DIVERGENCE-SUPPRESSING COUPLINGS FOR RECTIFIED FLOW

Anonymous authors
Paper under double-blind review

ABSTRACT

The promise of Rectified Flow rests on producing self-generated couplings whose trajectories are straight, or nearly so. In practice, trajectories generated by the base flow model can bend and intertwine, and the resulting coupling inherits this distortion. In this paper, we identify that such trajectory entanglement is often associated with regions of nonzero divergence in the learned velocity field, where local expansion or contraction distorts trajectories and steers particles away from their ideal endpoints. We then propose divergence-suppressing couplings for Rectified Flow, an offline correction that attenuate the divergent component of the learned velocity during coupling generation. The correction is paid only once per coupling pair and amortized over training, so deployment runs plain Euler at identical wall-clock cost to standard Rectified Flow. Empirically, this offline modification yields consistent improvements on 2D synthetic benchmarks and on image generation.

1 INTRODUCTION

Generative models based on continuous normalizing flows learn to transport a simple source distribution π_0, typically a Gaussian, to a complex target distribution π_1 by integrating an ordinary differential equation $\dot{x} = v_\theta(t, x)$ (Chen et al., 2018; Grathwohl et al., 2018). Training these models through numerical integration can be computationally expensive. Flow matching (FM) replaces solver-in-the-loop training with a regression objective, providing a scalable way to learn the velocity field (Lipman et al., 2023; Liu et al., 2023; Albergo & Vanden-Eijnden, 2023).

For the commonly used linear interpolant $x_t = (1 - t)x_0 + tx_1$, conditional flow matching (CFM) trains the network to predict the displacement $x_1 - x_0$ from the intermediate state and time. The initial source and target samples can be drawn independently, so training requires neither simulated trajectories nor a precomputed transport map. This simulation-free formulation is closely related to diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2020) and stochastic interpolants (Albergo et al., 2025). Flow matching has been applied to image synthesis (Esser et al., 2024), audio (Le et al., 2023), video (Polyak et al., 2024), and molecular design (Bose et al., 2023).

A shared challenge is inference cost. Although training is simulation-free, generation still requires integration of the learned ODE. Curved trajectories can require many neural function evaluations (NFE) for accurate numerical integration. The coupling used for training matters here: with independently paired endpoints, different source–target displacements can give conflicting regression targets near the same intermediate state. The learned marginal field averages these directions, and the resulting ODE trajectories need not follow the straight conditional paths. Reducing NFE without sacrificing generation quality motivates higher-order solvers (Lu et al., 2022; Zhang & Chen, 2022), distillation (Salimans & Ho, 2022; Song et al., 2023), and approaches that learn flows that are easier to integrate.

Rectified Flow (RectFlow) pursues the latter approach by changing the training coupling (Liu et al., 2023; 2024). Starting from a base model trained with independent coupling, it samples sources $x_0 \sim \pi_0$, integrates the model to obtain endpoints x_1, and retrains with the same CFM objective on these self-generated pairs. Repeating this process can produce straighter learned trajectories and support few-step or single-step sampling. The importance of coupling quality also motivates optimal-transport and multisample flow matching (Tong et al., 2023; Pooladian et al., 2023).

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.

The quality of RectFlow's generated coupling depends on both the learned field and the procedure used to generate endpoints. Reflow uses those endpoints as training targets; it does not independently correct errors in their marginal distribution or source–target assignments. Additional rounds may therefore provide limited gains when the self-generated pairs remain poor. This makes *offline coupling generation* a useful point of intervention. The pairs are cached and reused during training, so additional work in constructing them can benefit subsequent inference calls without adding work to the deployed sampler.

We introduce *DS-RectFlow*, an offline method that uses divergence as a local criterion for improving these couplings. Divergence measures instantaneous local volume change and supplies a computable proxy for compressibility. In our image implementation, each corrected solver step searches a small neighborhood and advances from a state with smaller estimated absolute divergence. The network remains fixed during pair generation; the correction changes the states at which its velocity is evaluated. The next model is trained with the standard CFM loss, architecture, and optimizer, and uses plain Euler at inference with no additional computation at a fixed NFE.

We evaluate this approach on Checkerboard and GMM using a related low-dimensional correction, and on CIFAR-10 and CelebA-64 using neighborhood search. The 2D experiments show improved single-step results after two reflow rounds. On images, DS-RectFlow improves FID at every reported inference budget with both Euler and RK45 pair generation. With Euler-generated pairs, single-step FID improves in mean FID across seeds from 17.276 to 12.272 on CIFAR-10 and from 6.737 to 5.826 on CelebA-64. These results support improving offline coupling quality as a practical route to low-NFE generation, without requiring divergence to explain all trajectory curvature or coupling ambiguity.

2 Background

2.1 Flow Matching

Flow matching (FM) learns a time-dependent velocity field $v_\theta : [0,1]\times\mathbb{R}^d \to \mathbb{R}^d$ such that integrating the ODE $\dot{x} = v_\theta(t,x)$ from a simple source π_0 (typically $\mathcal{N}(0,I)$) transports samples to the target distribution π_1 (Lipman et al., 2023). Training uses the CFM objective. Given pairs (x_0, x_1) with $x_0 \sim \pi_0$ and $x_1 \sim \pi_1$, define the linear interpolant $x_t = (1-t)x_0 + t\,x_1$. The target velocity is the direction $x_1 - x_0$, and the loss is:

$$\mathcal{L}_{\text{CFM}}(\theta) = \mathbb{E}_{t\sim U[0,1],\,(x_0,x_1)\sim\mathcal{C}}\big[\|v_\theta(t,x_t) - (x_1 - x_0)\|^2\big], \tag{1}$$

where $\mathcal{C}$ is the coupling (joint distribution over pairs). With independent coupling $\mathcal{C}_{\text{ind}}$, source and target samples are paired independently. Different pairs can provide conflicting regression targets near the same intermediate state.

At inference, the ODE is integrated numerically with Euler's method for N NFE:

$$x_{t+\Delta t} = x_t + \Delta t \cdot v_\theta(t,x_t), \quad \Delta t = 1/N. \tag{2}$$

2.2 Rectified Flow

RectFlow iteratively constructs source–target pairs from a learned flow and retrains on them (Liu et al., 2023). Let $v^{(0)} = v_{\text{base}}$ be a model trained with independent coupling. In round k, sources $x_0^{(i)} \sim \pi_0$ are integrated under the previous model to generate endpoints, for example with Euler:

$$x_1^{(i)} = \text{Euler}(v^{(k-1)}, x_0^{(i)}, N_{\text{gen}}). \tag{3}$$

The next model is trained with Eq. (1) on the coupling $\mathcal{C}_{\text{van}}^{(k)} = \{(x_0^{(i)}, x_1^{(i)})\}$. Each pair retains a source and the endpoint reached by the previous model, but the new training path is the straight segment between these two states. The procedure therefore reuses the previous flow's endpoint assignments while learning a new velocity field from their linear interpolants. In our notation, $k = 0$ denotes the base model and $k = 1$ denotes one reflow round.

For any fixed coupling, the population CFM minimizer is

$$v^*(t,x) = \mathbb{E}[x_1 - x_0 \mid x_t = x]. \tag{4}$$

The conditional variance of $x_1 - x_0$ describes the disagreement among training directions at an intermediate state. A coupling with consistent directions can make this regression easier and support straighter learned trajectories. In the ideal case where the conditional displacement is unambiguous along each interpolation path and the regression is exact, the learned velocity follows that straight path. With conflicting directions, the conditional mean averages their displacements. This connects the training coupling to the difficulty of learning a flow that can be sampled in few steps. Although solutions of a sufficiently regular ODE cannot intersect at the same time, the *linear interpolants* between their source and endpoint pairs need not inherit this property.

In practice, the quality of the generated coupling depends on the previous model and its forward integration. Endpoint errors or unfavorable pair geometry become part of the next round's training data. We therefore investigate whether changing only the offline pair-generation solver can improve the subsequent RectFlow model, while preserving its training objective and inference procedure. The integration budget N_{gen} is an offline choice and is separate from the inference NFE of the retrained model. This distinction allows us to spend additional computation on constructing the pairs, then assess whether that investment improves generation at the same deployment budget.

3 Related Work

Recent methods use divergence at different stages of flow-model training and sampling. The Flow Divergence Sampler (FDS) of Cha et al. (2026) uses divergence as a data-free criterion for sampling-time refinement, selecting among Gaussian perturbations of the current state before a solver step. Their analysis relates divergence to the optimal CFM residual, motivating it as a surrogate for local ambiguity. This connects to our use of a divergence score for candidate selection, although FDS applies its refinement directly when producing samples. Flow and Divergence Matching (FDM) (Huang et al., 2026) instead augments flow-matching training with a divergence-matching objective, estimated using Hutchinson's trace estimator. Its analysis relates errors in the probability path to discrepancies in both the velocity and its divergence, motivating control of these quantities during learning. Thus FDS and FDM place the additional computation at sampling time and training time, respectively.

Our method uses divergence as an *offline criterion* for improving the couplings used by Rectified Flow. The image implementation selects states with smaller estimated *absolute* divergence, and applies this search only when generating the cached pairs for reflow. This places the additional computation in pair generation: the CFM loss, architecture, and plain-Euler inference remain unchanged. Our generation experiments evaluate this offline use of the absolute-divergence score.

These approaches address different parts of the flow-model pipeline. DS-RectFlow changes the data used by reflow: the divergence score is used while constructing the coupling and is absent from the reflow loss and deployed sampler. Its extra cost is consequently incurred for each cached pair dataset, whereas a sampling-time search must be repeated for each generated batch. Combining offline correction with training-time divergence objectives is a possible extension, but is not evaluated here.

4 Divergence-Suppressed Coupling Generation

4.1 Motivation: Local Compressibility

RectFlow relies on endpoints generated by integrating a learned velocity field. During pair generation, this field may locally contract or expand volumes of neighboring states, changing the geometry of the generated pairs. Divergence provides a simple local diagnostic of this compressibility. For a differentiable ODE flow $x_t = \Phi_t(x_0)$ with invertible Jacobian $J_t = D_{x_0}\Phi_t(x_0)$,

$$\frac{d}{dt} \log |\det J_t| = \nabla \cdot v_\theta(t, x_t). \tag{5}$$

Thus divergence determines instantaneous local volume change. Appendix A gives the derivation.

We use estimated absolute divergence as a simple local ranking signal for candidate states. This motivates searching nearby states when generating offline pairs, then evaluating the resulting coupling

through the downstream reflow model. Divergence does not fully characterize curvature or coupling quality, and lower divergence is not a guarantee of a better endpoint. The method is an empirical intervention in pair generation.

4.2 Low-Divergence Coupling Integrator

At state x and time t, we draw m Gaussian perturbations $x'_j = x + \delta\xi_j$, with $\xi_j \sim \mathcal{N}(0, I)$, and include the current state among the candidates. We select

$$x^\star = \underset{x' \in \{x, x'_1, \ldots, x'_m\}}{\arg\min} \left| \widehat{\nabla \cdot v_\theta}(t, x') \right|. \tag{6}$$

For high-dimensional states, the divergence score is estimated with Hutchinson's trace estimator (Hutchinson, 1989; Grathwohl et al., 2018). Each probe requires a vector–Jacobian product; no full Jacobian or second-order derivative is formed. The implementation divides the estimated trace by the state dimension, which leaves candidate rankings unchanged. Estimator details and computational cost appear in Appendix D.

The corrected Euler step advances from the selected state:

$$x_{t+\Delta t} = \begin{cases} x_t^\star + \Delta t\, v_\theta(t, x_t^\star), & t \le t_{\text{stop}}, \\ x_t + \Delta t\, v_\theta(t, x_t), & t > t_{\text{stop}}. \end{cases} \tag{7}$$

Here δ is the per-coordinate perturbation scale and t_{stop} limits the correction to the early part of integration. The learned velocity field is evaluated without modification at a displaced state; the procedure does not compute a Helmholtz projection. Including x_t ensures a non-increasing *estimated* score among the evaluated candidates, but does not guarantee a reduction in true divergence or impose a bound on the Jacobian of the corrected solver. Algorithm 1 gives the search procedure. For RK45 pair generation, the same search is applied at accepted steps, as detailed in Section 5.3.

Algorithm 1 HutchinsonSearchCorrect: low-divergence neighbor search

Require: state x, time t, model v_θ, perturbation scale δ, candidates m, Hutchinson probes n_h
Ensure: selected state $x^\star$ and velocity $v^\star = v_\theta(t, x^\star)$

$(v^\star, \hat{d}^\star) \leftarrow$ HutchinsonDivEstimate(v_θ, t, x, n_h)
$x^\star \leftarrow x$
for $j = 1, \ldots, m$ **do**
  $x' \leftarrow x + \delta\xi_j, \quad \xi_j \sim \mathcal{N}(0, I)$
  $(v', \hat{d}') \leftarrow$ HutchinsonDivEstimate(v_θ, t, x', n_h)
  for each sample b in the batch **do**
    if $|\hat{d}'_b| < |\hat{d}^\star_b|$ **then**
      $(x^\star_b, v^\star_b, \hat{d}^\star_b) \leftarrow (x'_b, v'_b, \hat{d}'_b)$
    end if
  end for
end for
return $x^\star, v^\star$ ▷ Detach before the next solver step

4.3 DS-RectFlow Pipeline

We begin with a base model $v_{\text{DS}}^{(0)} = v_{\text{base}}$ trained with independent coupling. In each reflow round k, we draw sources $x_0^{(i)} \sim \pi_0$ and generate endpoints using the corrected offline solver under $v_{\text{DS}}^{(k-1)}$. The resulting coupling is

$$\mathcal{C}_{\text{DS}}^{(k)} = \{(x_0^{(i)}, x_1^{(i)})\}_{i=1}^n. \tag{8}$$

Each pair retains the *original* source $x_0^{(i)}$, even when the integration state is displaced during generation. We train $v_{\text{DS}}^{(k)}$ on these cached pairs using the standard CFM loss in Eq. (1), with the same architecture, optimizer, and training budget as vanilla RectFlow.

All divergence estimation and correction occur during offline pair generation. At inference, the trained model uses plain Euler, with the same number of network evaluations as vanilla RectFlow at

a fixed NFE and no divergence-estimation or search overhead. The additional pair-generation cost is incurred once per cached coupling dataset and amortized over training and subsequent sampling. The image experiments use the state-search integrator above; the 2D experiments use a related directional correction detailed in Appendix B.

5 EXPERIMENTS

We evaluate whether changing offline pair generation improves the subsequent RectFlow model under matched training and inference settings. The 2D benchmarks use the directional velocity correction in Appendix B, while the image benchmarks use the neighborhood-search integrator in Section 4.2. All reported inference uses plain Euler without correction.

Corrections and comparison protocol. For the 2D benchmarks, we compute divergence by exact autograd and use a random finite-difference direction to construct a local velocity correction. Its strength α scales the change to the integration velocity. For images, we use Hutchinson estimates to rank Gaussian candidate states, with δ controlling their perturbation scale. The two implementations share the offline intervention point but have distinct updates; neither computes a Helmholtz projection. Appendix B specifies the 2D rule, and Appendix D gives the image estimator and search cost. Within each comparison, vanilla and DS-RectFlow use the same architecture, training budget, and inference solver. We vary inference NFE separately from the offline generation solver to assess the usefulness of the resulting pairs.

5.1 CHECKERBOARD BENCHMARK

We first evaluate DS-RectFlow on Checkerboard, with setup detailed in Appendix C. We report the sliced Wasserstein distance (SWD) between a generated sample and a held-out reference drawn from π_1, using 2000 projections and $10{,}000$ samples per side, and the *forbidden cell fraction*, i.e. the fraction of generated points landing outside the eight black cells. Both metrics are lower-is-better; SWD measures overall distributional fidelity, while the forbidden cell fraction isolates the mode of failure most specific to this benchmark—mass leaking into regions that should have no density at all. Table 1 reports SWD across five NFE values for each method; Table 2 reports the corresponding

Table 1: Checkerboard: SWD (lower is better) and inference time on 100k samples, across NFE values. FM baselines fixed at NFE$= 20$. Best SWD per NFE in bold. Rectflow methods share inference time at every NFE because the divergence-suppressing correction is offline-only.

	NFE=1		NFE=5		NFE=10		NFE=15		NFE=20	
Method	SWD	Time	SWD	Time	SWD	Time	SWD	Time	SWD	Time
Flow Matching (ref)	–	–	–	–	–	–	–	–	0.162	0.066s
RectFlow-k=1	0.1654	0.004s	0.1662	0.017s	0.1658	0.032s	0.1653	0.048s	0.1658	0.064s
DS-RectFlow-k=1	0.1303	0.004s	0.1263	0.017s	0.1247	0.032s	0.1236	0.048s	0.1239	0.064s
RectFlow-k=2	0.1657	0.004s	0.1672	0.017s	0.1664	0.032s	0.1655	0.048s	0.1669	0.064s
DS-RectFlow-k=2	**0.1020**	0.004s	**0.1014**	0.017s	**0.1016**	0.032s	**0.1006**	0.048s	**0.1020**	0.064s

forbidden cell fractions. Timings are measured on 100k samples with the first batch excluded as warmup. At NFE$= 20$, vanilla RectFlow stays near SWD 0.166 at both $k = 1$ and $k = 2$, while DS-RectFlow reaches 0.124 and 0.102, respectively. DS-RectFlow-$k = 2$ also achieves SWD 0.102 in one Euler step, taking 0.004s compared with 0.066s for the 20-step FM reference. Its SWD varies little over the evaluated NFE range. These results show that the offline correction improves the reflow model under the same training setup. At one step, the second-round DS model improves SWD from 0.1657 to 0.1020 relative to vanilla, about a 38% reduction. Compared with the base FM reference, it uses about $16\times$ less measured inference time while achieving lower SWD. Across the full NFE sweep, its SWD remains between 0.1006 and 0.1020, so extra Euler steps provide little benefit for this metric. The nearly constant score is useful operationally: the improvement from corrected training pairs is already present at the smallest inference budget.

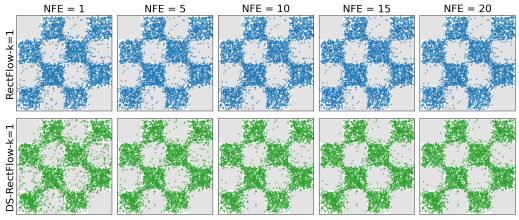


Figure 1: Checkerboard, k=1. Top: RectFlow. Bottom: DS-RectFlow. Vanilla samples smear across cell boundaries; the divergence-suppressing variant has sharper cells but visible diagonal artefacts that one reflow round has not yet absorbed.

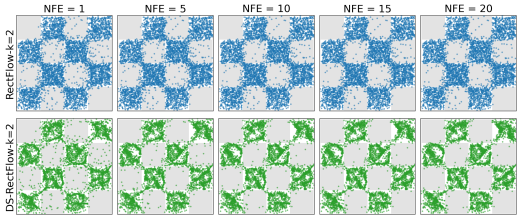


Figure 2: Checkerboard, k=2. Top: RectFlow, essentially unchanged from k=1. Bottom: DS-RectFlow. The diagonal artefacts collapse into thin filaments, and panels are nearly identical across NFE, consistent with the small change in SWD across NFE.

The forbidden-cell metric gives a more qualified picture (Table 2). After one round, DS-RectFlow places more mass outside the target cells at NFE= 1 (11.02% versus 7.85%), despite its better SWD. After two rounds, this fraction is 4.99% at NFE= 1 and 2.71% at NFE= 20, both below the corresponding vanilla values. Improvement in one metric therefore does not imply improvement in every metric or at every reflow round.

Figures 1 and 2 show 100,000 generated samples per panel. DS-RectFlow has visible diagonal artifacts after one round; after two rounds, these are reduced and the samples vary less across inference budgets, in line with the quantitative results. The samples retain the checkerboard structure under both methods, but vanilla places visible mass across cell boundaries. DS-RectFlow's sharper cells after the first round coexist with diagonal bands in forbidden regions, illustrating why SWD and the forbidden-cell fraction should be considered together. By the second round, these bands are thinner and the separation between target cells is clearer. This visual comparison complements the distributional metrics without identifying a particular cause of the remaining errors.

Table 2: Checkerboard: Forbidden cell fraction (%, lower is better). Fraction of generated samples landing outside the eight target black cells. FM baselines fixed at NFE= 20.

Method	NFE=1	NFE=5	NFE=10	NFE=15	NFE=20
FM (ref, NFE=20)			6.06% (fixed)		
RectFlow-k=1	7.85%	6.99%	6.80%	6.73%	6.72%
DS-RectFlow-k=1	11.02%	7.08%	5.99%	5.53%	5.33%
RectFlow-k=2	8.39%	7.54%	7.23%	7.12%	7.07%
DS-RectFlow-k=2	**4.99%**	**3.46%**	**3.06%**	**2.87%**	**2.71%**

5.2 GMM Crossing Benchmark

We next evaluate DS-RectFlow on the Gaussian mixture model (GMM) benchmark, with setup detailed in Appendix C. We report SWD against a held-out reference from π_1 and inference time on 100k samples. Table 3 shows that the benefit depends on both the reflow round and inference budget. After one round, DS-RectFlow is worse at NFE= 1 (SWD 1.755 versus 0.831) and NFE= 5 (0.888 versus 0.751). It improves upon vanilla at NFE= 10 (0.606 versus 0.751) and reaches 0.501

Table 3: GMM: SWD (lower is better) and inference time on 100k samples, across NFE values. FM baselines fixed at NFE= 20. Best SWD per NFE in bold.

	NFE=1		NFE=5		NFE=10		NFE=15		NFE=20	
Method	SWD	Time	SWD	Time	SWD	Time	SWD	Time	SWD	Time
FM (ref)	–	–	–	–	–	–	–	–	0.661	0.076s
RectFlow-k=1	0.831	0.006s	0.751	0.021s	0.751	0.040s	0.739	0.058s	0.751	0.077s
DS-RectFlow-k=1	1.755	0.006s	0.888	0.021s	0.606	0.040s	**0.533**	0.058s	**0.501**	0.076s
RectFlow-k=2	0.801	0.006s	0.790	0.021s	0.785	0.039s	0.782	0.058s	0.788	0.077s
DS-RectFlow-k=2	**0.586**	0.006s	**0.587**	0.021s	**0.573**	0.040s	0.564	0.058s	0.550	0.077s

at NFE$=20$. The low-step degradation is a limitation of this setting; these results alone do not explain its mechanism.

After two rounds, DS-RectFlow improves upon vanilla across all reported inference budgets. At NFE$=1$, SWD is 0.586 versus 0.801, with both models taking 0.006s per 100,000 samples. The DS-RectFlow one-step score improves substantially from the first round, while its 20-step score is slightly worse (0.550 versus 0.501). Thus additional rectification is useful here for low-step generation, but does not improve every operating point. Figures 3 and 4 show the corresponding generated samples across NFE budgets. Each panel shows 5,000 samples, with target modes marked by gold stars. After the second round, DS-RectFlow's clusters vary less visibly across NFE than after the first round, consistent with the smaller gap between its one-step and many-step SWD. The practical choice still depends on the sampling budget: $k = 2$ gives the best reported one-step result, while $k = 1$ gives the best 20-step result. Both the quantitative and qualitative comparisons therefore favor evaluating the complete reflow pipeline at its intended inference budget, rather than assuming that each additional round improves every metric.

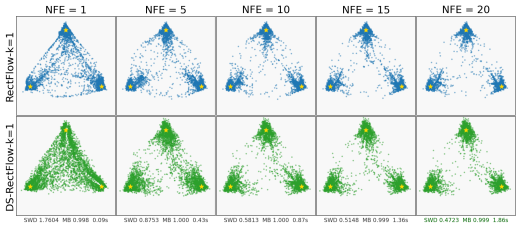


Figure 3: GMM, k=1. Top: RectFlow. Bottom: DS-RectFlow.

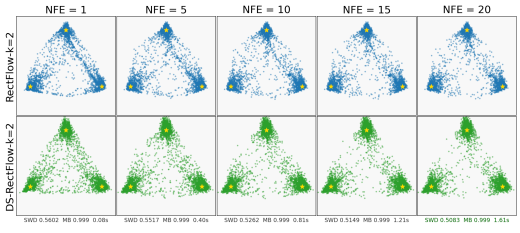


Figure 4: GMM, k=2. Top: RectFlow. Bottom: DS-RectFlow.

5.3 CIFAR-10 Benchmark

We then evaluate DS-RectFlow on CIFAR-10, with setup detailed in Appendix C. We report Fréchet Inception Distance (FID) (Heusel et al., 2017) computed against the full CIFAR-10 training set using $50{,}000$ generated samples and a fixed torchvision InceptionV3 feature extractor, ensuring internal consistency across all runs. FID is evaluated at seven inference budgets, $\mathrm{NFE} \in \{1, 2, 3, 5, 10, 15, 20\}$. We compare two offline pair generation strategies: a fixed 20-step Euler solver and an adaptive RK45 solver. In both cases, the vanilla and DS-RectFlow models use plain Euler at inference; the divergence-suppressing correction is applied *only* during offline pair generation and incurs no inference overhead.

In the Euler-20 setting we use $\delta = 0.06$, and the check is performed at each of the $N = 20$ uniform steps that satisfy $t \le t_{\text{stop}}$. In the RK45 setting we use $\delta = 0.03$, and the check is performed once per accepted adaptive step (rather than at every internal RK45 stage); the First Same As Last (FSAL) slope $f = v_\theta(x', t)$ is recomputed after each accepted correction to keep the solver state consistent. Beyond t_{stop}, both solvers revert to their standard updates, Appendices E.1 and E.2 report additional perturbation-scale results.

Tables 4 and 5 report FID as $\mu \pm \sigma$. For each method and offline solver, results are averaged over 3 independent reflow training runs; μ and σ denote the mean and standard deviation across these runs. DS-RectFlow has lower mean FID at every evaluated inference budget under both offline solvers. The tables show the reported variability; these aggregate statistics alone do not provide a paired significance test.

With Euler-20 pairs, single-step FID decreases from 17.276 ± 0.019 to 12.272 ± 0.066, a reduction of about 29% in the mean. At NFE$=20$, the corresponding means are 14.409 and 9.240. Across the NFE sweep, the absolute reduction is approximately 5.0–5.4 FID points. The one-step DS model also has lower mean FID than vanilla RectFlow at every tested inference budget, including 20 steps. Thus the improvement from offline pair generation is retained at deployment with a single network evaluation.

RK45-generated pairs yield lower mean FID for both methods. Even with this stronger baseline, DS-RectFlow reduces single-step FID from 9.388 ± 0.037 to 8.939 ± 0.053, and 20-step FID from

Table 4: CIFAR-10: Euler-20 offline pair generation, one reflow round. FID (↓): mean ± standard deviation over 3 independent reflow training runs. IS (↑) is shown with its reported spread. Inference uses plain Euler throughout.

	FID ↓		IS ↑	
NFE	RectFlow	DS-RectFlow	RectFlow	DS-RectFlow
1	17.276 ± 0.019	12.272 ± 0.066	6.515 ± 0.112	6.700 ± 0.065
2	16.050 ± 0.038	10.794 ± 0.025	6.724 ± 0.085	7.003 ± 0.103
3	15.645 ± 0.088	10.339 ± 0.002	6.834 ± 0.076	7.106 ± 0.070
5	15.234 ± 0.063	9.857 ± 0.069	6.896 ± 0.035	7.171 ± 0.051
10	14.692 ± 0.047	9.444 ± 0.027	6.971 ± 0.116	7.229 ± 0.058
15	14.485 ± 0.057	9.272 ± 0.009	7.001 ± 0.099	7.295 ± 0.080
20	14.409 ± 0.069	9.240 ± 0.095	7.028 ± 0.086	7.326 ± 0.125

Table 5: CIFAR-10: RK45 offline pair generation, one reflow round. FID (↓): mean ± standard deviation over 3 independent reflow training runs. Inference uses plain Euler throughout.

NFE	RectFlow FID	DS-RectFlow FID	Δ FID
1	9.388 ± 0.037	8.939 ± 0.053	-0.449
2	8.061 ± 0.020	7.545 ± 0.070	-0.516
3	7.614 ± 0.097	7.070 ± 0.017	-0.544
5	7.223 ± 0.027	6.703 ± 0.050	-0.520
10	6.924 ± 0.068	6.411 ± 0.007	-0.513
15	6.864 ± 0.088	6.277 ± 0.010	-0.587
20	6.754 ± 0.023	6.199 ± 0.048	-0.555

Δ FID is DS minus RectFlow, computed from displayed means.

6.754 to 6.199. The mean reductions range from 0.449 to 0.587 FID points. The gains are smaller than with Euler-generated pairs, but persist across all reported NFE without adding computation to the inference sampler.

Inception Score. Table 4 also reports Inception Score (IS, higher is better) (**?**) for the Euler-20 setting. Mean IS increases at every NFE: from 6.515 to 6.700 at one step, and from 7.028 to 7.326 at 20 steps. The FID improvement is therefore accompanied by improved mean IS in this setting. No IS result is available for the CIFAR-10 RK45 comparison.

5.4 CelebA-64 Benchmark

We evaluate CelebA-64 using $50{,}000$ generated samples per FID evaluation and inference budgets $\mathrm{NFE} \in \{1, 2, 3, 5, 10, 15, 20\}$. Offline pair generation uses either Euler-50 or adaptive RK45. Both DS settings use $\delta = 0.03$ and $t_{\mathrm{stop}} = 0.5$; all models use plain Euler at inference. Training details are given in Appendix C.4.

Table 6: CelebA-64: Euler-50 offline pair generation, one reflow round. FID (↓): mean ± standard deviation over 3 independent reflow training runs. IS (↑) is shown with its reported spread. Inference uses plain Euler throughout.

	FID ↓		IS ↑	
NFE	RectFlow	DS-RectFlow	RectFlow	DS-RectFlow
1	6.737 ± 0.011	5.826 ± 0.040	2.176 ± 0.013	2.242 ± 0.026
2	6.410 ± 0.037	4.954 ± 0.072	2.208 ± 0.021	2.275 ± 0.016
3	6.079 ± 0.002	4.759 ± 0.026	2.220 ± 0.014	2.279 ± 0.017
5	5.912 ± 0.106	4.507 ± 0.073	2.222 ± 0.024	2.288 ± 0.018
10	5.775 ± 0.092	4.228 ± 0.013	2.239 ± 0.018	2.317 ± 0.025
15	5.587 ± 0.038	4.126 ± 0.037	2.254 ± 0.010	2.322 ± 0.018
20	5.575 ± 0.060	4.167 ± 0.080	2.262 ± 0.016	2.312 ± 0.026

Table 7: CelebA-64: RK45 offline pair generation, one reflow round. FID (↓): mean ± standard deviation over 3 independent reflow training runs. IS (↑) is shown with its reported spread. Inference uses plain Euler throughout.

	FID ↓		IS ↑	
NFE	RectFlow	DS-RectFlow	RectFlow	DS-RectFlow
1	4.454 ± 0.082	4.283 ± 0.041	2.286 ± 0.022	2.303 ± 0.017
2	3.806 ± 0.023	3.618 ± 0.043	2.311 ± 0.017	2.318 ± 0.023
3	3.510 ± 0.070	3.426 ± 0.052	2.326 ± 0.019	2.342 ± 0.021
5	3.423 ± 0.080	3.268 ± 0.027	2.335 ± 0.018	2.345 ± 0.024
10	3.228 ± 0.006	3.087 ± 0.025	2.335 ± 0.028	2.365 ± 0.025
15	3.200 ± 0.034	3.036 ± 0.036	2.346 ± 0.023	2.356 ± 0.022
20	3.127 ± 0.034	3.024 ± 0.026	2.361 ± 0.023	2.365 ± 0.028

Tables 6 and 7 report FID means and standard deviations over 3 independent reflow training runs for each method and offline solver, with one reflow round per run. With Euler-50-generated pairs, single-step FID decreases from 6.737 ± 0.011 to 5.826 ± 0.040, a reduction of about 14% in the mean. At NFE$= 10$, mean FID falls from 5.775 to 4.228, and at NFE$= 20$ from 5.575 to 4.167. The mean improvement ranges from 0.911 to 1.547 FID points across the evaluated budgets. At two steps, DS-RectFlow already has lower mean FID than vanilla RectFlow at every NFE in this sweep.

With RK45-generated pairs, the baseline is substantially stronger, and the FID gains are correspondingly smaller. Single-step FID decreases from 4.454 ± 0.082 to 4.283 ± 0.041, while the 20-step mean decreases from 3.127 to 3.024. Across the evaluated inference budgets, the mean improvement ranges from 0.084 to 0.188 FID points. Although these gains are modest relative to the Euler-50 setting and should be interpreted in light of the reported across-seed variability, DS-RectFlow achieves lower mean FID at every reported NFE under both offline solvers. Together with the CIFAR-10 results, this suggests that the benefit of corrected offline couplings is not limited to a particular pair-generation solver.

Inception Score. The IS results provide a complementary view. With Euler-50-generated pairs, mean IS is higher for DS-RectFlow at every NFE, with improvements ranging from 0.050 to 0.078. Under RK45, the differences are smaller, ranging from 0.004 to 0.030, and are comparable to or smaller than the reported across-seed spread. We therefore treat IS as supporting evidence that the FID improvements are not accompanied by an obvious degradation in this metric, rather than as evidence of a statistically significant IS gain.

6 Conclusion

We presented DS-RectFlow, which modifies the procedure used for *offline* coupling generation in Rectified Flow. The method uses divergence as a local criterion for constructing couplings. In the image setting, it selects nearby states with smaller estimated absolute divergence and advances using the unmodified learned velocity field. The training objective and architecture remain unchanged, and inference uses plain Euler with no additional computation at a fixed NFE.

Experiments on Checkerboard, GMM, CIFAR-10, and CelebA-64 show improved low-NFE generation under the evaluated settings. The image experiments report lower mean FID across three independent reflow training runs with both Euler- and RK45-generated couplings. These results support offline coupling quality as an important factor in RectFlow performance and demonstrate a practical way to improve it without adding inference overhead. Evaluating similar offline corrections in stochastic interpolants, optimal-transport flow matching, and diffusion models is a direction for future work.

AI Assistance Disclosure. In this work, we used generative AI tools to assist with editing the manuscript for clarity and readability and with editing, debugging, and refactoring software code written by the authors. We did not use generative AI tools to generate synthetic datasets, develop the theoretical or conceptual framework, formulate or prove mathematical claims, implement the proposed methods, or design or interpret the experiments. All AI-assisted text was reviewed and

revised by the authors, and all AI-assisted code was manually inspected and tested for correctness. Numerical implementations were additionally checked against independent implementations where applicable, and all reported quantitative results were verified against stored experimental outputs. We take responsibility for the final content of this work, including text, claims, code, and artifacts produced with the aid of generative AI.

REFERENCES

Michael Albergo, Nicholas M Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. *Journal of Machine Learning Research*, 26(209):1–80, 2025.

Michael S. Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. *International Conference on Learning Representations*, 2023.

Avishek Joey Bose, Tara Akhound-Sadegh, Guillaume Huguet, Kilian Fatras, Jarrid Rector-Brooks, Cheng-Hao Liu, Andrei Cristian Nica, Maksym Korablyov, Michael Bronstein, and Alexander Tong. Se (3)-stochastic flow matching for protein backbone generation. *arXiv preprint arXiv:2310.02391*, 2023.

Yeonwoo Cha, Jaehoon Yoo, Semin Kim, Yunseo Park, Jinhyeon Kwon, and Seunghoon Hong. Training-free refinement of flow matching with divergence-based sampling. *arXiv preprint arXiv:2604.04646*, 2026.

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. *Advances in Neural Information Processing Systems*, 31, 2018.

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 *Forty-first international conference on machine learning*, 2024.

Will Grathwohl, Ricky TQ Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *arXiv preprint arXiv:1810.01367*, 2018.

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. *Advances in neural information processing systems*, 30, 2017.

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

Yuhao Huang, Taos Transue, Shih-Hsin Wang, William Feldman, Hong Zhang, and Bao Wang. Improving flow matching by aligning flow divergence. *arXiv preprint arXiv:2602.00869*, 2026.

Michael F Hutchinson. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 18(3):1059–1076, 1989.

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

Matthew Le, Apoorv Vyas, Bowen Shi, Brian Karrer, Leda Sari, Rashel Moritz, Mary Williamson, Vimal Manohar, Yossi Adi, Jay Mahadeokar, et al. Voicebox: Text-guided multilingual universal speech generation at scale. *Advances in neural information processing systems*, 36:14005–14034, 2023.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *International Conference on Learning Representations*, 2023.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *International Conference on Learning Representations*, 2023.

Xingchao Liu, Xiwen Zhang, Jianzhu Ma, Jian Peng, and Qiang Liu. Instaflow: One step is enough for high-quality diffusion-based text-to-image generation. *International Conference on Learning Representations*, 2024.

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.

Adam Polyak, Amit Zohar, Andrew Brown, Andros Tjandra, Animesh Sinha, Ann Lee, Apoorv Vyas, Bowen Shi, Chih-Yao Ma, Ching-Yao Chuang, et al. Movie gen: A cast of media foundation models. *arXiv preprint arXiv:2410.13720*, 2024.

Aram-Alexandre Pooladian, Heli Ben-Hamu, Carles Domingo-Enrich, Brandon Amos, Yaron Lipman, and Ricky TQ Chen. Multisample flow matching: Straightening flows with minibatch couplings. *arXiv preprint arXiv:2304.14772*, 2023.

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

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pp. 2256–2265. pmlr, 2015.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. *International Conference on Machine Learning*, 2023.

Alexander Tong, Kilian Fatras, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. *arXiv preprint arXiv:2302.00482*, 2023.

Qinsheng Zhang and Yongxin Chen. Fast sampling of diffusion models with exponential integrator. *arXiv preprint arXiv:2204.13902*, 2022.

A Divergence and Local Trajectory Compression

On Checkerboard, we evolve 6400 Gaussian source samples to $t = 0.5$ with 20 Euler steps. We compare local convergence $c = \max(0, -\nabla \cdot v(0.5, x_{0.5}))$, computed by exact autograd, with compression $s = \max(0, -\log|\det J|)$ of the numerical flow map, estimated by coordinate finite differences of size 0.05. The correlations in Table 8 describe an association between instantaneous convergence and accumulated compression. They do not compare candidate-ranking rules or explain the downstream generation gains. The recorded crossing rate is zero for every model, so this diagnostic supplies no evidence of crossing reduction.

Table 8: Checkerboard convergence–compression diagnostic at $t = 0.5$. The correlations concern c and s; mean absolute divergence is reported separately. These are descriptive measurements.

Model	Pearson r	Spearman ρ	Mean $\lvert\nabla \cdot v\rvert$	Crossings
Base FM	0.8344	0.8034	2.6413	0.0%
RectFlow ($k = 1$)	0.9632	0.9270	0.9813	0.0%
DS-RectFlow ($k = 1$)	0.9441	0.8926	0.9261	0.0%

Local volume identity. For a differentiable ODE flow with invertible Jacobian J_t, $\dot{J}_t = D_x v(t, x_t) J_t$. Jacobi's formula gives $\frac{d}{dt} \log|\det J_t| = \operatorname{tr}(J_t^{-1} \dot{J}_t) = \operatorname{tr}(D_x v(t, x_t)) = \nabla \cdot v(t, x_t)$. This is a volume-change identity, not a straightness guarantee or a bound on the state-search solver.

B Directional Correction for the 2D Benchmarks

The Checkerboard and GMM experiments use a low-dimensional implementation that modifies the integration velocity during offline pair generation. At state x and time t, exact autograd gives $v_0 = v_\theta(t, x)$ and $d_0 = \nabla \cdot v_\theta(t, x)$. For a random unit direction u and finite-difference step h, define

$$\widehat{g} = \frac{\nabla \cdot v_\theta(t, x + hu) - d_0}{h} \, u. \tag{9}$$

This is a directional estimate of the divergence gradient. The correction used in the 2D experiments is

$$v_{\text{corr}} = v_0 - \alpha \frac{\|v_0\|}{\|\widehat{g}\|} \operatorname{sign}(d_0) \widehat{g}, \tag{10}$$

for nonzero $\widehat{g}$; a zero estimate provides no correction direction. The intent is to bias the step toward locally smaller absolute divergence. This directional rule is not a Helmholtz projection, and $\alpha = 1$ does not make the field divergence-free.

The 2D correction strength α scales a velocity adjustment; the image parameter δ sets the scale of Gaussian state perturbations. These parameters and updates are distinct. Both implementations intervene only in offline pair generation and train the next model with the same CFM objective. The 2D results therefore support the broader offline-correction approach, while the image results specifically evaluate the neighborhood-search integrator.

C Setup

C.1 Checkerboard

The target distribution π_1 is a 4×4 checkerboard: a uniform mixture over the eight black cells, each of area $(0.45)^2$, with the remaining eight white cells forming *forbidden regions* where any generated mass represents an error. The source is a standard Gaussian, $\pi_0 = \mathcal{N}(0, I_2)$. We use $n = 200{,}000$ coupling pairs per reflow round, $N_{\text{gen}} = 20$ Euler steps for coupling generation, and $k = 2$ rectification rounds for both vanilla RectFlow and DS-RectFlow; the divergence-suppressing correction strength is $\alpha = 0.5$. All other training details are shared with the baselines.

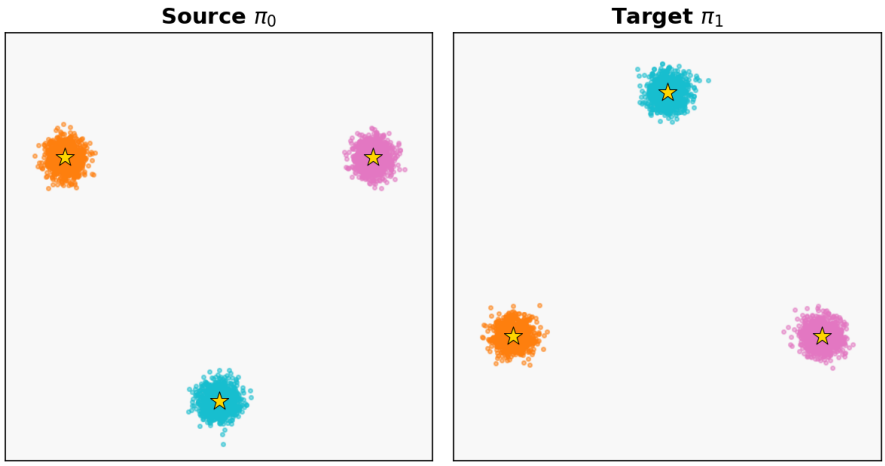


Figure 5: Source π_0 and target π_1 for the GMM crossing benchmark. Both are 3-mode Gaussian mixtures with $\sigma^2 = 0.3$ and modes on an equilateral triangle of circumradius $D = 10$; π_1 is π_0 rotated 60° about the origin. Colors indicate the ideal mode-to-mode correspondence (orange→orange, magenta→magenta, cyan→cyan), which forces all three transport paths through a congested central region and makes coupling crossings unavoidable under independent coupling.

C.2 GMM

Figure 5 illustrates the GMM crossing benchmark. Both π_0 and π_1 are 3-mode Gaussian mixtures whose modes lie on an equilateral triangle, with π_1 obtained by rotating π_0 by 60° about the origin. Under the colour-coded mode correspondence, each source cluster must be transported to a target cluster on the opposite side of the centre, so the three ideal transport paths are forced through a congested central region where mode assignments compete. This makes the benchmark a canonical stress test for RectFlow: coupling quality is determined by how cleanly the velocity field resolves these crossings near the centre, and we evaluate whether offline divergence-based correction improves the resulting reflow model.

Formally, π_0 has component variance $\sigma^2 = 0.3$ and modes at the vertices of an equilateral triangle of circumradius $D = 10$ centred at the origin,

$$\left\{\left(\tfrac{D\sqrt{3}}{2}, \tfrac{D}{2}\right),\ \left(-\tfrac{D\sqrt{3}}{2}, \tfrac{D}{2}\right),\ \left(0, -\tfrac{D\sqrt{3}}{2}\right)\right\},$$

and π_1 places its modes at $\{(\frac{D\sqrt{3}}{2}, -\frac{D}{2}), (-\frac{D\sqrt{3}}{2}, -\frac{D}{2}), (0, \frac{D\sqrt{3}}{2})\}$. We run $k = 3$ rectification rounds with $n = 100{,}000$ coupling pairs per round, $N_{\text{gen}} = 20$ Euler steps for coupling generation, and $\alpha = 0.5$ for the divergence-suppressing correction.

Both Checkerboard and GMM benchmarks use the same MLP architecture:

$$v_\theta(t, x) = \text{MLP}([x;\, t]), \tag{11}$$

where $[x;\, t] \in \mathbb{R}^3$ concatenates the 2D position x with the scalar time t. The MLP has three hidden layers of width 512 with ReLU activations. Total parameters: approximately 800k.

C.3 CIFAR-10

We evaluate on CIFAR-10 Krizhevsky et al. (2009) (32×32 RGB images, 50,000 training samples). Following the RectFlow framework, we adopt a two-stage pipeline. In *Stage 1* (base model), we train a Flow Matching model with independent Gaussian coupling using the NCSNpp U-Net architecture Song et al. (2020) ($n_f = 128$, channel multipliers $(1, 2, 2, 2)$, 4 residual blocks per resolution, attention at 16×16) for 50,000 iterations with batch size 128, Adam optimiser (lr$=2\times10^{-4}$, $\beta_1 = 0.9$), and EMA decay 0.9999. In *Stage 2* (reflow), we generate 50,000 (z_0, z_1) coupling pairs from the base model and train a new model on those pairs for a further 50,000 iterations under the same training configuration; this yields a 1-Rectified Flow model in our k-counting convention (k=1).

For each method and offline solver, the main FID comparison averages 3 independent reflow training runs and reports their mean and standard deviation. For DS-RectFlow, the divergence-suppressing Euler integrator is used during Stage 2 pair generation with hyperparameters $\delta = 0.06$, $t_{\text{stop}} = 0.5$, and 20 Euler steps. For the vanilla RectFlow baseline, pairs are generated with standard Euler integration under otherwise identical settings. The RK45 pair-generation comparison uses $\delta = 0.03$ for DS-RectFlow, as described in Section 5.3. All experiments are run on a single NVIDIA H100 (80 GB) GPU.

Comparison scope. Our absolute FID numbers are above those reported by Liu et al. (2023); we train for fewer iterations 50,000 instead of 400,000. Our goal is *not* to match the state-of-the-art absolute FID, but to isolate the effect of the divergence-suppressing coupling: both vanilla and DS-RectFlow share *identical* architectures, training budgets, and hyperparameters, so the comparison in Tables 4 and 5 isolates the choice of offline pair-generation procedure under a matched training budget.

C.4 CelebA-64 Training Setup

This section describes the training and inference configuration for the CelebA-64 experiments. The pipeline has three stages: (i) base 1-RectFlow pre-training, (ii) offline pair generation (vanilla and divergence-suppressed), and (iii) reflow training on the cached pairs. All stages share the same backbone, optimizer and data preprocessing; only the iteration budget, dataset and (in stage 2) the sampling solver change.

Dataset and preprocessing. We use the standard CelebA aligned/cropped split, resized to $3\times64\times64$, with the official partition file giving $162{,}770$ training images. Inputs are centered to $[-1, 1]$ and augmented with random horizontal flips during training. FID is evaluated against the CelebA-64 training set with $50{,}000$ generated samples using the torchvision InceptionV3 feature extractor.

All stages use the same NCSN++ / DDPM++ network, configured as a 4-scale variant for 64×64 inputs: channel multipliers $(1, 2, 2, 4)$ over resolutions $64\to32\to16\to8$, base width nf$=128$, 2 residual blocks per scale, BigGAN-style residual blocks with skip rescaling, attention at the 16×16 feature map, group normalization with Swish activations, dropout 0.1, positional time embedding with Fourier scale 16, 3×3 convolutions and EMA decay 0.9999. The reflow stage reuses the stage-1 weights as initialization and keeps the architecture identical.

We use Adam with learning rate 2×10^{-4}, $\beta_1{=}0.9$, $\varepsilon{=}10^{-8}$ and zero weight decay. A linear learning-rate warmup of $5{,}000$ iterations is applied at the start of each stage, followed by a constant schedule. Gradients are globally clipped at 1.0. Mini-batch size is 128 in every stage. The rectified-flow loss uses uniform $t \in [0, 1]$ and an ℓ_2 regression target with batch-mean reduction.

Stage 1: 1-RF pre-training. We train the base 1-RectFlow on the CelebA-64 training set for 5×10^5 iterations with the optimizer above. At batch size 128 over $162{,}770$ training images this is $\approx1{,}272$ iterations per epoch, i.e. ≈393 epochs in total. A snapshot is written every 5×10^4 iterations (snapshots $1, 2, \ldots, 10$) and a preemption-safe checkpoint every 10^4 iterations. The stage-1 model used as the shared starting point for both vanilla and DS-RectFlow branches is the final snapshot at iteration 5×10^5.

Stage 2: offline pair generation. Starting from the stage-1 checkpoint, we generate $50{,}000$ $(\mathbf{z}_0, \mathbf{z}_1)$ pairs by integrating the learned velocity field from $\mathbf{z}_0 \sim \mathcal{N}(0, I)$ to $\mathbf{z}_1$. Two solvers are reported:

- **Euler-50**: a fixed 50-step explicit Euler integrator.
- **RK45**: an adaptive Runge–Kutta–Fehlberg integrator (`scipy.integrate.solve_ivp`) with absolute and relative tolerances both set to 10^{-3}.

For DS-RectFlow, the divergence-suppressing correction is applied only for $t \le t_{\text{stop}}$, with perturbation scale δ, $n_H{=}8$ Hutchinson probes per divergence estimate and $K{=}8$ candidate perturbations per step. We sweep δ and t_{stop} over the same grid as on CIFAR-10 ($\delta \in \{0.01, 0.02, 0.03, 0.04, 0.05, 0.06\}$,

$t_{\text{stop}} \in \{0.2, 0.3, 0.4, 0.5\}$); the setting used for the multi-seed CelebA-64 comparison is δ=0.03, t_{stop}=0.5. The vanilla branch uses the same solvers and the same 50,000-sample budget but skips the DS correction. Pair generation runs on a single H100 GPU with batch size 128.

Stage 3: reflow training. On the cached pair dataset we train the reflowed model (k=1 reflow on top of the shared 1-RF base) for 1×10^5 iterations using the same optimizer, batch size, EMA decay and architecture as stage 1. The reflow loss is ℓ_2 with a uniform t-schedule. Snapshots are written every 10^4 iterations and the final (10^5-iter) checkpoint is used for all FID evaluations. For each method and offline solver, we perform 3 independent reflow training runs and report the mean and standard deviation of FID across these runs. Vanilla RectFlow-k=1 baselines are trained with exactly the same procedure on the vanilla pair dataset, so the method comparisons in Tables 6 and 7 differ only in the offline coupling.

Inference. At evaluation time we always use a plain Euler integrator, sweeping NFE $\in \{1, 2, 3, 5, 10, 15, 20\}$ on a uniform time grid in $[0, 1]$. Samples are generated from $\mathcal{N}(0, I)$ with initial noise scale 1 and decoded back to $[0, 1]$ before FID computation. Each FID number is computed from 50,000 samples against the CelebA-64 training reference statistics.

Compute. All stages are run on a single NVIDIA H100 (80 GB) GPU. Stage 1 takes roughly four days; stage 2 takes hours (Euler-50) to about a day (RK45 at 10^{-3} tolerance) per $(\delta, t_{\text{stop}})$ configuration; stage 3 takes about one day. End-to-end reproduction of one Euler or RK45 training run therefore requires ~ 6 GPU-days from scratch, or ~ 1.5 GPU-days starting from the shared stage-1 checkpoint. These estimates are per run, not the total cost of a multi-seed comparison.

D Hutchinson Divergence Estimation and Search Cost

D.1 Trace Estimator

At fixed (t, x), write $A = D_x v_\theta(t, x)$. Divergence is $\mathrm{tr}(A)$. For independent Rademacher vectors $\varepsilon_s \in \{\pm1\}^d$, the normalized Hutchinson estimate is

$$\hat{d}(t, x) = \frac{1}{n_h d} \sum_{s=1}^{n_h} \varepsilon_s^\top A \varepsilon_s, \qquad \mathbb{E}[\hat{d}(t, x)] = \frac{\nabla \cdot v_\theta(t, x)}{d}. \tag{12}$$

The constant normalization by d does not change candidate rankings. Each quadratic form is computed by a vector–Jacobian product:

$$\varepsilon_s^\top A \varepsilon_s = \nabla_x [v_\theta(t, x) \cdot \varepsilon_s] \cdot \varepsilon_s. \tag{13}$$

Autograd uses `create_graph=False`: this estimate is a score for selection, and is not differentiated to obtain a search direction. The evaluation graph is retained only as needed for additional probes.

For independent probes, the variance is

$$\mathrm{Var}(\hat{d}) = \frac{1}{n_h d^2} \sum_{i<j} (A_{ij} + A_{ji})^2. \tag{14}$$

This follows by expanding the Rademacher quadratic form and using independence of its off-diagonal products. Thus variance depends on the off-diagonal Jacobian entries; no dimension-only decay rate is assumed. Although $\hat{d}$ is unbiased for normalized divergence, $|\hat{d}|$ is generally biased for absolute normalized divergence. Selecting the smallest estimated score among candidates can therefore select estimation noise as well as a change in true divergence.

D.2 Search and Solver Integration

Algorithm 1 evaluates the original state and m independent perturbations centered on that state. Each candidate is scored once, and the current best state and its velocity are retained separately for each batch element. The Euler update is Eq. (7). For adaptive RK45, correction is performed once per accepted step within the correction horizon, and the cached slope is recomputed at the selected state before continuing. The original source sample is retained in the saved coupling pair.

D.3 Hyperparameters and Cost

Table 9: Neighborhood-search settings for the image experiments. The perturbation scale depends on the dataset and offline solver.

Parameter	Symbol	Setting
Perturbation scale	δ	CIFAR-10 Euler: 0.06; otherwise: 0.03
Perturbed candidates	m	8
Hutchinson probes	n_h	8
Correction horizon	t_{stop}	0.5
Euler generation steps	N_{gen}	CIFAR-10: 20; CelebA-64: 50

Each corrected search evaluates $m+1$ states, requiring $m+1$ forward evaluations and $(m+1)n_h$ vector–Jacobian products. With $m = n_h = 8$, this is 9 forward evaluations and 72 VJPs; these operations do not necessarily have equal wall-clock cost. On a uniform grid $t_i = i/N_{\text{gen}}$, $i = 0, \ldots, N_{\text{gen}} - 1$, the condition $t_i \le t_{\text{stop}}$ in Eq. (7) applies to $\min(N_{\text{gen}}, \lfloor t_{\text{stop}} N_{\text{gen}} \rfloor + 1)$ steps for $0 \le t_{\text{stop}} \le 1$. It therefore includes 11 of 20 Euler steps or 26 of 50 when $t_{\text{stop}} = 0.5$. All of this cost is confined to offline pair generation; inference requires neither input gradients nor candidate search.

E Perturbation-Scale Results

The main image tables report the updated multi-seed comparisons. The tables below retain additional scales from the earlier sweep as point estimates, whose seed aggregation was not specified. Superseded baseline and default-scale rows have been removed to avoid duplicating inconsistent versions of the main results. These additional-scale tables are not a matched multi-seed ranking of all hyperparameters.

E.1 CIFAR-10: RK45

Table 10: CIFAR-10, RK45: additional perturbation-scale FID point estimates from the earlier sweep. No across-seed uncertainty was provided for these settings. The updated baseline and default setting appear only in Table 5, as multi-seed results.

	Inference NFE						
δ	1	2	3	5	10	15	20
0.01	9.217	7.887	7.421	7.052	6.785	6.644	6.541
0.02	9.059	7.711	7.279	6.963	6.615	6.395	6.383
0.04	8.839	7.397	6.995	6.666	6.450	6.251	6.275
0.05	9.315	7.692	7.314	7.005	6.726	6.629	6.569

The updated $\delta = 0.03$ setting and its baseline are reported in Table 5. The additional point estimates here do not establish a statistically preferred perturbation scale.

E.2 CIFAR-10: Euler-20

The earlier sweep in Table 11 shows deterioration at large perturbation scales. The multi-seed evaluation of the chosen $\delta = 0.06$ setting is reported in Table 4.

E.3 CelebA-64: Euler-50

Table 12 shows higher FID at the largest tested perturbation scale, $\delta = 0.05$, than at the smaller scales in this earlier sweep. The default $\delta = 0.03$ comparison is now reported with multi-seed uncertainty in Table 6; it is not duplicated here.

Table 11: CIFAR-10, Euler-20: additional perturbation-scale FID point estimates from the earlier sweep. No across-seed uncertainty was provided for these settings. The updated baseline and default setting appear only in Table 4, as multi-seed results.

	Inference NFE						
δ	1	2	3	5	10	15	20
0.01	17.220	16.141	15.248	14.845	14.390	14.175	14.227
0.02	16.180	15.125	14.362	14.003	13.476	13.385	13.108
0.03	14.957	13.822	13.174	12.623	12.278	12.035	12.072
0.04	13.637	12.074	11.533	11.060	10.688	10.610	10.475
0.05	12.313	10.683	10.143	9.735	9.308	9.244	9.208
0.07	13.249	11.564	10.961	10.708	10.409	10.376	10.340
0.08	16.531	14.832	14.280	14.224	14.031	13.841	13.717
0.09	22.669	20.515	20.398	20.066	20.071	20.007	20.142
0.10	31.885	29.149	29.046	28.672	29.054	29.309	29.213

Table 12: CelebA-64, Euler-50: additional perturbation-scale FID point estimates from the earlier sweep. No across-seed uncertainty was provided for these settings. The updated baseline and default setting appear only in Table 6, as multi-seed results.

	Inference NFE						
δ	1	2	3	5	10	15	20
0.01	6.384	5.977	5.771	5.565	5.511	5.364	5.301
0.02	6.149	5.483	5.265	4.970	4.724	4.731	4.725
0.04	7.575	6.180	5.716	5.368	5.050	4.918	4.816
0.05	13.760	11.078	10.391	10.001	9.641	9.421	9.306

E.4 CelebA-64: RK45

Table 13: CelebA-64, RK45: additional perturbation-scale FID point estimates from the earlier sweep. No across-seed uncertainty was provided for these settings. The updated baseline and default setting appear only in Table 7, as multi-seed results.

	Inference NFE						
δ	1	2	3	5	10	15	20
0.01	4.455	3.768	3.505	3.221	3.163	3.048	3.158
0.02	4.507	3.716	3.633	3.409	3.194	3.133	3.093
0.04	4.410	3.711	3.474	3.250	3.171	3.094	3.099
0.05	4.450	3.776	3.465	3.315	3.141	3.175	2.930

The updated $\delta = 0.03$ setting and its baseline are reported in Table 7. The additional point estimates here do not establish a statistically preferred perturbation scale.