LEARNING TO RE-DRAFT: A VARIATIONAL STACKELBERG GAME FOR DISCRETE DIFFUSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Discrete diffusion models offer the ability to re-draft, revisiting and correcting earlier tokens throughout generation. This capability depends on the forward corruption process that defines what the denoiser learns to correct. Masked diffusion models fix tokens once they are unmasked, while uniform diffusion permits revisions but relies on uniformly random token substitutions. We instead learn which substitutions are most useful for training the denoiser to re-draft. We introduce Variational Stackelberg Discrete Diffusion (VSDD), a framework for learning a semantically aware corruption process. VSDD formulates training as a leader-follower game: the leader defines a Markovian corruption process parameterized by the denoiser's token embeddings, while the follower optimizes a variational denoising objective with the corruption process held fixed. The leader rewards corruptions based on how much the denoiser improves after learning from them, rather than on how easily the current denoiser can reconstruct them. We measure this improvement under a fixed reference corruption process, approximate the follower's response with a one-step gradient update, and optimize the leader using a score-function estimator. We evaluate VSDD across molecular, text, and playlist generation. VSDD substantially improves molecular validity over uniform and masked diffusion, reduces text perplexity relative to uniform diffusion while remaining competitive with masked diffusion, and achieves sizable improvements in offline playlist recommendation metrics.

1 INTRODUCTION

Discrete diffusion models (Austin et al., 2021; Hoogeboom et al., 2021) have emerged as a powerful paradigm for generating sequences of discrete tokens. These models learn to reverse a stochastic corruption process that gradually transforms a clean sequence into noise. Unlike autoregressive models, which generate tokens sequentially from left to right and can suffer from error accumulation (Bengio et al., 2015; Ranzato et al., 2015), diffusion models operate over the entire generated sequence and can iteratively revise earlier predictions during denoising. This ability to *re-draft* is particularly valuable for sequences with strong structural dependencies, such as molecular sequences.

The performance of discrete diffusion models depends on two closely coupled components: the *denoiser*, which predicts the clean sequence from a noisy observation, and the *forward noise process*, which determines how clean tokens are perturbed during training. While significant research effort has focused on denoiser architectures (Sahoo et al., 2024; Lou et al., 2024; Shi et al., 2024), the forward process is typically fixed to a simple predefined distribution, such as uniform or absorbing noise (Austin et al., 2021). Yet the choice of the forward corruption process can substantially affect learning efficiency and generation quality.

For instance, Sahoo et al. (2024) showed that absorbing noise can substantially outperform the uniform diffusion (Austin et al., 2021) for text generation, suggesting that *how* tokens are corrupted can be as important as *how* they are reconstructed. Similarly, Nichol and Dhariwal (2021) showed that a non-linear noise schedule can further improve generation quality. These observations motivate a natural question: can we *learn* a forward noise process that adapts to the data distribution and the denoiser's capabilities?

Learning the forward corruption process, however, introduces a fundamental optimization challenge. Recent work takes a step in this direction by jointly optimizing the forward process and the denoiser

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

through a score function based surrogate loss (Bartosh et al. 2026). With such joint optimization, the forward process may improve the training objective by compensating an underperforming denoiser rather than providing corruptions that help the denoiser improve. In particular, it may favor trivial corruptions that are easy to reconstruct, reducing the KL-loss without improving the resulting generative model. The challenge, however, is to learn corruptions based on how much the denoiser benefits from training on them, rather than how well the current denoiser can reconstruct them.

To address this challenge, we introduce **Variational Stackelberg Discrete Diffusion (VSDD)**, which formulates training as a *variational Stackelberg game*. Instead of jointly optimizing the forward process and the denoiser with a shared objective, the *leader* selects the corruption process while anticipating the follower's response, and the *follower* optimizes the variational objective of the denoiser with the corruption process fixed. This separation discourages the forward process from reducing the training loss by adapting to the current denoiser. Instead, the leader is rewarded for selecting corruptions that improve the denoiser after the follower has learned from them.

The posterior of the learned forward process affects how the model resamples, or re-drafts, tokens at inference time. Effective re-drafting therefore requires capturing relationships between tokens. Uniform noise treats all substitutions equally, regardless of the structure learned by the denoiser. In contrast, re-drafting an almost complete sequence may require distinguishing between plausible, closely related alternatives. We therefore parameterize the forward process using token embeddings learned by the denoiser. These embeddings capture relationships between tokens, while the leader learns how those relationships should shape the noise process. Related tokens can thus receive higher or lower transition probabilities depending on which substitutions provide useful training signals for the denoiser. We refer to this as a semantically aware noise process.

To optimize the forward noise process, the leader must anticipate the follower's response. Computing the best response would require fully optimizing the denoiser for each candidate noise process, which is prohibitively expensive. We therefore approximate it with a *better response* using a single virtual gradient update on samples from the proposed noise process. We measure how much this update improves the denoiser under a fixed reference noise process and use this improvement as the leader's reward. The leader then optimizes the noise process using a score function estimator.

We evaluate our approach across three structurally different domains: molecular, text, and playlist generation. Across these domains, VSDD improves the primary generation quality metrics over a number of diffusion baselines with and without learnable noise.

Our main contributions are:

1. We formulate learning the forward noise process and the denoiser as a variational Stackelberg game, where corruptions are optimized based on how much the denoiser improves after learning from them rather than how well the current denoiser can reconstruct them.
2. We introduce a semantically aware noise process parameterized by token embeddings learned by the denoiser, allowing the forward process to adapt to the learned structure of the domain.
3. We design an efficient training algorithm that approximates the follower's best response with a single virtual gradient step and optimizes the noise process via a score function estimator.
4. We show improved generation and re-drafting across molecular, text and playlist domains.

2 Related Work

Discrete diffusion models. Sohl-Dickstein et al. (2015) introduced diffusion based generative models for continuous data, later developed through DDPM (Ho et al., 2020) and score based generative modeling (Song et al., 2020). Austin et al. (2021) extended diffusion to discrete state spaces with D3PM, where the forward process is defined through categorical transition matrices, including uniform and absorbing corruption. Hoogeboom et al. (2021) proposed multinomial diffusion with argmax flows. More recent work focused on masked diffusion (Sahoo et al., 2024; Shi et al., 2024) and score entropy based methods (Lou et al., 2024). In particular, Sahoo et al. (2024) showed that masked diffusion with absorbing noise can substantially outperform uniform corruption for text generation. These results demonstrate that the choice of corruption process is consequential, but existing approaches typically prescribe this process in advance. Our work instead learns the structure of the corruption process according to how useful different corruptions are for training the denoiser.

Learned noise schedules and forward processes. In continuous diffusion, several works learn the noise schedule. Nichol and Dhariwal (2021) learn a variance schedule through the variational bound, while Kingma et al. (2021) parameterize the signal to noise ratio via a neural network. Dieleman et al. (2022) show that the forward process in continuous diffusion is equivalent up to reparameterization, implying that learning the schedule affects the weighting of the training objective rather than the underlying generative model. In discrete diffusion, however, different noise processes change what the denoiser is trained to reconstruct. Learning this forward process in discrete diffusion is thus a modeling decision rather than only a training convenience.

For discrete diffusion, Bartosh et al. (2026) recently proposed learning the forward process by jointly optimizing the noise model and the denoiser via a score function based surrogate objective. This approach presents two challenges: the score function gradient can have high variance, and joint optimization may allow the forward process to reduce the training objective by adapting to the current denoiser rather than by providing useful training corruptions. Our work differs in how the forward process is optimized: VSDD evaluates corruptions based on how much the denoiser improves after learning from them, rather than how well they suit the current denoiser. To reduce the variance of the score function estimator, we evaluate multiple corruption samples and use a leave one out baseline.

Game-theoretic concepts in machine learning. Stackelberg games model sequential interactions where a leader chooses a strategy while anticipating a follower's response (Conitzer and Sandholm, 2006). Stackelberg formulations have been studied in machine learning, including analyses of GANs (Goodfellow et al., 2014; Fiez et al., 2020). Our approach is related to MAML style meta learning (Finn et al., 2017), where outer optimization evaluates its objective after inner loop adaptation. Similarly, in VSDD the leader evaluates a corruption process via the denoiser's response after adapting to corruptions sampled from that process. Unlike these settings, the leader in VSDD parameterizes the forward process itself, learning which corruptions provide useful training signals for the denoiser.

3 Notation and Preliminaries

We let V be the *vocabulary size*, and let $\bar{\alpha}_t \in [0, 1]$ be the cumulative *retention schedule*. Following standard notation in diffusion modeling (Austin et al., 2021), we let $\alpha_t = \bar{\alpha}_t / \bar{\alpha}_{t-1}$ be the *per-step retention rates* and $\beta_t = 1 - \alpha_t$ be the respective *noise rates*. Let $x_t = (x_t^1, \ldots, x_t^L)$ be the sequence of L tokens at time t, where each $x_t^\ell \in \{1, \ldots, V\}$ is represented as a one-hot vector. One token is a `PAD` token that is never corrupted. We let x_0 be the *clean* (uncorrupted) sequence.

In standard discrete diffusion, the forward noise process is fixed in advance: one specifies the retention schedule $\bar{\alpha}_t$, constructs the per-step token transition matrices $Q_t \in \mathbf{R}^{V \times V}$, and only trains the *denoiser* $p_\theta(x_0|x_t, t)$ against this fixed noise process Q_t (Austin et al., 2021). To this end, we let $\bar{Q}_t = \prod_{\tau=1}^{t} Q_t$ be the *cumulative transition matrix* after t steps. The forward marginal probability distribution used to sample a noisy sequence x_t is:

$$q(x_t \mid x_0) = \mathrm{Cat}\big(x_t |\, \bar{Q}_t^\top\, x_0\big). \tag{1}$$

In this setting, the denoiser $p_\theta(x_0|x_t, t)$ is parameterized by a transformer with learnable parameters θ. At inference, it takes a corrupted sequence x_t, the timestep t and predicts the clean sequence $\hat{x}_0$.

We let $E_\theta \in \mathbb{R}^{V \times d}$ denote the *token embedding matrix* learned as part of the denoiser $p_\theta(x_0|x_t, t)$, with one d-dimensional embedding for each of the V vocabulary tokens.[1] Finally, we use the standard ELBO loss $\mathcal{L}$ that includes the per-step KL terms $\mathcal{L}_{\mathrm{KL}}(t)$ (for $t > 1$) and the auxiliary reconstruction cross-entropy $\mathcal{L}_{\mathrm{CE}}$, over non-`PAD` positions:

$$\mathcal{L} = \sum_{t=2}^{T} \underbrace{D_{\mathrm{KL}}\big(q(x_{t-1} \mid x_t, x_0) \,\|\, p_\theta(x_{t-1} \mid x_t)\big)}_{\mathcal{L}_{\mathrm{KL}}(t)} \underbrace{-\log p_\theta(x_0 \mid x_1)}_{\mathcal{L}_{\mathrm{CE}}}. \tag{2}$$

Here, $q(x_{t-1}|x_t, x_0)$ is the true reverse posterior induced by the forward process, and $p_\theta(x_{t-1}|x_t)$ is the corresponding learned posterior. In Section 5, we show how to compute these posteriors.

[1] Typically, the embedding layer is the first layer of the transformer used as denoiser.

4 SEMANTIC-AWARE NOISE PROCESS

As motivated in Section 1, effective re-drafting may need to distinguish between plausible, closely related tokens rather than treating all substitutions equally. We therefore construct a semantically aware noise whose transitions depend on the token representations learned by the denoiser. It provides the structure between tokens that we leverage to learn the forward process in Section 5.

To parameterize the forward noise process by the token representations we let

$$Q_{t,\phi} = \alpha_t \, I + (1 - \alpha_t) M_{t,\phi}, \quad t = 1, ..., T, \tag{3}$$

where I is the identity matrix of rank V, and $M_{t,\phi}$ is the learnable transition matrix parameterized by ϕ. Thus, at each step t, a token is retained with probability α_t, while with probability $1 - \alpha_t$ it is replaced according to $M_{t,\phi}$.

To construct $M_{t,\phi}$, we define $s_{t,\phi}(i, j) = e_i \, A_{t,\phi} \, e_j^T$ as a learned similarity score between tokens i and j. Here, $e_i, e_j \in \mathbf{R}^d$ are $L2$-normalized ith and jth rows from E_θ (Section 3), and $A_{t,\phi} \in \mathbb{R}^{d \times d}$ learns how relationships in the denoiser's embedding space should influence corruption at time t. In other words, the learned embeddings e_i, e_j provide the underlying token structure, while $A_{t,\phi}$ learns which aspects of this structure should determine likely substitutions. For all non-pad tokens we define

$$M_{t,\phi}[i, j] = \mathbf{1}_{\{i \neq j\}} \frac{\exp s_{t,\phi}(i, j)}{\sum_{\ell \neq i} \exp s_{t,\phi}(i, \ell)}, \tag{4}$$

which ensures that $M_{t,\phi}$ and $Q_{t,\phi}$ are row stochastic, meaning that every row defines a valid probability distribution over the next token. We let $\bar{Q}_{t,\phi} = \prod_{\tau=1}^{t} Q_{t,\phi}$ denote the corresponding cumulative transition matrix. Consequently, we let the forward marginal be $q_\phi(x_t|x_0) = \mathrm{Cat}\big(x_t; \bar{Q}_{t,\phi}^T x_0\big)$ and its corresponding reverse posterior be $q_\phi(x_{t-1}|x_t, x_0)$.

A direct approach would be to add ϕ to the diffusion loss (Equation (2)) and to jointly optimize the forward process and the denoiser. However, this can lead to learning a degenerate noise process and a poor denoiser. To see this, observe that the KL term in Equation (2) can decrease in two ways: (1) by improving the denoiser p_θ to better match the true posterior, or (2) by simplifying q_ϕ so that the posterior becomes trivial to match. Joint optimization permits the latter path: the forward process can adapt its off-diagonal transitions to the current denoiser, increasing the probabilities of substitutions that are already easy for p_θ, simplifying the posterior target. The denoising loss may therefore decrease without an improvement in generation quality. In Section 5, we address this by optimizing the noise process based on how the denoiser improves after learning from its corruptions.

5 VARIATIONAL STACKELBERG GAME

To prevent the degenerate solutions discussed above, we formulate the optimization problem from a game theoretic perspective. In particular, we consider two players, a *leader* and a *follower*, interacting in a Stackelberg game fashion (Conitzer and Sandholm, 2006). The leader chooses an action ϕ that parameterizes the forward corruption process $Q_{t,\phi}$, while the follower observes this process and chooses an action θ corresponding to the parameters of the denoiser $p_\theta(x_0|x_t, t)$. Thus, the leader determines how training sequences are corrupted, while the follower learns to denoise sequences generated by that process. We now elaborate on the objectives of the follower and the leader and show how their interaction is used to jointly learn ϕ and θ.

Follower. The follower optimizes the standard diffusion loss from Equation (2) under the noise process selected by the leader. Importantly, as is common in Stackelberg games we assume that the follower cannot directly affect the action of the leader (Conitzer and Sandholm, 2006). Therefore, when choosing its action θ the follower samples noisy sequences x_t from $q_{\mathrm{sg},\phi}(x_t|x_0) = \mathrm{Cat}(x_t|\,\mathrm{sg}\{\bar{Q}_{t,\phi}\}^T x_0)$, where $\mathrm{sg}\{.\}$ is the stop-gradient operator. This allows the follower to train on the corruption process selected by the leader while preventing gradients from the follower's objective from propagating to the leader parameters ϕ. Consequently, the follower's loss can be obtained from Equation (2) as follows:

$$\mathcal{L}^F(\theta|\phi) = (T - 1) D_{KL}\big(q_{\mathrm{sg},\phi}(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}|x_t)\big) - \log p_\theta(x_0|x_1), \tag{5}$$

Here, $q_{\mathrm{sg},\phi}(x_{t-1}|x_t, x_0)$ is the reverse posterior corresponding to $q_{\mathrm{sg},\phi}(x_t|x_0)$, and $p_\theta(x_{t-1}|x_t)$ is the denoiser's posterior, computed as

$$p_\theta(x_{t-1}|x_t) = \sum_{\hat{x}_0} p_\theta(\hat{x}_0|x_t, t) q_{\mathrm{sg},\phi}(x_{t-1}|x_t, \hat{x}_0), \tag{6}$$

Following Austin et al. (2021), this combines the denoiser's prediction over the clean sequence $\hat{x}_0$ with the reverse posterior of the fixed corruption process to obtain the distribution over the previous diffusion state x_{t-1}. Thus, the follower optimizes the standard discrete diffusion loss with the corruption process $Q_{t,\phi}$ held fixed.

Leader. The leader's goal is to find a corruption process $Q_{t,\phi}$ such that, after the follower optimizes the denoiser against this process (via Equation (5)), the quality of the generated sequences would be maximal. To this end, the leader needs to: (1) assess whether a candidate strategy ϕ improves the log-likelihood of cleaned data $\log p(\hat{x}_0)$, and (2) determine how to update ϕ while accounting for the follower's subsequent response θ.

To address the first problem, we would ideally evaluate the resulting denoiser through the log likelihood of clean data, but optimizing this quantity directly is intractable. We therefore use an ELBO objective as a tractable surrogate. To ensure that different leader strategies are evaluated against the same corruption distribution, we introduce a fixed reference noise process Q_{ref}, and we let $q_{\mathrm{ref}}(x_t|x_0)$ be its corresponding forward marginal distribution.[2] Using a fixed reference process provides a common evaluation distribution for comparing different leader strategies.

While the reference process determines the noise distribution used for evaluation, the current leader strategy ϕ determines the reverse posterior $q_\phi(x_{t-1}|x_t, x_0)$ used to construct the reverse transition probabilities. As in the follower's case, we assume that the leader cannot affect the follower's action θ directly, and therefore, we freeze the gradient flow via the follower's embeddings, i.e., $\mathrm{sg}\{E_\theta\}$ when computing $\mathrm{sg}_\theta\{q_\phi\}$. The leader's reference ELBO objective results in

$$\mathcal{L}^L_{\mathrm{ELBO}}(\phi|\theta) = (T-1) D_{KL}\big(q_{\mathrm{ref}}(x_{t-1}|x_t, x_0) || p_{\theta,\phi}(x_{t-1}|x_t)\big) - \log p_\theta(x_0|x_1), \tag{7}$$

where

$$p_{\theta,\phi}(x_{t-1}|x_t) = \sum_{\hat{x}_0} p_\theta(\hat{x}_0|x_t)\, \mathrm{sg}_\theta\{q_\phi(x_{t-1}|x_t, \hat{x}_0)\}. \tag{8}$$

The fixed reference process is important: the leader is evaluated against the same reference distribution as ϕ changes, rather than against an evaluation distribution that changes with its own strategy.

In a Stackelberg game, the leader anticipates the follower's best response to its strategy (Conitzer and Sandholm, 2006). In our setting, computing the exact best response would require fully optimizing the denoiser for each candidate corruption process, which is prohibitively expensive. We therefore use a single gradient update as an approximate *better response*. To this end, we sample a validation minibatch x_0 and inject K different noise realizations into it to create K new minibatches $x_t^{(k)}$, $k = 1, \ldots, K$. We let $d\theta_k = -\gamma \nabla_\theta \mathcal{L}^F(\theta; x_t^{(k)})$ denote the resulting virtual follower update for the kth validation minibatch, where γ is a step size hyperparameter. Thus, $\theta + d\theta_k$ approximates how the follower would change after learning from that particular corruption realization.

We then compute the *reward* for that noise realization as the relative improvement in the leader's reference loss:

$$R_k = \frac{\mathcal{L}^L_{\mathrm{ELBO}}(\phi|\theta) - \mathcal{L}^L_{\mathrm{ELBO}}(\phi|\theta + d\theta_k)}{\gamma |\mathcal{L}^L_{\mathrm{ELBO}}(\phi|\theta)|}. \tag{9}$$

A positive reward therefore indicates that training on the sampled noise improves the follower under the fixed reference objective. This evaluates a corruption by the improvement it induces in the denoiser, rather than by how easily the current denoiser can reconstruct it.

These rewards allow the leader to increase the probability of noise realizations that improve the follower and decrease the probability of those that do not, using a score function estimator (Bartosh

[2] For example, we can use a uniform Categorical distribution as in standard uniform D3PM, Equation (1).

Algorithm 1 Variational Stackelberg Training for Discrete Diffusion

Require: Reference process q_{ref}, block size N, number of probes K; learning rates η_ϕ, η_θ; γ, λ_T

1: Initialize denoiser $p_\theta(x_0|x_t, t)$ and noise kernel Q_ϕ
2: **for** epoch $= 1, \dots M$ **do**
3: block_steps $\leftarrow 0$
4: **for** each minibatch x_0 **do**
5: **if** block_steps $= 0$ **then**
6: Build $Q_\phi, \bar{Q}_\phi$
7: **Follower step:** $\theta \leftarrow \theta - \eta_\theta \nabla_\theta \mathcal{L}^F(\theta; \text{sg}\{Q_\phi\})$ {Eq. (5)}
8: block_steps $\leftarrow$ block_steps $+ 1$
9: **if** block_steps $= N$ **then**
10: Sample a validation batch x_0^{val}; compute $\mathcal{L}^L_{\text{ELBO}}(\phi|\theta)$ via Eq. (7)
11: **for** $k = 1, \dots, K$ **do**
12: $x_t^{(k)} \leftarrow$ Corrupt x_0^{val} under Q_ϕ
13: Follower's better response: $\theta_{br} \leftarrow \theta - \gamma \nabla_\theta \mathcal{L}^F(\theta; x_t^{(k)})$
14: Reward: $R_k \leftarrow \frac{\mathcal{L}^L_{\text{ELBO}}(\phi|\theta) - \mathcal{L}^L_{\text{ELBO}}(\phi|\theta_{br})}{\gamma d |\mathcal{L}^L_{\text{ELBO}}(\phi|\theta)|}$
15: Compute normalized rewards $\tilde{R}_k$ via Eq. 10
16: $\mathcal{L}^L(\phi) \leftarrow \mathcal{L}^L(\phi) + \lambda_T D_{KL}[\bar{Q}_{T,\phi} \| \mathcal{U}]$ {Eq. 11 + terminal regularization}
17: **Leader step:** $\phi \leftarrow \phi - \eta_\phi \nabla_\phi \mathcal{L}^L(\phi)$ {Eq. (11)}
18: block_steps $\leftarrow 0$

et al., 2026). Because such estimators may have high variance, we normalize and clip the rewards:

$$\tilde{R}_k = \min\Big\{ \max\big\{ \frac{R_k - B_k}{\sqrt{v + \epsilon}}, -c \big\}, c \Big\}, \quad \text{where } B_k = \frac{1}{K-1} \sum_{j \neq k} R_j, \tag{10}$$

where B_k is a leave one out baseline, v is the running variance of the rewards, and c is the clipping threshold. The final leader's objective is

$$\mathcal{L}^L(\phi) = -\frac{1}{K} \sum_{k=1}^{K} \text{sg}\{\widetilde{R}_k\} \log q_\phi\Big(x_t^{(k)} \mid x_0, t\Big). \tag{11}$$

Minimizing this objective increases the probability of noise realizations with positive relative reward and decreases the probability of those with negative relative reward.

Stackelberg Game Dynamics. As mentioned above, computing the exact best responses is prohibitively expensive. We therefore organize training into blocks of N steps. At the start of each block, the leader uses a one-step virtual update to estimate the follower's expected better response and commits to a forward noise process $Q_{t,\phi}$. The follower then performs N gradient updates while the noise process remains unchanged.

The full algorithm is summarized in Algorithm 1. Here, we let η_θ and η_ϕ denote the learning rates of the follower and the leader, respectively. At the beginning of each training block, we fix the leader's forward corruption process Q_ϕ and the cumulative transition matrix $\bar{Q}_\phi$, which remain unchanged during the N follower updates (lines 5-6). For each minibatch within the block, the follower performs a gradient update of θ by optimizing $\mathcal{L}^F$ via Equation (5) (line 7). At the end of the block, the leader samples a validation minibatch x_0^{val} and computes its reference loss using Equation (7) (line 10). The leader then creates K corrupted versions $x_t^{(k)}$ of the same validation minibatch, computes a one-step better response for each corruption realization, and evaluates the corresponding reward (lines 11-14). The rewards are then normalized using Equation 10 (line 15). In line 16, we compute the leader's loss via Equation (11) and add a terminal KL-regularization that encourages $\bar{Q}_{T,\phi}$ to match the prior (here, λ_T is a regularization term). Finally (line 17), the leader updates its action ϕ by minimizing its loss as in Equation (11). Importantly, this update is based on how much the sampled corruptions are predicted to improve the denoising process, rather than on how well they suit the current denoiser.

6 EVALUATION

We evaluate our approach on three domains that differ substantially in vocabulary size, structural constraints, and the nature of sequential dependencies: molecular, text, and playlist generation. We compare the following five forward process designs, all sharing the same denoiser architecture and comparable training settings. More details on the datasets and the setups are in Appendix B.

1. **MDLM** (Sahoo et al., 2024). The standard absorbing forward process where tokens are independently replaced with a designated `[MASK]` token.
2. **D3PM-Uniform** (Austin et al., 2021). A uniform noise process in which corrupted tokens are replaced by tokens sampled uniformly at random. The training objective combines a KL-term with a cross-entropy reconstruction loss following the standard discrete diffusion loss.
3. **Forward-Learned Discrete Diffusion (FLDD)** (Bartosh et al., 2026). A non-Markov forward noise process that uses a REINFORCE surrogate objective to learn the forward process.
4. **D3PM-Reinforce**. A learnable Markov forward process with the same parameterization as VSDD, in which ϕ and θ are jointly optimized using the score function surrogate objective of Bartosh et al. (2026).
5. **Variational Stackelberg Discrete Diffusion (VSDD)**. Our model, which uses the same learnable Markov forward process as D3PM-Reinforce but optimizes it through the Stackelberg dynamics (Algorithm 1). The leader is updated every $N = 100$ follower minibatch steps.

6.1 RESULTS

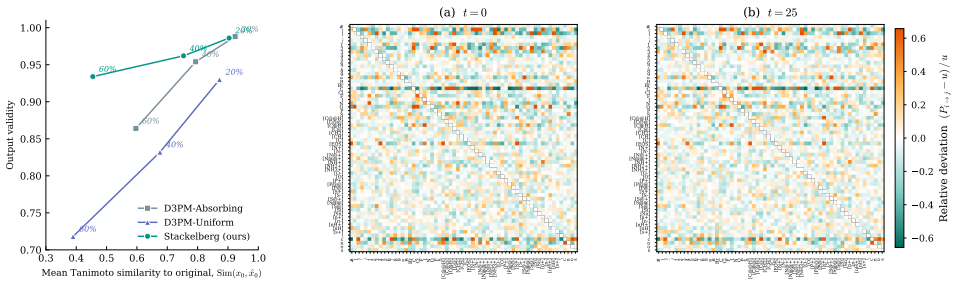
Molecular generation. We use molecular data from a publicly available large-scale ZINC database (Irwin et al., 2020), represented as SMILES strings (Weininger, 2002). This domain imposes strict structural constraints: small syntactic errors can render an entire generated sequence invalid, making it particularly suitable for evaluating the re-drafting capabilities of our model. The small vocabulary of 64 tokens allows us to visually inspect the learned noise processes. We evaluate the chemical validity of the generated molecular sequences, i.e., the fraction of generated SMILES strings that parse into valid molecules under RDKit (Landrum et al., 2026), as well as their uniqueness, novelty, and diversity. All evaluations are computed over 15K generated samples per run.

Table 1 shows that VSDD achieves a validity rate of 84.3%, substantially outperforming both the absorbing MDLM baseline (58.7%) and D3PM-Uniform (48.1%). We further inspect the learned off-diagonal transitions and observe several chemically interpretable patterns (Figure 1b). At $t = 25$, substitutions among halogens, such as `F`, `Cl`, `I`, are more likely than under uniform corruption. These halogens commonly attach to the rest of a molecule through a single chemical bond, so exchanging them can preserve the local bonding pattern. The learned corruption also favors substitutions between the aromatic atom tokens `c` and `n`. These patterns suggest that the learned semantically aware noise process captures meaningful chemical relationships between tokens.

We further examine the relationship between the learned noise process and the denoiser's token embeddings. The global Spearman rank correlation between token-embedding cosine similarity and transition probability is $\rho = 0.70$ (Figure 2a). Thus, tokens that are representationally similar under the denoiser's learned embeddings E_θ tend to be more likely to be substituted for one another. This correlation is absent by construction in D3PM-uniform and MDLM baselines.

Finally, to inspect the re-drafting abilities of VSDD, we partially corrupt clean molecules by running the forward process for 20%, 40% and 60% of the diffusion steps, then denoise them to reconstruct the originals. Since different noise processes reach different corruption levels after the same number of steps, we plot the reconstruction validity against the mean Tanimoto similarity (Bajusz et al., 2015) between the corrupted molecules and clean ones. This provides a model-agnostic measure of the trade-off between the injected noise and reconstruction quality. Figure 1a shows that VSDD achieves higher reconstruction validity than the baselines across all perturbation levels.

Text Generation. We evaluate VSDD on the TinyStories dataset (Eldan and Li, 2023), which allows us to study the learned forward process on natural language sequences with less rigid syntax and longer range semantic dependencies. We preprocess data using a BPE tokenizer with a vocabulary of 2048 tokens and sequence length of 256 tokens. For computational efficiency, we train all



(a) Validity/similarity after partial re-drafting.

(b) Learned transition matrices for the molecular domain. The color map is relative to the uniform noise.

Figure 1: Re-drafting and learned corruption in molecular generation.

Table 1: Chemical validity, uniqueness, novelty and diversity of generated molecules.

Method	**Validity (%)** ↑	**Uniqueness (%)** ↑	**Novelty (%)** ↑	**Diversity** ↑
MDLM	58.7	100	100	0.8717
D3PM-Uniform	48.1	100	100	0.8674
D3PM-Reinforce	62.0	100	100	0.8727
FLDD	3.0	100	100	0.9761
VSDD (ours)	**84.3**	100	100	0.8679

models on a random subset of 100K samples from the dataset. We evaluate the generated stories using GPT-2 perplexity (PPL) and Distinct-N language diversity metric (Li et al., 2016).

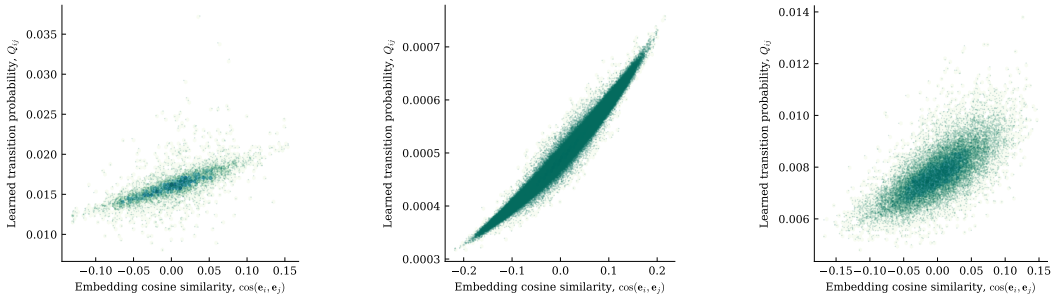
VSDD more than halves the GPT-2 perplexity of D3PM-Uniform, reducing it from 236.33 to 106.71 (Table 2a). This approaches MDLM's perplexity of 103.26 while achieving slightly higher Distinct-1 and Distinct-2 scores. Thus, VSDD substantially narrows the perplexity gap between uniform and masked diffusion while retaining the ability to revise previously generated tokens. These results show that the flexibility to re-draft need not incur the large perplexity penalty observed with unstructured uniform corruption.

In contrast, the jointly learned D3PM-Reinforce and FLDD baselines have significantly higher perplexity. As discussed in Section 4, jointly optimizing the forward process and the denoiser can favor corruptions that are more predictable rather than informative. Minimizing the denoising loss $\mathcal{L}_{KL}(t)$ can therefore yield nearly deterministic transitions (for example, $\Pr(\text{“was”} \to \text{“and”}) = 0.98$, and $\Pr(\text{“.”} \to \text{“to”}) = 0.99$; see Appendix D). In contrast, VSDD accounts for the follower's adaptation when updating the leader and favors transitions that improve generation quality under the reference noise process. We further analyze VSDD's re-drafting capability in Appendix C.1.

Playlist generation. We now use a real world playlist recommendation dataset comprising approximately 44K listening histories. Each history is a variable length sequence of discrete semantic IDs of the tracks the user interacted with. Each semantic ID is represented by a triplet of tokens (see Appendix B.3). This domain allows us to evaluate our approach on sequences whose coherence is governed by behavioral and semantic relationships rather than an explicit formal grammar.

To evaluate our model, we hold out the final three tracks (9 tokens) of each validation playlist and perform conditional reverse diffusion. The prefix positions remain fixed to their ground-truth values throughout all reverse steps, and we only denoise the held-out suffix positions. A generated triplet is considered relevant if it exactly matches any of the three ground-truth held-out triplets. We report NDCG@3 and HitRate@3 on 500 playlists in Table 2b. With only three relevant tracks among approximately 228K unique triplets, exact match prediction is challenging: a uniformly random prediction would have a hit probability of approximately 1.3×10^{-5}. VSDD achieves NDCG@3 of 0.108 and HitRate@3 of 0.246, outperforming all four baselines on both metrics.

In Appendix C.2, we further study the re-drafting properties of VSDD by analyzing the model's ability to repair targeted structural corruptions in playlists.



(a) Molecular generation. (b) Text generation. (c) Playlist generation.

Figure 2: Correlation between the similarity of token embeddings and the transition probabilities.

Table 2: Text and playlist generation results.

Model	PPL ↓	Dist-1 ↑	Dist-2 ↑
MDLM	103.26	0.1535	0.5303
D3PM-Uniform	236.33	0.1362	0.5718
D3PM-Reinforce	351.83	0.2447	0.7022
FLDD	871.51	0.2441	0.7671
VSDD (ours)	106.71	0.1570	0.5426

(a) Perplexity and diversity of generated stories.

Model	NDCG@3 ↑	HitRate@3 ↑
MDLM	0.0882	0.2020
D3PM-Uniform	0.0575	0.1540
D3PM-Reinforce	0.0526	0.1340
FLDD	0.0418	0.1100
VSDD (ours)	**0.1077**	**0.2460**

(b) NDCG@3 and HitRate@3 of generated playlists.

6.2 Discussion

Our evaluation highlights that both the choice of the corruption process and its optimization matter. VSDD outperforms the jointly learned D3PM-Reinforce and FLDD baselines on the primary metrics across all three domains. The comparison with D3PM-Reinforce is particularly informative because it uses the same noise parameterization but optimizes the forward process differently. These results support our central premise that useful corruptions should be selected by the improvement they induce after denoiser adaptation, rather than on how easily the current denoiser reconstructs them.

The comparison with fixed corruption processes varies across domains. For molecular generation, VSDD substantially improves validity over both uniform and masked diffusion, while for playlist generation it improves both NDCG@3 and HitRate@3. For text generation, VSDD substantially reduces the perplexity of uniform diffusion and is comparable to masked diffusion.

The learned noise and re-drafting analyses provide complementary evidence beyond generation quality. In molecules, the learned noise maps reveal pronounced, token-dependent structure, including chemically interpretable substitutions. Furthermore, it achieves higher reconstruction validity across the evaluated partial-corruption levels. We observed similar results for the text and playlist generation tasks. Together, these results show that the benefits of the learned corruption process extend beyond generation to re-drafting perturbed sequences across all three domains.

7 Conclusion

We introduced Variational Stackelberg Discrete Diffusion (VSDD), a framework for learning a semantically aware forward noise process through the denoiser's response. The leader evaluates sampled corruptions by the improvement they induce after a virtual follower update, while the follower learns to reverse the selected noise process. Experiments on molecular, text, and playlist generation show improvements over fixed and learnable forward noise process baselines. Our reconstruction and conditional-infilling experiments further show that these benefits extend to re-drafting, with VSDD effectively repairing controlled perturbations across all three domains. Together, these findings suggest that effective discrete diffusion benefits from learning not only how to reverse corruption, but also which corruptions help the denoiser learn to revise.

AI USE DISCLOSURE

Generative AI tools were used to improve the readability and presentation of this paper, including correcting grammatical errors, rephrasing text for clarity, and assisting with manuscript formatting. They were also used to help identify potentially relevant related work. All AI-assisted edits and all suggested references were independently reviewed and verified by the authors.

REFERENCES

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. 2021. Structured Denoising Diffusion Models in Discrete State-Spaces. In *Advances in Neural Information Processing Systems*, Marc'Aurelio Ranzato, Alina Beygelzimer, Yann Dauphin, Percy S. Liang, and Jennifer Wortman Vaughan (Eds.), Vol. 34. Curran Associates, Inc., 17981–17993. https://proceedings.neurips.cc/paper_files/paper/2021/file/958c530554f78bcd8e97125b70e6973d-Paper.pdf

Dávid Bajusz, Anita Rácz, and Károly Héberger. 2015. Why is Tanimoto index an appropriate choice for fingerprint-based similarity calculations? *Journal of Cheminformatics* 7 (2015). https://api.semanticscholar.org/CorpusID:1221969

Grigory Bartosh, Teodora Pandeva, Sushrut Karmalkar, and Javier Zazo. 2026. Forward-Learned Discrete Diffusion: Learning How to Noise to Denoise Faster. In *International Conference on Learning Representations*. arXiv:2605.18204 [stat.ML] doi:10.48550/arXiv.2605.18204

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks. *CoRR* abs/1506.03099 (2015). arXiv:1506.03099 http://arxiv.org/abs/1506.03099

Vincent Conitzer and Tuomas Sandholm. 2006. Computing the optimal strategy to commit to. In *Proceedings of the 7th ACM Conference on Electronic Commerce* (Ann Arbor, Michigan, USA) *(EC '06)*. Association for Computing Machinery, New York, NY, USA, 82–90. doi:10.1145/1134707.1134717

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. 2022. Continuous diffusion for categorical data. arXiv:2211.15089 [cs.CL] https://arxiv.org/abs/2211.15089

Ronen Eldan and Yuanzhi Li. 2023. TinyStories: How Small Can Language Models Be and Still Speak Coherent English? arXiv:2305.07759 [cs.CL] https://arxiv.org/abs/2305.07759

Tanner Fiez, Benjamin Chasnov, and Lillian Ratliff. 2020. Implicit Learning Dynamics in Stackelberg Games: Equilibria Characterization, Convergence Analysis, and Empirical Study. In *Proceedings of the 37th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 119)*, Hal Daumé III and Aarti Singh (Eds.). PMLR, 3133–3144. https://proceedings.mlr.press/v119/fiez20a.html

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In *Proceedings of the 34th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 70)*, Doina Precup and Yee Whye Teh (Eds.). PMLR, 1126–1135. https://proceedings.mlr.press/v70/finn17a.html

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2* (Montreal, Canada) *(NIPS'14)*. MIT Press, Cambridge, MA, USA, 2672–2680.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. In *Proceedings of the 34th International Conference on Neural Information Processing Systems* (Vancouver, BC, Canada) *(NIPS '20)*. Curran Associates Inc., Red Hook, NY, USA, Article 574, 12 pages.

Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. 2021. Argmax flows and multinomial diffusion: learning categorical distributions. In *Proceedings of the 35th International Conference on Neural Information Processing Systems (NIPS '21)*. Curran Associates Inc., Red Hook, NY, USA, Article 953, 12 pages.

John J. Irwin, Khanh G. Tang, Jennifer Young, Chinzorig Dandarchuluun, Benjamin R. Wong, Munkhzul Khurelbaatar, Yurii S. Moroz, John W. Mayfield, and Roger A. Sayle. 2020. ZINC20 – A Free Ultra Large-Scale Chemical Database for Ligand Discovery. *Journal of chemical information and modeling* 60 (2020), 6065 – 6073. https://api.semanticscholar.org/CorpusID:226059428

Diederik P. Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. 2021. Variational diffusion models. In *Proceedings of the 35th International Conference on Neural Information Processing Systems (NIPS '21)*. Curran Associates Inc., Red Hook, NY, USA, Article 1660, 12 pages.

Greg Landrum, Paolo Tosco, Ricardo Rodriguez, Brian Kelley, David Cosgrove, Riccardo Vianello, sriniker, Peter Gedeck, Gareth Jones, Dan Nealschneider, Eisuke Kawashima, NadineSchneider, tadhurst cdd, Andrew Dalke, Niels Maeder, Matt Swain, Yakov Pechersky, Brian Cole, Kevin Boyd, Samo Turk, Aleksandr Savelev, Rachel Walker, Alain Vaucher, Maciej Wójcikowski, Hussein Faara, Ichiru Take, Vincent F. Scalfani, Steven Kearnes, Kazuya Ujihara, and Daniel Probst. 2026. *rdkit/rdkit: 2026_03_6 (Q1 2026) Release*. doi:10.5281/zenodo.22140358

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A Diversity-Promoting Objective Function for Neural Conversation Models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Kevin Knight, Ani Nenkova, and Owen Rambow (Eds.). Association for Computational Linguistics, San Diego, California, 110–119. doi:10.18653/v1/N16-1014

Aaron Lou, Chenlin Meng, and Stefano Ermon. 2024. Discrete diffusion modeling by estimating the ratios of the data distribution. In *Proceedings of the 41st International Conference on Machine Learning* (Vienna, Austria) *(ICML'24)*. JMLR.org, Article 1333, 30 pages.

Dmitrii Moor, Yi Yuan, Rishabh Mehrotra, Zhenwen Dai, and Mounia Lalmas. 2023. Exploiting Sequential Music Preferences via Optimisation-Based Sequencing. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management* (Birmingham, United Kingdom) *(CIKM '23)*. Association for Computing Machinery, New York, NY, USA, 4759–4765. doi:10.1145/3583780.3615476

Alexander Quinn Nichol and Prafulla Dhariwal. 2021. Improved Denoising Diffusion Probabilistic Models. In *Proceedings of the 38th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 139)*, Marina Meila and Tong Zhang (Eds.). PMLR, 8162–8171. https://proceedings.mlr.press/v139/nichol21a.html

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence Level Training with Recurrent Neural Networks. *CoRR* abs/1511.06732 (2015). https://api.semanticscholar.org/CorpusID:7147309

Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T Chiu, Alexander Rush, and Volodymyr Kuleshov. 2024. Simple and Effective Masked Diffusion Language Models. In *Advances in Neural Information Processing Systems*, Vol. 37. doi:10.52202/079017-4135

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis K. Titsias. 2024. Simplified and generalized masked diffusion for discrete data. In *Proceedings of the 38th International Conference on Neural Information Processing Systems* (Vancouver, BC, Canada) *(NIPS '24)*. Curran Associates Inc., Red Hook, NY, USA, Article 3277, 37 pages.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. 2015. Deep Unsupervised Learning using Nonequilibrium Thermodynamics. In *Proceedings of the 32nd International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 37)*, Francis Bach and David Blei (Eds.). PMLR, Lille, France, 2256–2265. https://proceedings.mlr.press/v37/sohl-dickstein15.html

Yang Song, Jascha Narain Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. 2020. Score-Based Generative Modeling through Stochastic Differential Equations. *ArXiv* abs/2011.13456 (2020). https://api.semanticscholar.org/CorpusID:227209335

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*, I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

David Weininger. 2002. SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. *Journal of Chemical Information and Computer Sciences* 28, 1 (05 2002), 31–36. arXiv:https://pubs.acs.org/jcisd8/article-pdf/28/1/31/10787984/ci00057a005.pdf doi:10.1021/ci00057a005

A Appendix

B Additional Experimental Details

In all learnable Markov noise models in our experiments (i.e., D3PM-Reinforce and VSDD, see Section 6) we rely on the same architecture of the forward noise process $Q_{t,\phi}$ and the learnable noise kernel. Specifically, the off-diagonal transition kernel $M_{t,\phi}$ is parameterised by a time-conditioned bilinear scoring module that computes token-to-token affinities according to Equation (4), and $A(t) \in \mathbb{R}^{d\times d}$ is produced by a two-layer MLP with SiLU activations operating on sinusoidal time features. Rows of the score matrix are softmax-normalised, giving a valid row-stochastic off-diagonal kernel $M_{t,\phi}$. The full single-step transition matrix $Q_{t,\phi}$ is then computed according to Equation (3), and the cumulative product $\bar{Q}_{t,\phi} = \prod_{\tau=1}^{t} Q_{\tau,\phi}$ is computed sequentially. A terminal-uniform regularizer encourages $\bar{Q}_T$ to converge to a uniform stationary distribution over valid tokens.

B.1 Molecular Domain

Data & Vocabulary. In the molecular domain, we randomly sample 1M molecules and use a 90/10 train/validation split. We tokenize SMILES sequences provided in the dataset at the atom level using a custom tokenizer that handles bracket atoms (e.g., `[C@@H]`, `[NH+]`), two-letter elements (e.g., `Cl`, `Br`), single-letter atoms, and structural characters (bonds, branches, ring closures). This yields a compact vocabulary of $|\mathcal{V}| = 64$ tokens and a maximum sequence length of $L = 134$. An `[EOS]` token terminates each molecule; sequences shorter than L are right-padded with `[PAD]`.

Denoiser architecture. All baselines as well as our proposed model use the same denoising backbone: a bi-directional Transformer encoder (Vaswani et al., 2017) with the embedding dimensionality $d = 512$, ten layers, 32 attention heads, and sinusoidal time embeddings injected additively into the input representation. The diffusion horizon is $T = 50$ steps with a linear noise schedule $\bar{\alpha}_t$ decreasing from 1 to 0. Models are trained for 50 epochs with the Adam optimizer (learning rate 10^{-4}, batch size 512) on two A100 GPUs.

B.2 Text Generation Domain

Data & Vocabulary. We evaluate on the TinyStories dataset (Eldan and Li, 2023), tokenised with a byte-pair encoding (BPE) tokeniser trained on the full training split, yielding a vocabulary of $V = 2048$ tokens and a maximum sequence length of $L = 256$. Three special tokens are reserved: [PAD] for right-padding sequences shorter than L, [EOS] for end-of-sequence, and [MASK] (used only by the masked-diffusion baselines). We train on 100k sequences randomly selected from the training split and validate on a held-out set of 1,000 sequences.

Denoiser architecture. All models share the same denoising network $p_\theta(x_0|x_t, t)$: a bi-directional Transformer encoder with the embedding dimension $d = 512$, ten layers, sixteen attention heads, and sinusoidal time embeddings added to the input representation. The model takes noisy tokens x_t and a time embedding as input and outputs logits over the vocabulary at every sequence position. We perform training on multiple A100 80 GB GPUs with a batch size of 256. The diffusion horizon has $T = 50$ steps with a linear noise schedule $\bar{\alpha}_t$ decreasing from 1 to 0. The denoiser is trained with Adam optimizer with the learning rate $\eta_\theta = 10^{-4}$, weight decay 10^{-4}, and gradient clipping at 1.0. The kernel parameters ϕ are optimized with a separate Adam instance with the learning rate $\eta_\phi = 2 \times 10^{-4}$.

B.3 Playlist Domain

Data & Vocabulary. We evaluate our model on a proprietary playlist dataset comprising listening sequences derived from over one million unique music tracks, each represented by an 80-dimensional embedding vector. The dataset consists of 44,042 listening histories (36,765 training, 7,277 validation), where each history is a variable-length sequence of tracks. To obtain a discrete vocabulary suitable for our model, we map each track's 80-dimensional embedding vector to a *semantic ID* (SID) via locality-sensitive hashing (LSH): 21 random hyperplanes are partitioned into 3

levels of 7 bits each, yielding $2^7 = 128$ buckets per level plus 3 special tokens (PAD, MASK, EOS) for a vocabulary of $V = 131$. Each track is thus represented by a triplet (t_0, t_1, t_2) of *semantic tokens*, and the resulting corpus contains 228,132 unique triplets. Sequences are truncated or padded to a fixed length of $L = 150$ semantic tokens.

Denoiser architecture. The denoiser architecture for playlist generation is similar to the other two tasks. We let the dimensionality of the token embeddings be $d = 512$, and we use ten layers with 16 attention heads, trained for 50 epochs with batch size 28 and the Adam optimizer (follower lr $= 10^{-4}$, leader lr $= 2 \times 10^{-4}$). The diffusion process uses $T = 50$ steps.

C Re-Drafting Analysis

In this section, we expand our analysis of the *re-drafting* (conditional infilling) capabilities of VSDD for the text and playlist generation domains. Conceptually, such an analysis is similar to the qualitative analysis that we presented for the molecular generation domain (see Section 6.1, last paragraph). However, as both the playlist and the text generation tasks have significantly larger vocabularies, the visual analysis of the learned forward noise process (Figure 1b) is not straightforward in these domains. Instead, we construct a number of rule-based re-drafting tests, and we analyse the performance of our model on those tests.

C.1 Text Generation

In the text generation domain, we evaluate three discrete diffusion models trained on the TinyStories dataset. Re-drafting tests whether a model, given a story with a localised corruption, can restore the original content by conditioning on the uncorrupted context. This tests the model's ability to maintain a coherent narrative and factual agreement across the sequence.

To this end, we compare our VSDD model that learns the forward noise process against two baselines that rely on a fixed (pre-computed) forward noise process, namely, MDLM and D3PM-Uniform. We then apply *repaint-style* conditional infilling, i.e., given a sequence with a designated repainting mask, we use the model to regenerate only the masked positions while keeping the context positions fixed.

In the VSDD and D3PM-Uniform experiments, we initialise the positions marked with the repainting mask with uniform random noise over the valid tokens. The full reverse diffusion is then run for T steps. At each step, the model predicts $p_\theta(x_0|x_t, t)$ and computes the posterior sampling probabilities $p(x_{t-1}|x_t, x_0)$. It then samples the new tokens from this posterior. Only the masked positions are updated; context positions are held fixed. For the MDLM model the edit positions are set to the absorbing mask token [MASK]. Only currently-masked non-pad positions are updated at each step while the context positions remain unchanged. All experiments use five independent trials per corruption per model.

Experiment 1: Entity Swap. For the first experiment, we construct five templated stories with repeated entities (names and objects). Each story contains 3–5 occurrences of a target entity. We corrupt exactly one occurrence by replacing its BPE token (see Appendix B.2) with a different entity token, creating a single-token inconsistency. All entity names ("Lily", "Tom", "Max", "Sam", "Ben") and objects ("ball", "dog", "car", "cat", "train") are verified to be single BPE tokens, ensuring the corruption is a clean one-token swap with no sequence-length change.

Observe that in this setting the absorbing-state mechanism of MDLM is particularly well-suited for single-token infilling. Indeed, MDLM is directly optimized for this specific task of unmasking a single masked token at the corrupted position with full surrounding context. Consequently, its training objective is precisely to predict the clean token at masked positions. This provides an ideal solution for entity swap, and we are interested in analysing how close the rest of the models can approach MDLM on this task.

The five corruption instances are in Table 3, and we provide an example of such a corruption in Table 5. For each story we run five corruption trials, and average the resulting metrics across the trials and stories. We measure the exact repair rate as a fraction of the corrupted tokens restored to the original value (averaged over all trials) and consistency as the fraction of all entity occurrences

Table 3: Synthetic story corruptions. For each story, an entity is replaced at a selected occurrence.

Story name	Corruption	# occurrences	Corrupted occurrence
Lily and red ball	Lily $\rightarrow$ Tom	4	2
Tom and Max	Max $\rightarrow$ Sam	5	1
Sam and blue car	car $\rightarrow$ dog	4	3
Lily and cat	cat $\rightarrow$ dog	5	2
Ben and toy train	train $\rightarrow$ ball	5	3

Table 4: Aggregate repair performance. Results report the mean and the standard deviation across corruption types.

Model	Exact Repair $\uparrow$	Consistency $\uparrow$
VSDD	0.880 ± 0.160	0.974 ± 0.033
D3PM-Uniform	0.360 ± 0.265	0.860 ± 0.065
MDLM	0.960 ± 0.080	0.992 ± 0.016

that agree with the most-common value after infilling. A score of 1.0 means all instances of the entity hold the same token. Table 4 contains the aggregated results.

From Table 4 we see that VSDD closely approaches the “ideal" performance of MDLM, leaving a substantial gap with D3PM-Uniform. These results indicate that even though VSDD was not trained to perform well on the entity swap task, it manages to efficiently re-draft the corrupted parts of the sequence. Despite the small performance gap with MDLM, below we illustrate that the infilled stories are still structurally and semantically consistent.

In our first example (Table 5), D3PM-Uniform never recovers "Lily" across all 5 trials (repair = 0.0); it consistently generates the pronoun "She", which is grammatically plausible but breaks the entity pattern. VSDD recovers "Lily" in 4 out of 5 trials (repair = 0.8).

The second example (Table 6) is the hardest corruption: "Max" appears 5 times but the corrupted position is the first occurrence, where the model must choose between two plausible names ("Max" from later context vs. "Sam" or "Tom" from the immediate sentence). MDLM occasionally generates "Tom" (the human character’s name) instead of "Max", showing a consistency failure despite high overall repair rate.

With strong contextual cues (three other occurrences of "blue car" surrounding the corruption), all models perform well (Table 7). In this case, even D3PM-Uniform achieves 80% repair, demonstrating that sufficient redundancy can compensate for the lack of the structured noise process.

Overall, we see that using D3PM-Uniform with an unstructured noise process frequently generates plausible but incorrect tokens (e.g., the pronoun "She" instead of "Lily", or the article "a" instead of "cat"). In contrast, VSDD’s learned kernel provides intermediate token-level affinity structure, yielding substantially better repair than the uniform baseline.

Experiment 2: Subsequence Removal. In our second experiment, a contiguous middle subsequence (approximately 35%–55% of the story length, corresponding to 8–9 tokens) is removed and replaced with [MASK] tokens. The removed subsequence typically contains a narrative bridge connecting the story’s setup to its conclusion (see the example below). For all stories, we measure the *token overlap* (i.e., the fraction of in-filled tokens that exactly match the original removed subsequence). Table 11 shows the aggregated results.

From Table 11 we see that across the three stories VSDD achieves the highest average overlap (24.3%) and produces the most coherent bridges, followed by MDLM (18.4%) and D3PM-Uniform (16.3%). The low absolute values of the token overlaps are expected since there are many valid ways to bridge the narrative gap (so the metric provides a lower bound on generation quality).

Table 5: The 2nd occurrence of "Lily" is corrupted to "Tom", each model attempts to recover the original token.

Example 1: Lily → Tom *(2nd occurrence)*

Original Once upon a time there was a little girl named Lily. **Lily** had a big red ball. She loved to play with the red ball every day. One day Lily took the red ball to the park. Lily was very happy.

Corrupted Once upon a time there was a little girl named Lily. **Tom** had a big red ball. She loved to play with the red ball every day. One day Lily took the red ball to the park. Lily was very happy.

Model	Infilled text	Repair?
VSDD	. . . named Lily. **Lily** had a big red ball. She loved . . .	Yes
D3PM-Uniform	. . . named Lily. **She** had a big red ball. She loved . . . *Generates "She" instead of "Lily".*	No
MDLM	. . . named Lily. **Lily** had a big red ball. She loved . . .	Yes

Table 6: The 1st occurrence of "Max" is corrupted to "Sam".

Example 2: Max → Sam *(1st occurrence)*

Original Tom had a small dog named **Max**. Tom and Max liked to play in the park. Every morning Tom took Max for a long walk. Max was a good dog and Tom loved Max very much.

Corrupted Tom had a small dog named **Sam**. Tom and Max liked to play in the park. Every morning Tom took Max for a long walk. Max was a good dog and Tom loved Max very much.

Model	Infilled text	Repair?
VSDD	. . . dog named **Max**. Tom and Max liked . . . *Successful in 3/5 trials.*	Yes
D3PM-Uniform	. . . dog named **Max**. Tom and Max liked . . . *Successful in 1/5 trials.*	Rarely
MDLM	. . . dog named **Tom**. Tom and Max liked . . . *Repairs in 4/5 trials, but occasionally generates "Tom" instead of "Max".*	Partial

In Table 12, we provide an example from this experiment. From the example, we can see that VSDD preserves both the key entity ("red ball") and the narrative structure ("she loved ... She took her"), yielding the highest token overlap. D3PM-Uniform produces incoherent repetition ("The red liked red new toy"). MDLM generates a grammatical but nonsensical phrase ("ball with many hair"). Qualitatively, VSDD more consistently preserves key entities from the surrounding context and generates grammatical continuations.

Table 7: The third occurrence of "car" is corrupted to "dog".

Example 3: car → dog *(3rd occurrence)*

Original There was a little boy named Sam. Sam had a blue car. Sam liked to play with his blue car. One day Sam lost his blue **car**. Sam was sad but then he found the blue car under the bed.

Corrupted There was a little boy named Sam. Sam had a blue car. Sam liked to play with his blue car. One day Sam lost his blue **dog**. Sam was sad but then he found the blue car under the bed.

Model	Infilled text	Repair?
VSDD	. . . lost his blue **car**. Sam was sad . . . *Successful in 5/5 trials.*	Yes
D3PM-Uniform	. . . lost his blue **car**. Sam was sad . . . *Successful in 4/5 trials.*	Mostly
MDLM	. . . lost his blue **car**. Sam was sad . . . *Successful in 5/5 trials.*	Yes

Table 8: The second occurrence of "cat" is corrupted to "dog".

Example 4: cat → dog *(2nd occurrence)*

Original Lily had a pretty cat. The **cat** was soft and white. Lily liked to pet her cat every day. One day the cat found a little mouse. Lily and the cat played in the garden.

Corrupted Lily had a pretty cat. The **dog** was soft and white. Lily liked to pet her cat every day. One day the cat found a little mouse. Lily and the cat played in the garden.

Model	Infilled text	Repair?
VSDD	. . . The **cat** was soft and white . . . *Successful in 5/5 trials.*	Yes
D3PM-Uniform	. . . The **a** was soft and white . . . *Occasionally generates the article "a" instead of "cat".*	No
MDLM	. . . The **cat** was soft and white . . . *Successful in 5/5 trials.*	Yes

Table 9: Qualitative example for the train → ball corruption. The third occurrence of "train" is corrupted to "ball", and each model attempts to recover the original token.

Example 5: train → ball *(3rd occurrence)*

Original Ben had a toy train. Ben loved his toy train. Every day Ben played with the toy **train** in his room. One day Ben took the toy train to show his friend. His friend liked the toy train too.

Corrupted Ben had a toy train. Ben loved his toy train. Every day Ben played with the toy **ball** in his room. One day Ben took the toy train to show his friend. His friend liked the toy train too.

Model	Infilled text	Repair?
VSDD	. . . the toy **train** in his room . . . *Successful in 5/5 trials.*	Yes
D3PM-Uniform	. . . the toy **train** in his room . . . *Successful in 2/5 trials.*	Partial
MDLM	. . . the toy **train** in his room . . . *Successful in 5/5 trials.*	Yes

Table 10: Removed subsequences used for re-drafting. Each example masks a contiguous subsequence of tokens from the original story.

Story	Removed subsequence	subsequence length
Lily & red ball	*"red ball. She loved to play with the"*	9
Tom & Max	*"in the park. Every morning Tom took"*	8
Sam & blue car	*"liked to play with his blue car. One"*	9

Table 11: Span infilling performance. Token Overlap measures agreement between the generated and original removed spans.

Model	Token Overlap ↑
VSDD	$\mathbf{0.243 \pm 0.127}$
D3PM-Uniform	0.163 ± 0.159
MDLM	0.184 ± 0.082

Table 12: Example: Subsequence Removal.

Example: Lily story *(9 tokens removed)*		
Original	...Lily. Lily had a big **red ball. She loved to play with the** red ball every day ...	
Removed span	*"red ball. She loved to play with the"*	
Model	**Infilled span**	**Overlap ↑**
VSDD	"**red ball that she loved. She took her**"	$\mathbf{42.2}\%$
D3PM-Uniform	"**red toy. The red liked red new toy**"	11.1%
MDLM	"**ball with many hair. She played with the**"	8.9%

C.2 PLAYLIST GENERATION

Unlike in the molecular domain discussed in Section 6.1, analysing the re-drafting capabilities of VSDD by visually inspecting the learned forward noise matrices in the playlist generation domain is not feasible. This is because the different token substitutions are not based on the underlying world knowledge but on the subjective (unobservable) user preferences. In this section, we show that the model can successfully repair local rule-based corruptions that typically reduce playlist quality.

To evaluate whether the learned forward process captures meaningful structural properties of playlist sequences, we design a targeted repair experiment. To this end, we introduce controlled corruptions into held-out playlists, and we measure the ability of the trained model to restore the structural coherence of the playlists via conditional reverse diffusion.

In particular, we define three corruption operators, each modifying exactly one track (i.e., three semantic tokens) per sequence:

- **Track duplication**. A source track position s is selected uniformly at random, and a distinct destination position $d \neq s$ is selected uniformly at random. The SID triplet corresponding to the track at position d is overwritten with a copy of the triplet of the track at position s, simulating a repeated-track artifact.
- **Transition disruption**. We identify the consecutive track pair $(i, i{+}1)$ with the highest audio-feature[3] cosine similarity (i.e., the smoothest transition) across all adjacent pairs in the playlist. We then scan all remaining tracks $j \notin \{i, i{+}1\}$ and select the one whose audio-feature vector has the lowest cosine similarity to track i. The SID triplet corresponding to the track at position $i{+}1$ is then overwritten with the triplet of this maximally dissimilar track, creating a jarring transition at the previously smoothest point, thereby reducing consumption (Moor et al., 2023). The corrupted region is $\{3(i{+}1), 3(i{+}1){+}1, 3(i{+}1){+}2\}$.
- **Energy misplacement**. We identify the track with the highest energy value across all positions and swap it with the track at the very first position of the playlist (the playlist opening). If the highest-energy track is already at the first position, the lowest-energy track is swapped to this position instead. The corrupted region is $\{0, 1, 2\}$ (the triplet corresponding to the opening track only). Note that while the swap modifies two track positions, only the track in the first position is designated for repair, testing whether the model can restore an appropriate opening track given one-sided (right-only) context.

For each corrupted sequence, we construct a binary generation mask $\mathbf{m} \in \{0, 1\}^L$ with $m_\ell = 1$ at the three corrupted token positions and $m_\ell = 0$ elsewhere. The corrupted sequence $\tilde{\mathbf{x}}$ and mask $\mathbf{m}$ are passed to the conditional reverse diffusion procedure: masked positions are initialized with tokens drawn uniformly from the valid SID vocabulary, while unmasked positions retain the corrupted values throughout. The model then performs $T = 50$ reverse diffusion steps using the learned transition matrices from the Stackelberg-trained forward process. At each step, reverse sampling probabilities are computed from the model's predicted logits and the learned posterior, but token updates are applied only at the three masked positions. The surrounding 147 tokens serve as fixed conditioning context.

We report three quantities, each computed over 200 corrupted validation playlists:

- **Repair rate**. The fraction of corrupted token positions at which the model's output differs from the corrupted value. A high repair rate indicates that the model recognizes the corruption as inconsistent with the surrounding context.
- **Audio smoothness**. The mean cosine similarity between the 16-dimensional audio-feature vectors of the consecutive tracks, computed over all adjacent pairs in the playlist. We report this for the original (O), corrupted (C), and repaired (R) sequences.
- **Style diversity**. The number of unique genre/style descriptor tags across all tracks in the sequence, obtained from per-track metadata (with approximately 50 weighted descriptors per track from the editorial taxonomy). Reported for O, C, and R.

Table 13 summarizes the repair outcomes.

[3] In our dataset, audio features are 16-dimensional vectors of normalized acoustic descriptors (energy, danceability, valence, tempo, etc.) aggregated per SID triplet.

Table 13: Repair results for different corruption types.

		Smooth			Diversity		
Corruption	Repair rate	O	C	R	O	C	R
Duplication	78.9%	0.9800	0.9802	0.9807	2477	2453	2476
Transition disruption	86.1%	0.9800	0.9767	0.9792	2510	2497	2520
Energy misplacement	85.3%	0.9800	0.9797	0.9799	2477	2477	2481

First, we see that VSDD actively repairs corrupted positions in 78–86% of cases, with the highest repair rate for transition disruption (86.1%), where the corrupted track is maximally inconsistent with its immediate neighbours. This suggests the model has learned local coherence constraints: It detects that a track with very different audio characteristics does not belong between its neighbours and proposes a more suitable replacement.

For transition disruption, the corruption reduces audio smoothness from 0.9800 to 0.9767. After repair, the smoothness recovers to 0.9792. The model does not fully recover the original smoothness because it is not constrained to reproduce the original track. Instead, it generates any track consistent with the learned distribution conditioned on the context.

Track duplication yields a slightly lower repair rate (78.9%), consistent with the observation that a duplicated track may be contextually plausible at its destination: Unlike transition disruption, duplication does not necessarily create a local inconsistency, especially when the track is highly relevant for the user.

Energy misplacement achieves an intermediate repair rate (85.3%) despite having access to only right-side context at the very first position of the playlist. The model must infer an appropriate opening track from the subsequent sequence alone, which is a more complex task as users typically interact with the playlist from left to right (so the prefix context is often more informative when making the track allocation decision than the suffix-context).

Audio smoothness and style diversity show minimal variation across conditions because each corruption modifies only one of approximately 49 tracks. The smoothness metric averages over 48 consecutive pairs, diluting the local effect of a single corrupted transition. Style diversity is similarly insensitive: each track contributes dozens of tags, and the union over 49 tracks saturates at approximately 2,500 unique descriptors regardless of single-track substitutions.

The repair experiment provides evidence that the Stackelberg-trained diffusion model captures local sequential structure in playlist data. The learned forward process, optimized to improve the denoiser's generalization, produces a reverse model capable of context-sensitive infilling. The high repair rate for transition disruption (86.1%) demonstrates that the model encodes audio-feature continuity as an implicit constraint, even though audio features are not explicitly provided during training as the model operates solely on discrete SID tokens derived from track embeddings via locality-sensitive hashing. The correlation between embedding cosine similarity and learned transition probabilities (Spearman $\rho = 0.74$ for the learned noise vs. embedding similarity, see Figure 2c) provides a mechanistic explanation: The forward process preferentially corrupts tokens toward embedding-similar alternatives, and the reverse process inherits this inductive bias.

D Degenerate Noise

Table 14 illustrates the per-row entropy and the dominant transitions of $Q_{1,\phi}$ at $t{=}1$ of the most frequent tokens learned by naive joint optimisation of ϕ and θ (D3PM-Reinforce, see Section 6.1). The uniform maximum entropy in this example is $\ln(V{-}1) \approx 7.62$. From the table we can see that the top rows exhibit near-deterministic transitions, indicating degenerate noise process collapse targeting the high-frequency tokens.

Table 14: Naive joint optimisation of ϕ and θ learns trivial near-deterministic noise processes.

Token	Row Entropy	Top Target	$P(i \to j)$
.	0.023	to	0.998
the	0.062	t	0.996
and	0.095	it	0.993
,	0.096	that	0.993
to	0.111	.	0.992
it	0.115	was	0.991
in	0.216	,	0.983
was	0.232	and	0.981
t	0.439	was	0.963
she	0.552	that	0.952
She	0.730	They	0.935
so	0.863	in	0.921
out	1.733	up	0.829
on	1.944	for	0.806