Variance Reduction for Gradients from Diffusion Teachers

Jesse Bettencourt[1 2 4] **Xindi Wu**[1 3] **Matan Atzmon**[1] **James Lucas**[1] **Jonathan Lorraine**[1]

[1]NVIDIA [2]University of Toronto [3]Princeton University [4]Vector Institute

Project page: research.nvidia.com/labs/sil/projects/CARV/

Abstract

Pretrained diffusion models serve as frozen teachers for pipelines such as text-to-3D, single-step distillation, and data attribution. The teacher gradients these pipelines consume are expectations over noise levels and Gaussian noise, estimated by Monte Carlo (MC). Each MC draw requires expensive upstream work (rendering, simulation, encoding), so estimator variance dominates compute cost. We introduce CARV, a compute-aware variance-accounting framework that casts these gradients as a classical two-stage sampling problem with asymmetric per-level costs. This motivates a hierarchical MC estimator that amortizes each expensive upstream computation by reusing it across cheap diffusion draws, with within-unit variance further reduced by timestep importance sampling (IS) and stratified inverse-CDF sampling. CARV delivers 2 to $3\times$ effective compute multipliers on text-to-3D and data attribution without changing the objective. In text-to-3D, reuse provides most of the gain and IS with stratification adds about 25%. In single-step distillation, the same techniques cut gradient variance by an order of magnitude but do not improve downstream FID, marking the regime where MC variance is no longer the bottleneck.

1 Introduction

Diffusion models generate state-of-the-art images, video, audio, and 3D/4D by repeatedly evaluating a denoiser on a noisy input and stepping along the predicted denoising direction. That direction can instead be used as a gradient by a downstream pipeline, with the model frozen as a teacher: in text-guided 3D the gradient from an image model steers a differentiable renderer, in one-step distillation it trains a student generator, and in data attribution it scores training examples. The signal in each case is the teacher gradient, this direction averaged over noise levels and Gaussian noise, and it must be estimated by Monte Carlo. Each draw requires expensive upstream work (rendering, simulation, encoding), so estimator variance dominates the compute cost.

Most variance-reduction work targets teacher training via loss reweighting and schedule design. Downstream, practitioners inherit the teacher's timestep distribution, average draws ad hoc, sometimes with bias, and tune compute without knowing which source of randomness dominates variance or what each source costs. This leaves three open questions: which estimator components dominate variance, how to reduce variance without biasing the objective, and how to trade cheap operations (noising, denoising) against expensive ones (rendering, encoding) under a fixed budget.

We propose CARV, which treats frozen-teacher gradient estimation as an instance of classical two-level sampling with asymmetric per-level costs: expensive upstream computations are the primary units and cheap diffusion draws are the subunits. Classical theory then supplies the variance decomposition, the cost-optimal allocation between levels, importance sampling, and stratification. CARV makes them operational with an explicit cost model and parameter-gradient variance measurement. The resulting drop-ins are unbiased under the stated sampling construction and match baseline variance at a third to half the cost (2 to $3\times$ effective compute multipliers) in text-to-3D optimization and attribution, with no downstream FID gain in one-step distillation. Our contributions include:

1. **Frozen-teacher gradients as two-stage sampling.** We identify the frozen-teacher gradient estimator as the classical two-stage sample mean with asymmetric per-level costs; classical theory then gives the variance decomposition, the allocation $K^\star$, and the within-level variance reductions (importance sampling and stratification), which we operationalize with an explicit cost model and parameter-gradient variance measurement (Sec. 4).

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

2. **Timestep importance sampling using the explicit teacher weight.** A drop-in proposal $q \propto p\, w_{\text{SDS}}$ giving $\sim 1.2\times$ variance reduction over uniform at equal per-iteration cost (Table 2; toy in Fig. 1).
3. **Stratified-inverse-CDF sampling over noise levels.** Adapts the standard equal-mass quantile construction to frozen-teacher timestep allocation, composing stratification with importance sampling per render (App. Fig. 8, Fig. 3); close to a Sinkhorn-optimized two-sample allocation (App. Sec. D.1.6).
4. **Compute-aware variance-accounting framework (CARV) with broad evaluation.** A measurement framework for diffusion gradient variance (Sec. 4.2, Fig. 4) applied to diffusion-teacher-guided optimization (Sec. 5.1), one-step distillation (Sec. 5.2), and data attribution (Sec. 5.3).

2 Related Work

Classical Monte Carlo variance reduction. Importance sampling, stratification, control variates, and antithetic variates are standard [76, 65], including for gradient estimation in machine learning [59]. Two-stage sampling with unequal per-stage costs, including the cost-optimal number of subunits per primary unit, is classical sampling theory [21, 10, 91, 44], as is stratified inverse-transform sampling [16]. Graphics amortizes expensive scene queries over reused samples in the same spirit [3, 39].

Teacher-side schedule and weighting design. Most diffusion variance reduction targets training the teacher: noise schedules [23, 63, 24], learned signal-to-noise schedules that minimize the variance of the training objective [35], the noise schedule read as an importance-sampling density over noise levels [34], and loss reweighting [78, 29, 9, 20]. These optimize gradients with respect to denoiser parameters, under a cost structure with no expensive upstream stage.

Downstream frozen-teacher methods. SDS-style optimization [69, 38, 95], few-step distillation [105, 104, 79], and gradient-based attribution [66, 102, 98] consume teacher gradients as Monte Carlo estimates but treat the estimator monolithically: they inherit the teacher's timestep distribution or reshape it without a likelihood-ratio correction [26]; SDS-style and distillation methods draw one diffusion sample per expensive upstream computation (RewardSDS [6] reweights several noise draws per render by a reward, for alignment rather than variance). VSD [95] and SteinDreamer [94] lower SDS variance by changing the objective or adding a control variate; our estimator changes compose with both (App. Sec. E.4, App. Sec. E.5).

Our central claim is the identification that frozen-teacher gradient estimation is an instance of the classical two-level sampling problem with asymmetric per-level costs, with renders as primary sampling units and diffusion draws as subunits. Extended related work is in App. Sec. E.

3 Background

Further background is in App. Sec. B.

3.1 Diffusion Models

We use conditional latent diffusion via an autoencoder [23, 75]. Let $(\mathbf{x}, \mathbf{c})$ be a data point and its conditioning (e.g., image+text), and $\mathbf{z} = \text{Encode}(\mathbf{x})$ its latent code. Forward noising at level $t \in [0, 1]$, from clean data at $t = 0$ to pure noise at $t = 1$, makes $\mathbf{z}_t = \alpha_t \mathbf{z} + \sigma_t \boldsymbol{\epsilon}$ with $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, where α_t, σ_t are schedule-dependent coefficients (clamped away from $t = 0$). We use continuous notation; many systems use a discrete grid of T timesteps (mapping in App. Sec. C.2), and some flow-matching codebases run t in the opposite direction.

The teacher $\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\phi}}$, with parameters $\boldsymbol{\phi}$, is trained on $\mathbb{E}_{(\mathbf{x},\mathbf{c}),t,\boldsymbol{\epsilon}}[w(t)\ell_{\text{Diff}}]$, where $w(t)$ is the timestep weight, $\ell_{\text{Diff}} := \|\mathbf{r}\|_2^2$, and $\mathbf{r} := \hat{\boldsymbol{\epsilon}}_{\boldsymbol{\phi}}(\mathbf{z}_t, t, \mathbf{c}) - \boldsymbol{\epsilon}$ (App. Sec. B.1). App. Sec. B.1.1 covers the essentials of sampling from diffusion models and classifier-free guidance. Sec. 3.2.1 covers diffusion noise schedules and their connection to importance sampling.

3.2 Reducing Estimator Variance

We consider Monte Carlo estimators for expectations that arise in diffusion training and downstream uses of frozen diffusion teachers. These are typically vector-valued (e.g., a parameter gradient or an update direction), so for an unbiased estimator $\hat{\boldsymbol{\mu}}$ of a vector mean $\boldsymbol{\mu}$, mean-squared error and

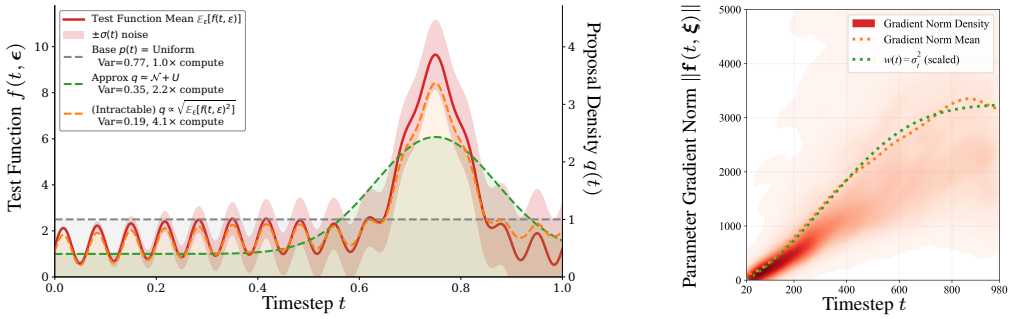


Figure 1: Importance sampling for timestep allocation: *Left:* Toy example showing a test function F, uniform proposal, oracle optimal proposal from Eq. 23, and a practical approximation adding a Gaussian at the peak. The oracle variance equals that of spending $\sim$3–4$\times$ the compute of uniform, while the approximation equals $\sim 2\times$. *Right:* Real parameter gradient norms from text-to-3D optimization. Our importance sampling weight function $w_{\text{SDS}}(t)$ closely tracks the mean gradient norm, making it an effective sampling proxy: it reduces variance as much as $\sim$20% more compute would (Table 2). Shading is the density of sampled norms over prompts, cameras, and noise. Additional analysis is in App. Fig. 23.

variance coincide, and we measure dispersion by:

$$\text{Var}(\hat{\boldsymbol{\mu}}) := \mathbb{E}[\|\hat{\boldsymbol{\mu}} - \boldsymbol{\mu}\|_2^2] = \text{tr}(\text{Cov}(\hat{\boldsymbol{\mu}})). \tag{1}$$

3.2.1 Importance Sampling & Noise Schedules

Importance sampling reduces Monte Carlo variance by sampling from important regions while preserving the expectation via a likelihood ratio. For noise level $t \in [0,1]$ with base density $p(t)$ and randomness $\boldsymbol{\xi}$, we estimate the mean:

$$\boldsymbol{\mu} = \mathbb{E}_{t\sim p, \boldsymbol{\xi}\sim p(\cdot|t)}[\mathbf{f}(t,\boldsymbol{\xi})] \tag{2}$$

where $\mathbf{f}(t,\boldsymbol{\xi})$ is a vector-valued contribution such as a gradient. Sampling $t^{(n)} \sim q$ with importance weight $\tilde{w}(t) = p(t)/q(t)$ gives the unbiased estimator:

$$\hat{\boldsymbol{\mu}}_q = {}^1\!/_N \sum_{n=1}^{N} \tilde{w}(t^{(n)}) \mathbf{f}(t^{(n)}, \boldsymbol{\xi}^{(n)}). \tag{3}$$

The variance-minimizing proposal is $q^\star(t) \propto p(t)\sqrt{\mathbb{E}[\|\mathbf{f}(t,\boldsymbol{\xi})\|_2^2 \mid t]}$ [28, 76]. This oracle is impractical, so practitioners use cheap surrogates such as the per-timestep loss [63], fixed non-uniform samplers [109], or learned schedules [35]. The right surrogate depends on the form of $\mathbf{f}$, which is task-specific; we instantiate $q^\star$ and pick a proxy for the SDS gradient in Sec. 4.1.2. As shown in Fig. 1 (left), the oracle proposal is variance-equivalent to $\sim$3–4$\times$ uniform compute and a Gaussian-at-peak approximation to $\sim 2\times$; full derivation in App. Sec. B.2.

Noise Schedules as Importance Sampling: A noise schedule defines how signal and noise coefficients α_t, σ_t vary with $t \in [0,1]$. With uniform sampling $t \sim \mathcal{U}(0,1)$, a schedule induces a distribution over noise levels. Schedule design can thus be viewed as choosing an importance distribution for the diffusion training objective; Kingma et al. [35] parametrize this in terms of signal-to-noise ratio and learn a schedule that minimizes the variance of the training-objective estimator. In downstream tasks, the schedule is inherited from the pretrained teacher, so we augment the timestep distribution via explicit importance sampling (Sec. 4.1.2) rather than retraining. See App. Sec. B.2.3 for details.

3.2.2 Stratified Sampling

Stratified sampling (visualized in App. Fig. 8) reduces variance by partitioning the domain into B disjoint strata and averaging estimates within strata before combining with probabilities. Partition the domain into strata $\{\mathcal{S}_b\}_{b=1}^B$ with probabilities p_b and M_b samples per stratum. The estimator is:

$$\hat{\boldsymbol{\mu}}_{\text{strat}} = \sum_{b=1}^{B} p_b \frac{1}{M_b} \sum_{m=1}^{M_b} \mathbf{F}(t_{b,m}),\ t_{b,m} \sim p(t \,|\, \mathcal{S}_b) \tag{4}$$

This estimator is unbiased for $\mathbb{E}_{t\sim p(t)}[\mathbf{F}(t)]$ and, with proportional allocation (e.g., equal-probability strata with equal M_b), has variance no greater than standard IID sampling from $p(t)$, with strict reduction whenever the stratum means of $\mathbf{F}$ differ [91]. As a simple example with a continuous domain $t \in [0,1]$, one can use B equal-width bins. With batch size B and $M_b = 1$, each bin contributes one draw; Sec. 4.1.3 applies this to diffusion timesteps.

3.3 Diffusion Model Applications

3.3.1 Diffusion Priors for Optimization

Score Distillation Sampling (SDS) uses a frozen pretrained diffusion model to supply gradients to a differentiable generator, renderer, or simulator, and underlies text-guided 3D/4D, material, and audio optimization [69, 2, 42, 13, 74]. Given generator parameters $\boldsymbol{\theta}$ and rendering condition $\boldsymbol{\pi}$ (e.g., camera pose), we form an encoded observation:

$$\mathbf{z} = g(\boldsymbol{\theta}, \boldsymbol{\pi}) = \mathrm{Encode}(g'(\boldsymbol{\theta}, \boldsymbol{\pi})) \tag{5}$$

where g' is the differentiable renderer (or generator or simulator) and Encode maps its output into the teacher's latent space, and we use the teacher to update $\boldsymbol{\theta}$. With $\mathbf{z}_t = \alpha_t \mathbf{z} + \sigma_t \boldsymbol{\epsilon}$ and residual $\mathbf{r} = \hat{\boldsymbol{\epsilon}}_{\boldsymbol{\phi}}(\mathbf{z}_t, t, \mathbf{c}; \omega) - \boldsymbol{\epsilon}$, SDS uses proxy score-direction $w(t)\mathbf{r}\,\mathrm{d}\mathbf{z}_t/\mathrm{d}\mathbf{z}$ and applies the chain rule:

$$\mathbf{u}_{\mathrm{SDS}}(\boldsymbol{\theta}) = \mathop{\mathbb{E}}_{\boldsymbol{\pi},t,\boldsymbol{\epsilon}}\left[w_{\mathrm{SDS}}(t)\,\mathbf{r}\,{}^{\mathrm{d}\mathbf{z}}/_{\mathrm{d}\boldsymbol{\theta}}\right], \text{ where } w_{\mathrm{SDS}}(t) = w(t)\alpha_t \text{ (using } {}^{\mathrm{d}\mathbf{z}_t}/_{\mathrm{d}\mathbf{z}} = \alpha_t \mathbf{I}) \tag{6}$$

so $w_{\mathrm{SDS}}(t)$ absorbs schedule factors, simplifying backprop and enabling importance sampling. The teacher is frozen; the stop-gradient surrogate is in App. Sec. B.3.1.

3.3.2 Single-Step Diffusion Distillation

Distribution Matching Distillation (DMD) [105] trains a one-step generator by averaging $w(t)\alpha_t(\mathbf{s}_{\mathrm{fake}} - \mathbf{s}_{\mathrm{real}})\,\partial\mathbf{z}/\partial\boldsymbol{\theta}$ over standard Gaussian generator inputs, $t \sim p$, and independent standard Gaussian forward noise. Here $\mathbf{z}$ is the generator output and $\mathbf{s}_{\mathrm{real}}, \mathbf{s}_{\mathrm{fake}}$ are scores of the noised data and generator distributions at $\mathbf{z}_t$, from the frozen teacher and a fake-score model trained on generator outputs (Eq. 51; App. Sec. B.3.2).

3.3.3 Data Attribution for Video Generation

Gradient-based attribution ranks training examples by averaging inner products of unit ℓ_2-normalized per-draw loss gradients $\nabla_{\boldsymbol{\phi}}\ell_{\mathrm{Diff}}$ from a query and training example, using shared $(t, \boldsymbol{\epsilon})$ draws (Eq. 58). For video, we use MOTIVE's motion-weighted loss and projected gradients [98], reducing variance in the underlying gradient estimator (Sec. 5.3; App. Sec. B.3.3).

4 Our Method

We present CARV: frozen-teacher gradient estimation as two-level sampling with asymmetric per-level costs (Sec. 4.1), the allocation between levels and the two within-level variance reductions this view prescribes (Secs. 4.1.1–4.1.3), and a compute-aware procedure for measuring variance and setting the allocation (Sec. 4.2). Implementation details are in App. Sec. C.

4.1 Frozen-Teacher Gradients as Two-Level Sampling

The estimand. Every gradient we estimate has the form $\mathbb{E}_{\mathbf{x}}\left[\mathbb{E}_{(t,\boldsymbol{\epsilon})|\mathbf{x}}[\mathbf{h}(\mathbf{x}, t, \boldsymbol{\epsilon})]\right]$, an average over an *expensive upstream draw* $\mathbf{x}$ and over *cheap diffusion randomness* $(t, \boldsymbol{\epsilon})$ drawn given $\mathbf{x}$. The upstream draw is a render at a sampled camera pose (SDS), a generator output for a sampled input noise (DMD), or a fixed training example (data attribution, where only the inner expectation is estimated). In SDS and DMD, $\mathbf{h} = \mathbf{f}(\mathbf{x}, t, \boldsymbol{\epsilon})\,{}^{\partial\mathbf{z}}/_{\partial\boldsymbol{\theta}}$, where $\mathbf{f}$ is the vector multiplying the renderer or generator Jacobian: $\mathbf{f} = w_{\mathrm{SDS}}(t)\mathbf{r}$ for SDS (Sec. 3.3.1), and a weighted score difference for DMD (Sec. 3.3.2). The Jacobian is shared by every $(t, \boldsymbol{\epsilon})$ drawn at the same $\mathbf{x}$, so the backward pass through the renderer or generator runs once per upstream draw, on the averaged inner vector; this is why inner draws are cheap.

The identification and the estimator. This is the classical two-level, or two-stage, sampling problem [21, 10, 44]: upstream draws are the primary sampling units and diffusion draws are the subunits, with very different per-unit costs. The standard estimator draws one subunit per primary unit, using a fresh render $\mathbf{x}^{(r)} = g'(\boldsymbol{\theta}, \boldsymbol{\pi}^{(r)})$ per sample:

$$\hat{\nabla}^{\mathrm{naive}}_{\boldsymbol{\theta}} = {}^1/_R \sum_{r=1}^{R} \mathbf{f}(\mathbf{x}^{(r)}, t^{(r)}, \boldsymbol{\epsilon}^{(r)})\,{}^{\partial\mathbf{z}^{(r)}}/_{\partial\boldsymbol{\theta}} \tag{7}$$

where $(\boldsymbol{\pi}^{(r)}, t^{(r)}, \boldsymbol{\epsilon}^{(r)})$ are IID, with cost $R(c_{\mathrm{render+encode}} + c_{\mathrm{denoise}})$. The two-stage estimator draws R primary units and K subunits $(t, \boldsymbol{\epsilon})$ for each, and averages within units and then across them:

$$\hat{\nabla}^{\mathrm{reuse}}_{\boldsymbol{\theta}} = {}^1/_R \sum_{r=1}^{R} \left({}^1/_K \sum_{k=1}^{K} \mathbf{f}(\mathbf{x}^{(r)}, t^{(r,k)}, \boldsymbol{\epsilon}^{(r,k)})\right) {}^{\partial\mathbf{z}^{(r)}}/_{\partial\boldsymbol{\theta}} \tag{8}$$

It is unbiased for every $K \geq 1$ because $(t, \boldsymbol{\epsilon})$ are drawn independently given $\mathbf{x}^{(r)}$, and it reduces to Eq. 7 at $K = 1$. Our hierarchical MC estimator is exactly this classical estimator; caching each upstream draw and re-noising it (compute reuse) is how it is implemented.

The cost model. Per optimization step, the two-stage estimator costs

$$\text{cost} \approx R\,(c_{\text{render+encode}} + K\,c_{\text{denoise}}). \tag{9}$$

The upstream cost $c_{\text{render+encode}}$ is paid once per primary unit: a forward and backward pass through the renderer or generator and the encoder in SDS and DMD, and encoding the training example in data attribution. The denoising cost c_{denoise} is paid once per subunit. In SDS and DMD it covers forward passes of the frozen teacher (and, in DMD, of the fake-score model) with no backward pass through them, since the gradients omit their Jacobians (Sec. 3.3); in data attribution it is the forward and backward pass that produces a per-example gradient. Both costs are wall-clock times measured on the hardware in use, including parallelization (App. Sec. C.1), so $c_{\text{render+encode}}/c_{\text{denoise}}$ must be measured (Sec. 4.2).

Variance decomposition. Define the across-unit variance of the conditional mean and the mean within-unit variance, $\sigma_A^2 = \text{Var}_{\mathbf{x}}\big(\mathbb{E}[\mathbf{h} \mid \mathbf{x}]\big)$ and $\sigma_W^2 = \mathbb{E}_{\mathbf{x}}\big[\text{Var}(\mathbf{h} \mid \mathbf{x})\big]$, with trace covariances as in Eq. 1. Writing $\hat{\mathbf{h}}^{(r)}$ for the average of $\mathbf{h}$ over the K subunits of unit r, Eq. 8 is $\hat{\nabla}_{\boldsymbol{\theta}} = {}^1\!/_R \sum_r \hat{\mathbf{h}}^{(r)}$. The $\hat{\mathbf{h}}^{(r)}$ are IID across r, and given $\mathbf{x}^{(r)}$ the subunit draws are IID, so the law of total variance gives $\text{Var}(\hat{\mathbf{h}}^{(r)}) = \sigma_A^2 + \sigma_W^2/K$ and the classical decomposition [10] (App. Sec. B.2.2)

$$\text{Var}(\hat{\nabla}_{\boldsymbol{\theta}}) = \frac{\sigma_A^2}{R} + \frac{\sigma_W^2}{RK}. \tag{10}$$

The two terms respond differently: σ_A^2/R falls only with more primary units, while $\sigma_W^2/(RK)$ falls with more subunits or with a smaller σ_W^2.

Efficiency and the three strategies. We compare estimators by efficiency $\propto 1/(\text{Var}\cdot\text{cost})$ [19, 17]; Sec. 4.2 defines the two metrics we report. For this structure, classical theory offers three strategies. Allocation between levels keeps the estimator of Eq. 8 and chooses only (R, K), the split of a budget between primary units and subunits (Sec. 4.1.1). Importance sampling of the subunit draws (Sec. 4.1.2) and their stratification within each primary unit (Sec. 4.1.3) change how t is drawn to lower σ_W^2, leaving σ_A^2 untouched.

4.1.1 Allocation Between Levels and Compute Reuse

The allocation problem. At a fixed budget $\mathcal{B}$, Eq. 9 gives $R = \mathcal{B}/(c_{\text{render+encode}} + K c_{\text{denoise}})$ (treating R as continuous), and substituting into Eq. 10 leaves the variance as a function of K alone:

$$\text{Var}(\hat{\nabla}_{\boldsymbol{\theta}}) = \frac{1}{\mathcal{B}}\Big(\sigma_A^2 c_{\text{render+encode}} + \sigma_W^2 c_{\text{denoise}} + \sigma_A^2 c_{\text{denoise}}\,K + \frac{\sigma_W^2 c_{\text{render+encode}}}{K}\Big). \tag{11}$$

The first two terms do not depend on K. The third grows with K, because more subunits per unit leave fewer units; the fourth shrinks with K, because the upstream cost is amortized over more subunits.

The optimum under the cost model. Eq. 11 is convex in $K > 0$ and minimized where $\sigma_A^2 c_{\text{denoise}} = \sigma_W^2 c_{\text{render+encode}}/K^2$, at $K^\star = \sqrt{(\sigma_W^2/\sigma_A^2)(c_{\text{render+encode}}/c_{\text{denoise}})}$ [10]; by convexity the best integer is the better of $\lfloor K^\star \rfloor$ and $\lceil K^\star \rceil$ (at least 1). The optimum exceeds one exactly when $\sigma_W^2 c_{\text{render+encode}} > \sigma_A^2 c_{\text{denoise}}$, and $K = 2$ beats $K = 1$ when $\sigma_W^2 c_{\text{render+encode}} > 2\sigma_A^2 c_{\text{denoise}}$ ($K^\star > \sqrt{2}$), so re-noising helps when upstream cost dominates, unless the within-unit variance is small relative to the across-unit variance. When upstream cost vanishes, $K^\star \to 0$, so $K = 1$ is best for IID subunits and re-noising alone cannot help; only the σ_W^2-lowering strategies of Secs. 4.1.2 and 4.1.3 remain. All four quantities in $K^\star$ can be estimated from a short pilot before training (Sec. 4.2). The optimum minimizes estimator variance at fixed cost; downstream metrics such as FID can respond differently (Sec. 5.2).

Compute reuse implements $K > 1$. We cache each upstream draw and re-noise it with K independent $(t, \boldsymbol{\epsilon})$ draws (Fig. 2). Since $c_{\text{render+encode}}$ includes backpropagation through the renderer or generator while c_{denoise} needs only forward passes of the score models, $c_{\text{render+encode}} \gg c_{\text{denoise}}$ often holds in SDS and DMD. With latent diffusion, each render is encoded once and its K noised latents are formed from the cached encoding. Per-task instantiations and the combined pseudocode (Algorithm 1) are in App. Sec. C.2.

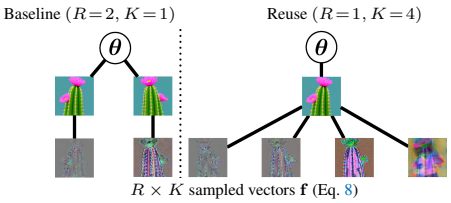


Figure 2: Compute Reuse Visualization: Baseline (left, $K=1$) and re-noising (right, $K>1$) use R sampled renders and K noisings per render. Both take $\boldsymbol{\theta}$ (e.g., NeRF weights or generator), render, encode, noise, denoise, combine into a residual, and backpropagate. Re-noising helps when (a) $(t, \boldsymbol{\epsilon})$ drives variance and (b) denoising is cheaper than rendering. Fig. 11 quantifies the trade-off. $\mathbf{f}$ is in the diffusion latent space but visualized in pixel space.

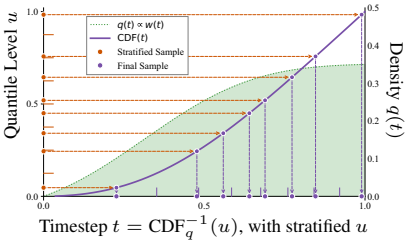


Figure 3: Combining Stratified Sampling with Importance Weighting: We illustrate how to use inverse-transform sampling to map a stratified sample uniformly in $[0, 1]$ (see App. Fig. 8) into a "stratified" sample for our target importance distribution, using the inverse-CDF, with the now non-uniform bins shown via purple ticks. This allows us to combine the benefits of both strategies, forming better estimators (see Fig. 11).

4.1.2 Importance Sampling the Timestep

The within-unit variance σ_W^2 is an expectation over $t \sim p$. Drawing $t \sim q$ with weight p/q leaves σ_A^2 and the form of Eq. 10 unchanged and replaces σ_W^2 with the within-unit variance under q, which $q^\star \propto p\sqrt{\mathbb{E}[\|\mathbf{h}\|_2^2 \mid t]}$ minimizes [28, 76].

For SDS, the per-sample contribution multiplying the renderer Jacobian is $\mathbf{f}(\mathbf{x}, t, \boldsymbol{\epsilon}) = w_{\text{SDS}}(t)\,\mathbf{r}$, giving parameter-gradient form $w_{\text{SDS}}(t)\,\mathbf{J}_{\boldsymbol{\theta}}^\top \mathbf{r}$ at fixed render. The variance-minimizing proposal from Sec. 3.2.1 is then $q^\star(t) \propto p(t)\sqrt{\mathbb{E}[\|w_{\text{SDS}}\mathbf{J}_{\boldsymbol{\theta}}^\top \mathbf{r}\|_2^2 \,|\, t]}$, which requires per-timestep gradient norms over renders and noise and is impractical. Loss-based proxies use only $\|\mathbf{r}\|_2^2$ and ignore w_{SDS}^2 and $\|\mathbf{J}_{\boldsymbol{\theta}}\|_2^2$; since $\|w_{\text{SDS}}\mathbf{J}_{\boldsymbol{\theta}}^\top \mathbf{r}\|_2^2 \le w_{\text{SDS}}^2 \|\mathbf{J}_{\boldsymbol{\theta}}\|_2^2 \|\mathbf{r}\|_2^2$, they can misrank timesteps when w_{SDS} varies with t or $\|\mathbf{J}_{\boldsymbol{\theta}}^\top \mathbf{r}\|_2$ correlates poorly with $\|\mathbf{r}\|_2$, and ill-conditioned Jacobian chains through encoders and differentiable generators can amplify this.

Because w_{SDS} is deterministic given t, $q^\star \propto p\, w_{\text{SDS}} \sqrt{\mathbb{E}[\|\mathbf{J}_{\boldsymbol{\theta}}^\top \mathbf{r}\|_2^2 \,|\, t]}$, so $q \propto p\, w_{\text{SDS}}$ is exact when this conditional second moment is flat in t; the estimator keeps the Jacobian, and only the proposal omits it. Empirically, w_{SDS} dominates the timestep dependence of the gradient norm (App. Fig. 22), so we use the negligible-cost proposal $q(t) \propto p(t) w_{\text{SDS}}(t)$ with likelihood-ratio correction.

For data attribution (Sec. 5.3), gradient norms are approximately constant across timesteps (App. Fig. 29), so we use uniform sampling. For DMD (Sec. 5.2), the weighting function is non-monotonic with data-dependent normalization, so we focus on stratification and compute reuse.

4.1.3 Stratified Sampling of the Re-noisings

Taking the K subunit draws of each primary unit one per equal-mass stratum of q removes the between-stratum part of σ_W^2 and leaves σ_A^2 unchanged [10, 65]; the inverse-CDF quantile construction [16] composes this with any proposal. Timestep discretization details are in App. Sec. C.2.

Per-render stratification. With re-noising, set the number of strata equal to the number of re-noisings, $B := K$. For a uniform p, partition $[0, 1]$ into equal-width bins $\mathcal{S}_b = [(b-1)/B, b/B]$ for $b = 1, \ldots, B$, each with probability $1/B$. For each render $\mathbf{x}^{(r)}$, draw one timestep per bin and independent noise $\boldsymbol{\epsilon}_b^{(r)}$. The per-render contribution is:

$$\bar{\mathbf{f}}_{\text{strat}}^{(r)} = 1/B \sum_{b=1}^{B} \mathbf{f}(\mathbf{x}^{(r)}, t_b^{(r)}, \boldsymbol{\epsilon}_b^{(r)}) \tag{12}$$

where $t_b^{(r)}$ is uniform on $\mathcal{S}_b$, and the gradient estimate is $\frac{1}{R}\sum_r \bar{\mathbf{f}}_{\text{strat}}^{(r)} \frac{\partial \mathbf{z}^{(r)}}{\partial \boldsymbol{\theta}}$ (see Eq. 8). This stratifies the subunits within each primary unit.

Stratified importance sampling. To combine IS and stratification for a non-uniform proposal, stratify in proposal-quantile space [16] rather than t (Fig. 3). For each render $\mathbf{x}^{(r)}$ and stratum $b = 1 \ldots B$, with proposal $q(t)$ and IID jitters $\boldsymbol{\xi}_b^{(r)} \sim \mathcal{U}(0, 1)$, define:

$$u_b^{(r)} = (b - 1 + \boldsymbol{\xi}_b^{(r)})/B, \quad t_b^{(r)} = \text{CDF}_q^{-1}(u_b^{(r)}) \tag{13}$$

so that $\{t_b^{(r)}\}_{b=1}^B$ contains one draw from each equal-mass stratum of q, or equivalently, non-uniform bins in t whose q-probabilities are all $1/B$. With independent $\boldsymbol{\epsilon}_b^{(r)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and importance

weights $\tilde{w}(t) = {}^{p(t)}/_{q(t)}$, the per-render stratified-IS contribution is:

$$\bar{\mathbf{f}}^{(r)}_{\text{strat-IS}} = {}^{1}/_{B} \sum_{b=1}^{B} \tilde{w}(t_b^{(r)}) \mathbf{f}(\mathbf{x}^{(r)}, t_b^{(r)}, \boldsymbol{\epsilon}_b^{(r)}) \tag{14}$$

This remains unbiased for sampling $t \sim p$ while reallocating draws toward timesteps with larger contributions via q and reducing variance by enforcing balanced coverage across the q-quantiles even when q is highly non-uniform.

Design choice. Both schemes add negligible compute. Per-render stratification (Eq. 12) is preferred when $K > 1$, since it stratifies the subunits and composes with the allocation of Sec. 4.1.1; when $K = 1$ it reduces to IID sampling, so the batch instead uses $B = RK$ equal-mass strata, with one timestep drawn per stratum (global stratification, Eq. 63 in App. Sec. C.2). App. Fig. 24 confirms the $K > 1$ preference, and we use per-render stratification for SDS and DMD, where $K > 1$ is efficient; for data attribution, global stratification is most effective (Sec. 5.3).

Why the gains compose. Allocation trades the two terms of Eq. 10 against each other. Importance sampling with a good proposal and stratification both lower σ_W^2, by changing the measure and by removing between-stratum variance under it, and neither touches σ_A^2, so the three constructions compose and, empirically, their gains approximately multiply (Table 2; App. Sec. C.1.5); the across-unit term eventually limits all three.

4.2 Variance Measurement and Allocation Procedure

Variance estimation. We estimate the variance of each estimator (Eq. 1) with Welford's online algorithm [97], without storing samples, until the estimate converges (stopping rule in App. Sec. D.1.1). For some experiments, a high-sample reference validates convergence (the MSE to it agrees with the online estimate) and gives cosine-similarity measures of directional accuracy.

Pilot estimation. The same machinery estimates both terms of Eq. 10 from a short pilot (App. Sec. F.3): the Welford variance over re-noisings of each cached render, averaged over renders, estimates σ_W^2; the variance of the per-render means minus σ_W^2/K estimates σ_A^2 (extend the pilot if this estimate is not positive); with the timed costs, these give $K^\star$ (Sec. 4.1.1).

Allocation procedure. In practice: (i) time $c_{\text{render+encode}}$ and c_{denoise}; (ii) choose the proposal: if a known timestep weight dominates how gradient norms vary with t (w_{SDS} in SDS), sample from p times that weight with the likelihood-ratio correction, and otherwise keep p (Sec. 4.1.2); (iii) run a short pilot under that proposal at $K > 1$ to estimate σ_A^2 and σ_W^2 as above; (iv) set K to $\lfloor K^\star \rfloor$ or $\lceil K^\star \rceil$, whichever has the lower variance in Eq. 11 (at least 1); (v) stratify per render when $K > 1$ and globally when $K = 1$ (Sec. 4.1.3). Stratification lowers σ_W^2 further, so check the final configuration by its measured ECM. App. Sec. F.3 gives the full decision tree with thresholds.

Efficiency metrics. To compare estimators differing in both variance and cost (wall-clock), we use efficiency $\propto 1/(\text{Var} \cdot \text{cost})$ and report two metrics (> 1 better; see App. Fig. 9 for intuition):

- *Effective compute multiplier* (ECM) compares to a baseline (uniform-IID with $K = 1$) at iso-variance: $\text{ECM} = \text{cost}_{\text{baseline}}/\text{cost}_{\text{method}}$. Computing ECM requires estimating $\text{cost}_{\text{baseline}}$ at the method's variance; we interpolate in log-log space along the baseline Pareto curve, exploiting the standard Monte Carlo variance rate $\text{Var} \propto 1/R$ to extrapolate when needed.
- *Relative efficiency* (RE) compares to uniform-IID at identical (R, K): $\text{RE} = \text{Var}_{\text{u}}/\text{Var}_{\text{m}}$, isolating importance weighting (IW) and stratification from batch-size effects.

Task-specific quantities. For SDS (Sec. 5.1), we measure variance of $\hat{\mathbf{u}}_{\text{SDS}}(\boldsymbol{\theta})$ estimating $\mathbf{u}_{\text{SDS}}(\boldsymbol{\theta})$ (Eq. 6). For DMD (Sec. 5.2), we measure the variance of the generator gradient from the score difference, with a positive control (App. Sec. D.2.3). For data attribution (Sec. 5.3), we measure the variance of per-example gradients used for influence scores. Details in App. Sec. C.1.

What is classical and what is new. The decomposition in Eq. 10, the two-level allocation, importance sampling, stratification, and their composition through equal-mass quantiles are classical [10, 65]. New to this setting are: the identification of frozen-teacher gradient estimation as a two-level problem with asymmetric per-level costs; the teacher's own weight w_{SDS} as a zero-cost proxy for the variance-optimal proposal; stratified-IS timestep allocation composed with per-render stratification, an adaptation of the standard construction; a cost model separating $c_{\text{render+encode}}$ from c_{denoise} (ECM, RE); and parameter-gradient variance measurement across three applications, which exposes the DMD boundary (Sec. 5.2). Details in App. Sec. E.6.

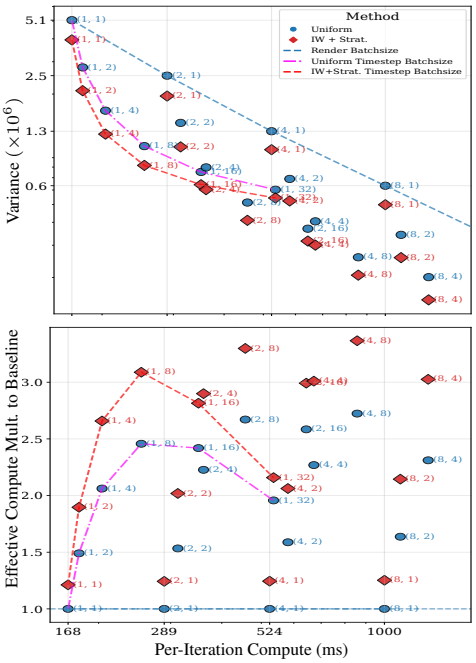


Figure 4: Quantifying variance reduction from IW and stratification (SDS). *Top:* Variance of the parameter-gradient estimator (Eq. 10) late in training vs. compute. Colors: uniform baseline and IW+Strat. Points annotated by (R, K). *Bottom:* Effective compute multiplier vs. uniform baseline. Lines trace $(R = 1, K)$ and peak at $K = 8$. Averaged over R, $K = 8$ gives $\sim 2.6\times$ (uniform), $\sim 3.3\times$ (IW+Strat; Table 1). Ablations in App. Fig. 11; RE in Table 2.

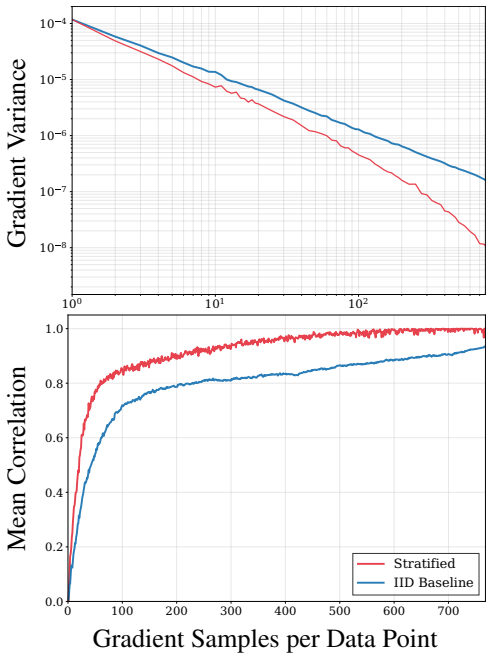


Figure 5: Quantifying Changes in Data Attribution. *Top:* Gradient variance vs. evaluations per data point. Stratified sampling beats uniform sampling at an equal budget. *Bottom:* Mean correlation of limited-evaluation rankings with ground-truth rankings ($\sim$1.3–3.8$\times$ compute multiplier at the tabulated budgets, $> 2\times$ at typical practical budgets; see Table 4 and App. Fig. 28). Qualitative examples in App. Fig. 30.

5 Experiments

We apply our methods to three diffusion-teacher tasks: optimization with diffusion priors (Sec. 5.1), single-step distillation (Sec. 5.2), and data attribution (Sec. 5.3). Across tasks, our framework reveals lower-variance setups per compute budget; details in App. Sec. D.

5.1 Diffusion Priors for Optimization

Setup: We use threestudio [18] with default hyperparameters per recent work [100]. Our *uniform* baseline is the standard SDS configuration used in DreamFusion [69] and Magic3D [38]: uniform timestep sampling on $[t_{\min}, t_{\max}]$ with one diffusion draw per render ($K = 1$); our methods are drop-in modifications that preserve the SDS objective. We measure: variance of SDS-latent-space updates and parameter gradients (and related dispersion); effective compute multipliers (how much baseline compute matches our variance); and CLIP scores [22] for coarse prompt alignment. We also compare fidelity using renders at matched iterations throughout training (Fig. 7; final renders in App. Fig. 12). Details in App. Sec. D.1.

Quantitative Results: At $K = 8$, ECM averages over R are $\sim 2.6\times$ for reuse alone and $\sim 3.3\times$ for IW+Strat (Table 1); ECM peaks at $K = 8$ and declines at $K \in \{16, 32\}$ as the across-unit term σ_A^2/R takes over (Sec. 4.1.1). Fig. 11 shows that, for $K \in \{2, 4, 8\}$ averaged over R, IW ($\sim$14–23%) and stratification ($\sim$6–12%) are complementary ($\sim$25–31% combined); Tables 1, 2 break this down. Fig. 6: across 30 prompts and 3 seeds, IW+Strat reaches each CLIP score of the standard-SDS baseline in fewer iterations, each costing less (Fig. 4). Per-render stratification beats global (App. Fig. 24); over our camera distribution (which we did not vary), the weight-based IW attains 94–97% of the relative efficiency of an empirical oracle that keeps the Jacobian ($\omega = 25$; App. Fig. 23, Table 6); IW+Strat reaches $\sim$91% of the efficiency of a Sinkhorn-optimal pair-probability allocation (per-snapshot, $N = 2$; App. Sec. D.1.6); the IW+Strat $>$ IW $\approx$ Strat $>$ Uniform ranking is stable across 5 prompts (App. Table 7).

Table 1: Effective compute multiplier (ECM) by re-noisings K. ECM is baseline cost divided by method cost at equal variance, using the uniform-IID $K{=}1$ curve (Sec. 4.2); higher is better. Averaged over the 5 variance prompts and over R.

K	Uniform	IW	Strat.	IW+Strat.
1	1.00×	**1.24**×	1.00×	**1.24**×
2	1.57×	1.93×	1.66×	**2.05**×
4	2.24×	2.68×	2.47×	**2.94**×
8	2.63×	3.01×	2.96×	**3.29**×
16	2.52×	2.75×	2.75×	**2.94**×
32	1.98×	2.09×	2.08×	**2.18**×

Table 2: Relative Efficiency (RE) versus (R, K)-uniform at each K, averaged over R. IW and stratification are complementary.

K	Uniform	IW	Strat.	IW+Strat.
1	1.00×	**1.24**×	1.00×	**1.24**×
2	1.00×	1.23×	1.06×	**1.30**×
4	1.00×	1.20×	1.11×	**1.31**×
8	1.00×	1.14×	1.12×	**1.25**×
16	1.00×	1.09×	1.09×	**1.17**×
32	1.00×	1.05×	1.05×	**1.10**×

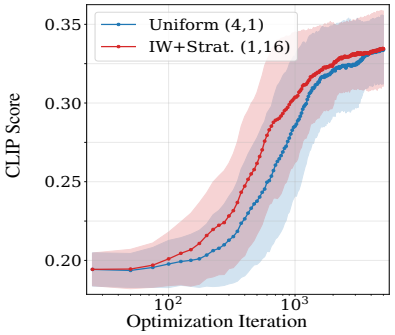


Figure 6: Performance Gains from Variance Reduction: CLIP score versus optimization iteration, averaged across 30 prompts, 3 seeds, and multiple views (± std. dev.). Baseline $(4, 1)$ vs. ours $(1, 16)$ (stratified+IS+re-noising), whose iterations cost less (Fig. 4). Higher CLIP at fixed iteration count from lower per-iteration variance (Fig. 11); final samples in App. Fig. 12.

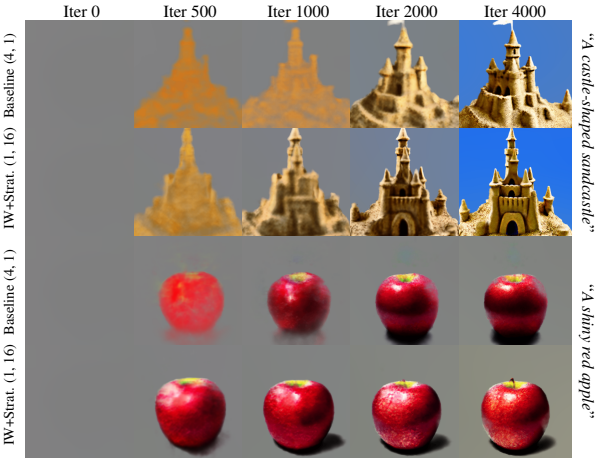


Figure 7: Qualitative Optimization Trajectories and Prompt Alignment: SDS renders over optimization. Baseline uses $(R, K){=}(4, 1)$; ours uses $(1, 16)$, whose iterations cost less, reaching comparable quality in fewer iterations (ECM averaged over R peaks at ∼3.3×, Table 1). Columns: iterations 0/500/1000/2000/4000. Qualitative improvements track CLIP-score trends in Fig. 6.

5.2 Single-Step Diffusion Distillation

Setup: We apply our methods to Distribution Matching Distillation (DMD) [105], in its DMD2 form [104], via the Monte Carlo estimator of Eq. 51, on top of the FastGen [64] reference implementation. We train generators on ImageNet-256 [12] using the pretrained DiT-XL/2 teacher [68]; details in App. Sec. D.2. The measured cost ratio is $c_{\text{render+encode}}/c_{\text{denoise}} \approx 26$, so upstream cost dominates (Sec. 4.1.1).

Results: Table 3 shows reuse cuts gradient variance by 3.4–16× at fixed $R = 8$, with ∼1.55× per-iteration wall-clock cost (App. Sec. D.2). Stratification adds 1.0–2.0× *at matched compute* within $(8, 16)$. The largest variance reduction is in parameter gradients, where combining reuse $(8, 16)$ with stratification yields ∼32× over baseline $(8, 1)$ (∼20× variance reduction per unit wall-clock). While variance reduction yields similar-or-better per-step FID convergence, no practical improvement remains at matched wall-clock time (App. Fig. 26, App. Fig. 25). We report DMD as a negative result and treat it as a scope condition of the framework: lower gradient variance does not improve FID at matched wall-clock time (condition (3) in Sec. 6). App. Sec. D.2.3 rules out insufficient compute headroom; generator-input diversity is most consistent with the data, while auxiliary-objective effects are plausible but not isolated. Applying reuse and stratification to the fake-score update likewise improves per-step convergence at higher compute, and the benefit fades once adjusted for compute (App. Fig. 27). Detailed hypotheses and ablations are in App. Sec. D.2.

5.3 Data Attribution

Setup: We follow MOTIVE [98] for video data attribution, using `Wan2.1-T2V-1.3B` [92], a flow-matching video model that illustrates our framework's reach beyond noise-prediction diffusion teachers (DiffSynth-Studio implementation), on `VIDGEN-1M` [89]; details in App. Sec. D.3. Data

Table 3: DMD variance at iter. 20k: variance (Eq. 1) of the Monte Carlo estimates of the teacher score, the score difference, and the parameter gradient. *Reuse* ($(8, 16)$ vs $(8, 1)$) reduces variance 3.4–16× *at higher compute* (16× more denoiser calls). *Stratification* ($(8, 16)$ Strat. vs IID) reduces variance 1–2× at *matched* compute. FID does not improve at matched wall-clock.

Method	Teacher Score Var	Score Diff. Var	Param. Grad. Var
$(8, 1)$ IID	3954	71.3	982
$(8, 16)$ IID	1175	13.3	59.9
$(8, 16)$ Strat.	1169	12.0	30.7

Table 4: IID vs. stratified sampling for data-attribution gradients. Correlations are measured against the full-budget attribution, which stratified sampling reproduces at 768 timesteps (App. Sec. D.3.2); stratified sampling gives $> 2\times$ compute multipliers at 64 to 256 timesteps.

Budget (timesteps)	IID Corr.	Stratified Corr.	Effective Multiplier
4	0.065	0.166	2.44×
16	0.245	0.407	1.80×
64	0.616	0.805	3.82×
256	0.813	0.916	2.85×
768	0.934	1.000	1.28×

attribution computes the influence of each training datum on a query, which MOTIVE uses to select fine-tuning data. We assess unbiased gradient estimators via gradient variance and the correlation between the rankings they produce and ground-truth rankings.

Results: Fig. 5 quantifies gradient variance (top) and ranking correlation (bottom) versus gradient evaluations per data point. Variance decreases with more samples, and stratified sampling consistently beats uniform at equal budget. This yields influence rankings that better match those computed from ground-truth gradients, achieving $> 2\times$ effective compute multipliers at moderate budgets of 64 to 256 timesteps (Table 4). Re-noising provides less benefit here than in other tasks because encoding cost is moderate relative to denoising, and we require accurate gradients for each fixed training example rather than averaging over sampled inputs. Global stratification is most effective and substantially reduces variance in this setting.

6 Discussion

The App. covers extended related work (App. Sec. E), limitations (App. Sec. F.1), and future directions (App. Sec. F.2). To our knowledge, no published work in the cited frozen-teacher SDS, DMD, or attribution lines applies timestep stratification, uses explicit per-loss weights as IS proxies, or measures parameter-gradient variance per unit compute for these tasks.

When Variance Reduction Helps: CARV helps when (1) the MC gradient dominates, (2) $c_{\text{render+encode}} > c_{\text{denoise}}$, and (3) variance limits convergence. Conditions (1) and (2) can be checked from cost measurements and a short pilot run (Sec. 4.2); DMD, the case that tests (3), needed training runs. A cost-ratio sweep confirms the $K^\star$ predictions of Sec. 4.1.1 and shows that condition (2) matters for re-noising only: at zero render cost, $K = 1$ is best for every method, and IW+Strat stays above the baseline at small K because of IW (App. Sec. D.1.7). In SDS, IW+Strat reaches $\sim$91% of the efficiency of a Sinkhorn-optimized pair allocation (App. Sec. D.1.6, App. Fig. 20). At low classifier-free guidance, $\omega = 25$, IW+Strat reaches an ECM of $\sim 3.8\times$ at $(1, 8)$ (App. Fig. 16), with CLIP-score and qualitative gains across prompts and seeds (App. Sec. D.1.5). Cost selection (thresholds in App. Sec. F.3): reuse when $c_{\text{render+encode}} \gg c_{\text{denoise}}$ ($\sim$1.6–2.6× in SDS), stratification when $c_{\text{render+encode}} \approx c_{\text{denoise}}$ ($\sim$1.3–3.8× in attribution), and IS when a known weight dominates how the gradient norm varies with t ($\sim 1.2\times$).

Broader Implications: The framework (Sec. 4.2) is application-agnostic wherever upstream cost dominates the denoiser; the DMD case suggests gradient-variance reduction does not improve FID at matched wall-clock time when auxiliary stabilizers or input diversity bind.

6.1 Conclusion

We presented CARV, a compute-aware variance-accounting framework for frozen-teacher Monte Carlo gradients, motivating a hierarchical Monte Carlo estimator with three unbiased drop-ins: timestep importance sampling, stratification, and amortized compute reuse. In our SDS and attribution settings, CARV delivers 2 to 3× effective compute multipliers without changing the objective; in DMD, the same techniques cut gradient variance by an order of magnitude without improving downstream FID, marking the boundary where MC variance no longer governs convergence.

Acknowledgments and Disclosure of Funding

We thank the Fundamental Generative AI Research (GenAIR) group at NVIDIA for many helpful discussions during this work. We thank Sanja Fidler and the Spatial Intelligence Lab (SIL) for hosting the internship that made this collaboration possible. This work was funded by NVIDIA; all authors were affiliated with NVIDIA during this work. Jesse Bettencourt also acknowledges support from the University of Toronto and the Vector Institute, and Xindi Wu from Princeton University. Resources used in preparing this research were provided, in part, by the Province of Ontario, the Government of Canada through CIFAR, and companies sponsoring the Vector Institute. The authors declare no other competing interests.

References

[1] Juhan Bae, Wu Lin, Jonathan Lorraine, and Roger Grosse. Training data attribution via approximate unrolling. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://arxiv.org/abs/2405.12186. 52

[2] Sherwin Bahmani, Ivan Skorokhodov, Victor Rong, Gordon Wetzstein, Leonidas Guibas, Peter Wonka, Sergey Tulyakov, Jeong Joon Park, Andrea Tagliasacchi, and David B. Lindell. 4D-fy: Text-to-4D generation using hybrid score distillation sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7996–8006, 2024. 4, 52

[3] Benedikt Bitterli, Chris Wyman, Matt Pharr, Peter Shirley, Aaron Lefohn, and Wojciech Jarosz. Spatiotemporal reservoir resampling for real-time ray tracing with dynamic direct lighting. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 39(4):148:1–148:16, 2020. doi: 10.1145/3386569.3392481. 2, 55

[4] Jonathan Brokman, Omer Hofman, Roman Vainshtein, Amit Giloni, Toshiya Shimizu, Inderjeet Singh, Oren Rachmil, Alon Zolfi, Asaf Shabtai, Yuki Unno, et al. MONTRAGE: Monitoring training for attribution of generative diffusion models. In *European Conference on Computer Vision*, pages 1–17. Springer, 2024. 52

[5] Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, Clarence Ng, Ricky Wang, and Aditya Ramesh. Video generation models as world simulators. OpenAI technical report. https://openai.com/research/video-generation-models-as-world-simulators, 2024. 51

[6] Itay Chachy, Guy Yariv, and Sagie Benaim. RewardSDS: Aligning score distillation via reward-weighted sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2026. 2

[7] Rui Chen, Yongwei Chen, Ningxin Jiao, and Kui Jia. Fantasia3D: Disentangling geometry and appearance for high-quality text-to-3D content creation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22246–22256, 2023. 52

[8] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025. 53

[9] Jooyoung Choi, Jungbeom Lee, Chaehun Shin, Sungwon Kim, Hyunwoo Kim, and Sungroh Yoon. Perception prioritized training of diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022. URL https://arxiv.org/abs/2204.00227. 2, 52

[10] William G. Cochran. *Sampling Techniques*. John Wiley & Sons, third edition, 1977. 2, 4, 5, 6, 7, 21, 54

[11] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems*, volume 26, pages 2292–2300, 2013. 35

[12] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255, 2009. 9, 47

[13] Kangle Deng, Timothy Omernick, Alexander Weiss, Deva Ramanan, Jun-Yan Zhu, Tinghui Zhou, and Maneesh Agrawala. FlashTex: Fast relightable mesh texturing with LightControlNet. In *European Conference on Computer Vision*, pages 90–107. Springer, 2024. 4, 52

[14] Zach Evans, Julian D. Parker, CJ Carr, Zack Zukowski, Josiah Taylor, and Jordi Pons. Stable Audio Open. In *ICASSP 2025 – 2025 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 1–5. IEEE, 2025. 51

[15] Kristian Georgiev, Joshua Vendrow, Hadi Salman, Sung Min Park, and Aleksander Madry. The journey, not the destination: How data guides diffusion models. *arXiv preprint arXiv:2312.06205*, 2023. 52

[16] Paul Glasserman. *Monte Carlo Methods in Financial Engineering*. Springer, New York, 2004. doi: 10.1007/978-0-387-21617-1. 2, 6

[17] Peter W. Glynn and Ward Whitt. The asymptotic efficiency of simulation estimators. *Operations Research*, 40(3):505–520, 1992. doi: 10.1287/opre.40.3.505. 5

[18] Yuan-Chen Guo, Ying-Tian Liu, Ruizhi Shao, Christian Laforte, Vikram Voleti, Guan Luo, Chia-Hao Chen, Zi-Xin Zou, Chen Wang, Yan-Pei Cao, and Song-Hai Zhang. threestudio: A unified framework for 3D content generation. `https://github.com/threestudio-project/threestudio`, 2023. 8, 19, 28, 56

[19] J. M. Hammersley and D. C. Handscomb. *Monte Carlo Methods*. Methuen, London, 1964. 5

[20] Tiankai Hang, Shuyang Gu, Chen Li, Jianmin Bao, Dong Chen, Han Hu, Xin Geng, and Baining Guo. Efficient diffusion training via Min-SNR weighting strategy. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7441–7451, 2023. 2, 52, 55

[21] Morris H. Hansen and William N. Hurwitz. On the theory of sampling from finite populations. *The Annals of Mathematical Statistics*, 14(4):333–362, 1943. doi: 10.1214/aoms/1177731356. 2, 4

[22] Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. CLIPScore: A reference-free evaluation metric for image captioning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7514–7528, 2021. 8

[23] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851, 2020. 2, 51, 52

[24] Emiel Hoogeboom, Jonathan Heek, and Tim Salimans. simple diffusion: End-to-end diffusion for high resolution images. In *International Conference on Machine Learning*, volume 202, pages 13213–13232. PMLR, 2023. 2, 52

[25] Safwan Hossain and Jonathan Lorraine. JacNet: Learning functions with structured Jacobian. In *First Workshop on Invertible Neural Nets and Normalizing Flows (INNF), ICML*, 2019. 55

[26] Yukun Huang, Jianan Wang, Yukai Shi, Boshi Tang, Xianbiao Qi, and Lei Zhang. DreamTime: An improved optimization strategy for diffusion-guided 3D generation. In *International Conference on Learning Representations*, 2024. 2

[27] Michael Janner, Yilun Du, Joshua B. Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning*, volume 162, pages 9902–9915. PMLR, 2022. URL `https://arxiv.org/abs/2205.09991`. 53

[28] H. Kahn and A. W. Marshall. Methods of reducing sample size in Monte Carlo computations. *Journal of the Operations Research Society of America*, 1(5):263–278, 1953. doi: 10.1287/opre.1.5.263. 3, 6

[29] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *Advances in Neural Information Processing Systems*, volume 35, pages 26565–26577, 2022. 2, 52

[30] Tero Karras, Miika Aittala, Jaakko Lehtinen, Janne Hellsten, Timo Aila, and Samuli Laine. Analyzing and improving the training dynamics of diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024. URL https://arxiv.org/abs/2312.02696. 52

[31] Bahjat Kawar, Michael Elad, Stefano Ermon, and Jiaming Song. Denoising diffusion restoration models. In *Advances in Neural Information Processing Systems*, volume 35, pages 23593–23606, 2022. 53

[32] Dongjun Kim, Seungjae Shin, Kyungwoo Song, Wanmo Kang, and Il-Chul Moon. Soft truncation: A universal training technique of score-based diffusion model for high precision score estimation. In *International Conference on Machine Learning*, volume 162, pages 11201–11228. PMLR, 2022. URL https://arxiv.org/abs/2106.05527. 52

[33] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. 29

[34] Diederik P. Kingma and Ruiqi Gao. Understanding diffusion objectives as the ELBO with simple data augmentation. In *Advances in Neural Information Processing Systems*, volume 36, pages 65484–65516, 2023. 2

[35] Diederik P. Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. In *Advances in Neural Information Processing Systems*, volume 34, 2021. URL https://arxiv.org/abs/2107.00630. 2, 3, 19, 22, 52, 54, 55

[36] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning*, pages 1885–1894. PMLR, 2017. 25, 52

[37] Yongchan Kwon, Eric Wu, Kevin Wu, and James Zou. DataInf: Efficiently estimating data influence in LoRA-tuned LLMs and diffusion models. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2310.00902. 52

[38] Chen-Hsuan Lin, Jun Gao, Luming Tang, Towaki Takikawa, Xiaohui Zeng, Xun Huang, Karsten Kreis, Sanja Fidler, Ming-Yu Liu, and Tsung-Yi Lin. Magic3D: High-resolution text-to-3D content creation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 300–309, 2023. 2, 8, 52

[39] Daqi Lin, Markus Kettunen, Benedikt Bitterli, Jacopo Pantaleoni, Cem Yuksel, and Chris Wyman. Generalized resampled importance sampling: Foundations of ReSTIR. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 41(4):75:1–75:23, 2022. doi: 10.1145/3528223.3530158. 2, 55

[40] Jinxu Lin, Linwei Tao, Minjing Dong, and Chang Xu. Diffusion attribution score: Evaluating training data influence in diffusion models. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2410.18639. 52

[41] Huan Ling, Seung Wook Kim, Antonio Torralba, Sanja Fidler, and Karsten Kreis. Align your Gaussians: Text-to-4D with dynamic 3D Gaussians and composed diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8576–8588, 2024. 52

[42] Fangfu Liu, Hanyang Wang, Shunyu Yao, Shengjun Zhang, Jie Zhou, and Yueqi Duan. Physics3D: Learning physical properties of 3D Gaussians via video diffusion. *arXiv preprint arXiv:2406.04338*, 2024. 4, 52

[43] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=S1eYHoC5FX. 55

[44] Sharon L. Lohr. *Sampling: Design and Analysis*. Chapman and Hall/CRC, third edition, 2021. doi: 10.1201/9780429298899. 2, 4

[45] Jonathan Lorraine. *Scalable Nested Optimization for Deep Learning*. PhD thesis, University of Toronto, 2024. 53, 55

[46] Jonathan Lorraine, Nihesh Anderson, Chansoo Lee, Quentin De Laroussilhe, and Mehadi Hassen. Task selection for AutoML system evaluation. *arXiv preprint arXiv:2208.12754*, 2022. 55

[47] Jonathan Lorraine, Paul Vicol, Jack Parker-Holder, Tal Kachman, Luke Metz, and Jakob Foerster. Lyapunov exponents for diversity in differentiable games. In *International Conference on Autonomous Agents and Multiagent Systems*, pages 842–852, 2022. 55

[48] Jonathan Lorraine, Kevin Xie, Xiaohui Zeng, Chen-Hsuan Lin, Towaki Takikawa, Nicholas Sharp, Tsung-Yi Lin, Ming-Yu Liu, Sanja Fidler, and James Lucas. ATT3D: Amortized text-to-3D object synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17946–17956, 2023. 52, 53, 54, 55

[49] Jonathan P. Lorraine, David Acuna, Paul Vicol, and David Duvenaud. Complex momentum for optimization in games. In *International Conference on Artificial Intelligence and Statistics*, volume 151, pages 7742–7765. PMLR, 2022. 55

[50] Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. RePaint: Inpainting using denoising diffusion probabilistic models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11461–11471, 2022. 53

[51] Artem Lukoianov, Haitz Sáez de Ocáriz Borde, Kristjan Greenewald, Vitor Guizilini, Timur Bagautdinov, Vincent Sitzmann, and Justin M. Solomon. Score distillation via reparametrized DDIM. In *Advances in Neural Information Processing Systems*, volume 37, pages 26011–26044, 2024. 52, 53, 54

[52] Weijian Luo, Tianyang Hu, Shifeng Zhang, Jiacheng Sun, Zhenguo Li, and Zhihua Zhang. Diff-Instruct: A universal approach for transferring knowledge from pre-trained diffusion models. In *Advances in Neural Information Processing Systems*, volume 36, 2023. 52

[53] Zhiyuan Ma, Yuxiang Wei, Yabin Zhang, Xiangyu Zhu, Zhen Lei, and Lei Zhang. ScaleDreamer: Scalable text-to-3D synthesis with asynchronous score distillation. In *European Conference on Computer Vision*, pages 1–19. Springer, 2024. 52

[54] Dougal Maclaurin, David Duvenaud, and Ryan Adams. Gradient-based hyperparameter optimization through reversible learning. In *International Conference on Machine Learning*, volume 37, pages 2113–2122. PMLR, 2015. 53, 55

[55] David McAllister, Songwei Ge, Jia-Bin Huang, David W. Jacobs, Alexei A. Efros, Aleksander Holynski, and Angjoo Kanazawa. Rethinking score distillation as a bridge between image distributions. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://arxiv.org/abs/2406.09417. 52

[56] Nikhil Mehta, Jonathan Lorraine, Steve Masson, Ramanathan Arunachalam, Zaid Pervaiz Bhat, James Lucas, and Arun George Zachariah. Improving hyperparameter optimization with checkpointed model weights. In *European Conference on Computer Vision Workshop on Efficient Deep Learning for Foundation Models (EFM)*, pages 75–96, 2024. doi: 10.1007/978-3-031-91979-4_8. 55

[57] Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. SDEdit: Guided image synthesis and editing with stochastic differential equations. In *International Conference on Learning Representations*, 2022. URL https://arxiv.org/abs/2108.01073. 53

[58] Bruno Kacper Mlodozeniec, Runa Eschenhagen, Juhan Bae, Alexander Immer, David Krueger, and Richard E. Turner. Influence functions for scalable data attribution in diffusion models. In *International Conference on Learning Representations*, 2025. 52

[59] Shakir Mohamed, Mihaela Rosca, Michael Figurnov, and Andriy Mnih. Monte Carlo gradient estimation in machine learning. *Journal of Machine Learning Research*, 21(132):1–62, 2020. 2

[60] Chong Mou, Xintao Wang, Liangbin Xie, Yanze Wu, Jian Zhang, Zhongang Qi, and Ying Shan. T2I-Adapter: Learning adapters to dig out more controllable ability for text-to-image diffusion models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 4296–4304, 2024. 53

[61] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 41(4):102:1–102:15, 2022. doi: 10.1145/3528223.3530127. 28, 53, 56

[62] Thuan Hoang Nguyen and Anh Tran. SwiftBrush: One-step text-to-image diffusion model with variational score distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7807–7816, 2024. 52

[63] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, volume 139, pages 8162–8171. PMLR, 2021. 2, 3, 20, 52

[64] Weili Nie, Julius Berner, Chao Liu, and Arash Vahdat. NVIDIA FastGen: Fast generation from diffusion models. https://github.com/NVlabs/FastGen, 2026. 9, 19

[65] Art B. Owen. Monte Carlo theory, methods and examples. https://artowen.su.domains/mc/, 2013. 2, 6, 7, 33, 34, 52, 54

[66] Sung Min Park, Kristian Georgiev, Andrew Ilyas, Guillaume Leclerc, and Aleksander Madry. TRAK: Attributing model behavior at scale. In *International Conference on Machine Learning*, pages 27074–27113. PMLR, 2023. 2, 25, 50, 52

[67] Fabian Pedregosa. Hyperparameter optimization with approximate gradient. In *International Conference on Machine Learning*, volume 48, pages 737–746. PMLR, 2016. 53, 55

[68] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4195–4205, 2023. 9, 47, 51, 58

[69] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. DreamFusion: Text-to-3D using 2D diffusion. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2209.14988. 2, 4, 8, 52

[70] Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. Estimating training data influence by tracing gradient descent. In *Advances in Neural Information Processing Systems*, volume 33, pages 19920–19930, 2020. 25, 52

[71] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021. 29

[72] Jiawei Ren, Liang Pan, Jiaxiang Tang, Chi Zhang, Ang Cao, Gang Zeng, and Ziwei Liu. DreamGaussian4D: Generative 4D Gaussian splatting. *arXiv preprint arXiv:2312.17142*, 2023. 52

[73] Jessie Richter-Powell, Jonathan Lorraine, and Brandon Amos. Input convex gradient networks. In *Advances in Neural Information Processing Systems Optimal Transport and Machine Learning Workshop*, 2021. 55

[74] Jessie Richter-Powell, Antonio Torralba, and Jonathan Lorraine. Score distillation sampling for audio: Source separation, synthesis, and beyond. arXiv preprint arXiv:2505.04621, 2025. Presented at the ICML 2025 AI Heard That! Workshop on Machine Learning for Audio. 4, 52, 55

[75] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022. 2, 28, 47, 51, 56

[76] Reuven Y. Rubinstein and Dirk P. Kroese. *Simulation and the Monte Carlo Method*. John Wiley & Sons, third edition, 2016. 2, 3, 6, 20, 52

[77] Amirmojtaba Sabour, Sanja Fidler, and Karsten Kreis. Align your flow: Scaling continuous-time flow map distillation. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2506.14603. 52

[78] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*, 2022. URL https://arxiv.org/abs/2202.00512. 2, 19, 52

[79] Tim Salimans, Thomas Mensink, Jonathan Heek, and Emiel Hoogeboom. Multistep distillation of diffusion models via moment matching. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://arxiv.org/abs/2406.04103. 2, 52

[80] Axel Sauer, Dominik Lorenz, Andreas Blattmann, and Robin Rombach. Adversarial diffusion distillation. In *European Conference on Computer Vision*. Springer, 2024. URL https://arxiv.org/abs/2311.17042. 52

[81] Yichun Shi, Peng Wang, Jianglong Ye, Long Mai, Kejie Li, and Xiao Yang. MVDream: Multi-view diffusion for 3D generation. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2308.16512. 52, 56

[82] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, volume 37, pages 2256–2265. PMLR, 2015. 51

[83] Yang Song and Prafulla Dhariwal. Improved techniques for training consistency models. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2310.14189. 52

[84] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *Advances in Neural Information Processing Systems*, volume 32, pages 11918–11930, 2019. 51

[85] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2011.13456. 51

[86] Yang Song, Liyue Shen, Lei Xing, and Stefano Ermon. Solving inverse problems in medical imaging with score-based generative models. In *International Conference on Learning Representations*, 2022. URL https://arxiv.org/abs/2111.08005. 53

[87] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *International Conference on Machine Learning*, volume 202, pages 32211–32252. PMLR, 2023. URL https://arxiv.org/abs/2303.01469. 52

[88] Yanke Song, Jonathan Lorraine, Weili Nie, Karsten Kreis, and James Lucas. Multi-student diffusion distillation for better one-step generators. *arXiv preprint arXiv:2410.23274*, 2024. ICML 2025 Efficient Systems for Foundation Models (ES-FoMo) Workshop. 52, 54, 55

[89] Zhiyu Tan, Xiaomeng Yang, Luozheng Qin, and Hao Li. VidGen-1M: A large-scale dataset for text-to-video generation. arXiv preprint arXiv:2408.02629, 2024. 9, 49, 51

[90] Vikas Thamizharasan, Nikitas Chatzis, Iliyan Georgiev, Matthew Fisher, Evangelos Kalogerakis, Difan Liu, Nanxuan Zhao, and Michal Lukáč. Mean-shift distillation for diffusion mode seeking. In *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2026. URL https://arxiv.org/abs/2502.15989. 54

[91] Steven K. Thompson. *Sampling*. Number 755 in Wiley Series in Probability and Statistics. John Wiley & Sons, third edition, 2012. 2, 3, 21, 33, 54

[92] Wan Team, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, et al. Wan: Open and advanced large-scale video generative models. arXiv preprint arXiv:2503.20314, 2025. 9, 49, 51

[93] Haochen Wang, Xiaodan Du, Jiahao Li, Raymond A. Yeh, and Greg Shakhnarovich. Score Jacobian chaining: Lifting pretrained 2D diffusion models for 3D generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12619–12629, 2023. 52

[94] Peihao Wang, Zhiwen Fan, Dejia Xu, Dilin Wang, Sreyas Mohan, Forrest Iandola, Rakesh Ranjan, Yilei Li, Qiang Liu, Zhangyang Wang, and Vikas Chandra. SteinDreamer: Variance reduction for text-to-3D score distillation via Stein identity. In *International Conference on Artificial Intelligence and Statistics*, volume 258, pages 4024–4032. PMLR, 2025. 2, 52, 53

[95] Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan Li, Hang Su, and Jun Zhu. ProlificDreamer: High-fidelity and diverse text-to-3D generation with variational score distillation. In *Advances in Neural Information Processing Systems*, volume 36, pages 8406–8441, 2023. 2, 52, 53, 54, 55, 58

[96] Zhengyi Wang, Jonathan Lorraine, Yikai Wang, Hang Su, Jun Zhu, Sanja Fidler, and Xiaohui Zeng. LLaMA-Mesh: Unifying 3D mesh generation with language models. *arXiv preprint arXiv:2411.09595*, 2024. 52, 53, 55

[97] B. P. Welford. Note on a method for calculating corrected sums of squares and products. *Technometrics*, 4(3):419–420, 1962. 7

[98] Xindi Wu, Despoina Paschalidou, Jun Gao, Antonio Torralba, Laura Leal-Taixé, Olga Russakovsky, Sanja Fidler, and Jonathan Lorraine. Motion attribution for video generation. In *International Conference on Machine Learning*, volume 306, pages 136625–136646. PMLR, 2026. URL https://proceedings.mlr.press/v306/wu26aq.html. 2, 4, 9, 19, 25, 49, 52, 55, 66

[99] Cheng Xie, Jonathan Lorraine, Xiaohui Zeng, James Lucas, Jun Gao, and Sanja Fidler. 3D model generation using multimodal generative AI, 2026. U.S. Patent 12,555,343 B2; granted 2026-02-17. Filed as U.S. Patent App. 18/622,045. 52, 53, 55

[100] Kevin Xie, Jonathan Lorraine, Tianshi Cao, Jun Gao, James Lucas, Antonio Torralba, Sanja Fidler, and Xiaohui Zeng. LATTE3D: Large-scale amortized text-to-Enhanced3D synthesis. In *European Conference on Computer Vision*, pages 305–322. Springer, 2024. 8, 52, 53, 54, 55

[101] Sirui Xie, Zhisheng Xiao, Diederik P. Kingma, Tingbo Hou, Ying Nian Wu, Kevin Patrick Murphy, Tim Salimans, Ben Poole, and Ruiqi Gao. EM distillation for one-step diffusion models. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://arxiv.org/abs/2405.16852. 52

[102] Tong Xie, Haoyu Li, Andrew Bai, and Cho-Jui Hsieh. Data attribution for diffusion models: Timestep-induced bias in influence estimation. *Transactions on Machine Learning Research*, 2024. URL https://arxiv.org/abs/2401.09031. 2, 25

[103] Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, et al. CogVideoX: Text-to-video diffusion models with an expert transformer. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2408.06072. 51

[104] Tianwei Yin, Michaël Gharbi, Taesung Park, Richard Zhang, Eli Shechtman, Fredo Durand, and William T. Freeman. Improved distribution matching distillation for fast image synthesis. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://arxiv.org/abs/2405.14867. 2, 9, 24, 47, 52

[105] Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Fredo Durand, William T. Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6613–6623, 2024. 2, 4, 9, 23, 52

[106] Xin Yu, Yuan-Chen Guo, Yangguang Li, Ding Liang, Song-Hai Zhang, and Xiaojuan Qi. Text-to-3D with classifier score distillation. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2310.19415. 52

[107] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3836–3847, 2023. 53

[108] Tianyuan Zhang, Hong-Xing Yu, Rundi Wu, Brandon Y. Feng, Changxi Zheng, Noah Snavely, Jiajun Wu, and William T. Freeman. PhysDreamer: Physics-based interaction with 3D objects via video generation. In *European Conference on Computer Vision*, pages 388–406. Springer, 2024. 52

[109] Tianyi Zheng, Cong Geng, Peng-Tao Jiang, Ben Wan, Hao Zhang, Jinwei Chen, Jia Wang, and Bo Li. Non-uniform timestep sampling: Towards faster diffusion model training. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 7036–7045, 2024. 3, 20

[110] Xiaosen Zheng, Tianyu Pang, Chao Du, Jing Jiang, and Min Lin. Intriguing properties of data attribution on diffusion models. In *International Conference on Learning Representations*, 2024. 52

[111] Mingyuan Zhou, Huangjie Zheng, Zhendong Wang, Mingzhang Yin, and Hai Huang. Score identity distillation: Exponentially fast distillation of pretrained diffusion models for one-step generation. In *International Conference on Machine Learning*, volume 235, pages 62307–62331. PMLR, 2024. 52

[112] Mingyuan Zhou, Zhendong Wang, Huangjie Zheng, and Hai Huang. Guided score identity distillation for data-free one-step text-to-image generation. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2406.01561. 52

[113] Junzhe Zhu, Peiye Zhuang, and Sanmi Koyejo. HiFA: High-fidelity text-to-3D generation with advanced diffusion guidance. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2305.18766. 52

A Broader Impacts

Our work improves compute efficiency of pipelines that use pretrained diffusion models as frozen teachers by reducing Monte Carlo estimator variance without changing the target objective. By reallocating samples across noise levels, stratifying timesteps, and reusing expensive upstream computations, practitioners can achieve comparable gradient quality with fewer denoiser, rendering, or encoding evaluations, which reduces energy use and the cost of experiments. The same techniques could also lower the cost of developing or deploying systems that generate synthetic media, which may be misused for deception or harmful content. This paper does not introduce new generative capabilities or datasets, and responsible use should follow the safety, provenance, and content policies of the underlying models. Measuring variance per unit of compute also makes comparisons between estimators fairer under a fixed budget.

LLM Usage

We used a large language model as a writing and engineering assistant during the preparation of this manuscript. Specifically, it was used to (i) suggest edits for clarity and concision, (ii) help reorganize prose and LaTeX for readability, and (iii) assist with routine coding tasks (e.g., debugging scripts and preparing plotting utilities). All technical contributions, methodological decisions, experimental design, and results are our own. We verified all generated suggestions and did not rely on the model for new scientific claims or conclusions.

Reproducibility

We take several steps to ensure reproducibility of our results. First, we provide complete mathematical specifications of all proposed estimators, including importance sampling (Sec. 4.1.2), stratified

sampling (Sec. 4.1.3), and compute reuse (Sec. 4.1.1), with explicit equations and a combined-pipeline pseudocode (Algorithm 1) that can be directly implemented. Second, we build on established open-source codebases: threestudio [18] for SDS (Sec. 5.1), the FastGen [64] reference implementation for DMD (Sec. 5.2), and MOTIVE [98] (DiffSynth-Studio backbone) for video data attribution (Sec. 5.3); each of our estimators in Sec. 4.1 is a small change to the per-step sampling and re-noising logic in these pipelines, not a structural change to the surrounding training loop. Third, we report all experimental settings, including batch sizes, compute budgets, number of renders and re-noises, and evaluation metrics, with additional hyperparameters and implementation details provided in App. Sec. D. Fourth, our experiments average results over multiple seeds and prompts and report standard deviations to quantify uncertainty (Fig. 6). Fifth, our variance measurement framework (Sec. 4.2) uses standard techniques (Welford's algorithm) and verifies unbiasedness by comparing the MSE and variance, enabling independent validation of our claims. A glossary of all notation is included in Sec. G.

B Additional Background

B.1 Diffusion Models

Weighted diffusion training objective. Let $\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\phi}}(\mathbf{z}_t, t, \mathbf{c})$ be the predicted noise at $(\mathbf{z}_t, t, \mathbf{c})$ with parameters $\boldsymbol{\phi}$. A common noise-prediction objective is:

$$\mathcal{L}_{\text{Diff}}(\boldsymbol{\phi}) = \mathop{\mathbb{E}}_{(\mathbf{x},\mathbf{c}),t,\boldsymbol{\epsilon}}\left[\|\mathbf{r}\|_2^2\right] = \mathop{\mathbb{E}}_{(\mathbf{x},\mathbf{c}),t,\boldsymbol{\epsilon}}\left[\ell_{\text{Diff}}(\text{Encode}(\mathbf{x}), \mathbf{c}, t, \boldsymbol{\epsilon}, \boldsymbol{\phi})\right] \tag{15}$$

where the residual $\mathbf{r} := \hat{\boldsymbol{\epsilon}}_{\boldsymbol{\phi}}(\mathbf{z}_t, t, \mathbf{c}) - \boldsymbol{\epsilon}$ and the per-sample cost $\ell_{\text{Diff}} := \|\mathbf{r}\|_2^2$. Following Kingma et al. [35], we use the weighted objective:

$$\mathcal{L}_{\text{wDiff}}(\boldsymbol{\phi}) = \mathop{\mathbb{E}}_{(\mathbf{x},\mathbf{c}),t,\boldsymbol{\epsilon}}\left[w(t)\ell_{\text{Diff}}(\text{Encode}(\mathbf{x}), \mathbf{c}, t, \boldsymbol{\epsilon}, \boldsymbol{\phi})\right]. \tag{16}$$

Different prediction parameterizations (e.g., x- or v-prediction) can be written in this form with an appropriate choice of $w(t)$ and the corresponding model output [35, 78].

B.1.1 Sampling from Diffusion Models

To sample, a pretrained latent diffusion model uses a multi-step sampler that starts from Gaussian noise and iteratively denoises. Let $\{t_k\}_{k=0}^{K}$ denote a discretization of the continuous noise schedule with $t_0 \approx 0$ and $t_K \approx 1$. The sampler initializes the latent at the highest noise level, $\mathbf{z}_{t_K} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, then applies a sequence of learned transitions:

$$\mathbf{z}_{t_{k-1}} \sim p_{\boldsymbol{\phi}}(\mathbf{z}_{t_{k-1}} \mid \mathbf{z}_{t_k}, t_k, \mathbf{c}), \qquad k = K, \ldots, 1 \tag{17}$$

where the transition kernels are determined by the denoiser $\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\phi}}(\mathbf{z}_{t_k}, t_k, \mathbf{c})$ and the chosen update rule (e.g., DDPM or a DDIM-like update). The composition of these K steps defines a stochastic generator that maps a single Gaussian seed $\mathbf{z}_{t_K} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ to a clean latent $\mathbf{z}_{t_0}$. In App. Sec. B.3.2, we treat this K-step procedure as the teacher and train a one-step generator $G_{\boldsymbol{\theta}}$ to match its sample distribution in a single forward pass from noise.

Classifier-free guidance. Many text-conditioned diffusion models are trained with a classifier-free guidance setup, where the conditioning $\mathbf{c}$ is randomly dropped during training so that a single network learns both conditional and unconditional predictions. At sampling time, the model is evaluated in both modes and its outputs are combined using a scalar guidance weight $\omega \geq 0$. Using our denoiser notation, the guided noise prediction is:

$$\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\phi}}(\mathbf{z}_t, t, \mathbf{c}; \omega) = (1+\omega)\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\phi}}(\mathbf{z}_t, t, \mathbf{c}) - \omega\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\phi}}(\mathbf{z}_t, t, \mathbf{c} = \varnothing) \tag{18}$$

where $\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\phi}}(\mathbf{z}_t, t, \mathbf{c})$ and $\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\phi}}(\mathbf{z}_t, t, \varnothing)$ denote the conditional and unconditional outputs of the same network. The guided prediction $\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\phi}}$ is used in the sampler to update $\mathbf{z}_{t_k}$ and appears in the SDS gradients in Sec. 3.3.1 and Sec. 5.1.

B.2 Reducing Estimator Variance

B.2.1 Importance Sampling Theory and Application to Diffusion

We expand the importance-sampling treatment of Sec. 3.2.1.

Setup. Let $t \in [0, 1]$ be a noise level with base density $p(t)$. Let $\boldsymbol{\xi}$ denote all other randomness (e.g., sampled input data, Gaussian noise) drawn from a conditional distribution $p(\boldsymbol{\xi} \mid t)$. For a

vector-valued contribution $\mathbf{f}(t, \boldsymbol{\xi})$ such as a training gradient, define the conditional mean integrand:

$$\mathbf{F}(t) = \mathbb{E}[\mathbf{f}(t, \boldsymbol{\xi}) \mid t] \tag{19}$$

and its mean:

$$\boldsymbol{\mu} = \mathbb{E}_{t\sim p, \boldsymbol{\xi}\sim p(\cdot|t)}[\mathbf{f}(t, \boldsymbol{\xi})] = \mathbb{E}_{t\sim p}[\mathbf{F}(t)] \tag{20}$$

Importance sampling estimator. For any proposal density $q(t)$ with $q(t) > 0$ whenever $p(t) > 0$, define the importance weight $\tilde{w}(t) = \frac{p(t)}{q(t)}$ and sample $t^{(n)} \sim q$, $\boldsymbol{\xi}^{(n)} \sim p(\cdot \mid t^{(n)})$. Then the following is an unbiased estimator for $\boldsymbol{\mu}$:

$$\hat{\boldsymbol{\mu}}_q = \frac{1}{N} \sum_{n=1}^{N} \tilde{w}(t^{(n)}) \mathbf{f}(t^{(n)}, \boldsymbol{\xi}^{(n)}) \tag{21}$$

Variance and optimal proposals. A direct calculation gives the trace-covariance dispersion

$$\operatorname{tr}(\operatorname{Cov}(\hat{\boldsymbol{\mu}}_q)) = \frac{1}{N} \left(\int \frac{p(t)^2}{q(t)} \mathbb{E}[\|\mathbf{f}(t, \boldsymbol{\xi})\|_2^2 \mid t] \mathrm{d}t - \|\boldsymbol{\mu}\|_2^2 \right) \tag{22}$$

which implies the variance-minimizing proposal under this criterion

$$q^\star(t) \propto p(t) \sqrt{\mathbb{E}[\|\mathbf{f}(t, \boldsymbol{\xi})\|_2^2 \mid t]} \tag{23}$$

$$= p(t) \sqrt{\|\mathbf{F}(t)\|_2^2 + \operatorname{tr}(\operatorname{Cov}(\mathbf{f}(t, \boldsymbol{\xi}) \mid t))}; \tag{24}$$

see standard treatments of optimal importance sampling [76]. If $\mathbf{f}(t, \boldsymbol{\xi})$ is deterministic given t, then $q^\star(t) \propto p(t)\|\mathbf{F}(t)\|_2$. For a scalar integrand, this reduces to the familiar form $q^\star(t) \propto p(t)|\mathbf{F}(t)|$. For vector-valued $\mathbf{F}$, even the oracle proposal typically does not yield zero variance because the contribution direction can vary with t. Intuitively, importance sampling reallocates samples toward noise levels with large root-mean-square contributions and away from those with small ones.

Loss-based proxies and their limitations. Evaluating $\|\mathbf{F}(t)\|_2$ in diffusion is prohibitively expensive. Nichol and Dhariwal [63] instead set timestep probabilities from per-timestep loss statistics, and Zheng et al. [109] use a fixed non-uniform schedule derived from an optimal-transport analysis.

Concretely, consider the per-timestep denoising loss from the weighted diffusion objective in Eq. 16. For diffusion model training, the gradient with respect to denoiser parameters $\boldsymbol{\phi}$ is

$$\mathbf{f}(t, \boldsymbol{\xi}) := \nabla_{\boldsymbol{\phi}} \ell = 2\mathbf{J}_{\boldsymbol{\phi}}^\top \mathbf{r} \text{ where } \mathbf{J}_{\boldsymbol{\phi}} = \nabla_{\boldsymbol{\phi}} \hat{\boldsymbol{\epsilon}}_{\boldsymbol{\phi}}(\mathbf{z}_t, t, \mathbf{c}) \tag{25}$$

Recall that the optimal importance sampling proposal from Eq. 23 requires $\sqrt{\mathbb{E}[\|\mathbf{f}(t, \boldsymbol{\xi})\|_2^2 \mid t]}$, which equals:

$$\sqrt{\mathbb{E}[\|\nabla_{\boldsymbol{\phi}} \ell\|_2^2 \mid t]} = \sqrt{\|\mathbb{E}[\nabla_{\boldsymbol{\phi}} \ell \mid t]\|_2^2 + \operatorname{tr}(\operatorname{Cov}(\nabla_{\boldsymbol{\phi}} \ell \mid t))} \tag{26}$$

Since estimating this is expensive, loss statistics serve as a cheap proxy, such as $\sqrt{\mathbb{E}[\ell \mid t]} = \sqrt{\mathbb{E}[\|\mathbf{r}\|_2^2 \mid t]}$; Nichol and Dhariwal [63] use the root mean square of the per-timestep loss. However, for any single sample

$$\|\nabla_{\boldsymbol{\phi}} \ell\|_2^2 = 4\|\mathbf{J}_{\boldsymbol{\phi}}^\top \mathbf{r}\|_2^2 \le 4\|\mathbf{J}_{\boldsymbol{\phi}}\|_2^2 \|\mathbf{r}\|_2^2 = 4\|\mathbf{J}_{\boldsymbol{\phi}}\|_2^2 \ell \tag{27}$$

Equality holds when $\mathbf{r}$ aligns with the leading left singular vector of $\mathbf{J}_{\boldsymbol{\phi}}$. The loss proxy ℓ captures only $\|\mathbf{r}\|_2^2$ and ignores $\|\mathbf{J}_{\boldsymbol{\phi}}\|_2^2$, so a loss-derived schedule can misrank timesteps when $\|\mathbf{J}_{\boldsymbol{\phi}}\|_2$ varies with t or correlates weakly with $\|\mathbf{r}\|_2$.

The gap in SDS-style optimization. This gap is amplified in the settings we focus on, where the gradient of interest is not with respect to denoiser parameters. For example, in SDS-style optimization (detailed in Sec. 3.3.1), we optimize parameters $\boldsymbol{\theta}$ of a differentiable generator using a frozen diffusion teacher. The per-timestep update takes the form:

$$\mathbf{f}(t, \boldsymbol{\xi}) = w_{\text{SDS}}(t) \mathbf{J}_{\boldsymbol{\theta}}^\top \mathbf{r} \text{ where} \tag{28}$$

$$\mathbf{J}_{\boldsymbol{\theta}} = \frac{\partial \mathbf{z}}{\partial \boldsymbol{\theta}} = \frac{\partial \text{Encode}}{\partial \mathbf{x}} \frac{\partial g'}{\partial \boldsymbol{\theta}} \tag{29}$$

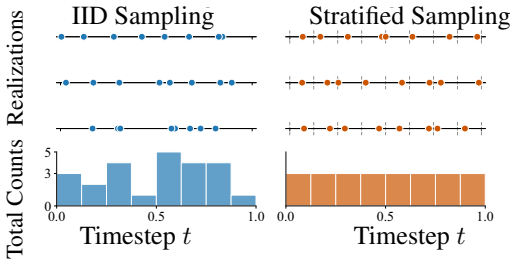


Figure 8: Stratified Sampling Visualization: We show 3 realizations/batches of 8 timestep samples for both IID and stratified sampling. For stratified sampling, we create a bin for each sample and require each batch to contain one sample from each bin. This often gives lower-variance estimators.

for the known scalar weight $w_{\text{SDS}}(t)$ of Eq. 6. Here $g'(\boldsymbol{\theta})$ is the generator output (e.g., a rendered image) and Encode maps it to latent space; SDS omits the teacher input Jacobian $\nabla_{\mathbf{z}}\hat{\boldsymbol{\epsilon}}_{\phi}$ from this product. The effective Jacobian $\mathbf{J}_{\boldsymbol{\theta}}$ thus includes a potentially ill-conditioned encoder-generator chain whose timestep dependence can differ from $\|\mathbf{r}\|_2$. Combined with $w(t)$, this leaves room for proposals that target gradient contributions rather than the loss proxy.

Stratified-IS unbiasedness (Eq. 14). We verify that the construction in Sec. 4.1.3 yields an unbiased estimator of $\mathbb{E}_{t\sim p}[\mathbf{F}(t)]$. Let F_q be the CDF of the importance proposal q and partition $[0,1]$ into B equal-mass strata in q-quantile space, $\mathcal{S}_b = \{t : F_q(t) \in [(b{-}1)/B, b/B]\}$, so that $\Pr_{t\sim q}[t \in \mathcal{S}_b] = 1/B$ and the conditional density is $q(t \mid \mathcal{S}_b) = B\,q(t)\mathbf{1}[t \in \mathcal{S}_b]$. Drawing $\boldsymbol{\xi}_b \sim \mathcal{U}(0,1)$ and setting $t_b = F_q^{-1}((b{-}1+\boldsymbol{\xi}_b)/B)$ gives t_b distributed according to $q(\cdot \mid \mathcal{S}_b)$. With importance weight $\tilde{w}(t) = p(t)/q(t)$ and contribution $\mathbf{f}(t_b, \boldsymbol{\xi}'_b)$ for $\boldsymbol{\xi}'_b \sim p(\cdot \mid t_b)$, the per-stratum expectation is

$$\mathbb{E}\big[\tilde{w}(t_b)\,\mathbf{f}(t_b,\boldsymbol{\xi}'_b)\big] = \int_{\mathcal{S}_b} \frac{p(t)}{q(t)}\,\mathbf{F}(t)\,B\,q(t)\mathrm{d}t = B\int_{\mathcal{S}_b} p(t)\,\mathbf{F}(t)\mathrm{d}t. \tag{30}$$

Averaging over the B strata,

$$\mathbb{E}\Big[\tfrac{1}{B}\sum_{b=1}^{B}\tilde{w}(t_b)\,\mathbf{f}(t_b,\boldsymbol{\xi}'_b)\Big] = \sum_{b=1}^{B}\int_{\mathcal{S}_b} p(t)\,\mathbf{F}(t)\mathrm{d}t = \int_0^1 p(t)\,\mathbf{F}(t)\mathrm{d}t = \mathbb{E}_{t\sim p}[\mathbf{F}(t)], \tag{31}$$

so Eq. 14 is unbiased for $\mathbb{E}_{t\sim p}[\mathbf{F}(t)]$ for any proposal q that satisfies $q(t){>}0$ wherever $p(t){>}0$. The variance-reduction argument follows from the standard stratified-sampling decomposition: the variance of the per-stratum-averaged estimator equals $\frac{1}{B}$ times the average within-stratum conditional variance (dropping the between-stratum component carried by simple Monte Carlo; Thompson [91]), composed with the usual importance-reweighting variance formula.

Fig. 8 visualizes the stratified sampling of Sec. 3.2.2.

B.2.2 Two-Stage Variance Decomposition and Cost-Optimal Allocation

Sec. 4.1 derives Eq. 10 from the law of total variance, and Sec. 4.1.1 substitutes the budget to obtain Eq. 11 and its minimizer $K^\star$. Given $\mathbf{x}$, $\hat{\mathbf{h}}^{(r)}$ has mean $\mathbb{E}[\mathbf{h} \mid \mathbf{x}]$ and variance $\mathrm{Var}(\mathbf{h} \mid \mathbf{x})/K$, so $\mathrm{Var}(\hat{\mathbf{h}}^{(r)}) = \mathrm{Var}_{\mathbf{x}}(\mathbb{E}[\mathbf{h} \mid \mathbf{x}]) + \mathbb{E}_{\mathbf{x}}[\mathrm{Var}(\mathbf{h} \mid \mathbf{x})]/K = \sigma_A^2 + \sigma_W^2/K$. The estimator is unbiased, $\mathbb{E}[\hat{\nabla}] = \mathbb{E}_{\mathbf{x}}\mathbb{E}[\mathbf{h} \mid \mathbf{x}]$, because each subunit draw has the target law given $\mathbf{x}$; independence across subunits is what gives the σ_W^2/K term. Eq. 11 is convex in $K > 0$, since its second derivative $2\sigma_W^2 c_{\text{render+encode}}/(\mathcal{B}K^3)$ is nonnegative (positive when $\sigma_W^2 c_{\text{render+encode}} > 0$), and minimizing cost at fixed variance gives the same $K^\star$ [10]. As $c_{\text{render+encode}} \to 0$, $K^\star \to 0$ and $K = 1$ is best, and $K^\star$ grows as $\sqrt{c_{\text{render+encode}}/c_{\text{denoise}}}$. These statements assume IID subunits: under per-render stratification the within-unit term is at most σ_W^2/K and usually falls faster, which shifts the optimum toward larger K, although $K = 1$ remains best for every method in our zero-render-cost sweep (App. Sec. D.1.7).

Importance sampling and stratification of the subunits change only σ_W^2. With proposal q and weight p/q, it becomes the reweighted within-unit variance (App. Sec. B.2.1, applied conditionally on $\mathbf{x}$); with one draw from each of K equal-mass strata of q, it becomes the average within-stratum variance. The IID structure across r, and so the form of Eq. 10, is unchanged, which is why the gains compose.

B.2.3 Diffusion Model Noise Schedules

We provide additional details on the connection between noise schedules and importance sampling in diffusion training, expanding on Sec. 3.2.1.

Following Kingma et al. [35], we view the noise schedule as a differentiable, strictly decreasing function

$$\lambda = f_\lambda(t), \qquad t \in [0, 1] \tag{32}$$

that maps continuous time to log signal-to-noise ratio $\lambda = \log(\alpha_t^2/\sigma_t^2)$. Monotonicity ensures invertibility, so there is a bijection between time and logSNR.

Induced distribution over noise levels. When sampling time uniformly $t \sim \mathcal{U}(0, 1)$ and evaluating $\lambda = f_\lambda(t)$, the change-of-variables formula gives a distribution over noise levels

$$p(\lambda) = \left| \frac{\mathrm{d}\lambda}{\mathrm{d}t} \right|^{-1} \tag{33}$$

where the absolute value accounts for the fact that λ decreases with t. Different schedules (linear, cosine, learned) induce different distributions $p(\lambda)$ even though all sample t uniformly.

Noise schedules as importance sampling. If the timestep weight carries the schedule's Jacobian, $w(t) = \frac{1}{2} w(\lambda) \, |\mathrm{d}\lambda/\mathrm{d}t|$ with $\lambda = f_\lambda(t)$ (the form of Kingma et al. [35]), changing variables from t to λ in the weighted objective from Eq. 16 gives

$$\mathcal{L}_{\text{wDiff}}(\boldsymbol{\phi}) = \tfrac{1}{2} \int_{\lambda_{\min}}^{\lambda_{\max}} w(\lambda) \mathbb{E}_{\boldsymbol{\epsilon}} \left[\|\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\phi}}(\mathbf{z}_\lambda, \lambda, \mathbf{c}) - \boldsymbol{\epsilon}\|_2^2 \right] \mathrm{d}\lambda \tag{34}$$

where $\lambda_{\min} = f_\lambda(1)$ and $\lambda_{\max} = f_\lambda(0)$ are the schedule endpoints and $\mathbf{z}_\lambda$ denotes the encoded data noised to level λ. This integral does not depend on the schedule f_λ except through the endpoints.

Equivalently, we can write the objective as an expectation over the induced distribution $p(\lambda)$:

$$\mathcal{L}_{\text{wDiff}}(\boldsymbol{\phi}) = \tfrac{1}{2} \mathbb{E}_{\lambda \sim p(\lambda), \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \left[\frac{w(\lambda)}{p(\lambda)} \|\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\phi}}(\mathbf{z}_\lambda, \lambda, \mathbf{c}) - \boldsymbol{\epsilon}\|_2^2 \right] \tag{35}$$

The noise schedule thus induces an importance distribution $p(\lambda)$ over noise levels. $\mathcal{L}_{\text{wDiff}}$ is invariant to the schedule (up to endpoints), but estimator variance depends on how $p(\lambda)$ aligns with the integrand: schedule design for diffusion-model training is an importance-sampling problem.

Schedules versus weights in downstream applications. In SDS and DMD (Secs. 3.3.1, 3.3.2), practitioners inherit a fixed teacher schedule and apply an additional weight $w(t)$ or $w_{\text{SDS}}(t)$ that conflates the intrinsic loss weight $w(\lambda)$ with the schedule-induced density $p(\lambda)$. The SDS weight $w_{\text{SDS}}(t)$ typically bundles modeling choices (e.g., α_t) with implicit timestep reweighting; if monotonic, it induces a new effective timestep distribution. DMD's weighting involves data-dependent normalization (e.g., $\|\boldsymbol{\mu}_{\text{base}}(\mathbf{z}_t, t) - \mathbf{z}\|$) and is non-monotonic, so it does not map to a simple schedule. Sec. 4.1.2 treats these weights as known functions, building proposals $q(t)$ via likelihood ratios to avoid conflating schedule and reweighting.

B.3 Diffusion Model Applications

B.3.1 Diffusion Priors for Optimization

Score Distillation Sampling (SDS) uses a pretrained diffusion model over an observation space (images, videos, audio, or latents) as a frozen conditional prior that supplies gradients to a parametrized generator, renderer, or simulator. Given parameters $\boldsymbol{\theta}$ and a sampled rendering condition $\boldsymbol{\pi}$ (for example, camera pose in text-to-3D), we render an observation

$$\mathbf{z} = g(\boldsymbol{\theta}, \boldsymbol{\pi}) = \text{Encode}\big(g'(\boldsymbol{\theta}, \boldsymbol{\pi})\big), \tag{36}$$

where g' is the differentiable renderer (or generator or simulator) and Encode maps its output into the teacher's observation space (often latent). We update $\boldsymbol{\theta}$ so $\mathbf{z}$ lies in high-density regions of $p_{\boldsymbol{\phi}}(\cdot \mid \mathbf{c})$ (with optional guidance), giving the chain rule

$$\nabla_{\boldsymbol{\theta}} \, \mathbb{E}_{\boldsymbol{\pi}}[\log p_{\boldsymbol{\phi}}(g(\boldsymbol{\theta}, \boldsymbol{\pi}) \mid \mathbf{c})] = \mathbb{E}_{\boldsymbol{\pi}} \left[\frac{\mathrm{d} \log p_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{c})}{\mathrm{d}\mathbf{z}} \frac{\mathrm{d}\mathbf{z}}{\mathrm{d}\boldsymbol{\theta}} \right]. \tag{37}$$

Diffusion teacher and the SDS residual. Write the forward noising process (in the teacher's observation space) as

$$\mathbf{z}_t = \mathbf{z}_t(\mathbf{z}, t, \boldsymbol{\epsilon}) = \alpha_t \, \mathbf{z} + \sigma_t \, \boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I}), \qquad t \sim p(t), \tag{38}$$

where α_t, σ_t are the usual diffusion coefficients (the exact parameterization is model-dependent). Let the teacher predict noise (optionally with classifier-free guidance scale ω):

$$\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\phi}} = \hat{\boldsymbol{\epsilon}}_{\boldsymbol{\phi}}(\mathbf{z}_t, t, \mathbf{c}; \omega). \tag{39}$$

We define the per-sample denoising residual

$$\mathbf{r}_{\boldsymbol{\phi}}(\boldsymbol{\theta}, \boldsymbol{\pi}, t, \boldsymbol{\epsilon}, \mathbf{c}, \omega) := \hat{\boldsymbol{\epsilon}}_{\boldsymbol{\phi}}(\mathbf{z}_t, t, \mathbf{c}; \omega) - \boldsymbol{\epsilon}, \tag{40}$$

where $\mathbf{z}_t$ is understood to be $\mathbf{z}_t(\mathbf{z}(\boldsymbol{\theta}, \boldsymbol{\pi}), t, \boldsymbol{\epsilon})$.

SDS gradient estimator (with stop-gradient through the teacher). SDS uses a simple surrogate for the score term $\mathrm{d} \log p_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{c})/\mathrm{d}\mathbf{z}$ by differentiating through the noising map but *not* through the teacher prediction. Concretely, in the backward pass we treat $\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\phi}}(\cdot)$ as a constant with respect to $\mathbf{z}_t$ (equivalently, we drop the Jacobian $\mathrm{d}\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\phi}}/\mathrm{d}\mathbf{z}_t$). This yields the estimator

$$\mathbf{u}_{\mathrm{SDS}}(\boldsymbol{\theta}) = \mathop{\mathbb{E}}_{\boldsymbol{\pi}}\left[\mathop{\mathbb{E}}_{t\sim p(t),\, \boldsymbol{\epsilon}\sim\mathcal{N}(0,I)}\left[w(t)\ \mathrm{sg}\big(\mathbf{r}_{\boldsymbol{\phi}}\big)\ \frac{\mathrm{d}\mathbf{z}_t}{\mathrm{d}\mathbf{z}}\right]\frac{\mathrm{d}\mathbf{z}}{\mathrm{d}\boldsymbol{\theta}}\right]. \tag{41}$$

Here $w(t)$ is the timestep weight of the SDS loss (Sec. 3.3.1). Under Eq. 38, $\mathrm{d}\mathbf{z}_t/\mathrm{d}\mathbf{z} = \alpha_t \mathbf{I}$, so with $w_{\mathrm{SDS}}(t) = w(t)\alpha_t$ this is Eq. 6, the common implementation pattern where the timestep weight includes the α_t factor.

Equivalent surrogate MSE form used in code. The same update can be obtained as the gradient of a mean-squared error objective with a stop-gradient target. Define the (detached) per-sample gradient direction in observation space

$$\widehat{g}_{\mathbf{z}} := w_{\mathrm{SDS}}(t)\ \mathrm{sg}\big(\mathbf{r}_{\boldsymbol{\phi}}\big) \tag{42}$$

and set the target as

$$\mathbf{z}_{\mathrm{tgt}} := \mathrm{sg}(\mathbf{z} - \widehat{g}_{\mathbf{z}})\,. \tag{43}$$

Then the surrogate loss

$$\mathcal{L}_{\mathrm{SDS}}(\boldsymbol{\theta}) := \tfrac{1}{2}\mathop{\mathbb{E}}_{\boldsymbol{\pi},t,\boldsymbol{\epsilon}}\left[\|\mathbf{z}(\boldsymbol{\theta}, \boldsymbol{\pi}) - \mathbf{z}_{\mathrm{tgt}}\|_2^2\right] \tag{44}$$

has gradient

$$\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\mathrm{SDS}}(\boldsymbol{\theta}) = \mathop{\mathbb{E}}_{\boldsymbol{\pi},t,\boldsymbol{\epsilon}}\left[\widehat{g}_{\mathbf{z}}\,\frac{\mathrm{d}\mathbf{z}}{\mathrm{d}\boldsymbol{\theta}}\right], \tag{45}$$

which matches Eq. 6. This is the form we implement: we compute $\widehat{g}_{\mathbf{z}}$ using the frozen teacher (with stop-gradient through $\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\phi}}$ and forward noising), form the detached target Eq. 43, and optimize the MSE Eq. 44.

B.3.2 Single-Step Diffusion Distillation

Objective and score-based formulation. Distribution Matching Distillation (DMD) [105] distills a pretrained multi-step diffusion teacher into a one-step generator $G_{\boldsymbol{\theta}} : \mathbb{R}^d \to \mathbb{R}^d$ parameterized by $\boldsymbol{\theta}$. Given $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, the generator produces $\mathbf{z} = G_{\boldsymbol{\theta}}(\boldsymbol{\epsilon})$ and induces a distribution $p_{\mathrm{fake}}(\mathbf{z})$. The goal is to match p_{fake} to the real data distribution p_{real} by minimizing the reverse KL divergence:

$$D_{\mathrm{KL}}(p_{\mathrm{fake}} \| p_{\mathrm{real}}) = \mathbb{E}_{\mathbf{z}\sim p_{\mathrm{fake}}}[\log p_{\mathrm{fake}}(\mathbf{z}) - \log p_{\mathrm{real}}(\mathbf{z})] \tag{46}$$

Taking the gradient with respect to $\boldsymbol{\theta}$ and applying the chain rule gives:

$$\nabla_{\boldsymbol{\theta}} D_{\mathrm{KL}}(p_{\mathrm{fake}} \| p_{\mathrm{real}}) = \mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}[(\mathbf{s}_{\mathrm{fake}}(\mathbf{z}) - \mathbf{s}_{\mathrm{real}}(\mathbf{z}))\tfrac{\partial G_{\boldsymbol{\theta}}(\boldsymbol{\epsilon})}{\partial \boldsymbol{\theta}}] \tag{47}$$

where $\mathbf{z} = G_{\boldsymbol{\theta}}(\boldsymbol{\epsilon})$ and the score functions are $\mathbf{s}_{\mathrm{real}}(\mathbf{z}) = \nabla_{\mathbf{z}} \log p_{\mathrm{real}}(\mathbf{z})$ and $\mathbf{s}_{\mathrm{fake}}(\mathbf{z}) = \nabla_{\mathbf{z}} \log p_{\mathrm{fake}}(\mathbf{z})$.

Score estimation via diffusion noising. Direct score evaluation is intractable and unstable when p_{fake} and p_{real} have disjoint support. DMD estimates scores by perturbing samples with forward diffusion noise and denoising them with diffusion models. For a clean sample $\mathbf{z}$ and timestep t, form the noised sample:

$$\mathbf{z}_t = \alpha_t \mathbf{z} + \sigma_t \boldsymbol{\epsilon}', \qquad \boldsymbol{\epsilon}' \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \tag{48}$$

where α_t, σ_t are the same diffusion schedule coefficients used in Sec. 3.1. The score of the noised distribution can be related to a denoising mean predictor. If $\boldsymbol{\mu}(\mathbf{z}_t, t)$ predicts $\mathbb{E}[\mathbf{z} \mid \mathbf{z}_t, t]$, then:

$$\nabla_{\mathbf{z}_t} \log p(\mathbf{z}_t \mid t) = -\frac{\mathbf{z}_t - \alpha_t \boldsymbol{\mu}(\mathbf{z}_t, t)}{\sigma_t^2} \tag{49}$$

DMD uses two mean predictors:

- $\boldsymbol{\mu}_{\text{base}}(\mathbf{z}_t, t)$: the frozen pretrained teacher, which estimates $\mathbb{E}[\mathbf{z} \mid \mathbf{z}_t, t]$ under p_{real}
- $\boldsymbol{\mu}^{\boldsymbol{\phi}}_{\text{fake}}(\mathbf{z}_t, t)$: a learned model parameterized by $\boldsymbol{\phi}$, which estimates $\mathbb{E}[\mathbf{z} \mid \mathbf{z}_t, t]$ under p_{fake}

The approximate scores are:

$$\begin{aligned} \mathbf{s}_{\text{real}}(\mathbf{z}_t, t) &= -\frac{\mathbf{z}_t - \alpha_t \boldsymbol{\mu}_{\text{base}}(\mathbf{z}_t, t)}{\sigma_t^2} \\ \mathbf{s}_{\text{fake}}(\mathbf{z}_t, t) &= -\frac{\mathbf{z}_t - \alpha_t \boldsymbol{\mu}^{\boldsymbol{\phi}}_{\text{fake}}(\mathbf{z}_t, t)}{\sigma_t^2} \end{aligned} \tag{50}$$

Noising ensures that both distributions have overlapping support in $\mathbf{z}_t$-space, stabilizing training.

Practical gradient estimator. Substituting the noised-score approximations and integrating over timesteps gives the DMD generator gradient:

$$\nabla_{\boldsymbol{\theta}} D_{\text{KL}} \simeq \mathop{\mathbb{E}}_{\substack{\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I}) \\ t\sim p(t) \\ \boldsymbol{\epsilon}'\sim\mathcal{N}(\mathbf{0},\mathbf{I})}} \left[w(t)\alpha_t \left(\mathbf{s}_{\text{fake}}(\mathbf{z}_t, t) - \mathbf{s}_{\text{real}}(\mathbf{z}_t, t)\right) \times \frac{\partial G_{\boldsymbol{\theta}}(\boldsymbol{\epsilon})}{\partial \boldsymbol{\theta}} \right] \tag{51}$$

where $\mathbf{z} = G_{\boldsymbol{\theta}}(\boldsymbol{\epsilon})$, $\mathbf{z}_t = \alpha_t \mathbf{z} + \sigma_t \boldsymbol{\epsilon}'$, and $w(t)$ is a weighting function. A common choice is:

$$w(t) = \frac{\sigma_t^2}{\alpha_t} \frac{CS}{\|\boldsymbol{\mu}_{\text{base}}(\mathbf{z}_t, t) - \mathbf{z}\|_1} \tag{52}$$

where C and S are the numbers of channels and spatial locations, respectively. This weight normalizes scale variations across timesteps to stabilize training. Intuitively, $\mathbf{s}_{\text{real}}$ pulls generator samples toward the data manifold, while $-\mathbf{s}_{\text{fake}}$ discourages mode collapse by repelling samples from regions of excessive fake density.

Auxiliary losses. To track the evolving fake distribution during training, DMD updates $\boldsymbol{\mu}^{\boldsymbol{\phi}}_{\text{fake}}$ online using a standard diffusion denoising loss on stop-gradient generator outputs:

$$\mathcal{L}_{\text{denoise}}(\boldsymbol{\phi}) = \mathop{\mathbb{E}}_{\boldsymbol{\epsilon}, t, \boldsymbol{\epsilon}'} \left[\|\boldsymbol{\mu}^{\boldsymbol{\phi}}_{\text{fake}}(\mathbf{z}_t, t) - \text{sg}(\mathbf{z})\|_2^2 \right] \tag{53}$$

where $\mathbf{z} = G_{\boldsymbol{\theta}}(\boldsymbol{\epsilon})$ and $\mathbf{z}_t = \alpha_t \mathbf{z} + \sigma_t \boldsymbol{\epsilon}'$.

Additionally, an optional regression loss aligns the one-step generator with deterministic samples from the teacher on a small paired dataset $\mathcal{D} = \{(\mathbf{x}, \mathbf{y})\}$:

$$\mathcal{L}_{\text{reg}}(\boldsymbol{\theta}) = \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}} \left[\ell(G_{\boldsymbol{\theta}}(\mathbf{x}), \mathbf{y}) \right] \tag{54}$$

where ℓ is a perceptual distance such as LPIPS. As in Yin et al. [104], we do not use this loss in our experiments. Yin et al. [104] also introduced a discriminator trained on the fake model's features to distinguish real data from generator outputs. We also utilize this objective in our experiments. The generator is trained with the combined objective while the fake model minimizes $\mathcal{L}_{\text{denoise}}(\boldsymbol{\phi})$ and remains detached in the generator gradient.

Classifier-free guidance. For conditional generation with text conditioning $\mathbf{c}$ and classifier-free guidance scale ω, the same construction applies. The real score uses the guided teacher prediction:

$$\boldsymbol{\mu}_{\text{base}}(\mathbf{z}_t, t, \mathbf{c}; \omega) = (1+\omega)\boldsymbol{\mu}_{\text{base}}(\mathbf{z}_t, t, \mathbf{c}) - \omega \boldsymbol{\mu}_{\text{base}}(\mathbf{z}_t, t, \varnothing) \tag{55}$$

while the fake score is unchanged. The generator trains at a fixed guidance scale to match the guided teacher distribution.

Connection to variance reduction. Like SDS, the DMD gradient is an expectation over generator input $\boldsymbol{\epsilon}$, timestep t, and forward noise $\boldsymbol{\epsilon}'$, which we estimate by Monte Carlo. Each gradient sample requires:

1. Generating $\mathbf{z} = G_{\boldsymbol{\theta}}(\boldsymbol{\epsilon})$ (potentially expensive)

2. Forward noising to form $\mathbf{z}_t$ (cheap)
3. Evaluating both $\boldsymbol{\mu}_{\text{base}}$ and $\boldsymbol{\mu}^{\boldsymbol{\phi}}_{\text{fake}}$ (moderate cost)
4. Backpropagating through $G_{\boldsymbol{\theta}}$ (expensive)

Since step (1) is independent of $(t, \boldsymbol{\epsilon}')$, the amortized resampling strategy from Sec. 4.1.1 can cache $\mathbf{z} = G_{\boldsymbol{\theta}}(\boldsymbol{\epsilon})$ and resample $(t, \boldsymbol{\epsilon}')$ multiple times per generator forward pass. Similarly, timestep stratification (Sec. 4.1.3) and importance sampling (Sec. 4.1.2) reduce variance over t. Unlike SDS, generator input variability often dominates in DMD, so allocating budget to more independent samples $\boldsymbol{\epsilon}$ (rather than many re-noisings per sample) can be more effective, as discussed in Sec. 5.2.

B.3.3 Data Attribution for Video Generation

Data attribution quantifies how training examples contribute to a model's outputs. In generative modeling, it identifies influential or harmful fine-tuning clips for data selection, debugging, and efficient specialization.

Influence functions and scalable approximations. Let $\mathcal{L}(\boldsymbol{\phi}; \mathbf{x}, \mathbf{c})$ be a per-example training loss with query $(\mathbf{x}_{\text{query}}, \mathbf{c}_{\text{query}})$. For training example $(\mathbf{x}_n, \mathbf{c}_n)$, infinitesimally upweighting it changes the query loss (under regularity) as in [36]:

$$I((\mathbf{x}_n, \mathbf{c}_n), (\mathbf{x}_{\text{query}}, \mathbf{c}_{\text{query}})) = -\nabla_{\boldsymbol{\phi}} \mathcal{L}(\boldsymbol{\phi}; \mathbf{x}_{\text{query}}, \mathbf{c}_{\text{query}})^\top \mathbf{H}(\boldsymbol{\phi})^{-1} \nabla_{\boldsymbol{\phi}} \mathcal{L}(\boldsymbol{\phi}; \mathbf{x}_n, \mathbf{c}_n) \tag{56}$$

where $\mathbf{H}(\boldsymbol{\phi}) = \nabla^2_{\boldsymbol{\phi}} \frac{1}{|\mathcal{D}|} \sum_{(\mathbf{x},\mathbf{c})\in\mathcal{D}} \mathcal{L}(\boldsymbol{\phi}; \mathbf{x}, \mathbf{c})$. Since applying $\mathbf{H}(\boldsymbol{\phi})^{-1}$ is infeasible at modern scales, practical methods approximate influence using gradient similarity computed across checkpoints (TracIn) or via projected gradient features (TRAK) [70, 66].

Diffusion attribution as gradient similarity over $(t, \boldsymbol{\epsilon})$. For diffusion/flow-matching teachers, training losses and gradients are expectations over noise levels and Gaussian noise. Using notation from Sec. 3.1, define the per-example, per-draw diffusion gradient:

$$\mathbf{g}(\boldsymbol{\phi}; \mathbf{x}, \mathbf{c}, t, \boldsymbol{\epsilon}) = \nabla_{\boldsymbol{\phi}} \ell_{\text{Diff}}(\text{Encode}(\mathbf{x}), \mathbf{c}, t, \boldsymbol{\epsilon}, \boldsymbol{\phi}) \tag{57}$$

and let $\mathcal{T}$ denote a multiset of $(t, \boldsymbol{\epsilon})$ shared across query and training. Following the gradient normalization of [102], we score attribution by the cosine similarity of normalized gradients, averaged over $\mathcal{T}$

$$I_{\text{diff}}(n, \text{query}) = \frac{1}{|\mathcal{T}|} \sum_{(t,\boldsymbol{\epsilon})\in\mathcal{T}} \frac{\mathbf{g}(\boldsymbol{\phi}; \mathbf{x}_{\text{query}}, \mathbf{c}_{\text{query}}, t, \boldsymbol{\epsilon})}{\|\mathbf{g}(\boldsymbol{\phi}; \mathbf{x}_{\text{query}}, \mathbf{c}_{\text{query}}, t, \boldsymbol{\epsilon})\|_2}^\top \frac{\mathbf{g}(\boldsymbol{\phi}; \mathbf{x}_n, \mathbf{c}_n, t, \boldsymbol{\epsilon})}{\|\mathbf{g}(\boldsymbol{\phi}; \mathbf{x}_n, \mathbf{c}_n, t, \boldsymbol{\epsilon})\|_2} \tag{58}$$

where $I_{\text{diff}}(n, \text{query})$ abbreviates influence between $(\mathbf{x}_n, \mathbf{c}_n)$ and $(\mathbf{x}_{\text{query}}, \mathbf{c}_{\text{query}})$. Sharing $(t, \boldsymbol{\epsilon})$ couples the query and candidate gradients through common random numbers, and per-draw normalization mitigates scale effects. Substantial Monte Carlo variance in Eq. 58 over $(t, \boldsymbol{\epsilon})$ limits influence ranking stability at fixed compute, making stable rankings expensive.

Why video is different: appearance-motion entanglement and length effects. For video, $\mathbf{x} \in \mathbb{R}^{F\times H\times W\times 3}$, and the diffusion loss aggregates frame and spatial contributions. Two issues arise. First, whole-video gradients overemphasize static appearance (objects, backgrounds) over temporal dynamics. Second, gradient magnitudes scale with clip length F, biasing similarity and selection toward longer clips.

Motion-centric attribution via loss-space masking (MOTIVE). MOTIVE [98] introduces a motion-weighted attribution loss that emphasizes dynamic regions (e.g., using optical-flow-derived motion magnitude) while suppressing static backgrounds, and corrects dominant length scaling. It also projects gradients efficiently for scalability. Let $\mathbf{M}(\mathbf{x}) \in [0, 1]^{F\times H'\times W'}$ be a motion mask aligned with the latent grid (after any required downsampling), and let $\tilde{\ell}_{\text{Diff}}(\text{Encode}(\mathbf{x}), \mathbf{c}, t, \boldsymbol{\epsilon}, \boldsymbol{\phi}) \in \mathbb{R}^{F\times H'\times W'}$ denote a per-location squared-error form of the diffusion cost. MOTIVE defines the motion-weighted per-example cost:

$$\ell_{\text{mot}}(\boldsymbol{\phi}; \mathbf{x}, \mathbf{c}, t, \boldsymbol{\epsilon}) = \frac{1}{F} \text{mean}_{f,h,w} \left[\mathbf{M}(\mathbf{x})_{f,h,w} \tilde{\ell}_{\text{Diff}}(\text{Encode}(\mathbf{x}), \mathbf{c}, t, \boldsymbol{\epsilon}, \boldsymbol{\phi})_{f,h,w}\right] \tag{59}$$

and the corresponding motion-weighted gradient:

$$\mathbf{g}_{\text{mot}}(\boldsymbol{\phi}; \mathbf{x}, \mathbf{c}, t, \boldsymbol{\epsilon}) = \nabla_{\boldsymbol{\phi}} \ell_{\text{mot}}(\boldsymbol{\phi}; \mathbf{x}, \mathbf{c}, t, \boldsymbol{\epsilon}) \tag{60}$$

which is substituted for $\mathbf{g}$ in Eq. 58. This isolates temporal dynamics while leaving the forward noising process unchanged, since the reweighting occurs only in the attribution loss.

Connection to variance reduction. Both I_{diff} and its motion-weighted variant are Monte Carlo estimators over $(t, \boldsymbol{\epsilon})$ with expensive upstream encoding $\text{Encode}(\mathbf{x})$, a natural target for the strategies in Sec. 4: timestep IS and stratification reduce variance over t, and amortized re-noising reuses cached $\text{Encode}(\mathbf{x})$ across $(t, \boldsymbol{\epsilon})$ draws. In our experiments (Sec. 5.3), we adopt this motion-centric setup and focus on variance reduction for these estimators.

C Additional Method Details

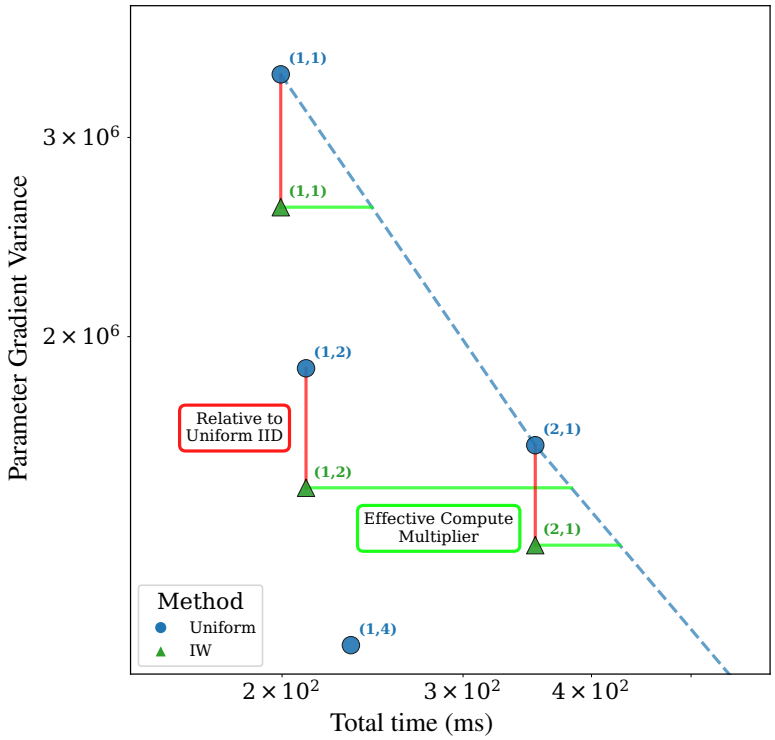


Figure 9: Geometric intuition for efficiency metrics: We visualize parameter-gradient variance versus compute cost (wall-clock time) to provide intuition into our performance metrics. The *effective compute multiplier* (ECM) compares a method to the baseline (uniform-IID with $K = 1$) at iso-variance: at a fixed variance level (horizontal line), ECM equals the ratio of baseline cost to method cost along their respective Pareto curves. The *relative efficiency* (RE) compares methods at identical (R, K), isolating the variance benefit from sampling strategies (importance weighting, stratification) independent of batch size. Higher ECM and RE indicate better efficiency. This geometric view clarifies how our variance-reduction methods achieve 2–3$\times$ compute multipliers across tasks by shifting the variance-cost frontier to the left.

C.1 Variance Measurement Framework

Implementation details for the framework are summarized in Sec. 4.2.

C.1.1 Ground-Truth Estimation and Dispersion Metrics

For an unbiased estimator $\hat{\boldsymbol{\mu}}$ of a target mean $\boldsymbol{\mu}$, the variance $\text{Var}(\hat{\boldsymbol{\mu}}) = \mathbb{E}[\|\hat{\boldsymbol{\mu}} - \boldsymbol{\mu}\|_2^2]$ equals the mean squared error. We approximate $\boldsymbol{\mu}$ using a high-sample Monte Carlo reference $\hat{\boldsymbol{\mu}}_{\text{GT}}$ formed by averaging N_{GT} independent draws (N_{GT}=20 000 in the SDS experiments; App. Sec. D.1.1). For a test estimator $\hat{\boldsymbol{\mu}}$ with N samples, we compute

$$\widehat{\text{Var}}(\hat{\boldsymbol{\mu}}) = \|\hat{\boldsymbol{\mu}} - \hat{\boldsymbol{\mu}}_{\text{GT}}\|_2^2 \tag{61}$$

and average it over independent realizations. For an independent, unbiased reference this estimates $\text{Var}(\hat{\boldsymbol{\mu}}) + \text{Var}(\hat{\boldsymbol{\mu}}_{\text{GT}})$, so the reference's own variance enters as an additive bias; we therefore use it to validate the online variance estimates (App. Sec. C.1.3, App. Sec. D.1.1) rather than as the primary measurement. We validate ground-truth quality by repeating the procedure with different random seeds and confirming agreement to within a small fraction of the standard error.

We measure variance at two stages of the SDS gradient pipeline:

1. **Latent-space update (SDS residual):** For render r, the vector $\bar{\mathbf{f}}^{(r)} = \frac{1}{K}\sum_{k=1}^{K} w_{\text{SDS}}(t^{(r,k)})\mathbf{r}^{(r,k)}$ that multiplies its renderer Jacobian. This is the cheapest to compute because it does not require backpropagation through the renderer.
2. **Parameter gradient:** The full gradient $\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\text{SDS}} = \frac{1}{R}\sum_{r=1}^{R} \bar{\mathbf{f}}^{(r)} \frac{\partial \mathbf{z}^{(r)}}{\partial \boldsymbol{\theta}}$, which includes the renderer Jacobian. This is more expensive but directly measures the quantity used for optimization.

Most prior work reports variance of the latent-space residual because it is easy to compute in a batched loop. However, the parameter gradient variance depends on the interaction between the residual and the renderer Jacobian, which can differ substantially across timesteps. We find that optimal importance-sampling proposals differ between the two metrics (see App. Fig. 22), so we report both and, when feasible, use parameter-gradient variance as our primary design criterion.

For parameter gradients, standard batched backpropagation aggregates contributions from all samples in a batch, so we cannot isolate individual sample gradients $\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\text{SDS}}^{(n)}$ without re-running backpropagation. To populate per-sample statistics, we use a batch size of 1 and loop over samples, which is expensive but necessary for accurate measurement. For DMD and data attribution, we follow a similar procedure, isolating per-sample contributions to the generator or denoiser gradients.

We also report cosine similarity between estimators and the ground truth,

$$\text{CosineSim}(\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\mu}}_{\text{GT}}) = \frac{\hat{\boldsymbol{\mu}}^{\top}\hat{\boldsymbol{\mu}}_{\text{GT}}}{\|\hat{\boldsymbol{\mu}}\|_2 \|\hat{\boldsymbol{\mu}}_{\text{GT}}\|_2} \tag{62}$$

which captures directional alignment independent of magnitude. Like the reference-based MSE, cosine similarity requires the reference to be precomputed; the online variance estimate does not.

C.1.2 Cost Metrics and Extrapolation

We measure compute cost using three complementary metrics:

- **Wall-clock time (ms):** End-to-end time for a gradient step, including parallelization and GPU scheduling effects. This is the metric we use for effective compute multipliers in the main text.
- **GPU memory (MB):** Peak memory usage, which bounds feasible batch sizes and reveals parallelization headroom.
- **Number of function evaluations (NFE):** Counts of expensive operations (renders, denoiser calls, encoder calls) independent of hardware. For SDS, NFE is typically (R, K) denoting the number of renders and re-noisings per render.

When variance scales as $\widehat{\text{Var}}(\hat{\boldsymbol{\mu}}) \approx C/N$, we fit constant C and extrapolate to larger N assuming no parallelization benefit (cost scales linearly with N). This gives the best-case variance estimates at higher compute budgets. We also measure wall-clock time at higher parallelism to capture GPU scheduling, memory bandwidth, and batching effects, which can cause the cost to plateau or increase as parallelism is reduced.

C.1.3 Caching and Computational Tricks

To reduce the cost of variance measurement, we exploit conditional independence in the estimators. For SDS, the renderer output $\mathbf{x} = g'(\boldsymbol{\theta}, \boldsymbol{\pi})$ is independent of the diffusion noise $(t, \boldsymbol{\epsilon})$, so we can cache $\mathbf{x}$ and its forward-pass encoding $\mathbf{z} = \text{Encode}(\mathbf{x})$ and reuse them across many $(t, \boldsymbol{\epsilon})$ draws. This amortizes the expensive render-and-encode step over many cheap denoiser calls, making it feasible to estimate $\hat{\boldsymbol{\mu}}_{\text{GT}}$ with thousands of samples. For parameter gradient variance, we cannot fully cache the backward pass because the render-and-encode Jacobian $\frac{\partial \mathbf{z}}{\partial \boldsymbol{\theta}}$ depends on $\mathbf{x}$, but we can still cache the forward computation and re-noise multiple times before backpropagating.

For online variance estimation, we use Welford's algorithm to update the mean and variance without storing samples; it is useful for monitoring variance trends during training, but it is not applicable to cosine similarity or any metric that requires the full ground truth.

C.1.4 Validation and Reproducibility

We validate variance estimates by repeating measurements with different random seeds and confirming consistency. For each configuration (e.g., batch size, importance proposal, stratification), we repeat measurements across prompts and seeds and report standard errors where appropriate.

We also check that ground-truth estimates $\hat{\boldsymbol{\mu}}_{\mathrm{GT}}$ from different seeds agree to within their Monte Carlo error, ensuring that N_{GT} is large enough.

C.1.5 Worked Example: ECM Computation for SDS

We illustrate how the two efficiency metrics of Sec. 4.2 combine, using the SDS values in Tables 1 and 2.

ECM factorizes into a reuse term and a sampling term. Along the uniform $(R, 1)$ baseline, variance scales as $1/R$ and cost as R, so the baseline reaches variance V at cost $c_{\mathrm{b}}(V) = C_{\mathrm{b}}/V$ for a constant C_{b}. An estimator with variance V_{m} and per-iteration cost c_{m} therefore has $\mathrm{ECM} = c_{\mathrm{b}}(V_{\mathrm{m}})/c_{\mathrm{m}} = C_{\mathrm{b}}/(V_{\mathrm{m}}c_{\mathrm{m}})$, which is also its variance reduction relative to the baseline at matched cost. Importance weighting and stratification reallocate the same draws, so they leave the per-iteration cost at a given (R, K) unchanged. The ratio of the ECMs of IW+Strat and uniform sampling at the same (R, K) is then their variance ratio, which is RE:

$$\mathrm{ECM}_{\mathrm{IW+Strat}} = \mathrm{ECM}_{\mathrm{uniform}} \cdot \mathrm{RE}_{\mathrm{IW+Strat}}.$$

Numbers. At $K = 8$, compute reuse alone gives $\mathrm{ECM}_{\mathrm{uniform}} \approx 2.63$ (Table 1) and IW+Strat gives $\mathrm{RE} \approx 1.25$ (Table 2), so $\mathrm{ECM}_{\mathrm{IW+Strat}} \approx 2.63 \times 1.25 \approx 3.29$, the $\sim 3.3\times$ headline in Sec. 5.1. The same product reproduces the other rows of Table 1 to within rounding, e.g., $2.24 \times 1.31 \approx 2.94$ at $K = 4$ and $2.52 \times 1.17 \approx 2.94$ at $K = 16$. Compute reuse thus supplies most of the multiplier, and IW+Strat a further $\sim 25\%$.

C.2 Algorithm: Combined IW + Stratified + Re-noising Estimator

Algorithm 1 compiles Eq. 8, Eq. 12, and Eq. 14 into a single drop-in pseudocode, exactly matching the SDS configuration we recommend. The DMD variant replaces the render with the generator forward pass and the residual with the weighted score difference (Eq. 51); the attribution variant differentiates the loss with respect to ϕ for each fixed example, with no generator Jacobian. Both keep the same outer loop.

Per-task instantiation. *SDS.* Per step, sample R poses $\boldsymbol{\pi}^{(r)}$, render $\mathbf{x}^{(r)} = g'(\boldsymbol{\theta}, \boldsymbol{\pi}^{(r)})$ once each, and (for latent diffusion) encode $\mathbf{z}^{(r)}$ once; for each r draw K pairs $(t, \boldsymbol{\epsilon})$ and backpropagate once per render via Eq. 8. *One-step distillation.* Treat the generator output $\mathbf{z}^{(r)} = G_{\boldsymbol{\theta}}(\boldsymbol{\epsilon}^{(r)})$ as the expensive state and apply the same re-noising pattern with fresh $(t, \boldsymbol{\epsilon}')$.

Timestep stratification details. For a uniform $p(t)$ (as in SDS, on $[t_{\min}, t_{\max}]$), equal-width bins each have probability $1/B$, and one sample per bin gives an unbiased estimator compatible with Eq. 8. For non-uniform densities, we use the quantile construction of Sec. 4.1.3 (Fig. 3). If $t \in \{0, \ldots, T-1\}$ (e.g., T=1000), we stratify in $[0, 1]$ using B equal bins with one draw per bin, then snap to the nearest index; this matches the discrete grid the teacher was trained on and moves each draw by at most half a grid step. We switch between continuous $[0, 1]$ and discrete $\{0, \ldots, T-1\}$ views for notation. With importance proposal q, we stratify in $[0, 1]$ then apply inverse-CDF sampling for stratified t.

Global stratification across renders. At $K = 1$ there is nothing to stratify within a unit, so timesteps are stratified across the whole batch instead, with bins $\mathcal{S}_b$ constructed as in Sec. 4.1.3 but with $B := R \times K$. For a uniform base density and equal-width bins, the global stratified estimator is:

$$\bar{\mathbf{f}}_{\mathrm{global}} = {}^1\!/_B \sum\nolimits_{b=1}^{B} \mathbf{f}(\mathbf{x}_b, t_b, \boldsymbol{\epsilon}_b) \tag{63}$$

where $t_b \sim \mathcal{U}(\mathcal{S}_b)$ and $\mathbf{x}_b$ is a (potentially reused) render; for a non-uniform proposal q, the bins come from its inverse CDF and each term carries the weight p/q, as in Eq. 14.

D Additional Experimental Details

D.1 Diffusion Priors for Optimization

D.1.1 Details

Model and Architecture. We use `stable-diffusion-2-1-base` [75] as the 2D diffusion prior within the threestudio framework [18]. The 3D representation is an implicit volume with a ProgressiveBandHashGrid encoder (instant-NGP style [61]), using 16 levels, 2 features per level, and $\log_2(\text{hashmap size}) = 19$. The density and color networks are VanillaMLP with 64 neurons and 1

hidden layer. Images are rendered at 256×256 resolution and bilinearly interpolated to 512×512 before being encoded by the frozen VAE encoder, yielding $4 \times 64 \times 64$ latents.

Diffusion Guidance. We use classifier-free guidance with default scale $\omega = 100$. For low guidance ablation experiments (Sec. D.1.5, Fig. 16) we use $\omega = 25$. Timesteps are sampled from the range $[t_{\min}, t_{\max}] = [20, 980]$ out of 1000 total steps, corresponding to $[\texttt{min_step_percent}, \texttt{max_step_percent}] = [0.02, 0.98]$. The noise schedule uses scaled-linear spacing in $\sqrt{\beta}$ space with $\beta_{\text{start}} = 0.00085$ and $\beta_{\text{end}} = 0.012$. With threestudio's default SDS weighting, $w_{\text{SDS}}(t) = \sigma_t^2$; the curves labelled $w(t) = \sigma_t^2$ in Fig. 1 (right) and Fig. 3 show this effective weight.

Camera Sampling. Camera poses are sampled uniformly with elevation in $[-10°, 45°]$, azimuth in $[0°, 360°]$, field of view in $[15°, 80°]$, and fixed camera distance of 2.0.

Optimization. We use the Adam optimizer [33] with $\beta_1 = 0.9$, $\beta_2 = 0.99$, $\epsilon = 10^{-15}$, and learning rates of 0.005 for geometry and 0.0001 for background. Training runs for 5000 iterations at $\omega = 100$ (longer at low guidance; App. Sec. D.1.5), saving checkpoints every 1000 steps.

Variance Evaluation Protocol. We evaluate variance reduction on a subset of the trained NeRF checkpoints, using 5 prompts at high guidance ($\omega = 100$; see Table 5) across 3 seeds $\{1, 2, 3\}$. The ground-truth gradient $\hat{\boldsymbol{\mu}}_{\text{GT}}$ is estimated by averaging 20,000 independent samples under uniform timestep sampling. It validates the online variance estimates (Variance Estimation, below) and is the reference for cosine similarity.

Timestep Sampling Strategies. We evaluate combinations of timestep distributions (uniform and importance-weighted) with batch sampling strategies (IID and stratified), described below. We additionally ablate a parameter-gradient-weighted proposal (`param_iw`) described in Sec. D.1.8.

Training Runs for 3D NeRF Checkpoints. We train NeRF models on 30 text prompts (listed in Table 5) using two configurations: (i) *Baseline*: uniform-iid timestep sampling with $(R, K) = (4, 1)$ (4 renders, 1 re-noising each); (ii) *Ours*: importance-weighted + stratified sampling with $(R, K) = (1, 16)$ (1 render, 16 re-noisings each). Ours costs less per iteration (Fig. 11). Each prompt is trained for 5000 iterations across 3 seeds $\{1, 2, 3\}$, yielding 180 final checkpoints (30 prompts $\times$ 2 methods $\times$ 3 seeds). Fig. 7 shows validation renders throughout training. We save intermediate checkpoints every 1000 iterations for use in variance experiments.

CLIP Score Evaluation. We use ViT-B/32 [71] on renders produced every 25 NeRF training steps. Each validation uses 10 views at fixed elevation (12.5°), camera distance (2.0), and FOV (40°), with azimuth uniform around the object. Scores are averaged across views, prompts, and seeds for the curves in Fig. 6.

Variance Estimation. We measure variance at two levels: (i) *SDS gradient variance*, the variance of the latent-space gradient output by the diffusion model, and (ii) *parameter gradient variance*, variance after backpropagation through the renderer and VAE encoder.

Algorithm 1 Combined IW + stratified + re-noising estimator (per gradient step, SDS).

Require: parameters $\boldsymbol{\theta}$; renders/step R; re-noisings/render $K = B$; base timestep density p; importance proposal $q \propto p\, w_{\text{SDS}}$; frozen teacher $\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\phi}}(\cdot; \boldsymbol{\phi})$; encoder Encode; renderer g'; conditioning $\mathbf{c}$; guidance scale ω.

for $r = 1, \ldots, R$ **do**
  Sample render condition $\boldsymbol{\pi}^{(r)}$; compute $\mathbf{x}^{(r)} = g'(\boldsymbol{\theta}, \boldsymbol{\pi}^{(r)})$, $\mathbf{z}^{(r)} = \text{Encode}(\mathbf{x}^{(r)})$
  for $b = 1, \ldots, B$ **do**
    Draw $\boldsymbol{\xi}_b^{(r)} \sim \mathcal{U}(0, 1)$; set quantile $u_b^{(r)} = (b - 1 + \boldsymbol{\xi}_b^{(r)})/B$
    $t_b^{(r)} \leftarrow \text{CDF}_q^{-1}(u_b^{(r)})$ ▷ stratified inverse-CDF; Fig. 3
    Draw $\boldsymbol{\epsilon}^{(r,b)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$; form $\mathbf{z}_t^{(r,b)} = \alpha_t \mathbf{z}^{(r)} + \sigma_t \boldsymbol{\epsilon}^{(r,b)}$
    $\mathbf{r}^{(r,b)} \leftarrow \hat{\boldsymbol{\epsilon}}_{\boldsymbol{\phi}}(\mathbf{z}_t^{(r,b)}, t_b^{(r)}, \mathbf{c}; \omega) - \boldsymbol{\epsilon}^{(r,b)}$
    $\tilde{w}_b^{(r)} \leftarrow p(t_b^{(r)}) \,/\, q(t_b^{(r)})$
  end for
  $\bar{\mathbf{f}}^{(r)} \leftarrow \frac{1}{B} \sum_b \tilde{w}_b^{(r)}\, w_{\text{SDS}}(t_b^{(r)})\, \mathbf{r}^{(r,b)}$ ▷ per-render IS+strat avg.
end for
return $\hat{\nabla}_{\boldsymbol{\theta}} \leftarrow \frac{1}{R} \sum_r \bar{\mathbf{f}}^{(r)}\, \partial \mathbf{z}^{(r)} / \partial \boldsymbol{\theta}$ ▷ single backward pass per render, through the encoder

We estimate variance using Welford's online algorithm with early stopping on the parameter-gradient estimate: after 1000 iters, check every 50 steps and terminate when relative change stays $< 0.1\%$ for 3 consecutive checks. We validated this rule against a 20 000-sample MSE reference and found agreement within MC noise. Runs are capped at 20 000 iterations, but most stop before 4000.

Table 5: Text prompts used for NeRF training and CLIP score evaluation. Prompts marked with † were used for variance experiments.

Prompt	Variance Experiments
A cactus with pink flowers	
An antique wooden rocking horse	
A golden retriever with a blue bowtie	
An ivory candlestick holder	
A bright red fire hydrant	
A castle-shaped sandcastle	
A plush teddy bear with a satin bow	†
A vintage porcelain doll with a frilly dress	
A tarnished brass pocket watch	
A ceramic teapot with floral patterns	
An antique ruby-studded brooch	
A shiny emerald green beetle	†
A crumpled silver aluminum soda can	
A shimmering emerald pendant necklace	
An antique glass perfume bottle	
A polished mahogany grand piano	†
A pristine white wedding gown	
A chipped porcelain teacup	
A rustic wrought-iron candle holder	
A rusty, vintage metal key	
A delicate, handmade lace doily	
A glossy grand black piano	
A gold glittery carnival mask	†
A shiny red apple	
An antique gold pocket watch	
A fluffy, orange cat	
A scuffed up soccer ball	
A sleek stainless steel teapot	
An intricate ceramic vase with peonies painted on it	
A blooming potted orchid with purple flowers	†

Variance Experiment Configurations. We evaluate SDS gradient variance across a grid of timestep sampling methods and batch configurations.

Batch configurations. We test all (R, K) pairs satisfying $R \times K \leq 32$, where $R \in \{1, 2, 4, 8\}$ is the number of rendered views and $K \in \{1, 2, 4, 8, 16, 32\}$ is the number of timesteps per view, yielding 18 unique configurations.

Timestep sampling methods. We compare two main sampling distributions over $[t_{\min}, t_{\max}]$:

1. *Uniform*: $p(t) = \text{Uniform}[t_{\min}, t_{\max}]$
2. *Importance-weighted* (IW): $q(t) \propto p(t) w_{\text{SDS}}(t)$ using the SDS weighting function from Eq. 6.

For importance-weighted sampling, gradient contributions are scaled by $p(t)/q(t)$ so the estimator is unbiased for the expectation under uniform $p(t)$.

Stratification. For each sampling method, we compare:

1. *iid*: independent sampling from $q(t)$ for each timestep
2. *stratified*: K timesteps per view are stratified into K equal-probability strata (for importance-weighted sampling, this uses inverse-CDF sampling; see Sec. 4.1.3 and Fig. 3).

Experimental matrix. The full matrix of 4 method combinations (2 timestep sampling methods $\times$ 2 batch sampling strategies) across 18 batch configurations is evaluated at training step 5,000 on 5 text prompts; see Sec. 5.1 for results. Variance experiments are conducted on checkpoints trained with 3 different seeds. For each checkpoint, variance is measured using 4 independent Monte Carlo seeds

(3 for the oracle ablation), yielding 5 prompts $\times$ 3 training seeds $\times$ 4 MC seeds $=$ 60 independent variance measurements per method configuration. The same configurations are additionally tested at intermediate training checkpoints (steps 1000–4000) to verify that the conclusions hold throughout optimization; Fig. 13 shows the low-guidance runs at steps 1000, 2000, and 9000.

Ablations. We additionally evaluate two ablations on a limited subset of prompts and steps: (i) *parameter-gradient-weighted (oracle) sampling*, where $q(t)$ is proportional to pre-computed parameter gradient norm per timestep (estimated from prior experiments); see Sec. D.1.8, Fig. 23, and Table 6. (ii) *global vs. per-render stratification*, where all $R \times K$ timesteps are jointly stratified across the entire batch (Eq. 63) rather than per render (Eq. 12); see Fig. 24.

Compute Usage. All experiments are conducted on NVIDIA A100-80GB GPUs (1 GPU per job).

Checkpoint training: We train 30 prompts $\times$ 2 training methods $\times$ 3 seeds $=$ 180 runs, each running 5,000 steps in $\sim$2 hours, totaling $\sim$360 GPU-hours.

Main variance experiments: We evaluate 2 timestep methods (uniform, importance-weighted) $\times$ 2 batch methods (IID, stratified) $\times$ 18 (R, K) configurations (all pairs where $R \times K \leq 32$) $\times$ 4 variance seeds $\times$ 5 prompts $\times$ 3 training seeds $=$ 4,320 runs, totaling $\sim$2,735 GPU-hours. Importance-weighted runs are slightly faster ($\sim$662 GPU-hours per method-batch combination) than uniform runs ($\sim$705 GPU-hours) due to variance reduction.

Oracle ablation (Sec. D.1.8): We compare the weight heuristic against the intractable oracle proposal using 1 timestep method (oracle IW) $\times$ 1 batch method (stratified) $\times$ 18 (R, K) configurations $\times$ 3 variance seeds $\times$ 5 prompts $\times$ 3 training seeds $=$ 810 runs, totaling $\sim$286 GPU-hours.

Total: $\sim$3,400 A100 GPU-hours ($\sim$142 GPU-days) for the three items above; the low-guidance runs (App. Sec. D.1.5) and the global-stratification ablation are not included.

D.1.2 Results

Fig. 11 compares all methods by variance, ECM and RE; Fig. 10 repeats it for one prompt with seed error bars, Fig. 13 tracks variance through training at low guidance, and Fig. 12 shows final renders.

D.1.3 Residual-Norm Variance Analysis

The preceding analysis (Fig. 11) measures variance of the full parameter gradient $\nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathrm{SDS}}$, which includes backpropagation through the renderer Jacobian. Here we present the analogous analysis for the latent-space residual $w_{\mathrm{SDS}}(t)\mathbf{r}$, a commonly used proxy that avoids the cost of backpropagation.

As discussed in Sec. C.1, the residual-norm metric captures variance at an intermediate stage of the gradient pipeline. While easier to compute, it does not account for how the renderer Jacobian modulates contributions across timesteps. Comparing Fig. 14 to Fig. 11 reveals that relative efficiency gains from importance weighting and stratification are qualitatively similar, but the absolute rankings and magnitudes can differ. This confirms that residual-norm variance is a reasonable heuristic for initial guidance on estimator design, but practitioners targeting parameter-space efficiency should validate with full gradient variance when feasible.

D.1.4 Alternative Dispersion Metrics

Throughout the main text, we measure estimator quality using the trace-covariance variance (Eq. 1), which captures the mean-squared error relative to the ground-truth gradient. However, practitioners may also care about directional alignment between estimated and ground-truth gradients, motivating cosine similarity as an alternative dispersion metric. Let $\hat{\boldsymbol{\mu}}$ denote the estimated gradient and $\boldsymbol{\mu}$ denote the ground truth. The cosine similarity

$$\mathrm{CosSim}(\hat{\boldsymbol{\mu}}, \boldsymbol{\mu}) = \frac{\hat{\boldsymbol{\mu}}^\top \boldsymbol{\mu}}{\|\hat{\boldsymbol{\mu}}\|_2 \|\boldsymbol{\mu}\|_2} \tag{64}$$

measures directional agreement independent of magnitude, which may be relevant when gradients are subsequently normalized or clipped. Fig. 15 shows expected cosine similarity to the ground-truth gradient as a function of compute budget for the SDS experiments in Sec. 5.1.

Takeaways. Cosine and variance metrics agree qualitatively: IW, stratification, and reuse all improve estimator quality per unit compute. ECM magnitudes can differ because cosine ignores scale,

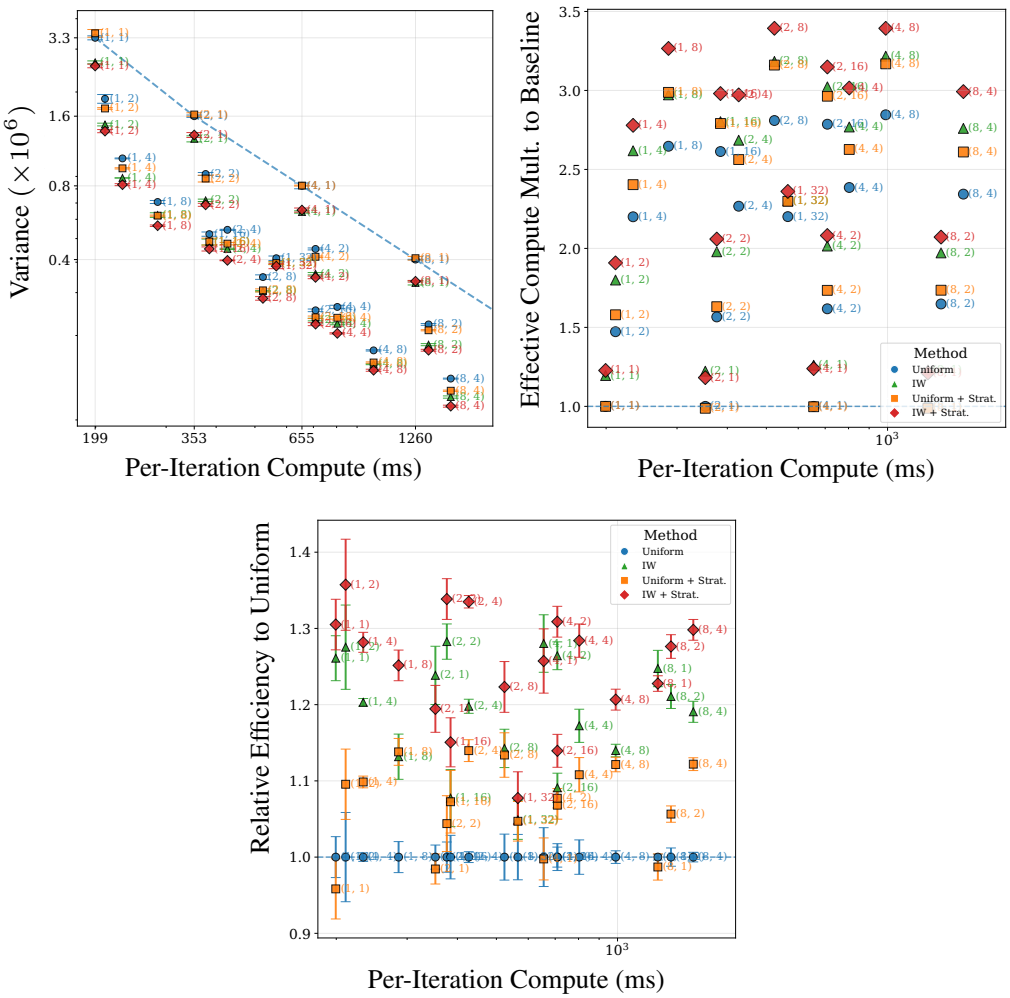


Figure 10: Variance reduction with Monte Carlo seed error bars (single SDS prompt). Same axes as Fig. 11, restricted to one prompt and overlaid with error bars of ± 1 s.d. across 4 independent Monte Carlo seeds for the variance estimator. *Top left:* variance vs. compute. *Top right:* effective compute multiplier relative to the uniform $K = 1$ baseline curve. *Bottom:* relative efficiency vs. uniform at matched (R, K). The relative ranking of methods is stable across seeds: IW+Strat consistently dominates and the per-seed dispersion is small relative to the gap to uniform, so the conclusions are not artifacts of estimator randomness.

while variance captures both directional and magnitude error. When gradients are normalized downstream (e.g., Adam), cosine is the more relevant metric; we report both for completeness.

D.1.5 Analysis of the Low Guidance Regime

Low classifier-free guidance (ω) is an increasingly popular regime for diffusion-guided optimization, as it enables more diverse generations by reducing over-reliance on the text conditioning signal. We investigate whether our variance-reduction strategies provide amplified benefits in this regime.

Setup. We repeat the variance analysis of Fig. 11 at $\omega = 25$ (vs. $\omega = 100$ in the main run) on a subset of 11 prompts, with longer training (Fig. 17 runs to 20 000 steps) and other hyperparameters fixed. We measure parameter-gradient variance at step 5000 and CLIP scores over the trajectory.

Variance reduction benefits. Fig. 16 repeats the variance analysis at $\omega = 25$. At $\omega = 100$, IW+Strat has an effective compute multiplier of $\sim 3.3\times$ averaged over R at $K = 8$ (Table 1); at $\omega = 25$, it reaches $\sim 3.8\times$ at $(R = 1, K = 8)$. The first value averages over R; the second is a single configuration.

Downstream performance. Fig. 17 shows CLIP score versus optimization iteration at low guidance. The performance gap between our method and the baseline is larger than at standard guidance

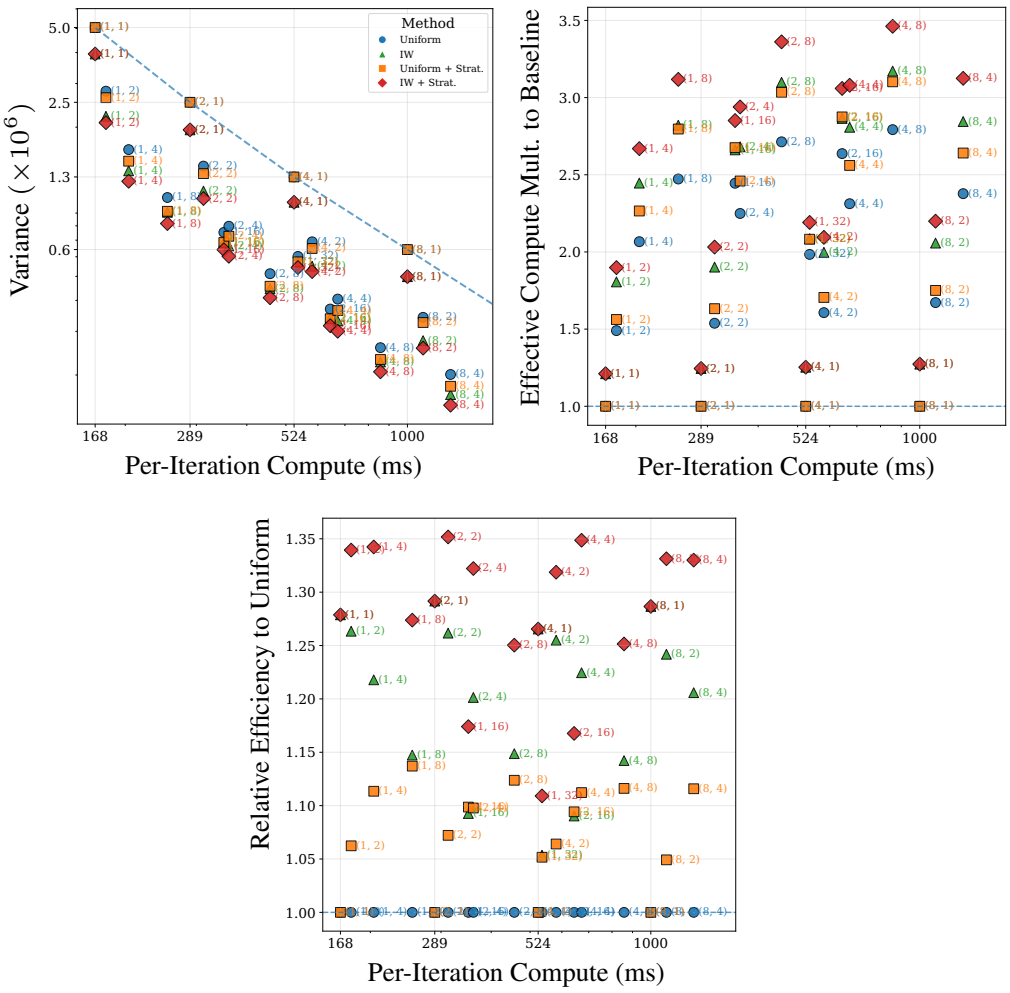


Figure 11: Quantifying variance reduction from hierarchical cost awareness with importance weighting (IW) and stratification (Strat.). Combined effect of IW, stratification, and compute reuse on variance and ECM. *Top left:* Variance (MSE to the ground-truth gradient late in SDS training, which approximates the variance for unbiased estimators) versus compute. Markers: uniform (blue circles), IW (green triangles), Strat (orange squares), IW+Strat (red diamonds); points annotated by (R, K). Compared with uniform $(R{=}2, K{=}1)$, IW+Strat $(R{=}1, K{=}4)$ achieves about two-thirds of the cost and about half the variance. *Top right:* ECM relative to the uniform $K = 1$ baseline curve (Sec. 4.2). Averaged over R, $K = 8$ gives $\sim 2.6\times$ (uniform), $\sim 3.0\times$ (IW), $\sim 3.0\times$ (Strat.), $\sim 3.3\times$ (IW+Strat; Table 1). *Bottom:* RE isolating IW/Strat gains at fixed (R, K): Strat $\sim$6–12%, IW $\sim$14–23%, combined $\sim$25–31% over the recommended sweet spot $K \in \{2, 4, 8\}$. The Table 2 envelope $K \in \{1, \ldots, 32\}$ widens to $\sim$ 5–24% (IW), $\sim$ 0–12% (Strat), $\sim$ 10–31% (combined); gains shrink at very large K as within-render variance saturates and across-render variability binds. Main-body Fig. 4 keeps only uniform and IW+Strat for clarity.

(Fig. 6), so variance reduction matters more for convergence at low guidance. Per-step qualitative trajectories on two prompts (Fig. 18) and across two seeds for a third (Fig. 19) corroborate this: the reduced-variance estimator reaches the baseline's later-stage geometry and appearance in fewer iterations. This suggests that variance reduction may be a complementary strategy for enabling lower guidance settings in practice, potentially reducing high-guidance artifacts without sacrificing convergence speed.

D.1.6 Optimal Pair Probability Distributions

We study two-sample $(K = 2)$ without-replacement estimators, complementing the empirical analysis in Sec. 4.1.3. The Horvitz-Thompson (HT) estimator is governed by a pair-probability matrix $\tilde{Q}(i, j)$ for unordered pairs (i, j) [91, 65]; we compare classical choices for $\tilde{Q}$ with a numerically computed optimum.

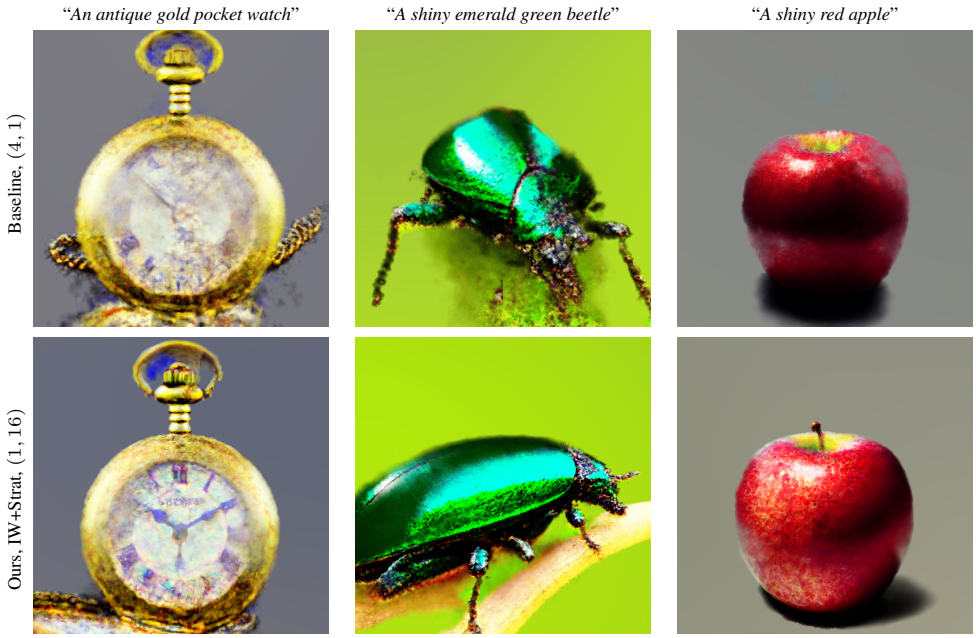


Figure 12: Qualitative Results from Variance Reduction: We show renders for several prompts at the end of training from Fig. 6. On the top, we show renders from a baseline method, while on the bottom, we display a reduced-variance method that combines stratified sampling, importance sampling, and re-noising. Ours costs less per iteration (Fig. 4); both methods run the same number of iterations and are unbiased estimators, yet our reduced-variance strategy (Fig. 11) yields higher visual quality. Fig. 7 shows renders throughout the optimization trajectory.

Setup. Consider N timestep indices with target values $\{\mathbf{y}_i\}_{i=1}^N$, where $\mathbf{y}_i = p_i \mathbf{f}_i$ for base probabilities p_i and per-timestep gradient contributions $\mathbf{f}_i$ (as in Sec. 4.1). With $\tilde{Q}$ symmetric and normalized over unordered pairs, $\sum_{i<j} \tilde{Q}(i,j) = 1$, the marginal inclusion probability $\kappa_i = \sum_{j \neq i} \tilde{Q}(i,j)$ (so $\sum_i \kappa_i = 2$) determines the HT estimator for a sampled pair $(i,j) \sim \tilde{Q}$:

$$\hat{\boldsymbol{\mu}} = \mathbf{y}_i/\kappa_i + \mathbf{y}_j/\kappa_j \tag{65}$$

Following our variance definition (Eq. 1), the theoretical variance is

$$\mathrm{Var}(\hat{\boldsymbol{\mu}}) = \sum_{i<j} \tilde{Q}(i,j) \|\hat{\boldsymbol{\mu}}_{ij} - \boldsymbol{\mu}_{\mathbf{y}}\|_2^2 \tag{66}$$

where $\boldsymbol{\mu}_{\mathbf{y}} = \sum_i \mathbf{y}_i$ is the target sum and $\hat{\boldsymbol{\mu}}_{ij}$ denotes the estimator value for pair (i,j).

Asymptotic rates as context. Stratified sampling on $[0,1]$ is never worse than IID at the same sample count, and for sufficiently smooth integrands (e.g., Lipschitz) it improves the variance rate from $\mathcal{O}(N^{-1})$ to $\mathcal{O}(N^{-3})$ in 1D with one sample per equal-width stratum [65]. Importance sampling with a fixed proposal changes the leading constant (a good proposal shrinks it) but keeps the IID rate. The two-sample analysis below isolates the constant-factor structure these asymptotic arguments hide.

Pair distributions compared. We compare five pair probability matrices (Fig. 20):

1. *IID (uniform):* $\tilde{Q}(i,j) \propto 1$ for $i \neq j$, corresponding to uniform sampling.
2. *Stratified (index):* Partition indices into halves $\mathcal{S}_0, \mathcal{S}_1$ and sample one from each, giving $\tilde{Q}(i,j) = 1/(|\mathcal{S}_0||\mathcal{S}_1|)$ for $i \in \mathcal{S}_0, j \in \mathcal{S}_1$.
3. *IW only:* $\tilde{Q}(i,j) \propto w(t_i)w(t_j)$, with w the diffusion objective's timestep weight.
4. *IW+Stratified:* Stratify in CDF space of the importance distribution (Sec. 4.1.3), then sample within each stratum proportionally to $w(t)$.
5. *Optimal (Sinkhorn):* An entropy-regularized approximation of the variance-minimizing $\tilde{Q}$ with target marginals $\kappa_i \propto \|\mathbf{y}_i\|_2$.

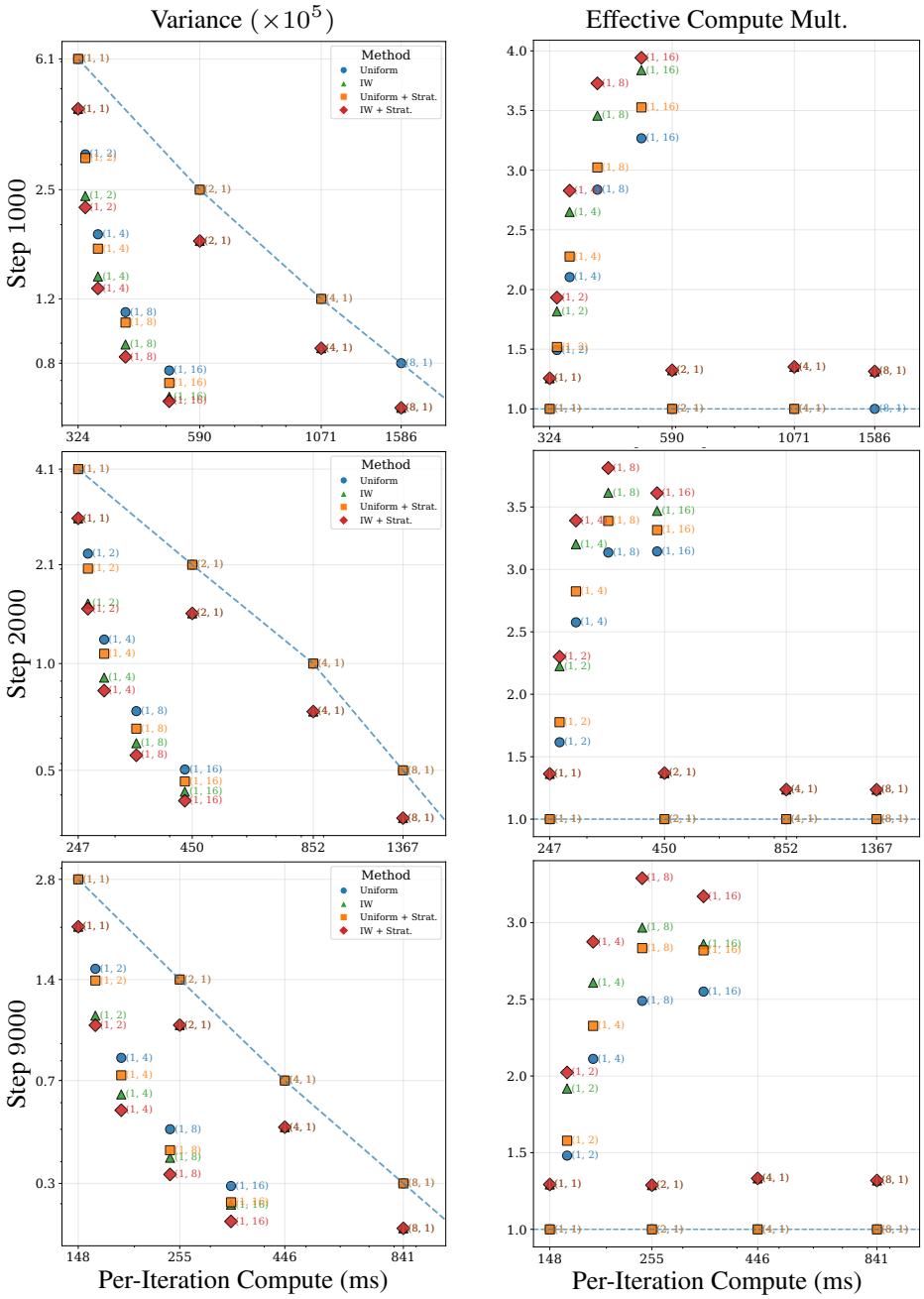


Figure 13: Variance reduction across training, low classifier-free guidance ($\omega = 25$). Analogous to Fig. 11, measured at three optimization checkpoints. Rows: training step 1000, 2000, and 9000. *Left:* variance vs. compute. *Right:* effective compute multiplier relative to the uniform $K = 1$ baseline curve. The continuation shows relative efficiency vs. uniform at matched (R, K). Higher K wins more strongly early in training, when rendering is more expensive relative to denoising and re-noising amortizes that cost most efficiently; the gap narrows in late training but variance reduction continues to dominate the uniform baseline at every checkpoint.

Optimal pair distribution via Sinkhorn. The optimal $\tilde{Q}$ minimizes variance subject to marginal constraints $\sum_{j \neq i} \tilde{Q}(i,j) = \kappa_i$ for target inclusion probabilities $\kappa_i \propto \|\mathbf{y}_i\|_2$ (a per-snapshot optimum that treats $\mathbf{y}_i$ as deterministic). We cast this as an entropy-regularized optimal-transport problem [11] with cost matrix $C_{ij} = (\mathbf{y}_i/\kappa_i)^\top (\mathbf{y}_j/\kappa_j)$, penalizing pairs whose scaled contributions are aligned. The Gibbs kernel $K_{ij} = \exp(-\beta C_{ij}/\text{scale})$ is normalized to the 95th percentile of off-diagonal costs. Sinkhorn iteration rescales the kernel to match the target marginals, produc-

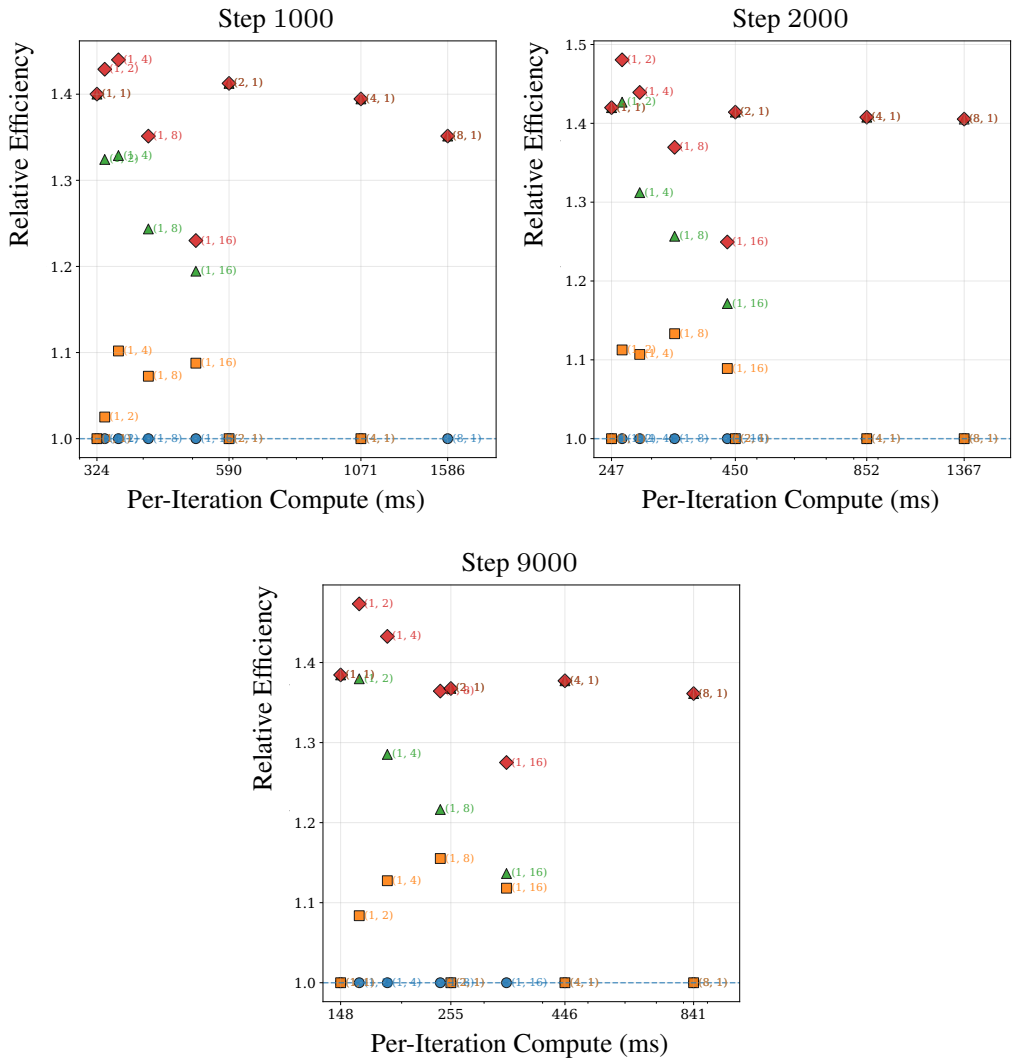


Figure 13: (continued) Relative efficiency vs. uniform at matched (R, K) at training steps 1000, 2000, and 9000 ($\omega = 25$).

ing a $\tilde{Q}$ that concentrates mass on pairs with diverse gradient directions while respecting the target marginals.

Results. Fig. 20 visualizes the five pair-probability matrices computed from gradient data on a single SDS prompt at the end of training (Sec. 5.1). The Sinkhorn distribution has the lowest variance of the five, pairing timesteps with complementary gradient directions. IW+Stratified comes close, reaching $\sim 91\%$ of its efficiency multiplier without solving the transport problem. The variance ordering (IID $>$ Stratified $>$ IW $>$ IW+Strat $>$ Optimal) and the corresponding effective compute multipliers over IID ($1.04\times$, $1.22\times$, $1.29\times$, $1.42\times$) match our empirical findings, so the tractable estimator captures most of the theoretical benefit.

D.1.7 Sensitivity to Render vs. Denoise Cost Ratio

The results in Fig. 11 use measured wall-clock compute, which conflates rendering, encoding, and denoising costs. To isolate the effect of the render-to-denoise cost ratio, we repeat the analysis (Fig. 21) with simulated cost models $\mathcal{B} = \alpha R + RK$, where α controls the relative cost of rendering versus denoising. Setting $\alpha = 0$ simulates the extreme where rendering is free (only denoising contributes), $\alpha = 1$ simulates equal per-operation cost, and $\alpha = 100$ simulates the rendering-dominated regime that occurs in practice for differentiable rendering, latent-diffusion encoders with backpropagation, and physics simulators.

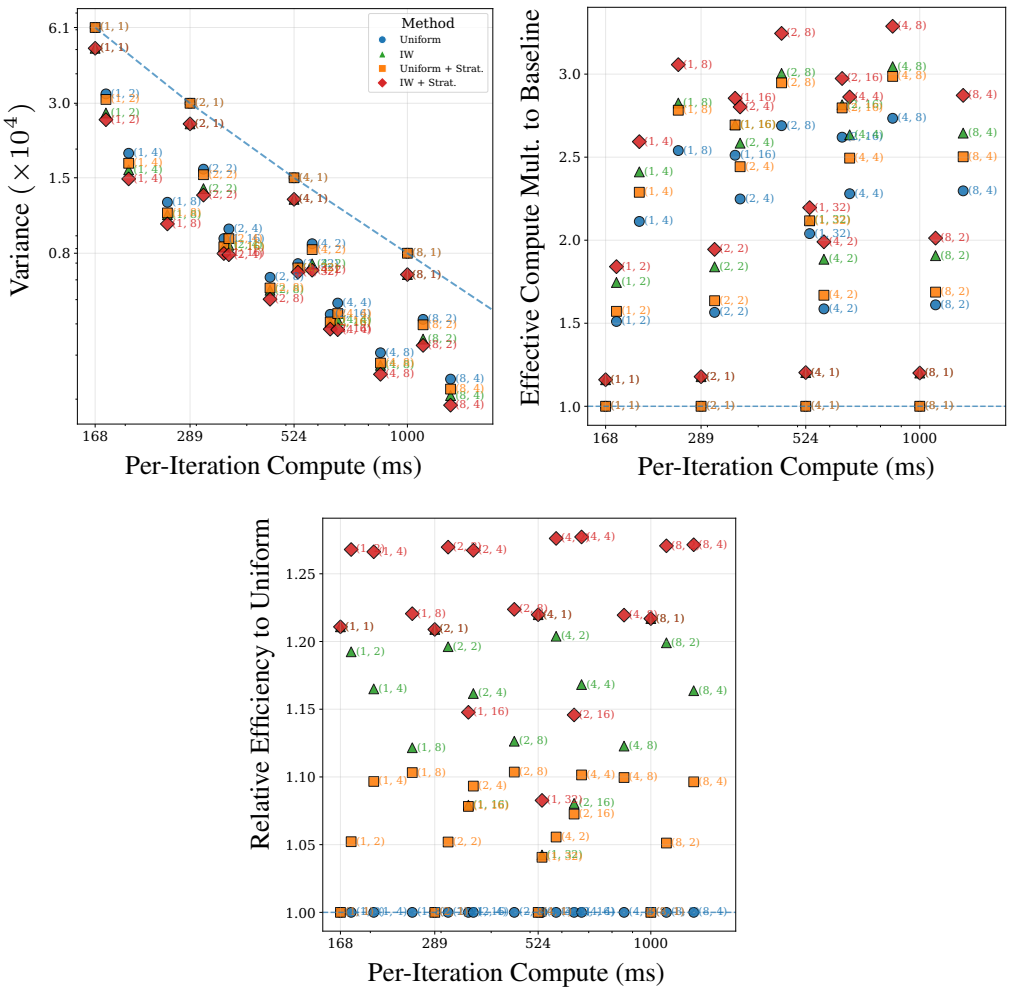


Figure 14: Variance reduction measured via latent-space residual norm. Analogous to Fig. 11, but measuring variance of the weighted residual $w_{\text{SDS}}(t)\mathbf{r}$ rather than the full parameter gradient. *Top left:* Variance versus compute budget. Colors denote uniform baseline, IW only, stratification only, and IW+Strat combined. Points are annotated by (R, K) tuples. *Top right:* Effective compute multiplier relative to the uniform $K = 1$ baseline curve. *Bottom:* Relative efficiency to uniform sampling at matched (R, K) configurations. The qualitative trends match Fig. 11: importance weighting and stratification both reduce variance, and their benefits combine. However, the residual-norm metric does not account for how the renderer Jacobian modulates per-timestep contributions, so absolute efficiency gains and optimal configurations may differ from the parameter-gradient analysis. This supports using residual-norm variance as an inexpensive diagnostic during development, while validating final design choices with parameter-gradient variance.

Takeaway. At zero render cost, $K = 1$ is best for every method, and IW+Strat stays above the uniform baseline at small K because of IW. When the render cost grows, larger K becomes increasingly beneficial because the expensive upstream computation amortizes over more cheap denoiser calls. Our wall-clock experiments (Fig. 11) reflect the effective scale of these costs after parallelization, and the cost-ratio sweep here shows that the conclusion (IW+Strat beats the uniform baseline) holds across the full range of plausible cost ratios, not only the regime our hardware happens to occupy.

D.1.8 Importance Sampling: Weight Heuristic vs. Oracle

We compare the weight heuristic IW ($q \propto p(t)w_{\text{SDS}}(t)$; Sec. 4.1.2) against the intractable oracle IW ($q^\star$ from Eq. 23). Here $\mathbf{f}$ denotes the full parameter-gradient contribution ($\mathbf{h}$ in Sec. 4.1).

Why is the oracle intractable? The variance-minimizing proposal $q^\star(t) \propto p(t)\sqrt{\mathbb{E}[\|\mathbf{f}(t, \boldsymbol{\xi})\|_2^2 \mid t]}$ requires estimating the conditional second moment of the gradient contribution at each timestep. For parameter gradients, this requires estimating $\mathbb{E}[\|\nabla_{\boldsymbol{\theta}} \mathcal{L}_{\text{SDS}}\|_2^2 \mid t]$ across the distribution of renders, camera views, and noise samples. This is prohibitively expensive during training: it requires iso-

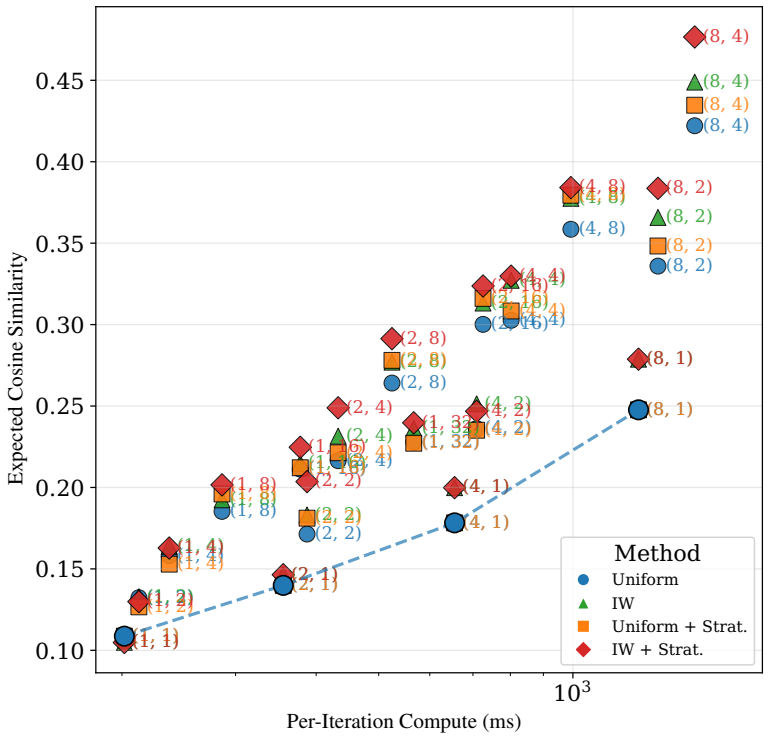


Figure 15: Cosine similarity to ground-truth gradient versus compute budget. Cosine similarity between estimated and ground-truth parameter gradients (SDS) for uniform, IW, Strat. and IW+Strat across (R, K). Higher is better. Ranking matches the variance analysis in Fig. 4: IW+Strat wins at matched compute, and reuse ($K > 1$) helps. ECM magnitudes differ from the variance metric because cosine similarity is scale-invariant.

lating per-timestep gradient norms, but standard batched backpropagation aggregates contributions across all K re-noisings per render (Sec. 4.1.1), making per-timestep norms inaccessible without running K separate backward passes.

How do we evaluate the oracle? Although per-timestep norms are inaccessible when $K > 1$, our $K = 1$ variance measurement experiments (Sec. C.1) do isolate per-timestep gradient contributions. We construct an empirical oracle by collecting these per-timestep gradient norms, binning $\|\mathbf{f}(t, \boldsymbol{\xi})\|_2^2$ by timestep, computing the empirical mean within each bin, and using these to define the oracle proposal $q^\star$. Fig. 22 visualizes these tracked gradient norms, showing both the weighted latent-space residual $\|w_{\text{SDS}}(t)\mathbf{r}\|_2$ and the full parameter gradient norm $\|\mathbf{f}(t, \boldsymbol{\xi})\|_2$ as a function of timestep.

Results. Fig. 23 and Table 6 summarize results on the low-guidance ($\omega = 25$) runs of App. Sec. D.1.5 at training step 5000, averaged over prompts; the uniform and weight-heuristic points are those of Fig. 16. Both IW methods reduce variance more than uniform sampling (Table 6). The weight heuristic attains 94% to 97% of the oracle's relative efficiency across all K, so $w_{\text{SDS}}(t)$ is a practical proxy for the optimal proposal. Because w_{SDS} is deterministic given t, the oracle factors as $q^\star(t) \propto p(t)\, w_{\text{SDS}}(t) \sqrt{\mathbb{E}[\|\mathbf{J}_{\boldsymbol{\theta}}^\top \mathbf{r}\|_2^2 \mid t]}$, so $q \propto p\, w_{\text{SDS}}$ is exact when that conditional second moment is flat in t. The Jacobian stays in the estimator and is dropped only from the proposal; our empirical oracle includes it, so the 94% to 97% relative efficiency measures what dropping it costs. This close agreement is explained by Fig. 22: the weight function $w_{\text{SDS}}(t)$ closely tracks the empirical gradient norm profile, so importance sampling with $q \propto p(t) w_{\text{SDS}}(t)$ approximates the oracle without requiring expensive norm estimation.

D.1.9 Prompt Ablation

We evaluate the consistency of variance reduction benefits across five diverse text prompts: emerald beetle, gold mask, mahogany piano, orchid pot, and teddy bear. Table 7 summarizes the results.

Results. Our variance reduction methods behave consistently across prompts. All prompts achieve peak ECM at $K = 8$ (Table 7a), with values ranging from 2.73× (gold) to 3.58× (mahogany). The marginal benefit of IW+Strat over uniform peaks at K from 2 to 4 (Table 7b), consistent with dimin-

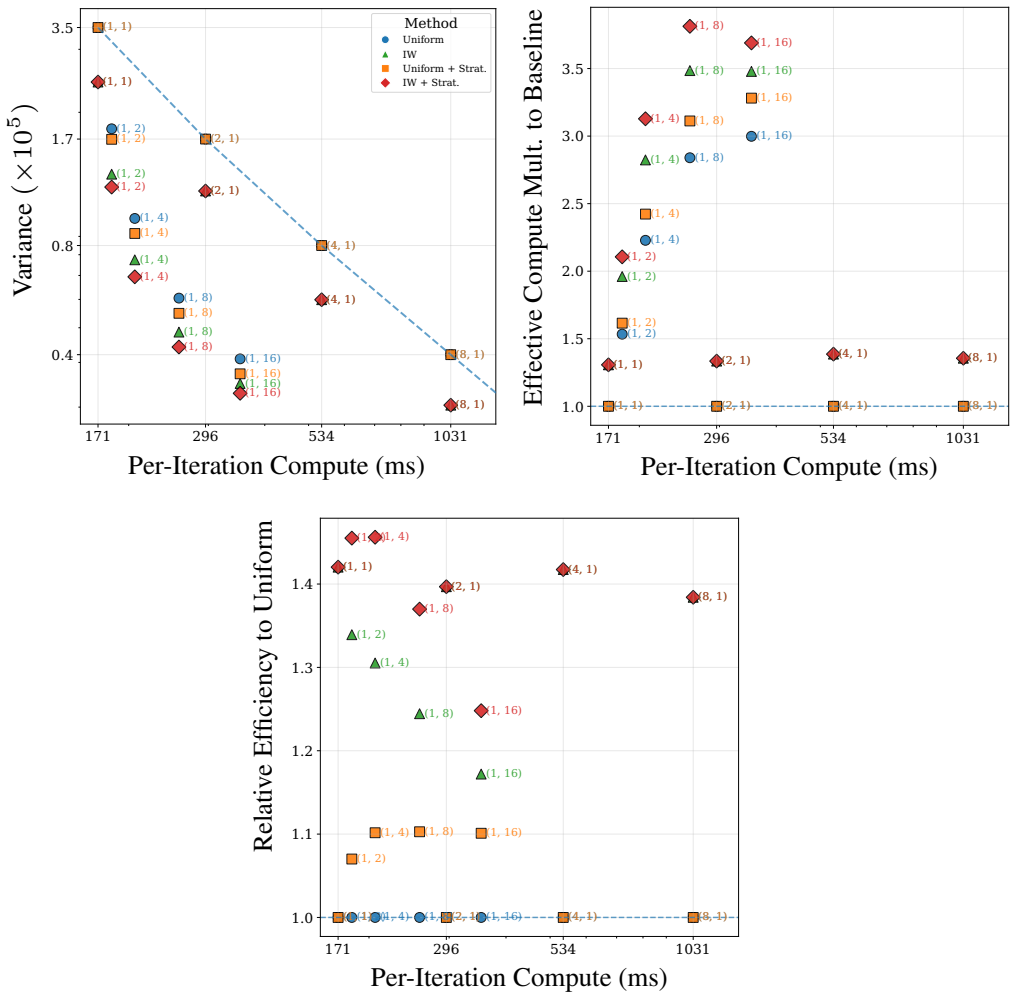


Figure 16: Variance reduction in the low guidance regime ($\omega = 25$)**.** Analogous to Fig. 11, measured at training step 5000 with reduced classifier-free guidance. *Top left:* Variance versus compute budget. Colors denote uniform baseline, IW only, stratification only, and IW+Strat combined (red). Points are annotated by (R, K) tuples. *Top right:* Effective compute multiplier relative to the uniform $K = 1$ baseline curve. At $(R, K) = (1, 8)$, the only $K = 8$ configuration plotted, the multipliers are $\sim$2.8$\times$ (uniform), $\sim$3.5$\times$ (IW), $\sim$3.1$\times$ (Strat.), and $\sim$3.8$\times$ (IW+Strat). *Bottom:* Relative efficiency to uniform at matched (R, K) configurations. For $K \geq 2$ ($R = 1$), importance weighting gives $\sim$17–34% and stratification $\sim$7–10%, and their combination $\sim$25–46%.

Table 6: Importance sampling ablation on the low-guidance ($\omega = 25$) runs of App. Sec. D.1.5 at training step 5000, averaged over prompts. **(a)** ECM relative to uniform $K = 1$. **(b)** Relative efficiency vs. uniform at matched K. The weight heuristic attains 94% to 97% of the oracle's relative efficiency.

(a) Effective Compute Multiplier

K	Uniform	Weight Heur. IW	Oracle IW
1	1.00×	1.31×	**1.39**×
2	1.65×	2.12×	**2.23**×
4	2.41×	2.98×	**3.12**×
8	3.00×	3.55×	**3.66**×
16	3.13×	3.54×	**3.61**×

(b) Relative Efficiency vs. Uniform

K	Weight Heur. IW	Oracle IW
1	1.40×	**1.47**×
2	1.34×	**1.43**×
4	1.31×	**1.37**×
8	1.24×	**1.28**×
16	1.17×	**1.21**×

ishing returns from importance weighting at higher K as the estimator approaches the continuous limit. Method rankings remain stable across all prompts (Table 7c,d): IW+Strat consistently outperforms both IW-only and Strat-only, which in turn outperform uniform sampling. This confirms that the complementary nature of importance weighting and stratification is not prompt-specific.

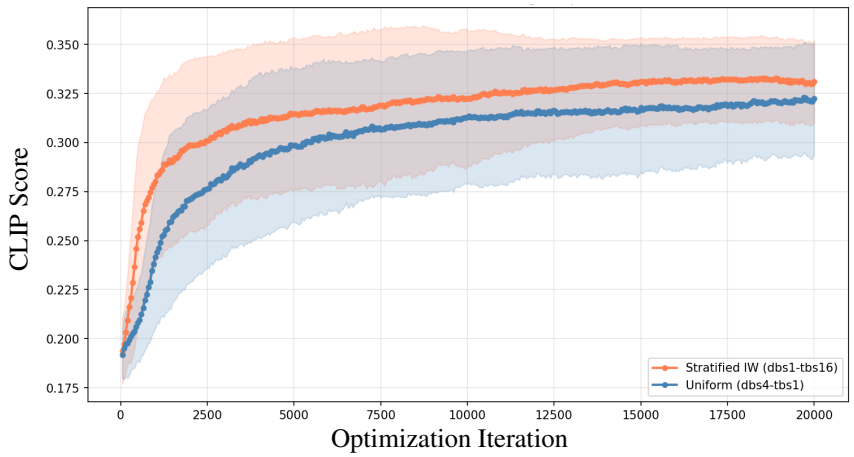


Figure 17: Performance gains from variance reduction at low guidance ($\omega = 25$). CLIP score vs. iteration at $\omega = 25$, averaged over 11 prompts, 3 seeds, and multiple views ($\pm$ std. dev.). Baseline $(4, 1)$ vs. ours $(1, 16)$ (Strat+IW+re-noising), whose iterations cost less (Fig. 16); the legend's dbs and tbs are R and K. The gap is larger than at standard guidance (Fig. 6), so variance reduction matters more at low guidance.

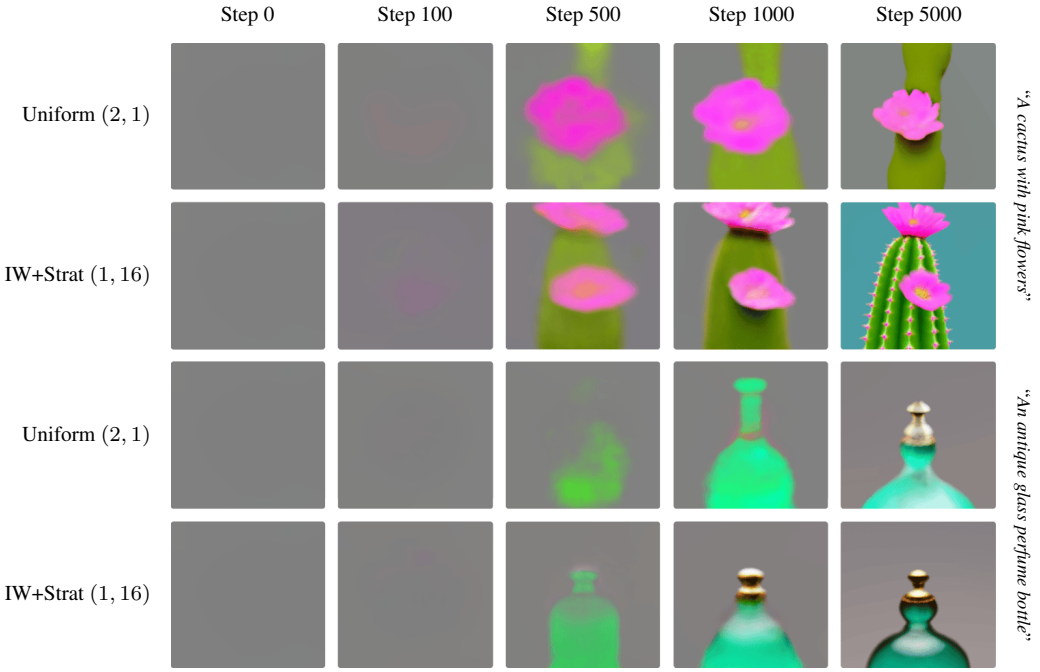


Figure 18: Qualitative SDS trajectories at low classifier-free guidance ($\omega = 25$). Comparison of the uniform $(R, K) = (2, 1)$ baseline and our IW+Strat $(1, 16)$ method on two prompts. Consistent with the CLIP curves in Fig. 17, the reduced-variance estimator reaches the baseline's later-stage geometry and appearance in fewer iterations, with the advantage becoming clearer in this lower-guidance regime where the SDS gradient is noisier. Multi-seed trajectories for an additional prompt are in Fig. 19.

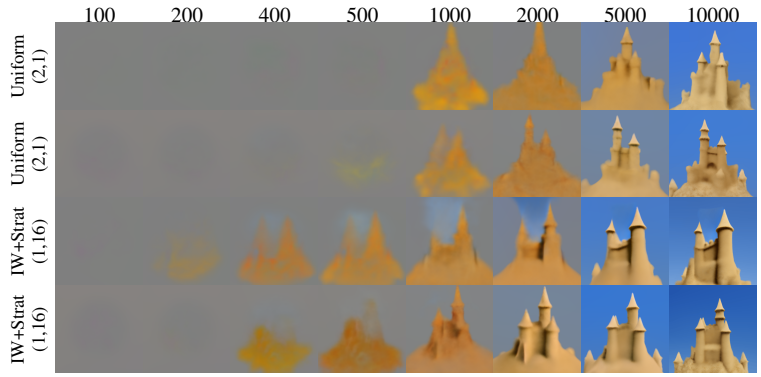


Figure 19: Qualitative SDS trajectories at low classifier-free guidance ($\omega = 25$): comparison of the uniform $(R, K) = (2, 1)$ baseline (top two rows) and our IW+Strat $(1, 16)$ method (bottom two rows) for the prompt "*A castle-shaped sandcastle*" across 2 training seeds. Columns show renders at training steps 100, 200, 400, 500, 1000, 2000, 5000, 10 000. Consistent with the CLIP curves in Fig. 17, our method reaches the baseline's converged geometry in substantially fewer iterations.

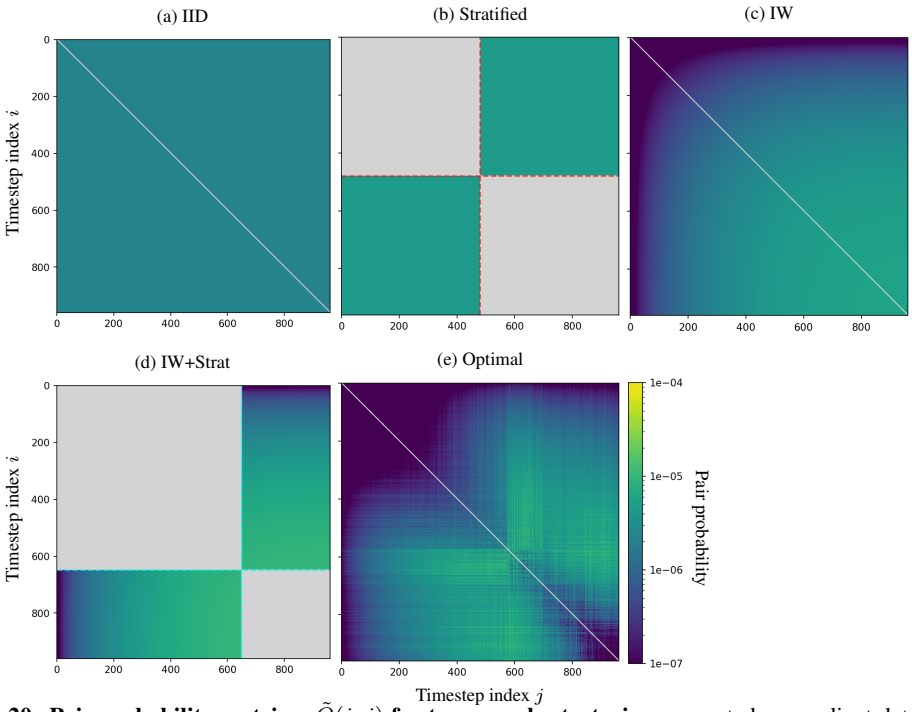


Figure 20: Pair probability matrices $\tilde{Q}(i, j)$ for two-sample strategies, computed on gradient data from a single SDS prompt at the end of training. Each panel shows the probability of selecting pair (i, j) on a log scale (brighter = higher probability, gray = zero). (a) IID places equal mass on all pairs ($1.00\times$, baseline). (b) Index-based stratification concentrates mass in off-diagonal blocks ($1.04\times$). (c) Importance weighting concentrates on high-weight timesteps ($1.22\times$). (d) IW+Stratified combines both via CDF-space stratification ($1.29\times$). (e) Sinkhorn-optimal pairs timesteps with complementary gradient directions ($1.42\times$). Effective compute multipliers (in parentheses) are computed from theoretical variances. IW+Stratified reaches $\sim 91\%$ of the optimal allocation's efficiency multiplier, validating it as a practical near-optimum.

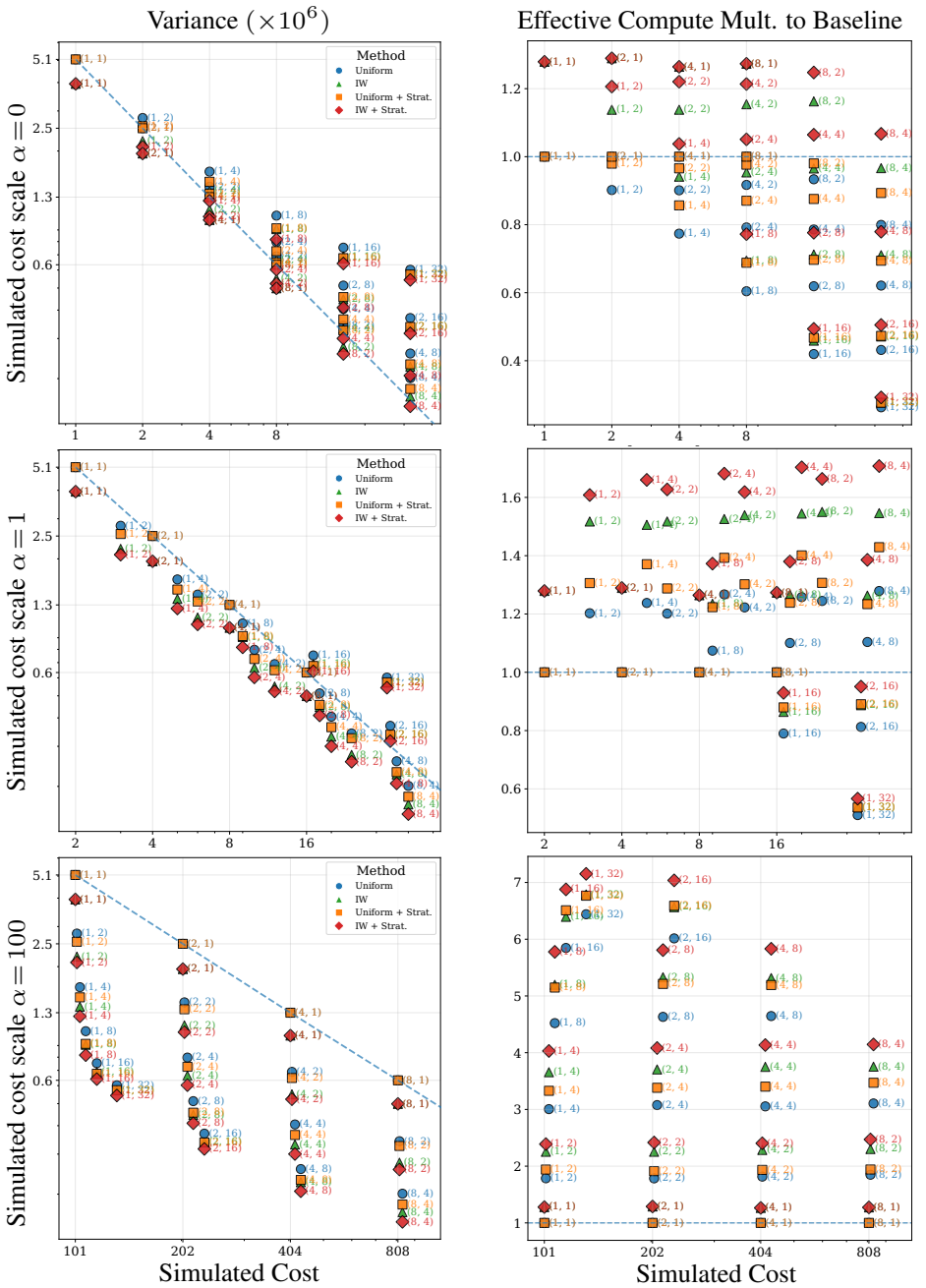


Figure 21: Sensitivity of variance reduction to render-vs-denoise cost ratio. Analysis of Fig. 11 repeated with simulated cost $\mathcal{B} = \alpha R + RK$ to isolate render cost. *Top* ($\alpha = 0$)*:* Render free; $K = 1$ is best for every method, and IW+Strat stays above the baseline at small K. *Middle* ($\alpha = 1$)*:* Equal render and denoise cost; modest re-noising improves ECM. *Bottom* ($\alpha = 100$)*:* Rendering dominates; larger K gives larger ECM through render amortization. Columns: variance and effective compute multiplier to baseline; the continuation shows relative efficiency to uniform. Colors and annotations follow Fig. 11.

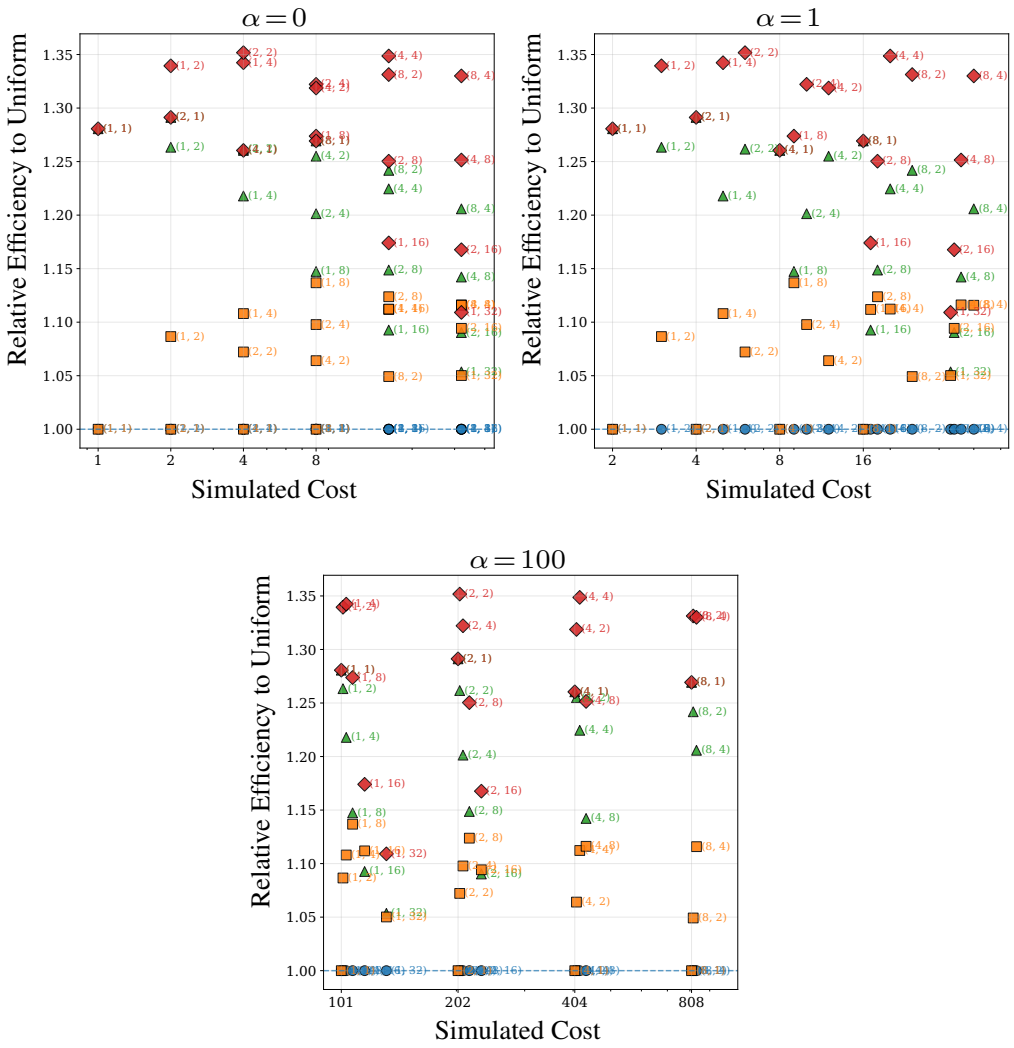


Figure 21: (continued) Relative efficiency to uniform at matched (R, K) for simulated cost scales $\alpha = 0$, 1, and 100.

Table 7: Prompt ablation for variance reduction methods. Five prompts (emerald beetle, gold mask, mahogany piano, orchid pot, teddy bear). **(a)** ECM for IW+Strat by K; $K = 8$ is optimal across prompts. **(b)** RE vs. uniform for IW+Strat; peak at $K = 2$–4. **(c)** ECM for all methods at $K = 8$; rankings (IW+Strat $>$ IW $\approx$ Strat $>$ Uniform) are stable. **(d)** RE vs. uniform at $K = 8$; IW and Strat are complementary across prompts.

(a) ECM by K (IW+Strat)

Prompt	$K=1$	$K=2$	$K=4$	$K=8$	$K=16$	$K=32$
Emerald	1.23×	2.05×	2.99×	**3.41**×	3.12×	2.40×
Gold	1.20×	1.95×	2.59×	**2.73**×	2.33×	1.73×
Mahogany	1.28×	2.07×	3.12×	**3.58**×	3.24×	2.32×
Orchid	1.24×	2.04×	2.98×	**3.33**×	2.97×	2.15×
Teddy	1.26×	2.13×	3.01×	**3.42**×	3.05×	2.31×
Avg	1.24×	2.05×	2.94×	**3.29**×	2.94×	2.18×

(b) RE by K (IW+Strat)

Prompt	$K=1$	$K=2$	$K=4$	$K=8$	$K=16$	$K=32$
Emerald	1.23×	**1.29**×	1.28×	1.21×	1.14×	1.07×
Gold	1.20×	**1.27**×	1.25×	1.18×	1.12×	1.12×
Mahogany	1.28×	1.32×	**1.35**×	1.29×	1.22×	1.11×
Orchid	1.24×	1.31×	**1.35**×	1.28×	1.19×	1.09×
Teddy	1.26×	**1.33**×	1.33×	1.28×	1.17×	1.11×
Avg	1.24×	1.30×	**1.31**×	1.25×	1.17×	1.10×

(c) ECM at $K = 8$ (all methods)

Prompt	Uniform	IW	Strat.	IW+Strat.
Emerald	2.80×	3.17×	3.15×	**3.41**×
Gold	2.32×	2.55×	2.55×	**2.73**×
Mahogany	2.78×	3.22×	3.07×	**3.58**×
Orchid	2.60×	2.98×	2.98×	**3.33**×
Teddy	2.67×	3.13×	3.04×	**3.42**×
Avg	2.63×	3.01×	2.96×	**3.29**×

(d) RE at $K = 8$ (all methods)

Prompt	Uniform	IW	Strat.	IW+Strat.
Emerald	1.00×	1.13×	1.12×	**1.21**×
Gold	1.00×	1.10×	1.10×	**1.18**×
Mahogany	1.00×	1.16×	1.10×	**1.29**×
Orchid	1.00×	1.15×	1.15×	**1.28**×
Teddy	1.00×	1.17×	1.14×	**1.28**×
Avg	1.00×	1.14×	1.12×	**1.25**×

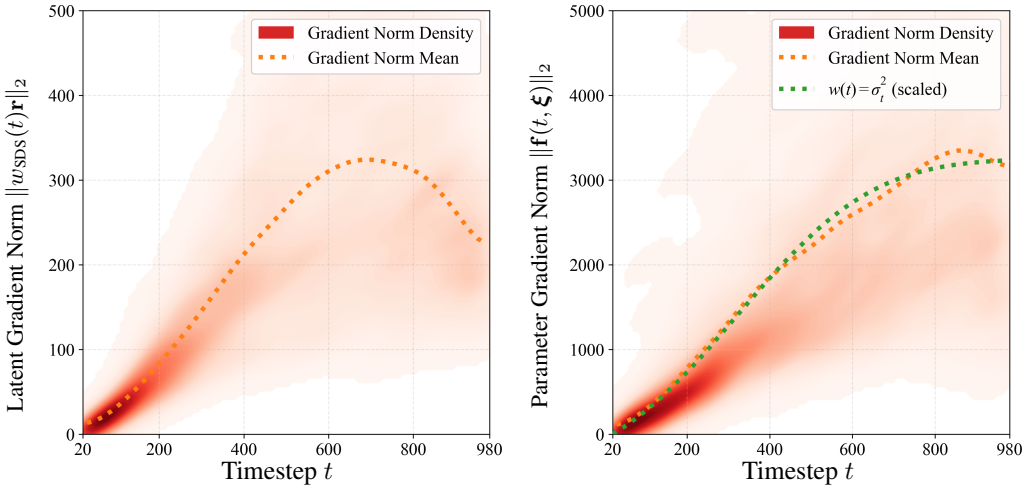


Figure 22: Weight function closely tracks gradient magnitude across timesteps. We visualize empirical gradient norms as a function of timestep t during SDS optimization, alongside the scaled weight function behind the importance-sampling proposal (right panel). *Left:* Latent-space gradient contribution $\|w_{\text{SDS}}(t)\mathbf{r}\|_2$, where $\mathbf{r} = \hat{\boldsymbol{\epsilon}}_{\boldsymbol{\phi}}(\mathbf{z}_t, t, \mathbf{c}; \omega) - \boldsymbol{\epsilon}$ is the noise prediction residual. *Right:* Full parameter gradient norm $\|\mathbf{f}(t, \boldsymbol{\xi})\|_2$, aggregated over camera views, renders, and noise. The weight-based proposal $q \propto p(t)w_{\text{SDS}}(t)$ closely tracks the empirical gradient norm profile, explaining why it attains 94% to 97% of the oracle proposal's relative efficiency (Table 6) without requiring expensive per-timestep norm estimation. The oracle proposal $q^\star \propto p(t)\sqrt{\mathbb{E}[\|\mathbf{f}(t, \boldsymbol{\xi})\|_2^2 \mid t]}$ is given for reference.

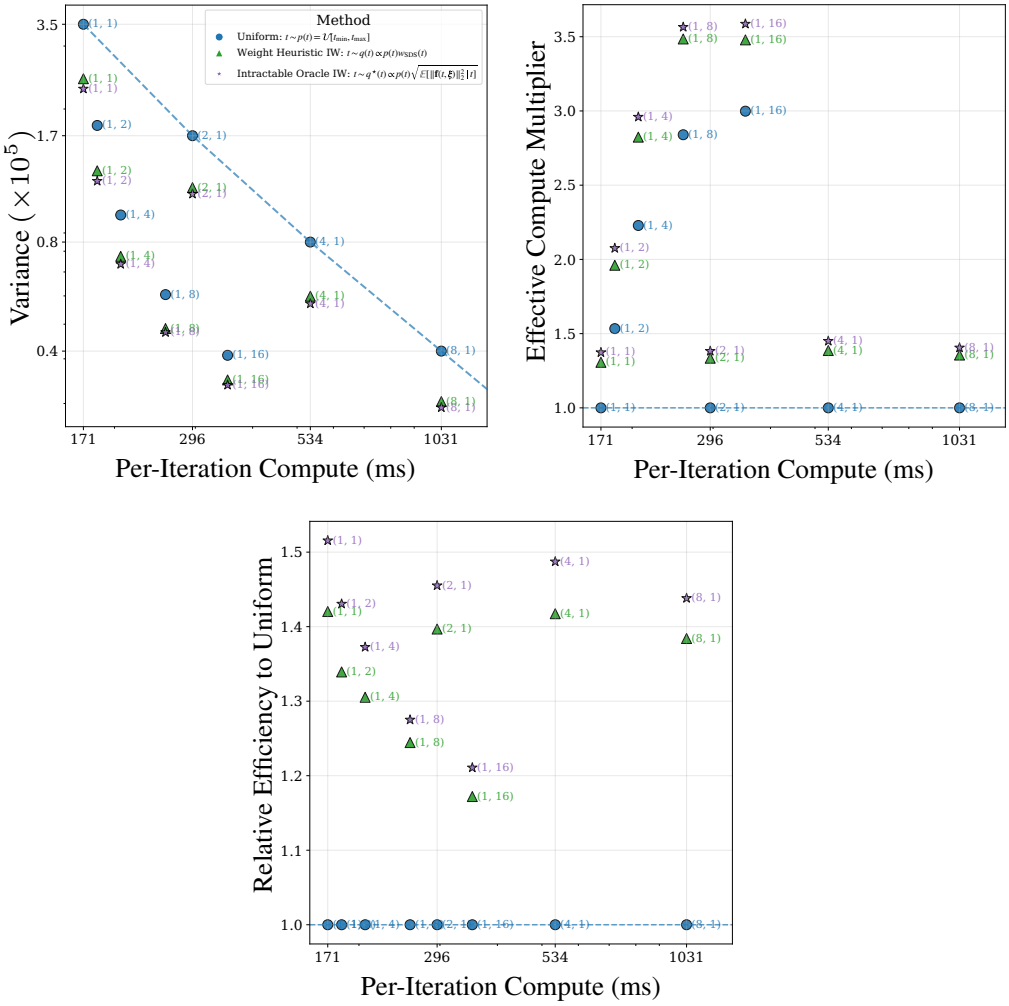


Figure 23: Importance Sampling Strategy Comparison: Weight-Based Heuristic versus Oracle. We compare three importance sampling approaches for parameter gradient estimation: uniform sampling (baseline), our weight-based importance sampling using $q(t) \propto p(t) w_{\text{SDS}}(t)$ as described in Sec. 4.1.2, and the intractable oracle proposal $q^\star(t) \propto p(t)\sqrt{\mathbb{E}[\|w_{\text{SDS}} \mathbf{J}_{\boldsymbol{\theta}}^\top \mathbf{r}\|_2^2 \mid t]}$ that requires estimating conditional second moments of parameter-gradient contributions. *Top left:* Parameter gradient variance versus compute budget in milliseconds. Points are annotated by (R, K) configurations. *Top right:* Effective compute multiplier relative to the uniform $K = 1$ baseline curve. *Bottom:* Relative efficiency (RE) isolating the gain from importance sampling by comparing to uniform sampling at the same (R, K) configuration. The weight-based heuristic achieves $\sim$17–40% improvements. Our zero-cost, weight-based importance-sampling heuristic performs nearly as well as the intractable oracle across all compute budgets, so $w_{\text{SDS}}(t)$ is a practical proxy for gradient magnitude when designing importance proposals.

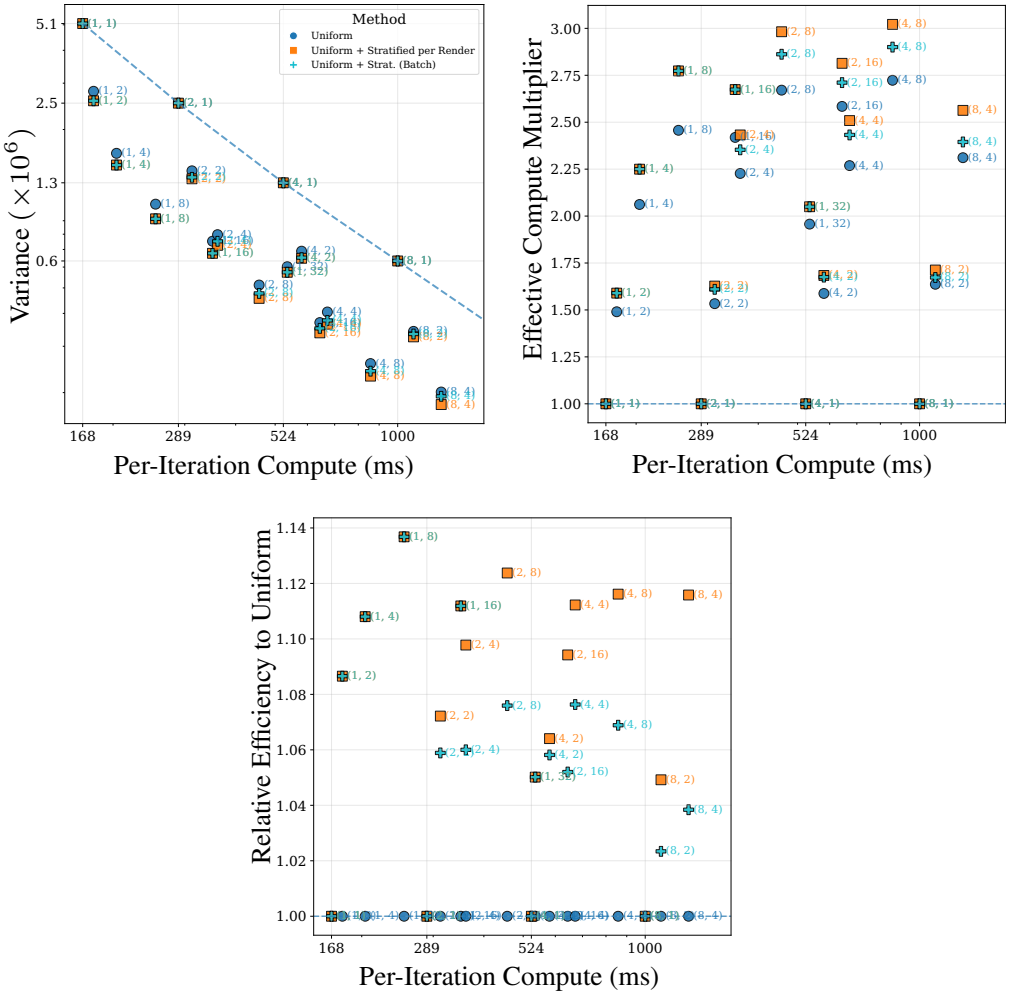


Figure 24: Comparing Per-Render and Global Stratification Strategies. This figure ablates two stratified sampling approaches: per-render stratification (Eq. 12), which stratifies timesteps independently within each render's K re-noisings, versus global stratification (Eq. 63), which stratifies timesteps across all $R \times K$ samples in the batch. *Top left:* Variance versus compute budget for the uniform baseline (blue circles), global stratification (cyan crosses; "Batch" in the legend), and per-render stratification (orange squares). Points are annotated by (R, K) configurations. *Top right:* Effective compute multiplier relative to the uniform $K = 1$ baseline curve. *Bottom:* Relative efficiency (RE) isolating the gain from stratification by comparing to uniform sampling at the same (R, K) configuration. Per-render stratification matches or outperforms global stratification when $K > 1$, as it exploits the hierarchical structure by reducing within-render variance. When $K = 1$, per-render stratification degenerates to uniform sampling (no timesteps to stratify within a single re-noising), so global stratification is the only stratified option there; this sweep shows no gain from it at $K = 1$ (bottom). This motivates our choice of per-render stratification for SDS experiments, where renders are expensive, and configurations with $K > 1$ are most efficient.

D.2 Single-Step Diffusion Distillation

D.2.1 Details

Hyperparameters: For all of our diffusion distillation experiments, we use the public DiT-XL/2 weights published alongside the DiT source code [68]. We used the ImageNet-256 dataset [12], encoded with the pretrained Stable Diffusion encoder [75], per the original teacher model.

We train one-step generators using the DMD2 algorithm [104]. This replaces the regression loss with an additional learned-feature discriminator and an alternating update schedule for the fake score and the generator network. Unless otherwise specified, we use a learning rate of 10^{-5} for all experiments and perform 5 fake-score/discriminator updates for each student update.

Compute Usage: NVIDIA A100 GPUs. Batch 8 baseline $(8, 1)$: ~ 0.38 s/iter; $8\times$ resampling ~ 0.47 s; $16\times$ resampling ~ 0.59 s. Batch 48: ~ 1.0 s/iter. We evaluate FID at checkpoints, averaging over 5 seeds.

D.2.2 Results

We follow our variance computation framework to evaluate the variance of the Monte Carlo estimator of Eq. 51. We run the online variance estimator for at least 1000 steps, stopping when the parameter gradient variance remains within 0.1% for 50 consecutive steps. We compute the variance estimates after training a one-step generator for 20 000 steps using a batch size of 8. The $(8, 1)$ to $(8, 16)$ IID parameter-gradient ratio in Table 3, $982/59.9 \approx 16.4$, slightly exceeds the factor of 16 that Eq. 10 allows for 16 IID re-noisings, so these ratios carry estimation noise. We also investigated the effectiveness of variance reduction at initialization[1] and earlier checkpoints, finding a similar overall pattern.

Investigating the effect on FID. These experiments ask why lower gradient variance did not translate into better FID in diffusion distillation.

We explore student-timestep resampling at batch 8, which gives large variance reduction at low cost. Per-iter timings: batch 48 ~ 1 s; batch 8 ~ 0.38 s; $8\times$ resampling ~ 0.47 s; $16\times$ ~ 0.59 s.

We show FID over the course of optimization. We show the convergence rate as a function of the optimization step and also rescale the x-axis based on the compute time spent. Because the raw FID values at each checkpoint exhibit high variance, we compute the average over 5 random training seeds and display a shaded band indicating one standard deviation. In Fig. 26, top, we plot the best FID achieved by iteration i while the raw (much noisier) FID values per iteration are shown in Fig. 26, bottom, for this setting only.

Overall, we find that optimization steps with variance reduction yield convergence rates and final values that are similar to or better than those of the baseline. However, when accounting for the additional compute cost, the baseline method remains comparable.

We also applied resampling and stratification to the fake-score update step (Fig. 27). This costs more compute per step but gives a larger per-step convergence gain, in line with the larger batch size; adjusted for compute, the benefit is less apparent.

D.2.3 Understanding the DMD Variance-Metric Disconnect

The methods in Table 3 cut parameter-gradient variance by up to $\sim 32\times$ vs. the $(8, 1)$ baseline (with $(8, 16)$ resampling + stratification) and score-difference variance by $\sim 5.9\times$, but Fig. 26 shows no best-FID gain vs. wall-clock. We triage four candidate explanations and discuss them in numerical order below: H4 is the only one ruled out; H2 is most consistent with the data; H1 is plausible but not isolated; and H3 is untested by our experiments.

Positive control: VR machinery acts on the teacher signal. At initialization, $\boldsymbol{\mu}^{\phi}_{\text{fake}}$ equals the teacher, and the score-difference term is zero. In this regime, we observe $\sim 4\times$ variance reduction in the teacher-side SDS-like signal using the same machinery, confirming the pipeline runs end-to-end on the DMD codebase (consistent with Sec. 5.1). This does not validate VR on the trained score-difference gradient at intermediate iterations where $\boldsymbol{\mu}^{\phi}_{\text{fake}}$ has trained but is not at equilibrium; the strongest closure would be measuring score-difference VR at iter ~ 5000–$10\,000$, which we leave

[1] At initialization, the VSD loss gradient is zero as the fake-score model is initialized from the teacher weights. But we found that the teacher SDS loss exhibited an $\sim 4\times$ variance reduction.

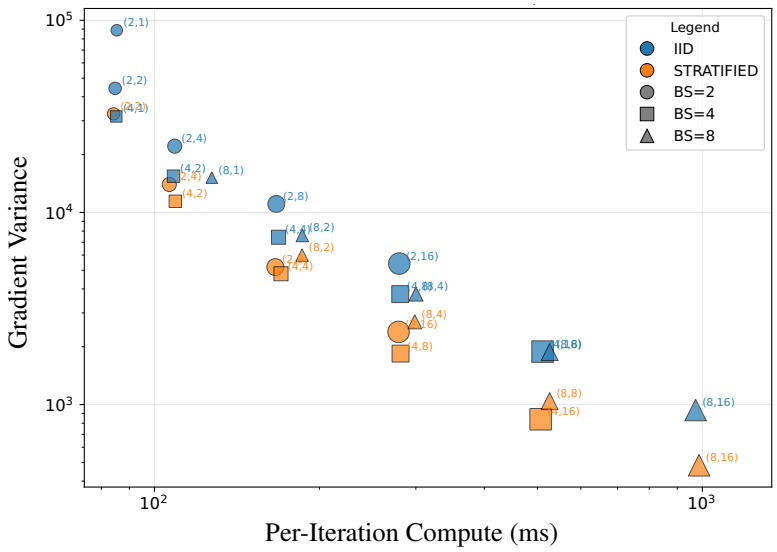


Figure 25: Quantifying variance reduction against compute cost for one-step distillation. Variance of the parameter-gradient estimator (Eq. 10) versus per-step compute in the variance runs; absolute values are on a different scale from Table 3 and the training timings of App. Sec. D.2.1, but variance ratios between configurations agree with Table 3 ($\sim$16$\times$ from $(8,1)$ to $(8,16)$ IID). Points are labeled with (R,K); marker shape encodes R (legend "BS"). Resampling and stratification both reduce variance, with combined methods achieving the lowest variance per unit compute. However, these variance reductions do not translate to improved FID scores at matched wall-clock time (Fig. 26).

to future work. So, FID flatness is a downstream insensitivity rather than a measurement artifact. Approximation error in $\boldsymbol{\mu}^{\boldsymbol{\phi}}_{\text{fake}}$ is a plausible contributor that our unbiased-estimator analysis does not isolate.

Hypothesis 1 (auxiliary objectives stabilize training): plausible, not isolated. At fixed $R=8$ and $1.55\times$ per-iteration wall-clock cost, score-difference variance shrinks $\sim 5.94\times$ while parameter-gradient variance shrinks $\sim$32$\times$ (Table 3, $(8,16)$ Strat vs $(8,1)$ IID). The two ratios need not agree, because Eq. 51 maps the score difference through $\partial G_{\boldsymbol{\theta}}/\partial\boldsymbol{\theta}$, which can amplify the reduction. Both measurements are of the distribution-matching gradient alone: the GAN-feature discriminator's generator gradient is not included, and $\mathcal{L}_{\text{denoise}}$ trains the fake score on stop-gradient generator outputs, so these data do not isolate the auxiliary objectives. They remain a plausible reason that the variance reduction does not reach FID; with the positive control above, the reduction itself is real.

Hypothesis 2 (generator-input diversity is the bottleneck): consistent with the data. Fig. 26: batch 48 with K=1 (1.0 s/iter, 48 generator-input draws) reaches the lowest final FID, while batch 8 with K=16 (0.59 s/iter, 8 draws) does not, despite 16$\times$ more $(t,\boldsymbol{\epsilon}')$ per render. The comparison is consistent with H2 but not strictly compute-matched (1.0 s vs 0.59 s/iter means the $(48,1)$ run also gets more total $\boldsymbol{\epsilon}$ draws per wall-clock); a clean (R,K) sweep at fixed wall-clock is future work. The qualitative pattern matches what reverse-KL matching predicts: per-input mass-shifting moves FID, while per-input timestep variance has a smaller marginal effect.

Hypothesis 3 (data-dependent weighting): untested by our data. The DMD weight in Eq. 51 has a data-dependent normalizer ($\|\boldsymbol{\mu}_{\text{base}}(\mathbf{z}_t,t)-\mathbf{z}\|_1$), so $q^\star \propto p\sqrt{\mathbb{E}[\|\mathbf{f}\|_2^2 \mid t]}$ is data-dependent and the SDS weight proxy of Sec. 4.1.2 need not approximate it. Our ratios use stratification with a uniform t; whether a DMD-tuned proposal helps is left for future work.

Hypothesis 4 (compute scaling): ruled out. From the timings in App. Sec. D.2.1, scaling $K=1\to16$ raises per-iter cost 0.38 s$\to$ 0.59 s (0.21 s for 15 extra denoising calls), giving $c_{\text{denoise}} \approx 0.014$ s, $c_{\text{render+encode}} \approx 0.37$ s and $c_{\text{render+encode}}/c_{\text{denoise}} \approx 26$. DMD sits in the *render-dominated* regime ($\alpha \approx 26$ on App. Fig. 21, between $\alpha=1$ and $\alpha=100$), where re-noising amortizes its largest gains. The $(8,16)$ Strat vs $(8,1)$ IID parameter-gradient variance reduction of $\sim$32$\times$ at $1.55\times$ wall-clock gives $\sim$20$\times$ variance reduction per unit wall-clock, ruling out H4 ("no compute headroom"); FID flatness must come from H1, H2, or H3.

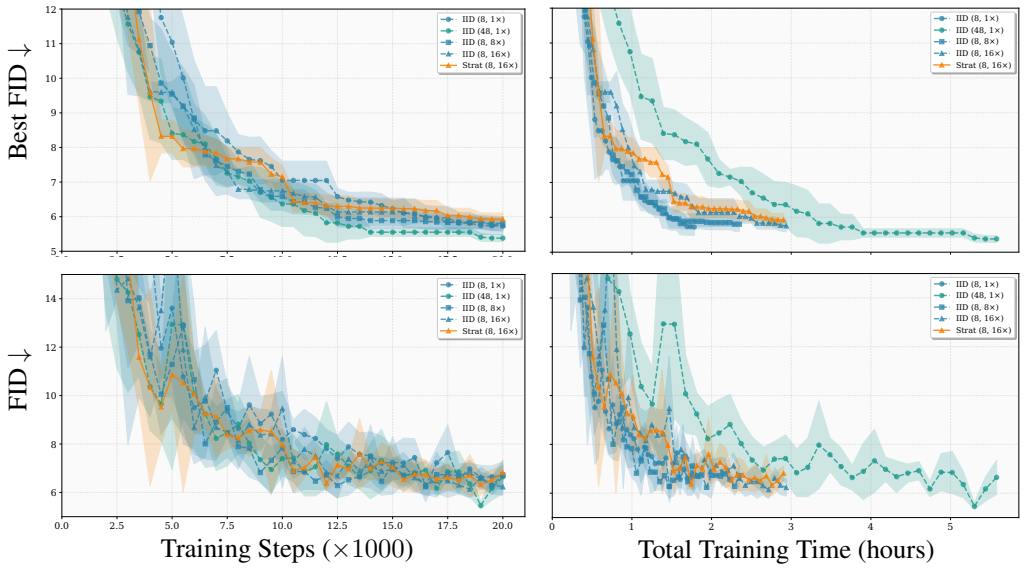


Figure 26: FID convergence during DMD training for student-step resampling. *Top:* best FID vs. steps (left) and wall-clock (right). *Bottom:* per-iteration FID vs. steps (left) and wall-clock (right). Shaded $\pm 1\sigma$ over 5 seeds. Batch 48 reaches the lowest final FID but is the slowest in wall-clock time at 1.0 s/iter. Resampling $(8, 8)$/$(8, 16)$ matches baseline $(8, 1)$ wall-clock convergence despite 3.4–16× variance reduction (Table 3); raw FID is noisier than best-so-far.

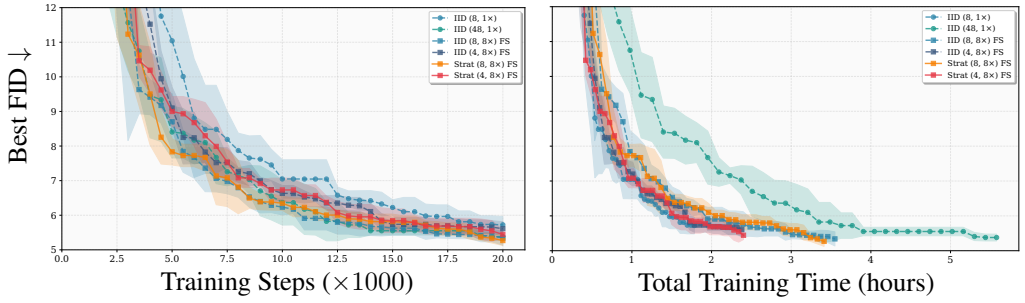


Figure 27: Best FID during training with resampling and stratification in the fake-score step. *Left:* Best FID versus training iteration. *Right:* Best FID versus wall-clock time. Shaded regions show ± 1 standard deviation across 5 seeds. Resampling and stratifying the fake-score update (runs marked FS) improves per-iteration convergence and reaches a final FID similar to batch size 48 at a lower per-iteration cost, but the batch-8 baseline remains the most compute-efficient overall.

Implications for practitioners. VR in DMD is most likely to translate to FID gains when the practitioner: (1) raises generator-input diversity ϵ jointly with timestep VR, not holding R fixed; (2) lowers the GAN discriminator and denoising auxiliary weights, isolating the distribution-matching gradient as binding; (3) builds a DMD-specific proposal that handles the data-dependent normalization in $w(t)$. Otherwise, our methods still yield stable, lower-variance gradient estimates, but should not be expected to move FID on their own.

D.3 Data Attribution

D.3.1 Details

Hyperparameter Choices: We follow MOTIVE [98] settings for `Wan2.1-T2V-1.3B` [92]. We fix random seeds for the Gaussian noise and the TRAK projector. Per-sample gradients are projected to 512 dimensions. Gradients are computed without classifier-free guidance. For influence scores, we use 11 video samples from `VIDGEN-1M` [89] with leave-one-out evaluation: each sample serves as query, and the remaining 10 form the candidate set, averaging influence scores across all 11 query-candidate assignments. Influence scores average the cosine similarities of normalized projected gradients over shared (t, ϵ) draws, as in Eq. 58.

Compute Usage: NVIDIA A100. Per-video gradients ~ 54 s (DiT forward-backward); 512-dim TRAK ~ 2 s/sample. Influence scores ~ 46 ms/pair. Sampling-strategy runs (1000 MC trials, 768 timesteps) finish in < 4 min on one GPU with cached scores.

D.3.2 Results

Our experiments demonstrate that stratified sampling substantially improves the correlation in influence rankings (Table 4). While we do not validate downstream fine-tuning performance, the metric measures agreement with the full-budget attribution, and correlation-based evaluation is standard for attribution methods [66]. At 768 timesteps, the full budget, the stratified estimate reproduces this reference (correlation 1.000; App. Fig. 28, top left), so the last row of Table 4 is not a like-for-like comparison. Direct validation via fine-tuning on variance-reduced rankings remains future work.

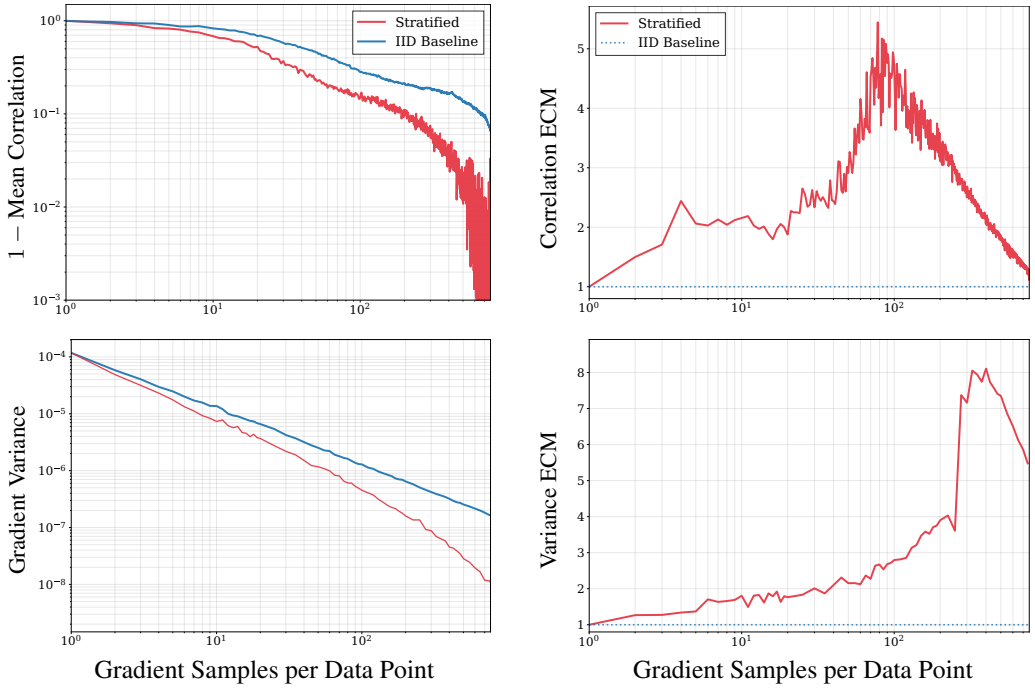


Figure 28: (Extended) Quantifying Changes in Data Attribution: Extended version of Fig. 5 showing convergence (left) and effective compute multiplier (right) for both correlation and variance metrics. *Top row:* Correlation between rankings computed with limited evaluations and full-budget rankings. Left: $1-$correlation versus number of evaluations, showing convergence to ground truth. Right: effective compute multiplier, quantifying how much more compute uniform sampling requires to match stratified sampling's correlation. *Bottom row:* Gradient variance convergence. Left: variance versus number of evaluations, showing the expected decrease with more samples. Right: variance effective compute multiplier, quantifying stratified sampling's variance reduction relative to uniform sampling. Stratified sampling consistently outperforms uniform sampling, with compute multipliers above $1.5\times$ in correlation and above $1.25\times$ in variance for 2 to 500 samples.

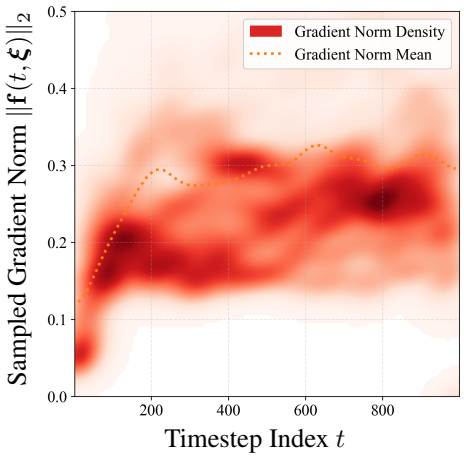


Figure 29: Is there an improvement from importance sampling for data attribution? We show the sampled gradient norm $\|\mathbf{f}(t,\boldsymbol{\xi})\|_2$, where $\mathbf{f}(t,\boldsymbol{\xi}) = \nabla_{\boldsymbol{\phi}}\ell_{\text{Diff}}(\text{Encode}(\mathbf{x}),\mathbf{c},t,\boldsymbol{\epsilon},\boldsymbol{\phi})$ and $\boldsymbol{\xi}$ denotes all other sources of randomness, including the data and conditioning $(\mathbf{x},\mathbf{c})$ and the Gaussian noise $\boldsymbol{\epsilon}$. *Takeaway:* By Eq. 23 we have $q^\star(t) \propto p(t)\sqrt{\mathbb{E}[\|\mathbf{f}(t,\boldsymbol{\xi})\|_2^2 \mid t]}$, and $\mathbb{E}[\|\mathbf{f}(t,\boldsymbol{\xi})\|_2^2 \mid t]$ is already roughly constant in t, except near 0 where the norm contribution is low, so the current sampling may not see large improvements.



Figure 30: Example Videos for Attribution: We show assorted clips from `VIDGEN-1M` [89] used for our video data attribution experiments, where the influence is being calculated for `Wan2.1-T2V-1.3B` [92].

E Related Work

We now cover related works on latent diffusion models in Sec. E.1, followed by related methods to reduce estimator variance in Sec. E.2, and finally an exploration of variance-bounded diffusion model applications of interest in Sec. E.3.

E.1 Latent Diffusion Models

Diffusion models dominate generative modeling across images, video, audio, and 3D. Early work established denoising diffusion and score-based formulations [82, 23, 84, 85]. Latent diffusion [75] compresses diffusion into a learned latent via an autoencoder, cutting compute while preserving quality, and underpins Stable Diffusion [75], Stable Audio Open [14], and video generators such as Sora [5], CogVideoX [103], and Wan [92]. Diffusion transformers (DiT) [68] and related architectures scaled these models with transformer backbones. We treat pretrained teachers as given and target gradient-estimator variance for downstream optimization and distillation.

E.2 Reducing Estimator Variance in Diffusion Models

Monte Carlo variance reduction is a standard topic in simulation and optimization. Standard techniques include importance sampling, stratified sampling, control variates, and antithetic variates [76, 65]. In diffusion model training, variance reduction primarily focuses on noise schedule design, loss reweighting, and improvements to training dynamics. Early noise schedules used simple linear or cosine interpolations [23, 63]; later work parametrizes diffusion training in terms of signal-to-noise ratio and learns noise schedules that minimize estimator variance [35] or shifts them with image resolution [24]. Loss reweighting adjusts per-timestep contributions to balance gradient magnitudes and prioritize perceptually important noise levels [78, 29, 9, 20]. Complementary training-side techniques randomize the supervised time interval to avoid a fixed low-noise truncation [32] and stabilize training dynamics through normalization and architecture choices [30].

Downstream users of frozen diffusion teachers typically inherit the teacher's training schedule without revisiting the timestep distribution. Recent work has begun exploring variance reduction here: Variational Score Distillation [95] introduces a particle-based variational objective that, as later analyses note [e.g., 94], lowers SDS gradient noise at extra memory and compute. Concurrent few-step distillation work, such as moment matching [79], does not measure estimator variance or use unbiased IS. We give a principled compute-aware framework and show that simple unbiased techniques (IS, stratification, reuse) deliver large variance reduction across diffusion-teacher tasks while preserving the objective.

E.3 Diffusion Model Applications

We now cover diffusion tasks related to our setup, including optimizing parametrized models with diffusion priors (Sec. E.3.1), single-step diffusion model distillation (Sec. E.3.2), data attribution (Sec. E.3.3), and beyond (Sec. E.3.4).

E.3.1 Diffusion Priors for Optimization

Diffusion models are widely used as frozen teachers that provide gradients for optimizing parametrized generators, beginning with diffusion-teacher-guided text-to-3D optimization and its refinements for fidelity, stability, and efficiency [69, 38, 7, 113, 93, 95, 53, 51, 94, 106, 55, 81]. This teacher-gradient template has since been extended to amortized text-to-3D [48, 100], dynamic 4D scenes [41, 72, 2], physics and simulation parameters [42, 108], and materials and textures [13], alongside other routes to 3D content such as language-model mesh generation [96] and multimodal generative 3D [99]. It also appears outside vision, for example in diffusion-guided audio optimization [74]. We keep these SDS-style objectives fixed and reduce the variance of the Monte Carlo gradients they share, using unbiased, compute-aware sampling.

E.3.2 Single-Step Distillation

Single-step and few-step distillation compress a pretrained diffusion teacher into a fast student sampler by training the student to match the teacher's induced sample distribution or score field. The training objective is an expectation over diffusion timesteps and noise, so optimization relies on Monte Carlo gradient estimators whose variance and cost depend on the timestep and noise-sampling designs. A prominent family matches student and teacher distributions via score-difference objectives, including distribution-matching distillation and follow-ups [105, 104, 52, 111, 62, 112]; related methods match moments [79] or use EM-based maximum likelihood [101]. A complementary adversarial branch combines distillation with a discriminator on student outputs [80]. Variants include multi-student distillation [88] and flow-alignment formulations that bridge diffusion and flow-based distillation views [77]. Consistency models [87, 83] provide an alternative distillation path by enforcing self-consistency along ODE trajectories. Our variance-reduction methods apply naturally to these score-difference and consistency objectives, reducing the cost of gradient estimation without modifying the distillation loss.

E.3.3 Data Attribution

Gradient-based data attribution estimates how training examples affect model outputs via influence functions [36] or approximations such as TracIn [70], TRAK [66], and approximate-unrolled-differentiation methods [1]. Recent work adapts these to diffusion and LoRA-tuned models, proposing attribution methods for image and video generation [15, 110, 4, 40, 58, 37]. MOTIVE [98] specializes in attribution for motion in video diffusion models. These estimators rely on Monte

Carlo averages over diffusion timesteps and noise, so reducing variance improves attribution quality at fixed compute. Our variance-reduction techniques apply naturally and yield more stable influence rankings.

E.3.4 Other Diffusion-Guided Tasks

Beyond the three applications we evaluate, diffusion-teacher gradients appear in many other settings where variance reduction could provide similar benefits. Diffusion guidance has been applied to image editing and inpainting [57, 50], controllable generation via spatial or semantic constraints [107, 60], and inverse problems such as super-resolution and deblurring [86, 31]. In robotics and reinforcement learning, diffusion models serve as planners and policies [27, 8]. Our variance-accounting framework (Sec. 4.2) and drop-in variance-reduction techniques (Sec. 4.1) extend naturally to these tasks whenever the computational bottleneck lies in Monte Carlo estimation over timesteps and noise.

E.4 Comparison to Variational Score Distillation (VSD)

VSD [95] keeps the frozen teacher and replaces SDS's injected-noise term with a LoRA-adapted score of the rendered images, turning SDS into a variational objective that jointly optimizes the scene and that score; the learned score gives more consistent gradients than the injected noise.

Relationship to our work. Our re-noising, importance-sampling, and stratification construction can be applied to the VSD gradient with an appropriate timestep proposal, though we have not tested this combination. A head-to-head comparison of SDS with our estimator against VSD would confound estimator changes with VSD's learned-score objective and the additional joint-tuning budget [54, 67, 45].

Other related methods: Beyond VSD, many techniques improve SDS-based text-to-3D generation through domain-specific modifications: other 3D representations such as Instant-NGP hash grids [61], amortization across prompts [48, 100], multimodal 3D generation [99], trajectory reparametrizations [51], regularization terms, camera pose scheduling, and guidance-scale annealing. Language-model mesh generation [96] is a separate route to 3D that does not use SDS. These approaches are orthogonal to ours in two ways. First, they are specific to 3D optimization and do not generalize to other diffusion-teacher applications (distillation, data attribution), whereas our methods apply to any Monte Carlo gradient from a frozen teacher. Second, many approaches introduce bias or alter the target objective (e.g., via regularization or modified guidance), whereas we focus exclusively on changes to unbiased estimators that preserve the original objective. These 3D-specific techniques could be combined with our variance-reduction strategies to further improve text-to-3D tasks.

E.5 Comparison to SteinDreamer

SteinDreamer [94] reduces SDS variance via a Stein-identity control variate, instantiated with a frozen MiDaS depth (or normal) estimator and a learnable scaling $\boldsymbol{\mu}$ for the Stein term. The control variate is zero-mean in expectation and operates on the noise-direction estimator at fixed t; our methods operate at the timestep-sampling and compute-amortization levels. The two attack different randomness axes of the same estimator: SteinDreamer targets the noise-direction component, while CARV targets the joint $(t, \boldsymbol{\epsilon})$ allocation and the rendering-vs-denoising compute split.

Relationship to our work. Our hierarchical Monte Carlo construction (Algorithm 1) can be applied to SteinDreamer's control-variate-corrected estimator, though the combined gains remain untested. A head-to-head substitution would confound noise-direction control variates with changes to timestep allocation and compute reuse, whereas a stacked ablation would isolate their contributions. SteinDreamer's implementation has not been released at the time of writing (`https://github.com/Vita-Group/SteinDreamer` contains a README, a teaser, and project-page assets).

Why CLIP rather than FID for our setting. FID requires a reference distribution. Single-prompt text-to-3D produces one scene per prompt; rendered views of that scene are not drawn from a meaningful distribution, so FID measures distance between an ad hoc reference set and a single object's renders rather than fidelity-and-diversity. We therefore follow the standard SDS literature in using CLIP score for prompt alignment and qualitative visualizations for fidelity (Fig. 12, Fig. 7); Stein-

Dreamer reports CLIP distance for their convergence study while also reporting FID in their Tab. 1, consistent with this choice for convergence-curve comparisons.

Independent/concurrent variance-reduction directions for SDS. Beyond control variates and our timestep-allocation approach, several recent directions change the SDS objective or its gradient: particle-based learned-score reformulations [95], multi-step trajectory reparametrizations [51], and mean-shift formulations of the distillation gradient [90]. Others change the training setup: multi-student distillation [88] and amortization across prompts [48, 100]. Our work is the only direction we are aware of that preserves the SDS objective *and* provides compute-aware, estimator-level accounting. Amortized text-to-3D [48, 100] also trains with SDS-style losses, so it can use the same estimator.

E.6 Novelty vs. Prior Work

Importance sampling, stratified sampling, and compute reuse are standard Monte Carlo variance-reduction techniques; adapting them to frozen diffusion teachers requires care:

What is standard: The frameworks for importance sampling (Sec. 3.2.1) and stratified sampling (Sec. 3.2.2) are classical, as are the two-level variance decomposition (Eq. 10), the cost-optimal allocation between levels, the composition of stratification with subsampling, and stratified inverse-transform sampling [10, 91, 65]. Reusing expensive computation while resampling cheap randomness is a general Monte Carlo principle.

What is novel:

1. **The identification:** frozen-teacher gradient estimation is an instance of the classical two-level sampling problem with asymmetric per-level costs, with renders as primary sampling units and diffusion draws as subunits (Sec. 4.1). The items below operationalize it.
2. **Weight-based importance sampling proxy:** We identify that the explicit weight functions $w(t)$ already present in diffusion objectives (e.g., $w_{\text{SDS}}(t)$ in SDS) can serve as effective proxies for the variance-optimal proposal without requiring gradient-norm estimation. Prior frozen-teacher work in the cited SDS, DMD, and attribution lines typically uses uniform or loss-based timestep sampling; $w_{\text{SDS}}(t)$ itself, motivated by its empirical tracking of parameter gradient norms (Fig. 1), is the proxy we add.
3. **Stratified inverse-CDF timestep allocation (an adaptation):** Stratification and inverse-transform sampling are each standard, and the equal-mass quantile construction is the standard way to combine them. What we add is its application to frozen-teacher timestep allocation, composed with per-render stratification; we are not aware of prior frozen-teacher pipelines applying it to timestep allocation.
4. **Compute-aware variance accounting:** Our framework (Sec. 4.2) separates rendering/encoding from denoising costs and measures variance per unit compute via effective compute multipliers (ECM) and relative efficiency (RE). Prior work in this setting counts gradient evaluations or wall-clock time without decomposing costs, masking the asymmetric-cost regime in which reuse pays off.
5. **Systematic empirical measurement of parameter-gradient variance:** Prior SDS work typically reports latent-space update variance or scalar losses. We measure *parameter-gradient estimator variance* (Eq. 10) across three applications, and use the resulting measurements to expose a regime (DMD, Sec. 5.2) where variance reduction does not improve downstream metrics.
6. **Timestep stratification in this context:** To our knowledge, no published work in the cited frozen-teacher SDS, DMD, or attribution lines applies stratified sampling over diffusion timesteps. Timestep stratification has been used in diffusion model *training* (e.g., the low-discrepancy timestep sampler of VDM [35, App. I.1]), but not in the downstream frozen-teacher pipelines we evaluate, where the computational hierarchy and gradient structure differ.

Other estimator-level baselines. Adaptive importance sampling over t is bounded by the gradient-norm oracle of App. Sec. D.1.8, and App. Sec. D.1.6 computes a Sinkhorn-optimal two-sample design; IW+Strat reaches $\sim$91% of its efficiency. Both serve as reference points rather than competing heuristics. Antithetic sampling and control variates act on the innermost $\boldsymbol{\epsilon}$ level, conditional on render and timestep, whereas Eq. 10 covers the two levels that carry the cost asymmetry; applying the same accounting to a third level is a natural extension.

Related variance reduction in diffusion and graphics. The most closely related prior work is on noise-schedule design for diffusion model training. Variational diffusion training [35] parametrizes diffusion training in terms of SNR and learns a noise schedule that minimizes the variance of the training-objective estimator, and Min-SNR weighting [20] reweights training losses by signal-to-noise ratio to balance contributions across t. Both, however, target training the teacher itself, where the gradient is with respect to denoiser parameters ϕ; we focus on downstream use of frozen teachers, where gradients flow through generators, encoders, or renderers, and the computational hierarchy is fundamentally different. Methods like Variational Score Distillation [95] reduce SDS variance by changing the objective (replacing the injected noise with a learned score while keeping the frozen teacher); we preserve the objective and work at the estimator level. Compute reuse has a long pedigree in graphics: spatiotemporal resampled importance sampling (ReSTIR) [3] and its formal generalized basis (GRIS) [39] amortize expensive scene queries across reused samples, the same principle our hierarchical estimator exploits in the diffusion-teacher setting.

E.7 Adjacent Tools and Methods

Several adjacent research threads outside the immediate diffusion-teacher setting provide tools and theory that complement our framework.

Bilevel and nested optimization. DMD's learned-score formulation and VSD (App. Sec. D.2.3, App. Sec. E.4) are bilevel (inner: auxiliary score; outer: generator/scene). Gradient-based bilevel optimization [54, 67, 43, 45] provides implicit differentiation, structured best responses, and scalable nested optimization, useful when gradient-variance reduction is muted by auxiliary stabilizers. Hyperparameter optimization that uses checkpointed model weights [56] could speed up tuning of the extra hyperparameters (LoRA rank, β, and the q shape). The optimization-in-games view [49, 47] captures DMD's alternating-update dynamics.

Structured Jacobians and architecture-aware tooling. Our analysis of the SDS gradient $w_{\text{SDS}}\,\mathbf{J}_{\boldsymbol{\theta}}^{\top}\mathbf{r}$ depends on the structure of $\mathbf{J}_{\boldsymbol{\theta}}$ through encoder-renderer chains. Structured-Jacobian networks [25, 73] provide tools for analyzing or learning such Jacobians directly, and AutoML task selection [46], which picks representative tasks for evaluating system changes, could choose the prompts and modalities on which to test estimator settings.

Distillation and 3D-generation pipelines that benefit from estimator-level VR. Multi-student distillation [88] trains several specialized one-step students with distillation losses that are Monte Carlo estimates, so the same estimators apply. In 3D generation, amortized text-to-3D [48, 100] and multimodal 3D generation [99] train with SDS-style losses, where our IW + Strat + Reuse estimator applies unchanged; language-model mesh generators [96] take a route that does not use SDS. Score distillation for audio [74] is a direct application in a modality we do not evaluate, and our video attribution experiments build on MOTIVE [98].

F Additional Discussion

F.1 Limitations

Variance reduction does not always translate to downstream gains. In DMD (Sec. 5.2, App. Sec. D.2), resampling reduces gradient variance by 3.4–16$\times$ without improving FID at matched wall-clock time. Generator-input diversity is most consistent with the data, while auxiliary-objective effects are plausible but not isolated (App. Sec. D.2.3). We treat this as a deliberate negative result that maps the boundary of applicability rather than a failure of the method.

IS proxy depends on gradient structure. Our weight-based IS proposal $q(t) \propto p(t)w_{\text{SDS}}(t)$ assumes the explicit weight is correlated with the per-timestep gradient norm. For tasks where gradient contributions are roughly uniform across timesteps (e.g., data attribution, App. Fig. 29), IS provides minimal benefit, and stratification alone suffices. When $w(t)$ is unavailable or miscalibrated, adaptive binned proposals can substitute but require periodic recomputation. We did not test SDS variants that change the form of the per-timestep gradient, such as VSD [95].

Compute reuse depends on cost structure. Re-noising (Sec. 4.1.1) is most effective when upstream costs (rendering, encoding, generator forwards) exceed denoising and within-render variance is timestep/noise-driven. When input variability dominates or $c_{\text{render+encode}}/c_{\text{denoise}}$ is small, marginal gains shrink (App. Fig. 21).

Variance-measurement overhead. The framework (Sec. 4.2) runs each estimator to convergence, an upfront cost for method comparison. Its cost grows with the number of configurations compared and the precision required, so practitioners optimizing wall-clock time may prefer to validate on downstream metrics directly and use our framework as design guidance.

Frozen-teacher assumption. We assume the teacher is frozen downstream. Settings where the teacher is fine-tuned or co-adapted (e.g., joint distillation) would require accounting for teacher-parameter drift and its interaction with timestep-sampling strategies.

SDS evaluation stack vintage. Our SDS experiments use Stable Diffusion 2.1 as the teacher [75], NeRF / Instant-NGP [61] as the 3D representation, and the threestudio [18] framework. These were SOTA at the time of the experimental sweep, but more recent stacks (FLUX or SDXL teachers, MVDream-style multi-view conditioning [81], 3D Gaussian Splatting renderers) would change the absolute compute budget and the precise $c_{\text{render+encode}}/c_{\text{denoise}}$ ratio. Our framework is teacher- and renderer-agnostic by construction (Sec. 4.2), so the qualitative wins should transfer; precise quantitative ECMs in those stacks would require re-running the variance sweep with the new pipeline.

F.2 Future Directions

Several promising directions extend our framework beyond the tasks and methods explored here.

Extension to diverse diffusion-guided tasks. Our evaluation covers text-to-3D optimization, one-step distillation, and data attribution, but the teacher-gradient pattern appears across many other settings. Natural extensions include 4D scene optimization, physics-informed diffusion guidance, material and texture synthesis, audio generation, and video editing pipelines. Our framework provides a systematic way to quantify variance-cost trade-offs in these domains, so practitioners can identify efficient sampling strategies without exhaustive tuning.

Alternative prediction parameterizations and teacher architectures. Our framework applies unchanged across noise-prediction, $\mathbf{x}$-prediction, and v-prediction parameterizations: each corresponds to a different per-timestep weight $w(t)$, and the IS proxy $q \propto p\,w$ uses whichever weight the teacher exposes. Stratification and compute reuse are parameterization-agnostic. Our data-attribution experiments (Sec. 5.3) already cover the flow-matching case (Wan2.1), where the velocity field carries an analogous time-dependent weight. Investigating how optimal (R, K) shifts across these parameterizations on a fixed task is a natural next step.

Biased variance reduction and trade-offs. Our methods preserve unbiasedness. Many practical pipelines introduce bias: they clip gradients, truncate guidance, or clamp timesteps to improve stability or perceptual quality. We leave two questions open. How do stratification and importance sampling interact with these biased techniques? Can we trade a slight bias for further variance reduction? Control variates that use inexpensive auxiliary estimates could complement our approach and reduce variance further.

Non-frozen teachers and co-adaptation. Our framework assumes a frozen teacher, but some pipelines fine-tune or distill the teacher alongside downstream optimization. In such settings, the optimal timestep distribution may shift as the teacher adapts, and variance-reduction strategies may need to account for the joint dynamics of the teacher and downstream parameters. Extending our methods to these coupled settings could improve both training efficiency and final performance.

Connecting estimator design to downstream performance. While variance reduction improves gradient quality, the relationship between gradient variance and downstream metrics (e.g., CLIP score, FID, influence ranking correlation) is task-dependent and not fully understood. Developing theory or empirical principles that predict when variance reduction will translate into metric improvements and which variance sources matter most for a given task would help practitioners allocate compute more effectively and design better estimators.

F.3 Detailed Practitioner Guidance

We provide a decision tree to help practitioners choose variance-reduction methods based on their application's computational structure and objectives. Sec. 4.2 gives the procedure in brief; this section adds thresholds and the full decision tree.

Step 1: Assess cost structure

- Measure rendering/encoding cost $c_{\text{render+encode}}$ vs. denoising cost c_{denoise}.
- As a rule of thumb, $c_{\text{render+encode}} > 10 \times c_{\text{denoise}}$ motivates testing compute reuse; the cost ratio alone is not sufficient.
- If $c_{\text{render+encode}} \approx c_{\text{denoise}}$: stratification may be more effective (data attribution).
- If $c_{\text{render+encode}} < c_{\text{denoise}}$: focus on importance sampling and stratification with small K.
- To set K, estimate σ_A^2 and σ_W^2 of Eq. 10 from a short pilot using Welford's algorithm with cached-render re-noising (Sec. 4.2, App. Sec. C.1.3), time $c_{\text{render+encode}}$ and c_{denoise}, and use $K^\star \approx \sqrt{(\sigma_W^2/\sigma_A^2)(c_{\text{render+encode}}/c_{\text{denoise}})}$ (Sec. 4.1.1). For IID subunits, compare the feasible floor and ceiling (at least 1) using Eq. 11; $K = 2$ beats $K = 1$ exactly when $K^\star > \sqrt{2}$. With stratification, validate the final allocation by its measured ECM.

Step 2: Identify gradient structure

- If a known timestep weight dominates how gradient norms vary with t (e.g., $w_{\text{SDS}}(t)$ in SDS): use weight-based importance sampling with the likelihood-ratio correction.
- If gradient norms vary significantly across timesteps: use adaptive importance sampling.
- If gradient norms are approximately constant: skip importance sampling.

Step 3: Check for auxiliary objectives

- If training uses strong auxiliary losses (DMD): variance reduction may not improve final metrics, but can stabilize training.
- If using only Monte Carlo gradients (SDS, data attribution): variance reduction will likely help.

Step 4: Choose stratification design

- Per-render stratification: Use when rendering dominates, and you sample multiple timesteps per render (SDS default).
- Global stratification: Use when encoding is moderate relative to denoising (data attribution).
- Number of strata: $B = K$ per render, or $B = R \times K$ for global stratification; use equal-mass quantile strata for a non-uniform proposal.

Step 5: Combine methods

- Importance + stratification: Use stratified inverse-CDF.
- Stratification + reuse: Compatible, combine for complementary benefits.
- All three: Best results in SDS experiments.

Expected effective compute multipliers (envelope across our experiments):

- Importance sampling alone: 1.05–$1.24\times$ (RE vs uniform; Table 2), up to $1.40\times$ at $\omega = 25$ (Table 6)
- Stratification plus reuse: ~ 1.0–$3.0\times$ ECM in SDS (Table 1); stratification's isolated gain is reported as matched-configuration RE in Table 2
- Compute reuse alone (high $c_{\text{render+encode}}/c_{\text{denoise}}$): ~ 1.6–$2.6\times$ on SDS (Table 1)
- Combined IW+Strat+Reuse: ~ 2–$3.3\times$ (SDS averages over R, peaking at $K = 8$; Table 1)

G Glossary and Notation

Table 8: Glossary and notation (Part I: Fundamentals)

Acronyms and abbreviations	
IID	Independent and identically distributed
IS	Importance sampling
MC	Monte Carlo
SDS	Score Distillation Sampling
DMD	Distribution Matching Distillation
VSD	Variational Score Distillation [95]
CFG	Classifier-free guidance
ECM	Effective compute multiplier (Sec. 4.2)
RE	Relative efficiency (Sec. 4.2)
HT	Horvitz-Thompson (estimator; App. Sec. D.1.6)
FID	Fréchet inception distance
NeRF	Neural radiance field
VAE	Variational autoencoder
DiT	Diffusion transformer [68]
Core mathematical notation	
$\mathbb{E}$	Expectation operator
D_{KL}	Kullback-Leibler divergence
$\mathcal{N}$	Gaussian/normal distribution
$\mathcal{U}$	Uniform distribution
$\mathcal{N}(\mathbf{0}, \mathbf{I})$	Standard normal $\mathcal{N}(\mathbf{0}, \mathbf{I})$
$\mathbf{I}$	Identity matrix
d	Differential (use as $\mathrm{d}x/\mathrm{d}y$)
$:=$	Defined to be equal to
$\mathbf{J}$	Jacobian matrix
Diffusion model time and schedules	
t	Continuous time / noise level in $[0, 1]$
T	Maximum time / total timesteps
λ	Log signal-to-noise ratio
f_λ	Schedule mapping $t \to \lambda$
$\lambda_{\min}, \lambda_{\max}$	Schedule endpoints
α_t, σ_t	Signal and noise coefficients at time t

Table 9: Glossary and notation (Part II: Data and Models)

Data and latent representations	
$\mathbf{x}$	Clean data/image; generic upstream draw in Sec. 4.1
$\mathbf{z}$	Latent-space representation (VAE-encoded; generator output in DMD)
$\mathbf{z}_t$	Forward-noised latent: $\mathbf{z}_t = \alpha_t \mathbf{z} + \sigma_t \boldsymbol{\epsilon}$
$\boldsymbol{\epsilon}$	Gaussian noise input $\sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
$\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\phi}}$	Predicted noise from denoiser network
$\mathbf{r}$	Denoising residual $\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\phi}} - \boldsymbol{\epsilon}$
Encode	Encoder to latent space (e.g., VAE encoder)
Model components and parameters	
$\boldsymbol{\theta}$	Generator or renderer parameters (e.g., NeRF weights)
$\boldsymbol{\phi}$	Denoiser or score network parameters
$G_{\boldsymbol{\theta}}$	One-step generator mapping noise to samples
g	Encoded rendering map Encode $\circ\, g'$
g'	Differentiable renderer before encoding (e.g., NeRF)
$\mathbf{c}$	Conditioning signal (e.g., text prompt)
$\boldsymbol{\pi}$	Camera or rendering condition sample
Weighting and guidance	
w	Timestep weighting function
w_{SDS}	SDS-specific timestep weight
ω	Guidance scale for CFG

Table 10: Glossary and notation (Part III: Losses and Distributions)

Loss functions and objectives	
$\mathcal{L}$	Generic loss function
ℓ	Per-sample cost function
$\mathcal{L}_{\text{Diff}}$	Diffusion training loss
$\mathcal{L}_{\text{wDiff}}$	Weighted diffusion objective
ℓ_{Diff}	Per-sample diffusion cost
$\mathcal{L}_{\text{SDS}}$	SDS objective (parameter-space)
$\mathcal{L}_{\text{denoise}}$	Auxiliary denoising loss for fake model
$\mathcal{L}_{\text{reg}}$	Regression loss
$\mathbf{u}$	Update vector (gradient-style)
$\mathbf{u}_{\text{SDS}}$	SDS update direction
$\hat{\mathbf{u}}_{\text{SDS}}$	Stochastic estimator of $\mathbf{u}_{\text{SDS}}$
Distributions and density models	
p_{real}	Real data distribution
p_{fake}	Generated/fake distribution
q	Importance sampling proposal distribution
$\tilde{w}$	Importance weight $p(t)/q(t)$
$\mathbf{s}_{\text{real}}, \mathbf{s}_{\text{fake}}$	Real and fake score functions
$\boldsymbol{\mu}$	Target mean (the expectation being estimated)
$\hat{\boldsymbol{\mu}}$	Estimated mean (the realized estimator)
$\hat{\boldsymbol{\mu}}_{\text{GT}}$	High-sample ground-truth mean reference (App. Sec. C.1)
$\boldsymbol{\mu}_{\mathbf{y}}$	Pair-prob target sum $\sum_i \mathbf{y}_i$ (App. Sec. D.1.6)
$\boldsymbol{\mu}_{\text{base}}$	Denoised mean from base diffusion model
$\boldsymbol{\mu}^{\boldsymbol{\phi}}_{\text{fake}}$	Denoised mean from learned fake model
base, fake, real	Labels `base`, `fake`, `real`

Table 11: Glossary and notation (Part IV: Sampling and Variance Reduction)

Monte Carlo sampling	
N	Number of samples (also N_{GT} for the ground-truth reference)
n	Sample index $\in \{1, \dots, N\}$
k, K	Sampler index and step count (diffusion sampling only)
$\mathbf{F}$	Test function in expectations: $\mathbf{F}(t) = \mathbb{E}_{\boldsymbol{\xi}}[\mathbf{f}(t, \boldsymbol{\xi}) \mid t]$
$\boldsymbol{\xi}$	Generic randomness variable (data, conditioning, noise)
$\boldsymbol{\xi}_b^{(r)}$	Uniform $(0, 1)$ jitter for stratum b of render r
Stratified sampling	
B	Number of strata for stratified sampling
$\mathcal{S}_b$	The b-th stratum
b	Stratum index
M_b	Number of samples in stratum b
m	Index within a stratum
u	Quantile level for inverse-CDF sampling
Compute reuse and efficiency	
R	Number of distinct renders per step
K	Number of re-noisings per render (compute reuse)
r	Render index
k	Re-noise index
$c_{\mathrm{render+encode}}$	Cost per primary unit (render or generator pass and encode; Sec. 4.1)
c_{denoise}	Cost per subunit (one noising and denoising; Sec. 4.1)
$\mathcal{B}$	Compute budget
c	Generic per-operation cost
$\mathbf{f}$	Per-$(t, \boldsymbol{\xi})$ contribution vector (e.g., $w_{\mathrm{SDS}}\,\mathbf{r}$ in latent space; Sec. 4.1)
$\mathbf{h}$	Full gradient contribution, including the Jacobian
$\hat{\mathbf{h}}^{(r)}$	Mean of K contributions for primary unit r
σ_A^2	Across-unit variance of conditional means (Eq. 10)
σ_W^2	Mean within-unit variance (Eq. 10)
$K^\star$	Continuous cost-optimal allocation for IID subunits

Table 12: Glossary and notation (Part V: Attribution and Auxiliary)

Data attribution	
I	Influence score between training and query examples
$\mathbf{g}$	Per-example gradient for attribution
query	Query example label
n	Training example index
$\mathcal{T}$	Set of shared $(t, \boldsymbol{\epsilon})$ samples
$\mathcal{D}$	Training dataset
$\mathbf{H}$	Hessian matrix
Auxiliary notation	
C, S	Number of channels and spatial locations

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and Sec. 5 state our headline claims (a compute-aware variance-accounting framework and three unbiased drop-in estimators yielding 2 to 3× effective compute multipliers in text-to-3D optimization and attribution; an order-of-magnitude gradient variance reduction in DMD without downstream FID improvement). These claims match the experimental results in Sec. 5 and the detailed analyses in App. Sec. D.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: App. Sec. F.1 (Limitations) discusses the DMD negative result, IS-proxy preconditions, compute-reuse cost-ratio dependence, the variance-measurement overhead, and the frozen-teacher assumption. The body's "When Variance Reduction Helps" paragraph (Discussion) summarizes the boundary of applicability, and App. Sec. F.3 provides a step-by-step decision tree for method selection.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper makes no formal theorem-style claims requiring proofs. Estimator unbiasedness for our three strategies follows directly from their construction in Sec. 4.1 (derived for stratified IS in App. Sec. B.2.1), and the classical two-level decomposition and $K^\star$ are derived with their IID-subunit assumption in App. Sec. B.2.2; the asymptotic-rate aside in App. Sec. D.1.6 and the variance-minimizing proposal in App. Sec. B.2.1 cite standard Monte Carlo references rather than introducing new theory.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Sec. 4.1 specifies all three estimators with explicit equations directly implementable on top of standard frameworks. App. Sec. D provides hyperparameters, model checkpoints, dataset splits, training schedules, prompt lists, and variance-measurement protocols for each task. The Reproducibility paragraph at the start of the appendix summarizes the steps taken.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: Code is not released with the paper. The paper builds on established open-source codebases (threestudio for SDS, the FastGen reference implementation for DMD, MOTIVE / DiffSynth-Studio for video data attribution) and provides complete equations, pseudocode (Algorithm 1), and hyperparameters (Sec. 4.1, App. Sec. D) sufficient for re-implementation as a small modification to the per-step sampling and re-noising logic of these pipelines.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: All experiment-specific hyperparameters (model versions, optimizer settings, batch sizes, render and re-noising counts R and K, classifier-free-guidance scales for SDS, evaluation prompts and seeds) are reported in App. Sec. D.1 (SDS), App. Sec. D.2 (DMD), and App. Sec. D.3 (data attribution).

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: CLIP-score curves in Fig. 6 are averaged over 30 prompts $\times$ 3 seeds with ± 1 standard-deviation shading. Each configuration in the main SDS variance sweep (App. Sec. D.1.1) is averaged over 60 independent measurements (5 prompts $\times$ 3 training seeds $\times$ 4 Monte Carlo seeds). DMD FID curves (App. Sec. D.2) average 5 seeds per configuration. The error-bar variant of Fig. 4 (App. Fig. 10) shows the seed-dispersion of the variance estimator itself.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: App. Sec. D.1 reports SDS compute (NVIDIA A100-80GB GPUs, $\sim$ 3,400 GPU-hours total across checkpoint training, main variance experiments, and the oracle ablation). App. Sec. D.2 reports DMD per-iteration cost in seconds for each batch configuration on NVIDIA A100. App. Sec. D.3 reports per-sample timings for gradient computation, TRAK projection, and influence scoring on A100.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: We have read and conform to the NeurIPS Code of Ethics. The paper does not introduce new generative capabilities, does not collect new scraped or crowdsourced data, and uses only existing pretrained models and public datasets per their original licenses.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: App. Sec. A (Broader Impacts) discusses the positive impact of reduced compute and energy use for diffusion-guided pipelines and the indirect potential for misuse via cheaper synthetic-media generation. We do not introduce new generative capabilities, datasets, or model weights in this paper.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: We do not release new pretrained models, datasets, or other high-risk artifacts. We use existing open-source pretrained models (Stable Diffusion 2.1, DiT-XL/2, Wan2.1-T2V-1.3B) per their original licenses; safeguards remain the responsibility of the original model providers.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Models: Stable Diffusion 2.1 base (CreativeML Open RAIL++-M), DiT-XL/2 (CC BY-NC 4.0) and Wan2.1-T2V-1.3B (Apache 2.0). Datasets: ImageNet (ImageNet terms of access, non-commercial research) and VIDGEN-1M (Apache 2.0). Code: threestudio, FastGen and DiffSynth-Studio (Apache 2.0), and the MOTIVE attribution pipeline [98], whose authors include two of ours. All are cited in App. Sec. D and used under these terms.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: We do not release new datasets, pretrained models, or other assets in this paper. The proposed estimator equations and algorithms are described in full in Sec. 4.1 and App. Sec. C.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing experiments or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve research with human subjects, so IRB review is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: We used a large language model only as a writing and engineering assistant (clarity edits, prose reorganization, debugging plotting and analysis scripts), not as a component of the core methodology, experimental design, or scientific claims. Per the NeurIPS LLM policy, declaration is not required for writing-only usage; we include the "LLM Usage" paragraph in App. Sec. A as a courtesy disclosure.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.