IMPROVING SINGLE NOISE LEVEL DENOISING SAMPLERS WITH RESTRICTED GAUSSIAN ORACLES

Leello Tadesse Dadi
EPFL-STI-IEM-LIONS
Lausanne, Switzerland
leello.dadi@epfl.ch

Andrej Janchevski
EPFL-STI-IEM-LIONS
Lausanne, Switzerland
andrej.janchevski@epfl.ch

Volkan Cevher
EPFL-STI-IEM-LIONS
Lausanne, Switzerland
volkan.cevher@epfl.ch

ABSTRACT

Practical generative modeling pipelines and diffusion Monte-Carlo schemes, which adapt diffusion models for sampling from unnormalized log-densities, both rely on denoisers (or score estimates) at different noise scales. This complicates the sampling process as denoising schedules require careful tuning and nested inner-MCMC loops. In this work, we propose a single noise level sampling procedure that only requires a single low-noise denoiser. Our framework results from improvements we bring to the multimeasurement Walk-Jump sampler of Saremi & Srivastava (2021) by mixing in ideas from the proximal sampler of Shen et al. (2020). Our analysis shows that annealing (or multiple noise scales) is unnecessary if one is willing to pay an increased memory cost. We demonstrate this by proposing an *entirely log-concave* sampling framework.

1 INTRODUCTION

Denoising diffusion models Ho et al. (2020b); Dhariwal & Nichol (2021) generate samples by first training a sequence of denoisers at various noise levels and chaining them in a clever manner to transform unstructured random noise into a sample from a target distribution. The different noise levels, the methods in which the denoisers are chained, and the balancing of the loss for different noise levels all require careful tuning Karras et al. (2022). So much so that one might ask why we must sample using different noise levels and not simply just one? Why is the annealing (i.e., the gradual decrease of noise) during the sampling stage necessary?

The reason, as discussed in section 2, is to break down a difficult sampling problem into a sequence of uni-modal sampling problems. Indeed, distributions of interest such as natural images are often multi-modal. The denoising diffusion framework breaks down this sampling task into a sequence of uni-modal distributions that are easily approximable by Gaussians. In this work, we show that this sequence of simpler distributions need not be at different noise levels. To propose our framework that operates at a single noise level, we build on ideas from log-concave sampling.

In particular, we build on the work of Saremi & Srivastava (2021) and earlier investigations in Jain & Poole (2022) that propose a single noise level generative framework. Their sampling process involves sampling from a distribution that is not log-concave. To mitigate this, Saremi et al. (2023) propose a *chain of log-concave* breakdown of the task but at the cost of requiring denoisers at several different noise scales. Moreover, their work merely guarantees that the distributions sampled in their framework are *increasingly* log-concave. The following question remained unanswered

Is there an entirely log-concave single-noise level generative framework ?

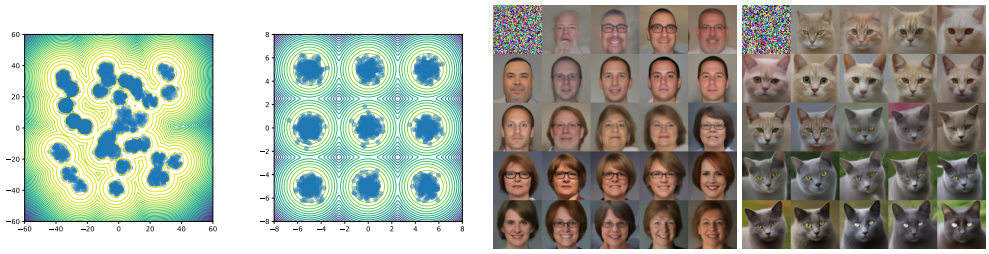

Figure 1: The MultiProx Sampler on a mixture of Gaussians and on the FFHQ and AFHQ 64×64 datasets. The samples above are generated using *a single MCMC chain*: our sampler is able to traverse between modes effectively.

In this work, we first study the case when the unnormalized log-density is known to propose the `MultiProx` sampler. Then, we conduct experiments to show that our framework still yields good results when using a score function from data on real datasets.

2 A CASE AGAINST ANNEALING

In this preliminary section, we present the necessary background on the variance-preserving variant of Denoising Diffusion Probabilistic Models (DDPMs) Ho et al. (2020a). Our goal is to characterize the motivation for *time annealing* in diffusion models and question its necessity.

2.1 DDPMS AND THE MOTIVATION BEHIND ANNEALING

For a data distribution $X_0 \sim p_0$ over $\mathbb{R}^d$, DDPMs first learn how to denoise by generating noisy versions of p_0 (i.e., the forward process). Concretely, noisy measurements X_t, with density denoted p_t, of the target distribution are generated as

$$X_t = \sqrt{\alpha_t} X_0 + \sqrt{1-\alpha_t} Z_t, \tag{FWD}$$

where $\alpha_t < 1$ is a chosen scaling factor; $Z_t \sim \mathcal{N}(0, I)$ is an independently sampled standard Gaussian; and X_0 is a sample from the data distribution p_0. Given these noisy measurements, a network $\texttt{net}_\theta(x, t)$ is trained to denoise X_t by learning to predict the noise Z_t given X_t. This boils down to approximating the rescaled score of X_t given by $-\sqrt{1-\alpha_t}\nabla \log p_t$. With simple algebra, we can show that the network is merely a reparametrized approximation to the conditional expectation of $X_0|X_t$. In other words, we can show that

$$\frac{1}{\sqrt{\alpha_t}}\left(X_t - \sqrt{1-\alpha_t}\texttt{net}_\theta(X_t, t)\right) = \mathbb{E}[X_0|X_t]. \tag{1}$$

Once the network is trained, diffusion models use this network to progressively denoise a standard Gaussian to generate new samples. The network has learned to compute conditional expectations and the goal of the denoising process is to exploit these conditional expectations to output a new sample. If T is the largest amount of noise added during training, the denoising process starts from an approximation $\tilde{X}_T$ of X_T, which is sampled from a standard Gaussian $\mathcal{N}(0, I)$. The closer the distribution of $\tilde{X}_T$ is to p_T the better: the quality of the Gaussian initialization improves as T increases since the larger T is, the closer p_T is to $\mathcal{N}(0, I)$. After the initialization, the denoising is achieved by simulating the reverse process. A single step of the latter yields the following Gaussian approximation,

$$X_0|X_T \approx \mathbb{E}[X_0|X_T] + \sigma_T Z, \tag{2}$$

where $Z \sim \mathcal{N}(0, I)$. Unfortunately, this single-step estimation of X_0 from X_T becomes increasingly poor as T becomes large. This large discrepancy between $X_0|X_T$ and its Gaussian approximation $\mathcal{N}(\mathbb{E}[X_0|X_T], \sigma_T I)$ is the *central motivation for time-annealing*. One-step denoising does not work because $X_0|X_T$ is too multi-modal. Indeed *multi-modal distributions are not well approximated by their expectation* so we must progressively denoise in smaller steps where multi-modality is not an issue and each individual step is well-approximated by a Gaussian distribution.

To restate the trade-off: the one-step Gaussian approximation is increasingly worsened by large choices of T, yet we must choose T to be large in order for the initialization at $\hat{X}_T$ to be valid. Time annealing is precisely the design choice that allows diffusion models to balance these competing trade-offs. For any large T, we can define a discretization sequence $0 = t_0 < t_1 < \cdots < t_N = T$ such that the Gaussian approximation $X_{t_{i-1}}|X_{t_i} \approx \mathbb{E}\left[X_{t_{i-1}}|X_{t_i}\right] + \sigma_{t_{i-1}} Z_{t_{i-1}}$, where $Z_{t_{i-1}} \sim \mathcal{N}(0, I)$ is valid. These Gaussian approximations using the conditional expectation correspond precisely to the exponential integration scheme with N discretization points applied to equation 2. This is the reason why annealing is necessary: it breaks down the difficult multi-modal sampling step of going from X_T to X_0 into a sequence of uni-modal sampling problems $X_{t_{i-1}}|X_{t_i}$.

Diffusion models are a sequence of Gaussian approximations and time-annealing is what makes these Gaussian approximations valid by defining a sequence of steps over which Gaussian approximations using the learned conditional expectations are appropriate. This comes at a cost: a series of conditional expectations $\mathbb{E}\left[X_{t_{i-1}}|X_{t_i}\right]$ must be approximated, which requires denoisers at different noise scales.

2.2 The disadvantages of time annealing

Having expounded on the motivation for time-annealing, we now argue that time-annealing comes with critical drawbacks.

First, annealing is taxing to the denoising network. In practice, the most successful denoising networks are built on the ADM networks of Dhariwal & Nichol (2021) which have been carefully improved in Karras et al. (2024). There, a single network is trained to denoise images at varying noise scales, where a separate component informs the network of the noise level.

The task of denoising varies greatly at different noise scales. At high-noise levels, the denoiser must learn how to introduce low-frequency elements in the images; at low-noise levels, it must act solely on the high frequencies to introduce detail. Performing both these tasks using the same network requires careful balancing of the training loss. Indeed, as observed in Karras et al. (2024), a multi-task training approach is adopted to stabilize training. Denoising different noise levels can be so radically different that recent work Balaji et al. (2022) trains an ensemble of denoisers that separately focus on the high-noise and low-noise stages. In addition to the architectural difficulties, time annealing also introduces tuning difficulties: the training noise schedule was observed to be highly sensitive to the output resolution in Chen (2023). Finally, taking inspiration from Rissanen et al. (2022); Dieleman (2024), we perform a frequency analysis on intermediate images generated during the denoising process of a standard diffusion model, EDM Karras et al. (2022). Figure 2 illustrates the power spectrum throughout the denoising stages of EDM, which was trained on the FFHQ dataset. We see that the time-annealing schedule, despite requiring architectural modifications and meticulous tuning, results in a significant portion of the denoising process adding minimal structural detail to the images. Specifically, the bottom middle plot of Figure 2 demonstrates that the spectrum remains flat, akin to white noise, for a substantial part of the process.

In summary, while time annealing is a pivotal component as discussed earlier in this section, it places a considerable demand on the design of denoising architectures and necessitates precise tuning that is both resolution- and dataset-dependent. Yet, a significant part of this process is spent without contributing any meaningful structure to the images.

In the ensuing sections, we will introduce an alternative approach to decomposing the sampling task. This method does not require a time-embedding sub-network and alleviates the tuning burden by reducing the number of variables involved, offering a more streamlined and efficient framework.

3 Related work

Diffusion Monte Carlo: Our work builds on previous schemes that mix ideas from log-concave sampling and denoising diffusions. Huang et al. (2023) first introduce the idea of inner langevin loops for sampling from potentials and provide a theoretical analysis. Similar ideas also appear in McDonald & Barron (2022) and Vargas et al. (2023) which incorporate score estimation for diffusive sampling, as do Akhound-Sadegh et al. (2024). Chen et al. (2024) propose DiGs which corresponds exactly to the proximal sampler of Shen et al. (2020) made practical with MALA as an

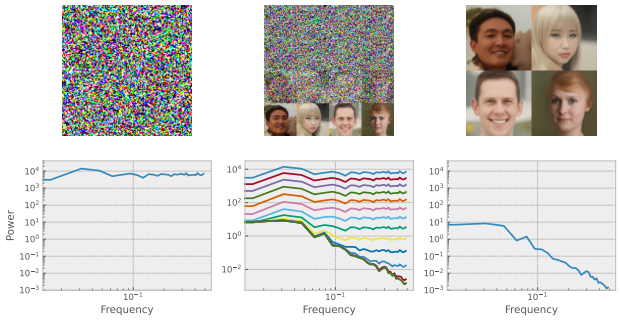

Figure 2: A visualization of the average spectral power density throughout the denoising process of EDM Karras et al. (2022) on FFHQ64: Observe that pure noise has a flat power spectrum (bottom left) and natural images have a linearly decreasing power spectrum on a log-log plot (bottom right) since lower frequencies dominate in natural images. We plot the power spectrum of the intermediate images in the denoising process (middle). Observe that the first half of the denoising acts on all frequencies by lowering the power spectrum *but does not add structure*, it is only later in the process that the denoising starts acting on higher frequencies and adds structure in the images.

approximation of the restricted Gaussian Oracle. They circumvent the difficulty of sampling from high-noise levels by initializing in a "clever way", which unlike our proposal is only a heuristic. He et al. (2024) suggest a zeroth order method to improve over the proximal sampler. Phillips et al. (2024) also propose a sequential Monte Carlo scheme based on diffusions. The work of Chehab & Korba (2024) reviews all recent tempering schemes. Crucially, the fundamental difficulty of *duality of log-concavity* which underlies sampling tasks is detailed in Grenioux et al. (2024). It is this difficulty that our scheme addresses.

The Restricted Gaussian Oracle (RGO) in the literature: Practical implementations of the RGO of Shen et al. (2020) appears in earlier forms in Gao et al. (2020) and Xiao et al. (2021).

Multi-measurement process: Using multiple correlated measurements appears in Saremi & Srivastava (2021) and Saremi et al. (2023). More generally, multi-measurements correspond to linear observation as defined in Montanari (2023) or posterior sampling in linear regression with fixed design.

4 A PRINCIPLED MULTI-MEASUREMENT SAMPLING FRAMEWORK

4.1 MULTI-MEASUREMENT MODEL

Let $X_0 \sim p_0$. Let $t > 0$ be a fixed noise level. We define m, correlated, noisy measurements $(Y_1, \ldots, Y_m)$ of X, at noise level $t > 0$, as

$$Y_i = e^{-t} X + \sqrt{1 - e^{-2t}} Z_i \tag{3}$$

where $Z_1, \ldots, Z_m$ are independent $\mathcal{N}(0, I)$ variables. These m measurements are correlated through X. Our method builds on the observation from Saremi & Srivastava (2021) that denoising from an entire set of correlated noisy measurements $(Y_1, \ldots, Y_m)$ is easier than denoising from a single noisy measurement. The multimeasurement sampling framework we analyze in this section consists of generating X_0 by denoising from the m fixed-noise level measurements $(Y_1, \ldots, Y_m)$. Unlike DDPMs, this framework does not progressively denoise following an annealing schedule. It consists of two components: a denoising component that produces a sample X_0 from the noisy measurements and an initialization component that generates the noisy measurements $(Y_1, \ldots, Y_m)$ to denoise from. We summarize the framework in Algorithm 1 and explain its soundness in detail in what follows.

Algorithm 1: Gibbs Multi-measurement Sampling Framework

Input: $m, t \geq 0$
$(Y_1^0, \ldots, Y_m^0) \leftarrow \mathcal{N}(0, I_{md})$;
for $i \in [1, m]$ **do**
  $Y_i \leftarrow$ Sample from $Y_i | Y_{-i}$
end
return *Sample from* $X | Y_{1,\ldots,m}$

4.2 Denoising from multiple measurements

We first consider the denoising component: the process generating X_0 from $(Y_1, \ldots, Y_m)$. This can be achieved by sampling from the conditional distribution $X_0 | (Y_1, \ldots, Y_m)$. Fortunately, for any noise level t, the denoising distribution $X_0 | (Y_1, \ldots, Y_m)$ can be made to be uni-modal by accumulating enough measurements, i.e., by choosing m large enough. This follows from the simple observation that the average of the m measurements yields

$$\frac{1}{m} \sum_{i=1}^{m} Y_i \overset{\mathcal{L}}{=} e^{-t} X + \sqrt{\frac{1 - e^{-2t}}{m}} Z \tag{4}$$

where $Z \sim \mathcal{N}(0, I)$, which reduces the amount of noise by a factor $\sqrt{m}$. The parameter m can thus be set to compensate for a large choice of t. Indeed, as discussed in section 2, a large choice of t is necessary in order to be able to initialize $(Y_1, \ldots, Y_m)$ before denoising, but the denoising becomes increasingly difficult as multi-modality increases with t. With the parameter m, we can act on this trade-off to ensure that the denoising stays a uni-modal sampling problem. We formally show this in the following theorem.

Assumption 4.1. *The score $\nabla \log p_0$ is L-Lipschitz.*

Theorem 4.2 (Uni-modality of Denoising distribution)**.** *Under assumption 4.1, for any t and $Y_1, \ldots, Y_m$ defined as in equation 3, the Hessian of the conditional distribution $X | Y_1, \ldots, Y_m$ satisfies*

$$\nabla^2 \log p(x | y_1, \ldots, y_m) \preceq \left(L - m \frac{e^{-2t}}{1 - e^{-2t}} \right) I. \tag{5}$$

According to equation 5, as long as

$$m > (e^{2t} - 1) L,$$

the denoising distribution $X_0 | (Y_1, \ldots, Y_m)$ is strongly-log-concave, i.e, unimodal, making the denoising tractable. Consequently, as long as a sample from $(Y_1, \ldots, Y_m)$ can be obtained, a sample of X_0 can be obtained through sampling from a strongly log-concave distribution.

Remark 4.3. *Our result holds for log-smooth distributions (Assumption 4.1), this assumption is standard in the analysis of denoising diffusions Chen et al. (2022); Benton et al. (2023). An alternative way of obtaining a log-smooth distribution is through Gaussian convolution of a bounded distribution: if X is bounded, then $X + \sqrt{\delta} Z$, with $Z \sim \mathbb{N}(0, I)$ is log-smooth. This convolution trick to obtain log-smooth distributions corresponds to the* early-stopping *criterion assumption in convergence proofs of diffusion samplers.*

4.3 Sampling the noisy measurements

We turn our attention to the initialization problem of obtaining a sample from $(Y_1, \ldots, Y_m)$. If t is large, each Y_i approaches a standard Gaussian according to equation 3. The individual coordinates approximate standard Gaussians, but they must be appropriately correlated in order to accurately approximate $(Y_1, \ldots, Y_m)$. Indeed, the benefit of multiple measurements follows from the correlation of the measurements and is a crucial component as the noise reduction in equation 4 depends on it.

Instead of sampling each coordinate by sequentially increasing the conditioning as in Saremi et al. (2023), we propose a Gibbs sampling strategy Casella & George (1992); Roberts & Sahu (1997). Gibbs sampling consists of iteratively sampling each component Y_i conditioned on all the other $m-1$ measurements, denoted Y_{-i}. Gibbs sampling is guaranteed to converge to the joint distribution

Rosenthal (1995) and is a well-studied strategy for sampling when each conditioned marginal is easy to sample from(see 12.3.1 of Koller & Friedman (2009)). The following theorem will help show that there exists a choice of t such that each conditional marginal $Y_i|Y_{-i}$ is easy to sample from.

Theorem 4.4 (Uni-modality of conditional noisy measurements)**.** *Under 4.1. For any, $t > 0$ and $Y_1, \ldots, Y_m$ as in equation 3, we have that for any $i \in [m]$, the hessian of $\log p(y_i|y_{-i})$ is given by*

$$\begin{aligned}\nabla^2_{y_i} \log p(y_i|y_{-i}) &= \operatorname{cov}_{X|Y_1,\ldots,Y_m=(y_1,\ldots,y_m)} \left(\frac{e^{-t}X - y_i}{1-e^{-2t}} \right) - \frac{1}{1-e^{-2t}} I \\ &= \left(\frac{e^{-t}}{1-e^{-2t}} \right)^2 \operatorname{cov}_{X|Y_1,\ldots,Y_m=(y_1,\ldots,y_m)} (X) - \frac{1}{1-e^{-2t}} I.\end{aligned} \tag{6}$$

In order to use the theorem above to show that the distributions are log-concave, we need to control the covariance term in equation 6. We can upper-bound this covariance term using Bakry-Emery's result, showing that strongly log-concave measures verify the Poincaré inequality.

Theorem 4.5 (Variance of log-concave measures)**.** *Let X be an L-smooth density, if m is such that $m \geq 2(e^{2t} - 1)L$, then it holds that*

$$\operatorname{cov}_{X|Y_1,\ldots,Y_m=(y_1,\ldots,y_m)} (X) \leq \frac{L}{2}.$$

Equipped with the result above, we can show that, in order for equation 6 to be negative, the noise level should be set such that

$$t \geq \frac{1}{2} \log \left(1 + \frac{L}{2} \right).$$

With the above conditions, we can show that our sampling framework, detailed in 1, is a fully uni-modal sampling pipeline. This was achieved not with annealing but with multiple measurements. Indeed combining the previous theorems, we have that the following theorem holds for the Gibbs multi-measurement sampler.

Theorem 4.6 (Fully log-concave pipeline)**.** *Under assumption 4.1. For $t > \frac{1}{2} \log \left(1 + \frac{L}{2}\right)$ and $m \geq 2(e^{2t} - 1)L$, we have that the framework 1 is a fully log-concave pipeline, where each step only samples from a unimodal distribution.*

5 THE MULTIPROX SAMPLER

In the previous section, we showed that the Gibbs multimeasurement is an entirely log-concave sampling framework. In this section, we show that each individual sampling step is computationally tractable. We first show that each sampling step in Algorithm 1 can be implemented exactly. We derive from this result an implementation of our framework we call the `MultiProx` sampler.

5.1 HIGH-ACCURACY SAMPLERS FOR STRONGLY-LOG-CONCAVE DISTRIBUTIONS

Having reduced all sampling steps in Algorithm 1 to log-concave sampling, we now further improve over Saremi & Srivastava (2021); Saremi et al. (2023) by using high-accuracy samplers. The work Saremi et al. (2023) suggests using Tweedie's denoiser to approximately sample from $X|Y_{1,\ldots,m}$ and a Langevin MCMC scheme to sample from $Y_i|Y_{-i}$. Unfortunately, both these choices do not fully exploit the strong log-concavity achieved with the careful selection of m and t in Theorem 4.6. Their work requires estimations of $\nabla \log p(y_i|y_{-i})$ even when $\nabla \log p(x)$ is known, a severe drawback that was shown to negatively affect the performance of OAT in Grenioux et al. (2024). In what follows, we show that using a Restricted Gaussian Oracle (RGO) implemented with a rejection sampling algorithm yields *exact* samplers for both sampling steps in Algorithm 1.

Rejection sampling applied to a strongly-log-concave distribution is a sampling scheme that always outputs an exact sample. It functions by accepting samples from a proposal distribution under a well-chosen acceptance probability. As shown in 5, the denoising distribution $X_0|Y_{1,\ldots,m}$ is strongly log-concave if m is chosen appropriately. In the following theorem, we show that we can exactly sample from the denoising distribution.

Algorithm 2: Restricted Gaussian Oracle with Rejection sampling

Input: Averaged m noisy measurements $\bar{y}$

Output: A sample from $p \propto \exp\left(\bar{f}(x) := -f(x) + \frac{m}{2(1-e^{-2t})}\|\bar{y} - e^{-t}x\|^2\right)$

Compute $x^\star_{\bar{y}} = \arg\min_{x\in\mathbb{R}^d} \bar{f}(x)$;

while *True* **do**

  $s \leftarrow$ Sample from $\mathcal{N}\left(x^\star_{\bar{y}}, \frac{1}{L}I\right)$;

  $u \leftarrow$ Sample from $\mathcal{U}(0,1)$;

  if $\log u < -\bar{f}(s) + \bar{f}(x^\star_{\bar{y}}) + \frac{L}{2}\|s - x^\star_{\bar{y}}\|^2$ **then**

    return s;

  end

end

Figure 3: CIFAR10 and FFHQ 64x64

Theorem 5.1 (Exact denoising RGO)**.** *Under assumption equation 4.1, for $m > d(e^{2t} - 1)$, the rejection sampling algorithm 2 outputs an exact sample from $X|Y_{1,\dots,m}$ in $\mathcal{O}(1)$ iterations.*

Observe that the condition on m was strengthened by including the dimension d. The additional dimension factor ensures that $X|Y_{1,\dots,m}$ is well-conditioned. Consequently, rejection sampling can output an exact sample from $X|Y_{1,\dots,m}$ in a constant number of iterations.

Remarkably, this very same result allows us to sample from $Y_i|Y_{-i}$. Indeed observe that

$$Y_i|Y_{-i} = e^{-t}\left(X_0|Y_{-i}\right) + \sqrt{1 - e^{-2t}}Z_i,$$

since Z_i is independent from Y_{-i}. Consequently, to obtain a sample from it suffices to obtain a sample from $X|Y_{-i}$, which is the denoising distribution with one less sample, i.e, $m-1$ noisy measurements. If follows that, with choice of $m > 1 + d(e^{2t} - 1)$, the RGO applied to denoise from the average of the $m-1$ measurements Y_{-i} outputs an exact sample from $Y_i|Y_{-i}$ in a finite number of steps.

With these observations, we derive the implementable `MultiProx` sampler which outputs a sample in a finite number of steps.

Algorithm 3: `MultiProx` sampler

Input: $K, m, t \geq 0$

$(Y_1^0, \dots, Y_m^0) \leftarrow \mathcal{N}(0, I_{md})$;

for $k = 1, \dots, K$ **do**

  for $i \in [1, m]$ **do**

    $Y_i^k \leftarrow$ Sample from $Y_i|(Y_1^k, \dots, Y_{i-1}^k, Y_{i+1}^{k-1}, \dots, Y_m^{k-1})$ using $\texttt{RGO}_{m-1}$ 2

  end

  $X \leftarrow$ Sample from using $\texttt{RGO}_m$ 2 from $\bar{y} = \text{average}(Y_{1,\dots,m})$

end

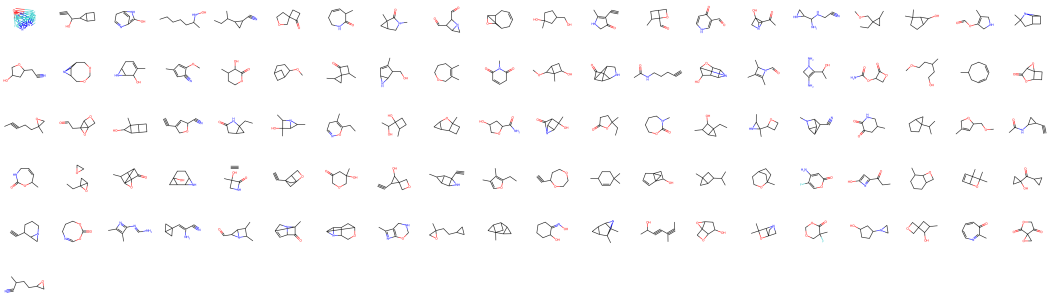

Figure 4: An example of a sample chain obtained with `MultiProx` and a ConGress denoiser trained on QM9. We set for QM9 $K = 10$, $m = 100$, $t = 250$, $t' = 2$ (50% and 0.4% of T).

6 EXPERIMENTS

We present our results using the EDM denoiser of Karras et al. (2022) at a *a single noise level* using our `MultiProx` framework in figures 1 and 3.

We also provide results on discrete distributions. Modeling and sampling from discrete distributions such as graphs is a more complicated generative learning task but of high interest in many domains. We thus focused on evaluating our sampling technique on two classical graph distribution learning tasks: learning to generate novel small molecular structures using the QM9 dataset of 9-node graphs with 4 atom (node) types and 5 bond (edge) types Ruddigkeit et al. (2012); Ramakrishnan et al. (2014), and Comm20, a toy dataset of 100 2-block SBM-sampled random graphs. Evaluation metrics for molecules include measuring the sampling rates of valid, unique, and novel molecular structures. For the simpler Comm20 dataset lacking graph interpretation, one can perform Maximum Mean Discrepancy (MMD) statistical tests for the difference between sampled and true distributions of various classical graph features Gretton et al. (2012); Liao et al. (2019).

Graph diffusion model experiments differ from standard practice in that the noise process and denoising model act potentially on both continuous and discrete features, where discrete features are sampled through denoising the per-class log-probability and a final argmax over all logits to sample a graph node or edge type. DiGress Vignac et al. (2023), the state-of-the-art model that we re-evaluated with `MultiProx`, also introduces a discrete noise process with a categorical instead of a Gaussian distribution, however, we focused on the authors' ConGress version with Gaussian continuous-space noising for consistency with other experiments.

Table 1 displays our performance on the QM9 dataset, while Figure 4 visualizes the molecule graphs in an example sampled Gibbs chain. Refer to Appendix B for our results on the Comm20 dataset.

Table 1: Sampling metrics for each noise level hyperparameter configuration we tested for the QM9 dataset of molecule graphs. Left to right: `MultiProx` fixed noise level t and final output noise level $t' \leq t$ (as percentages of T), percentage of valid molecule graphs, percentage of unique valid molecules, percentage of novel unique valid molecules (not present in the dataset) and total execution time of sampling. Arrows indicate whether higher or lower values of a metric are better. Best values per metric are in bold, while second-best values are underlined.

t	t'	↑ Valid [%]	↑ Unique [%]	↑ Novel [%]	↓ Wall time [s]
Baseline		<u>96.52</u>	78.34	54.92	6302
50%	50%	0.00	0.00	0.00	**151**
50%	25%	0.16	**100.0**	**96.01**	<u>1655</u>
50%	10%	74.61	<u>87.74</u>	66.28	2561
50%	0.4%	**96.62**	77.95	55.20	3137
10%	10%	14.52	30.64	<u>95.32</u>	**151**

REFERENCES

Tara Akhound-Sadegh, Jarrid Rector-Brooks, Avishek Joey Bose, Sarthak Mittal, Pablo Lemos, Cheng-Hao Liu, Marcin Sendera, Siamak Ravanbakhsh, Gauthier Gidel, Yoshua Bengio, et al. Iterated denoising energy matching for sampling from boltzmann densities. *arXiv preprint arXiv:2402.06121*, 2024.

Yogesh Balaji, Seungjun Nah, Xun Huang, Arash Vahdat, Jiaming Song, Qinsheng Zhang, Karsten Kreis, Miika Aittala, Timo Aila, Samuli Laine, et al. ediff-i: Text-to-image diffusion models with an ensemble of expert denoisers. *arXiv preprint arXiv:2211.01324*, 2022.

Joe Benton, Valentin De Bortoli, Arnaud Doucet, and George Deligiannidis. Linear convergence bounds for diffusion models via stochastic localization. *arXiv preprint arXiv:2308.03686*, 2023.

George Casella and Edward I George. Explaining the gibbs sampler. *The American Statistician*, 46 (3):167–174, 1992.

Omar Chehab and Anna Korba. A practical diffusion path for sampling. *arXiv preprint arXiv:2406.14040*, 2024.

Sitan Chen, Sinho Chewi, Jerry Li, Yuanzhi Li, Adil Salim, and Anru R Zhang. Sampling is as easy as learning the score: theory for diffusion models with minimal data assumptions. *arXiv preprint arXiv:2209.11215*, 2022.

Ting Chen. On the importance of noise scheduling for diffusion models. *arXiv preprint arXiv:2301.10972*, 2023.

Wenlin Chen, Mingtian Zhang, Brooks Paige, José Miguel Hernández-Lobato, and David Barber. Diffusive gibbs sampling. *arXiv preprint arXiv:2402.03008*, 2024.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

Sander Dieleman. Diffusion is spectral autoregression, 2024. URL `https://sander.ai/2024/09/02/spectral-autoregression.html`.

Ruiqi Gao, Yang Song, Ben Poole, Ying Nian Wu, and Diederik P Kingma. Learning energy-based models by diffusion recovery likelihood. *arXiv preprint arXiv:2012.08125*, 2020.

Louis Grenioux, Maxence Noble, Marylou Gabrié, and Alain Oliviero Durmus. Stochastic localization via iterative posterior sampling. In *Forty-first International Conference on Machine Learning*, 2024.

Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. A Kernel Two-Sample Test. *Journal of Machine Learning Research*, 13(25):723–773, 2012. ISSN 1533-7928. URL `http://jmlr.org/papers/v13/gretton12a.html`.

Ye He, Kevin Rojas, and Molei Tao. Zeroth-order sampling methods for non-log-concave distributions: Alleviating metastability by denoising diffusion. *arXiv preprint arXiv:2402.17886*, 2024.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 6840–6851. Curran Associates, Inc., 2020a.

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

Xunpeng Huang, Hanze Dong, HAO Yifan, Yian Ma, and Tong Zhang. Reverse diffusion monte carlo. In *The Twelfth International Conference on Learning Representations*, 2023.

Ajay Jain and Ben Poole. Journey to the baoab-limit: finding effective mcmc samplers for score-based models. In *NeurIPS 2022 Workshop on Score-Based Methods*, 2022.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35:26565–26577, 2022.

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*, pp. 24174–24184, 2024.

Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

Renjie Liao, Yujia Li, Yang Song, Shenlong Wang, Will Hamilton, David K Duvenaud, Raquel Urtasun, and Richard Zemel. Efficient Graph Generation with Graph Recurrent Attention Networks. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/hash/d0921d442ee91b896ad95059d13df618-Abstract.html.

Curtis McDonald and Andrew Barron. Proposal of a score based approach to sampling using monte carlo estimation of score and oracle access to target density. *arXiv preprint arXiv:2212.03325*, 2022.

Andrea Montanari. Sampling, diffusions, and stochastic localization. *arXiv preprint arXiv:2305.10690*, 2023.

Angus Phillips, Hai-Dang Dau, Michael John Hutchinson, Valentin De Bortoli, George Deligiannidis, and Arnaud Doucet. Particle denoising diffusion sampler. *arXiv preprint arXiv:2402.06320*, 2024.

Raghunathan Ramakrishnan, Pavlo O. Dral, Matthias Rupp, and O. Anatole von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific Data*, 1(1):140022, August 2014. ISSN 2052-4463. doi: 10.1038/sdata.2014.22. URL https://www.nature.com/articles/sdata201422. Publisher: Nature Publishing Group.

Severi Rissanen, Markus Heinonen, and Arno Solin. Generative modelling with inverse heat dissipation. *arXiv preprint arXiv:2206.13397*, 2022.

Gareth O Roberts and Sujit K Sahu. Updating schemes, correlation structure, blocking and parameterization for the gibbs sampler. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 59(2):291–317, 1997.

Jeffrey S Rosenthal. Minorization conditions and convergence rates for markov chain monte carlo. *Journal of the American Statistical Association*, 90(430):558–566, 1995.

Lars Ruddigkeit, Ruud van Deursen, Lorenz C. Blum, and Jean-Louis Reymond. Enumeration of 166 Billion Organic Small Molecules in the Chemical Universe Database GDB-17. *Journal of Chemical Information and Modeling*, 52(11):2864–2875, November 2012. ISSN 1549-9596. doi: 10.1021/ci300415d. URL https://doi.org/10.1021/ci300415d. Publisher: American Chemical Society.

Saeed Saremi and Rupesh Kumar Srivastava. Multimeasurement generative models. *arXiv preprint arXiv:2112.09822*, 2021.

Saeed Saremi, Ji Won Park, and Francis Bach. Chain of log-concave markov chains. *arXiv preprint arXiv:2305.19473*, 2023.

Ruoqi Shen, Kevin Tian, and Yin Tat Lee. Composite logconcave sampling with a restricted gaussian oracle. *arXiv preprint arXiv:2006.05976*, 2020.

Francisco Vargas, Will Grathwohl, and Arnaud Doucet. Denoising diffusion samplers. *arXiv preprint arXiv:2302.13834*, 2023.

Clément Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. DiGress: Discrete Denoising diffusion for graph generation. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=UaAD-Nu86WX.

Zhisheng Xiao, Karsten Kreis, and Arash Vahdat. Tackling the generative learning trilemma with denoising diffusion gans. *arXiv preprint arXiv:2112.07804*, 2021.

A PROOFS

Theorem 4.2 (Uni-modality of Denoising distribution)**.** *Under assumption 4.1, for any t and $Y_1, \dots, Y_m$ defined as in equation 3, the Hessian of the conditional distribution $X|Y_1, \dots, Y_m$ satisfies*

$$\nabla^2 \log p(x|y_1, \dots, y_m) \preceq \left(L - m \frac{e^{-2t}}{1 - e^{-2t}} \right) I. \tag{5}$$

Proof. For $y_1, \dots, y_m$ fixed, we now from Bayes that

$$p(x|y_1, \dots, y_m) \propto p(x, y_1, \dots, y_m) = p(x) \Pi_{i=1}^m p(y_i|x)$$

The log-gradient with respect to x is then given by

$$\nabla \log p(x|y_1, \dots, y_m) = \nabla \log p(x) + \sum_{i=1}^m \nabla \log p(y_i|x).$$

Recall that, since $Y_i - e^{-t}X$ is a Gaussian, we have that $\log p(y_i|x) = -\frac{\|y_i - e^{-t}x\|^2}{2(1-e^{-2t})} + C$, where C is a constant. It follows that the Hessian is

$$\nabla^2 \log p(x|y_1, \dots, y_m) = \nabla^2 \log p(x) - \sum_{i=1}^m \frac{e^{-2t}}{1 - e^{-2t}} \preceq LI - m \frac{e^{-2t}}{1 - e^{-2t}}$$

where the last inequality follows from the Lipschitz Assumption 4.1. □

Theorem 4.4 (Uni-modality of conditional noisy measurements)**.** *Under 4.1. For any, $t > 0$ and $Y_1, \dots, Y_m$ as in equation 3, we have that for any $i \in [m]$, the hessian of $\log p(y_i|y_{-i})$ is given by*

$$\begin{aligned} \nabla^2_{y_i} \log p(y_i|y_{-i}) &= \operatorname{cov}_{X|Y_1,\dots,Y_m=(y_1,\dots,y_m)} \left(\frac{e^{-t}X - y_i}{1 - e^{-2t}} \right) - \frac{1}{1 - e^{-2t}} I \\ &= \left(\frac{e^{-t}}{1 - e^{-2t}} \right)^2 \operatorname{cov}_{X|Y_1,\dots,Y_m=(y_1,\dots,y_m)} (X) - \frac{1}{1 - e^{-2t}} I. \end{aligned} \tag{6}$$

Proof. The result follows from the following identity

$$\nabla^2 \log p = \nabla \frac{\nabla p}{p} = \frac{\nabla^2 p}{p} - \frac{\nabla p}{p} \left(\frac{\nabla p}{p} \right)^\top$$

The joint distribution of the noisy measurements is easy to compute, and we find that for any noisy measurement Y_i, given the other $m - 1$ measurements, we have that

$$p_{m,e^{-t}}(y_i|y_{-i}) \propto p(y_1, \dots, y_m) = \int p(y_1, \dots, y_m|x) p(x) dx = \mathbb{E}_X \left[\prod_{k=1}^m \gamma_{0,1} \left(\frac{y_k - e^{-t}X}{\sqrt{1 - e^{-2t}}} \right) \right]$$

We can compute the gradient of its log-density as follows

$$\begin{aligned} \nabla_{y_i} \log p_{m,e^{-t}}(y_i|y_{-i}) &= \frac{\mathbb{E}_X \left[\nabla_{y_i} \prod_{k=1}^m \gamma_{0,1} \left(\frac{y_k - e^{-t}X}{\sqrt{1 - e^{-2t}}} \right) \right]}{p(y_1 \dots, y_m)} \\ &= \mathbb{E}_X \left[\left(\frac{e^{-t}X - y_i}{1 - e^{-2t}} \right) \prod_{k=1}^m \gamma_{0,1} \left(\frac{y_k - e^{-t}X}{\sqrt{1 - e^{-2t}}} \right) \right] \frac{1}{p(y_1, \dots, y_m)} \\ &= \int \left(\frac{e^{-t}X - y_i}{1 - e^{-2t}} \right) \frac{p(y_1, \dots, y_m|x) p(x)}{p(y_1, \dots, y_m)} dx \\ &= \mathbb{E}_{X|Y_1,\dots,Y_m=(y_1,\dots,y_m)} \left[\frac{e^{-t}X - y_i}{1 - e^{-2t}} \right] \\ &= \frac{e^{-t}}{1 - e^{-2t}} \mathbb{E}\left[X|Y_1, \dots, Y_m = y_1, \dots, y_m \right] - \frac{1}{1 - e^{-2t}} y_i \end{aligned}$$

□

B GRAPH DIFFUSION EXPERIMENTS FOR COMM20

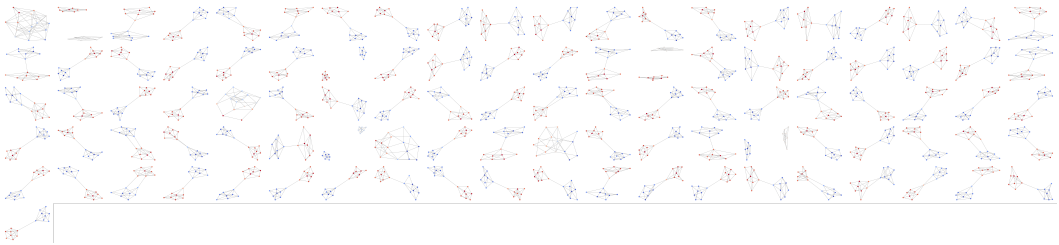

Figure 5: An example of a sample chain obtained with `MultiProx` and a ConGress denoiser trained on Comm20. We set for Comm20 $K = 10$, $m = 100$, $t = 200, t' = 50$ (40% and 10% of T).

Table 2: Sampling metrics for each noise level hyperparameter configuration we tested for the Comm20 dataset of molecule graphs. Left to right: `MultiProx` fixed noise level t and final output noise level $t' \leq t$ (as percentages of T), Maximum Mean Discrepancy (MMD) metrics for node degrees, per-node clustering coefficients, node orbits, and node eigenvalues between sampled and dataset graph distributions, and total execution time of sampling. Arrows indicate whether higher or lower values of a metric are better. Best values per metric are in bold, while second-best values are underlined.

t	t'	↓ Degree MMD	↓ Cluster MMD	↓ Orbit MMD	↓ Spectral MMD	↓ Wall time [s]
Baseline		0.0064	**0.0725**	<u>0.0214</u>	**0.0232**	846
40%	40%	0.1630	0.1230	0.6954	0.0718	**139**
40%	20%	**0.0022**	0.1148	0.0666	<u>0.0633</u>	<u>305</u>
40%	10%	<u>0.0037</u>	0.0864	**0.0051**	0.1219	388
40%	0.2%	0.0150	<u>0.0833</u>	0.0376	0.1965	467
10%	10%	0.0734	0.1272	0.6653	0.0752	**138**

C IMPLEMENTATION

For the molecule generation experiments, we extend the implementation of Vignac et al. (2023) (available at `https://github.com/cvignac/DiGress`) with code for our novel specific sampling method tailored to the architecture of DiGress. Our graph experiment code implementation is fully available at `https://github.com/LIONS-EPFL/MultiProxDiffusion` to facilitate reproducibility.

All code was executed on a single machine with the following specifications:

- AMD EPYC™ 7742 64-core CPU @ 2.25GHz;
- NVIDIA A100-SXM4-40GB GPU;
- 1.0TB RAM.

D IMPLEMENTATION OF AN RGO WHEN THE LOG-DENSITY IS AVAILABLE

```python
import jax
import jax.numpy as jnp
from jax.scipy.optimize import minimize
from jax_tqdm import scan_tqdm
import optimistix as optx


def build_restricted_gaussian_oracle(logprob_fn, m, sigma, alpha):

```

```
    def rgo(y_bar, rng):
        def f_bar(x, *_):
            return -logprob_fn(x, None) + 0.5 * m / sigma * jnp.sum((x - y_bar) ** 2)

        minimizer_sol = optx.minimise(f_bar, optx.BFGS(rtol=1e-4, atol=1e-4), y_bar)
        x_star = minimizer_sol.value

        def propose(params):
            rng_key_x, _, _, count = params
            rng_key_x, rng_proposal = jax.random.split(rng_key_x)
            noise = jax.random.normal(rng_proposal, shape=y_bar.shape)
            z = x_star + jnp.sqrt(1 / alpha) * noise
            rng_key_x, rng_uniform = jax.random.split(rng_key_x)
            u = jax.random.uniform(rng_uniform, shape=(1,))
            log_accept_ratio_term = -f_bar(z, None) + f_bar(x_star, None) + 0.5 * jnp.sum(noise**2)
            return rng_key_x, jnp.log(u) - log_accept_ratio_term, z, count + 1

        def cond(params):
            rng_key_x, log_u_minus_log_accept, z, count = params
            is_rejected = log_u_minus_log_accept > 0
            return is_rejected

        rng, rng_loop_init = jax.random.split(rng)
        initial_gap = jnp.array([1.0])
        initial_state = (rng_loop_init, initial_gap, x_star, 0)

        _, _, sample, final_count = jax.lax.while_loop(cond, propose, initial_state)

        return sample

    return rgo
```

Listing 1: Python code for a restricted Gaussian oracle using JAX and Optimistix.

E IMPLEMENTATION OF AN APPROXIMATE RGO WITH EDM

```
def sample_images(M:int, sigma:float, N: int, b_sz:int, res:int, start=None):
    sigma = torch.tensor(sigma).cuda()
    M = torch.tensor(M).cuda()
    if start is None:
        repeated_noised = sigma * torch.randn(M, b_sz, 3, res, res).cuda()
        start = repeated_noised[0, 0]
    else:
        repeated_noised = start.repeat(M, b_sz, 1, 1, 1).cuda() + sigma * torch.randn(M,b_sz, 3, res, res).cuda()
    denoised_lst = [start.unsqueeze(0).repeat(b_sz, 1, 1, 1).detach().cpu()]
    previous = start.repeat(b_sz, 1, 1, 1).cuda()
    for i in range(N):
        for k in tqdm.tqdm(range(M)):
            averaged = torch.mean(repeated_noised, dim=0)
            denoised = net(averaged, sigma / torch.sqrt(torch.tensor(M)), None)
            denoised = 2*denoised - previous
            previous = denoised
            repeated_noised[k] = denoised + sigma * torch.cuda.FloatTensor(b_sz, 3, res, res).normal_()
```

```
18          averaged = torch.mean(repeated_noised, dim=0)
19          denoised = net(averaged, sigma / torch.sqrt(torch.tensor(M)),
    None)
20          denoised_lst.append(denoised.detach().cpu())
21          repeated_noised = denoised.unsqueeze(0).repeat(M, 1, 1, 1, 1) +
    sigma * torch.cuda.FloatTensor(M, b_sz, 3, res, res).normal_()
22      return denoised_lst
```