Contents

Impact Statement

The proposed method facilitates weighted sampling for a wide range of existing pretrained score-based models and weight functions. We explicitly clarify that *the primary objective of this study is to advance methodological rigor and the field of Machine Learning*; this work neither intends to advocate nor promote any particular social perspective through the experimental results presented. We do acknowledge the potential for this approach to be misused in reinforcing biased sampling practices, though, and thus strongly encourage careful consideration of the implications associated with its application.

A. Related Work

Importance sampling. Importance sampling [45] and weighted sampling play a key role in various foundational tasks such as variance reduction, data augmentation, root cause analysis, and reliability assessment [46–48]. However, directly estimating the weighted PDF q via Monte Carlo methods using samples drawn from p is challenging, primarily due to instability in accurately computing the normalization constant.

To address this, conventional techniques often leverage Cross-Entropy methods [49, 50], which iteratively optimize a parameterized PDF to minimize the Kullback-Leibler (KL) divergence from the target weighted sampling PDF. However, for complex, high-dimensional datasets—such as natural images with intractable distributions—the selection of a suitable class of PDFs becomes highly challenging. Recent developments in generative modeling offer a solution via bijective mappings, where a tractable initial distribution is progressively transformed through the change-of-variables formula, resulting in a valid end distribution known as normalizing flow [51, 52]. Normalizing flow-based methods have been applied to weighted sampling, employing efficient invertible functions as in [53] or interpretable, shape-constrained networks [54].

Nevertheless, Cross-Entropy or normalizing flow-based methods often exhibit limited representational flexibility and can struggle with mapping selection [55, 56]. Additionally, training separate generative models for distinct weight functions is computationally prohibitive, particularly for applications requiring diverse importance criteria or multiple estimations across varied weighting functions.

Score-based generative models. Recent advances in generative modeling have focused on score-based models, which learn the score function of a target distribution instead of estimating its PDF directly [3, 57, 58]. This approach enables efficient sampling via stochastic differential equations or iterative methods [1–3], achieving notable success in high-resolution image generation [59]. By eliminating the need for shape-constrained bijective mappings required by normalizing flow-based models, score-based models allow for greater design flexibility. Direct applications of existing score-based methods to weighted sampling, however, present challenges, as they require samples from the target distribution, limiting adaptability. Also, training distinct score-based generative models for various weight functions is inefficient and impractical.

To address these challenges, recent techniques [19] model weighted sampling as a diffusion process and approximate its score function using the original PDF's score combined with the weight function, rather than learning the weighted-sampling score directly. This avoids training separate generative models for different importance weight functions while still fully exploiting the expressive power of score-based models.

Connections and differences between conditional and weighted sampling. Advances in generative modeling have enabled conditional generation across various modalities, including text-to-image synthesis [34, 60–64], image-to-image translation [65–67], and text-to-audio generation [68]. These approaches typically rely on explicit conditioning during training to enforce alignment between input modalities and output distributions.

More recently, score-based generative frameworks have introduced a new class of methods for *conditional sampling* that do not require conditioning during training. Instead, conditioning is applied at inference time by modifying the sampling dynamics of pretrained, unconditional models [69–71]. This contrasts with conventional methods that train diffusion models explicitly with conditioning [4].

While conceptually related, weighted sampling differs from conditional sampling as conditional sampling typically enforces constraints derived from partial observations or degraded inputs, where the conditioning variable is inherently tied to the instance being sampled (e.g., an incomplete image in inverse problems [72, 73]) and sometimes such noisy observation can be modelled with known quantities. In contrast, weighted sampling reweights samples drawn from a base distribution using an externally defined weight function. This function need not be derived from or coupled with the original distribution, thereby enabling flexible sampling objectives that extend beyond pointwise conditioning.

Nonetheless, several frameworks originally proposed for conditional generation can be adapted for weighted sampling, as they share the common goal of steering the sampling trajectory toward a desired distribution. For example, the guidance

Table 2. Notation and Description

Notation	Description	Note
$\mathbf{X}_t^q$	A random vector of the weighted sampling diffusion process at time t	$\forall t, \mathbf{X}_t^q \in \mathbb{R}^d$
$\mathbf{X}_t^p$	A random vector of the original (base) diffusion process at time t	$\forall t, \mathbf{X}_t^p \in \mathbb{R}^d$
$\mathbf{Z}$	A Gaussian random vector	$\mathbf{Z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
$w(\mathbf{x})$	a weight function	$0 < m < w(\mathbf{x}) < \infty$
$q(\mathbf{x})$	The weighted sampling PDF	$q(\mathbf{x}) = q_0(\mathbf{x})$
$q_t(\mathbf{x})$	The PDF of $\mathbf{X}_t^q$	
$p(\mathbf{x})$	The original (base) PDF	$p(\mathbf{x}) = p_0(\mathbf{x})$
$p_t(\mathbf{x})$	The PDF of $\mathbf{X}_t^p$	
$\nabla_{\mathbf{x}} \log q_t(\mathbf{x})$	The score function of q_t	
$\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$	The score function of p_t	
$\mathcal{N}(\mu, \sigma^2)$	A Gaussian distribution with mean μ and variance σ^2	
$\mathcal{U}(\mathcal{S})$	A uniform distribution over space $\mathcal{S}$	
$\bar{\mathbf{x}}_0'\|_{\mathbf{x},t}$	The conditional mean of $\mathbf{X}_0^p$ for a given observation $\mathbf{x}$ at t	
$[a, b]$	A closed interval from $a \in \mathbb{R}$ to $b \in \mathbb{R}$	
$[v]_i$	The i-th element of a vector v	
$[M]_{i,j}$	The (i, j)-th element of a matrix M	
$\mathbb{N}$	The set of natural numbers	
$\mathbb{R}$	The set of real numbers	

mechanisms in [14, 16], while designed for solving inverse problems, can be naturally repurposed for importance-weighted sampling due to their structural similarity. As such, we include these representative diffusion guidance methods in our performance comparison.

Additionally, recent efforts to enhance sample quality and estimation accuracy have focused on improving the sampling dynamics in diffusion models. Techniques such as stochastic gradient-based refinement [74, 75] and adaptive MCMC approaches [76–78] have demonstrated strong performance in sampling from complex, multimodal distributions. In particular, the sequential Monte Carlo (SMC) strategy proposed in [19] provides an effective mechanism for reward-aware sampling through recursive estimation and adaptive resampling. This method has shown substantial improvements over finetuning-based alternatives and is included in our evaluation as a high-performing baseline.

Scope of This Work. The primary objective of this work is to develop an efficient and scalable algorithm for performing weighted sampling using score-based generative models (SGMs). A key motivation stems from the observation that existing guidance-based sampling methods often incur substantial computational overhead due to repeated resampling or multi-step refinement procedures, particularly when applied to large-scale diffusion models. These approaches typically require accurate estimation of the score function associated with the target density, which can be prohibitively expensive in practice.

To address this limitation, we propose a lightweight approximation framework for constructing the target guidance function. Building on this, we further introduce an adaptive scheduling strategy derived from the theoretical analysis of guidance accuracy, enabling efficient control of the sampling dynamics. The resulting algorithm is well-suited for practical deployment across diverse generative weighted sampling tasks.

B. Technical Results

B.1. Proof of Theorem 1

Proof. Fix $t \in [0, T]$ and $\mathbf{x} \in \mathbb{R}^d$. Recall that the weighted-sampling diffusion marginal can be written as

$$q_t(\mathbf{x}) \;\propto\; p_t(\mathbf{x})\, \mathbb{E}[w(\mathbf{X}_0^p) \mid \mathbf{X}_t^p = \mathbf{x}]\,,$$

where the proportionality constant does not depend on $\mathbf{x}$. Therefore,

$$\nabla_{\mathbf{x}} \log q_t(\mathbf{x}) - \nabla_{\mathbf{x}} \log p_t(\mathbf{x}) = \nabla_{\mathbf{x}} \log \mathbb{E}[w(\mathbf{X}_0^p) \mid \mathbf{X}_t^p = \mathbf{x}]\,. \tag{18}$$

Let $\mu(\mathbf{x}) \coloneqq \bar{\mathbf{x}}_0'|_{\mathbf{x},t}$ and define

$$v(\mathbf{x},t) \;\coloneqq\; \nabla_{\mu(\mathbf{x})} \log w(\mu(\mathbf{x})).$$

By the definition of the first-order approximated guidance (finite-difference HVP),

$$\tilde{g}^{(1)}(\mathbf{x},t) \coloneqq \frac{1}{\sqrt{\bar{\alpha}(t)}}\, v(\mathbf{x},t) + \frac{1-\bar{\alpha}(t)}{\sqrt{\bar{\alpha}(t)}}\, \frac{\nabla_{\mathbf{x}} \log p_t(\mathbf{x} + \epsilon v(\mathbf{x},t)) - \nabla_{\mathbf{x}} \log p_t(\mathbf{x})}{\epsilon}. \tag{19}$$

Using (18), the approximation error satisfies

$$\begin{aligned}
\left\| g(\mathbf{x},t) - \tilde{g}^{(1)}(\mathbf{x},t) \right\| &= \left\| \nabla_{\mathbf{x}} \log \mathbb{E}[w(\mathbf{X}_0^p) \mid \mathbf{X}_t^p = \mathbf{x}] - \tilde{g}^{(1)}(\mathbf{x},t) \right\| \\
&\le \underbrace{\left\| \nabla_{\mathbf{x}} \log \mathbb{E}[w(\mathbf{X}_0^p) \mid \mathbf{X}_t^p = \mathbf{x}] - \nabla_{\mathbf{x}} \log w(\mu(\mathbf{x})) \right\|}_{=:\, T_1(\mathbf{x},t)} + \underbrace{\left\| \nabla_{\mathbf{x}} \log w(\mu(\mathbf{x})) - \tilde{g}^{(1)}(\mathbf{x},t) \right\|}_{=:\, T_2(\mathbf{x},t)}.
\end{aligned} \tag{20}$$

By the chain rule and $\mu(\mathbf{x}) = \frac{1}{\sqrt{\bar{\alpha}(t)}}\big(\mathbf{x} + (1-\bar{\alpha}(t))\nabla_{\mathbf{x}} \log p_t(\mathbf{x})\big)$, we have

$$\nabla_{\mathbf{x}} \log w(\mu(\mathbf{x})) = \frac{(\mathbf{I} + (1-\bar{\alpha}(t)) H_{\log p_t}(\mathbf{x}))}{\sqrt{\bar{\alpha}(t)}}\, v(\mathbf{x},t). \tag{21}$$

Therefore, subtracting (19) from (21) yields

$$\begin{aligned}
T_2(\mathbf{x},t) &= \left\| \nabla_{\mathbf{x}} \log w(\mu(\mathbf{x})) - \tilde{g}^{(1)}(\mathbf{x},t) \right\| \\
&= \frac{1-\bar{\alpha}(t)}{\sqrt{\bar{\alpha}(t)}} \left\| H_{\log p_t}(\mathbf{x})\, v(\mathbf{x},t) - \frac{\nabla_{\mathbf{x}} \log p_t(\mathbf{x} + \epsilon v(\mathbf{x},t)) - \nabla_{\mathbf{x}} \log p_t(\mathbf{x})}{\epsilon} \right\|.
\end{aligned} \tag{22}$$

We now bound the two terms $T_1(\mathbf{x},t)$ and $T_2(\mathbf{x},t)$ separately. By applying the chain rule, the term $T_1(\mathbf{x},t)$ can be further reformulated as follows.

Bounding $T_1(\mathbf{x},t)$. Define the Jensen gap

$$J_t(\mathbf{x}) \;\coloneqq\; \mathbb{E}[w(\mathbf{X}_0^p) \mid \mathbf{X}_t^p = \mathbf{x}] - w(\mu(\mathbf{x})).$$

Then

$$\begin{aligned}
T_1(\mathbf{x},t) &= \|\nabla_{\mathbf{x}} \log \mathbb{E}[w(\mathbf{X}_0^p) \mid \mathbf{X}_t^p = \mathbf{x}] - \nabla_{\mathbf{x}} \log w(\mu(\mathbf{x}))\| \\
&= \left\| \frac{\nabla_{\mathbf{x}} \mathbb{E}[w(\mathbf{X}_0^p) \mid \mathbf{X}_t^p = \mathbf{x}]}{\mathbb{E}[w(\mathbf{X}_0^p) \mid \mathbf{X}_t^p = \mathbf{x}]} - \frac{\nabla_{\mathbf{x}} w(\mu(\mathbf{x}))}{w(\mu(\mathbf{x}))} \right\| \\
&= \left\| \frac{w(\mu(\mathbf{x}))\, \nabla_{\mathbf{x}} J_t(\mathbf{x}) - J_t(\mathbf{x})\, \nabla_{\mathbf{x}} w(\mu(\mathbf{x}))}{\mathbb{E}[w(\mathbf{X}_0^p) \mid \mathbf{X}_t^p = \mathbf{x}]\; w(\mu(\mathbf{x}))} \right\| \\
&\le \frac{\|\nabla_{\mathbf{x}} J_t(\mathbf{x})\|}{\mathbb{E}[w(\mathbf{X}_0^p) \mid \mathbf{X}_t^p = \mathbf{x}]} + \frac{|J_t(\mathbf{x})|\, \|\nabla_{\mathbf{x}} w(\mu(\mathbf{x}))\|}{\mathbb{E}[w(\mathbf{X}_0^p) \mid \mathbf{X}_t^p = \mathbf{x}]\; w(\mu(\mathbf{x}))} \\
&\le \frac{\|\nabla_{\mathbf{x}} J_t(\mathbf{x})\|}{m} + \frac{|J_t(\mathbf{x})|\, \|\nabla_{\mathbf{x}} \log w(\mu(\mathbf{x}))\|}{m},
\end{aligned} \tag{23}$$

where we used $w(\cdot) \ge m$ so that $\mathbb{E}[w(\mathbf{X}_0^p) \mid \mathbf{X}_t^p = \mathbf{x}] \ge m$ and $\|\nabla_{\mathbf{x}} w(\mu)\| = w(\mu)\|\nabla_{\mathbf{x}} \log w(\mu)\|$.

Bounding $J_t(\mathbf{x})$ and $\nabla_{\mathbf{x}} J_t(\mathbf{x})$. Let $e(\mathbf{y},\mathbf{x}) \coloneqq \mathbf{y} - \mu(\mathbf{x})$. Since $\|H_w(\mathbf{z})\| \le \eta_2$ for all $\mathbf{z}$ (Assumption 1), Taylor's theorem implies the smoothness inequality

$$\left| w(\mathbf{y}) - w(\mu(\mathbf{x})) - \nabla w(\mu(\mathbf{x}))^\top (\mathbf{y} - \mu(\mathbf{x})) \right| \le \frac{\eta_2}{2}\, \|e(\mathbf{y},\mathbf{x})\|^2. \tag{24}$$

Taking conditional expectation w.r.t. $\mathbf{X}_0^p \sim p_{\mathbf{X}_0^p|\mathbf{X}_t^p}(\cdot \mid \mathbf{x})$ and using $\mathbb{E}[e(\mathbf{X}_0^p, \mathbf{x}) \mid \mathbf{X}_t^p = \mathbf{x}] = 0$, we obtain

$$|J_t(\mathbf{x})| \le \frac{\eta_2}{2} \mathbb{E}\Big[\|e(\mathbf{X}_0^p, \mathbf{x})\|^2 \;\Big|\; \mathbf{X}_t^p = \mathbf{x}\Big]. \tag{25}$$

Define the conditional score

$$S(\mathbf{y}, \mathbf{x}, t) \;:=\; \nabla_{\mathbf{x}} \log p_{\mathbf{X}_0^p|\mathbf{X}_t^p}(\mathbf{y} \mid \mathbf{x}).$$

By differentiation under the integral sign,

$$\nabla_{\mathbf{x}} \mathbb{E}[w(\mathbf{X}_0^p) \mid \mathbf{X}_t^p = \mathbf{x}] = \mathbb{E}[w(\mathbf{X}_0^p)\, S(\mathbf{X}_0^p, \mathbf{x}, t) \mid \mathbf{X}_t^p = \mathbf{x}],$$

and since $\int p_{\mathbf{X}_0^p|\mathbf{X}_t^p}(\mathbf{y} \mid \mathbf{x})\, d\mathbf{y} = 1$, we have $\mathbb{E}[S(\mathbf{X}_0^p, \mathbf{x}, t) \mid \mathbf{X}_t^p = \mathbf{x}] = 0$. Using the chain rule $\nabla_{\mathbf{x}} w(\mu(\mathbf{x})) = (\nabla_{\mathbf{x}} \mu(\mathbf{x}))^\top \nabla w(\mu(\mathbf{x}))$ and $\nabla_{\mathbf{x}} \mu(\mathbf{x}) = \mathbb{E}[e(\mathbf{X}_0^p, \mathbf{x}) S(\mathbf{X}_0^p, \mathbf{x}, t)^\top \mid \mathbf{X}_t^p = \mathbf{x}]$, we obtain

$$\nabla_{\mathbf{x}} J_t(\mathbf{x}) = \mathbb{E}\Big[\Big(w(\mathbf{X}_0^p) - w(\mu(\mathbf{x})) - \nabla w(\mu(\mathbf{x}))^\top e(\mathbf{X}_0^p, \mathbf{x})\Big)\, S(\mathbf{X}_0^p, \mathbf{x}, t) \;\Big|\; \mathbf{X}_t^p = \mathbf{x}\Big], \tag{26}$$

where we used $\mathbb{E}[S(\mathbf{X}_0^p, \mathbf{x}, t) \mid \mathbf{X}_t^p = \mathbf{x}] = 0$ to insert the term $-w(\mu(\mathbf{x}))$. Hence, by Jensen's inequality, $\|S(\cdot, \mathbf{x}, t)\| \le \gamma_t$ (Assumption 2), and (24),

$$\begin{aligned}
\|\nabla_{\mathbf{x}} J_t(\mathbf{x})\| &\le \gamma_t\, \mathbb{E}\Big[\Big|w(\mathbf{X}_0^p) - w(\mu(\mathbf{x})) - \nabla w(\mu(\mathbf{x}))^\top e(\mathbf{X}_0^p, \mathbf{x})\Big| \;\Big|\; \mathbf{X}_t^p = \mathbf{x}\Big] \\
&\le \frac{\eta_2 \gamma_t}{2}\, \mathbb{E}\Big[\|e(\mathbf{X}_0^p, \mathbf{x})\|^2 \;\Big|\; \mathbf{X}_t^p = \mathbf{x}\Big].
\end{aligned} \tag{27}$$

Bounding $\|\nabla_{\mathbf{x}} \log w(\mu(\mathbf{x}))\|$. From (21) and $\|H_{\log p_t}(\mathbf{x})\| \le \zeta_t$ (Assumption 2), and $\|v(\mathbf{x}, t)\| \le \eta$ (Assumption 1),

$$\|\nabla_{\mathbf{x}} \log w(\mu(\mathbf{x}))\| \le \frac{1 + (1 - \bar{\alpha}(t))\zeta_t}{\sqrt{\bar{\alpha}(t)}}\, \eta. \tag{28}$$

Putting together the bound for $T_1(\mathbf{x}, t)$. Substituting (25), (27), and (28) into (23) yields

$$T_1(\mathbf{x}, t) \le \left(\frac{\gamma_t \eta_2}{2m} + \frac{(1 + (1 - \bar{\alpha}(t))\zeta_t)\, \eta\, \eta_2}{2m\sqrt{\bar{\alpha}(t)}}\right) \mathbb{E}\Big[\|\mathbf{X}_0^p - \mu(\mathbf{x})\|^2 \;\Big|\; \mathbf{X}_t^p = \mathbf{x}\Big]. \tag{29}$$

Bounding $T_2(\mathbf{x}, t)$. Using Taylor's theorem with integral remainder for $\nabla_{\mathbf{x}} \log p_t$,

$$\nabla_{\mathbf{x}} \log p_t(\mathbf{x} + \epsilon v) = \nabla_{\mathbf{x}} \log p_t(\mathbf{x}) + \int_0^\epsilon H_{\log p_t}(\mathbf{x} + sv)\, v\, ds,$$

we obtain

$$\frac{\nabla_{\mathbf{x}} \log p_t(\mathbf{x} + \epsilon v) - \nabla_{\mathbf{x}} \log p_t(\mathbf{x})}{\epsilon} - H_{\log p_t}(\mathbf{x})\, v = \frac{1}{\epsilon} \int_0^\epsilon \big(H_{\log p_t}(\mathbf{x} + sv) - H_{\log p_t}(\mathbf{x})\big)\, v\, ds.$$

Taking norms and using the Hessian Lipschitz property $\|H_{\log p_t}(\mathbf{x}) - H_{\log p_t}(\mathbf{y})\| \le L_t \|\mathbf{x} - \mathbf{y}\|$ gives

$$\begin{aligned}
T_2(\mathbf{x}, t) &= \frac{1 - \bar{\alpha}(t)}{\sqrt{\bar{\alpha}(t)}} \left\| H_{\log p_t}(\mathbf{x})\, v - \frac{\nabla_{\mathbf{x}} \log p_t(\mathbf{x} + \epsilon v) - \nabla_{\mathbf{x}} \log p_t(\mathbf{x})}{\epsilon} \right\| \\
&\le \frac{1 - \bar{\alpha}(t)}{\sqrt{\bar{\alpha}(t)}}\, \frac{1}{\epsilon} \int_0^\epsilon L_t\, s\, \|v(\mathbf{x}, t)\|^2\, ds \le \frac{(1 - \bar{\alpha}(t))\, L_t\, \eta^2\, \epsilon}{2\sqrt{\bar{\alpha}(t)}}.
\end{aligned} \tag{30}$$

Conclusion. Combining (20), (29), and (30), we obtain

$$\Big\|g(\mathbf{x}, t) - \tilde{g}^{(1)}(\mathbf{x}, t)\Big\| \le \frac{(1 - \bar{\alpha}(t))\, L_t\, \eta^2\, \epsilon}{2\sqrt{\bar{\alpha}(t)}} + \left(\frac{\gamma_t \eta_2}{2m} + \frac{(1 + (1 - \bar{\alpha}(t))\zeta_t)\, \eta\, \eta_2}{2m\sqrt{\bar{\alpha}(t)}}\right) \mathbb{E}\Big[\|\mathbf{X}_0^p - \mu(\mathbf{x})\|^2 \;\Big|\; \mathbf{X}_t^p = \mathbf{x}\Big],$$

which matches the definition of $u(\mathbf{x}, t)$ in Theorem 1. This completes the proof. □

B.2. Proof of Proposition 1

We have the mean squared error risk measure which is given as follows.

$$r(t;\tau) = \mathbb{E}[\|\nabla_{\mathbf{X}^q} \log q_t(\mathbf{X}^q) - \nabla_{\mathbf{X}^q} \log \tilde{q}_t(\mathbf{X}^q)\|^2] = \mathbb{E}[\|g(\mathbf{X}^q, t) - \tau(t)\tilde{g}^{(1)}(\mathbf{X}^q, t)\|^2] \tag{31}$$

$$= (1-\tau(t))^2 \mathbb{E}[\|g(\mathbf{X}^q, t)\|^2] + (\tau(t))^2 \mathbb{E}[\|\Delta^{(1)}(\mathbf{X}^q, t)\|^2]. \tag{32}$$

Recall the definition of the ground-truth guidance term,

$$g(\mathbf{x}, t) = \frac{\nabla_{\mathbf{x}} \mathbb{E}_{\mathbf{X}_0^p \sim p_{\mathbf{X}_0^p|\mathbf{X}_t^p}(\cdot|\mathbf{x})}[w(\mathbf{X}_0^p)]}{\mathbb{E}_{\mathbf{X}_0^p \sim p_{\mathbf{X}_0^p|\mathbf{X}_t^p}(\cdot|\mathbf{x})}[w(\mathbf{X}_0^p)]}. \tag{33}$$

Then, the numerator of (33) can be represented as follows.

$$\nabla_{\mathbf{x}} \mathbb{E}_{\mathbf{X}_0^p \sim p_{\mathbf{X}_0^p|\mathbf{X}_t^p}(\cdot|\mathbf{x})}[w(\mathbf{X}_0^p)] = \int w(\mathbf{y})\, \nabla_{\mathbf{x}} p_{\mathbf{X}_0^p|\mathbf{X}_t^p}(\mathbf{y} \mid \mathbf{x})\, d\mathbf{y}$$

$$= \int w(\mathbf{y})\, p_{\mathbf{X}_0^p|\mathbf{X}_t^p}(\mathbf{y} \mid \mathbf{x})\, \nabla_{\mathbf{x}} \log p_{\mathbf{X}_0^p|\mathbf{X}_t^p}(\mathbf{y} \mid \mathbf{x})\, d\mathbf{y} = \mathbb{E}_{\mathbf{X}_0^p|\mathbf{X}_t^p=\mathbf{x}}\big[w(\mathbf{X}_0^p)\, S(\mathbf{X}_0^p, \mathbf{x}, t)\big], \tag{34}$$

where $S(\mathbf{X}_0^p, \mathbf{x}, t) \triangleq \nabla_{\mathbf{x}} \log p_{\mathbf{X}_0^p|\mathbf{X}_t^p}(\mathbf{X}_0^p \mid \mathbf{x})$ is the conditional score.

Let $h(\mathbf{y}, \mathbf{x}) = w(\mathbf{y}) p_{\mathbf{X}_0^p|\mathbf{X}_t^p}(\mathbf{y} \mid \mathbf{x})$. Using $\nabla_{\mathbf{x}} p = p \nabla_{\mathbf{x}} \log p$, we have

$$\big\|\nabla_{\mathbf{x}} h(\mathbf{y}, \mathbf{x})\big\| = w(\mathbf{y})\, \big\|\nabla_{\mathbf{x}} p_{\mathbf{X}_0^p|\mathbf{X}_t^p}(\mathbf{y} \mid \mathbf{x})\big\| = w(\mathbf{y})\, p_{\mathbf{X}_0^p|\mathbf{X}_t^p}(\mathbf{y} \mid \mathbf{x}) \big\|\nabla_{\mathbf{x}} \log p_{\mathbf{X}_0^p|\mathbf{X}_t^p}(\mathbf{y} \mid \mathbf{x})\big\| \le \gamma_t\, w(\mathbf{y})\, p_{\mathbf{X}_0^p|\mathbf{X}_t^p}(\mathbf{y} \mid \mathbf{x}).$$

Since $\mathbb{E}[w(\mathbf{X}_0^p) \mid \mathbf{X}_t^p = \mathbf{x}] = \int w(\mathbf{y}) p_{\mathbf{X}_0^p|\mathbf{X}_t^p}(\mathbf{y} \mid \mathbf{x}) d\mathbf{y} < \infty$ and $\gamma_t < \infty$, we obtain

$$\int \big\|\nabla_{\mathbf{x}} h(\mathbf{y}, \mathbf{x})\big\| d\mathbf{y} \le \gamma_t\, \mathbb{E}[w(\mathbf{X}_0^p) \mid \mathbf{X}_t^p = \mathbf{x}] < \infty.$$

Therefore, differentiation under the integral sign is justified (Leibniz rule), giving $\nabla_{\mathbf{x}} \int h = \int \nabla_{\mathbf{x}} h$.

Dividing (34) by $\mathbb{E}[\, w(\mathbf{X}_0^p) \mid \mathbf{X}_t^p = \mathbf{x}\,]$ gives

$$\|g(\mathbf{x}, t)\| \;=\; \frac{\|\mathbb{E}[w(\mathbf{X}_0^p)\, S(\mathbf{X}_0^p, \mathbf{x}, t) \mid \mathbf{X}_t^p = \mathbf{x}]\|}{\|\mathbb{E}[w(\mathbf{X}_0^p) \mid \mathbf{X}_t^p = \mathbf{x}]\|} \le \frac{\mathbb{E}[w(\mathbf{X}_0^p)\, \|S(\mathbf{X}_0^p, \mathbf{x}, t)\| \mid \mathbf{X}_t^p = \mathbf{x}]}{\mathbb{E}[w(\mathbf{X}_0^p) \mid \mathbf{X}_t^p = \mathbf{x}]} \le \frac{\gamma_t \mathbb{E}[w(\mathbf{X}_0^p) \mid \mathbf{X}_t^p = \mathbf{x}]}{\mathbb{E}[w(\mathbf{X}_0^p) \mid \mathbf{X}_t^p = \mathbf{x}]} = \gamma_t. \tag{35}$$

By Assumption 2, we have $\|g(\mathbf{x}, t)\| \le \gamma_t$.

We now examine $\mathbb{E}[\|\Delta^{(1)}(\mathbf{X}, t)\|^2]$. Recall the definition of the upper bound of the uncertainty u and for all $\mathbf{x}$ and t, we have

$$u(\mathbf{x}, t) \le \frac{(1-\bar{\alpha}(t)) L \eta^2 \epsilon}{2\sqrt{\bar{\alpha}(t)}} + \left(\frac{\gamma \eta_2}{2m} + \frac{(1 + (1-\bar{\alpha}(t))\zeta)\, \eta \eta_2}{2m\sqrt{\bar{\alpha}(t)}}\right) \mathbb{E}\left[\|\mathbf{X}_0^p - \bar{\mathbf{x}}_0'|_{\mathbf{x},t}\|^2 | \mathbf{X}_t^p = \mathbf{x}\right], \tag{36}$$

where $\gamma = \max_{t\in[0,T]} \gamma_t$, $\zeta = \max_{t\in[0,T]} \zeta_t$, $L = \max_{t\in[0,T]} L_t$. This replacement converts the time-dependent bound in Theorem 1 into a time-uniform (and hence looser) deterministic envelope. The conditional covariance admits the matrix identity

$$\mathrm{Cov}(\mathbf{X}_0^p \mid \mathbf{X}_t^p = \mathbf{x}) = \frac{1-\bar{\alpha}(t)}{\bar{\alpha}(t)} \Big(\mathbf{I} + (1-\bar{\alpha}(t)) H_{\log p_t}(\mathbf{x})\Big). \tag{37}$$

Hence, it can be represented as [72]

$$\mathbb{E}\big[\|\mathbf{X}_0^p - \bar{\mathbf{x}}_0'|_{\mathbf{x},t}\|^2 \mid \mathbf{X}_t^p = \mathbf{x}\big] = \mathrm{Tr}(\mathrm{Cov}(\mathbf{X}_0^p \mid \mathbf{X}_t^p = \mathbf{x})) \tag{38}$$

$$= d\frac{1-\bar{\alpha}(t)}{\bar{\alpha}(t)} + \frac{(1-\bar{\alpha}(t))^2}{\bar{\alpha}(t)} \mathrm{Tr}(H_{\log p_t}(\mathbf{x})). \tag{39}$$

Since $\mathrm{Cov}(\mathbf{X}_0^p \mid \mathbf{X}_t^p = \mathbf{x}) \succeq 0$ and $\frac{1-\bar{\alpha}(t)}{\bar{\alpha}(t)} > 0$, it follows that

$$\mathbf{I} + (1-\bar{\alpha}(t)) H_{\log p_t}(\mathbf{x}) \succeq 0, \tag{40}$$

equivalently,

$$\lambda_{\min}(H_{\log p_t}(\mathbf{x})) \geq -(1-\bar{\alpha}(t))^{-1}. \tag{41}$$

Under the assumptions $\|H_{\log p_t}(\mathbf{x})\| \leq \zeta$, $\mathrm{Tr}(H_{\log p_t}(\mathbf{x})) \leq d\zeta$, and substituting (38) to the upper bound in (36), we have $u(\mathbf{x},t) \leq u'(t)$ where

$$\begin{aligned} u'(t) = \frac{L\eta^2\epsilon}{2} \frac{1-\bar{\alpha}(t)}{\bar{\alpha}(t)^{1/2}} + \frac{d(\gamma\eta_2)}{2m} \frac{1-\bar{\alpha}(t)}{\bar{\alpha}(t)} + \frac{d\zeta(\gamma\eta_2)}{2m} \frac{(1-\bar{\alpha}(t))^2}{\bar{\alpha}(t)} \\ + \frac{d\eta\eta_2}{2m} \frac{1-\bar{\alpha}(t)}{\bar{\alpha}(t)^{3/2}} + \frac{d\zeta\eta\eta_2}{m} \frac{(1-\bar{\alpha}(t))^2}{\bar{\alpha}(t)^{3/2}} + \frac{d\zeta^2\eta\eta_2}{2m} \frac{(1-\bar{\alpha}(t))^3}{\bar{\alpha}(t)^{3/2}}. \end{aligned} \tag{42}$$

Define $\gamma \coloneqq \max_{t\in[0,T]} \gamma_t$. Set $c_1 \coloneqq \gamma^2$. Since $\|g(\mathbf{x},t)\| \leq \gamma_t$ for all $\mathbf{x}$, we have $\mathbb{E}[\|g(\mathbf{X}_t^q, t)\|^2] \leq \gamma_t^2 \leq \gamma^2$. Moreover, $\|\Delta^{(1)}(\mathbf{x},t)\| \leq u(\mathbf{x},t) \leq u'(t)$ implies $\mathbb{E}[\|\Delta^{(1)}(\mathbf{X}_t^q, t)\|^2] \leq (u'(t))^2$. Therefore,

$$r(t;\tau) \leq r'(t;\tau) \coloneqq (1-\tau(t))^2\gamma^2 + (\tau(t))^2 (u'(t))^2. \tag{43}$$

For each fixed t, $r'(t;\tau)$ is a strictly convex quadratic in τ, and expanding (43) gives

$$r'(t;\tau) = \gamma^2 - 2\gamma^2\tau + \big(\gamma^2 + (u'(t))^2\big)\tau^2.$$

Hence its unique minimizer over $\tau \in \mathbb{R}$ is

$$\tau^*(t) = \frac{\gamma^2}{\gamma^2 + (u'(t))^2}. \tag{44}$$

It remains to write $(u'(t))^2$ in closed form. Define $c_1 \coloneqq \gamma^2$ and rewrite (42) as

$$u'(t) = \sum_{k=1}^{6} A_k\, \phi_k(t), \qquad \phi_k(t) = \frac{(1-\bar{\alpha}(t))^{i_k}}{\bar{\alpha}(t)^{j_k}}, \tag{45}$$

with exponent pairs

$$(i_k, j_k) \in \Big\{(1,\tfrac{1}{2}), (1,1), (2,1), (1,\tfrac{3}{2}), (2,\tfrac{3}{2}), (3,\tfrac{3}{2})\Big\},$$

and coefficients

$$A_1 = \frac{L\eta^2\epsilon}{2}, \quad A_2 = \frac{d\,\gamma\eta_2}{2m}, \quad A_3 = \frac{d\,\zeta\,\gamma\eta_2}{2m}, \quad A_4 = \frac{d\eta\eta_2}{2m}, \quad A_5 = \frac{d\zeta\eta\eta_2}{m}, \quad A_6 = \frac{d\zeta^2\eta\eta_2}{2m}.$$

Therefore, $(u'(t))^2$ expands exactly as

$$(u'(t))^2 = \sum_{k=1}^{6}\sum_{\ell=1}^{6} A_k A_\ell \frac{(1-\bar{\alpha}(t))^{i_k+i_\ell}}{\bar{\alpha}(t)^{j_k+j_\ell}} = \sum_{(i,j)\in\mathcal{I}} c_{(i,j)} \frac{(1-\bar{\alpha}(t))^i}{\bar{\alpha}(t)^j}, \tag{46}$$

where the (finite) index set is

$$\mathcal{I} = \Big\{(2,1), (2,\tfrac{3}{2}), (2,2), (2,\tfrac{5}{2}), (2,3), (3,\tfrac{3}{2}), (3,2), (3,\tfrac{5}{2}), (3,3), (4,2), (4,\tfrac{5}{2}), (4,3), (5,\tfrac{5}{2}), (5,3), (6,3)\Big\},$$

and the coefficients are obtained by collecting equal exponent pairs:

$$\begin{aligned} &c_{(2,1)} = A_1^2, \qquad c_{(2,\frac{3}{2})} = 2A_1A_2, \qquad c_{(2,2)} = A_2^2 + 2A_1A_4, \\ &c_{(2,\frac{5}{2})} = 2A_2A_4, \qquad c_{(2,3)} = A_4^2, \qquad c_{(3,\frac{3}{2})} = 2A_1A_3, \\ &c_{(3,2)} = 2A_1A_5 + 2A_2A_3, \qquad c_{(3,\frac{5}{2})} = 2A_2A_5 + 2A_3A_4, \qquad c_{(3,3)} = 2A_4A_5, \\ &c_{(4,2)} = A_3^2 + 2A_1A_6, \qquad c_{(4,\frac{5}{2})} = 2A_2A_6 + 2A_3A_5, \qquad c_{(4,3)} = A_5^2 + 2A_4A_6, \\ &c_{(5,\frac{5}{2})} = 2A_3A_6, \qquad c_{(5,3)} = 2A_5A_6, \qquad c_{(6,3)} = A_6^2. \end{aligned}$$

Substituting (46) into (44) yields

$$\tau^*(t) = \frac{c_1}{c_1 + \sum_{(i,j)\in\mathcal{I}} c_{(i,j)} \frac{(1-\bar{\alpha}(t))^i}{\bar{\alpha}(t)^j}} = \left(1 + \sum_{(i,j)\in\mathcal{I}} \frac{c_{(i,j)}}{c_1} \frac{(1-\bar{\alpha}(t))^i}{\bar{\alpha}(t)^j}\right)^{-1}.$$

Approximation of $\tau^*(t)$. Figure 5 illustrates the behavior of the function $\left(1 + \frac{(1-\bar{\alpha}(t))^i}{\bar{\alpha}(t)^j}\right)^{-1}$, for various index pairs $(i,j) \in \mathcal{I}$, under the cosine-based variance schedule. Here, $\bar{\alpha}(t)$ denotes the cumulative product of a scheduling function applied to diffusion coefficients, as typically derived from variance-preserving score-based generative models.

It is noteworthy that, due to the decreasing nature of the term $\frac{(1-\bar{\alpha}(t))^i}{\bar{\alpha}(t)^j}$ in t, the composite function above exhibits a monotonic increase over time. Rather than precisely estimating the weight coefficients $c_{(i,j)}$ for each index pair (i,j), we propose to approximate the overall effect with a single representative parameter. This simplification is motivated by the empirical observation that the variation across terms in $\mathcal{I}$ often follows a coherent monotonic trend that can be effectively captured using a single-parameter approximation.

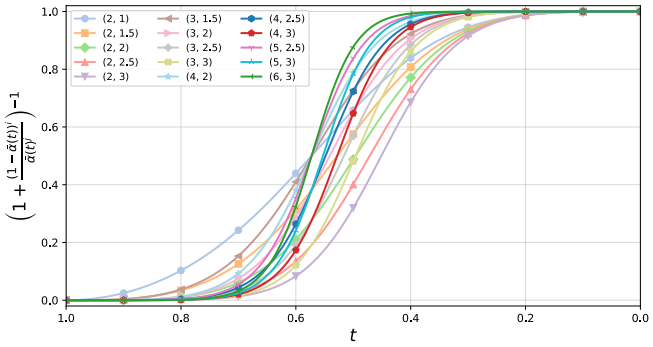

Figure 5. Sigmoid-shaped curves defined by $\left(1 + \frac{(1-\bar{\alpha})^i}{\bar{\alpha}^j}\right)^{-1}$ for various $(i,j) \in \mathcal{I}$, under the cosine-based cumulative schedule $\bar{\alpha}(t)$. For practical implementation, we approximate the behavior of $\tau^*(t)$ using a single-parameter surrogate.

Scheduling without Assumption 3. Recall the mean squared error between the score functions $\nabla_{\mathbf{X}} \log q_t(\mathbf{X}^q)$ and $\nabla_{\mathbf{X}} \log \tilde{q}_t(\mathbf{X}^q)$, which is expressed as

$$r(t;\tau) = \mathbb{E}[\|\nabla_{\mathbf{X}} \log q_t(\mathbf{X}^q) - \nabla_{\mathbf{X}^q} \log \tilde{q}_t(\mathbf{X}^q)\|^2] = \mathbb{E}[\|g(\mathbf{X}^q, t) - \tau(t)\tilde{g}^{(1)}(\mathbf{X}^q, t)\|^2] \tag{47}$$

$$= (1-\tau(t))^2 \mathbb{E}[\|g(\mathbf{X}^q, t)\|^2] + \mathbb{E}[2(1-\tau(t))\tau(t) g(\mathbf{X}^q, t)^\top \Delta^{(1)}(\mathbf{X}^q, t)] + (\tau(t))^2 \mathbb{E}[\|\Delta^{(1)}(\mathbf{X}^q, t)\|^2] \tag{48}$$

Applying the Cauchy–Schwarz inequality and upper bounds on the second and third terms yields

$$r(t;\tau) \le (1-\tau(t))^2\gamma^2 + 2(1-\tau(t))\tau(t)\sqrt{\mathbb{E}[\|g(\mathbf{X}^q, t)\|^2]}\sqrt{\mathbb{E}[\|\Delta^{(1)}(\mathbf{X}^q, t)\|^2]} + (\tau(t))^2(u'(t))^2$$

$$\le (1-\tau(t))^2\gamma^2 + 2(1-\tau(t))\tau(t)\gamma u'(t) + (\tau(t))^2(u'(t))^2 \tag{49}$$

$$= r_{\text{ub}}(t;\tau) \coloneqq \gamma^2 + 2\gamma\big(u'(t) - \gamma\big)\tau + \big(\gamma - u'(t)\big)^2\tau^2. \tag{50}$$

Here, we denote $\tau(t)$ by τ since t is fixed. The upper bound $r_{\text{ub}}(t;\tau)$ is a quadratic function of τ with a non-negative leading coefficient $(\gamma - u'(t))^2$, and hence is convex.

We now examine the minimizer of $r_{\text{ub}}(t;\tau)$ over $\tau \in [0,1]$. Taking derivative yields

$$\frac{\partial r_{\text{ub}}(t;\tau)}{\partial \tau} = 2\gamma\big(u'(t) - \gamma\big) + 2\big(\gamma - u'(t)\big)^2\tau$$

which vanishes at

$$\tau_{\text{crit}} = \frac{\gamma}{\gamma - u'(t)}.$$

Case $u'(t) < \gamma$**.** Then $\gamma - u'(t) > 0$, which implies $\tau_{\text{crit}} > 1$. Convexity forces the minimum on $[0, 1]$ to occur at the right endpoint, that is, $\tau^*(t) = 1$.

Case $u'(t) > \gamma$**.** Now $\gamma - u'(t) < 0$, hence $\tau_{\text{crit}} < 0$. The minimum is attained at the left endpoint, $\tau^*(t) = 0$.

Case $u'(t) = \gamma$**.** Every τ in the interval therefore minimizes the function.

Importantly, the scheduler $\tau^*(t)$ is monotonic in t due to the behavior of $u'(t)$, which denotes the upper bound of uncertainty as defined in (42). As $t \to 0$, we have $u'(t) \to 0$ and thus $u'(t) < \gamma$ for any $\gamma > 0$, leading to $\tau^*(t) = 1$. Conversely, at large t, the uncertainty increases and $u'(t) > \gamma$, resulting in $\tau^*(t) = 0$. This implies that there exists a critical threshold $t' \in [0, T]$ such that $\tau^*(t)$ behaves as a step function, switching from 1 to 0 at $t = t'$. In our experiments (Appendices C.3–C.5), we adopt this scheduler using a fixed threshold of $t' = 0.7T$ for heuristic estimation of the strategy.

B.3. Discretization of the SDE for weighted sampling

To generate samples following the weighted sampling PDF q, the proposed SDE can be implemented via discretization methods. For example, the Euler-Maruyama scheme discretizes the time interval $[0, T]$ with step size Δt, allowing iterative solutions to the SDE. In this section, we consider $T \in \mathbb{N}$ (a positive integer) and adopt a unit time discretization, i.e., $\Delta t = 1$. Recall the proposed SDE which is given as follows.

$$\mathrm{d}\mathbf{X}_t^{\tilde{q}} = -\frac{\beta(t)}{2}\left[\mathbf{X}_t^{\tilde{q}} + 2\nabla_{\mathbf{X}_t^{\tilde{q}}} \log \tilde{q}_t(\mathbf{X}_t^{\tilde{q}})\right] \mathrm{d}t + \sqrt{\beta(t)}\, \mathrm{d}\tilde{\mathbf{W}}_t. \tag{51}$$

Let $\{\beta(t)\}_{t\in[0,T]}$ denote the variance schedule of the forward diffusion process, and define the discrete instants $t_k = \frac{k}{N}T,\ k = 0, \ldots, N$, with unit–step discretisation $\Delta t = T/N = 1$ for ease of exposition. Denote by $\mathbf{x}_k \triangleq \mathbf{x}_{t_k}^{\tilde{q}}$ the discretised state. Starting from the reverse–time SDE proposed in (51), the Euler–Maruyama scheme yields

$$\mathbf{x}_{k-1} = \mathbf{x}_k + \frac{\beta_k}{2}\left(\mathbf{x}_k + 2\nabla_{\mathbf{x}_k} \log \tilde{q}_{t_k}(\mathbf{x}_k)\right) + \sqrt{\beta_k}\, \boldsymbol{\xi}_k, \qquad \boldsymbol{\xi}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{52}$$

where $\beta_k = \beta(t_k)$. To connect (52) to the standard diffusion notation, define the single–step and cumulative attenuation factors $\alpha_k = 1 - \beta_k$, and $\bar{\alpha}_k = \prod_{j=1}^{k} \alpha_j$. Using the representation of the SGM, we denote the denoising model with the following representation $\nabla_{\mathbf{x}_k} \log \tilde{q}_{t_k}(\mathbf{x}_k) \approx -\frac{1}{\sqrt{1-\bar{\alpha}_k}}\, \hat{\epsilon}_\theta(\mathbf{x}_k, k)$. Based on these, we adopt the discrete DDPM [1] update

$$\mathbf{x}_{k-1} = \frac{\sqrt{\bar{\alpha}_{k-1}}}{\sqrt{\bar{\alpha}_k}}\left(\mathbf{x}_k - \sqrt{1-\bar{\alpha}_k}\, \hat{\epsilon}_\theta(\mathbf{x}_k, k)\right) + \sqrt{1 - \bar{\alpha}_{k-1} - \sigma_k^2}\, \hat{\epsilon}_\theta(\mathbf{x}_k, k) + \sigma_k \boldsymbol{\xi}_k, \tag{53}$$

with $\sigma_k \in [0, 1]$ the noise level, $\sigma_k = \eta_s \sqrt{(1-\bar{\alpha}_{k-1})(1-\alpha_k)/\,(1-\bar{\alpha}_k)}$, η_s the parameter controlling the stochasticity. When we use DDIM, we have $\eta_s = 0$ in (53) suppressing the stochastic component and transforms (53) into the following update rule

$$\mathbf{x}_{k-1} = \sqrt{\bar{\alpha}_{k-1}}\, \hat{\mathbf{x}}_{0,k} + \sqrt{1-\bar{\alpha}_{k-1}}\, \hat{\epsilon}_\theta(\mathbf{x}_k, k), \quad \hat{\mathbf{x}}_{0,k} = \frac{\mathbf{x}_k - \sqrt{1-\bar{\alpha}_k}\, \hat{\epsilon}_\theta(\mathbf{x}_k, k)}{\sqrt{\bar{\alpha}_k}}. \tag{54}$$

For experiments in Appendix C.1, C.2, and C.5, we adopt DDIM sampling. For those in Appendix C.4, we use DDPM sampling.

B.4. Relation to Existing Methods

In the main text, we adopt a first-order Taylor approximation of the weighted score function based on the conditional mean of the initial sample:

$$\nabla_{\mathbf{x}} \log q_t(\mathbf{x}) = \nabla_{\mathbf{x}} \log p_t(\mathbf{x}) + \nabla_{\mathbf{x}} \log \mathbb{E}_{\mathbf{X}_0^p \sim p(\cdot|\mathbf{x})}[w(\mathbf{X}_0^p)] \tag{55}$$

$$\approx \nabla_{\mathbf{x}} \log p_t(\mathbf{x}) + \nabla_{\mathbf{x}} \log w(\bar{\mathbf{x}}_0'|_{\mathbf{x},t}), \tag{56}$$

where $\bar{\mathbf{x}}_0'|_{\mathbf{x},t}$ denotes the conditional expectation of $\mathbf{X}_0^p$ given $\mathbf{x}$ under the reference diffusion process.

This approximation is closely related to the denoising-based posterior sampling technique introduced in [16, 19], which was originally proposed for solving inverse problems via diffusion guidance. Consider a binary random variable $C \in \{0, 1\}$ such that $p(C = 1 \mid \mathbf{x}) \propto w(\mathbf{x})$. Then, by Bayes' rule, the posterior over $\mathbf{x}$ conditioned on $C = 1$ becomes

$$p(\mathbf{x} \mid C = 1) = \frac{p(\mathbf{x})\, p(C = 1 \mid \mathbf{x})}{p(C = 1)} \propto p(\mathbf{x})\, w(\mathbf{x}), \tag{57}$$

yielding the desired reweighted distribution. Taking the gradient of the log posterior gives

$$\nabla_{\mathbf{x}} \log p_t(\mathbf{x} \mid C = 1) = \nabla_{\mathbf{x}} \log p_t(\mathbf{x}) + \nabla_{\mathbf{x}} \log w(\mathbf{x}). \tag{58}$$

Following [16], one can replace $\mathbf{x}$ in the second term with the estimated conditional mean $\bar{\mathbf{x}}_0'|_{\mathbf{x},t}$, resulting in the approximation

$$\nabla_{\mathbf{x}} \log w(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log w(\bar{\mathbf{x}}_0'|_{\mathbf{x},t}).$$

This formulation has been well investigated in [19].

Compared to existing methods [14, 16, 19, 79, 80], which compute the full gradient $\nabla_{\mathbf{x}} \log w(\bar{\mathbf{x}}_0'|_{\mathbf{x},t})$ by backpropagating through the score network, our approach avoids differentiating through the composition of the weight function and the score model. This distinction reduces computational overhead, especially when the score function is parameterized by a large-scale neural network.

Specifically, we approximate the directional update using a finite-difference estimate of the Hessian-vector product, requiring only one gradient evaluation of the task-specific loss. This results in substantially lower computational cost, particularly in settings where the guidance model is significantly larger than the task model, as demonstrated in Sec. 4.

C. Implementations and Experiments

C.1. Experiments with Synthetic Datasets

Base density. We evaluate the proposed method and baselines on a 2D weighted sampling task where the base distribution is multimodal, and the importance weight concentrates along a heart-shaped manifold. Specifically, the base distribution is a mixture of 25 isotropic Gaussians with variance 0.2, arranged on a 5×5 grid over $[-4, 4]^2$. The score model is a 2-layer U-Net trained via DDIM [2].

For clarity, we slightly abuse notation in this subsection; symbols are local to this subsection.

Heart shape. The heart-shaped importance region is defined by the parametric curve $u(\phi) \in \mathbb{R}^2$, where $u(\phi) = \left[16 \sin^3 \phi,\ 13 \cos \phi - 5 \cos(2\phi) - 2 \cos(3\phi) - \cos(4\phi)\right]^\top$ for $\phi \in [0, 2\pi]$. We discretize the curve into $N = 1000$ points as $\phi_n = 2\pi(n-1)/(N-1)$, $n = 1, \ldots, N$, and let $\mathcal{U} = \{u(\phi_n)\}_{n=1}^N$. The importance weight is defined as $w(\mathbf{x}) = \exp\left(-\frac{1}{s} \min_n \|\mathbf{x} - u_n\|^2\right)$, where $u_n \in \mathcal{U}$ and $s = 0.05$. The corresponding target distribution is $q(\mathbf{x}) \propto w(\mathbf{x}) p(\mathbf{x})$.

Butterfly shape. For the experiment setup of the top row in Figure 6, the butterfly-shaped importance region is defined by the parametric curve

$$u(\phi) = a \cdot \rho(\phi) \begin{bmatrix} \cos \phi \\ \sin \phi \end{bmatrix}, \quad \rho(\phi) = \exp(\cos \phi) - 2 \cos(4\phi) + \sin^5\left(\frac{\phi}{12}\right),$$

for $\phi \in [0, 12\pi]$. We set $a = 1.2$ and discretize the curve into $N = 1000$ points: $\phi_n = 12\pi(n-1)/(N-1)$, $n = 1, \ldots, N$. Let $\mathcal{U} = \{u(\phi_n)\}_{n=1}^N$. The importance weight is defined as $w(\mathbf{x}) = \exp\left(-\frac{1}{s} \min_n \|\mathbf{x} - u_n\|^2\right)$, with $s = 0.08$.

Concentric rings. For the experiment setup of the second row in Figure 6, the ring-shaped importance region consists of $K = 4$ concentric circles centered at the origin. The radius of the k-th ring is $\delta_k = k \cdot \delta$ with $\delta = 1.0$ and $k = 1, \ldots, K$. Each ring is parameterized as $u_k(\phi) = [\delta_k \cos \phi,\ \delta_k \sin \phi]^\top$ for $\phi \in [0, 2\pi]$ and discretized into $N = 300$ points. Let $\mathcal{U} = \bigcup_{k=1}^K \{u_k(\phi_n)\}_{n=1}^N$. The importance weight is defined as $w(\mathbf{x}) = \exp\left(-\frac{1}{s} \min_n \|\mathbf{x} - u_n\|^2\right)$, with $s = 0.05$.

Infinity shape. For the experiment setup of the bottom row in Figure 6, the infinity-shaped importance region is defined by $u(\phi) = [\sin \phi,\ \sin \phi \cos \phi]^\top$ for $\phi \in [0, 2\pi]$. We set $a = 4.0$ and discretize the curve into $N = 1000$ points as $u_n = a \cdot u(\phi_n)$, where $\phi_n = 2\pi(n-1)/(N-1)$ for $n = 1, \ldots, N$. Let $\mathcal{U} = \{u_n\}_{n=1}^N$. The importance weight is defined as $w(\mathbf{x}) = \exp\left(-\frac{1}{s} \min_n \|\mathbf{x} - u_n\|^2\right)$, with $s = 0.1$.

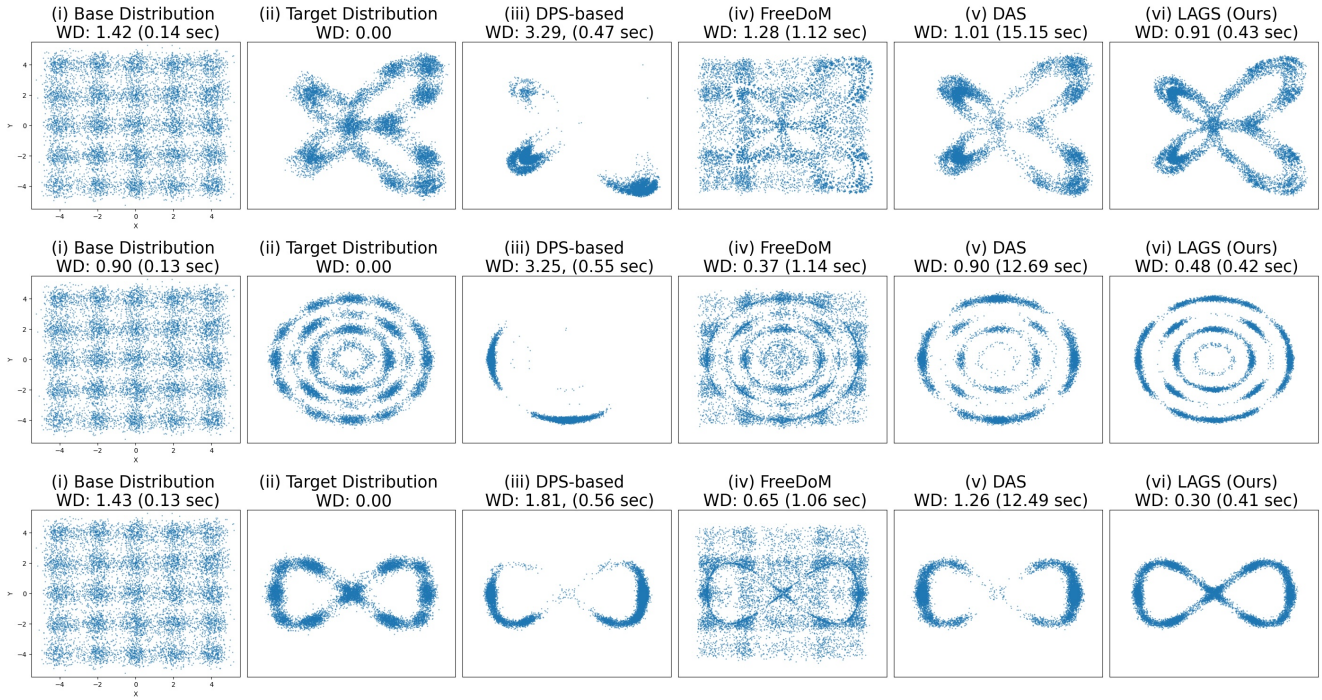

Figure 6. ***Left* to *Right***: Base distribution, target distribution, and estimated sample densities. Wasserstein Distance and runtime (in seconds) are reported for DPS, FreeDoM, DAS, and LAGS (ours). Our method achieves the ***lowest Wasserstein Distance (WD) and fastest runtime***.

Generative model. We consider a lightweight score-based network with (64,64,2) dense layers with ReLU activations for the first two layers. We employ a standard linear schedule for the forward diffusion process: $\beta_k = \beta_{\text{start}} + \frac{k}{K-1}(\beta_{\text{end}} - \beta_{\text{start}})$, for $k = 0, \ldots, K-1$, with $\beta_{\text{start}} = 10^{-4}$ and $\beta_{\text{end}} = 0.02$. We adopt the DDIM sampler [2] and the optimization is performed over 10^4 epochs using the Adam optimizer. The number of forward diffusion steps is set to 10^3. For inference, we subsample 10^2 timesteps for accelerated generation.

Sampling from the Optimal Weighted Sampling Distribution. When the true distribution $p(\mathbf{x})$ is explicitly known and sampling from the base probability density function is possible, we can construct a weighted sampling distribution $q(\mathbf{x})$. This is feasible under the assumption that there exists a known constant M satisfying $w(\mathbf{x}) \leq M, \quad \forall \mathbf{x} \in \mathcal{X}$.

Under these conditions, weighted sampling can be performed via the acceptance-rejection method as follows:

1. Sampling from $p(\mathbf{x})$: Draw $\mathbf{x} \sim p(\mathbf{x})$.
2. Computing Acceptance Probability: Compute $a(\mathbf{x}) = \frac{w(\mathbf{x})}{M}$, ensuring $0 \leq a(\mathbf{x}) \leq 1$.
3. Acceptance/Rejection Step: Draw $u \sim \mathcal{U}(0, 1)$. Accept $\mathbf{x}$ if $u \leq a(\mathbf{x})$; otherwise, reject and repeat.

This acceptance-rejection method for weighted sampling is infeasible in high-dimensional scenarios in general. Specifically, in cases such as high-resolution image generation, the exact domain $\mathcal{X}$ may be unknown, or the acceptance probability $a(\mathbf{x})$ may be extremely small, leading to prohibitively high computational costs.

However, this method remains feasible for the 2D synthetic datasets considered in this section, where the exact domain, base distribution, and weight function are known in closed form. This controlled setting allows us to quantitatively measure the accuracy of weighted sampling and serves as a benchmark for evaluating training-free importance weighting methods.

We evaluate performance using the Wasserstein Distance between the generated samples and the ground-truth target density, as well as the wall-clock time to generate 10^4 samples.

Table 3. CLIP-based diversity (Mean Pairwise Distance). Higher is better. Based on Fig. 2.

SD	DPS	FreeDoM	DAS	DAS-1P	LAGS (Ours)
0.406	0.375	0.387	**0.424**	0.360	0.381

C.2. Experiments of Text-Image Alignment Maximization

C.2.1. Baseline Implementation

We implement several competitive baselines under a unified evaluation framework. DPS method is adapted by directly applying posterior sampling to the weighted sampling formulation, following the derivation presented in Appendix B.4, consistent with the approach in [16]. The guidance control mechanism is integrated based on the FreeDoM framework proposed in [14], but *without any resampling* steps; FreeDoM baseline is used as released, with no modifications, and follows the official hyperparameter setting—including learning rate control—as described in Appendix 2-(3) of [14]; For DAS [19], we employ the released implementation and settings designed for Stable Diffusion, without modification. To further assess computational efficiency, we also use a variant denoted as DAS-1p, where the number of particles is set to 1 and sampling is executed using a single batch. This represents the fastest and most lightweight version of DAS.

Fairness. To ensure a fair comparison across all methods, the weight function is fixed to $8 \times$ PickScore, following the scaling scheme used in [19]. Furthermore, the backward diffusion process is consistently configured to use 100 discrete timesteps for all baselines. This facilitates an evaluation by isolating the effect of each algorithm's guidance strategy. Note that the guidance methods differ in how they approximate the true importance weight or posterior score function, and then they are all constrained to operate under a shared target backward diffusion process.

For evaluations on SD, we adopt prompts used in [19]. For SDXL experiments, we construct a comprehensive prompt set by merging samples from [39] and [19], along with additional manually curated prompts designed to test generalization to diverse semantic configurations.

C.2.2. Additional Results

Weight–diversity trade-off. We examine the trade-off between weight, i.e., reward optimization and sample diversity, focusing on whether weighted sampling degrades the diversity of generated images. Diversity is quantified using the CLIP-based mean pairwise distance, where higher values indicate greater diversity.

As shown in Table 3, most guidance-based methods incur a reduction in diversity relative to the base model, SD. DAS is a notable exception, achieving the highest diversity, likely due to its multi-particle resampling at each diffusion step.

LAGS preserves diversity with a 6.2% relative decrease compared to SD, while still improving reward. These results indicate that LAGS achieves a favorable reward–diversity trade-off without relying on costly multi-particle resampling.

Weight-realism tradeoff. Weighted sampling with human preference scores may exhibit a potential tradeoff between *alignment*, i.e., elevating samples that maximize the target weight function, and quality measures, often translated into realism or perceptual naturalness. When the guidance signal is strong, either due to a sharp curvature of the weight function or large scheduler coefficients, the resulting refinement can occasionally over-amplify semantically meaningful but visually brittle directions, leading to reduced image naturalness. This phenomenon has also been reported in prior analyses of score amplification in diffusion models.

Our method already achieves the highest target metric (PickScore) with the fastest inference time; nevertheless, it is desirable to examine whether realism can be further improved without compromising alignment much. To this end, we adopt a lightweight adaptation of the *Adaptive Projected Guidance* (APG) mechanism proposed by Sadat et al. 2025 [81], originally developed to suppress over-saturation artifacts in diffusion guidance.

Consider the deterministic DDIM update and its corresponding drift direction as

$$\boldsymbol{\mu}_k = \sqrt{\bar{\alpha}_{k-1}}\,\hat{\mathbf{x}}_{0,k} + \sqrt{1-\bar{\alpha}_{k-1}}\,\hat{\boldsymbol{\epsilon}}_k, \qquad \mathbf{u}_k = \frac{\boldsymbol{\mu}_k - \mathbf{x}_k}{\|\boldsymbol{\mu}_k - \mathbf{x}_k\|_2}. \tag{59}$$

Based on this, we can decompose the guidance into a component parallel to this direction and a component orthogonal to it as

$$\mathbf{g}_k^{\parallel} = \frac{\langle \tilde{g}^{(1)}(\mathbf{x}_k, t_k),\, \mathbf{u}_k \rangle}{\|\mathbf{u}_k\|_2^2}\,\mathbf{u}_k, \qquad \mathbf{g}_k^{\perp} = \tilde{g}^{(1)}(\mathbf{x}_k, t_k) - \mathbf{g}_k^{\parallel}, \qquad \hat{\mathbf{g}}_k = \mathbf{g}_k^{\perp} + \varsigma\,\mathbf{g}_k^{\parallel}. \tag{60}$$

We set $\varsigma = 0$ and the computational overhead of this projection and scaling is negligible with below 0.5% of sampling time.

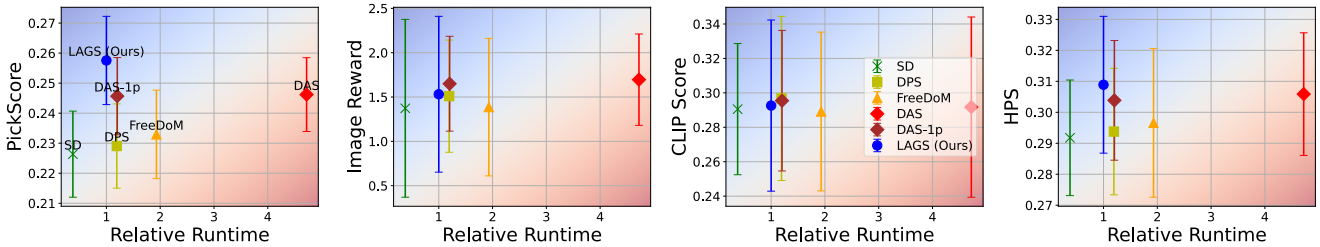

Figure 7. **Results on SDXL**: Evaluation on the same metrics as in Figure 2. LAGS is implemented with the projected guidance (60).

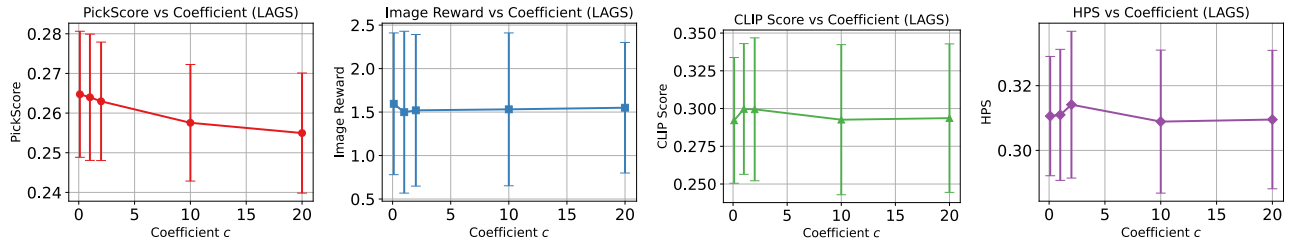

Figure 8. Performance over the choice of c. **Columns (left to right):** PickScore, Image Reward, CLIP Score, and HPS. Each subplot plots metric values against scheduler coefficient $c \in [0.1, 20]$.

Metric	SDXL (base)	DPS	FreeDoM	DAS	DAS-1P
Runtime ↓	0.377	1.193	1.930	4.722	1.202
BRISQUE ↓	21.48 ± 12.45	24.52 ± 13.03	34.49 ± 15.20	**19.55 ± 12.75**	22.56 ± 11.97
MANIQA ↑	0.733 ± 0.139	0.720 ± 0.139	0.711 ± 0.144	0.720 ± 0.145	0.728 ± 0.130

Metric	LAGS (c=0.1)	LAGS (c=1)	LAGS (c=2)	LAGS (c=10)	LAGS (c=20)
Runtime ↓	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
BRISQUE ↓	25.04 ± 15.56	21.88 ± 12.50	21.24 ± 12.18	19.87 ± 13.97	20.12 ± 14.39
MANIQA ↑	**0.745 ± 0.119**	0.737 ± 0.131	0.744 ± 0.132	0.732 ± 0.131	0.728 ± 0.132

Table 4. Runtime and no-reference image quality measures (BRISQUE, MANIQA) for LAGS (ours) with different guidance strengths c. Values are mean ± standard deviation.

Results. Using $c = 10$, we compare LAGS against existing guidance methods in Figure 7. LAGS continues to achieve the highest PickScore and HPS among all methods, while maintaining competitive ImageReward and CLIP Score. Importantly, LAGS remains on the Pareto frontier of *both* alignment (PickScore, HPS) and efficiency: it simultaneously attains the strongest alignment performance and the lowest runtime among all guidance-based sampling approaches.

As shown in Figure 8, increasing the scheduler parameter c reduces the magnitude of the guidance and thus decreases PickScore monotonically. The other metrics such as ImageReward, CLIP Score, and HPS exhibit non-monotonic behavior. Remarkably, with the wide range of $c \in [0.1, 20]$, our method remains on the Pareto frontier across PickScore, HPS, and runtime.

We further analyze the image quality measured by BRISQUE and MANIQA in Table 4. Notably, despite achieving the fastest runtime, the proposed method attains the highest MANIQA score among all guidance methods. This indicates that, although there exists a trade-off between perceptual quality and the target PickScore, our method achieves the best overall performance across three axes: runtime, image quality, and alignment with the target weighting function.

For BRISQUE, DAS achieves the lowest score (corresponding to the highest perceptual quality). Nevertheless, the proposed method yields competitive performance, with BRISQUE values ranging from 19.87 to 25.04, indicating consistently acceptable image quality.

In the top row of Figure 9, we show images generated by LAGS for different scheduler coefficients $c \in \{20, 10, 1, 0.1, 0.01\}$ (left to right). As c decreases and the sampler relies more heavily on the approximated guidance, the model progressively emphasizes the weight function, leading to a stronger presence of the blue-toned features encouraged by the target score.

In the bottom row, we apply the projected guidance variant using $\mathbf{g}_k^{\perp}$, which removes the original DDIM direction from

Figure 9. **Prompt:** *"A blue-furred tiger on a glacier."* Each column shows images generated by LAGS with scheduler coefficients $c \in \{20, 10, 1, 0.1, 0.01\}$ (left to right). Smaller values of c increase reliance on the approximated guidance. **Top row:** standard LAGS. **Bottom row:** projected guidance variant using $\mathbf{g}_k^{\perp}$, which removes the DDIM direction from the guidance term. As c decreases, standard LAGS amplifies the blue-toned weighting effect, while projected guidance produces softer and more natural textures.

Figure 10. Images generated by baseline methods—SD, DPS, FreeDoM, DAS-1p, and DAS (left to right)—using the same prompt and seed as Figure 9. Among the baselines, DAS (rightmost column) achieves the highest PickScore (0.256) but requires more than $3\times$ the computational cost of our method.

the guidance term. This modification reduces the influence of the color-specific weighting while producing noticeably softer and more natural textures. For example, at $c = 0.01$, the projected version yields smoother and more coherent structures, whereas the standard LAGS output exhibits stronger but slightly harsher color contrast.

Figure 10 presents results from the baseline methods (SD, DPS, FreeDoM, DAS-1p, DAS) generated with the same prompt and random seed. Among these baselines, DAS achieves the highest PickScore (0.256) and correctly captures the "blue-furred" appearance, but requires more than $3\times$ the computational cost of our method, highlighting the efficiency advantage of LAGS.

C.2.3. Ablation Studies

We introduced an uncertainty-based scheduler parameterized by c, which modulates the strength of the guidance signal across the diffusion trajectory. Importantly, the scheduler converges to 1 regardless of the specific choice of c, ensuring that the model increasingly relies on the approximated guidance term near the terminal phase of the diffusion process.

The role of the parameter c is to control the steepness and onset of the scheduler. A smaller c yields a stronger guidance signal and an earlier rise in the scheduler curve (See Appendix B.2), whereas a larger c results in a weaker signal and delays its saturation, resembling a sharper sigmoid. In this subsection, we conduct ablations across a range of c values to understand their influence on generation quality and alignment.

Figure 11. **Prompts (top to bottom):** *"A green colored rabbit", "A red Ford Mustang in the style of Starry Night by Van Gogh"*. **Each column:** Images generated by LAGS with different scheduler parameters $c = 100, 10, 1, 0.1, 0.01, 0.001$ (left to right). Smaller c indicates high reliance on the approximated guidance.

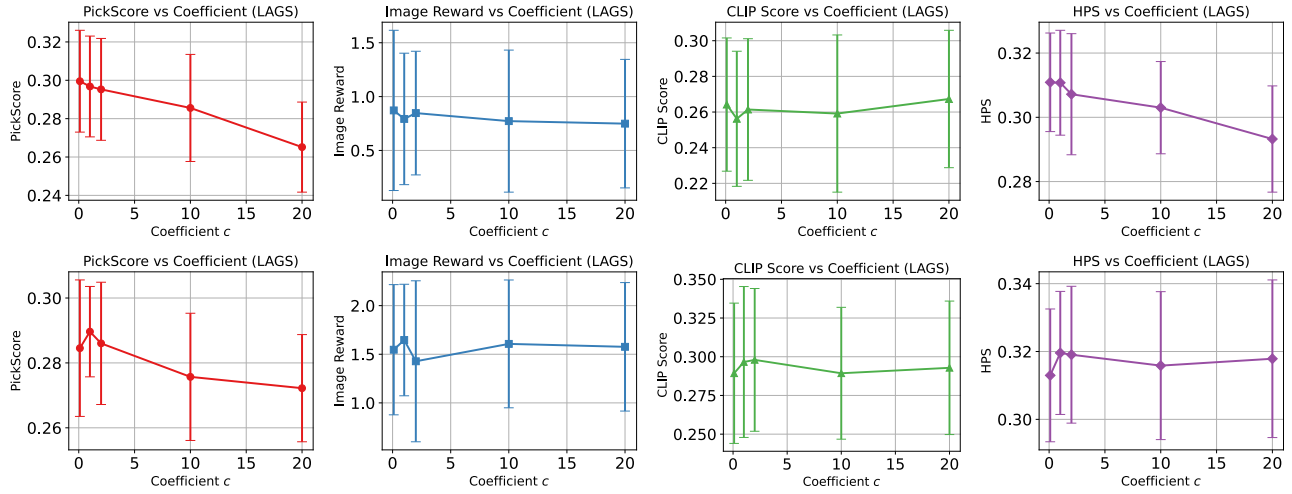

Figure 12. Performance over the choice of c. **Columns (left to right):** PickScore, Image Reward, CLIP Score, and HPS. **Rows:** Top – Stable Diffusion (SD), Bottom – SDXL. Each subplot plots metric values against scheduler coefficient $c \in [0.1, 20]$.

Visualization with varying c. Figure 11 visualizes the effect of varying c. As expected, smaller values of c produce stronger guidance, resulting in stronger alignment between the image and the prompt semantics. In contrast, large c values (e.g., $c = 100$) often fail to capture key prompt attributes. For instance, the leftmost column does not preserve the color cue ("green" rabbit) or the object category ("red car"). Conversely, the rightmost column ($c = 0.001$) yields images that are semantically faithful to the prompts and achieve the highest PickScore among the variants.

As c increases, the generated images tend to gradually emphasize visual features, particularly evident in color fidelity for the first row. The leftmost column shows the failure mode of generated images with a missing color property, semantic entity. As c increases, with higher probability, the generated samples show high alignment with the given texts. This behavior may vary depending on the prompt and the content of the generated image or the given conditional texts, pointing to potential discrepancies between quantitative alignment metrics and visual perception [42], especially for some specific set of prompts or samples.

Performance cross validation with varying c. We test the entire prompt set with various c values and measure the target PickScore along with the different alignment metrics to examine the performance transition.

Figure 12 summarizes the empirical behavior of PickScore under different choices of the scheduler coefficient c. Since larger c attenuates the contribution of the guidance term $\tilde{g}(\mathbf{x}, t)$, increasing c naturally reduces the influence of the weighted-sampling direction during the backward diffusion process. This manifests as a consistent downward trend in PickScore, i.e., the average target weight $\mathbb{E}[w(\mathbf{x})]$. For SD, this decay is strictly monotonic. For SDXL, the trend is similar, with $c = 1$ achieving the highest PickScore before gradually declining for larger values of c.

This monotonic behavior in PickScore does not always mirror movement in the other alignment metrics such as CLIP Score, HPS, or ImageReward. Although guidance-based methods, including ours, reliably improve the target-specific objective (PickScore), these metrics occasionally exhibit misalignment. This reflects inherent metric heterogeneity rather than a limitation of the proposed method and baselines, and suggests that more general alignment-oriented weight functions may be needed to simultaneously elevate all alignment metrics.

Across a broad range of scheduler strengths, the proposed method consistently achieves the highest PickScore among all training-free guidance approaches, reaffirming its ability to reliably increase the expected weight $\mathbb{E}[w(\mathbf{x})]$. Empirically, we observe that even moderate values of c retain strong performance, demonstrating robustness of the lightweight approximation and validating the effectiveness of the uncertainty-adaptive scheduling strategy.

C.3. Experiments on CelebA Dataset

This section examines an application of the proposed weighted sampling method in scenarios where the weight function is parameterized by a neural classifier. Consider a setting in which an SGM is trained without any class supervision, for example, without labels related to gender in the CelebA dataset. Although the SGM has no inherent notion of class, we assume access to an external binary classifier that predicts whether an image is labeled as "man" or "not man."

The goal is to increase the sampling probability of a particular class. This is especially relevant in situations where the base SGM exhibits class imbalance or bias; in such cases, classifier-driven weighted sampling can adjust the output distribution and mitigate these biases.

To implement this, the weight function $w(\mathbf{x})$ is defined using the classifier's logit output. Images that the classifier is more confident belong to the target class (e.g., "man") are assigned larger weights, thereby steering the sampling process toward that class through the proposed weighted sampling mechanism.

Dataset. The CelebA dataset [82] is utilized for training the classifier in our experiments. For the score function used in the generation process, we employ a pretrained score model based on DDPM [1]. This pretrained model is obtained from Huggingface[1] and is capable of generating RGB images at a resolution of 256×256 pixels.

Weight function. The design of the weight function begins with training a classifier based on the ResNet18 architecture using the CelebA dataset [82]. The classifier is specifically trained to predict whether a given instance is classified as "man" or not, utilizing the provided labels. The final activation layer of the classifier employs a sigmoid function to produce probabilistic outputs. During training, we use a batch size of 256, a total of 25 epochs, and the Adam optimizer with a learning rate of 10^{-3}. We denote this classifier as F_{cl}, where $F_{\text{cl}} : \mathbb{R}^d \mapsto \{0, 1\}$.

Assume that the objective of weighted sampling is to increase the probability density of samples classified as "man." To construct the weight function $w(\mathbf{x})$, we adopt an approach similar to that described in Appendix C.4. Specifically, the weight function is defined as $w(\mathbf{x}) = D_{\text{bce}}(F_{\text{cl}}(\mathbf{x}), 0)$, where D_{bce} represents the scaled binary cross-entropy loss function. Intuitively, this formulation indicates that the weight $w(\mathbf{x})$ takes higher values when the instance $\mathbf{x}$ is classified as "man."

Results. By applying the neural classifier-based weighting function, the proportion of instances classified as "man" increased from 35.1% to 51.2%. This result empirically validates that our proposed weighted sampling approach can effectively modulate the output distribution by leveraging a classifier that is independently trained from the generative model. Notably, this experiment highlights the scalability of our method, demonstrating its applicability irrespective of the generative model's size.

Figure. 13 illustrates pairs of two diffusion processes: the top row corresponds to the standard backward diffusion process, while the bottom represents the weighted backward diffusion process achieved using our proposed approach. From left to right, each column depicts discrete timesteps $800, 600, 400, 200, 0$, respectively, out of a total of 1000. The rightmost column, therefore, represents the final generated samples drawn from the distributions $p(\mathbf{x})$ and $q(\mathbf{x})$.

[1] https://huggingface.co/google/ddpm-celebahq-256

Figure 13. **Pairs of Two Diffusion Backward Processes.** ***Top***: The standard backward diffusion process for sampling from $p(\mathbf{x})$. ***Bottom***: The proposed backward diffusion process for weighted sampling from $q(\mathbf{x})$. The proposed method leverages an externally defined weight function, such as a neural classifier, to prioritize samples with higher weight. This approach can accommodate *any differentiable weight function, i.e., neural classifiers*, for weighted sampling—entirely independent of the base SGM—without requiring additional training.

In Figure 13, the class outputs of the two diffusion processes differ and show that the weighted sampling mechanism guides the backward process toward generating samples classified as "man." Importantly, both processes share identical sources of randomness, including the same initial state and identical diffusion process noise, with the only distinction being the difference in score functions.

The results show promise for advancing AI fairness by mitigating class bias introduced by generative models with fast guidance. Additionally, it can be effectively employed for class-specific filtering tasks, such as identifying and removing inappropriate images containing undesirable content. These capabilities underscore the versatility and utility of the proposed method in various practical applications.

This method represents a distinct and efficient alternative to density-estimation-based fairness derivation techniques, such as [83], which require extensive retraining to achieve fairness adjustments. We anticipate that our training-free weighted sampling approach will pave the way for new advancements in fairness, a goal of growing importance in the development of machine learning models [84–86].

C.4. Experiments on MNIST Dataset

When using a pretrained SGM, the generated samples may inherently exhibit class-wise biases. In certain applications, it is crucial either to mitigate such biases to ensure fairness or to intentionally introduce biases to suppress specific classes, such as filtering out potentially harmful or undesirable categories.

These problems can be naturally formulated within a weighted sampling framework by defining a weight function $w(\mathbf{x})$ that assigns higher values to samples that do not belong to the target class to be suppressed.

Furthermore, a similar approach can be employed to promote class-wise uniformity in the generated distribution. By assigning higher weights to underrepresented classes, the sampling process can be adjusted to achieve a more balanced class distribution, thereby approximating a uniform class-wise representation.

Dataset. We utilize the MNIST dataset [87], which comprises ten classes of handwritten digits. Each sample is normalized to the range $[-1, 1]$ for both training the neural classifier and the score function of the original PDF.

Weight function. In this experiment, a neural classifier-based weight function is employed. Specifically, we utilize ResNet18 [88] as the backbone classifier, with its first layer modified to accommodate the single-channel input structure of the MNIST dataset. The classifier is trained using the Adam optimizer with a learning rate of 1×10^{-3}, a batch size of 256, and for 100 epochs. This configuration yields a neural classifier capable of achieving a classification accuracy of 99.4% on the MNIST test set. We denote the trained classifier as $F_{\text{cl}}(\mathbf{x})$, where $F_{\text{cl}} : \mathbb{R}^d \mapsto \mathbb{R}^{10}$, i.e., the classifier maps each input sample $\mathbf{x}$ to a 10-dimensional logit vector.

The softmax output of the logits is expected to exhibit a high value at the index corresponding to the correct label. Using this neural classifier, we construct a weight function $w(\mathbf{x})$ to selectively downweight specific labels. For instance, consider a scenario where the objective is to downweight the probability of sampling a particular label $l \in \{0, 1, \ldots, 9\}$. Let $\mathbf{e}_l$ denote a one-hot vector with a value of 1 at the l-th position (0-indexed) and zeros elsewhere. Setting $w(\mathbf{x}) = D_{\text{ce}}(F_{\text{cl}}(\mathbf{x}), \mathbf{e}_l)^2$,

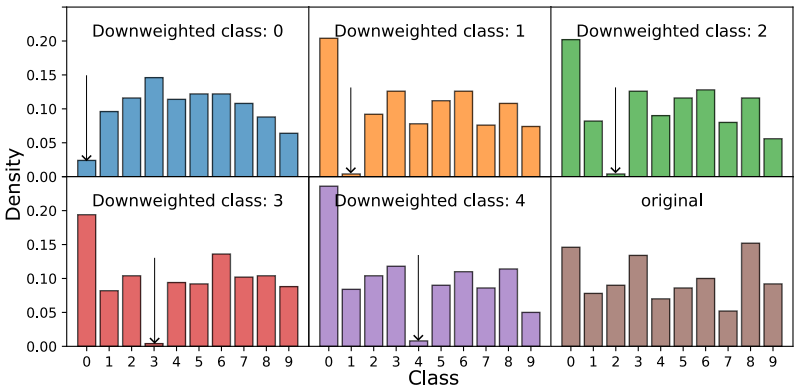

Figure 14. Class histograms from the original generative model trained without class labels (bottom right) and from the proposed weighted sampling (first five histograms, left to right, top to bottom) using a neural network classifier-based importance weight. Samples with lower logits for the specified downweighted class are assigned higher importance weights.

where D_{ce} represents the cross-entropy loss, ensures that samples with lower logit values at the l-th index are assigned higher importance weights. Consequently, weighted sampling with this weight function increases the likelihood of sampling samples that are less likely to be classified as label l.

Assume we wish to downweight class $l \in [9]$. We define the weight function as $w(\mathbf{x}) = D_{\text{ce}}(F_{\text{cl}}(\mathbf{x}), \mathbf{e}_l)^2$ where $\mathbf{e}_l$ is a one-hot encoded vector with the l-th element set to one and all others set to zero, and D_{ce} represents the cross-entropy loss. This weight function assigns greater significance to samples yielding small logit value for the l-th label, thereby effectively downweighting class l.

Base score function. The score function of the original PDF, $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$ is derived through DDPM training. The score function is implemented using a U-Net-based architecture. The architecture consists of three downsampling blocks and three corresponding upsampling blocks, organized in a symmetrical encoder-decoder configuration. The channel dimensions for the blocks increase progressively, with depths set to (32, 64, 128), respectively. To ensure stable and efficient training, group normalization is applied with four groups per feature map. Furthermore, a padding value of 1 is used during downsampling operations. The model accepts input tensors $\mathbf{x}_t$ and the diffusion time step t, and outputs the denoised sample prediction.

After obtaining the score function of the original PDF, DDPM sampling is utilized for the implementation of weighted sampling.

Results. In Figure 14, the first five histograms (from left to right, top to bottom) correspond to $l = 0$ to $l = 4$, while the bottom-right histogram displays results without weighted sampling. These histograms are generated by the neural classifier $F_{\text{cl}}(\mathbf{x})$, by sampling 10^4 instances followed by classification through $F_{\text{cl}}(\mathbf{x})$. The results demonstrate that the proposed weighted sampling technique effectively attenuates the representation of target classes relative to the original distribution by assigning greater weight to instances classified outside of the target class. Note that this weighted sampling was achieved without additional model training for different values of l, relying solely on small modifications to l.

The left image in Figure 15 illustrates 100 samples drawn from the original PDF $p(\mathbf{x})$ using its score function, $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$, in conjunction with the DDPM sampling method. We maintain consistent randomness across experiments, i.e., identical initial states and noise values for the stochastic backward process. The right image in Figure 15 shows importance samples generated using the proposed weighted sampling method, wherein the weight function is defined as $w(\mathbf{x}) = D_{\text{ce}}(F_{\text{cl}}(\mathbf{x}), \mathbf{e}_0)^2$. This weighting mechanism assigns higher importance to samples with lower logit values for class 0.

A key observation from Figure 15 is the marked reduction in the frequency of samples classified as class 0—only 3 out of 100—compared to the original distribution. This result demonstrates the efficacy of the proposed weighted sampling method in selectively reducing the likelihood of sampling instances associated with a specified class. The approach leverages a neural classifier as a main component in defining the weight function, highlighting its flexibility in terms of designing the weight function.

In Figure 16, we illustrate the outcomes of employing different weight functions, denoted as $w(\mathbf{x}) = D_{\text{ce}}(F_{\text{cl}}(\mathbf{x}), \mathbf{e}_1)^2$, $w(\mathbf{x}) = D_{\text{ce}}(F_{\text{cl}}(\mathbf{x}), \mathbf{e}_2)^2$, $w(\mathbf{x}) = D_{\text{ce}}(F_{\text{cl}}(\mathbf{x}), \mathbf{e}_3)^2$, $w(\mathbf{x}) = D_{\text{ce}}(F_{\text{cl}}(\mathbf{x}), \mathbf{e}_4)^2$, from left to right, respectively. These

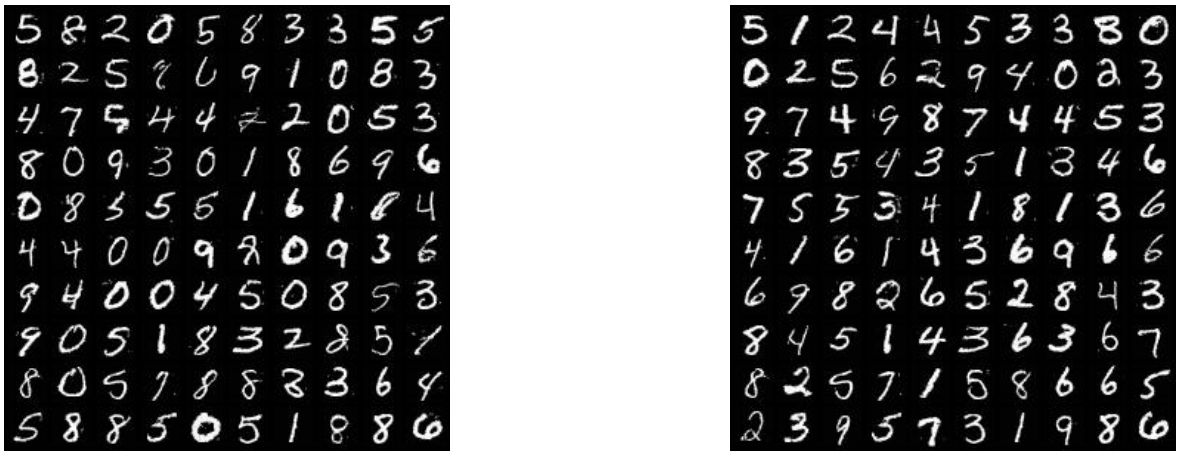

Figure 15. ***Left***: Random samples from $p(\mathbf{x})$. ***Right***: Random samples from $q(\mathbf{x})$ with *higher importance weights for instances with low values at the 0-th logit* of the neural classifier. Those instances in identical positions in the two figures share the same randomness (i.e., the initial state and noise for the diffusion backward processes). In the original distribution, approximately 15% of the instances are classified with label 0. Through weighted sampling, the density of instances with label 0 can be significantly reduced. On the right, only 3 out of 100 instances are assigned label 0 under weighted sampling.

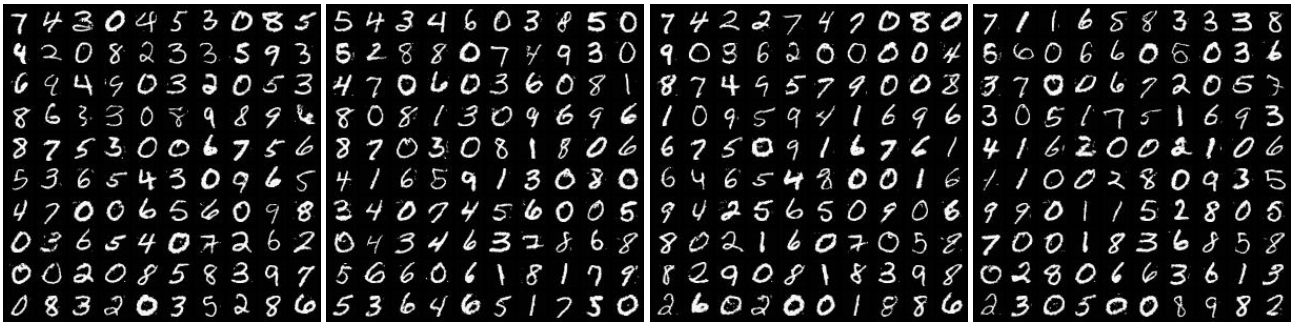

Figure 16. ***From left to right***: Importance samples generated using the weight function $w(\mathbf{x}) = D_{\text{ce}}(F_{\text{cl}}(\mathbf{x}), \mathbf{e}_l)^2$, where $l = 1, 2, 3, 4$, respectively. For each case, higher importance weights are assigned to instances with low logit values corresponding to class l. For instance, in the first image, no instances classified as class 1 are present. Similarly, the second image contains only one instance classified as class 2, while the third and fourth images lack instances classified as classes 3 and 4, respectively. This is in contrast to the set of samples from the original distribution shown in Figure 15.

functions assign higher importance weights to instances with lower logit values for the respective classes $l = 1, 2, 3, 4$. Consistent with Figure 15, the examples demonstrate that the proportion of instances classified into each specified class is less than 3% over 100 generated samples which show the neural classifier driven weight function can be utilized to downweight specific set of instances' sampling probability.

Weighted sampling towards unbiased class probability The weighted sampling methodology can also be employed to mitigate bias and enhance class-wise fairness. For instance, consider a scenario where the objective is to control the class probabilities to achieve a uniform distribution. Let $\mathbf{p} \in [0, 1]^{10}$ represent the vector of class probabilities, where the l-th element, $[\mathbf{p}]_c$, corresponds to the probability density of class l from the original generative model. These probabilities can be readily obtained using a pretrained score-based generative model in conjunction with a classifier, as illustrated in the histogram at the bottom right of Figure 14, by generating samples and classifying them via the classifier.

An intuitive approach to achieve class-wise uniformity is to define a weight function $w(\mathbf{x})$ such that if $\mathbf{x}$ is classified as class l, the importance weight is given by $1/[\mathbf{p}]_c$, where $[\mathbf{p}]_c$ is the l-th element of the vector $\mathbf{p}$. By employing this weighting mechanism, the effective importance weight for each class is equalized, thereby leading to a class-wise distribution with significantly improved uniformity.

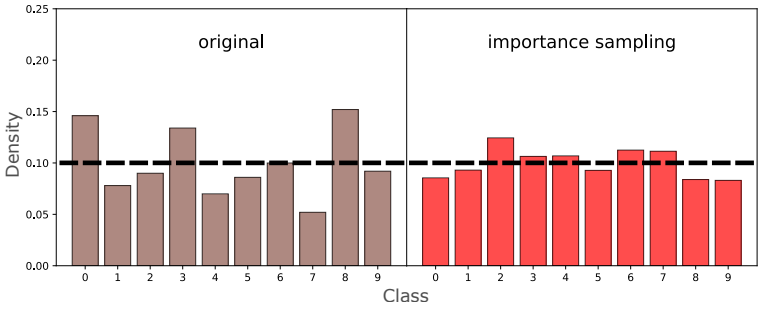

Figure 17. ***Left***: Class probability distribution from the original score-based generative model. ***Right***: Class probability distribution from weighted sampling. The bias across classes is significantly reduced, with the density variance narrowing from 10^{-3} to 1.9×10^{-4}, demonstrating the effectiveness of the proposed method in balancing the class densities.

To implement this approach, we utilize the Gumbel-softmax function F_G, which processes the logit outputs of the classifier to approximate the behavior of an argmax function. This allows us to reweight the logits by the inverse of $\mathbf{p}$. Specifically, the weight function is expressed as: $w(\mathbf{x}) = F_G(F_{\text{cl}}(\mathbf{x}))^\top \left(\frac{1}{\mathbf{p}}\right)$, where $F_{\text{cl}}(\mathbf{x})$ denotes the logit output of the classifier.

In Figure 17, we present the class probability density distribution generated by the original score-based generative model on the left and the class probability density obtained via weighted sampling using $w(\mathbf{x}) = F_G(F_{\text{cl}}(\mathbf{x}))^\top \left(\frac{1}{\mathbf{p}}\right)$ on the right. Notably, the bias in the class probability density across the classes is significantly reduced, with the class density variance decreasing from 10^{-3} to 1.9×10^{-4}. This demonstrates that control over class density can be effectively achieved by simply modifying the weight function.

C.5. Experiments on Stable Cascade

In this section, we explore various weight functions that emphasize different image attributes, such as color and frequency components. By leveraging these functions, our framework provides an additional layer of control over sampling in foundation diffusion models beyond text-prompt conditioning. Specifically, we examine how our weighted sampling method enables controlled variation in the generated distribution, even when the text prompt lacks explicit color, frequency, or stylistic attributes.

It should be noted that the examples provided in this section should not be interpreted as image transformations. The objective of weighted sampling is to adjust the distribution while remaining within the original support set, rather than altering individual images.

Model. We utilize StableCascade model as our foundation SGM [89]. This model was trained on a curated subset of LAION-5B. We adopt its default DDIM-based sampling process [2] and integrate our method with the weight functions defined below.

C.5.1. Color-biased Sampling

Weight function. To emphasize specific colors in generated images, we define weight functions that bias the image toward one of the RGB color channels. An image, represented as $\mathbf{x} \in \mathbb{R}^{3\times h\times w}$, contains red, green, and blue channels. The average intensity for each channel is computed as

$$I_r(\mathbf{x}) \coloneqq \frac{1}{h\cdot w}\sum_{i=1}^{h}\sum_{j=1}^{w}[\mathbf{x}]_{1,i,j}, \quad I_g(\mathbf{x}) \coloneqq \frac{1}{h\cdot w}\sum_{i=1}^{h}\sum_{j=1}^{w}[\mathbf{x}]_{2,i,j}, \quad I_b(\mathbf{x}) \coloneqq \frac{1}{h\cdot w}\sum_{i=1}^{h}\sum_{j=1}^{w}[\mathbf{x}]_{3,i,j}.$$

For instance, to enhance red, we use $w_r(\mathbf{x};\xi) = \exp\left(\xi \cdot \text{sig}\left(I_r(\mathbf{x}) - \frac{I_g(\mathbf{x})+I_b(\mathbf{x})}{2}\right)\right)$, where sig is the sigmoid function, and ξ determines the strength of the bias. Analogously, $w_g(\mathbf{x};\xi)$ and $w_b(\mathbf{x};\xi)$ are defined for green and blue channels, respectively, allowing flexible control over the generated image's color emphasis.

Results. In Figure 18, we present weighted sampling results, where the top row corresponds to w_r, the second row to w_g, and the bottom row to w_b. The weight function is applied with varying values of $\xi \in \{10, 20, 30, 50, 100, 150\}$ from left to right, enabling controlled emphasis on each color channel.

Figure 18. Images generated using the prompt *"An ocean with a beach"* for each RGB color channel. The rows correspond to the Red, Green, and Blue channels, respectively, while the columns represent increasing values of the parameter $\xi = (10, 20, 30, 50, 100, 150)$ from left to right. Lower ξ values produce typical colors, whereas higher ξ values enhance the intensity of the corresponding color channel. Notably, this control is achieved solely through weighted sampling, ***without*** *use of any conditional prompts related to the channels or colors*.

By adjusting ξ, we observe that the generated images progressively shift toward the target color channel while maintaining a fixed text prompt, *"An ocean with a beach."*, ***without any color-related text conditioning***. At low ξ values, the images retain natural colors, whereas higher values progressively enhance the target channel, producing visually distinct outputs. This demonstrates the effectiveness of our method in biasing the generated distribution based on externally defined weight functions, independent of textual context.

C.5.2. Sampling High-Spatial-Frequency Images from Foundation Diffusion Models

Weight function. To control the frequency content of generated images for each RGB channel, we leverage the 2D Fourier transform to separate high- and low-frequency components. Starting with an image $\mathbf{x} \in \mathbb{R}^{3\times h\times w}$, its three color channels are processed separately. The frequency representations for each channel are computed as

$$\mathbf{f}_r(\mathbf{x}) \coloneqq \mathcal{F}([\mathbf{x}]_{1,i,j}), \quad \mathbf{f}_g(\mathbf{x}) \coloneqq \mathcal{F}([\mathbf{x}]_{2,i,j}), \quad \mathbf{f}_b(\mathbf{x}) \coloneqq \mathcal{F}([\mathbf{x}]_{3,i,j})$$

where $\mathcal{F}(\cdot)$ represents the 2D Fast Fourier Transform (FFT), and the output is FFT-shifted to center low frequencies. To differentiate between low- and high-frequency components, we define a distance matrix $D(i, j)$ as

$$D(i,j) \coloneqq \sqrt{(i - h/2)^2 + (j - w/2)^2}$$

where $D(i, j)$ measures the distance of each frequency component from the center of the FFT-shifted representation. This matrix is shared across all channels. Using this distance matrix, masks are constructed to emphasize high- or low-frequency components with a smooth transition. Specifically, the high-frequency mask is defined as

$$\mathbf{m}_{\text{high}}(i,j) \coloneqq \frac{1}{1 + \exp\left(-\frac{D(i,j)-r}{s}\right)}$$

where r is the cutoff radius (e.g., $r = \frac{\min(h,w)}{4}$) and s controls the smoothness of the transition. This mask assigns higher values to frequency components farther from the center, emphasizing high frequencies. Similarly, the low-frequency mask is defined as

$$\mathbf{m}_{\text{low}}(i,j) \coloneqq \frac{1}{1+\exp\left(\frac{D(i,j)-r}{s}\right)}$$

which emphasizes components closer to the center (low frequencies). To quantify the emphasis on specific frequency ranges, we compute the relative strength of high- versus low-frequency components by directly applying the selected mask to the sum of the frequency representations of each channel using element-wise multiplication ($\odot$):

$$w(\mathbf{x};\xi) = \exp\left(\xi \cdot \frac{\sum_{i,j} |(\mathbf{f}_r(\mathbf{x}) + \mathbf{f}_g(\mathbf{x}) + \mathbf{f}_b(\mathbf{x})) \odot \mathbf{m}_{\text{high}}|_{i,j}}{\sum_{i,j} |(\mathbf{f}_r(\mathbf{x}) + \mathbf{f}_g(\mathbf{x}) + \mathbf{f}_b(\mathbf{x})) \odot \mathbf{m}_{\text{low}}|_{i,j}}\right).$$

where ξ controls the bias. For $\xi > 0$, high-frequency components are emphasized, enhancing textures and details. For $\xi = 0$, there is neutral sampling without frequency emphasis. For $\xi < 0$, low-frequency components are emphasized, enhancing smoothness and gradients. This unified weight function $w(\mathbf{x};\xi)$ allows a single formulation for both high- and low-frequency emphasis based on the value of ξ.

Results. Figure 19 presents images generated with different random seeds for each $\xi \in \{-30.0, 0.0, 7.0\}$, corresponding to the left, middle, and right columns, respectively. The prompt *"A castle under a cloudy sky"* was used, which does not contain explicit stylistic descriptors.

For $\xi < 0$, where images with dominant low-frequency components receive higher importance, the generated images exhibit more prominent cloud structures while the castle appears smaller. This observation is consistent with the fact that cloud formations typically contain gradual pixel variations, which contribute predominantly to low-frequency components. Conversely, for $\xi > 0$, where high-frequency components are emphasized, the images display sharper edges, more pronounced outlines, and increased contrast with simplified color transitions.

Figure 20 further illustrates the impact of weighted sampling using different text prompts: *"A woven basket filled with fruit"* (top row), *"An old-fashioned pocket watch"* (middle row), and *"An astronaut"* (bottom row). For each prompt, weighted sampling is performed with $\xi \in \{-30.0, 0.0, 7.0\}$, corresponding to the left, middle, and right columns, respectively. Higher values of ξ assign greater importance to samples with dominant high-frequency components.

A similar trend is observed in Figure 20, where images generated with low-frequency emphasis exhibit blurred or simplified details. For instance, in the case of the pocket watch, low-frequency emphasis results in an image without visible tick marks, reflecting the predominance of smooth, low-frequency components. In contrast, high-frequency emphasis enhances fine details, producing images with well-defined edges and a pen-illustration-like appearance.

In Figure 21, the top row shows five randomly generated samples from the original diffusion backward process using the prompts: *"An adorable cat," "An ice cream cone," "A snowy mountain," "A steaming cup of coffee,"* and *"A castle under a cloudy sky."* The middle row illustrates results produced by our weighted sampling method, which prioritizes instances with higher high-frequency components, following the same methodology as employed in the preceding examples in this section. The bottom row presents the frequency component analysis obtained via the Fourier transform.

The results indicate that our weighted sampling method effectively enhances high-frequency components, leading to the generation of images with more pronounced edges, often resembling a pen illustration style. The frequency component plots in the bottom row depict the red-shaded regions, which represent the increased magnitude of high-frequency components induced by our weighted sampling approach. As evident across all samples, our method consistently amplifies these components, demonstrating that the weighted sampling technique directs the foundational diffusion model to generate samples enriched with high-frequency details through an externally defined importance weighting function.

Crucially, the images produced via our weighted sampling method exhibit stylistic modifications without any explicit textual conditioning related to the appearance or characteristics of the generated images. The transformation is achieved purely through the externally defined frequency emphasis function, while maintaining the exact same textual prompts as those used in the baseline sample generation. As such, the proposed approach opens the door to incorporating more sophisticated, task-specific importance functions, enabling controllable and flexible guidance without the need for prompt engineering or additional model training.

$\leftarrow$
$w(\cdot;\xi = -30.0)$

$w(\cdot;\xi = 0.0)$

$\rightarrow$
$w(\cdot;\xi = 7.0)$

Figure 19. ***Left*:** Images generated with the prompt *"A castle under a cloudy sky,"* emphasizing low-frequency details ($w(\cdot;\xi = -30.0)$). These images exhibit smoother gradients and softer transitions, highlighting subtle atmospheric effects and giving a calm, natural feel. ***Center*:** Neutral images without frequency bias ($w(\cdot;\xi = 0.0)$), showcasing the original generative capabilities of the model with diverse interpretations of the prompt. ***Right*:** Images generated with the same prompt, emphasizing high-frequency details ($w(\cdot;\xi = 7.0)$). These images display sharper textures and vivid contrasts, producing a slightly stylized, cartoon-like appearance. This comparison demonstrates how adjusting the emphasis on high or low frequencies can yield a wide variety of results ***without*** prompt condition related to the feature.

Figure 20. ***Left*:** Images generated with low-frequency emphasis ($\xi = -30.0$). ***Center*:** Neutral images without frequency bias ($\xi = 0.0$). ***Right*:** Images generated with high-frequency emphasis ($\xi = 7.0$). Each row corresponds to a different prompt: *"A woven basket filled with fruit," "An old-fashioned pocket watch,"* and *"An astronaut,"* respectively.

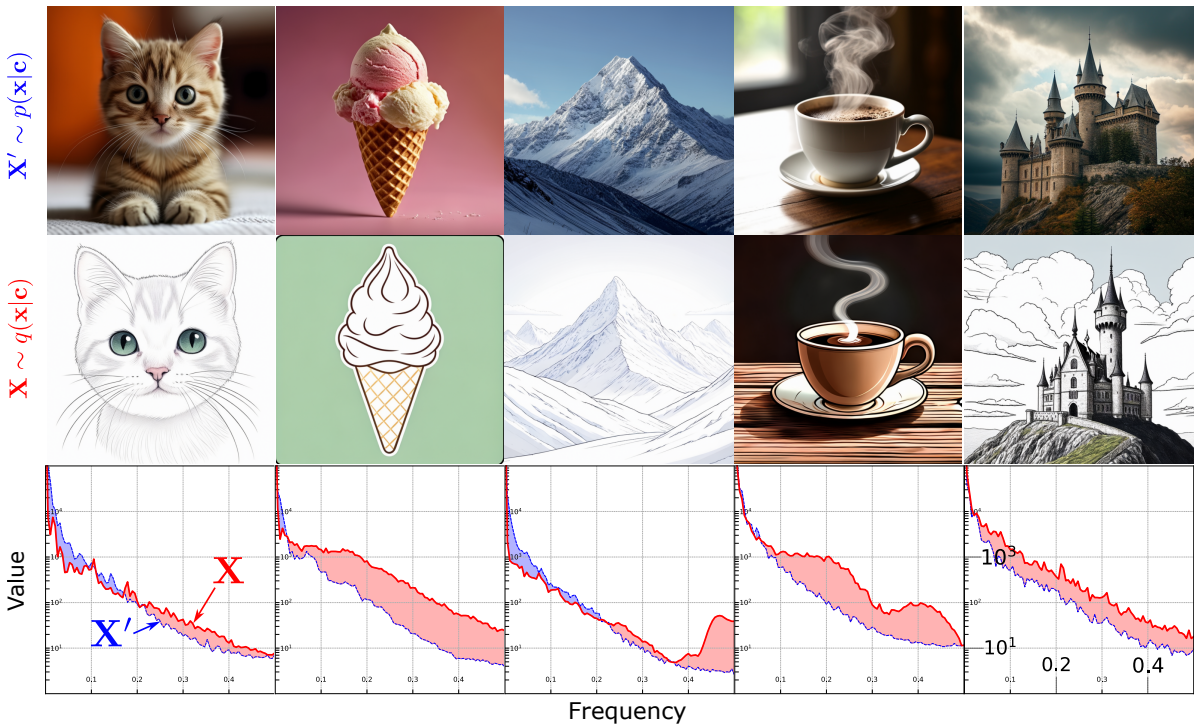

Figure 21. Comparison of generated samples: (top) images synthesized by the original diffusion model, (middle) images obtained via our weighted sampling process, and (bottom) corresponding frequency component analysis.