Why Gaussian Diffusion Models Fail on Discrete Data and How to Prevent It?

Alexander Shabalin[1] **Simon Elistratov**[2] **Viacheslav Meshchaninov**[1]
Ildus Sadrtdinov[1*] **Dmitry Vetrov**[1*]
[1]Constructor University, Bremen, Germany
[2]Lomonosov Moscow State University, Moscow, Russia

Abstract

Diffusion models have become a standard approach for generative modeling in continuous domains, yet their application to discrete data remains challenging. We investigate why Gaussian diffusion models with the DDPM solver struggle to sample from discrete distributions that are represented as a mixture of delta-distributions in the continuous space. Using a toy Random Hierarchy Model, we identify a critical sampling interval in which the density of noisified data becomes multimodal. In this regime, DDPM occasionally enters low-density regions between modes producing out-of-distribution inputs for the model and degrading sample quality. We show that existing heuristics, including self-conditioning and a solver we term q-sampling, help alleviate this issue. Furthermore, we demonstrate that combining self-conditioning with switching from DDPM to q-sampling within the critical interval improves generation quality on real data. We validate these findings across conditional and unconditional tasks in multiple domains, including text, programming code, and proteins.

1 Introduction

In recent years, diffusion models (Sohl-Dickstein et al., 2015) have achieved state-of-the-art generation quality across continuous domains, including images (Rombach et al., 2022), audio (Chen et al., 2021), and video (Ho et al., 2022), but extending them to discrete domains such as text, graphs, or molecules remains challenging. Consequently, autoregressive models (Brown et al., 2020) remain the default for text generation, despite well-known drawbacks: left-to-right generation, one-token-at-a-time decoding, and the inability to revise earlier decisions. Diffusion models offer a path to alleviating these limitations.

Existing diffusion-based text generation methods fall into two paradigms: discrete (Austin et al., 2021; Sahoo et al., 2024) and continuous (Li et al., 2022; Shabalin et al., 2025a). In this work, we focus on continuous approaches because they do not suffer from the token factorization error (Lee et al., 2026; Deschenaux et al., 2026) and enable the reuse of techniques developed for continuous domains, such as classifier-free guidance (Ho & Salimans, 2021) and consistency distillation (Luo et al., 2023).

The most common baseline represents text as a sequence of token embeddings and applies Gaussian diffusion in the resulting embedding space. However, since the underlying data distribution is discrete, this approach may be prone to errors, as we illustrate in Figure 1. In the continuous case, clean data lie on a manifold, and the density transitions smoothly to a standard Gaussian. Small score-estimation errors rarely lead to irrecoverable failures, as they do not push the sampling trajectory off the manifold. In the discrete case, by contrast, the density $p(\mathbf{x}_t)$ separates into distinct modes centered at data points as $t \to 0$, and score-estimation errors combined with time discretization can trap samples between modes, yielding incorrect generations. While the training set is finite for both discrete and continuous domains, diffusion models on continuous data in the generalization regime

*Shared senior authorship. Correspondence to: ashabalin@constructor.university

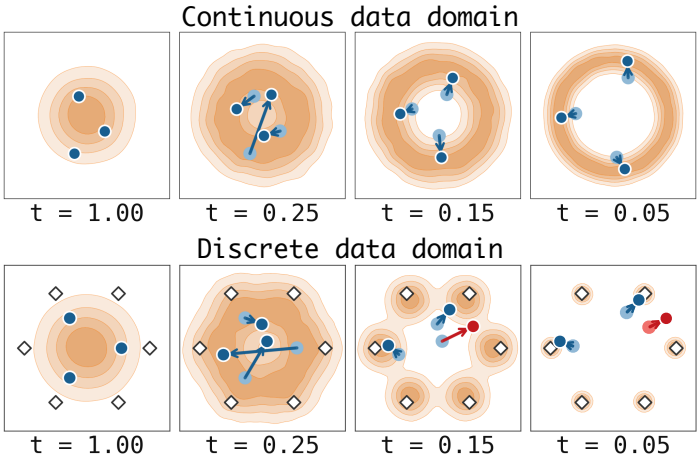

Figure 1: Toy visualization of sampling with continuous diffusion models in continuous and discrete data domains. Orange contours visualize the density of noisified data $p(\mathbf{x}_t)$. Blue points and arrows show in-distribution sampling trajectories, red — out-of-distribution trajectories. White diamonds show original data points in the discrete setting.

recover the underlying distribution (Zhang et al., 2024; Favero et al., 2025) and the data manifold (Pidstrigach, 2022), and thus do not exhibit similar issues.

In this work, we systematically analyze when and why Gaussian diffusion fails on discrete data by studying the generation process in detail. We argue that the most critical phase of sampling is the late stage, when the model's likelihood becomes multimodal and the model becomes highly sensitive to errors. From this perspective, we explain why heuristics proposed in prior work, such as modified solvers (Karimi Mahabadi et al., 2024; Shabalin et al., 2025b) and self-conditioning (Chen et al., 2023), improve generation of discrete data, despite lacking a comprehensive justification. Our contributions are as follows:

- We analyze the diffusion sampling process for discrete text generation. Using a simplified Random Hierarchy Model (RHM), we identify a *critical time interval* in which the DDPM solver (Ho et al., 2020) occasionally enters low-likelihood regions, producing out-of-distribution inputs for the model.
- We explain why existing heuristics, including *self-conditioning* and the solver we call *q-sampling*, mitigate this problem. We also explore *Minimum Bayes-Risk* (Kumar & Byrne, 2004) decoding for unconditional text generation.
- We validate our findings on unconditional and conditional generation across several domains, including text, code, and proteins, and provide practical recommendations for improving generation quality, such as switching from DDPM to q-sampling and tuning the self-conditioning probability during training.

2 Background

Text generation In natural language processing, unconditional text generation aims to draw a sample $\mathbf{w}$ from an unknown distribution $p(\mathbf{w})$, where $\mathbf{w} = [w^1, \ldots, w^n]$ denotes a token sequence of variable length n. For conditional text generation, the target distribution becomes $p(\mathbf{w}|\mathbf{y})$, where $\mathbf{y}$ is an observed conditioning variable; the goal is to produce text consistent with this condition.

Gaussian diffusion models Gaussian diffusion models (Ho et al., 2020) consist of forward and reverse processes. Given a continuous data sample $\mathbf{x}_0 \sim p_{\text{data}}$, a forward process produces a sequence of progressively noisier samples $\{\mathbf{x}_1, \ldots, \mathbf{x}_T\}$, where

$\mathbf{x}_t \sim \mathcal{N}(\sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I})$ and $\mathbf{x}_T \sim \mathcal{N}(0, \mathbf{I})$. Alternatively, $\mathbf{x}_t$ can be sampled directly from $\mathbf{x}_0$ with $q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$, where $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$.

To reverse this process, the model $\hat{\mathbf{x}}_\theta$ is trained to reconstruct a clean data sample $\mathbf{x}_0$ from the noisy latent $\mathbf{x}_t$ by minimizing the denoising objective:

$$\mathcal{L}(\theta) = \mathbb{E}_{\mathbf{x}_0, t, q(\mathbf{x}_t|\mathbf{x}_0)} \left[\|\mathbf{x}_0 - \hat{\mathbf{x}}_\theta(\mathbf{x}_t, t)\|^2 \right] \tag{1}$$

Continuous Text Diffusion In continuous text diffusion, $\mathbf{x}_0$ is obtained by applying an encoder $\mathcal{E}$ to an input sequence $\mathbf{w}$. Commonly, $\mathcal{E}$ is a simple embedding layer, although some works employ a transformer-based encoder instead (Lovelace et al., 2023; Shabalin et al., 2025a; Meshchaninov et al., 2025a). Unless otherwise stated, this paper uses the former approach. To recover tokens from the latent, a linear decoder $\mathcal{D}$ is used. In Appendix A, we discuss related works with their contributions in more detail.

To train the model, one can use pre-trained embeddings (Strudel et al., 2023; Shabalin et al., 2025a;b) and optimize the loss in Eq. 1. More commonly, embeddings are trained jointly with the model by minimizing the following objective (Li et al., 2022; Gong et al., 2023):

$$\mathcal{L}^{\text{e2e}} = \mathbb{E}_{\mathbf{w}, t, \varepsilon \sim \mathcal{N}(0, \mathbf{I})} \left[\left\| \mathcal{E}_\theta(\mathbf{w}) - \hat{\mathbf{x}}_\theta\left(\sqrt{\bar{\alpha}_t}\mathcal{E}_\theta(\mathbf{w}) + \sqrt{1-\bar{\alpha}_t}\varepsilon, t\right) \right\|^2 - \log p_{\mathcal{D}}\left(\mathbf{w}|\mathcal{E}_\theta(\mathbf{w})\right) \right] \tag{2}$$

DDPM solver DDPM solver (Ho et al., 2020) is the most popular approach for reversing a diffusion process given the trained model $\hat{\mathbf{x}}_\theta$. It is defined as a Markov chain with Gaussian transitions starting at $p(\mathbf{x}_T) = \mathcal{N}(0, \mathbf{I})$:

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}\left(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \tilde{\beta}_t \mathbf{I}\right), \tag{3}$$

$$\text{where } \boldsymbol{\mu}_\theta(\mathbf{x}_t, t) = \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\hat{\mathbf{x}}_\theta(\mathbf{x}_t, t) + \frac{\sqrt{\alpha_t}\,(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t \quad \text{and} \quad \tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t \tag{4}$$

3 Random Hierarchy Model

3.1 Setup description

To analyze the dynamics of the DDPM solver on discrete data, we use the Random Hierarchy Model (RHM) (Cagnetta et al., 2024) as a toy example, which is designed to mimic the hierarchical structure of text. The dataset is generated by rules defined on a tree-structured graph with multiple levels, yielding sequences over a vocabulary of $V = 16$ tokens with length $n = 8$. Of the V^n possible sequences, only $N = 16384$ satisfy the generation rules; the remaining $V^n - N \approx 4 \cdot 10^9$ do not. We refer to these as *correct* and *incorrect* sequences, respectively. A detailed description of the data generation protocol is given in Appendix B.

To obtain the diffusion latent $\mathbf{x}_0$, we map each token to an embedding from a matrix $E \in \mathbb{R}^{V \times d}$, so that $\mathbf{x}_0 \in \mathbb{R}^{n \times d}$ with $d = 2$. We split the correct sequences into train and test sets in equal parts and optimize the loss in Eq. 2. We train the model until convergence, ensuring a memorization of ~ 0.5, meaning the model generates samples from both splits. As a primary metric we use *correctness*, the rate of the correct sequences among generated.

Since the number of correct sequences is finite, the data distribution can be written as $p_{\text{data}}(\mathbf{x}) = \frac{1}{N}\sum_{\mathbf{x}_0} \delta(\mathbf{x} - \mathbf{x}_0)$, where δ is the Dirac delta function. This allows us to compute the density of the noised latents $p(\mathbf{x}_t)$ analytically (see Eq. 6 in Appendix B), as well as the optimal denoising prediction $\mathbf{x}_0^*(\mathbf{x}_t) = \mathbb{E}[\mathbf{x}_0|\mathbf{x}_t] = \sum_{\mathbf{x}_0} \mathbf{x}_0 p(\mathbf{x}_0|\mathbf{x}_t)$, which minimizes Eq. 1.

3.2 Exploratory experiments

To start, we exploit the advantages of the RHM setup and compare the trained diffusion model against the optimal prediction $\mathbf{x}_0^*(\mathbf{x}_t) = \mathbb{E}[\mathbf{x}_0|\mathbf{x}_t]$. As expected, DDPM with optimal predictions recovers sequences from the true data distribution, achieving a correctness of exactly 1. In contrast, DDPM with a trained model yields a correctness of 0.79, indicating a

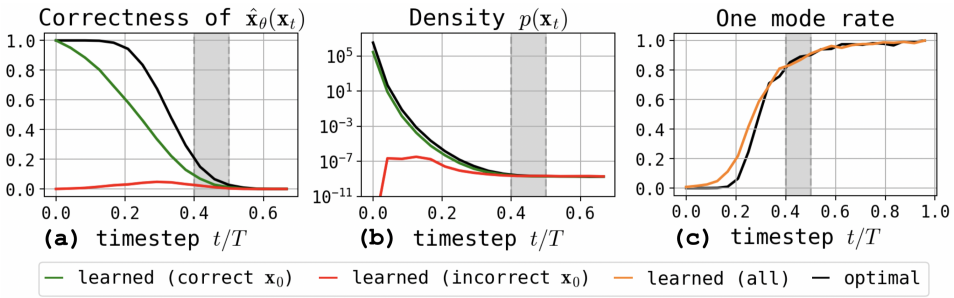

Figure 2: **(a)** correctness of predictions $\hat{\mathbf{x}}_\theta(\mathbf{x}_t)$ and **(b)** mean probability density $p(\mathbf{x}_t)$ along correct, incorrect and optimal sampling trajectories; **(c)** share of pairs of trajectories lying within the same mode (separately for all learned and optimal trajectories). Gray vertical stripes show the transition interval.

deviation from the optimal prediction. We therefore conclude that DDPM with a trained model is prone to errors, and proceed to identify when and why these errors occur.

To understand the failures of the sampling process, we partition DDPM trajectories into two groups based on whether they produce *correct* or *incorrect* samples at $t = 0$, and compare their dynamics against optimal prediction trajectories in Figure 2. Figure 2a shows that for $t/T > 0.5$, all three trajectories exhibit identical zero correctness. For $t/T < 0.4$, however, correct and optimal trajectories show growing correctness, while incorrect trajectories remain near zero for all t. A similar pattern appears in the density plot in Figure 2b: all trajectories remain aligned until $t/T \approx 0.4$, after which correct trajectories trend upward while incorrect ones drop sharply, approaching zero density near $t/T = 0$.

Overall, correct and incorrect trajectories are practically indistinguishable for $t/T > 0.5$, but begin to diverge at $t/T \in [0.4, 0.5]$, where incorrect trajectories fall behind in both correctness and density. We call this range the *transition interval* and hypothesize that it marks a qualitative change in the shape of the distribution, making subsequent steps critical for accurate sampling. We refer to $t/T \in [0, 0.4]$ as the *critical interval* and focus our analysis on it to identify the factors that prevent DDPM from recovering correct sequences.

Since the data distribution is a mixture of delta functions and $p(\mathbf{x}_T)$ is unimodal, the distribution must transition from unimodal to multimodal at some point during sampling, as shown in Figure 1. We suppose that this transition happens in the *transition interval*. To verify this, we estimate the degree of multimodality at each sampling timestep by measuring whether pairs of samples $\mathbf{x}_t^1$ and $\mathbf{x}_t^2$ lie in the same mode (see details in Appendix B).

Figure 2c plots the fraction of pairs $(\mathbf{x}_t^1, \mathbf{x}_t^2)$ sharing the same mode. At $t/T = 1$, all trajectories occupy a single mode, and this fraction decreases steadily as t/T decreases towards 0.5. Between $0.1 < t/T < 0.4$, however, the fraction drops rapidly, and as t/T approaches zero it becomes negligible, since each mode collapses to a single data sample. Thus, around $t/T \approx 0.4$ a large number of modes begin to form, giving rise to *low-density regions* in the diffusion space. Incorrect trajectories may enter these regions and become trapped: since samples from such regions are unlikely to appear during the forward process, they act as out-of-distribution inputs for the model, which then fails to denoise them, producing incorrect sequences (we justify this empirically in Appendix C). In Figure 2b, we show that incorrect trajectories indeed exhibit densities several orders of magnitude lower than correct ones.

Takeaway 1: When modeling discrete data, the density of noised samples splits into multiple modes at some timestep, creating low-density regions between them. DDPM occasionally enters these regions, producing out-of-distribution inputs for the model and leading to incorrect generations.

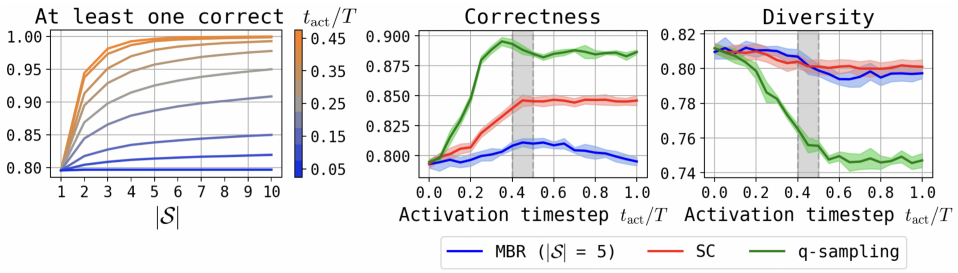

Figure 3: **left:** ratio of candidate sets $\mathcal{S}$ with at least one correct sample vs. its size $|\mathcal{S}|$ for different activation timesteps t_{act}; **right:** correctness and diversity of MBR with $|\mathcal{S}| = 5$, self-conditioning (SC), and q-sampling vs. their activation timestep t_{act}. Gray vertical stripes show the transition interval.

3.3 Mitigating DDPM failures

In this section, we discuss how the failures of DDPM can be mitigated. Correct and incorrect trajectories differ only in sampled noise realizations and the resulting model predictions, so in principle, resampling the noise at a suitable timestep could shift an incorrect trajectory into a correct one. To test this, we run DDPM up to $t = t_{\text{act}} + 1$. Then, at activation time $t = t_{\text{act}}$, we branch the trajectory by sampling multiple noise realizations and completing the generation for each of them, which yields a set of samples $\mathcal{S}$. Figure 3 (left) shows the fraction of such sets containing at least one correct sample, averaged over several repetitions.

For $t_{\text{act}}/T = 0.05$, increasing $|\mathcal{S}|$ gives no improvement: at this stage the sequence is nearly fully formed, and resampling noise can no longer repair an incorrect outcome. For larger activation timesteps, even a modest increase in $|\mathcal{S}|$ substantially improves correctness; at $t_{\text{act}}/T \geq 0.4$, correctness approaches 1 for sufficiently large $\mathcal{S}$. These results suggest that DDPM sampling is reliable up to around $t/T = 0.4$, which matches with the start of the critical interval, and the steps below this threshold require more careful handling. In the following paragraphs, we discuss three techniques that help mitigate errors in this regime.

Minimum Bayes-Risk (MBR) While branching trajectories improves correctness by up to 0.2 (Figure 3, left), in practice we cannot directly filter out incorrect samples, which necessitates a different selection criterion. *Minimum Bayes-Risk* (MBR) decoding (Kumar & Byrne, 2004) proposes such criterion for conditional text generation. It selects the candidate with the highest expected utility relative to all other generated samples, for a given utility function (e.g., BLEU). MBR has been applied to conditional text diffusion models (Li et al., 2022; Gong et al., 2023), but is not directly applicable to unconditional generation, where candidates are typically too diverse, causing MBR to favor the most "average" samples.

However, when candidates share a substantial part of their trajectory, namely the segment $t \in [t_{\text{act}} + 1, T]$, they become much more similar, enabling MBR to filter incorrect samples without sacrificing diversity. Figure 3 (right) shows correctness and diversity as a function of t_{act}, using negative Hamming distance as the utility function. Diversity is measured as the number of unique correct samples in a batch of 8192, divided by the total number of correct samples. MBR improves correctness when applied within the transition interval, but the gain diminishes for larger t_{act} as candidates become too diverse. Crucially, it is possible to choose a timestep within the transition interval, such as $t_{\text{act}}/T = 0.4$, where correctness improves while diversity remains high. MBR examples are provided in Appendix G.1.

q-sampling While DDPM is the standard solver, several works (Karimi Mahabadi et al., 2024; Shabalin et al., 2025b) employ an alternative update rule:

$$\mathbf{x}_{t-1} = \sqrt{\bar{\alpha}_{t-1}}\hat{\mathbf{x}}_\theta(\mathbf{x}_t, t) + \sqrt{1 - \bar{\alpha}_{t-1}}\varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \mathbf{I}) \tag{5}$$

We refer to this as *q-sampling*, since it is derived from the forward process $q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1 - \bar{\alpha}_t)\mathbf{I})$. Unlike DDPM, q-sampling is not a valid solver in the sense that it does

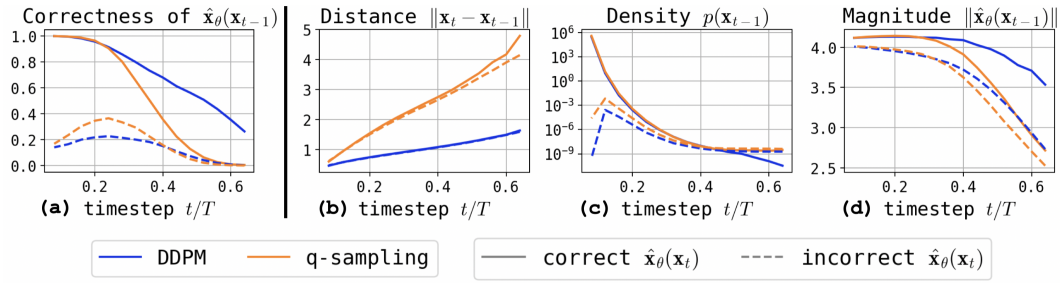

Figure 4: Comparison of one-step updates for DDPM and q-sampling. For each plotted timestep t, we run DDPM from $t' = T$ to $t' = t$. Then, we divide the trajectories into correct and incorrect based on $\hat{\mathbf{x}}_\theta(\mathbf{x}_t)$. The next step $t' = t - 1$ is obtained with either DDPM or q-sampling. The plots show **(a)** correctness of predictions $\hat{\mathbf{x}}_\theta(\mathbf{x}_{t-1})$, **(b)** mean distance $\|\mathbf{x}_t - \mathbf{x}_{t-1}\|$, **(c)** mean density $p(\mathbf{x}_{t-1})$, **(d)** mean magnitude of predictions $\|\hat{\mathbf{x}}_\theta(\mathbf{x}_{t-1})\|$.

not reproduce the reverse process. As shown in Appendix D, q-sampling trajectories have lower marginal variance than those of DDPM, indicating that the two solvers effectively sample from different distributions.

Despite this, Figure 3 (right) shows that switching from DDPM to q-sampling at $t_{\text{act}}/T > 0.2$ significantly improves correctness at the cost of reduced diversity. Notably, activating q-sampling near the transition interval outperforms both pure DDPM ($t_{\text{act}}/T = 0$) and pure q-sampling ($t_{\text{act}}/T = 1$), suggesting that q-sampling does not inherit the pitfalls of DDPM and is particularly effective within the critical interval.

To understand this behavior, we compare single-step updates from DDPM and q-sampling. We first generate multiple sampling trajectories using DDPM. For each latent $\mathbf{x}_t$, we apply both solvers to obtain the corresponding $\mathbf{x}_{t-1}$ and compare the results in Figure 4. Figure 4a shows that q-sampling improves correctness for samples that were incorrectly predicted at previous step for $t/T < 0.4$, while slightly reducing correctness for previously-correct predictions. This is explained by Figure 4b: DDPM keeps $\mathbf{x}_{t-1}$ closer to $\mathbf{x}_t$, which is beneficial when the current prediction is correct, but harmful when it is not. In the latter case, moving farther away, as q-sampling does, is preferable. Figure 4c further shows that when the prediction is incorrect, q-sampling updates have higher density than DDPM updates, meaning the resulting $\mathbf{x}_{t-1}$ is less out-of-distribution. Together, these properties explain why switching to q-sampling in the critical interval mitigates the failures of DDPM.

Finally, q-sampling should not be applied during the early stages of generation, where DDPM does not exhibit the above pitfalls and q-sampling underperforms in terms of prediction magnitude, as reflected in Figure 4d.

Self-conditioning (SC) Self-conditioning proposed in Chen et al. (2023) is a technique that improves text generation quality (Yuan et al., 2022; Lovelace et al., 2023; Shabalin et al., 2025a) by providing the model with its own previous predictions during generation. SC requires modifying the training procedure: with probability $p = 0.5$, the model receives its previous prediction as an additional input, $\hat{\mathbf{x}}_0^t = \hat{\mathbf{x}}_\theta(\mathbf{x}_t, t, \text{SG}(\bar{\mathbf{x}}_0^t))$, where $\bar{\mathbf{x}}_0^t = \hat{\mathbf{x}}_\theta(\mathbf{x}_t, t, \mathbf{0})$ and SG denotes the stop-gradient operator. Otherwise, the SC input is set to zero, $\hat{\mathbf{x}}_0^t = \hat{\mathbf{x}}_\theta(\mathbf{x}_t, t, \mathbf{0})$. During generation, the first prediction is made with SC input set to zero, and at all subsequent steps the previous prediction is passed as the SC input: $\hat{\mathbf{x}}_0^t = \hat{\mathbf{x}}_\theta(\mathbf{x}_t, t, \hat{\mathbf{x}}_0^{t+1})$.

Figure 3 (right) shows that SC improves correctness within the critical interval, though, like MBR and q-sampling, at the cost of reduced diversity. Prior work attributes the benefits of SC to preventing information loss (Chen et al., 2023), more efficient use of model capacity (Dieleman et al., 2022), and increased prediction confidence (Shabalin et al., 2025a). However, these explanations do not clarify what precisely happens when SC is applied to discrete data, nor why it is not used in continuous domain. We argue that SC plays a role analogous to MBR and q-sampling: it helps generation trajectories avoid low-density regions within the critical interval. Unlike those methods, SC achieves this by training

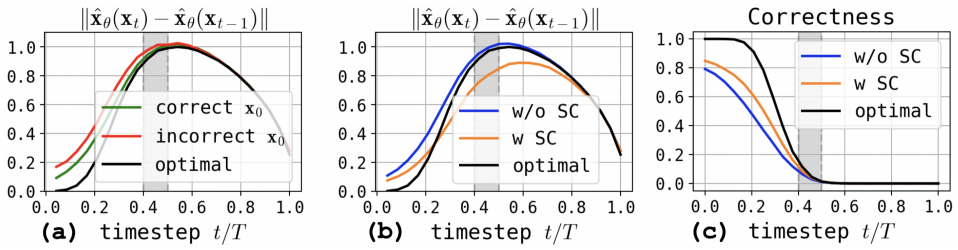

Figure 5: **(a)** mean distance between predictions $\|\hat{\mathbf{x}}_\theta(\mathbf{x}_t) - \hat{\mathbf{x}}_\theta(\mathbf{x}_{t-1})\|$ for correct and incorrect trajectories without SC and for the optimal model; **(b)** mean distance between predictions $\|\hat{\mathbf{x}}_\theta(\mathbf{x}_t) - \hat{\mathbf{x}}_\theta(\mathbf{x}_{t-1})\|$ with and without SC; **(c)** correctness along the trajectory with and without SC. Gray vertical stripes show the transition interval.

the model to condition on its previous prediction, encouraging consecutive predictions to remain close to one another, as illustrated in Figure 5.

Figure 5a shows that within the critical interval, correct trajectories have smaller distances between consecutive predictions than incorrect ones, with SC reducing these distances further (Figure 5b). Although SC deviates from optimal trajectories for $t/T > 0.3$, it more closely tracks optimal predictions than the baseline model at smaller timesteps, and eventually improves generation correctness (Figure 5c). Therefore, SC mitigates the problem with unfavorable noise realizations by anchoring each prediction to the previous one, and can also be viewed as providing the model with two inputs: if one becomes out-of-distribution, the model can still rely on the other.

Takeaway 2: MBR, q-sampling, and SC reduce DDPM generation errors within the critical interval by guiding trajectories away from low-density regions, but at the cost of discarding some correct trajectories and thus reducing diversity.

In Appendices N and O, we confirm that the identified failure modes are specific to discrete distributions: SC and q-sampling provide no benefit in the continuous image domain.

4 Unconditional generation on real data

In this section, we evaluate the impact of MBR, self-conditioning, and q-sampling on unconditional generation. We experiment with Diffusion-LM (Li et al., 2022), which trains embeddings jointly with the diffusion model, and TEncDM (Shabalin et al., 2025a) in two latent space configurations: shallow BERT embeddings (TEncDM Emb) and context-dependent BERT encodings (TEncDM Enc). Context-dependent encodings vary smoothly with the input, forming a continuous latent manifold rather than a set of isolated points. As a result, the probability mass spreads more uniformly across the latent space, leaving fewer low-density regions. Thus, we expect vanilla DDPM to be a stronger baseline in this setting.

Datasets We evaluate the models on ROCStories (Mostafazadeh et al., 2016) (98k examples), English Wikipedia (Foundation) (6.4M examples), and OpenWebText (Gokaslan et al., 2019) (8M examples). Full dataset descriptions are provided in Appendix F.

Metrics We assess performance using **perplexity** computed with GPT-2 Large (Radford et al., 2019), **diversity** (Su et al., 2022), and **Mauve** (Pillutla et al., 2021). Perplexity and diversity are individually limited as metrics: perplexity favors low-diversity outputs, while diversity can be high for incoherent text and low when outputs are repetitive. We therefore prioritize Mauve, which directly measures the proximity between the distributions of generated and reference texts, and encourages a balance between the perplexity and diversity. Mauve is computed by comparing 1 000 generated and reference texts over 10 random seeds

Model	ROCStories			Wikipedia		
	Mauve ↑	PPL ↓	Div ↑	Mauve ↑	PPL ↓	Div ↑
Diffusion-LM	$16.2_{\pm 2.9}$	$148.9_{\pm 2.7}$	$\mathbf{33.7}_{\pm 0.4}$	$1.2_{\pm 0.2}$	$419.8_{\pm 9.3}$	$49.5_{\pm 0.6}$
+ $SC_{0.5}$	$71.3_{\pm 3.8}$	$73.8_{\pm 0.8}$	$30.2_{\pm 0.3}$	$12.1_{\pm 1.9}$	$300.2_{\pm 6.7}$	$\mathbf{58.7}_{\pm 0.4}$
+ $QS_{0.3}$	$72.7_{\pm 4.3}$	$42.7_{\pm 0.4}$	$17.4_{\pm 0.2}$	$5.8_{\pm 0.8}$	$166.6_{\pm 3.3}$	$40.0_{\pm 0.6}$
+ $QS_{0.3}$ + $SC_{0.5}$	$\mathbf{81.6}_{\pm 2.7}$	$\mathbf{41.0}_{\pm 0.6}$	$22.9_{\pm 0.3}$	$36.4_{\pm 4.4}$	$\mathbf{114.2}_{\pm 2.4}$	$48.4_{\pm 0.4}$
+ $SC_{0.95}$	$75.2_{\pm 2.4}$	$59.5_{\pm 1.1}$	$29.0_{\pm 0.4}$	$27.8_{\pm 3.2}$	$197.4_{\pm 4.8}$	$56.6_{\pm 0.5}$
+ $QS_{0.2}$ + $SC_{0.95}$	$80.9_{\pm 3.0}$	$50.1_{\pm 0.4}$	$27.1_{\pm 0.3}$	$\mathbf{44.8}_{\pm 4.7}$	$142.8_{\pm 3.2}$	$54.6_{\pm 0.5}$
TEncDM Emb	$12.5_{\pm 1.4}$	$131.4_{\pm 2.7}$	$\mathbf{37.4}_{\pm 0.4}$	$3.1_{\pm 0.4}$	$541.7_{\pm 10.5}$	$\mathbf{63.4}_{\pm 0.5}$
+ $SC_{0.5}$	$56.1_{\pm 3.4}$	$42.4_{\pm 0.6}$	$24.7_{\pm 0.3}$	$44.5_{\pm 4.6}$	$138.2_{\pm 2.3}$	$50.2_{\pm 0.4}$
+ $QS_{0.4}$	$41.1_{\pm 6.8}$	$71.6_{\pm 1.2}$	$30.5_{\pm 0.2}$	$10.1_{\pm 2.0}$	$255.4_{\pm 7.5}$	$53.3_{\pm 0.6}$
+ $QS_{0.4}$ + $SC_{0.5}$	$62.7_{\pm 3.5}$	$\mathbf{28.6}_{\pm 0.2}$	$19.5_{\pm 0.3}$	$\mathbf{53.5}_{\pm 3.5}$	$86.8_{\pm 1.9}$	$44.5_{\pm 0.5}$
+ $SC_{0.95}$	$77.7_{\pm 3.5}$	$36.0_{\pm 0.7}$	$27.8_{\pm 0.4}$	$46.5_{\pm 4.9}$	$131.6_{\pm 2.9}$	$51.7_{\pm 0.6}$
+ $QS_{0.5}$ + $SC_{0.95}$	$\mathbf{80.2}_{\pm 4.4}$	$29.5_{\pm 0.3}$	$25.7_{\pm 0.2}$	$51.2_{\pm 4.8}$	$\mathbf{79.3}_{\pm 2.0}$	$45.6_{\pm 0.5}$
TEncDM Enc	$67.1_{\pm 4.6}$	$57.1_{\pm 0.9}$	$\mathbf{32.2}_{\pm 0.3}$	$70.1_{\pm 4.2}$	$236.6_{\pm 5.5}$	$\mathbf{63.2}_{\pm 0.5}$
+ $SC_{0.5}$	$81.1_{\pm 2.8}$	$29.0_{\pm 0.4}$	$26.4_{\pm 0.5}$	$87.3_{\pm 3.0}$	$94.0_{\pm 1.8}$	$55.2_{\pm 0.3}$
+ $QS_{0.4}$	$76.9_{\pm 3.1}$	$29.9_{\pm 0.2}$	$24.3_{\pm 0.2}$	$88.2_{\pm 2.2}$	$93.6_{\pm 1.7}$	$53.5_{\pm 0.4}$
+ $QS_{0.4}$ + $SC_{0.5}$	$82.1_{\pm 2.5}$	$29.1_{\pm 0.4}$	$26.6_{\pm 0.4}$	$88.4_{\pm 2.4}$	$79.1_{\pm 1.0}$	$53.3_{\pm 0.4}$
+ $SC_{0.95}$	$\mathbf{86.8}_{\pm 2.9}$	$29.6_{\pm 0.3}$	$31.4_{\pm 0.5}$	$\mathbf{90.1}_{\pm 2.1}$	$92.6_{\pm 1.6}$	$57.3_{\pm 0.3}$
+ $QS_{0.5}$ + $SC_{0.95}$	$85.8_{\pm 2.3}$	$25.5_{\pm 0.3}$	$29.9_{\pm 0.2}$	$\mathbf{89.6}_{\pm 2.4}$	$\mathbf{59.9}_{\pm 0.8}$	$52.6_{\pm 0.4}$

Table 1: Impact of q-sampling (QS) and self-conditioning (SC) on ROCStories and Wikipedia. SC trained with probability p is denoted SC_p; QS with optimal t_{act} is denoted $QS_{t_{\text{act}}/T}$.

for ROCStories and 20 for other datasets, then averaging the results. All metric values are provided in a $\text{mean}_{\pm \text{std}}$ notation. See other experimental details in Appendix E.

Minimum Bayes-Risk As discussed above, MBR can be applied to unconditional generation by branching the trajectory at an activation timestep t_{act} within the transition interval, at which the coarse structure of the sample has been determined and only fine details are to refine[1]. Table 2 shows that this transfers to text generation. Tuning t_{act} on ROCStories with standard parameters (BLEU utility function and $|\mathcal{S}| = 10$ (Li et al., 2022; Gong et al., 2023)) improves both Mauve and perplexity across all three diffusion models. This confirms that errors arising in the critical interval can be mitigated by resampling noise realizations. More detailed analisys of MBR with examples is provided in Appendix G.2. While MBR improves quality, it drastically slows generation and thus, we omit it from further experiments.

Method	Mauve ↑	PPL ↓	Div ↑
Diffusion-LM	$16.2_{\pm 3.0}$	$148.9_{\pm 2.7}$	$33.7_{\pm 0.4}$
+ $MBR_{0.3}$	$20.8_{\pm 3.5}$	$136.3_{\pm 1.8}$	$32.2_{\pm 0.3}$
TEncDM Emb	$15.7_{\pm 2.9}$	$123.2_{\pm 1.8}$	$36.6_{\pm 0.3}$
+ $MBR_{0.3}$	$18.9_{\pm 1.8}$	$101.5_{\pm 1.9}$	$31.6_{\pm 0.5}$
TEncDM Enc	$70.1_{\pm 3.2}$	$54.2_{\pm 1.1}$	$31.2_{\pm 0.4}$
+ $MBR_{0.3}$	$72.3_{\pm 2.0}$	$48.0_{\pm 44.5}$	$28.3_{\pm 0.3}$

Table 2: MBR on ROCStories ($|\mathcal{S}| = 10$).

Self-conditioning We apply SC along the entire sampling trajectory, as this matches the quality of restricting SC to the critical interval while avoiding an additional t_{act} hyperparameter. As confirmed by our experiments in Table 1, SC substantially improves generation quality across all three models. The improvement is most pronounced for embedding-based models (Diffusion-LM and TEncDM Emb), which completely fail without SC but reach competitive performance with it. The occasional increase in diversity with SC is a metric artifact: without SC, generated texts are often of such low quality (e.g., containing repetitions) that they suppress the diversity score. For TEncDM Enc, the Mauve improvement is notable but less dramatic, consistent with our intuition that a more continuous latent space provides a stronger baseline. Overall, as in the RHM setting, SC substantially improves perplexity at some cost to diversity, yielding a better Mauve trade-off.

Prior work universally uses the default SC probability $p = 0.5$, meaning previous predictions are used only half the time during training. We propose increasing p closer to 1: since the

[1]The size of the critical interval, and thus the optimal value of t_{act}, depends on the noise schedule.

model conditions on its previous prediction at every step during sampling except the first, a higher p better matches the inference procedure and encourages stronger reliance on self-conditioning. Our results with $p = 0.95$ confirm that this improves generation quality with only a $1.13\times$ increase in training time over the $p = 0.5$ baseline. In Appendix J, we provide an additional ablation of p values.

q-sampling We observe that q-sampling improves generation quality both with and without SC, and similarly to SC provides a better trade-off between perplexity and diversity. A key practical advantage over SC is that q-sampling requires no retraining, whereas SC modifies the training protocol and increases training time. The hyperparameter t_{act} can be tuned by the inference-time grid-search within a narrow range ($t_{\text{act}} \in [0.3, 0.5]$). In Appendix H, we show that the effect of varying t_{act} on text generation is consistent with the RHM experiments, and that its optimal value tends to be stable across datasets for a given model. Moreover, Appendix L shows the same behavior for OpenWebText and protein generation. Best overall performance is achieved by combining q-sampling with SC.

Takeaway 3: Self-conditioning is essential for unconditional generation, and tuning its probability p above the default $p = 0.5$ can yield significant further gains. Switching from DDPM to q-sampling also improves generation quality, and the best results are achieved by combining both techniques.

5 Conditional generation on real data

We additionally evaluate SC and q-sampling on conditional generation, i.e. seq2seq tasks. These differ fundamentally from unconditional generation, since the input sequence constrains the output. We present results for TEncDM Emb trained on the XSum summarization task (Narayan et al., 2018) and the GENIE model (Lin et al., 2023b) on the CoNaLa dataset (Yin et al., 2018), where the task is to generate a snippet of code given a natural language docstring.

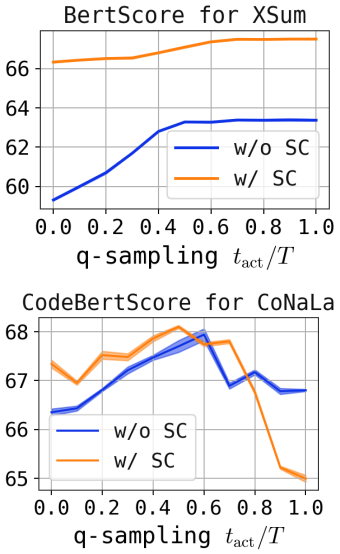

Figure 6: XSum (top) and CoNaLa (bottom) quality for varying q-sampling activation timestep with and without SC.

Figure 6 reports BertScore (Zhang et al., 2020) (XSum) and CodeBertScore (Zhou et al., 2023) (CoNaLa) as a function of the q-sampling activation timestep, with and without SC. Both techniques improve generation quality, but their contributions differ across tasks. For XSum, SC yields a much larger gain, and the optimal q-sampling strategy is to apply it from the very beginning ($t_{\text{act}}/T = 1$). For CoNaLa, SC provides a smaller improvement, and the optimal q-sampling activation timestep is $t_{\text{act}}/T \approx 0.5$. We hypothesize that this discrepancy reflects the strength of the conditioning signal: the more tightly the input constrains the output, the more accurate the model predictions are even in the low-density regions. Therefore, there is less room for q-sampling and SC to improve generation. In a sense, a weak conditioning signal (as in XSum and CoNaLa, where the correct output can vary substantially for a given input) makes seq2seq generation resemble unconditional generation, explaining why these techniques remain effective. This intuition is further supported in Appendix M with results on machine translation and question paraphrasing, which have much stronger condition, and thus q-sampling and SC have little to no improvement.

Takeaway 4: In seq2seq tasks, q-sampling and self-conditioning still improve generation quality, but their benefit is more significant for a weaker conditioning signal.

6 Conclusion

In this work, we investigate why Gaussian diffusion with the DDPM solver underperforms on discrete data generation. We find that DDPM trajectories tend to enter low-density regions during the critical interval — the final phase of generation, where the distribution becomes multimodal. Using the RHM toy example, we analyze how MBR, q-sampling, and SC mitigate this failure mode, and validate our findings on real data. Combining SC and q-sampling substantially improves generation quality for shallow-embedding models, and to a lesser extent for context-dependent embedding models, suggesting that more continuous latent spaces are a promising direction for future research in text diffusion.

Acknowledgments

The authors gratefully acknowledge the computing time made available to them on the high-performance computer at the NHR Center of TU Dresden. This center is jointly supported by the Federal Ministry of Research, Technology and Space of Germany and the state governments participating in the NHR (www.nhr-verein.de/unsere-partner). We also thank Anton Ragin and QST Financial for providing additional computational resources.

References

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=h7-XixPCAL.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020.

Francesco Cagnetta, Leonardo Petrini, Umberto M. Tomasini, Alessandro Favero, and Matthieu Wyart. How deep neural networks learn compositional data: The random hierarchy model. *Phys. Rev. X*, 14:031001, Jul 2024. doi: 10.1103/PhysRevX.14.031001. URL https://link.aps.org/doi/10.1103/PhysRevX.14.031001.

Nanxin Chen, Yu Zhang, Heiga Zen, Ron J Weiss, Mohammad Norouzi, and William Chan. Wavegrad: Estimating gradients for waveform generation. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=NsMLjcFaO8O.

Ting Chen, Ruixiang Zhang, and Geoffrey Hinton. Analog bits: Generating discrete data using diffusion models with self-conditioning, 2023.

The UniProt Consortium. Uniprot: the universal protein knowledgebase in 2021. *Nucleic Acids Research*, 49(D1):D480–D489, 11 2020. ISSN 0305-1048. doi: 10.1093/nar/gkaa1100. URL https://doi.org/10.1093/nar/gkaa1100.

DataCanary, hilfialkaff, Lili Jiang, Meg Risdal, Nikhil Dandekar, and tomtung. Quora question pairs. https://kaggle.com/competitions/quora-question-pairs, 2017. Kaggle.

Justin Deschenaux, Caglar Gulcehre, and Subham Sekhar Sahoo. The diffusion duality, chapter II: ψ-samplers. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=RSIoYWIzaP.

Sander Dieleman, Laurent Sartran, Arman Roshannai, Nikolay Savinov, Yaroslav Ganin, Pierre H. Richemond, Arnaud Doucet, Robin Strudel, Chris Dyer, Conor Durkan, Curtis Hawthorne, Rémi Leblond, Will Grathwohl, and Jonas Adler. Continuous diffusion for categorical data, 2022. URL https://arxiv.org/abs/2211.15089.

Rahim Entezari, Hanie Sedghi, Olga Saukh, and Behnam Neyshabur. The role of permutation invariance in linear mode connectivity of neural networks. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=dNigytemkL.

Alessandro Favero, Antonio Sclocchi, and Matthieu Wyart. Bigger isn't always memorizing: Early stopping overparameterized diffusion models. *arXiv preprint arXiv:2505.16959*, 2025.

Wikimedia Foundation. Wikimedia downloads. URL https://dumps.wikimedia.org.

Zhujin Gao, Junliang Guo, Xu Tan, Yongxin Zhu, Fang Zhang, Jiang Bian, and Linli Xu. Empowering diffusion models on the embedding space for text generation. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 4664–4683, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.261. URL https://aclanthology.org/2024.naacl-long.261/.

Aaron Gokaslan, Vanya Cohen, Ellie Pavlick, and Stefanie Tellex. Openwebtext corpus. http://Skylion007.github.io/OpenWebTextCorpus, 2019.

Shansan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and Lingpeng Kong. Diffuseq: Sequence to sequence text generation with diffusion models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=jQj-_rLVXsj.

Michael Heinzinger, Konstantin Weissenow, Joaquin Gomez Sanchez, Adrian Henkel, Milot Mirdita, Martin Steinegger, and Burkhard Rost. Bilingual language model for protein sequence and structure. *NAR Genomics and Bioinformatics*, 6(4):lqae150, 11 2024. ISSN 2631-9268. doi: 10.1093/nargab/lqae150. URL https://doi.org/10.1093/nargab/lqae150.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021. URL https://openreview.net/forum?id=qw8AKxfYbI.

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., 2020.

Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 8633–8646. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/39235c56aef13fb05a6adc95eb9d8d66-Paper-Conference.pdf.

Rabeeh Karimi Mahabadi, Hamish Ivison, Jaesung Tae, James Henderson, Iz Beltagy, Matthew Peters, and Arman Cohan. TESS: Text-to-text self-conditioned simplex diffusion. In Yvette Graham and Matthew Purver (eds.), *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2347–2361, St. Julian's, Malta, March 2024. Association for Computational Linguistics. URL https://aclanthology.org/2024.eacl-long.144.

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.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open source toolkit for statistical machine translation. In Sophia Ananiadou (ed.), *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pp. 177–180, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL https://aclanthology.org/P07-2045/.

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

Shankar Kumar and William Byrne. Minimum Bayes-risk decoding for statistical machine translation. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*, pp. 169–176, Boston, Massachusetts, USA, May 2 - May 7 2004. Association for Computational Linguistics. URL https://aclanthology.org/N04-1022/.

Chanhyuk Lee, Jaehoon Yoo, Manan Agarwal, Sheel Shah, Jerry Huang, Aditi Raghunathan, Seunghoon Hong, Nicholas M. Boffi, and Jinwoo Kim. Flow map language models: One-step language modeling via continuous denoising. *arXiv preprint arXiv:2602.16813*, 2026.

Xiang Li, John Thickstun, Ishaan Gulrajani, Percy S Liang, and Tatsunori B Hashimoto. Diffusion-lm improves controllable text generation. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 4328–4343. Curran Associates, Inc., 2022.

Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pp. 74–81, 2004.

Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Robert Verkuil, Ori Kabeli, Yaniv Shmueli, Allan dos Santos Costa, Maryam Fazel-Zarandi, Tom Sercu, Salvatore Candido, and Alexander Rives. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637): 1123–1130, 2023a. doi: 10.1126/science.ade2574. URL https://www.science.org/doi/abs/10.1126/science.ade2574.

Zhenghao Lin, Yeyun Gong, Yelong Shen, Tong Wu, Zhihao Fan, Chen Lin, Nan Duan, and Weizhu Chen. Text generation with diffusion language models: a pre-training approach with continuous paragraph denoise. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023b.

Justin Lovelace, Varsha Kishore, Chao Wan, Eliot Shekhtman, and Kilian Q Weinberger. Latent diffusion for language generation. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 56998–57025. Curran Associates, Inc., 2023.

Simian Luo, Yiqin Tan, Longbo Huang, Jian Li, and Hang Zhao. Latent consistency models: Synthesizing high-resolution images with few-step inference, 2023.

Viacheslav Meshchaninov, Egor Chimbulatov, Alexander Shabalin, Aleksandr Abramov, and Dmitry Vetrov. Compressed and smooth latent space for text diffusion modeling. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025a. URL https://openreview.net/forum?id=Rv6Lz84FlZ.

Viacheslav Meshchaninov, Pavel Strashnov, Andrey Shevtsov, Fedor Nikolaev, Nikita Ivanisenko, Olga Kardymon, and Dmitry Vetrov. Diffusion on language model encodings for protein sequence generation. In *Forty-second International Conference on Machine Learning*, 2025b. URL https://openreview.net/forum?id=xB9eROwBCB.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. A corpus and cloze evaluation for

deeper understanding of commonsense stories. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 839–849, 2016.

Muhammad Ferjad Naeem, Seong Joon Oh, Youngjung Uh, Yunjey Choi, and Jaejun Yoo. Reliable fidelity and diversity metrics for generative models. In *International conference on machine learning*, pp. 7176–7185. PMLR, 2020.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 1797–1807, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1206. URL https://aclanthology.org/D18-1206/.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. fairseq: A fast, extensible toolkit for sequence modeling. In Waleed Ammar, Annie Louis, and Nasrin Mostafazadeh (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pp. 48–53, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-4009. URL https://aclanthology.org/N19-4009/.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In Pierre Isabelle, Eugene Charniak, and Dekang Lin (eds.), *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pp. 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL https://aclanthology.org/P02-1040/.

Jakiw Pidstrigach. Score-based generative models detect manifolds. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=AiNrnIrDfD9.

Krishna Pillutla, Swabha Swayamdipta, Rowan Zellers, John Thickstun, Sean Welleck, Yejin Choi, and Zaid Harchaoui. Mauve: Measuring the gap between neural text and human text using divergence frontiers. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 4816–4828. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/260c2432a0eecc28ce03c10dadc078a4-Paper.pdf.

Patrick Pynadath, Jiaxin Shi, and Ruqi Zhang. Candi: Hybrid discrete-continuous diffusion models, 2025. URL https://arxiv.org/abs/2510.22510.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 130136–130184. Curran Associates, Inc., 2024. doi: 10.52202/079017-4135. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/eb0b13cc515724ab8015bc978fdde0ad-Paper-Conference.pdf.

Julian Salazar, Davis Liang, Toan Q. Nguyen, and Katrin Kirchhoff. Masked language model scoring. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 2699–2712, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.240. URL https://aclanthology.org/2020.acl-main.240/.

Alexander Shabalin, Viacheslav Meshchaninov, Egor Chimbulatov, Vladislav Lapikov, Roman Kim, Grigory Bartosh, Dmitry Molchanov, Sergey Markov, and Dmitry Vetrov. Tencdm: Understanding the properties of the diffusion model in the space of language model encodings. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(23):25110–25118, Apr. 2025a. doi: 10.1609/aaai.v39i23.34696. URL https://ojs.aaai.org/index.php/AAAI/article/view/34696.

Alexander Shabalin, Viacheslav Meshchaninov, and Dmitry Vetrov. Smoothie: Smoothing diffusion on token embeddings for text generation, 2025b. URL https://arxiv.org/abs/2505.18853.

Mukul Singh, José Cambronero, Sumit Gulwani, Vu Le, Gust Verbruggen, and Carina Negreanu. Codefusion: A pre-trained diffusion model for code generation. In *EMNLP 2023*, November 2023. URL https://www.microsoft.com/en-us/research/publication/codefusion-a-pre-trained-diffusion-model-for-code-generation/.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In Francis Bach and David Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 2256–2265, Lille, France, 07–09 Jul 2015. PMLR.

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

Robin Strudel, Corentin Tallec, Florent Altché, Yilun Du, Yaroslav Ganin, Arthur Mensch, Will Sussman Grathwohl, Nikolay Savinov, Sander Dieleman, Laurent Sifre, and Rémi Leblond. Self-conditioned embedding diffusion for text generation, 2023. URL https://openreview.net/forum?id=0pzV3lp3IMC.

Yixuan Su, Tian Lan, Yan Wang, Dani Yogatama, Lingpeng Kong, and Nigel Collier. A contrastive framework for neural text generation. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=V88BafmH9Pj.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

Jiasheng Ye, Zaixiang Zheng, Yu Bao, Lihua Qian, and Mingxuan Wang. Dinoiser: Diffused conditional sequence learning by manipulating noises. *Transaction of ACL*, 2024.

Pengcheng Yin, Bowen Deng, Edgar Chen, Bogdan Vasilescu, and Graham Neubig. Learning to mine aligned code and natural language pairs from stack overflow. In *Proceedings of the 15th International Conference on Mining Software Repositories*, MSR '18, pp. 476–486, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450357166. doi: 10.1145/3196398.3196408. URL https://doi.org/10.1145/3196398.3196408.

Hongyi Yuan, Zheng Yuan, Chuanqi Tan, Fei Huang, and Songfang Huang. Seqdiffuseq: Text diffusion with encoder-decoder transformers. *ArXiv*, abs/2212.10325, 2022.

Huijie Zhang, Jinfan Zhou, Yifu Lu, Minzhe Guo, Peng Wang, Liyue Shen, and Qing Qu. The emergence of reproducibility and consistency in diffusion models. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 60558–60590. PMLR, 21–27 Jul 2024.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert, 2020. URL https://arxiv.org/abs/1904.09675.

Shuyan Zhou, Uri Alon, Sumit Agarwal, and Graham Neubig. CodeBERTScore: Evaluating code generation with pretrained models of code. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 13921–13937, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.859. URL https://aclanthology.org/2023.emnlp-main.859/.

Appendix

A Related work . . . **17**

B Random Hierarchy Model: setup description . . . **17**

C Low-density latent samples result in incorrect predictions . . . **18**

D Variance of latents updated with q-sampling and DDPM . . . **19**
- D.1 Setup . . . 19
- D.2 Result . . . 19

E Implementation details . . . **21**

F Datasets . . . **21**
- F.1 Unconditional generation . . . 21
- F.2 Conditional generation . . . 22

G Minimum Bayes-Risk examples . . . **22**
- G.1 Random Hierarchy Model . . . 22
- G.2 ROCStories . . . 22

H q-sampling ablation . . . **22**

I Changing the number of steps . . . **26**

J Self-conditioning probability . . . **26**

K Unsuccessful ideas . . . **29**

L Additional results for unconditional generation . . . **30**
- L.1 OpenWebText results . . . 30
- L.2 Protein generation results . . . 30

M Additional results for conditional generation . . . **31**

N Image diffusion example . . . **31**

O Image toy example . . . **33**

A Related work

While we conduct a thorough analysis of specifics of the continuous text diffusion's generation process, several previous works have attempted to identify the most critical diffusion components and to understand why text diffusions underperform their image counterparts.

Gao et al. (2024); Ye et al. (2024) and Shabalin et al. (2025a) suggest that text diffusion models require more noise at the initial steps of the forward process, because embeddings are well-separated in the latent space and a small amount of noise does not destroy any useful information, which makes the corresponding timesteps useless during generation. However, when embeddings are trained jointly with the diffusion model using Eq 2, they begin to collapse until the decoder can no longer distinguish them, so that even a small amount of noise corrupts them (Li et al., 2022). In this regime, more aggressive noise schedules become redundant.

Dieleman et al. (2022) further proposes to select the noise schedule adaptively based on the principle that reconstruction loss should increase linearly with the timestep. Yuan et al. (2022) builds on it and suggests fitting the noise schedule for each token position independently.

Shabalin et al. (2025a) analyzes the impact of self-conditioning on generation and find that it increases the magnitude of the model's outputs, which can be interpreted as a measure of confidence. This property can cause a large mismatch between the self-conditioning inputs seen during training and those encountered at inference, since the magnitude at inference is much larger. Based on this observation, the authors propose reducing the number of denoising steps to limit this magnitude growth, which improves generation quality.

Shabalin et al. (2025b) proposes using q-sampling for a specific variant of Gaussian diffusion, arguing that it yields better quality than DDPM on seq2seq tasks, though without explaining the source of this improvement. Karimi Mahabadi et al. (2024) uses the same solver but rounds the model's predictions to the nearest embedding at each denoising step. This ensures that all predictions match the magnitude of the original data, at the cost of introducing bias into the predictions.

Pynadath et al. (2025) provide a complementary perspective on the failures of continuous text diffusion. They consider two perspective of signal degradation: discrete identity corruption (is the correct token identifiable?) and continuous rank degradation (how distinguishable is the correct token?), attributing model failures to a temporal dissonance between the two. However, their analysis is limited to one-hot token representations, which disregard semantic relationships between tokens. As a result, their findings do not directly transfer to the more common setting considered in this paper, where tokens are represented as shallow or context-dependent embeddings and such dissonance does not arise.

B Random Hierarchy Model: setup description

Data generation RHM consists of L hierarchy levels. At each level l, a feature from vocabulary V is expanded into a tuple of s lower-level features. Each feature has m distinct but equivalent expansions (synonyms). This creates a tree of depth L and branching factor s, where $n = s^L$ leaves are the observable input features. The number of possible sequences to generate is $N = m^{\sum_{i=0}^{L-1} s^i} = m^{\frac{n-1}{s-1}}$, while the number of all unique sequences is V^n. In our setting, $L = 3$, $m = 4$, $s = 2$, $V = 16$, meaning that $N = 16384$ and $V^n \approx 4 \cdot 10^9$. The sequence length $n = 8$. Throughout the paper, we refer to all sequences that can be generated by RHM as *correct* and all others as *incorrect*. We use 25 steps during sampling with *sqrt* noise schedule (Li et al., 2022). Our diffusion model has a Transformer-encoder architecture with 4 Transformer layers and the hidden size of 64. Attention layers have 4 attention heads and the imtermediate hidden size in FFN is 64. To work with the inputs of dimention $d = 2$, we upsample the input and downsample the output with linear layers.

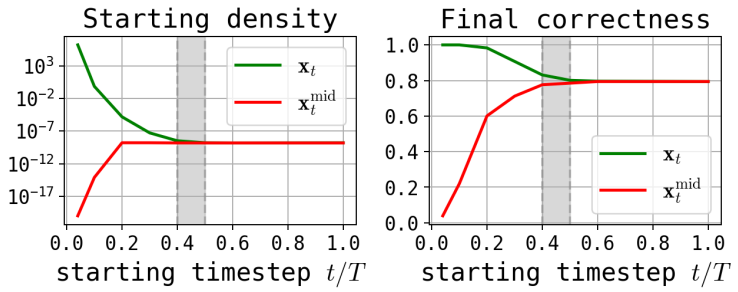

Figure 7: (left) Density of the starting latent sample. (right) Correctness of the generation trajectories started from $\mathbf{x}_t$ and $\mathbf{x}_t^{\text{mid}}$. The *transition interval* defined in Section 3.2 is marked with the gray band.

Density and optimal model The density of noisified data $p(\mathbf{x}_t)$ is given by

$$p(\mathbf{x}_t) = \sum_{\mathbf{x}_0} p(\mathbf{x}_t|\mathbf{x}_0)p(\mathbf{x}_0) = \frac{1}{(2\pi(1-\bar{\alpha}_t))^{\frac{nd}{2}}} \frac{1}{N} \sum_{\mathbf{x}_0} \exp\left(-\frac{\|\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\mathbf{x}_0\|^2}{2(1-\bar{\alpha}_t)}\right) \tag{6}$$

The optimal prediction $\mathbf{x}_0^*$ for $\mathbf{x}_t$ that minimizes Eq 1 can be explicitly calculated as

$$\mathbf{x}_0^*(\mathbf{x}_t) = \mathbb{E}[\mathbf{x}_0|\mathbf{x}_t] = \sum_{\mathbf{x}_0} \mathbf{x}_0 p(\mathbf{x}_0|\mathbf{x}_t) = \frac{1}{N} \sum_{\mathbf{x}_0} \mathbf{x}_0 \frac{p(\mathbf{x}_t|\mathbf{x}_0)}{p(\mathbf{x}_t)} \tag{7}$$

Measuring one-mode rate To determine whether two points $\mathbf{x}_t^1, \mathbf{x}_t^2$ lie within the same probability mode, we calculate a log-density barrier $B(\mathbf{x}_t^1, \mathbf{x}_t^2)$ inspired by research on linear mode connectivity (Entezari et al., 2022):

$$B(\mathbf{x}_t^1, \mathbf{x}_t^2) = \inf_{\alpha \in [0,1]} \left\{ \log p\big(\alpha \mathbf{x}_t^1 + (1-\alpha)\mathbf{x}_t^2\big) - \alpha \log p(\mathbf{x}_t^1) - (1-\alpha) \log p(\mathbf{x}_t^2) \right\}$$

When $B(\mathbf{x}_t^1, \mathbf{x}_t^2) < 0$, then there exists a point on a line segment between $\mathbf{x}_t^1$ and $\mathbf{x}_t^2$, where the log-density is below the interpolated log-density at the endpoints, implying that the endpoints are located in different modes.

C Low-density latent samples result in incorrect predictions

In this section, we empirically support the claim that latents $\mathbf{x}_t$ with low density tend to be denoised into incorrect samples. On the toy RHM example, we compare trajectories started from $\mathbf{x}_t \sim q(\mathbf{x}_t \mid \mathbf{x}_0)$ with trajectories started from a deliberately low-density latent. Specifically, we take a norm-preserving average of two independent latents, $\mathbf{x}_t^{\text{mid}} = (\mathbf{x}_t^{(1)} + \mathbf{x}_t^{(2)})/\sqrt{2}$, where $\mathbf{x}_t^{(i)} \sim q(\mathbf{x}_t \mid \mathbf{x}_0^{(i)})$. The rationale is that after the *transition interval*, $\mathbf{x}_t^{(1)}$ and $\mathbf{x}_t^{(2)}$ lie in different modes with high probability, so their midpoint falls between the modes and has low density. This construction isolates the effect of the starting density and lets us test directly whether the model can recover once a trajectory enters a low-density region.

Figure 7 (left) plots the density of the two starting points as a function of the starting timestep t/T. Before the transition interval, where the latent distribution is still unimodal, the two densities coincide: averaging two samples from a single mode yields another typical sample. After it, once the modes separate, the density of $\mathbf{x}_t^{\text{mid}}$ drops by many orders of magnitude, making it an out-of-distribution input for the model. Figure 7 (right) shows the correctness of the corresponding generations and mirrors this picture. Before the transition interval the two starting points are indistinguishable in terms of the final correctness, whereas after it the curves separate: correctness of the $\mathbf{x}_t$ trajectories approaches 1, while correctness of the

$\mathbf{x}_t^{\text{mid}}$ trajectories collapses toward 0. The onset of this drop coincides with the onset of the density gap, confirming that the diffusion model rarely recovers once its trajectory enters a low-density region.

D Variance of latents updated with q-sampling and DDPM

In this appendix, we show that q-sampling deviates more from the forward process than DDPM: although both solvers produce marginal variances below that of the forward process, the marginal variance under q-sampling is even lower. To simplify the analysis, we assume that $\mathbf{x}_t$ is sampled from the marginal density $p(\mathbf{x}_t)$ and compare the variances of the one-step updates $\mathbf{x}_{t-1}^{\text{DDPM}}$ and $\mathbf{x}_{t-1}^{\text{QS}}$.

D.1 Setup

Consider a data distribution supported on a finite set of points, $p_{\text{data}}(\mathbf{x}) = \frac{1}{N}\sum_{\mathbf{x}_0} \delta(\mathbf{x} - \mathbf{x}_0)$, and a forward diffusion process

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\,\boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I}). \tag{8}$$

We compare two reverse sampling procedures, both using the Bayes-optimal prediction $\mathbf{x}_0^*(\mathbf{x}_t) := \mathbb{E}[\mathbf{x}_0|\mathbf{x}_t]$, with $\mathbf{x}_t \sim p(\mathbf{x}_t)$.

DDPM solver

$$\mathbf{x}_{t-1}^{\text{DDPM}} = a\,\mathbf{x}_0^*(\mathbf{x}_t) + b\,\mathbf{x}_t + \sqrt{\tilde{\beta}_t}\,\boldsymbol{\epsilon}', \qquad \boldsymbol{\epsilon}' \sim \mathcal{N}(0, \mathbf{I}), \tag{9}$$

where

$$a = \frac{\sqrt{\bar{\alpha}_{t-1}}\,\beta_t}{1-\bar{\alpha}_t}, \qquad b = \frac{\sqrt{\alpha_t}\,(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}, \qquad \tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\,\beta_t. \tag{10}$$

q-sampling solver

$$\mathbf{x}_{t-1}^{\text{QS}} = \sqrt{\bar{\alpha}_{t-1}}\,\mathbf{x}_0^*(\mathbf{x}_t) + \sqrt{1-\bar{\alpha}_{t-1}}\,\boldsymbol{\epsilon}'', \qquad \boldsymbol{\epsilon}'' \sim \mathcal{N}(0, \mathbf{I}). \tag{11}$$

In both cases, $\boldsymbol{\epsilon}'$ and $\boldsymbol{\epsilon}''$ are independent of $(\mathbf{x}_0, \mathbf{x}_t)$.

D.2 Result

Theorem 1. *Let $\mathbf{x}_t \sim p(\mathbf{x}_t)$ and define $\mathbf{V}_t := \mathbb{E}[\mathrm{Var}[\mathbf{x}_0|\mathbf{x}_t]]$, the average posterior uncertainty. Then*[2]

$$\mathrm{Var}[\mathbf{x}_{t-1}^{\text{DDPM}}] - \mathrm{Var}[\mathbf{x}_{t-1}^{\text{QS}}] = b^2\bar{\alpha}_t\,\mathbf{V}_t \geq 0, \tag{12}$$

Moreover, both underestimate the true forward marginal variance:

$$\mathrm{Var}_p[\mathbf{x}_{t-1}] - \mathrm{Var}[\mathbf{x}_{t-1}^{\text{DDPM}}] = (\bar{\alpha}_{t-1} - b^2\bar{\alpha}_t)\,\mathbf{V}_t, \tag{13}$$

$$\mathrm{Var}_p[\mathbf{x}_{t-1}] - \mathrm{Var}[\mathbf{x}_{t-1}^{\text{QS}}] = \bar{\alpha}_{t-1}\,\mathbf{V}_t. \tag{14}$$

Proof. Since $\mathbf{x}_t \sim p(\mathbf{x}_t)$, there exists an underlying $\mathbf{x}_0 \sim p_{\text{data}}(\mathbf{x}_0)$ such that $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\,\boldsymbol{\epsilon}$. We decompose $\mathbf{x}_0$ into its prediction and residual:

$$\mathbf{x}_0 = \mathbb{E}[\mathbf{x}_0|\mathbf{x}_t] + \boldsymbol{\delta}, \qquad \boldsymbol{\delta} := \mathbf{x}_0 - \mathbb{E}[\mathbf{x}_0|\mathbf{x}_t]. \tag{15}$$

By the defining property of conditional expectation, $\boldsymbol{\delta}$ is uncorrelated with any function of $\mathbf{x}_t$, and in particular with $\mathbb{E}[\mathbf{x}_0|\mathbf{x}_t]$. Moreover,

$$\mathbb{E}[\boldsymbol{\delta}\boldsymbol{\delta}^\top] = \mathbb{E}[\mathrm{Var}[\mathbf{x}_0|\mathbf{x}_t]] =: \mathbf{V}_t. \tag{16}$$

[2]We operate with covariance matrices, so $\mathbf{A} \geq \mathbf{B}$ denotes that $\mathbf{A} - \mathbf{B}$ is positive semidefinite.

Step 1: Variance of the q-sampling solver. Since $\mathbb{E}[\mathbf{x}_0|\mathbf{x}_t]$ is a function of $\mathbf{x}_t$ and $\boldsymbol{\epsilon}''$ is independent:

$$\mathrm{Var}[\mathbf{x}_{t-1}^{\mathrm{QS}}] = \bar{\alpha}_{t-1}\,\mathrm{Var}[\mathbb{E}[\mathbf{x}_0|\mathbf{x}_t]] + (1-\bar{\alpha}_{t-1})\,\mathbf{I}. \tag{17}$$

Step 2: Expanding the DDPM mean. Substituting the decomposition from Eq. 15 into $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\,\boldsymbol{\epsilon}$:

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\,\mathbb{E}[\mathbf{x}_0|\mathbf{x}_t] + \sqrt{\bar{\alpha}_t}\,\boldsymbol{\delta} + \sqrt{1-\bar{\alpha}_t}\,\boldsymbol{\epsilon}. \tag{18}$$

The DDPM mean $\mu(\mathbf{x}_t) := a\,\mathbb{E}[\mathbf{x}_0|\mathbf{x}_t] + b\,\mathbf{x}_t$ becomes:

$$\mu(\mathbf{x}_t) = \underbrace{(a + b\sqrt{\bar{\alpha}_t})}_{\sqrt{\bar{\alpha}_{t-1}}}\,\mathbb{E}[\mathbf{x}_0|\mathbf{x}_t] + b\sqrt{\bar{\alpha}_t}\,\boldsymbol{\delta} + b\sqrt{1-\bar{\alpha}_t}\,\boldsymbol{\epsilon}, \tag{19}$$

where the identity $a + b\sqrt{\bar{\alpha}_t} = \sqrt{\bar{\alpha}_{t-1}}$ follows from:

$$\begin{aligned} a + b\sqrt{\bar{\alpha}_t} &= \frac{\sqrt{\bar{\alpha}_{t-1}}\,\beta_t}{1-\bar{\alpha}_t} + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\sqrt{\bar{\alpha}_t} = \left\{\text{since } \sqrt{\alpha_t}\sqrt{\bar{\alpha}_t} = \alpha_t\sqrt{\bar{\alpha}_{t-1}}\right\} \\ &= \frac{\sqrt{\bar{\alpha}_{t-1}}\big[\beta_t + \alpha_t(1-\bar{\alpha}_{t-1})\big]}{1-\bar{\alpha}_t} = \frac{\sqrt{\bar{\alpha}_{t-1}}(1-\bar{\alpha}_t)}{1-\bar{\alpha}_t} = \sqrt{\bar{\alpha}_{t-1}}, \end{aligned} \tag{20}$$

using $\beta_t + \alpha_t(1-\bar{\alpha}_{t-1}) = (1-\alpha_t) + \alpha_t - \alpha_t\bar{\alpha}_{t-1} = 1-\bar{\alpha}_t$.

Step 3: Variance of the DDPM solver. The three terms in Eq. 19 are mutually uncorrelated: $\mathbb{E}[\mathbf{x}_0|\mathbf{x}_t]$ is a function of $\mathbf{x}_t$, $\boldsymbol{\delta}$ is uncorrelated with any function of $\mathbf{x}_t$ by construction (Eq. 15), and $\boldsymbol{\epsilon}$ is independent of $\mathbf{x}_0$. Therefore:

$$\mathrm{Var}[\mu(\mathbf{x}_t)] = \bar{\alpha}_{t-1}\,\mathrm{Var}[\mathbb{E}[\mathbf{x}_0|\mathbf{x}_t]] + b^2\bar{\alpha}_t\,\mathbf{V}_t + b^2(1-\bar{\alpha}_t)\,\mathbf{I}. \tag{21}$$

Adding the noise term $\tilde{\beta}_t\boldsymbol{\epsilon}'$:

$$\mathrm{Var}[\mathbf{x}_{t-1}^{\mathrm{DDPM}}] = \bar{\alpha}_{t-1}\,\mathrm{Var}[\mathbb{E}[\mathbf{x}_0|\mathbf{x}_t]] + b^2\bar{\alpha}_t\,\mathbf{V}_t + \underbrace{\big(b^2(1-\bar{\alpha}_t) + \tilde{\beta}_t\big)}_{1-\bar{\alpha}_{t-1}}\,\mathbf{I}, \tag{22}$$

where the identity $b^2(1-\bar{\alpha}_t) + \tilde{\beta}_t = 1-\bar{\alpha}_{t-1}$ follows from:

$$\begin{aligned} b^2(1-\bar{\alpha}_t) + \tilde{\beta}_t &= \frac{\alpha_t(1-\bar{\alpha}_{t-1})^2}{(1-\bar{\alpha}_t)^2}(1-\bar{\alpha}_t) + \frac{(1-\bar{\alpha}_{t-1})\beta_t}{1-\bar{\alpha}_t} \\ &= \frac{(1-\bar{\alpha}_{t-1})\big[\alpha_t(1-\bar{\alpha}_{t-1}) + \beta_t\big]}{1-\bar{\alpha}_t} = \frac{(1-\bar{\alpha}_{t-1})(1-\bar{\alpha}_t)}{1-\bar{\alpha}_t} = 1-\bar{\alpha}_{t-1}. \end{aligned} \tag{23}$$

Step 4: Comparison. Subtracting Eq. 17 from Eq. 22:

$$\boxed{\mathrm{Var}[\mathbf{x}_{t-1}^{\mathrm{DDPM}}] - \mathrm{Var}[\mathbf{x}_{t-1}^{\mathrm{QS}}] = b^2\bar{\alpha}_t\,\mathbf{V}_t \geq 0.} \tag{24}$$

Step 5: Comparison with the true marginal. The forward marginal variance is $\mathrm{Var}_p[\mathbf{x}_{t-1}] = \bar{\alpha}_{t-1}\,\mathrm{Var}[\mathbf{x}_0] + (1-\bar{\alpha}_{t-1})\mathbf{I}$. By the law of total variance,

$$\mathrm{Var}[\mathbf{x}_0] = \mathrm{Var}[\mathbb{E}[\mathbf{x}_0|\mathbf{x}_t]] + \mathbf{V}_t, \tag{25}$$

so $\mathrm{Var}[\mathbb{E}[\mathbf{x}_0|\mathbf{x}_t]] = \mathrm{Var}[\mathbf{x}_0] - \mathbf{V}_t$. Substituting into Eq. 17 and Eq. 22:

$$\mathrm{Var}_p[\mathbf{x}_{t-1}] - \mathrm{Var}[\mathbf{x}_{t-1}^{\mathrm{QS}}] = \bar{\alpha}_{t-1}\,\mathbf{V}_t, \tag{26}$$

$$\mathrm{Var}_p[\mathbf{x}_{t-1}] - \mathrm{Var}[\mathbf{x}_{t-1}^{\mathrm{DDPM}}] = (\bar{\alpha}_{t-1} - b^2\bar{\alpha}_t)\,\mathbf{V}_t. \tag{27}$$

□

Remark 1. *The difference $b^2\bar{\alpha}_t\,\mathbf{V}_t$ vanishes in two regimes: (i) when $\mathbf{V}_t \to 0$, i.e., the posterior $p(\mathbf{x}_0|\mathbf{x}_t)$ concentrates on a single point (small t); (ii) when $\bar{\alpha}_t \to 0$ (large t, near pure noise). The gap is largest at intermediate noise levels where the posterior is genuinely uncertain yet $\bar{\alpha}_t$ is still appreciable.*

Remark 2. *Both solvers underestimate the true marginal variance, but by different amounts. The q-sampling solver discards all posterior uncertainty through resampling fresh noise, losing $\bar{\alpha}_{t-1}\mathbf{V}_t$. The DDPM solver partially recovers this through the coupling between $\mathbb{E}[\mathbf{x}_0|\mathbf{x}_t]$ and $\mathbf{x}_t$ in its mean formula, retaining $b^2\bar{\alpha}_t\mathbf{V}_t$ and losing only $(\bar{\alpha}_{t-1} - b^2\bar{\alpha}_t)\mathbf{V}_t$.*

	RHM	ROCStories	Wikipedia	OWT
Max length	8	80	128	512
	XSum	**QQP**	**IWSLT14**	**CoNaLa**
Max input length	512	50	64	64
Max target length	64	50	64	64

Table 3: Max sequence lengths used for each dataset.

E Implementation details

In this paper, we train models from prior work using their original code, keeping the tokenization methods, model architectures and all hyperparameters unchanged. Below we describe all introdused modifications which were required for our analyzis.

Self-conditioning Diffusion-LM (Li et al., 2022) and GENIE (Lin et al., 2023b) do not include SC in their original implementations. We add SC support following the approach of SeqDiffuSeq (Yuan et al., 2022), since all three models share the same codebase. In all our experiments, we the same model for generating with and without SC. As every model trained with SC also learns to make predictions without it, we can turn SC on and off during the inference to avoid model retrainig.

Adaptation of GENIE to code generation The original GENIE paper does not provide a code generation pipeline, and while CodeFusion (Singh et al., 2023) builds on the GENIE method, no implementation is publicly available. We therefore reimplement the CodeFusion setup using GENIE's source code. Additionally, the original approach includes a pre-training stage before fine-tuning; we omit this step to reduce computational cost, as the original paper reports only a marginal impact on quality, and we focus on the generation process rather than achieving state-of-the-art performance.

Number of generation steps Diffusion-LM, SeqDiffuSeq, and GENIE originally use more than 1000 generation steps, which we find excessive and impractical. We reduce the number of steps to 200 for Diffusion-LM, 100 for GENIE, and 50 for SeqDiffuSeq; preliminary experiments confirm that this has a negligible impact on seq2seq quality. For TEncDM-based models (Shabalin et al., 2025a), we use 50 steps as recommended in the original paper, except for MBR on ROCStories and OpenWebText, where we use 100 steps. The increased step count for MBR is chosen to better demonstrate the effect, as 50 steps yield lower quality for embedding-based diffusion and make the improvement less pronounced. For OpenWebText, which contains sequences of 512 tokens (substantially longer than other datasets, see Table 3), 100 steps also better suit the task. The effect of the number of generation steps on unconditional generation quality is analyzed in Appendix I.

F Datasets

F.1 Unconditional generation

ROCStories The dataset (Mostafazadeh et al., 2016) contains five-sentence commonsense stories and consists of 98 161 instances. 93 161 instances are held out for training, and 5 000 instances are used for validation.

Wikipedia For experiments of larger scale, we utilize the English Wikipedia dataset (Foundation). The dataset comprises a total of 6.4M sequences, of which 10k are allocated as a validation set.

OpenWebText OpenWebText (Gokaslan et al., 2019) is the largest dataset we use. It contains 8M texts, which we truncate to the length of 512 tokens without any additional preprocessing. We allocate 100k samples for validation and keep the remaining texts for training.

F.2 Conditional generation

XSum The dataset (Narayan et al., 2018) is used for summarization task and it contains 204k BBC articles, which are provided as document and summary pairs and covered wide range of topics (Sports, Politics, etc.). It has 204 045 training instances, 11 332 validation instances, and 11 334 test instances.

QQP The subset of QQP dataset, proposed in DataCanary et al. (2017), consists of 149k question pairs from the Quora platform that are paraphrases of each other. The data is splitted in 144 715 training instances, 2 048 validation instances, and 2 500 test instances.

IWSLT14 IWSLT14 is a widely used benchmark for machine translation. We use the German-English language pairs and translate from German to English. We follow the preprocessing pipeline of fairseq (Ott et al., 2019), which relies on the Moses tokenizer (Koehn et al., 2007). In IWSLT14, 160 239 pairs are allocated for training, 7 283 are used for validation, and 6 750 for testing.

CoNaLa The CoNaLa dataset (Yin et al., 2018) consists of complex, multi-statement StackOverflow Python code snippets and associated NL questions. It has 594k train pairs and 500 test pairs.

G Minimum Bayes-Risk examples

In this section, we present examples of MBR selection for RHM and ROCStories dataset.

G.1 Random Hierarchy Model

MBR for two different generation trajectories is presented in Table 4. For the smallest activation timesteps $t_{\text{act}}/T = 0.1$ and $t_{\text{act}}/T = 0.3$, the candidates are either identical or differ in only a few tokens. MBR is therefore ineffective in this regime: the sample has been mostly determined (i.e., the prediction $\hat{\mathbf{x}}_\theta$ at $t_{\text{act}}/T = 0.1$ coincides with the majority of candidates), leaving no room to correct invalid outputs. At $t_{\text{act}}/T = 0.5$, by contrast, the candidate sequences are substantially more diverse, containing both correct and incorrect samples; here MBR successfully identifies and selects the correct ones. Finally, at $t_{\text{act}}/T = 0.7$ the candidates are too diverse for averaging to be meaningful: in the presented examples, MBR selects an incorrect sequence even though correct candidates are present in the set.

G.2 ROCStories

Table 5 presents a representative set of examples for the TEncDM Emb model without self-conditioning on the ROCStories dataset. At $t_{\text{act}} = 0.3$, candidates differ only in a few words, typically proper names or contextual synonyms, reflecting that most of the generation has already been committed. At $t_{\text{act}} = 0.5$, variation extends to word combinations and phrasing, with candidates differing in stylistic suitability while preserving the overall theme. At $t_{\text{act}} = 0.7$, the texts diverge substantially, sharing only high-level structural elements such as key pronouns (e.g., "they"), which leaves MBR with too little signal to reliably select the best candidate. These examples confirm that MBR is most effective when applied around $t_{\text{act}} = 0.5$, where candidates are similar enough for meaningful comparison yet diverse enough to benefit from selection.

H q-sampling ablation

In this section, we analyze how generation quality varies with the q-sampling activation timestep t_{act}. Results are presented in Figures 8 and 9. Overall, enabling self-conditioning consistently improves Mauve and reduces perplexity while also decreasing diversity, which is consistent across all datasets and models. Regarding the effect of t_{act}, two trends emerge. First, activating q-sampling earlier (larger t_{act}/T) mostly monotonically degrades both perplexity and diversity. Second, Mauve exhibits a clear peak as a function of t_{act}.

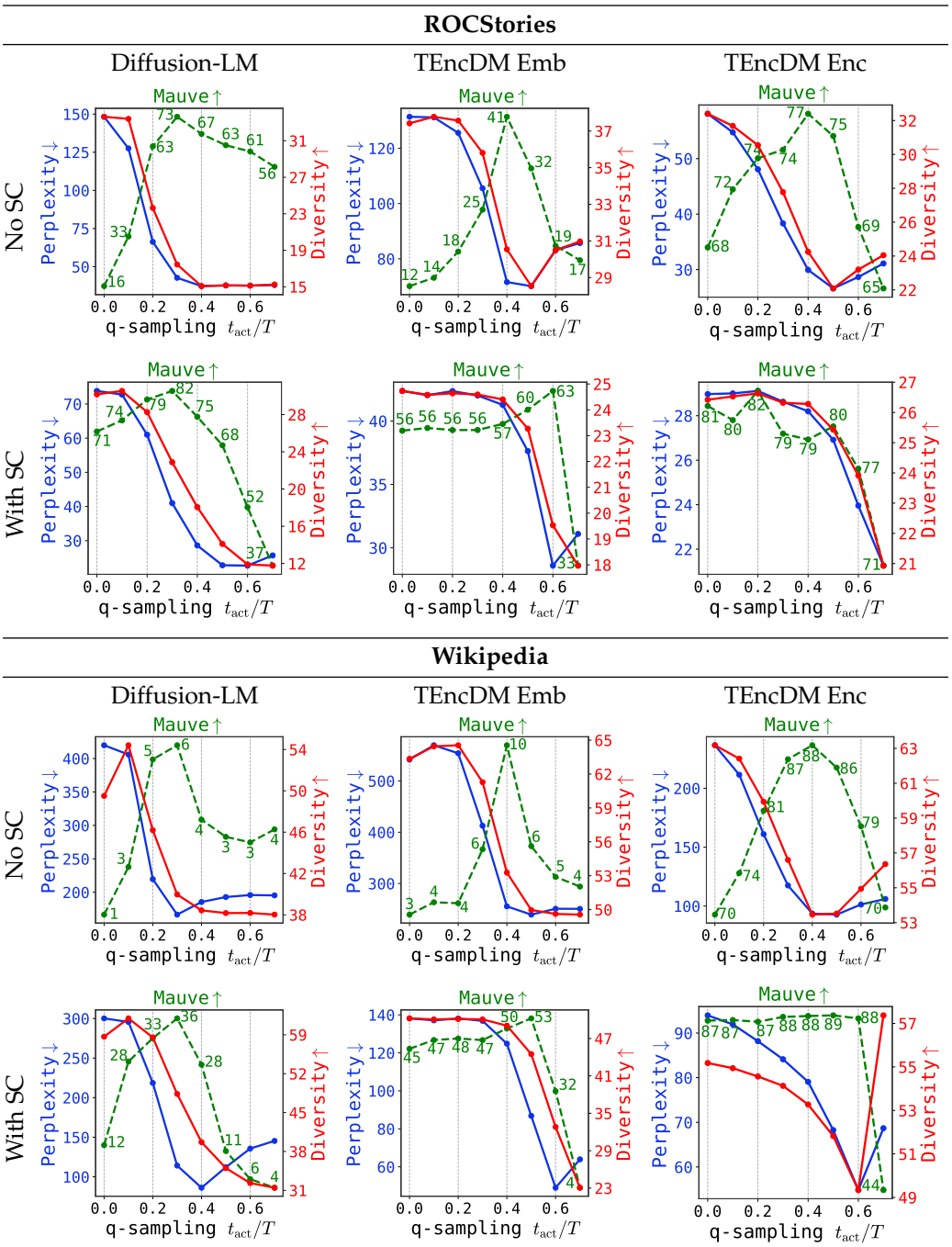

Figure 8: Quality metrics with and without self-conditoning (SC) for different q-sampling activation timestep on ROCStories and Wikipedia. We use the baseline SC model with $p = 0.5$.

	$t_{\text{act}}/T = 0.1$	$t_{\text{act}}/T = 0.3$	$t_{\text{act}}/T = 0.5$	$t_{\text{act}}/T = 0.7$
MBR candidate	blbbjbjk	blfbjbjk	infblddi	knjioapa
Candidate #1	blbbjbjk	icbbjbjk	infblddi	kgockihn
Candidate #2	blbbjbjk	infbjbjk	nobhkoki	hncefble
Candidate #3	blbbjbjk	blfbjbok	enfboomb	knjioapa
Candidate #4	blbbjbjk	blfbjbjk	icbbldjb	jkjmaioa
Candidate #5	blbbjbjk	blfbjbjk	bgfbjbhb	dejdpbcb
Prediction $\hat{\mathbf{x}}_\theta$ at t_{act}	blbbjbjk	blfbjbmk	nmnbmimb	nphnhnmn
MBR candidate	ldmmkoda	ldmmkoda	jdockida	jdocoako
Candidate #1	ldmmkoda	ldmmkoda	jdockoda	jdocoako
Candidate #2	ldmmkoda	ldmmkida	phockidi	jdocokof
Candidate #3	ldmmkoda	ldmmkoda	jdockida	mmofecdi
Candidate #4	ldmmkida	ldmmkoda	jdockidi	noilicle
Candidate #5	ldmmkoda	ldmmkida	jdockoda	jkblicba
Prediction $\hat{\mathbf{x}}_\theta$ at t_{act}	ldmmkoda	ldmmkida	jdocdidf	moomhoof

Table 4: Five MBR candidates and chosen samples for different activation timesteps t_{act}. We use latin letters to represent each token. Correct sequences are highlighted in green and incorrect in red. Activating MBR at $t_{\text{act}}/T = 0.3$ or $t_{\text{act}}/T = 0.5$ ensures that the correct sequence is sampled; this would not be the case without MBR.

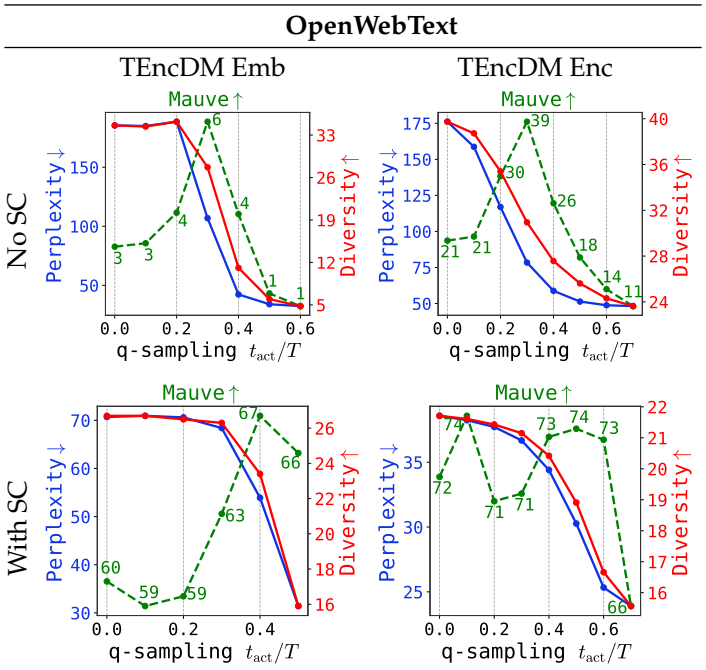

Figure 9: Quality metrics with and without self-conditoning (SC) for different q-sampling activation timestep on OpenWebText. We use the baseline SC model with $p = 0.5$.

	$t_{\text{act}}/T = 0.3$
MBR	John and John decided to build a lemonade stand. They got together in a big lemonade stand in their yard. picking out the lemonade, they put it in a jar. They ate the lemonade for a week. The neighbor told them to go home without their lemonade.
#1	Amy and John decided to build a lemonade stand. They walked together in a big lemonade stand in their yard. picking out the lemonade, they put it in a bowl. They ate the lemonade for a week. The neighbor told them to go home without their lemonade.
#2	Sam and John decided to make a lemonade stand. They got together in the big lemonade stand in their yard. picking out the lemonade, they put it in a bowl. They ate the lemonade for a week. The neighbor told them to go home without their lemonade.
#3	John and John decided to build a lemonade stand. They sat together in a large lemonade stand in their yard. picking out the lemonade, they put it in a box. They ate the lemonade for a week. The neighbor told them to go home without their lemonade.
#4	John and John decided to make a lemonade stand. They got together in a large lemonade stand in their yard. picking out the lemonade, they put it in a jar. They ate the lemonade for a week. The neighbor told them to go home without their lemonade.
	$t_{\text{act}}/T = 0.5$
MBR	Joe and Joe decided to have a lemonade stand. They quickly set up the best lemonade stand in their yard. They made the lemonade, and put it in a bowl. They ate the lemonade for an hour. The owner told them to go home with the lemonade.
#1	Larry and Bob decided to make a lemonade stand. They went down to the best lemonade stand in their yard. They bought the lemonade, and put it in a box. They ate the lemonade for an hour. The neighbor told them to go back with the lemonade.
#2	Ron and Mark decided to make a lemonade stand. They found some of the lemonade stand stand in their yard. They picked the lemonade, and put them in a bowl. They ate the lemonade for an hour. The owner told them to go home without their lemonade.
#3	Joe and Joe decided to have a lemonade stand. They quickly set up the best lemonade stand in their yard. They made the lemonade, and put it in a bowl. They ate the lemonade for an hour. The owner told them to go home with the lemonade.
#4	Kim and Mark decided to have a lemonade sale. They both went to an ice lemonade stand in the yard. They got the lemonade, and placed them in a bowl. They ate the lemonade for an hour. The manager told them to go home with their lemonade.
	$t_{\text{act}}/T = 0.7$
MBR	Ava and her friends went to grow tomato flowers every week. They found a large arout in the field. Then she watered them brouted one more time. Soon they could find three dozen flowers. They were relieved when the owner told them to go home safely.
#1	A local artist wanted to start a new music business. There was a contest that wanted to do a popie. He went on the first show and it was great. When they arrived to stage the whole crowd felt very sad. The artist told them to leave, so they left early.
#2	Ava and her friends went to grow tomato flowers every week. They found a large arout in the field. Then she watered them brouted one more time. Soon they could find three dozen flowers. They were relieved when the owner told them to go home safely.
#3	Rita and Roger decided they would own their own farm. They visited a farm and picked two free lorots. The chef people from the farm were happy to have them. They offered to pay for the lorots for a while. The mechanic told them that they would return for good again.
#4	John and his friends went to an ice cream shop. They picked a few chips and sugar candy soda. Then, John started eating a slice. After that slice they had 10 minutes in one bite. John and his friends walked for a few that day to go shopping and go home.

Table 5: Four MBR candidates and the chosen sample for different activation timesteps t_{act}. Parts matching the selected candidate are highlighted in green; differing parts in red. Parts shared across all candidates are not highlighted.

Without SC, this peak is pronounced and consistent across datasets: $t_{\text{act}}/T = 0.3$ for Diffusion-LM and $t_{\text{act}}/T \approx 0.3$–$0.4$ for TEncDM Emb and TEncDM Enc. With SC, pure DDPM ($t_{\text{act}}/T = 0$) already provides a strong baseline (except for Diffusion-LM on Wikipedia), so the relative gain from q-sampling is smaller, though the optimal threshold shifts slightly: $t_{\text{act}}/T = 0.3$ for Diffusion-LM and $t_{\text{act}}/T = 0.5$ for TEncDM Emb. For TEncDM Enc, q-sampling yields only a marginal improvement over pure DDPM regardless of t_{act}, which is consistent with the intuition that context-dependent encodings already produce a smoother latent space more suitable for Gaussian diffusion. In all settings, activating q-sampling too early ($t_{\text{act}}/T = 0.7$) leads to degraded performance, in line with our findings from the RHM experiments.

I Changing the number of steps

In this section, we provide a qualitative comparison of the TEncDM Emb and TEncDM Enc models when using different numbers of generation steps at inference. We choose TEncDM-based models because they do not require retraining to change the number of steps.

In Figure 10 and Figure 11, we show how varying the number of steps affects generation quality for models without and with SC, respectively. For models with SC, increasing the number of steps significantly reduces both perplexity and diversity, regardless of whether q-sampling is used. This is consistent with the findings of Shabalin et al. (2025a) and can be explained by the growing SC mismatch discussed in Appendix A.

The results for models without SC are more nuanced. In the DDPM setting ($t_{\text{act}}/T = 0$), the difference between runs with different numbers of steps is marginal. However, as the q-sampling activation interval increases, the gap widens: perplexity and diversity drop more rapidly with more steps.

This behavior can be explained using the analysis in Appendix D. The q-sampling solver reduces the variance of $\mathbf{x}_{t-1}$ at each update compared to DDPM. Consequently, the more updates that are performed with q-sampling, the lower the resulting variance. With larger number of steps, the final latent ends up closer to the center of the distribution, meaning it is less committed to any particular text. This leads to lower diversity and perplexity, as the two are correlated.

When DDPM perplexity is high, as for TEncDM Emb, such a reduction in perplexity is beneficial and Mauve increases. However, when perplexity is already low, as for TEncDM Enc, further reduction hurts Mauve.

J Self-conditioning probability

In this section, we ablate the probability p of self-conditioning (SC) used during training. We argue that the commonly used value of $p = 0.5$ is suboptimal: since SC is applied at every iteration during inference, it may be beneficial to expose the model to conditioning more frequently during training.

In Table 6, we report a numerical evaluation across varying values of p and make two observations. First, increasing p generally decreases perplexity, while diversity initially rises and then falls. This suggests that overall text quality improves as p increases, up to a point. This trend is further supported by the Mauve scores, which peak at an intermediate value of p around 0.9. Second, the optimal value of p is difficult to identify, as it varies across models and datasets. As a rule of thumb, however, we recommend setting p to at least 0.8 and tuning upward from there.

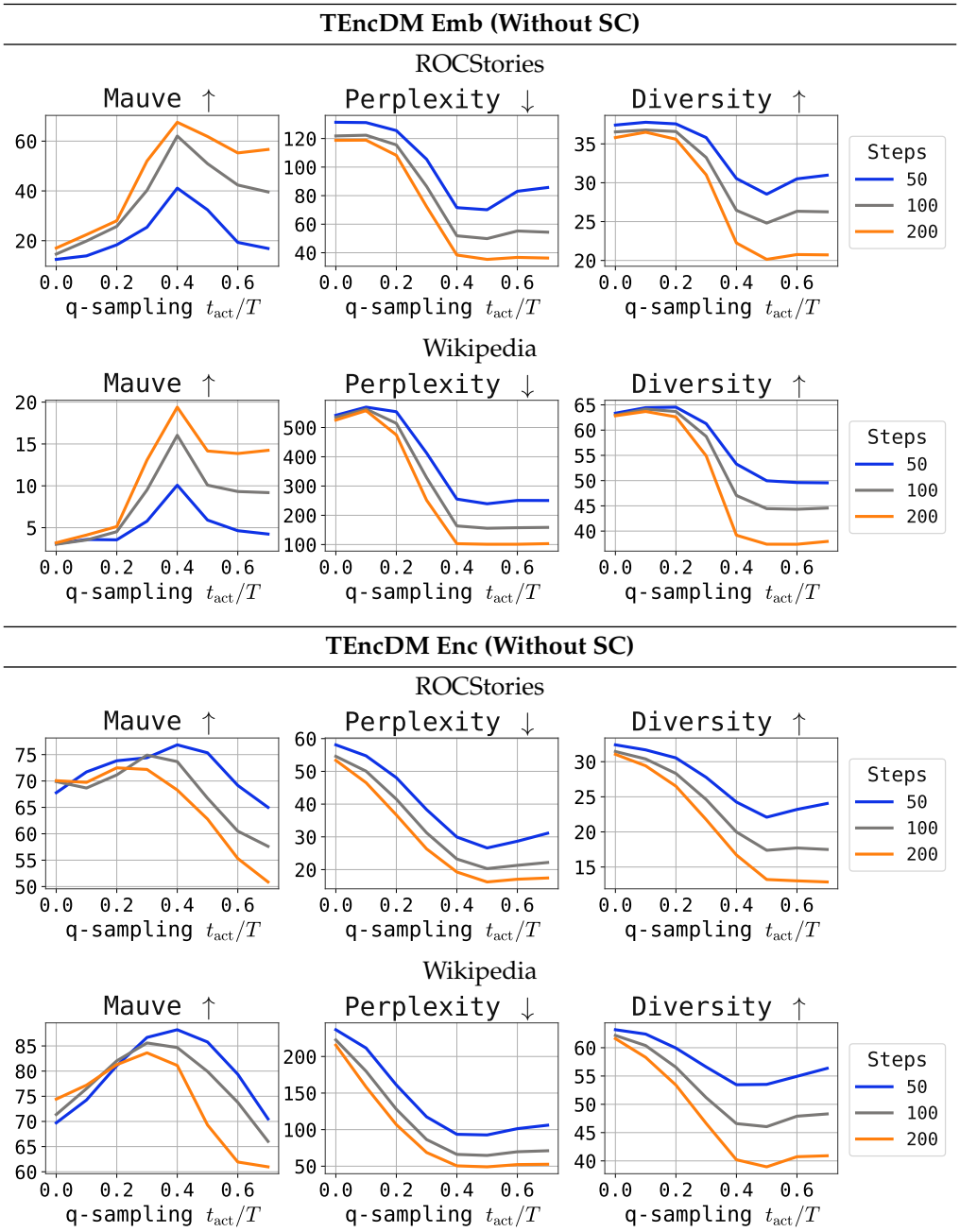

Figure 10: Quality metrics for TEncDM-based models **without self-conditining (SC)** for different q-sampling activation timesteps and varying amount of generation steps on ROCStories and Wikipedia. We use the baseline SC model with $p = 0.5$.

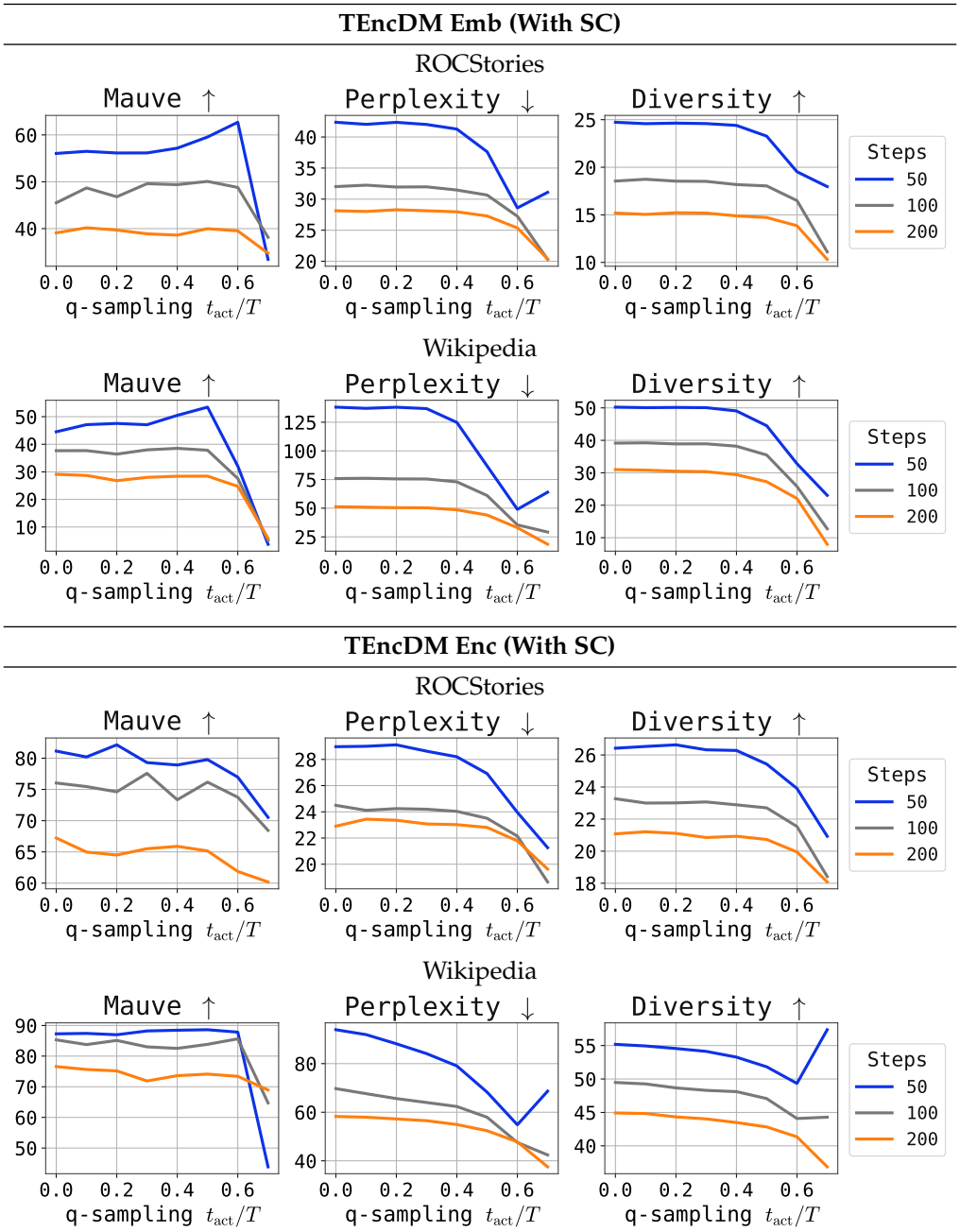

Figure 11: Quality metrics for TEncDM-based models **with self-conditining (SC)** for different q-sampling activation timesteps and varying amount of generation steps on ROCStories and Wikipedia. We use the baseline SC model with $p = 0.5$.

Model	ROCStories			Wikipedia			OpenWebText		
	Mauve ↑	PPL ↓	Div ↑	Mauve ↑	PPL ↓	Div ↑	Mauve ↑	PPL ↓	Div ↑
Diffusion-LM									
+ $SC_{0.5}$	$71.3_{\pm 3.8}$	$74.5_{\pm 0.9}$	$\mathbf{30.2}_{\pm 0.3}$	$12.1_{\pm 1.9}$	$300.2_{\pm 6.7}$	$58.7_{\pm 0.4}$	—	—	—
+ $SC_{0.8}$	$74.0_{\pm 4.3}$	$66.5_{\pm 1.0}$	$30.1_{\pm 0.4}$	$16.8_{\pm 1.6}$	$271.2_{\pm 7.4}$	$\mathbf{58.8}_{\pm 0.5}$	—	—	—
+ $SC_{0.9}$	$74.0_{\pm 1.9}$	$61.8_{\pm 0.5}$	$29.2_{\pm 0.3}$	$24.3_{\pm 3.5}$	$229.5_{\pm 5.1}$	$\mathbf{58.8}_{\pm 0.5}$	—	—	—
+ $SC_{0.95}$	$\mathbf{75.2}_{\pm 2.4}$	$59.5_{\pm 1.1}$	$29.0_{\pm 0.4}$	$27.8_{\pm 3.2}$	$197.4_{\pm 4.8}$	$56.6_{\pm 0.5}$	—	—	—
+ $SC_{0.99}$	$74.8_{\pm 3.4}$	$\mathbf{58.7}_{\pm 1.0}$	$28.8_{\pm 0.3}$	$\mathbf{31.0}_{\pm 4.2}$	$\mathbf{158.0}_{\pm 6.2}$	$48.6_{\pm 1.3}$	—	—	—
TEncDM Embeddings									
+ $SC_{0.5}$	$56.5_{\pm 4.2}$	$42.4_{\pm 0.8}$	$24.8_{\pm 0.3}$	$44.5_{\pm 4.6}$	$138.1_{\pm 2.3}$	$50.2_{\pm 0.4}$	$\mathbf{59.9}_{\pm 4.7}$	$70.7_{\pm 1.0}$	$\mathbf{26.7}_{\pm 0.3}$
+ $SC_{0.8}$	$73.2_{\pm 4.3}$	$37.7_{\pm 0.6}$	$26.2_{\pm 0.5}$	$\mathbf{51.1}_{\pm 3.9}$	$129.6_{\pm 3.0}$	$51.7_{\pm 0.6}$	$48.6_{\pm 4.5}$	$43.9_{\pm 0.5}$	$17.4_{\pm 0.2}$
+ $SC_{0.9}$	$73.6_{\pm 3.6}$	$37.6_{\pm 0.5}$	$\mathbf{28.7}_{\pm 0.4}$	$50.1_{\pm 3.6}$	$126.5_{\pm 4.3}$	$\mathbf{52.9}_{\pm 0.9}$	$35.4_{\pm 4.4}$	$36.9_{\pm 0.5}$	$13.9_{\pm 0.3}$
+ $SC_{0.95}$	$\mathbf{78.9}_{\pm 3.0}$	$\mathbf{36.0}_{\pm 0.5}$	$27.7_{\pm 0.3}$	$46.5_{\pm 4.9}$	$131.6_{\pm 2.9}$	$51.7_{\pm 0.6}$	$34.8_{\pm 4.2}$	$\mathbf{32.9}_{\pm 0.7}$	$12.8_{\pm 0.4}$
+ $SC_{0.99}$	$78.8_{\pm 2.7}$	$36.7_{\pm 0.6}$	$27.0_{\pm 0.4}$	$43.1_{\pm 3.5}$	$\mathbf{116.3}_{\pm 2.5}$	$50.5_{\pm 0.7}$	—	—	—
TEncDM Encodings									
+ $SC_{0.5}$	$81.0_{\pm 2.9}$	$29.1_{\pm 0.5}$	$26.5_{\pm 0.4}$	$87.3_{\pm 3.0}$	$94.0_{\pm 1.8}$	$55.2_{\pm 0.3}$	$71.6_{\pm 3.8}$	$\mathbf{38.6}_{\pm 0.4}$	$21.7_{\pm 0.2}$
+ $SC_{0.8}$	$81.1_{\pm 2.3}$	$30.7_{\pm 0.3}$	$28.4_{\pm 0.4}$	$\mathbf{91.4}_{\pm 1.3}$	$\mathbf{90.2}_{\pm 1.5}$	$57.0_{\pm 0.4}$	$64.3_{\pm 4.5}$	$47.1_{\pm 0.4}$	$24.4_{\pm 0.2}$
+ $SC_{0.9}$	$84.1_{\pm 2.8}$	$\mathbf{29.0}_{\pm 0.5}$	$31.0_{\pm 0.4}$	$90.8_{\pm 1.8}$	$90.8_{\pm 1.4}$	$\mathbf{58.3}_{\pm 0.4}$	$\mathbf{72.4}_{\pm 3.8}$	$49.2_{\pm 0.5}$	$26.5_{\pm 0.2}$
+ $SC_{0.95}$	$\mathbf{86.8}_{\pm 1.9}$	$29.9_{\pm 0.3}$	$\mathbf{31.5}_{\pm 0.4}$	$90.1_{\pm 2.1}$	$92.6_{\pm 1.6}$	$57.3_{\pm 0.3}$	$67.1_{\pm 3.2}$	$50.7_{\pm 0.6}$	$\mathbf{26.6}_{\pm 0.2}$
+ $SC_{0.99}$	$85.2_{\pm 1.5}$	$29.8_{\pm 0.5}$	$31.1_{\pm 0.4}$	$86.4_{\pm 2.5}$	$93.6_{\pm 1.6}$	$56.1_{\pm 0.3}$	—	—	—

Table 6: Impact of self-conditioning (SC) probability on ROCStories, Wikipedia and OpenWebText. SC trained with probability p is denoted SC_p. The SC probability giving the highest Mauve is highlighted in blue.

K Unsuccessful ideas

We conducted preliminary experiments on ROCStories exploring several ideas that did not ultimately improve generation quality, but which we include here as potentially useful context for the reader.

Adding variance to $\mathbf{x}_0$ A key finding of this paper is that more continuous $\mathbf{x}_0$ distributions yield better generation quality. Although unnecessary in principle, methods that train embeddings jointly with the diffusion model typically add a small variance to $\mathbf{x}_0$, making its distribution continuous rather than a delta-mixture. We hypothesized that this smoothing might reduce low-density regions and improve generation. While experiments on RHM supported this idea, the effect did not transfer to real data. Ye et al. (2024) reached a similar conclusion.

Training embeddings jointly with diffusion We hypothesized that since learnable embeddings evolve throughout training, they introduce additional variance into $\mathbf{x}_0$, effectively making the latent space more continuous. To test this, we trained one model with learnable embeddings and another with frozen embeddings extracted from the first model, ensuring both models operate on identical embeddings at inference. As a result, the two models produced equivalent quality. We suppose that embeddings converge early in training; they do not vary much across iterations, and therefore the diffusion model does not perceive the latents as elements of the continuous space, leading to no improvement.

Resampling $\mathbf{x}_t$ for self-conditioning During training, SC is introduced by first computing a prediction with the SC input set to zero, $\bar{\mathbf{x}}_0^t = \hat{\mathbf{x}}_\theta(\mathbf{x}_t, t, \mathbf{0})$, and then using this as the SC input for a second prediction, $\hat{\mathbf{x}}_0^t = \hat{\mathbf{x}}_\theta(\mathbf{x}_t, t, \mathrm{SG}(\bar{\mathbf{x}}_0^t))$. Crucially, $\mathbf{x}_t$ is held fixed between the two predictions during training, whereas during generation it updates between steps. We hypothesized that this discrepancy could hurt generation quality, particularly for q-sampling where the latent changes substantially between iterations. To address this, we experimented with resampling $\mathbf{x}_t$ for the second prediction during training with some probability p', and additionally providing the model with a flag indicating which type of SC input was used. Neither modification improved generation quality.

L Additional results for unconditional generation

In this section, we present additional results for the unconditional generation. Namely, we conduct the experiments on the large scale OpenWebText dataset to show that our theory extends to longer sequences, and on protein generation task to extend our results to discrete domains beyond text.

L.1 OpenWebText results

Model	OpenWebText		
	Mauve ↑	**PPL** ↓	**Div** ↑
TEncDM Emb	$2.6_{\pm 0.4}$	$185.3_{\pm 2.4}$	$\mathbf{34.6}_{\pm 0.2}$
+ $SC_{0.5}$	$59.8_{\pm 4.7}$	$70.6_{\pm 0.8}$	$26.7_{\pm 0.3}$
+ $QS_{0.4}$	$6.2_{\pm 0.8}$	$106.8_{\pm 1.7}$	$27.7_{\pm 0.3}$
+ $QS_{0.4}$ + $SC_{0.5}$	$\mathbf{67.3}_{\pm 4.5}$	$\mathbf{53.9}_{\pm 0.7}$	$23.4_{\pm 0.3}$
TEncDM Enc	$20.8_{\pm 3.0}$	$176.1_{\pm 2.0}$	$\mathbf{39.7}_{\pm 0.3}$
+ $SC_{0.5}$	$71.6_{\pm 3.8}$	$38.6_{\pm 0.4}$	$21.7_{\pm 0.2}$
+ $QS_{0.4}$	$38.5_{\pm 3.9}$	$78.4_{\pm 1.1}$	$30.9_{\pm 0.3}$
+ $QS_{0.4}$ + $SC_{0.5}$	$\mathbf{73.2}_{\pm 3.8}$	$\mathbf{34.4}_{\pm 0.3}$	$20.4_{\pm 0.2}$

Table 7: Impact of q-sampling (QS) and self-conditioning (SC) on OpenWebText. SC trained with probability p is denoted SC_p; QS with optimal activation timestep t_{act} is denoted $QS_{t_{\text{act}}/T}$.

In Table 7, we evaluate the impact of q-sampling and self-conditioning on the OpenWebText dataset. The text length for this task is 512, which is 4 times larger that 128 used for Wikipedia. The results are fully consistent with those in Table 1 for other datasets, though SC proves especially important on longer sequences, yielding a substantial boost in performance. Applying q-sampling within the critical interval also improves quality both with and without SC, confirming that it addresses the DDPM limitations discussed in Section 3.3. As in the other experiments, both SC and q-sampling reduce the diversity of generated texts in exchange for lower perplexity.

L.2 Protein generation results

	FD-seq ↓	**pppl** ↓	$\mathbf{CD}_{0.5}$ ↑
DiMA Emb	$1.086_{\pm 0.006}$	$12.62_{\pm 0.25}$	$87.9_{\pm 0.9}$
+ $SC_{0.5}$	$1.015_{\pm 0.024}$	$11.33_{\pm 0.21}$	$\mathbf{92.5}_{\pm 1.6}$
+ $QS_{0.4}$	$1.045_{\pm 0.009}$	$12.04_{\pm 0.25}$	$87.1_{\pm 1.3}$
+ $QS_{0.4}$ + $SC_{0.5}$	$\mathbf{1.004}_{\pm 0.026}$	$\mathbf{10.89}_{\pm 0.23}$	$91.5_{\pm 1.5}$

Table 8: Impact of q-sampling (QS) and self-conditioning (SC) on SwissProt protein generation. SC trained with probability p is denoted SC_p; QS with optimal activation timestep t_{act} is denoted $QS_{t_{\text{act}}/T}$.

In this section, we demonstrate that the quality improvements from q-sampling and self-conditioning are not specific to text, by conducting experiments on protein generation, a distinct discrete domain. We use the SwissProt subset of the UniProt database (Consortium, 2020).

Model We use the DiMA-35M model for protein sequence generation (Meshchaninov et al., 2025b) trained on shallow ESM-2 150M embeddings (Lin et al., 2023a), following the training protocol described in the original paper. We refer to this model as DiMA Emb, for consistency with TEncDM Emb. We use 100 steps during sampling.

Metrics As a sequence quality metric, we use **pseudoperplexity** (Salazar et al., 2020) (ESM-2 pppl) computed with the ESM-2 650M encoder transformer-based language model (Lin et al., 2023a), and **cluster density** (Meshchaninov et al., 2025b) with an identity threshold of $t = 50\%$ ($CD_{0.5}$) as a diversity measure. For consistency with the text generation experiments, we report $100 \cdot CD_{0.5}$. To assess the overall similarity between generated and reference protein distributions, we use **Fréchet Distance** (FD-seq) computed on top of ProtT5 sequence representations (Heinzinger et al., 2024).

Results Table 8 compares baseline DDPM, self-conditioning, and q-sampling for protein generation. Similarly to text experiments, both techniques reduce pseudoperplexity, but their effects on $CD_{0.5}$ differ: q-sampling decreases diversity while SC tends to increase it. Both approaches reduce FD-seq, though the improvement from SC is substantial while q-sampling yields only a marginal gain. Figure 12 further illustrates how quality metrics vary with t_{act}, showing a pattern consistent with text generation: activating q-sampling earlier simultaneously reduces pseudoperplexity and diversity, with the optimal FD-seq achieved at an intermediate value of t_{act}.

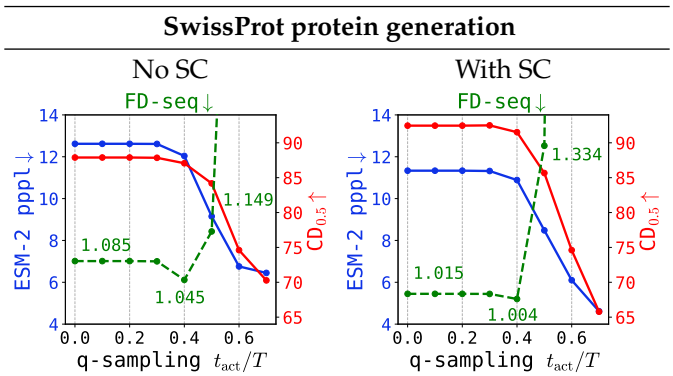

Figure 12: Quality metrics with and without self-conditoning (SC) for different q-sampling activation timestep on SwissProt protein generation. We use the baseline SC model with $p = 0.5$.

M Additional results for conditional generation

This appendix presents additional results for sequence-to-sequence tasks. Depending on the dataset, we measure BLEU (Papineni et al., 2002), ROUGE-1/2/L (Lin, 2004), BertScore (Zhang et al., 2020), or CodeBertScore (Zhou et al., 2023). Tables 9 and 10 report the performance of SC and q-sampling with optimal t_{act} across all four datasets, while Figure 13 shows BLEU as a function of t_{act} for IWSLT14 (machine translation) and QQP (question paraphrasing). For IWSLT14, both results are negative: SC degrades performance, and pure DDPM outperforms q-sampling at any activation timestep. For QQP, SC yields a statistically significant but modest improvement, while BLEU remains nearly constant across t_{act}. These results point to a qualitative difference from XSum and CoNaLa.

We hypothesize that the key factor is the strength of the conditioning signal. In some equence-to-sequence tasks, the input makes the output nearly deterministic: in machine translation, for instance, valid outputs differ only in word order or synonym choice. In others, such as code generation or summarization, the output space is much larger. For example, in code generation, the same functionality can be implemented with different algorithms, reordered code blocks, or renamed variables. In the strong-conditioning regime, the input already guides sampling away from low-density regions, preventing the failure mode that is addressed by SC and q-sampling. Moreover, both techniques deviate the generation trajectory from the forward process, a deviation that proves beneficial in unconditional generation but harmful under strong conditioning. This is particularly true for q-sampling, which does not replicate the forward process by design, as discussed in Section 3.3. In practice, determining whether the conditioning signal is strong enough may not always be straightforward. When strong conditioning is suspected, it may be preferable to first evaluate the model without SC and with pure DDPM, since applying SC without necessity increases both training and inference time and may even degrade generation quality.

N Image diffusion example

In the main part of the paper, we identify a failure mode of Gaussian diffusion models that is specific to discrete data. To verify that this effect is indeed tied to discreteness, in this appendix we consider a continuous domain and study image generation on CIFAR-10 (Krizhevsky et al., 2009). In the continuous case, data lie on a manifold, and the corresponding density transitions smoothly to a standard Gaussian, so the same failure mode is not expected to arise.

Experimental setup For this experiment, we train a Song U-Net (Song et al., 2020; Karras et al., 2022) with approximately 62M parameters on CIFAR-10 using the variance-preserving

Solver	IWSLT14 (SeqDiffuSeq) BLEU ↑	R-1/2/L ↑	QQP (SeqDiffuSeq) BS ↑	BLEU ↑	R-1/2/L ↑	XSum (TEncDM Emb) BS ↑	R-1/2/L ↑
DDPM	**29.19**	**57.4/32.3/54.3**	**82.8**	33.58	**60.2/35.7/57.8**	59.3	27.2/6.7/20.7
+ $SC_{0.5}$	28.93	56.8/31.8/53.7	**82.8**	**34.13**	60.1/35.6/57.5	66.3	30.5/9.1/23.4
+ $QS_{1.0}$	28.60	56.4/31.7/53.5	82.1	33.50	59.6/35.1/57.0	63.4	30.1/8.4/23.1
+ $QS_{1.0}$ + $SC_{0.5}$	28.79	56.8/31.9/53.7	82.7	**34.11**	59.8/35.4/57.3	**67.5**	**31.9/9.9/24.5**

Table 9: Impact of q-sampling (QS) and self-conditioning (SC) on IWSLT14, QQP, and XSum with their corresponding models. SC trained with probability p is denoted SC_p; QS with optimal activation timestep t_{act} is denoted $QS_{t_{\text{act}}/T}$. For IWSLT-14 and QQP, q-sampling does not give a significant improvement for any t_{act}, thus we report results for pure q-sampling $t_{\text{act}}/T = 1.0$, which is also optimal for XSum.

CoNaLa	CodeBS ↑
GENIE	66.35
+ $SC_{0.5}$	67.34
+ $QS_{0.5}$	67.70
+ $QS_{0.5}$ + $SC_{0.5}$	**68.10**

Table 10: Impact of q-sampling (QS) and self-conditioning (SC) on CoNaLa. SC trained with probability p is denoted SC_p; QS with optimal activation timestep t_{act} is denoted $QS_{t_{\text{act}}/T}$.

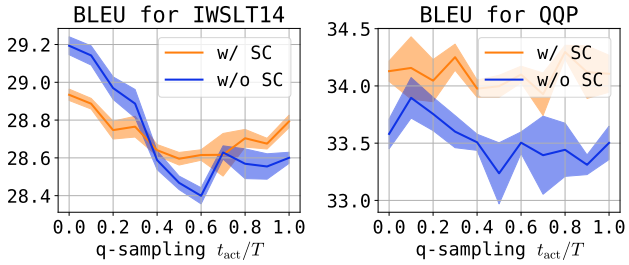

Figure 13: IWSLT-14 (left) and QQP (right) quality for varying q-sampling t_{act} with and without SC.

scheme and standard DDPM generation with 800 sampling steps. To assess the generated distribution, we report FID (Heusel et al., 2017) together with Precision, Recall, Density, and Coverage (Naeem et al., 2020), which allows us to evaluate both generation quality and sample diversity. Precision, Recall, Density, and Coverage (PRDC) are computed in a fixed feature space by first extracting embeddings for real and generated images and then defining a local neighborhood radius for each sample via its k-nearest-neighbor distance within the same set.

- **Precision** is the fraction of generated samples that lie inside at least one real-sample neighborhood.
- **Recall** is the fraction of real samples that lie inside at least one generated-sample neighborhood.
- **Density** quantifies how many generated samples populate each real neighborhood on average.
- **Coverage** is the fraction of real samples for which at least one generated neighbor falls within the corresponding real neighborhood radius.

Overall, FID and Precision primarily capture sample fidelity, while Recall and Coverage primarily capture diversity and support coverage. This choice is consistent with the methodology of the paper, where the effect of different heuristics is interpreted through the trade-off between quality and diversity rather than through a single metric alone.

Results We then evaluate the same heuristics as in the discrete setting: self-conditioning, q-sampling, and Minimum Bayes-Risk (MBR). In contrast to text and proteins, the overall picture on CIFAR-10 is qualitatively different: q-sampling degrades the quality of generated images and leads to visibly smoother samples. This observation is consistent with our analysis from the main text: q-sampling does not reproduce the trajectory of the reverse process, and it has smaller marginal variance that of DDPM (see Appendix. D). In the discrete setting, this can be beneficial because it helps trajectories avoid low-density regions within the critical interval; however, in a continuous image domain, where this failure mode is absent, the same contraction becomes harmful.

Table 11: Impact of q-sampling (QS), self-conditioning, and MBR on image diffusion on CIFAR-10.

Method	FID ↓	Precision ↑	Recall ↑	Density ↑	Coverage ↑
SongUNet	8.47	0.811	0.673	1.127	0.956
+ $QS_{0.025}$	37.40	0.693	0.434	0.713	0.630
+ $QS_{0.05}$	56.72	0.661	0.251	0.581	0.455
+ $QS_{0.075}$	64.22	0.647	0.195	0.557	0.412
+ $QS_{0.1}$	70.07	0.630	0.172	0.529	0.392
+ SC	16.48	0.853	0.565	1.467	0.922
+ $MBR_{0.1}$	9.42	0.886	0.568	1.606	0.974
+ $MBR_{0.2}$	10.17	0.907	0.546	1.772	0.976
+ $MBR_{0.3}$	11.15	0.927	0.507	1.894	0.980

Figure 14: Qualitative comparison on CIFAR-10. Generation is initialized from the same seeds. The first column corresponds to MBR-selected samples, while the last column corresponds to q-sampling (QS). As can be seen, q-sampling produces significantly smoother images.

Self-conditioning can improve precision-oriented metrics by making the model's predictions more confident, but this comes at the cost of diversity, as reflected in lower recall and coverage. In other words, the generated distribution becomes narrower: samples are more concentrated around typical modes, while a smaller fraction of the data manifold is covered. We observe a similar effect for MBR. By selecting the most central sample among several candidates, MBR favors more typical generations and suppresses outliers. In our implementation, MBR uses the ℓ_2 distance in the InceptionV3 (Szegedy et al., 2016) latent space, i.e. the same representation used for FID evaluation.

Overall, these results support our main claim. The improvements brought by self-conditioning, q-sampling, and MBR in text diffusion are not generic properties of these heuristics. Rather, they arise because these methods mitigate the specific DDPM pitfalls caused by the multimodal structure of noisified discrete data. On CIFAR-10, where the density transitions smoothly and sampling trajectories are not trapped between discrete modes, the same heuristics do not provide the same benefit and mainly induce the usual quality-diversity trade-off.

O Image toy example

To provide additional evidence for our hypothesis that the heuristics considered in this work improve generation in discrete domains such as text, but do not provide the same benefit for images, we introduce an intermediate experimental domain. The goal is to construct a task in which the model still operates on continuous images, while the data distribution has an internal discrete structure. This allows us to isolate the role of discreteness without changing the diffusion parameterization or the form of the model output.

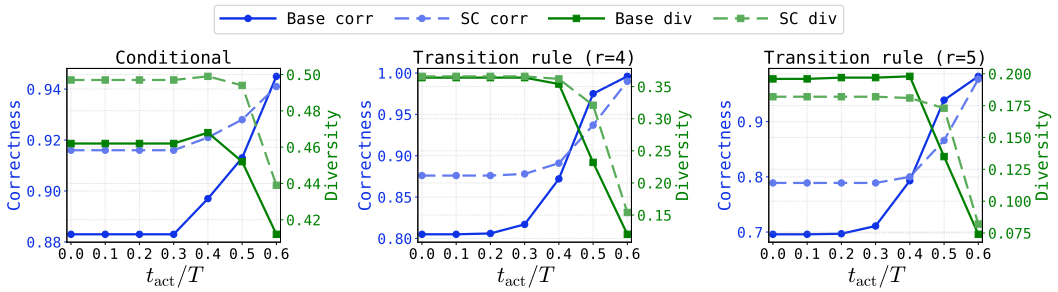

Figure 15: Correctness (left axis, blue) and diversity (right axis, green) as functions of the q-sampling activation time t_{act}, shown for the three patch-based CIFAR-10 experiments. Solid lines correspond to the base model, dashed lines to self-conditioning (SC).

Experimental setup To this end, we select a small subset S of CIFAR-10 images. Each 32×32 image is split into four 16×16 patches, which are stored in a patch bank. New images are then formed by composing patches from this bank. In this way, the model still predicts images in the continuous pixel space, but the set of valid outputs is determined by a finite collection of patch combinations. Thus, the model must memorize the atomic patches themselves while also generalizing over the discrete structure induced by their composition.

We consider two variants of this task. In the first one, we study a conditional setting. We train a conditional diffusion model in which the conditioning variable specifies an unordered set of patch identities contained in the image. Because the order of the patches is not provided, the same conditioning label may correspond to multiple valid outputs that differ by spatial permutation. This creates a multimodal conditional distribution while keeping the output space continuous. For the task we set $|S| = 2$, which yields a population of $8^4 = 4096$ possible objects.

In the second variant, we introduce transition rules between patches. The first patch, corresponding to the top-left position, is sampled uniformly from the full patch bank. Each subsequent patch is then sampled uniformly from a restricted subset determined by a transition graph over patch identities. In this case, the total number of valid images is $4|S| \cdot r^3$, where r denotes the number of allowed transitions per step. The purpose of this construction is to introduce a more complex discrete structure and prevent the model from simply memorizing the full set of training images. For this task we set $|S| = 10, r \in \{4, 5\}$ and for both tasks we limit train size to 1000 images.

Metrics For evaluation, we introduce two main metrics: correctness and diversity. Correctness is defined as the fraction of valid generated objects, where validity means consistency with the corresponding conditional label in the conditional setup, or with the transition rules in the transition-rule setup. Diversity is defined as the fraction of unique correct generated objects that do not belong to the training set, measured with respect to the full population of valid objects. In this way, correctness captures whether the model follows the discrete structure of the task, while diversity measures its ability to generate valid but non-memorized combinations.

To compute these metrics, each generated image is first split into four patches. For every patch, we then find the closest patch from the training bank in ℓ_2 distance and assign the corresponding patch index. In this way, each generated image is mapped to an ordered vector of four discrete patch identities, after which correctness and diversity can be computed directly in the discrete space.

Results We then analyze the effect of q-sampling and self-conditioning on generation. In both experimental setups, q-sampling with different values of t_{act} acts as a strong control parameter for the correctness-diversity trade-off, consistent with the results in the main part of the paper and with our theoretical interpretation. In particular, earlier activation of q-sampling leads to a substantial reduction in diversity. At the same time, the effect

Figure 16: Examples of patch-based training images.

becomes pronounced only for t_{act}/T around 0.3 and higher. This is explained by the fact that the discrete part of the distribution is already largely formed during the earlier stages of sampling, whereas the later steps mostly refine the continuous appearance of the patches. As a result, activating q-sampling too late has little influence on the discrete structure of generated objects, while activating it earlier causes the trajectory to avoid a substantial fraction of valid discrete modes, leading to lower diversity.

Self-conditioning shows a different behavior. It improves correctness without a comparable drop in diversity, which is consistent with our main observation that self-conditioning is particularly useful in discrete domains. In addition, when q-sampling is applied on top of a self-conditioned model, it again provides a mechanism for controlling the correctness-diversity trade-off. Thus, these experiments with synthetic discrete data embedded in a continuous space further support our interpretation of the problems arising in text diffusion and show that the proposed heuristics are indeed useful in discrete domains.