we compute the online basis $\mathbf{U}_i$ using the actual `compute_pca_basis` function from our codebase (Gram–Schmidt on the direction history buffer). The truncation error is $\mathbf{e} = \mathbf{x}_{i-1}^{\text{tea}} - \mathbf{x}_{i-1}^{\text{euler}}$, and the capture ratio is:

$$\rho_i = \frac{\|\mathbf{U}_i^\top (\mathbf{U}_i\, \mathbf{e})\|^2}{\|\mathbf{e}\|^2}. \tag{18}$$

We average ρ_i over 500 (CIFAR-10), 200 (ImageNet-64), and 100 (LSUN Bedroom) samples. The results (Figure 2b) show that the online basis captures ∼77% on CIFAR-10, ∼74% on ImageNet-64, and only ∼56% on LSUN Bedroom, with the gap most pronounced at early intervals where the direction history is sparse.

D Ablation Study Details

Global-only vs. dual-stream (Table 3). We compare the full dual-stream model with a global-only variant that removes the local DS stream entirely (setting $\mathbf{y}_2^* = \mathbf{0}$). Both variants are trained from scratch with the same hyperparameters. The comparison spans both an unconditional dataset (CIFAR-10, 32×32) and a class-conditional dataset (ImageNet-64, 64×64) under the AFS configuration across all four NFE budgets. As shown in Table 3, the local stream provides up to 23% FID improvement at

NFE=5 but contributes negligibly at NFE≥8, confirming that the spatially structured residual is most significant when truncation error is large.

Effect of micro-step count (Table 4). We evaluate the effect of the number of micro-steps n by training a single model with n=50 and running inference with different n values at test time (the checkpoint is shared). This isolates the effect of correction granularity from training differences. Results across all three datasets (noAFS, NFE=5) are shown in Table 4. The Euler baselines (no correction) are: CIFAR-10 49.66, ImageNet-64 43.81, LSUN 34.34.

Table 3: FID↓ comparison of global-only vs. dual-stream on CIFAR-10 (unconditional) and ImageNet-64 (class-conditional), AFS configuration.

	CIFAR-10				**ImageNet-64**			
Variant	5	6	8	10	5	6	8	10
Global only	5.90	3.75	2.60	2.40	13.76	**7.39**	**4.30**	**3.43**
Dual-stream	**4.55**	**3.23**	**2.57**	**2.39**	**10.67**	7.60	4.33	3.48
Δ	−23%	−14%	−1%	0%	−23%	+3%	+1%	+1%

Table 4: Effect of micro-step count n on FID (noAFS, NFE=5). Euler baseline (no correction): CIFAR-10 49.66, ImageNet-64 43.81, LSUN 34.34.

n	1	2	5	10	20	40	50
CIFAR-10	46.77	44.95	39.50	30.22	18.13	8.30	**7.95**
ImageNet-64	43.72	43.18	39.72	34.79	26.10	17.35	**17.03**
LSUN 256	32.87	31.45	27.83	23.41	19.06	15.27	**14.90**

E Qualitative Results

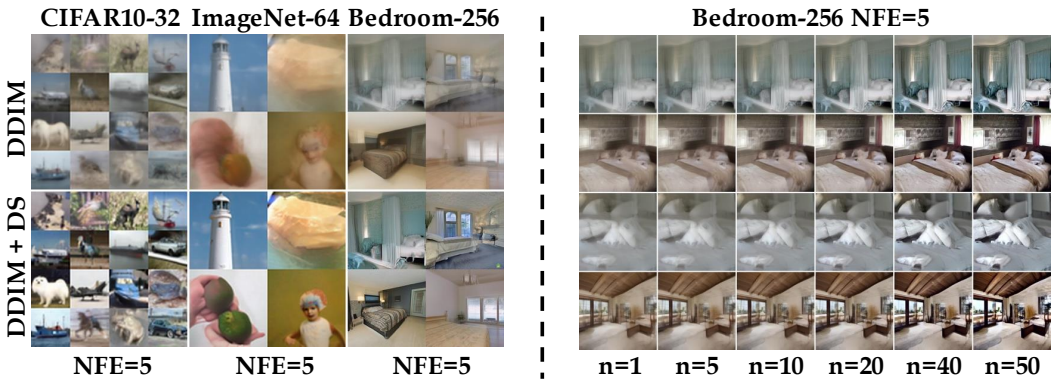


Figure 6: **Left**: DDIM vs. DS-Flow at NFE=5. **Right**: Effect of micro-step count n on LSUN Bedroom (NFE=5); quality improves from n=1 to n=50.

Qualitative results in Figure 6 corroborate these gains. At NFE=5, DDIM samples appear blurry and structurally inconsistent, whereas DS-Flow recovers sharp images with coherent global layout and rich local detail (left). The right panel further visualizes how DS-Flow's sample quality on LSUN Bedroom progressively improves as the micro-step count n increases from 1 to 50, providing visual evidence for the micro-step ablation quantified in Appendix D.

Table 5: FID on CIFAR-10 at NFE=5 without AFS, as a function of converter bit-width.

Precision	fp32 (ideal)	12-bit	10-bit	8-bit	6-bit	4-bit
FID	7.41	7.54	7.59	7.61	11.43	25.69

F Architecture Details

The main text presents a two-stage cascade for clarity. In practice, we use deeper cascades with additional intermediate coupling stages and clamping nonlinearities to increase expressivity, following the multi-stage pipeline of EADS (Liu et al., 2025b). The full cascade dynamics for each stream are given below. All coupling parameters are trained offline on GPU; at deployment, they are programmed into the resistive network once. The total corrector has $\sim$52K coupling parameters ($\sim$31K global $+$ $\sim$21K local); including interval embeddings and self-reaction coefficients, the full corrector has $\sim$58K parameters ($<$0.1% of the pretrained denoiser).

Global DS (dense coupling). The global stream implements a 3-stage cascade in the k-dimensional PCA subspace. The input $\tilde{\mathbf{x}} \in \mathbb{R}^{d_{\text{in}}}$ ($d_{\text{in}} = 108$ for k=4) drives the first stage:

$$\begin{aligned}
\text{Stage 1:} \quad & \tfrac{\mathrm{d}\mathbf{h}^{(1)}}{\mathrm{d}\tau} = \mathbf{J}^{(1)}\,\tilde{\mathbf{x}} - r\,\mathbf{h}^{(1)}, \quad \mathbf{J}^{(1)} \in \mathbb{R}^{128\times 108}, \\
\text{Stage 2:} \quad & \tfrac{\mathrm{d}\mathbf{h}^{(2)}}{\mathrm{d}\tau} = \mathbf{J}^{(2)}\,\sigma(\mathbf{h}^{(1)*}) - r\,\mathbf{h}^{(2)}, \quad \mathbf{J}^{(2)} \in \mathbb{R}^{128\times 128}, \\
\text{Stage 3:} \quad & \tfrac{\mathrm{d}\mathbf{y}_1}{\mathrm{d}\tau} = \mathbf{J}^{(3)}\,\sigma(\mathbf{h}^{(2)*}) - r\,\mathbf{y}_1, \quad \mathbf{J}^{(3)} \in \mathbb{R}^{k\times 128},
\end{aligned}$$

where σ is a clamping nonlinearity applied between stages, and the output coupling $\mathbf{J}^{(3)}$ is zero-initialized. Total: $\sim$31K coupling parameters.

Local DS (grid coupling). The local stream implements a 4-stage cascade operating in pixel space $\mathbb{R}^{C\times H\times W}$:

$$\begin{aligned}
\text{Stage 1:} \quad & \tfrac{\mathrm{d}\mathbf{h}^{(1)}}{\mathrm{d}\tau} = \mathbf{K}^{(1)} * \hat{\mathbf{x}} - r\,\mathbf{h}^{(1)}, \quad \mathbf{K}^{(1)} : 3{\times}3,\ (2C{+}1) \to 32, \\
\text{Stage 2:} \quad & \tfrac{\mathrm{d}\mathbf{h}^{(2)}}{\mathrm{d}\tau} = \mathbf{K}^{(2)} * \sigma(\mathbf{h}^{(1)*}) - r\,\mathbf{h}^{(2)}, \quad \mathbf{K}^{(2)} : 3{\times}3,\ 32 \to 32, \\
\text{Stage 3:} \quad & \tfrac{\mathrm{d}\mathbf{h}^{(3)}}{\mathrm{d}\tau} = \mathbf{K}^{(3)} * \sigma(\mathbf{h}^{(2)*}) - r\,\mathbf{h}^{(3)}, \quad \mathbf{K}^{(3)} : 3{\times}3,\ 32 \to 32, \\
\text{Stage 4:} \quad & \tfrac{\mathrm{d}\mathbf{y}_2}{\mathrm{d}\tau} = \mathbf{K}^{(4)} * \sigma(\mathbf{h}^{(3)*}) - r\,\mathbf{y}_2, \quad \mathbf{K}^{(4)} : 3{\times}3,\ 32 \to C,
\end{aligned}$$

where $*$ denotes the $3{\times}3$ grid coupling operation and the output kernel $\mathbf{K}^{(4)}$ is zero-initialized. Total: $\sim$21K coupling parameters. Both streams share the same architecture across all datasets; only the trained coupling parameters differ.

G Hardware Simulation Details

All hardware numbers are obtained in circuit-level simulation; here we detail the per-interval latency and verify robustness to analog non-idealities. Non-ideality results use CIFAR-10 at NFE=5 without AFS and report FID.

Per-interval latency breakdown. The $\sim$1.4 μs per-interval latency comprises three parts, all at 8-bit precision with standard column-parallel converters. The trained coupling $\{\mathbf{J}, \mathbf{K}\}$ is programmed only once before sampling; per interval, a single DAC write loads the current state and the online PCA basis $\mathbf{U}_i$, taking $\approx 0.25\,\mu$s; the 50 micro-steps then relax on-array with no intermediate conversion, taking $\approx 1.0\,\mu$s of annealing; and a single ADC readout takes $\approx 0.15\,\mu$s. These sum to $0.25 + 1.0 + 0.15 = 1.4\,\mu$s. Because conversion is performed once per interval rather than once per micro-step, it does not scale with the number of micro-steps n; on larger images only the DAC and ADC portions grow while the annealing stays fixed, so the total latency remains in the microsecond range.

Converter bit-width. Table 5 sweeps the converter precision. At 8-bit the FID is already near-lossless, only about 0.2 above the fp32 ideal, and higher precision gives marginal further gains, so we adopt 8-bit throughout. Quality degrades sharply only below 6-bit.

Table 6: FID increase on CIFAR-10 at NFE=5 without AFS, as a function of thermal-noise standard deviation (fraction of signal range).

Noise std	$\leq 10\%$	15%	20%
ΔFID	~ 0.2	~ 0.8	~ 2.2

Thermal noise. We inject zero-mean Gaussian thermal noise into the DS state and vary its standard deviation as a fraction of the signal range (Table 6). The corrector is near-immune at moderate noise, with FID rising by only about 0.2 up to 10% noise, because zero-mean perturbations average out over the micro-step sequence. It degrades gracefully to about 0.8 and 2.2 at 15% and 20%.