FINITE-BUDGET REPRESENTATIVE SAMPLING FROM DIFFUSION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion models are usually sampled with independent denoising trajectories, but under tight inference budgets an iid batch can duplicate common outcomes and miss important probability regions. We formulate finite-budget sampling as the construction of a representative empirical measure, separating two goals: placing support across density-separated regions and assigning weights for accurate expectation estimation. Without access to the original data, we learn a kernel from model samples that treats samples connected through high-density regions as similar and use it during sampling to guide the batch toward distribution matching and nonredundant coverage. We then assign quadrature weights to the sampled points for expectation estimation. Experiments show that, under limited sampling budgets, our method improves support coverage and produces weights that better represent the model distribution. When used in Monte Carlo tree search, the resulting weighted batches yield more accurate action-value estimates.

1 INTRODUCTION

Diffusion models are increasingly used as probabilistic components in systems that estimate quantities, compare alternatives, or plan under uncertainty. Such systems need more than individually plausible samples: they need a finite batch that adequately represents the model distribution. Yet generating each outcome requires repeated denoiser evaluations, making inference budgets tight. In a small iid batch, several trajectories can resolve to similar outcomes while other probability regions receive no support. This wastes model calls and can distort both coverage and expectation estimates. We ask how to construct a batch that better represents the target distribution under a fixed sampling budget.

We formulate finite-budget diffusion sampling as constructing a weighted empirical measure by jointly choosing sample locations and their weights. This exposes two complementary requirements: its support must cover distinct outcomes, and its weights must represent how much each outcome contributes. Reweighting cannot recover a missing outcome, while broad support alone need not reproduce the model distribution. Existing diffusion sampling methods typically promote diversity and return uniformly weighted outputs. Greater diversity, however, can overemphasize atypical outcomes rather than improve representativeness.

We therefore coordinate support placement and weight assignment. We learn a kernel from model samples that treats samples connected through high-density regions as similar. Using this kernel during sampling, we encourage the batch to match the model distribution and cover distinct density-connected regions instead of generating redundant samples. We then assign quadrature weights to the sampled points for expectation estimation. The functions whose expectations will be needed downstream are unknown. We therefore choose weights to match expectations for a selected set of test functions derived from the learned kernel and model score. In sequential batch sampling, given previously generated samples, the method guides only the new samples to complement them while leaving the previous samples unchanged.

Experiments on image diffusion models show improved finite-budget coverage and distributional fidelity. In Monte Carlo tree search, the resulting weighted batches yield more accurate action-value estimates and better decisions than iid and diversity-based alternatives.

Our contributions are:

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.

- We formulate finite-budget diffusion sampling as a weighted representative batch, jointly addressing support coverage and weight assignment.
- We use model-adapted spectral geometry, profiled MMD, and log-determinant coverage to guide one-shot or sequential sampling without a candidate pool.
- We combine spectral matching with a Stein discrepancy to compute final quadrature weights for expectation estimation.
- Experiments show improved coverage and weighted representation, yielding more accurate Monte Carlo tree search value estimates.

2 Related Work

Representative finite measures. Representative-point methods replace iid Monte Carlo with finite measures for integration. Kernel herding and Bayesian quadrature use target kernel expectations to select support or assign weights (Chen et al., 2010; Huszár & Duvenaud, 2012); support points optimize energy distance, while kernel thinning compresses an existing sample approximation (Mak & Joseph, 2018; Dwivedi & Mackey, 2024). Score-based alternatives avoid the normalized target density: Stein points optimize locations, whereas black-box importance sampling optimizes Stein-discrepancy weights on fixed samples (Chen et al., 2018; Liu & Lee, 2017). MMD gives the corresponding worst-case expectation error over an RKHS unit ball (Gretton et al., 2012). These methods require target quantities, direct point optimization, or existing samples. We instead coordinate support and weights during diffusion sampling using learned kernels.

Distribution-adapted spectral geometry. Laplacian eigenmaps and diffusion maps use graph or diffusion operators to obtain low-dimensional representations that preserve intrinsic connectivity (Belkin & Niyogi, 2001; Coifman & Lafon, 2006). Their leading eigenfunctions can reveal metastable regions separated by low-density bottlenecks (Nadler et al., 2005). Neural eigensolvers extend this principle to parametric functions and high-dimensional operators (Pfau et al., 2019; Zhang et al., 2022), while VAMPnets learn slow transfer-operator modes from observed trajectories (Mardt et al., 2018). We learn analogous features of the model distribution's Langevin operator at each noise level, and form a noise-dependent kernel to guide a finite batch during generation.

Non-iid diverse generative sampling. Recent methods replace independent generation with interacting trajectories. Particle Guidance adds a time-dependent population potential (Corso et al., 2024); DiverseFlow couples trajectories through the DPP volume of predicted outputs in a chosen feature space (Morshed & Boddeti, 2025); and Letting Trajectories Spread combines feature-space lateral control with flow-orthogonal stochastic perturbations (Wu et al., 2025). EDDY instead uses antisymmetric Stein interactions that preserve each particle marginal in its ideal formulation (Vinograd et al., 2026). These methods optimize diversity and return uniformly weighted outputs; even marginal preservation does not optimize the batch as a quadrature rule. Our method targets a representative weighted measure, using model-adapted geometry for support placement and a spectral–Stein objective for weight assignment.

Sequential sampling and planning. Kernel herding and Bayesian quadrature append points directly in the target space (Chen et al., 2010; Huszár & Duvenaud, 2012); we instead fix previous samples and guide only new diffusion trajectories. Low-discrepancy policy learning, particle-based control, and sampled chance nodes demonstrate the value of finite-sample expectation estimates (Arnold et al., 2022; Chua et al., 2018; Antonoglou et al., 2022). We use an otherwise unchanged planner to isolate the effect of our returned measures.

3 Method

3.1 Finite representative measure and MMD

We assume that a fixed pretrained diffusion model adequately represents the data distribution for the intended application. Let p_σ denote the model's sample distribution at noise level $\sigma \in [0, \sigma_{\max}]$ on $\mathcal{Z} = \mathbb{R}^d$, with p_0 its output distribution. Below, we assume $s_{\theta,\sigma}(z) = \nabla_z \log p_\sigma(z)$, $\sigma > 0$.

We write z_σ^i for particle i at noise level σ, with initial state $z_{\sigma_{\max}}^i$ and generated output z_0^i. We use z when neither index is needed. Given an budget N, one constructs a finite empirical measure

$$\widehat{p}_N(w) = \sum_{i=1}^{N} w_i \delta_{z_0^i}, \qquad w \in \Delta^{N-1}. \tag{1}$$

Here $\Delta^{N-1} = \{v \in \mathbb{R}^N_{\geq 0} : \mathbf{1}^\top v = 1\}$ is the probability simplex.

We assess a finite measure through maximum mean discrepancy (MMD). For a positive-semidefinite kernel k with RKHS $\mathcal{H}_k$ and feature map φ_k, define

$$m_k(P) = \mathbb{E}_{X\sim P}[\varphi_k(X)],$$
$$\mathrm{MMD}_k(P,Q) = \|m_k(P) - m_k(Q)\|_{\mathcal{H}_k} = \sup_{\|f\|_{\mathcal{H}_k}\leq 1} |\mathbb{E}_P[f] - \mathbb{E}_Q[f]|.$$

Thus MMD is the worst-case expectation error over the kernel's function class. MMD is only as informative as its kernel, which determines which distributional differences are visible. We therefore construct a kernel adapted to the probability geometry of the diffusion model.

3.2 Score-induced spectral kernel

At a small particle budget, we seek a description of density connectivity: states connected through high-density paths should have similar representations, while low-density bottlenecks should separate them. Using the model score at noise level $\sigma > 0$, define the Langevin operator

$$\mathcal{L}_\sigma f(z) = \Delta f(z) + s_{\theta,\sigma}(z)^\top \nabla f(z). \tag{2}$$

Intuitively, $\mathcal{L}_\sigma f(z)$ measures the expected rate of change of f when a point explores randomly while drifting toward higher probability density. The Laplacian term $\Delta f(z)$ captures the average effect of random exploration, while $s_{\theta,\sigma}(z)^\top \nabla f(z)$ captures the change along the density-increasing drift. At each fixed noise level σ, this movement describes connectivity within p_σ; its slow eigenfunctions provide the features used to build the kernel. Under standard regularity conditions, $\mathcal{L}_\sigma$ generates a Langevin diffusion with stationary distribution p_σ. We omit its constant mode and index the remaining eigenfunctions by increasing eigenvalue:

$$-\mathcal{L}_\sigma \phi_{\ell,\sigma} = \lambda_{\ell,\sigma}\phi_{\ell,\sigma}, \qquad \mathbb{E}_{p_\sigma}[\phi_{\ell,\sigma}] = 0, \qquad \mathbb{E}_{p_\sigma}[\phi_{\ell,\sigma}\phi_{m,\sigma}] = \delta_{\ell m}, \tag{3}$$

where $\delta_{\ell m}$ is the Kronecker delta. For a rank-r representation, we collect the retained eigenfunctions in the spectral feature map

$$\Phi_\sigma(z) = \big(\phi_{1,\sigma}(z), \ldots, \phi_{r,\sigma}(z)\big)^\top. \tag{4}$$

The low-eigenvalue functions vary slowly within well-connected regions and can change across low-density bottlenecks, making their joint values soft region indicators (Nadler et al., 2005). Appendix A gives the Langevin construction and Rayleigh–Ritz characterization.

We next use these features to construct a positive-semidefinite kernel. For a scale $\tau > 0$, we assign eigenfunction ℓ the spectral weight $e^{-\tau\lambda_{\ell,\sigma}}$ and remove the constant mode, giving

$$k^\star_{\sigma,\tau}(z,z') = \sum_{\ell\geq 1} e^{-\tau\lambda_{\ell,\sigma}}\phi_{\ell,\sigma}(z)\phi_{\ell,\sigma}(z'). \tag{5}$$

The superscript $\star$ denotes the ideal kernel with the full infinite expansion.

In practice, we truncate the expansion after the first r nonconstant eigenfunctions and denote the resulting kernel by $k^{(r)}_{\sigma,\tau}$.

$$k^{(r)}_{\sigma,\tau}(z,z') = \sum_{\ell=1}^{r} e^{-\tau\lambda_{\ell,\sigma}}\phi_{\ell,\sigma}(z)\phi_{\ell,\sigma}(z'). \tag{6}$$

Both kernels are positive semidefinite because their spectral weights are nonnegative. Larger τ suppresses higher-eigenvalue features more strongly, whereas increasing r or decreasing τ retains finer variation. The learned kernel used below approximates this rank-r construction.

The resulting kernel provides a model-adapted geometry: states connected through high-density regions have similar spectral representations, whereas low-density bottlenecks separate them. Reducing MMD under this kernel therefore encourages a batch to occupy the density-connected regions captured by the kernel similarly to p_σ.

3.3 PROFILED SPECTRAL MMD

Spectral MMD measures how well a weighted particle batch represents p_σ. Given fixed particle locations, we solve for simplex profile weights using a regularized spectral MMD objective. The optimized value defines a profiled loss on the current support. Profiling separates representational capacity from particle counts: a point in a low-probability region need not be treated as carrying the same mass as every other point.

For fixed particle locations $Z_\sigma = \{z_\sigma^1, \dots, z_\sigma^N\}$, we optimize the batch weights $\alpha \in \Delta^{N-1}$. Let $A_\sigma \in \mathbb{R}^{N\times r}$ collect the spectrally weighted eigenfunction values. Since each retained eigenfunction has zero mean under p_σ, the squared MMD between the batch weighted by α and p_σ under the rank-r kernel is

$$A_\sigma[i,\ell] = e^{-\tau\lambda_{\ell,\sigma}/2}\phi_{\ell,\sigma}(z_\sigma^i), \qquad i = 1,\dots,N, \quad \ell = 1,\dots,r,$$
$$E_\sigma^{\mathrm{MMD}}(Z_\sigma,\alpha) = \|A_\sigma^\top \alpha\|_2^2 = \sum_{\ell=1}^{r} e^{-\tau\lambda_{\ell,\sigma}} \left(\sum_{i=1}^{N} \alpha_i \phi_{\ell,\sigma}(z_\sigma^i)\right)^2. \tag{7}$$

Finite-rank matching can admit multiple or boundary weight solutions, so we add a negative-entropy regularizer for stability. The regularized objective, optimized profile weights, and profiled loss are

$$\mathcal{J}_\sigma(Z_\sigma,\alpha) = E_\sigma^{\mathrm{MMD}}(Z_\sigma,\alpha) + \tau_\alpha \sum_{i=1}^{N} \alpha_i \log \alpha_i,$$
$$\alpha_\sigma^\star = \operatorname*{arg\,min}_{\alpha\in\Delta^{N-1}} \mathcal{J}_\sigma(Z_\sigma,\alpha), \qquad E_\sigma^{\mathrm{prof}}(Z_\sigma) = \mathcal{J}_\sigma(Z_\sigma,\alpha_\sigma^\star). \tag{8}$$

After optimizing the weights, we use the resulting loss $E_\sigma^{\mathrm{prof}}(Z_\sigma)$ to guide particle movement toward a better representation of p_σ. Here $\tau_\alpha \geq 0$ discourages degenerate profile weights, $0\log 0 = 0$, and $\tau_\alpha = 0$ recovers the unregularized profiled MMD.

3.4 SUPPORT COVERAGE

Spectral MMD depends on the weighted mean of the particle features. Two particles with identical features contribute the same as one particle carrying their combined weight, so a small MMD does not ensure that particles provide distinct information. We therefore add a coverage term that encourages diversity among the particle features.

Let $\widetilde{A}_\sigma \in \mathbb{R}^{N\times r}$ be a row-wise norm-tempered version of A_σ that preserves feature directions while limiting sensitivity to feature scale (Appendix B). We define the coverage Gram matrix

$$K_\sigma = \widetilde{A}_\sigma \widetilde{A}_\sigma^\top \in \mathbb{R}^{N\times N}. \tag{9}$$

The stabilized coverage energy is

$$E_\sigma^{\mathrm{cov}}(Z_\sigma) = -\log\det\left(I_N + \epsilon_{\mathrm{cov}}^{-1} K_\sigma\right), \tag{10}$$

where I_N is the $N \times N$ identity and $\epsilon_{\mathrm{cov}} > 0$ stabilizes the determinant. The determinant provides a stabilized measure of the volume spanned by the particle feature vectors. Minimizing its negative logarithm encourages a larger volume, favoring diverse feature directions over redundant ones.

The combined support-placement objective is

$$E_\sigma^{\mathrm{tot}}(Z_\sigma) = E_\sigma^{\mathrm{prof}}(Z_\sigma) + \beta_\sigma E_\sigma^{\mathrm{cov}}(Z_\sigma), \tag{11}$$

where $\beta_\sigma \geq 0$ controls the contribution of the log-determinant term. We activate it primarily at intermediate noise levels where p_σ retains informative density structure.

Profiled MMD matches target feature moments, while the log-determinant term encourages the support to span complementary spectral directions. Together, they provide a surrogate objective for representative support. The profile weights are optimized for the spectral guidance objective. Once sampling is complete, we choose the returned weights using both spectral and Stein features.

3.5 SPECTRAL–STEIN QUADRATURE WEIGHTS

The functions whose expectations will be needed downstream are unknown, so we take a task-agnostic approach to choosing quadrature weights. Once sampling is complete, we hold the final locations Z_0 fixed and optimize their weights to match target expectations over a chosen function class.

At noise level $\sigma > 0$, the model score transforms a vector-valued test function $f : \mathbb{R}^d \to \mathbb{R}^d$ into a scalar function $g_{f,\sigma} : \mathbb{R}^d \to \mathbb{R}$. Stein's identity gives $\mathbb{E}_{p_\sigma}[g_{f,\sigma}] = 0$. We call these scalar functions Stein features and choose weights that bring their weighted averages close to zero, while also matching the retained spectral features.

These two expectation mismatches give the quadratic terms in the weight objective. The normalized spectral Gram matrix $\widetilde{G}^{\mathrm{sp}}$ is obtained from $A_0 A_0^\top$, while the normalized Stein Gram matrix $\widetilde{G}^{\mathrm{St}}$ is constructed from pairwise inner products of the Stein features. The terms $w^\top \widetilde{G}^{\mathrm{sp}} w$ and $w^\top \widetilde{G}^{\mathrm{St}} w$ measure the corresponding squared weighted expectation mismatches. Appendix C gives the construction and normalization details. The final weights solve

$$w^\star = \operatorname*{arg\,min}_{w \in \Delta^{N-1}} w^\top \left(\widetilde{G}^{\mathrm{sp}} + \rho \widetilde{G}^{\mathrm{St}} \right) w + \mathcal{R}(w), \tag{12}$$

where $\rho \geq 0$ controls the Stein term and $\mathcal{R}$ stabilizes the finite-batch solution through shrinkage toward uniform and a cap on any single weight. We return the weighted measure

$$\widehat{p}_N^{\mathrm{quad}} = \sum_{i=1}^{N} w_i^\star \delta_{z_0^i}. \tag{13}$$

3.6 LEARNING THE NOISE-CONDITIONED SPECTRAL KERNEL

We train a feature network $h_\vartheta(z, \sigma) \in \mathbb{R}^r$ that takes a noisy state z and its noise level σ and returns r scalar feature values. The loss follows three criteria for the desired eigenfunctions in Eq. (3): small eigenvalues, zero mean, and orthonormality under p_σ. For a minibatch at one fixed σ, let $\widehat{m}$, $\widehat{C}$, and $\widehat{D}$ estimate the feature mean, covariance, and matrix of gradient inner products. We use

$$\mathcal{L}_{\mathrm{kernel}} = \mathrm{tr}(\widehat{D}) + \lambda_{\mathrm{mean}} \|\widehat{m}\|_2^2 + \lambda_{\mathrm{orth}} \|\widehat{C} - I_r\|_F^2 + \lambda_{\mathrm{diag}} \|\operatorname{off}(\widehat{D})\|_F^2. \tag{14}$$

The trace term estimates total gradient energy and encourages small eigenvalues. The mean penalty encourages zero mean, while the covariance penalty encourages orthonormality: unit variance for each feature and zero covariance between distinct features. The first three terms can learn mixtures of the leading eigenfunctions. The fourth term penalizes off-diagonal entries of $\widehat{D}$, which estimate mean gradient inner products between distinct features. These expectations vanish for orthonormal eigenfunctions. Appendix A derives these correspondences and the resulting training objective.

We keep the diffusion model fixed and obtain a minibatch at noise level σ by running independent denoising trajectories until they reach σ. We compute all statistics within this minibatch and average the loss over minibatches at sampled noise levels. We treat τ as a kernel-scale hyperparameter and keep it fixed across noise levels.

After training, we center and whiten h_ϑ using independent calibration samples to obtain ψ_ϑ. With $J_\psi = \nabla_z \psi_\vartheta$, we estimate its gradient energy matrix and form

$$\begin{aligned} D_\sigma &= \mathbb{E}_{Z \sim p_\sigma}[J_\psi(Z, \sigma) J_\psi(Z, \sigma)^\top], \\ A_\sigma[i, :] &= \psi_\vartheta(z_\sigma^i, \sigma)^\top e^{-\tau D_\sigma / 2}, \qquad i = 1, \ldots, N. \end{aligned} \tag{15}$$

This A_σ replaces the ideal features in Eq. (7), giving the learned kernel Gram matrix $A_\sigma A_\sigma^\top$ used by profiled MMD and coverage. Appendix D gives the noise-sampling, and estimation details.

3.7 REPRESENTATIVE SAMPLER

Our sampler adds a batch-level correction to the model score $s_{\theta,\sigma}$. At each noise level, we optimize the profile weights for the current particle locations using Eq. (8) and hold them fixed while differentiating E_σ^{tot}. We then use

$$\widetilde{s}_\sigma^i(Z_\sigma) = s_{\theta,\sigma}(z_\sigma^i) - \eta_\sigma \nabla_{z_\sigma^i} E_\sigma^{\mathrm{tot}}(Z_\sigma), \tag{16}$$

Table 1: Controlled-mixture component ablation over 50 paired runs.

Target	N	Method	$E_{\mathrm{MMD}} \downarrow$	$E_{\mathrm{mode}} \downarrow$	Any miss $\downarrow$	Covered $\uparrow$
1D	8	iid uniform	0.570	0.519	0.88	3.58
1D	8	iid, quad.	0.235	0.332	0.88	3.58
1D	8	MMD + quad.	0.057	0.126	0.52	4.48
1D	8	MMD + cov., quad.	**0.047**	**0.104**	**0.32**	**4.68**
1D	12	iid uniform	0.364	0.401	0.66	4.16
1D	12	iid, quad.	0.119	0.184	0.66	4.16
1D	12	MMD + quad.	0.039	0.119	0.24	4.76
1D	12	MMD + cov., quad.	**0.010**	**0.045**	**0.04**	**4.96**
2D	12	iid uniform	0.272	0.374	0.36	3.62
2D	12	iid, quad.	0.063	0.100	0.36	3.62
2D	12	MMD + quad.	0.022	0.072	0.14	3.86
2D	12	MMD + cov., quad.	**0.018**	**0.051**	**0.12**	**3.88**

in place of the individual model score. The usual diffusion sampler then proceeds with these coupled scores. After sampling reaches the clean endpoint, we hold the final locations fixed, compute their quadrature weights using Eq. (12), and return the measure in Eq. (13). Appendix E gives the exact discretization and stochastic extension.

3.8 Sequential and anytime sampling

For sequential sampling, our distribution-matching and coverage objectives naturally extend by conditioning on previous batches, which remain unchanged. Each new batch is optimized to complement the earlier samples and improve the overall representation of the model distribution.

4 Experiments

4.1 Controlled mixtures

We use a symmetric 1D five-mode Gaussian mixture with component masses $(.08, .17, .50, .17, .08)$ and an asymmetric 2D four-mode mixture with masses $(.44, .26, .20, .10)$; Appendix Figure 4 visualizes the targets. Here p_0 is the analytic mixture, and $s_{\theta,\sigma}$ is its exact noisy-marginal score. Clean training samples are drawn directly from the mixture, and component assignments are used only for evaluation. Sampling uses a learned rank-12 spectral map. Figure 1 shows that its leading features recover the broad 2D partitions identified by numerical eigenfunctions; Appendix Figure 5 gives the 1D comparison. Figure 6 shows the effect of each guidance term on the 2D mixture.

We evaluate the weighted measure by spectral MMD—the squared difference between the weighted batch average and target expectation of the numerical spectral features—and by the ℓ_1 error between its represented and true component masses. For support, *Any miss* is the fraction of runs omitting at least one component, and *Covered* is the mean number of components represented. Both targets use 50 paired runs; the 1D mixture uses $N \in \{8, 12\}$ and the 2D mixture uses $N = 12$.

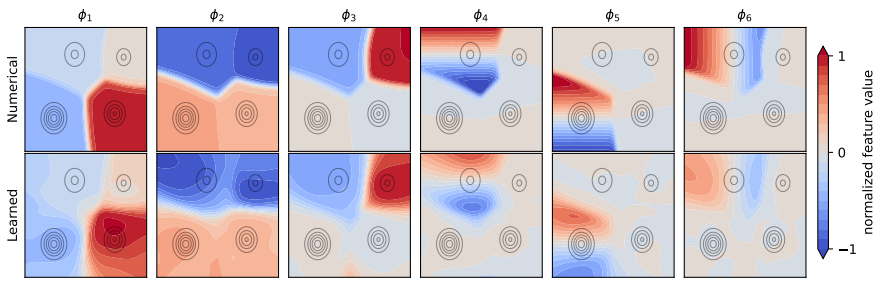


Figure 1: The first six numerical (top) and learned (bottom) slow eigenfunctions of the controlled 2D mixture at $\sigma = 0$, after orthogonal alignment. Contours show p_0; colors are normalized within each column.

Within each target and budget, the rows successively isolate quadrature weighting, MMD guidance, and coverage. They provide complementary gains: quadrature reduces weighted errors, MMD guidance reduces missed modes, and coverage improves both representation and support. In 1D, the full method with $N = 8$ also outperforms iid sampling with $N = 12$ on all four metrics. Appendix F gives the target definitions and numerical eigenbasis construction.

4.2 Image Experiments

We evaluate MNIST (LeCun et al., 1998), Fashion-MNIST (Xiao et al., 2017), EuroSAT-RGB (Helber et al., 2019), CIFAR-10 (Krizhevsky, 2009). We compare with iid sampling, Particle Guidance (Corso et al., 2024), EDDY-RBF (Vinograd et al., 2026), and DiverseFlow (Morshed & Boddeti, 2025). All methods share the same base sampler and initial noise within each run, and none draws an enlarged candidate pool. The four main experiments use frozen latent EDMs and learned rank-16 spectral maps; MNIST and Fashion-MNIST use 32-dimensional latents, while EuroSAT and CIFAR-10 use 128-dimensional latents. In the figures, *Ours* denotes the complete method, including quadrature weights when the metric evaluates a weighted measure.

For evaluation, we draw a large independent iid reference batch from the same frozen diffusion model and use it to estimate the model's average kernel features. We test two embeddings: the diffusion latents and the penultimate layer outputs of a separately trained classifier. In each embedding, reference MMD is the squared distance between the returned batch's weighted kernel-feature average and the reference average. Class TV uses the same classifier to compare their predicted-class distributions. These three metrics use the returned weights and are lower when the batch better represents the model. On MNIST and Fashion-MNIST, we additionally report the number of classes absent from the support and the fraction of runs missing any class. References and classifiers are used only for evaluation. Appendix G gives complete definitions and protocols.

Figure 2 evaluates finite-budget class coverage on MNIST and Fashion-MNIST and shows that our one-shot sampler consistently reduces missing classes. At $N = 20$, the mean falls from 1.31 to 0.13 on MNIST and from 1.40 to 0.33 on Fashion-MNIST, while the corresponding any-miss rates fall from 0.835 to 0.127 and from 0.825 to 0.305. The sequential sampler retains much of this benefit when samples must be added in stages.

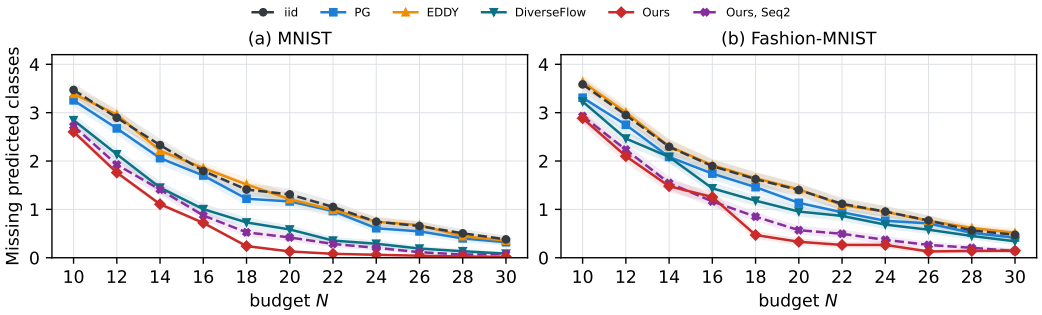


Figure 2: Missing-class count for MNIST and Fashion-MNIST. Seq2 first generates $N/2$ samples and then conditionally adds $N/2$ while keeping the first batch fixed; bands are 95% intervals.

Figure 3 compares three distributional metrics on all four datasets. Moving from latent to classifier embeddings changes the ranking of the baselines, showing that representativeness depends on the evaluation function class. Ours remains strongest overall: it achieves the lowest classifier-embedding MMD at every budget and is best or competitive in latent-reference MMD. It also markedly reduces class TV on MNIST, Fashion-MNIST, and EuroSAT while remaining competitive on CIFAR-10. Methods designed primarily to spread samples—Particle Guidance, EDDY-RBF, and DiverseFlow—can perform worse than iid on these distributional metrics, showing that broader support alone does not guarantee a more representative batch. Appendix G reports additional validation on a pixel-space diffusion model.

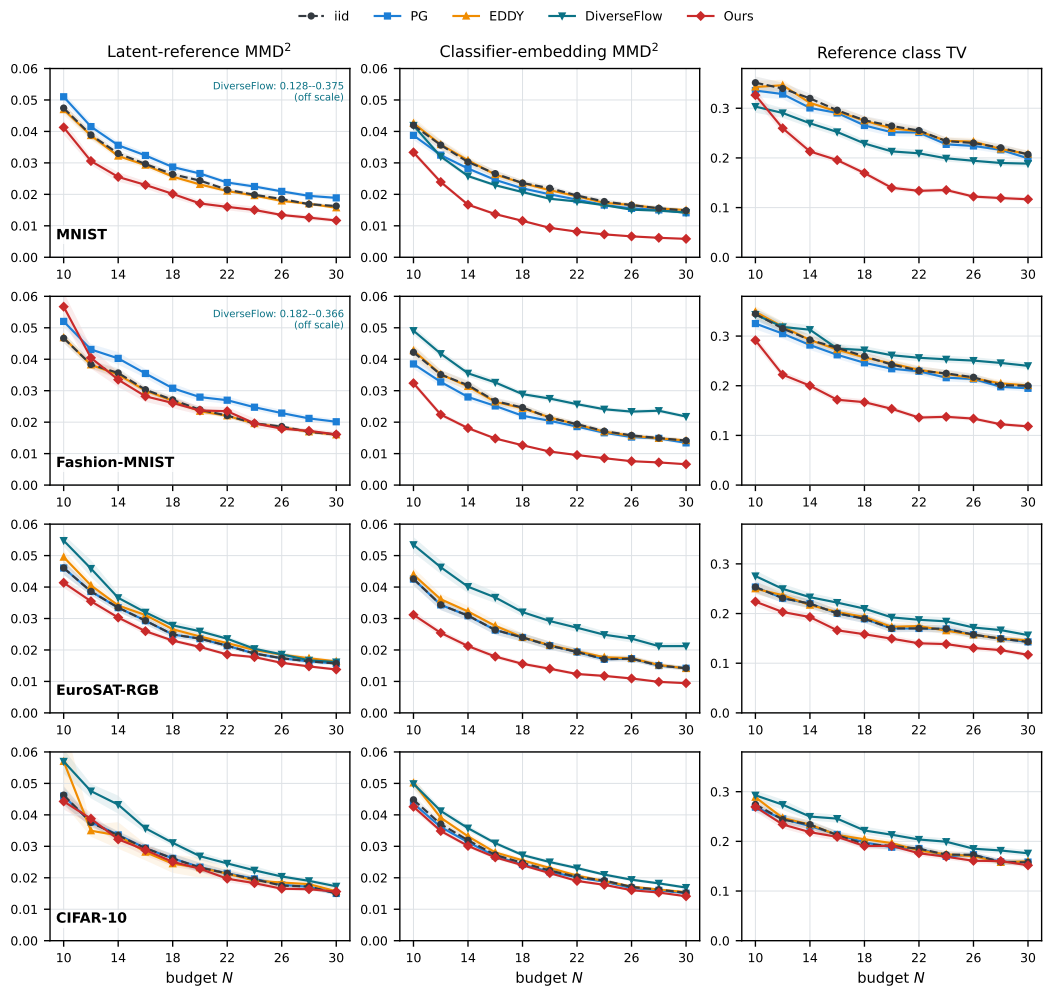


Figure 3: Distributional fidelity across four image datasets. Columns show MMD to independent model references in latent and classifier-embedding spaces, and reference class TV. DiverseFlow is off scale and omitted from the latent-MMD panels for MNIST and Fashion-MNIST. Bands are 95% intervals; lower is better.

4.3 Closed-loop planning in a constructed visual MDP

Setup. We construct a family of finite-horizon MDPs $(\mathcal{S}, \mathcal{A}, P, R, H)$ that share the same states, actions, and transition kernel but differ in their reward functions. A state records the encounter index, health, and whether the agent is choosing a route or responding to a revealed outcome. At each route state, the agent chooses among safe, opportunity, and risky routes. The transition then reveals one of eight route-dependent combinations of hostile, supply, and hazard events. At the resulting response state, the agent chooses to engage or retreat, which determines its reward and health update. Three encounters give a maximum horizon $H = 6$—three route and three response decisions—with earlier termination if health reaches zero. We discount future rewards by $\gamma = 0.99$ per encounter; the route-to-response transition carries no discount.

To make the visual state representation nontrivial, ViZDoom (Kempka et al., 2016) renders each route–outcome state as one of 16 128×128 RGB variants with different camera angles and object positions. Thus the same semantic state has multiple visual realizations. A visual encoder maps these observations to latent states, and an action-conditional diffusion model learns the stochastic route transitions used by MCTS; the dynamics themselves are defined by the constructed MDP. Its known transition kernel allows exact dynamic programming for evaluation, but neither it nor the exact values

are available to the planner. Four reward functions assign different utilities to the same outcomes, testing whether a sampled batch supports accurate expectations across tasks.

Protocol. Each diffusion call returns $N_p = 16$ successors for a stochastic route transition. These samples estimate the chance-node expectation used to compute action values: uniform methods average them, whereas ours uses its returned quadrature weights. At each route-decision state, MCTS replans from the current observation with a total budget $B \in \{48, 96, 192\}$ of diffusion calls distributed across its chance nodes. Search proceeds to the end of the six-decision environment horizon, so its remaining depth decreases throughout the episode. All methods share the learned world, reward and value models, planner, and paired randomness. We evaluate 200 paired closed-loop runs using route regret and action-value MAE (Q-MAE) against exact dynamic-programming values. Route regret measures the loss from selecting a suboptimal route, while Q-MAE measures value-estimation error. Appendix H gives complete experimental details.

Table 2: Route regret / Q-MAE across stochastic-replan call budgets; lower is better. Bold marks the best value, and underlining marks values worse than iid at the same profile and budget. Additional metrics are in Appendix H.

Method	B	Balanced	Supply rush	Survivor escort	Combat resupply
iid	48	.123/.319	.095/.479	.120/.548	.340/.737
	96	.124/.299	.092/.439	.120/.502	.311/.701
	192	.113/.292	.086/.435	.111/.498	.265/.682
Particle Guidance	48	.132/.275	.070/.410	.098/.467	.374/.626
	96	.137/.264	.073/.379	.098/.434	.356/.604
	192	.124/.259	.066/.378	.083/.423	.314/.600
EDDY-RBF	48	.200/.404	.081/.589	.099/.696	.594/.868
	96	.187/.378	.085/.533	.092/.645	.557/.821
	192	.172/.370	.082/.532	.081/.643	.550/.804
DiverseFlow	48	.124/.311	.092/.473	.116/.540	.355/.729
	96	.118/.294	.087/.434	.115/.495	.306/.697
	192	.112/.286	.085/.432	.108/.489	.291/.681
Ours	48	**.047/.109**	**.049/.157**	**.043/.209**	**.020/.291**
	96	**.052/.114**	**.048/.150**	**.036/.209**	**.020/.282**
	192	**.043/.110**	**.038/.146**	**.025/.196**	**.010/.278**

Results. Table 2 shows that ours has the lowest route regret and Q-MAE in all 12 profile–budget settings, with paired 95% intervals favoring it over every baseline. It also gives the highest route optimality and expected policy value throughout (Appendix H). At $B = 96$, its route optimality exceeds the strongest baseline by 5.2–21.1 percentage points across the four profiles. Thus more representative diffusion batches improve not only chance-node expectations, but also multistep value estimates and closed-loop decisions. Some diversity-oriented methods underperform iid, showing that spreading successors without representing their probabilities can bias planning.

A root-level ablation replaces our quadrature weights with uniform weights. At $B = 192$, for survivor escort this increases Q-MAE from .196 to .351, which highlights the importance of weighting.

5 Conclusion

We introduced a method for representing a diffusion model with a small weighted batch. Model-adapted sample relations guide support toward complementary outcomes, while quadrature weights improve expectation estimation. Experiments show better finite-budget coverage and representation; in MCTS, the weighted batches yield more accurate values and decisions. These results show that diversity alone is insufficient: diversity-oriented sampling can underperform iid when a downstream task exposes its structural bias, motivating support placement and weighting together. The method uses only the pretrained model and its generated samples to construct a weighted representation of the model's output distribution.

AI USE STATEMENT

We used AI tools to polish the writing, assist with the literature survey, help brainstorm and discuss possible solutions, check the correctness of the mathematical formulation, and support coding and plot generation. We have reviewed all AI-assisted work. All decisions regarding methods and experiments presented in the paper is made by human authors. LLM-generated code was verified and tested for correctness. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

Our experiments do not involve human-subject studies. Experiments are done on publicly available datasets or in simulation which is fully specified in the paper.

REPRODUCIBILITY STATEMENT

We disclose full details of the method, including the training and sampling procedures, hyperparameters, and evaluation metrics in the main text and appendices.

REFERENCES

I. Antonoglou, J. Schrittwieser, S. Ozair, T. K. Hubert, and D. Silver. Planning in stochastic environments with a learned model. In *ICLR*, 2022.

S. M. R. Arnold, P. L'Ecuyer, L. Chen, Y.-f. Chen, and F. Sha. Policy learning and evaluation with randomized quasi-Monte Carlo. In *AISTATS*, 2022.

M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *NeurIPS*, 2001.

W. Y. Chen, L. Mackey, J. Gorham, F.-X. Briol, and C. Oates. Stein points. In *ICML*, 2018.

Y. Chen, M. Welling, and A. Smola. Super-samples from kernel herding. In *UAI*, 2010.

K. Chua, R. Calandra, R. McAllister, and S. Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *NeurIPS*, 2018.

R. R. Coifman and S. Lafon. Diffusion maps. *Applied and Computational Harmonic Analysis*, 21(1):5–30, 2006.

G. Corso, Y. Xu, V. De Bortoli, R. Barzilay, and T. Jaakkola. Particle guidance: non-I.I.D. diverse sampling with diffusion models. In *ICLR*, 2024.

A. Couëtoux, M. Milone, M. Brendel, H. Doghmen, M. Sebag, and O. Teytaud. Continuous rapid action value estimates. In *Asian Conference on Machine Learning*, 2011.

R. Dwivedi and L. Mackey. Kernel thinning. *Journal of Machine Learning Research*, 2024.

A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 2012.

P. Helber, B. Bischke, A. Dengel, and D. Borth. EuroSAT: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 12(7):2217–2226, 2019.

F. Huszár and D. Duvenaud. Optimally-weighted herding is Bayesian quadrature. In *UAI*, 2012.

T. Karras, M. Aittala, T. Aila, and S. Laine. Elucidating the design space of diffusion-based generative models. In *NeurIPS*, 2022.

M. Kearns, Y. Mansour, and A. Y. Ng. A sparse sampling algorithm for near-optimal planning in large Markov decision processes. In *IJCAI*, 1999.

M. Kempka, M. Wydmuch, G. Runc, J. Toczek, and W. Jaśkowski. ViZDoom: A Doom-based AI research platform for visual reinforcement learning. In *IEEE Conference on Computational Intelligence and Games*, 2016.

L. Kocsis and C. Szepesvári. Bandit based Monte-Carlo planning. In *European Conference on Machine Learning*, 2006.

A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

Q. Liu and J. D. Lee. Black-box importance sampling. In *AISTATS*, 2017.

S. Mak and V. R. Joseph. Support points. *Annals of Statistics*, 2018.

A. Mardt, L. Pasquali, H. Wu, and F. Noé. VAMPnets for deep learning of molecular kinetics. *Nature Communications*, 9:5, 2018.

M. M. Morshed and V. N. Boddeti. DiverseFlow: sample-efficient diverse mode coverage in flows. In *CVPR*, 2025.

B. Nadler, S. Lafon, R. R. Coifman, and I. G. Kevrekidis. Diffusion maps, spectral clustering and eigenfunctions of Fokker–Planck operators. In *NeurIPS*, 2005.

D. Pfau, S. Petersen, A. Agarwal, D. G. T. Barrett, and K. L. Stachenfeld. Spectral inference networks: unifying deep and spectral learning. In *ICLR*, 2019.

J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, T. Lillicrap, and D. Silver. Mastering Atari, Go, chess and shogi by planning with a learned model. *Nature*, 588:604–609, 2020.

Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole. Score-based generative modeling through stochastic differential equations. In *ICLR*, 2021.

Z. Sunberg and M. Kochenderfer. Online algorithms for POMDPs with continuous state, action, and observation spaces. In *ICAPS*, pp. 259–263, 2018.

G. Vinograd, I. Achituve, and E. Fetaya. Diverse sampling in diffusion models with marginal preserving particle guidance. arXiv preprint, 2026.

J. Wu, Z. Wan, X. Yu, Y. Yang, B. An, I. Tsang, and Y. You. Letting trajectories spread: quality-preserving control for diverse flow matching. arXiv preprint arXiv:2510.09060, 2025.

H. Xiao, K. Rasul, and R. Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747, 2017.

W. Zhang, T. Li, and C. Schütte. Solving eigenvalue PDEs of metastable diffusion processes using artificial neural networks. *Journal of Computational Physics*, 465:111377, 2022.

A LANGEVIN OPERATOR AND GRADIENT ENERGY

Spectral features can be learned by minimizing density-weighted gradient energy under centering and orthonormality constraints.

For a fixed noise level σ, consider the auxiliary Langevin diffusion

$$dZ_u = \nabla_z \log p_\sigma(Z_u)du + \sqrt{2}\, dW_u. \tag{17}$$

Here u is auxiliary time, $Z_u \in \mathbb{R}^d$ is the process state, and W_u is standard d-dimensional Brownian motion; σ remains fixed throughout the construction. This process explains the feature geometry, and training does not require simulating its trajectories. Under standard regularity conditions, the auxiliary process has stationary distribution p_σ and generator $\mathcal{L}_\sigma$. For $\sigma > 0$, the exact-score assumption gives its network-score form in Eq. (2). At $\sigma = 0$, the same construction defines $\mathcal{L}_0 f = \Delta f + (\nabla_z \log p_0)^\top \nabla f$. The auxiliary Langevin time u is distinct from the changing noise level along the generative sampling trajectory.

To connect this process to density connectivity, consider a scalar function f along a stationary Langevin trajectory. We seek functions that change slowly under the process: Langevin trajectories

move readily within high-density connected regions but cross low-density bottlenecks less often. Integration by parts connects this behavior to gradient energy:

$$\langle f, -\mathcal{L}_\sigma f \rangle_{p_\sigma} = \mathbb{E}_{p_\sigma}\left[\|\nabla_z f(Z)\|_2^2\right]. \tag{18}$$

Here $\langle f, g\rangle_{p_\sigma} = \mathbb{E}_{p_\sigma}[fg]$. Small gradient energy means small mean-squared changes in f over short Langevin time intervals. Because this gradient energy is weighted by p_σ, changes in f are costly in high-density regions but cheaper across low-density bottlenecks.

Applying the identity to an eigenfunction from Eq. (3) gives

$$\mathbb{E}_{p_\sigma}\left[\|\nabla_z \phi_{\ell,\sigma}(Z)\|_2^2\right] = \langle \phi_{\ell,\sigma}, -\mathcal{L}_\sigma \phi_{\ell,\sigma}\rangle_{p_\sigma} = \lambda_{\ell,\sigma}\mathbb{E}_{p_\sigma}[\phi_{\ell,\sigma}^2] = \lambda_{\ell,\sigma}. \tag{19}$$

The last equality uses unit variance and zero mean. Thus each normalized eigenfunction's gradient energy equals its eigenvalue.

To see why minimizing gradient energy selects leading eigenfunctions, expand a zero-mean, unit-variance function f with finite gradient energy in the eigenbasis of Eq. (3):

$$\begin{aligned} f &= \sum_{j\geq 1} c_j \phi_{j,\sigma}, \qquad \sum_{j\geq 1} c_j^2 = 1, \\ \mathbb{E}_{p_\sigma}[\|\nabla_z f\|_2^2] &= \sum_{j\geq 1} \lambda_{j,\sigma} c_j^2. \end{aligned} \tag{20}$$

The energy is a weighted average of the nonconstant eigenvalues, so its minimum is the smallest one. Requiring orthogonality to the preceding eigenfunctions sets their coefficients to zero and selects the next smallest eigenvalue. This gives the sequential characterization

$$\phi_{\ell,\sigma} \in \underset{\substack{\mathbb{E}_{p_\sigma}[f]=0 \\ \mathbb{E}_{p_\sigma}[f^2]=1 \\ \mathbb{E}_{p_\sigma}[f\phi_{j,\sigma}]=0,\ j<\ell}}{\arg\min} \mathbb{E}_{p_\sigma}\left[\|\nabla_z f(Z)\|_2^2\right]. \tag{21}$$

The zero-mean constraint excludes the constant solution, the unit-variance constraint fixes the function scale, and orthogonality to preceding solutions yields distinct features in increasing order of energy (Pfau et al., 2019; Zhang et al., 2022). By the Rayleigh–Ritz principle, these minimizers are the slow eigenfunctions in Eq. (3), and their minimum energies are the corresponding eigenvalues.

For r features learned jointly, let $h : \mathbb{R}^d \to \mathbb{R}^r$ and $J_h = \nabla_z h$. Their gradient energy matrix is $D_h = \mathbb{E}_{p_\sigma}[J_h J_h^\top]$, whose trace is the total energy $\sum_{j=1}^r \mathbb{E}_{p_\sigma}[\|\nabla_z h_j\|_2^2]$. The corresponding constrained problem is

$$\min_h \ \mathbb{E}_{p_\sigma}\left[\|\nabla_z h(Z)\|_F^2\right] \quad \text{subject to} \quad \mathbb{E}_{p_\sigma}[h] = 0, \qquad \mathbb{E}_{p_\sigma}[hh^\top] = I_r. \tag{22}$$

The minimum total gradient energy is $\sum_{\ell=1}^r \lambda_{\ell,\sigma}$, attained by an orthonormal basis of a leading r-dimensional eigenspace. The constraints prevent constant, vanishing, or duplicate outputs from reducing the objective trivially.

This yields the first three terms of Eq. (14). The empirical trace $\mathrm{tr}(\widehat{D})$ estimates total gradient energy and therefore targets small eigenvalues under the constraints. The mean penalty relaxes $\mathbb{E}_{p_\sigma}[h] = 0$. At zero mean, the covariance equals $\mathbb{E}_{p_\sigma}[hh^\top]$, so the covariance penalty relaxes orthonormality. Finite penalties approximate these constraints rather than enforcing them exactly.

The auxiliary fourth term concerns the choice of basis. Integration by parts also gives

$$\mathbb{E}_{p_\sigma}[\nabla_z \phi_{\ell,\sigma}^\top \nabla_z \phi_{m,\sigma}] = \langle \phi_{\ell,\sigma}, -\mathcal{L}_\sigma \phi_{m,\sigma}\rangle_{p_\sigma} = \lambda_{m,\sigma}\delta_{\ell m}. \tag{23}$$

Hence the gradient energy matrix is diagonal in an eigenfunction basis. An orthogonal rotation preserves total gradient energy and the mean and covariance penalties. The first three terms therefore do not select a particular rotation of the learned features. The penalty $\|\operatorname{off}(\widehat{D})\|_F^2$ additionally encourages a basis with a diagonal gradient energy matrix. Appendix D gives the empirical estimators and calibration procedure used to approximate this construction.

B COVERAGE-FEATURE NORMALIZATION

Feature lengths affect the log-determinant coverage objective. We preserve feature directions and control lengths by clipping extreme norms, dividing by a reference root-mean-square (RMS) norm, and applying the power γ_{norm}.

Let $a_\sigma(z) = e^{-\tau D_\sigma/2}\psi_\vartheta(z,\sigma)$ denote the learned spectral feature vector from Eq. (15). Using independent calibration samples $X_\sigma \sim p_\sigma$, we estimate the reference RMS norm and clipping bounds:

$$\bar{r}_\sigma = \left(\mathbb{E}\|a_\sigma(X_\sigma)\|_2^2\right)^{1/2}, \qquad (b_\sigma^-, b_\sigma^+) = \mathrm{Quantile}_{(\delta_{\mathrm{cov}},1-\delta_{\mathrm{cov}})}\left(\|a_\sigma(X_\sigma)\|_2\right). \tag{24}$$

Here $\delta_{\mathrm{cov}} \in (0, 1/2)$ sets the clipping quantiles. At the current particles, let $r_{i,\sigma} = \|A_\sigma[i,:]\|_2$ and $u_{i,\sigma} = A_\sigma[i,:]/\max(r_{i,\sigma}, \epsilon_{\mathrm{dir}})$, where a small $\epsilon_{\mathrm{dir}} > 0$ prevents division by zero. The rescaled coverage row is

$$\widetilde{A}_\sigma[i,:] = \left(\frac{\mathrm{clip}(r_{i,\sigma}; b_\sigma^-, b_\sigma^+)}{\bar{r}_\sigma}\right)^{\gamma_{\mathrm{norm}}} u_{i,\sigma}, \qquad \gamma_{\mathrm{norm}} \in [0,1]. \tag{25}$$

Here clip clamps norms to $[b_\sigma^-, b_\sigma^+]$. Setting $\gamma_{\mathrm{norm}} = 0$ uses normalized directions; $\gamma_{\mathrm{norm}} = 1$ retains the clipped norms relative to the reference RMS norm. Intermediate values compress differences in feature length.

We use $\widetilde{A}_\sigma$ only in the coverage Gram matrix. Profiled MMD and endpoint spectral quadrature use the original features A_σ and A_0, respectively.

C SPECTRAL–STEIN QUADRATURE DETAILS

Why the weights work. At a fixed noise level $\sigma > 0$, Stein's identity states that

$$\mathbb{E}_{p_\sigma}\left[\nabla\cdot f + s_{\theta,\sigma}^\top f\right] = 0$$

for any smooth vector-valued function f satisfying the boundary and integrability conditions. Thus, given such a function f, we can construct a new scalar function $g_f(z) = \nabla\cdot f(z) + s_{\theta,\sigma}(z)^\top f(z)$ whose target expectation is known: $\mathbb{E}_{p_\sigma}[g_f] = 0$. We call these constructed functions *Stein features*.

For fixed samples $z_\sigma^1, \ldots, z_\sigma^N$, the weighted empirical expectation of a Stein feature is $\sum_i w_i g_f(z_\sigma^i)$. Since its target value is zero, we choose weights that make this weighted average as close to zero as possible. To match a family of features together, we collect them in a feature map ξ_σ and minimize the squared expectation mismatch over $w \in \Delta^{N-1}$. Expanding this squared norm gives

$$\left\|\sum_i w_i \xi_\sigma(z_\sigma^i)\right\|^2 = \sum_{i,j} w_i w_j \langle \xi_\sigma(z_\sigma^i), \xi_\sigma(z_\sigma^j)\rangle = w^\top G_\sigma^{\mathrm{St}} w.$$

Here each entry $G_\sigma^{\mathrm{St}}[i,j]$ is the inner product of the Stein feature vectors at samples i and j. Thus minimizing expectation mismatch is equivalent to minimizing this quadratic form. This motivates the Stein component of Eq. (12), which also includes the spectral term and weight regularization.

To connect these positive-noise tests to the returned endpoint measure, we use the additive Gaussian parameterization of the diffusion model. In this parameterization, if $X_0 \sim p_0$ and $\varepsilon \sim \mathcal{N}(0, I)$ are independent, then $X_0 + \sigma\varepsilon \sim p_\sigma$. Equivalently, for $\sigma > 0$, $p_\sigma = p_0 * \mathcal{N}(0, \sigma^2 I)$, where $*$ denotes convolution. Define the noise-averaged endpoint feature

$$h_\sigma(x) = \mathbb{E}_\varepsilon[\xi_\sigma(x + \sigma\varepsilon)].$$

Under the same integrability conditions, iterated expectation gives

$$\mathbb{E}_{X_0\sim p_0}[h_\sigma(X_0)] = \mathbb{E}_{Z\sim p_\sigma}[\xi_\sigma(Z)] = 0.$$

Thus positive-noise Stein tests provide known expectation constraints for the model's endpoint distribution without requiring a score evaluation at $\sigma = 0$.

How we compute the weights. We only need the pairwise inner products to evaluate the quadratic form. To see how a base kernel computes them, consider one dimension and suppress σ. Start with scalar feature functions f_j and their kernel

$$q(x, y) = \sum_j f_j(x) f_j(y).$$

The Stein construction transforms each feature into $g_j(x) = f_j'(x) + s(x) f_j(x)$. The inner product of these transformed features is $k^{\mathrm{St}}(x, y) = \sum_j g_j(x) g_j(y)$. Derivatives of q give the feature products needed to evaluate this sum; for example,

$$\partial_y q(x, y) = \sum_j f_j(x) f_j'(y).$$

Expanding the transformed products therefore gives

$$\begin{aligned} k^{\mathrm{St}}(x, y) &= \sum_j [f_j'(x) + s(x) f_j(x)][f_j'(y) + s(y) f_j(y)] \\ &= s(x)s(y)q(x, y) + s(x)\partial_y q(x, y) \\ &\quad + s(y)\partial_x q(x, y) + \partial_x \partial_y q(x, y). \end{aligned}$$

A positive-semidefinite kernel admits a feature representation, so we can choose q directly without listing its possibly infinitely many features. With a twice continuously differentiable kernel, the same calculation uses kernel derivatives to compute the Stein feature inner products (Chen et al., 2018). In d dimensions, applying the construction along each coordinate and summing gives the Gram matrix below. For the current batch and base kernel q_σ, abbreviate $s_i = s_{\theta,\sigma}(z_\sigma^i)$ and $q_{ij} = q_\sigma(z_\sigma^i, z_\sigma^j)$. Write $\nabla_1 q_{ij}$ and $\nabla_2 q_{ij}$ for derivatives with respect to the first and second kernel arguments, evaluated at this pair. The resulting computable Gram matrix is

$$G_\sigma^{\mathrm{St}}[i, j] = s_i^\top s_j q_{ij} + s_i^\top \nabla_2 q_{ij} + s_j^\top \nabla_1 q_{ij} + \mathrm{tr}(\nabla_1 \nabla_2 q_{ij}). \tag{26}$$

The four terms combine the score and derivative at each of the two points.

We use an equal mixture of M_q inverse-multiquadric kernels,

$$q_\sigma(z, z') = \frac{1}{M_q} \sum_{m=1}^{M_q} \left(c_{m,\sigma}^2 + \|z - z'\|_2^2\right)^{-1/2}, \tag{27}$$

where $c_{m,\sigma} > 0$ are dimension- and noise-scaled bandwidths; controlled mixtures instead use each perturbed batch's median pairwise distance to set the bandwidths. Because Eq. (26) acts on the full state rather than only on the learned rank-r features, it supplies tests beyond the finite spectral span.

For final samples $x_i = z_0^i$, we set $G^{\mathrm{sp}} = A_0 A_0^\top$. To avoid evaluating the score only at the clean endpoint, draw independent $\varepsilon_i^{(b)} \sim \mathcal{N}(0, I)$ and form $x_{i,\sigma}^{(b)} = x_i + \sigma \varepsilon_i^{(b)}$ for fixed positive noise levels $\sigma \in \mathcal{S}$. Applying Eq. (26) to each perturbed batch gives the raw matrix $G_{\sigma,b}^{\mathrm{St}}$. We divide each Gram matrix by its mean diagonal value, with a small floor to avoid division by zero. Writing $d_G(H) = \max\{|N^{-1} \mathrm{tr}(H)|, \epsilon_G\}$, we use

$$\begin{aligned} \widetilde{G}^{\mathrm{sp}} &= \frac{G^{\mathrm{sp}}}{d_G(G^{\mathrm{sp}})}, \\ \widetilde{G}^{\mathrm{St}} &= \frac{1}{|\mathcal{S}|B} \sum_{\sigma \in \mathcal{S}} \sum_{b=1}^{B} \frac{G_{\sigma,b}^{\mathrm{St}}}{d_G(G_{\sigma,b}^{\mathrm{St}})} + \epsilon_{\mathrm{St}} I_N, \end{aligned} \tag{28}$$

where $\epsilon_G = 10^{-8}$ and $\epsilon_{\mathrm{St}} = 10^{-7}$ stabilize the scale and diagonal, respectively. Thus each noisy Gram is normalized before averaging. This puts their quadratic discrepancies on comparable scales before applying ρ in Eq. (12). For fixed endpoints and weights, averaging unnormalized squared noisy feature sums retains a perturbation-variance term in addition to the squared noise-averaged feature sum. We use the normalized finite-noise average as a practical surrogate for the endpoint tests above. For regularization, we add $\lambda_{\mathrm{unif}} \|w - N^{-1}\mathbf{1}\|_2^2$ and impose $w_i \le \kappa/N$, with $\lambda_{\mathrm{unif}} \ge 0$ and $\kappa \ge 1$. These are the uniform-shrinkage penalty and weight cap represented by $\mathcal{R}(w)$. With the final locations fixed, we solve Eq. (12) once for the returned weights; they do not participate in sampling or location refinement.

D FEATURE TRAINING AND CALIBRATION

This appendix gives the sampling, estimation, and calibration details for Eq. (14); Appendix A derives the objective.

Training samples and noise levels. We keep the diffusion model fixed. At each feature update, we draw one log-uniform noise level from each log-noise bin. For each sampled noise level σ, we obtain a minibatch $(z_\sigma^i)_{i=1}^B$ by running independent denoising trajectories until they reach σ. All examples in that minibatch share exactly the same σ. Pooling different noise levels would allow noise dependence alone to supply apparent feature variance. We compute Eq. (14) separately for each minibatch and average the losses. Only h_ϑ is updated; it conditions on σ through a fixed sinusoidal embedding.

Empirical estimates. For one such fixed-σ minibatch $(z_\sigma^i)_{i=1}^B$, let $h_i = h_\vartheta(z_\sigma^i, \sigma)$. Its empirical mean and covariance are

$$\widehat{m} = \frac{1}{B}\sum_{i=1}^{B} h_i, \qquad \widehat{C} = \frac{1}{B}\sum_{i=1}^{B}(h_i - \widehat{m})(h_i - \widehat{m})^\top. \tag{29}$$

To estimate the gradient energy matrix without forming the full Jacobian, write $\chi_\vartheta = h_\vartheta$ and $J_\chi(z,\sigma) = \nabla_z \chi_\vartheta(z,\sigma) \in \mathbb{R}^{r\times d}$. Draw independent probe directions V satisfying $\mathbb{E}[VV^\top] = I_d$ and compute

$$a(z,\sigma;V) = J_\chi(z,\sigma)V \in \mathbb{R}^r. \tag{30}$$

Since $\mathbb{E}_V[aa^\top] = J_\chi J_\chi^\top$, n_{probe} probes per example give

$$\widehat{D}_{\chi,\sigma} = \frac{1}{Bn_{\text{probe}}}\sum_{i=1}^{B}\sum_{k=1}^{n_{\text{probe}}} a_{ik}a_{ik}^\top. \tag{31}$$

Here $a_{ik} = a(z_\sigma^i, \sigma; V_{ik})$. This is the estimator $\widehat{D}$ in Eq. (14); its trace is unbiased for the gradient energy. Probe noise adds variance to the squared off-diagonal penalty, so that auxiliary loss is biased upward. Final spectral weights are estimated from independent post-training calibration rather than this training penalty.

Post-training calibration. After optimization, we freeze h_ϑ and use a larger independent calibration sample to estimate

$$\begin{aligned} m_\sigma &= \mathbb{E}_{p_\sigma}[h_\vartheta(Z,\sigma)], \\ C_\sigma &= \mathbb{E}_{p_\sigma}\left[(h_\vartheta(Z,\sigma) - m_\sigma)(h_\vartheta(Z,\sigma) - m_\sigma)^\top\right]. \end{aligned} \tag{32}$$

Define the unrotated whitened feature

$$\psi_\vartheta^{(0)}(z,\sigma) = (C_\sigma + \epsilon_{\text{white}} I_r)^{-1/2}(h_\vartheta(z,\sigma) - m_\sigma), \qquad \epsilon_{\text{white}} > 0, \tag{33}$$

where the ridge ϵ_{white} stabilizes directions with small calibration variance. We then estimate its gradient energy matrix $D_\sigma^{(0)}$ as in Eq. (15). Let

$$D_\sigma^{(0)} = R_\sigma \operatorname{diag}(\widehat{\lambda}_{1,\sigma}, \ldots, \widehat{\lambda}_{r,\sigma}) R_\sigma^\top. \tag{34}$$

The stored inference quantities are

$$\psi_\vartheta(z,\sigma) = R_\sigma^\top \psi_\vartheta^{(0)}(z,\sigma), \qquad D_\sigma = \operatorname{diag}(\widehat{\lambda}_{1,\sigma}, \ldots, \widehat{\lambda}_{r,\sigma}), \qquad H_\sigma = e^{-\tau D_\sigma}. \tag{35}$$

The simultaneous rotation of the features and gradient energy matrix preserves the kernel Gram matrix, so this diagonal storage is equivalent to the matrix-exponential construction in Eq. (15). We recalibrate these quantities on the complete sampler noise grid, including its lowest-noise endpoint, and store the norm statistics from Appendix B. These statistics are held fixed during sampling, so no minibatch calibration enters the guidance Jacobian.

E REPRESENTATIVE SAMPLER DETAILS

The generation budget is the number of returned particles, with all methods using the same base sampling schedule; it does not imply equal wall-clock computation. Our correction additionally requires feature and score evaluations, weight optimization, and population-objective gradients at active noise levels.

For the deterministic EDM specialization, let $\widehat{z}_\theta(z,\sigma)$ denote the trained denoiser, with implemented score $s_{\theta,\sigma}(z) = (\widehat{z}_\theta(z,\sigma) - z)/\sigma^2$ for $\sigma > 0$. One probability-flow Euler step from σ to $\sigma^- < \sigma$ is

$$\bar{z}^i_{\sigma^-} = z^i_\sigma + (\sigma^- - \sigma)\frac{z^i_\sigma - \widehat{z}_\theta(z^i_\sigma,\sigma)}{\sigma}. \tag{36}$$

Applying the same Euler step to the modified score in Section 3.7 gives

$$z^i_{\sigma^-} = z^i_\sigma + \sigma(\sigma - \sigma^-)\widetilde{s}^i_\sigma(Z_\sigma) = \bar{z}^i_{\sigma^-} - \eta_\sigma\sigma(\sigma - \sigma^-)\nabla_{z^i_\sigma} E^{\mathrm{tot}}_\sigma(Z_\sigma). \tag{37}$$

Both score and guidance are evaluated at the current particles and current noise level; $\bar{z}^i_{\sigma^-}$ denotes the unguided part of this update. For one-shot inference, initialize N particles independently from the EDM highest-noise reference law. At each active noise level, evaluate the features on Z_σ, solve Eq. (8), hold the optimized weights fixed while differentiating E^{tot}_σ, and apply Eq. (37). Set $\eta_\sigma = 0$ outside the guidance window. After the final denoiser output, hold the generated support fixed and solve Eq. (12). The support can therefore be evaluated with uniform weights or returned with the quadrature weights in Eq. (13).

For a stochastic reverse sampler, replace the score in its drift by the same modified score and retain its diffusion coefficient and random increment. The resulting guidance coefficient follows from that sampler's discretization. We use this extension only where stated; the image experiments use deterministic base transitions.

Practical approximation. The theoretical sampling interpretation assumes that the reverse process starts from $p_{\sigma_{\max}}$ and follows the model's denoising dynamics exactly, yielding the marginals p_σ and the output distribution p_0. In implementation, the highest-noise reference distribution can approximate the required terminal marginal, finite solver steps and the final denoiser approximate the ideal reverse process, and the learned score can depart from the exact marginal score. Consequently, the numerical sampler's intermediate and endpoint distributions need not equal the ideal marginals. In particular, the implemented score's Stein features can have nonzero target means. These are approximation errors relative to the model target; the method still uses only the pretrained model and its generated samples.

Finite feature rank, imperfect feature learning, estimated calibration statistics, and finitely many noise draws also approximate the population construction. The objectives remain well defined: feature and Stein Gram matrices are positive semidefinite even with an approximate score, and shrinkage, weight caps, and Gram stabilization regularize the finite optimization. Model-generated reference statistics determine the spectral target, while the score supplies additional tests for weight assignment. The method therefore retains a concrete criterion for representing the model with a small batch. Its practical accuracy depends on these approximations and on how well the chosen tests capture the expectations of interest.

F CONTROLLED MIXTURES AND NUMERICAL EIGENBASIS

Target definitions. Both targets are Gaussian mixtures, with component probabilities π_k, means μ_k, and isotropic variances v_k:

$$q(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x;\mu_k, v_k I_d). \tag{38}$$

The 1D target has $K = 5$, means $(-4,-2,0,2,4)$, probabilities $(.08,.17,.50,.17,.08)$, and variance $v_k = 0.10$ for every component; we use $p_0 = q$ directly. The 2D target has $K = 4$, means $(-2.25,-1.35)$, $(1.85,-1.10)$, $(-0.85,2.20)$, and $(2.45,2.05)$, with probabilities $(.44,.26,.20,.10)$ and variances $(.20,.16,.24,.12)$ in the same order. For this target, p_0 is the law obtained by subtracting the mixture mean and dividing each coordinate by its marginal standard deviation. Figure 4 shows the two targets. At noise level σ, the target density is $p_\sigma = p_0 * \mathcal{N}(0,\sigma^2 I_d)$.

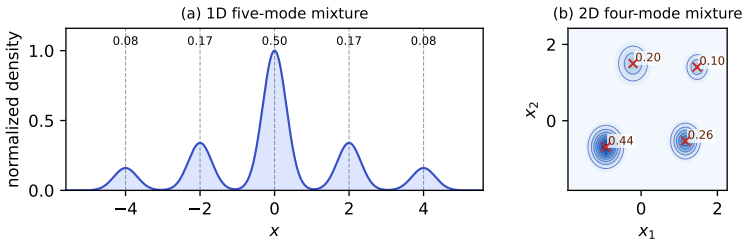


Figure 4: Controlled targets at $\sigma = 0$. Labels give component probabilities.

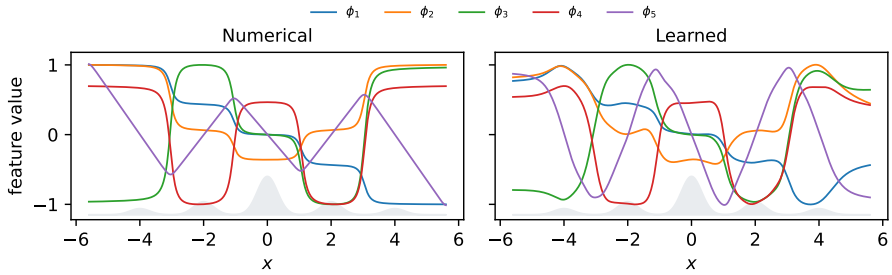


Figure 5: Numerical and learned spectral features for the controlled 1D mixture at $\sigma = 0$, after orthogonal alignment.

Numerical eigenbasis. At each evaluated noise level, we compute reference eigenfunctions and eigenvalues using a finite-volume discretization of the target Langevin operator on a regular grid with zero-flux boundary conditions. We use a sparse Lanczos eigensolver and retain the 12 eigenfunctions with the smallest eigenvalues after excluding the constant mode. We center and normalize them under the discretized target distribution and interpolate their values between grid points. They provide an independent numerical reference for evaluation and visualization; sampling guidance uses the learned feature map.

Figure 5 compares the learned and numerical 1D features after orthogonal alignment; Figure 1 provides the corresponding 2D comparison.

Figure 6 illustrates the 2D guidance mechanism: the score favors a likely region, while the set-level objectives redirect the population toward a missing mode.

G IMAGE EVALUATION DETAILS

All images are normalized to $[-1, 1]$. The method reserves classifiers for evaluation and excludes them from model and kernel training, hyperparameter selection, and sampling. Some reported image results used classifier-based development metrics for hyperparameter selection and therefore do not meet this restriction.

Let $x_1, \dots, x_N$ be the generated images and $\widehat{P}_N = \sum_{i=1}^{N} w_i \delta_{x_i}$ their returned measure. *Ours* uses returned quadrature weights for distributional metrics. Baselines and explicitly uniform-weight variants use $w_i = 1/N$. Support-coverage metrics do not use weights. To evaluate class allocation, we report

$$\mathrm{TV}_{\mathrm{cls}}(\widehat{P}_N, P_{\mathrm{ref}}) = \frac{1}{2} \sum_{c=1}^{C} \left| \widehat{m}_c - \pi_c^{\mathrm{ref}} \right|, \tag{39}$$

where $p_{\mathrm{clf}}(c \mid x)$ is the evaluation classifier probability, $\widehat{m}_c = \sum_i w_i p_{\mathrm{clf}}(c \mid x_i)$, and $\pi_c^{\mathrm{ref}} = \mathbb{E}_{x \sim P_{\mathrm{ref}}}[p_{\mathrm{clf}}(c \mid x)]$. The independent iid reference pools contain 50,000 samples for MNIST and Fashion-MNIST and 20,000 for EuroSAT and CIFAR-10. Thus this metric compares with the trained model's class allocation, which need not equal the dataset-label prior. For MNIST and Fashion-MNIST, we also use hard classifier assignments to report the number of missing classes and the

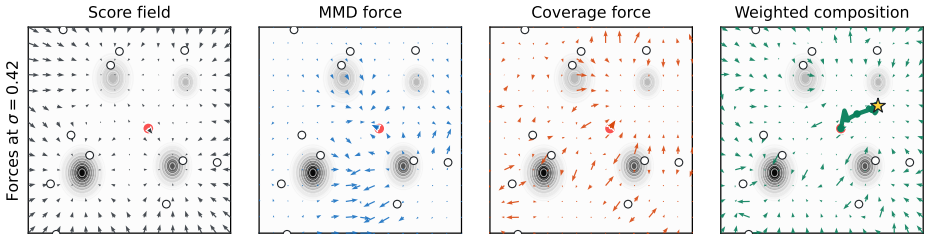


Figure 6: Guidance decomposition on the controlled 2D mixture. With the white particles fixed, the score points away from the missing upper-right mode, whereas the combined set-level field redirects the selected trajectory into it.

fraction of runs missing at least one class. These two coverage metrics depend only on the support locations.

To measure representativeness beyond class allocation, let y_i denote a generated state in a fixed evaluation space, let $\widetilde{y}_1, \dots, \widetilde{y}_M$ be independent reference states from the trained model, and let φ_{eval} be a fixed random-feature map. We report the empirical squared MMD

$$E_{\mathrm{ref}} = \left\| \sum_{i=1}^{N} w_i \varphi_{\mathrm{eval}}(y_i) - \frac{1}{M} \sum_{j=1}^{M} \varphi_{\mathrm{eval}}(\widetilde{y}_j) \right\|_2^2 . \tag{40}$$

Depending on the experiment, the evaluation space is the standardized diffusion latent space or the normalized penultimate embedding of an image classifier.

The held-out accuracies of the MNIST, Fashion-MNIST, EuroSAT, and CIFAR-10 classifiers are 0.9926, 0.9237, 0.9202, and 0.8865, respectively. Dataset-specific evaluation spaces, kernels, and run counts are given with the corresponding results.

G.1 Image sampling protocols

The latent experiments use a 64-step deterministic probability-flow Euler sampler and rank-16 spectral maps trained for $5{,}000$ updates on $20{,}000$ unlabeled draws from the corresponding frozen EDM. MNIST and Fashion-MNIST use 32-dimensional MLP-VAE latents; EuroSAT uses a 128-dimensional convolutional VAE and CIFAR-10 a 128-dimensional residual VAE. PG and EDDY-RBF are tuned separately on development seeds, and RGB sampler settings are selected on disjoint development runs. Comparisons use held-out seeds.

Table 3 gives the dataset-specific settings. All endpoint quadrature configurations use shrinkage 3, cap $1.5/N$, and perturbation levels $\mathcal{S} = \{0.1, 0.3, 1\}$. EuroSAT and CIFAR-10 retain all 16 spectral features. In the coverage experiment of Figure 2, the one-shot grayscale sampler uses $r_{\mathrm{eff}} = 10$ at $N = r = 16$ and rank 16 otherwise. For the grayscale distributional comparisons in Figure 3, profile guidance and endpoint quadrature use $r_{\mathrm{eff}}(N) = \min(16, N - 6)$.

In Figure 3, classifier-embedding MMD and class TV use 200 held-out runs per method and budget. Latent-reference MMD uses 100 runs for grayscale baselines and 20 for grayscale *Ours*, 20 for all EuroSAT methods, and 10 for all CIFAR-10 methods. Figure 2 uses 200 runs per method and budget, except for one-shot MNIST *Ours*, which uses 1,000 runs. Each confidence band uses the run count of its own series.

The external latent MMD uses three Gaussian-RBF bandwidths at half, one, and twice the reference median distance. MNIST and Fashion-MNIST use $50{,}000$ reference draws and 2,048 random Fourier features per bandwidth; EuroSAT and CIFAR-10 use $10{,}000$ draws and 512 features. The classifier-embedding evaluation uses the same three-bandwidth construction against independent 50,000-sample grayscale or 20,000-sample RGB model references. These evaluation kernels are fixed independently of the learned spectral kernel.

Table 3: Latent-image settings for *Ours*. The two-step rows give the sequential grayscale configurations. Shared quadrature settings are stated in the text.

Configuration	η	β	γ_{norm}	τ	Guidance window	Stein ρ
MNIST, one-shot	8	4	0.6	0.25	$[0.03, 8]$	0.1
Fashion-MNIST, one-shot	12	4	0.6	0.25	$[0.03, 8]$	0.1
MNIST, two-step	8	4	0.3	0.25	$[0.03, 8]$	0.1
Fashion-MNIST, two-step	10	3	0.15	0.25	$[0.03, 8]$	0.1
EuroSAT	13.50	0.196	0.170	0.071	$[0.298, 19.37]$	0.03
CIFAR-10	2.652	2.855	0.683	0.300	$[0.245, 8.760]$	0.3

RGB configuration	Profile entropy	Force clip
EuroSAT	0.00516	21.62
CIFAR-10	1.49×10^{-4}	2.590

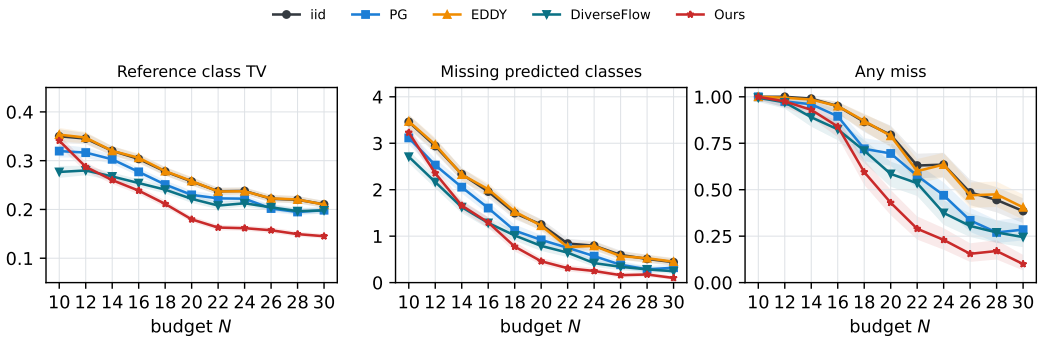


Figure 7: Direct MNIST pixel-EDM validation over 200 paired held-out runs. Panels show class TV, missing-class count, and any-miss rate; bands are 95% intervals. Lower is better.

The pixel-space validation uses a separately trained MNIST pixel EDM, a rank-16 convolutional spectral map, and a 32-step probability-flow sampler. Ours uses the same $(\eta, \beta, \gamma_{\text{norm}})$ and endpoint quadrature settings as one-shot latent MNIST, with $r_{\text{eff}}(N) = \min(16, N - 6)$. PG and EDDY-RBF are retuned for this model, and Figure 7 reports 200 paired held-out runs.

G.2 Additional image results

Figure 7 shows weaker coverage than the latent-space MNIST results in Figure 2. At $N = 20$, Ours has a mean missing-class count of 0.46 and an any-miss rate of 0.430 in pixel space, compared with 0.13 and 0.127 in latent space. This is consistent with VAE latents providing more favorable structure for spectral guidance, although the base models and sampling schedules also differ. Within pixel space, Ours nevertheless outperforms all baselines on class TV, missing-class count, and any-miss rate at this budget.

H Constructed visual MDP and weighted-chance MCTS

H.1 MDP and observation construction

For reward profile ℓ, we define the finite-horizon MDP $\mathcal{M}_\ell = (\mathcal{S}, \mathcal{A}, P, R_\ell, H, \gamma)$ with $H = 6$ and $\gamma = 0.99$ per encounter. A nonterminal state is either a route state $(k, h, \texttt{route})$ or a response state $(k, h, r, m, \texttt{response})$, where $k \in \{0, 1, 2\}$ is the encounter index, h is the current health, $r \in \{0, 1, 2\}$ is the selected route, and $m = (m_H, m_S, m_Z) \in \{0, 1\}^3$ records hostile, supply, and hazard presence. The initial state is $(0, 100, \texttt{route})$; terminal states record completion or death.

At a route state, the legal actions are safe, opportunity, and risky. Choosing r gives zero immediate reward and transitions to $(k, h, r, m, \texttt{response})$ with probability

$$p(m \mid r) = 0.75 \prod_{j=1}^{3} q_{rj}^{m_j} (1 - q_{rj})^{1-m_j} + \frac{0.25}{8}, \tag{41}$$

with route event rates $q_0 = (0.15, 0.35, 0.08)$, $q_1 = (0.45, 0.55, 0.30)$, and $q_2 = (0.65, 0.65, 0.65)$. The uniform component gives every semantic combination nonzero probability, so a route transition is eight-modal; three encounters induce $8^3 = 512$ possible mode sequences.

At a response state, the legal actions are *engage* and *retreat*, with base rewards

$$R_\ell(s, a) = \begin{cases} b_r + c_H m_H + c_S m_S - c_Z m_Z, & a = \texttt{engage}, \\ -0.4, & a = \texttt{retreat}. \end{cases} \tag{42}$$

Engage updates health to $h' = \min(100, h - 30m_H - 45m_Z + 25m_S)$, whereas retreat leaves $h' = h$. If $h' \leq 0$, the transition is terminal and adds death penalty c_D; if $k = 2$ and $h' > 0$, it is terminal and adds completion reward $c_C h'/100$. Otherwise the transition is deterministic to $(k+1, h', \texttt{route})$. Thus the process alternates three stochastic route transitions and three deterministic response transitions, and can terminate before the six-decision horizon through death.

Each route–mode state has 16 128×128 ViZDoom renderings with randomized camera angles and actor or item positions. A HUD encodes health and encounter index. We use 12 variants for training and reserve four for evaluation.

We retain one transition kernel and vary only the reward coefficients in Table 4. The task-focused profiles were selected using disjoint development seeds and frozen before the confirmation runs.

Table 4: Reward coefficients for the constructed visual MDP. The transition probabilities, health dynamics, retreat reward, renderer, and learned transition model are unchanged across profiles.

Profile	(b_0, b_1, b_2)	c_H	c_S	c_Z	c_D	c_C
Balanced	$(1.00, 1.10, 1.40)$	2.50	1.00	4.00	-10.00	3.00
Supply rush	$(1.65, 1.62, 2.96)$	1.01	3.11	3.96	-8.02	3.36
Survivor escort	$(1.78, 0.85, 1.35)$	5.88	0.53	4.35	-16.02	2.76
Combat resupply	$(1.50, 2.25, 1.75)$	6.00	4.00	7.50	-10.00	5.00

H.2 Learned visual world

Using these visual variants, we enumerate all reachable stages and health states, yielding 22,464 training and 7,488 held-out transitions. Enumerated route modes receive importance weight $8p(m \mid r)$, so model fitting targets the natural MDP conditional rather than the uniform enumeration distribution. A visual encoder maps RGB frames to an 11-dimensional continuous state containing phase, stage, health, route, and the three event indicators. An action-conditional latent EDM models the multimodal route-transition deltas; separate networks model deterministic responses and reward/termination. A common depth-conditioned leaf network is fitted to exact finite-MDP dynamic-programming targets and frozen for all samplers. The exact transition probabilities are not queried during test-time search. On held-out frames, joint state decoding accuracy is 98.7%, mode accuracy is 99.7%, reward MAE is 0.0187, and terminal-outcome accuracy is 99.5%.

In learned MCTS systems, value-training targets can incorporate search estimates (Schrittwieser et al., 2020). Sampling quality can therefore affect both current decisions and subsequent value learning, potentially amplifying its impact. We use the same frozen, DP-trained value network for all samplers to isolate their effect on planning from differences in value-network training. Training value networks with improved sampling could yield additional gains, but this feedback is not evaluated here.

H.3 Weighted-chance MCTS implementation

The planner combines standard components of stochastic tree search: UCT action selection (Kocsis & Szepesvári, 2006), sampled chance-node expectations from a generative model (Kearns et al., 1999), progressive widening for large or continuous successor spaces (Couëtoux et al., 2011; Sunberg &

Kochenderfer, 2018), and learned reward and leaf values in model-based tree search (Schrittwieser et al., 2020; Antonoglou et al., 2022). A simulation follows one root-to-tip path through alternating decision and chance nodes, expands at most one new stochastic successor batch, and then backs up the resulting value.

Every legal action is expanded once. Subsequent decision-node selection uses

$$a = \arg\max_{a'} \left[\widehat{Q}(s, a') + \sqrt{2} \sqrt{\frac{\log(N(s)+1)}{\max(N(s,a'),1)}} \right]. \tag{43}$$

For a stochastic route edge, one atomic diffusion call returns $N_p = 16$ successor particles. Outcome progressive widening permits

$$K(s,a) = \max\left(1, \left\lceil \frac{kN(s,a)^\alpha}{N_p} \right\rceil \right), \qquad k = 1, \quad \alpha = \tfrac{1}{2}, \tag{44}$$

independent particle batches on that edge; rounding to a whole batch respects the atomic sampler interface. Let w_{bi} be the normalized weight of particle i in query batch b. The route chance backup is the empirical Bellman expectation

$$\widehat{Q}_b(s,a) = \sum_{i=1}^{N_p} w_{bi} \left[\widehat{r}_{bi} + \widehat{V}(s'_{bi}) \right], \qquad \widehat{Q}(s,a) = \frac{1}{K(s,a)} \sum_{b=1}^{K(s,a)} \widehat{Q}_b(s,a). \tag{45}$$

Here $\widehat{V}$ is a child-tree value when expanded and the common learned leaf value otherwise. Uniform methods set $w_{bi} = 1/N_p$; our method uses its returned quadrature weights. Equal weighting across independent batches means that an extra model call contributes one further Monte Carlo/quadrature estimate rather than receiving extra probability merely because it was added later. Equation (45) is therefore the usual chance-node expectation over a finite sampled measure; the experimental intervention is only which finite measure the diffusion sampler returns.

H.4 Closed-loop protocol and metrics

Following the closed-loop protocol in Section 4.3, deterministic response actions use Bellman backups $\widehat{r} + \gamma \widehat{V}(s')$, with zero continuation at termination, and consume no diffusion calls. Each stochastic route replan permits $B \in \{48, 96, 192\}$ atomic diffusion calls and at most 4,096 tree simulations; B counts calls rather than the $16B$ returned particles. Because the remaining horizon shrinks after each encounter, the corresponding mean whole-episode totals are 103, 199, and 391 calls rather than $3B$. The leaf network values branches not expanded before the remaining horizon.

We evaluate 200 paired closed-loop runs. In addition to route regret and Q-MAE, we report route optimality, the fraction of route decisions selecting a true maximizing action, and route Q-gap MAE, the error in pairwise action-value differences. We also report true encounter-discounted return ($\gamma = 0.99$) and separate its single-outcome noise using

$$V_{\mathrm{DP}} = V^*(s_0) - \sum_j \gamma^{e(j)} \left[V^*(s_j) - Q^*(s_j, a_j) \right], \tag{46}$$

where j enumerates route and response decisions and $e(j)$ is their encounter index. By the performance-difference identity, $\mathbb{E}[V_{\mathrm{DP}}]$ is the true expected return of the executed replanning policy. These exact quantities are evaluation targets only; search uses the frozen leaf approximation rather than querying the dynamic program.

H.5 Complete planning results

Table 5 reports empirical averages over 200 paired runs. The realized returns in panel (a) have high trial-to-trial variance because each run follows a single stochastic environment trajectory; sample means and method rankings can therefore vary across repeated sets of trials. The DP decomposition in panel (b) removes the direct outcome noise, but its estimates still vary with the visited states and planning randomness. Bold entries identify the largest observed means and do not by themselves indicate statistically significant differences.

Across all 48 ours–baseline comparisons, paired 95% intervals for the additional Q-gap MAE metric favor ours. The stochastic route frames are semantically exact in all 36,000 audited decisions.

Table 5: Closed-loop results over 200 paired runs. Each entry is the mean for stochastic-replan call-budget caps $B = 48/96/192$. Panel (a) uses one true environment outcome per run; panel (b) is the exact DP-decomposed policy value in Eq. (46). Higher is better; bold marks the largest sample mean in each setting. Underlining marks a sample mean below iid at the same budget.

Method	Balanced	Supply rush	Survivor escort	Combat resupply
(a) True realized encounter-discounted return				
iid	7.304/7.159/7.178	10.538/**10.721**/10.290	10.087/10.360/**10.260**	17.501/17.603/17.493
Particle Guidance	7.153/7.284/7.258	10.651/10.298/10.412	9.909/9.901/9.968	17.187/17.019/17.376
EDDY-RBF	6.995/7.045/6.963	10.571/10.250/10.362	10.114/9.577/9.727	16.494/16.598/16.762
DiverseFlow	7.255/7.294/7.238	10.685/10.594/10.399	10.025/10.178/10.176	17.192/17.334/17.291
Ours	**7.496**/**7.552**/**7.440**	**10.702**/10.669/**10.912**	**10.173**/**10.370**/10.192	**18.202**/**18.258**/**18.306**
(b) DP-decomposed expected policy value V_{DP}				
iid	7.169/7.168/7.200	10.347/10.354/10.373	9.983/9.982/10.008	16.999/17.151/17.223
Particle Guidance	7.144/7.129/7.168	10.420/10.411/10.433	10.047/10.047/10.091	16.872/16.965/17.064
EDDY-RBF	6.941/6.982/7.025	10.389/10.376/10.386	10.046/10.066/10.098	16.362/16.394/16.427
DiverseFlow	7.168/7.185/7.202	10.356/10.369/10.377	9.994/9.996/10.019	16.915/17.100/17.121
Ours	**7.398**/**7.383**/**7.411**	**10.483**/**10.485**/**10.516**	**10.211**/**10.233**/**10.265**	**17.831**/**17.833**/**17.861**

Table 6: Route optimality for budgets $B = 48/96/192$. Each entry averages 600 route decisions. Higher is better; bold marks the best value, and underlining marks a paired 95% interval strictly below iid.

Method	Balanced	Supply rush	Survivor escort	Combat resupply
iid	.555/.527/.553	.472/.445/.482	.432/.445/.475	.750/.768/.803
Particle Guidance	.512/.495/.523	.592/.568/.610	.555/.548/.602	.713/.727/.760
EDDY-RBF	.348/.350/.367	.563/.540/.572	.583/.603/.630	.545/.570/.573
DiverseFlow	.548/.553/.562	.485/.473/.493	.458/.468/.490	.738/.772/.783
Ours	**.727**/**.703**/**.747**	**.622**/**.620**/**.665**	**.778**/**.813**/**.865**	**.983**/**.983**/**.992**