CHOOSE BEFORE YOU DECODE: LATENT Q-GUIDED GENERATION FOR DIFFUSION LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Continuous latent diffusion language models generate an entire response by iteratively refining a continuous latent representation, yet existing samplers typically assign the same denoising budget to every trajectory. Consequently, test-time methods such as Best-of-N must fully denoise and often decode every candidate before selection. We introduce *Latent Q-Guided Generation (LQG)*, a value-guided inference framework that evaluates latent trajectories before denoising is complete. LQG formulates the reverse process as a finite-horizon Markov decision process with terminal text utility and trains a Transformer critic through temporal-difference policy evaluation of the frozen denoiser. At inference time, multiple trajectories are advanced to an intermediate decision step, where the critic identifies promising candidates and prunes the remainder. Across unconditional and conditional generation tasks, LQG consistently improves expected terminal utility under independently specified scorers while substantially reducing denoising and decoding costs relative to terminal-scored Best-of-N. Ablations further establish the importance of cross-trajectory selection and local margin regularization, while revealing an early-horizon limit on reliable trajectory ranking. These results demonstrate that latent value estimation provides a principled mechanism for efficient test-time generation in latent diffusion language models.

1 INTRODUCTION

Latent diffusion language models (LDLMs) offer a compelling alternative to autoregressive generation (Lovelace et al., 2023; Zhang et al., 2023; Shabalin et al., 2024; Guo et al., 2026a; Kang et al., 2026). Rather than generating text token by token from left to right, they use an encoder-decoder architecture to construct and iteratively refine an entire response in a continuous latent space before decoding it into text. This formulation enables sequence-level semantic modeling and parallel refinement across positions. However, continuous text representations can be fragile as accumulated denoising errors can degrade the decoded text, while generation quality can be sensitive to the latent geometry and noise schedule (Lovelace et al., 2023; Zhang et al., 2023; Shabalin et al., 2024). More broadly, recent studies show that generative performance depends critically on learning a compressed, smooth, and diffusion-friendly latent geometry, motivating improved representation learning and regularization across text and other modalities (Meshchaninov et al., 2025; Zhu et al., 2025; Kouzelis et al., 2025; Skorokhodov et al., 2025). Since different random initializations induce different latent trajectories and responses, a natural strategy is to sample multiple initial noises and select or aggregate the resulting candidates (Lovelace et al., 2023; Kang et al., 2026). Efficiently evaluating and allocating test-time compute across trajectories is therefore increasingly important.

Despite this progress, existing LDLM inference procedures provide no principled mechanism for making decisions from intermediate latent states, giving rise to three closely related challenges: (i) *Fixed-budget denoising at inference time*: Once a latent trajectory is initialized, the denoiser typically executes the entire reverse process without assessing whether the trajectory is likely to produce a useful response. Although intermediate latents determine subsequent denoising updates, they are not used to decide whether a trajectory remains worth continuing. (ii) *Lack of a learned intermediate decision signal*: Determining whether a latent trajectory is worth continuing requires a reliable intermediate signal. However, language utility is naturally defined on the final decoded text, whereas intermediate diffusion states are noisy continuous representations. The resulting challenge is to propagate terminal utility backward through the denoising process without relying

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.

on heuristic stepwise rewards, such as reconstruction quality or denoiser confidence, which may be poorly aligned with final response quality. (iii) *Inefficient test-time compute allocation*: Without a reliable intermediate decision signal, test-time computation cannot be allocated selectively across trajectories. Conventional sampling-based methods, such as Best-of-N sampling and majority voting, follow a generate-then-select procedure: they assign the same denoising budget to every candidate and evaluate or aggregate the candidates only after all of them have been fully denoised and decoded. Consequently, trajectories that become predictably inferior during denoising still incur the full completion cost. These challenges motivate a fundamental yet underexplored question: *Can we predict a latent trajectory's terminal utility before denoising is complete and use this prediction to achieve efficient test-time response generation?*

To address these challenges, we propose **Latent Q-Guided Generation (LQG)**, a value-guided framework for adaptive test-time generation in LDLMs. LQG is built on three core ideas: (1) We reformulate latent denoising as a finite-horizon Markov decision process (MDP), treating intermediate latent states as decision points at which the prospects of a trajectory can be assessed. (2) We learn a latent Q-function through a temporal-difference method in reinforcement learning (RL) and enforce policy evaluation of the frozen denoiser. The learned Q-function estimates the terminal utility of a partially denoised trajectory under the remaining reverse process, without requiring heuristic stepwise rewards or intermediate decoding. (3) We use the learned Q-function to perform trajectory selection and pruning: multiple candidates are explored up to an intermediate decision step, after which unpromising trajectories are discarded and the remaining denoising computation is allocated to the most promising candidate. Together, these components enable LDLMs to use intermediate latent information to guide test-time computation and reduce the cost of sampling-based inference.

Our experiments across unconditional and conditional generation tasks show that LQG consistently improves expected terminal utility under both training-time and held-out scorers. It matches or outperforms supervised latent critics and output-dependent baselines while requiring autoregressive decoding only for the selected candidate. Ablation studies further demonstrate that cross-trajectory selection and local margin regularization are essential for learning a reliable latent critic and achieving strong generation quality.

We highlight the main contributions of this paper as follows: (i) *A new perspective on LDLM inference*: We formulate latent denoising as a sequential decision-making process in which intermediate latent states inform whether a trajectory is worth continuing, providing a principled basis for value-guided inference. (ii) *Latent Q-guided generation*: We propose LQG, an RL-based framework that models latent denoising as a finite-horizon MDP and learns a latent Q-function to predict the terminal utility of unfinished denoising trajectories. (iii) *Practical efficacy across generation settings*: We demonstrate that LQG improves terminal utility while reducing the cost of sampling-based inference in both unconditional and conditional language generation.

2 Related Work

Continuous Latent Diffusion Language Models. A growing line of work models text by first compressing it into a continuous latent space with a pretrained or jointly trained encoder-decoder, and then learning a diffusion prior over that space. Specifically: (i) *Token-embedding diffusion methods* such as Diffusion-LM (Li et al., 2022), SED (Strudel et al., 2022), GENIE (Lin et al., 2023), and Difformer (Gao et al., 2024) diffuse directly on token embeddings. (ii) *Latent-space methods* such as LD4LG (Lovelace et al., 2023), PLANNER (Zhang et al., 2023), and TEncDM (Shabalin et al., 2024) compress text into a lower-dimensional manifold defined by a frozen BART or T5 encoder before applying a diffusion transformer (DiT). Most recently, Cola DLM (Guo et al., 2026a) pushes this hierarchical view to its logical end, training a dedicated Text-VAE together with a block-causal DiT prior over the latent and arguing that the diffusion process should model *latent prior transport* rather than direct observation recovery. All of these methods, however, share the same inference template, where each denoising step is a single feed-forward pass of a network trained to regress the clean latent, the noise, or the velocity, followed by a deterministic DDIM update. The per-step computation is therefore fixed at train time and decoupled from problem difficulty. By contrast, our proposed LQG preserves the continuous-latent, encoder-decoder backbone of LD4LG but replaces it with a learned Q function obtained via TD learning, exposing a test-time compute knob within an

otherwise standard latent-diffusion pipeline. Due to the page limit, a more detailed discussion about the related work can be found in Appendix C.

Latent-Space and Output-Dependent Guidance. Related Test-time methods can be distinguished by whether they evaluate native intermediate representations or partially or fully realized outputs: (1) *Latent-space guidance* evaluates diffusion trajectories before mapping them into the output space. Feynman-Kac diffusion steering (Singhal et al., 2025) maintains a particle population and periodically reweights and resamples it using intermediate potentials, with applications to image and discrete text diffusion. LQG also intervenes before completion but directly predicts terminal text utility from LDLM latents, enabling selection and pruning without intermediate or terminal decoding. (2) *Output-dependent guidance* relies on partially or fully realized outputs. Progressive Seed Pruning (Guimaraes & Perona, 2026b) and TTSnap (Yu et al., 2025b) prune image trajectories early but score clean-image estimates reconstructed from intermediate diffusion states. Their pruning principle may transfer to LDLMs, although doing so requires reliable intermediate decoding and evaluation. For autoregressive models, TrajSelector and SWIFT select among completed trajectories using lightweight scorers derived from hidden states or logits (Yu et al., 2025a; Guo et al., 2026b). Value-Augmented Sampling (Han et al.), Evolutionary Guided Decoding (Liu et al., 2026), and Energy-Based Decoding (Wang et al., 2026) instead guide token-level generation with value, reward, or energy signals, while LatentSeek (Li et al., 2025) optimizes latent representations at test time. Most of these methods do not evaluate latent diffusion trajectories directly, motivating LQG's use of a latent Q-function for selection and pruning before text realization.

3 Preliminaries

LDLMs (Lovelace et al., 2023; Zhang et al., 2023) take a different route from token-level diffusion models, which directly corrupt and denoise discrete tokens or token embeddings (Li et al., 2022; Lin et al., 2023). Instead of asking the diffusion model to handle both high-level semantics and low-level token realization, an LDLM operates on a continuous representation of an entire piece of text. Concretely, an encoder maps an input sequence into contextual representations, which can be further compressed by an autoencoder into a compact, fixed-length latent $\mathbf{x}_0$. Diffusion is then performed in this latent space, and the final denoised latent is mapped back into decoder-compatible features and realized as text by a pretrained autoregressive decoder. Accordingly, in LDLMs, the diffusion process can focus primarily on global semantic and structural information, while a pretrained language model handles the generation of coherent text.

Training of an LDLM. Training typically proceeds in two stages: (1) The latent autoencoder, including the compression and reconstruction modules, is trained to preserve the input text, commonly using a token-level cross-entropy reconstruction loss while keeping the underlying pretrained encoder and decoder frozen. (2) The diffusion model is trained on the resulting clean latents. Given a noise schedule $\{\bar{\alpha}_t\}$ with $\bar{\alpha}_t \in [0, 1]$ (*e.g.,* a cosine schedule), the forward process constructs a noisy latent at a randomly sampled time step $t \in [T]$ as $\mathbf{z}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$ with $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. A regression denoiser f_θ then predicts the corresponding clean latent as $\hat{\mathbf{x}}_0^{(t)} := f_\theta(\mathbf{z}_t, t)$, and is trained using the standard loss function of diffusion models (Ho et al., 2020; Song et al., 2021a):

$$\mathcal{L}_{\text{diff}}(\theta) := \mathbb{E}_{\mathbf{x}_0,t,\boldsymbol{\epsilon}}\left[\|f_\theta(\mathbf{z}_t, t) - \mathbf{x}_0\|_2^2\right]. \tag{1}$$

Thus, the denoiser f_θ can learn to recover the global text representation from different levels of Gaussian corruption.

Inference with an LDLM. At inference time, generation runs this process in reverse. Starting from a Gaussian latent $\mathbf{z}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, the trained denoiser is repeatedly applied to obtain progressively cleaner semantic representations. Under the ancestral DDPM sampler (Ho et al., 2020), each reverse step is written in terms of the clean-latent estimate $\hat{\mathbf{x}}_0^{(t)}$ as

$$\mathbf{z}_{t-\Delta t} = \boldsymbol{\mu}_t\big(\mathbf{z}_t, \hat{\mathbf{x}}_0^{(t)}\big) + \sqrt{\beta_t}\,\boldsymbol{\xi}_t, \quad \boldsymbol{\mu}_t(\mathbf{z}, \hat{\mathbf{x}}_0) := \frac{\sqrt{\bar{\alpha}_{t-\Delta t}}\,\beta_t}{1-\bar{\alpha}_t}\,\hat{\mathbf{x}}_0 + \frac{\sqrt{1-\beta_t}\,(1-\bar{\alpha}_{t-\Delta t})}{1-\bar{\alpha}_t}\,\mathbf{z}, \tag{2}$$

where $\beta_t := 1 - \bar{\alpha}_t/\bar{\alpha}_{t-\Delta t}$, $\boldsymbol{\xi}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is fresh Gaussian noise injected at every step, and $\hat{\mathbf{x}}_0^{(t)} = f_\theta(\mathbf{z}_t, t)$ is determined by the denoiser learned during training. After the reverse trajectory reaches $t = 0$, the resulting latent is expanded into the representation space expected by the pretrained decoder, which converts the semantic representation into a variable-length natural-language sequence.

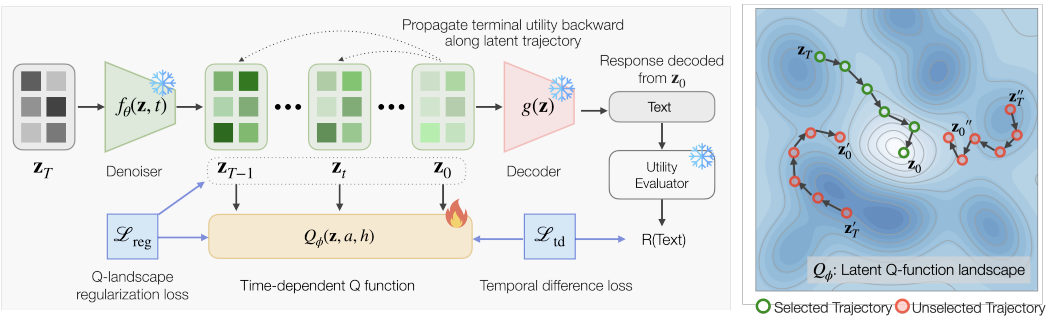


Figure 1: An illustration of the latent Q-guided generation framework.

Multi-Trajectory Latent Generation. At inference time, an LDLM can generate multiple candidates by sampling N initial latents $\mathbf{z}_T^{(1)}, \cdots, \mathbf{z}_T^{(N)} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(\mathbf{0}, \mathbf{I})$, and denoising them in parallel to obtain $\mathbf{z}_0^{(1)}, \cdots, \mathbf{z}_0^{(N)}$, which are subsequently decoded into texts $y^{(1)}, \cdots, y^{(N)}$. These candidates can then be aggregated in the text space. For example, LD4LG (Lovelace et al., 2023) applies minimum Bayes-risk decoding to select the candidate with the highest average similarity to the other sampled outputs. More generally, the resulting texts could be aggregated using Best-of-N selection or self-consistency, although these strategies have primarily been studied for autoregressive or token-level diffusion language models rather than continuous LDLMs. A common limitation of such output-level aggregation is that it assigns the same full denoising budget to every latent trajectory and can operate only after all N latent trajectories have been completely denoised and decoded. Consequently, even candidates that are already unpromising at an intermediate stage incur the full cost of both latent denoising and autoregressive decoding.

4 Methodology

4.1 Latent Denoising Process as a Finite-Horizon MDP

Latent denoising MDP. We propose to recast the denoising chain at inference time as a discrete-time finite-horizon MDP over latents. Specifically, to put this in the context of MDP: (i) *Horizon*: Each rollout in this MDP has T steps, which corresponds to the number of denoising steps. (ii) *State*: The initial state is defined as the sampled noisy initialization, *i.e.,* $\boldsymbol{s}_0 = \mathbf{z}_T$. Accordingly, in a rollout of this MDP, the state at each step h is defined as the current noisy latent $\boldsymbol{s}_h \equiv \mathbf{z}_{T-h}$. (iii) *Action*: The action is defined as the per-step predicted clean latent, *i.e.,* $\boldsymbol{a}_h \equiv \hat{\mathbf{x}}_0^{(T-h)}$. Accordingly, we can reinterpret the learned denoiser as a *policy* for this MDP, *i.e.,* $\pi_\theta(\boldsymbol{s}_h, h) \equiv f_\theta(\mathbf{z}_{T-h}, T-h)$. (iv) *Transition kernel*: The state transition satisfies the Markov property and is governed by the ancestral DDPM update in Equation (2). In particular, the transition is stochastic: given the current state and action, the next state is drawn from the Gaussian kernel

$$P(\boldsymbol{s}_{h+1} \mid \boldsymbol{s}_h, \boldsymbol{a}_h) = \mathcal{N}\big(\boldsymbol{s}_{h+1};\, \boldsymbol{\mu}_{T-h}(\boldsymbol{s}_h, \boldsymbol{a}_h),\, \beta_{T-h}\mathbf{I}\big), \tag{3}$$

or equivalently, $\boldsymbol{s}_{h+1} = \boldsymbol{\mu}_{T-h}(\boldsymbol{s}_h, \boldsymbol{a}_h) + \sqrt{\beta_{T-h}}\,\boldsymbol{\xi}_{T-h}$, where $\boldsymbol{\mu}_{T-h}$ and $\beta_{T-h} = 1 - \bar{\alpha}_{T-h}/\bar{\alpha}_{T-h-1}$ follow the definitions in Equation (2), and $\boldsymbol{\xi}_{T-h} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is sampled independently of the history. Hence, the policy itself is deterministic, and all randomness along the denoising trajectory enters through the transition kernel. (v) *Reward function*: The reward signal shall reflect the quality of the decoded text. Since only the final denoised latent $\mathbf{z}_0$ is decoded, this induces a sparse reward MDP where only the terminal states have non-zero rewards. Let r_h denote the reward accrued at time step h. Then, we have $r_h(\boldsymbol{s}_h, \boldsymbol{a}_h) = 0$, for all $h \in \{0, \cdots, T-1\}$, regardless of the visited state and action. (vi) A *discount factor* $\gamma \in [0, 1]$ can be associated with this denoising MDP despite that γ can be one in the finite-horizon setting.

Based on the above finite-horizon MDP formulation, we can define the standard *time-dependent* Q function as follows: For a control policy π,

$$Q^\pi(\boldsymbol{s}, \boldsymbol{a}, h) := \mathbb{E}_\pi \left[\sum_{k=h}^{T-1} \gamma^{k-t} r_k + \gamma^{T-h} r_T \,\middle|\, \boldsymbol{s}_h = \boldsymbol{s},\, \boldsymbol{a}_h = \boldsymbol{a} \right], \quad h \in \{0, \cdots, T\}. \tag{4}$$

The time dependence is needed since the expected return from the same state–action pair can differ depending on the remaining horizon. Given the Markovian nature, the time-dependent Q function satisfies the Bellman equation as follows: For each $h \in \{0, \cdots, T-1\}$,

$$Q^{\pi}(\boldsymbol{s}, \boldsymbol{a}, h) = \mathbb{E}\Big[r_h + \gamma \mathbb{E}_{\boldsymbol{a}' \sim \pi(\boldsymbol{s}_{h+1}, h+1)}\big[Q^{\pi}(\boldsymbol{s}_{h+1}, \boldsymbol{a}', h+1)\big] \,\Big|\, \boldsymbol{s}_h = \boldsymbol{s},\ \boldsymbol{a}_h = \boldsymbol{a}\Big]. \tag{5}$$

Latent Q function as a utility predictor of final decoded text. Recall that reward is assigned only at the terminal state and measures the sample-level utility of the decoded text. Since all intermediate rewards are zero, the Q function at an intermediate denoising step represents the expected discounted utility of the final text, as shown in Equation (4). Thus, the Q-function provides an intermediate prediction of the quality of the text that will eventually be produced by a latent trajectory, without requiring the intermediate latent to be decoded.

For open-ended language generation of LDLMs, common evaluation measures such as MAUVE (Pillutla et al., 2021) and corpus-level diversity measures (Zhu et al., 2018) depend on a set of generated texts and evaluate the output quality relative to a reference corpus, and therefore they do not directly provide rewards for individual denoising transitions. We therefore define the terminal reward using a sample-level density-ratio score between the reference and frozen-policy output distributions:

$$R(\text{text}) := \log \hat{p}_{\text{ref}}\big(c(\text{text})\big) - \log \hat{p}_{\text{model}}\big(c(\text{text})\big), \tag{6}$$

where $c(\text{text})$ maps a decoded text to a fixed semantic cluster, and $\hat{p}_{\text{ref}}$ and $\hat{p}_{\text{model}}$ denote the corresponding cluster distributions under the reference corpus and frozen denoiser policy, respectively. Intuitively, this reward favors texts that occupy semantic regions that are common in the reference corpus but underrepresented by the model. For conditional tasks, where each source has a reference target y, the terminal utility is instead a reference-based task metric, $R_{\text{cond}}(\hat{y}, y) = \frac{1}{4}\big(\text{R-1} + \text{R-2} + \text{R-L} + \text{BERTScore}\big)(\hat{y}, y)$, computed per source so that the N candidates of one source are compared against the same reference. Details are given in Section A.1.

Remark 1 (Sparse rewards) We focus on the sparse-reward setting in which the reward signal is given only by the utility of the final decoded text. This setting reflects the evaluation structure of LDLMs, where generation quality is typically defined on the final output rather than on intermediate noisy latents. For example, in unconditional open-ended generation, MAUVE is a corpus-level, non-decomposable statistic and therefore does not provide a natural state-action level target along the denoising trajectory. Therefore, we reserve MAUVE only for population-level evaluation at inference time. Details of the density-ratio estimator and its validation are provided in Section A.1. Intermediate rewards can in principle be introduced through reward shaping, but doing so would require an additional design choice beyond the underlying text-generation objective.

4.2 Learning the Latent Q-Function: Temporal Difference Learning and Regularization

Given the time-dependent Q-function defined in Section 4.1, we next describe how it is learned for the frozen denoiser policy.

Temporal difference loss. Based on the Bellman equation in Equation (5), we learn the Q function using one-step temporal difference (TD) policy evaluation with function approximation. Specifically, let Q_ϕ denote the network that learns the Q function of the frozen denoiser policy π_θ. Given a replay buffer $\mathcal{D}$ of latent transitions generated by π_θ, we consider the TD loss function as

$$\mathcal{L}_{\text{td}}(\phi) := \mathbb{E}_{(\boldsymbol{s}_h, \boldsymbol{a}_h, r_h, \boldsymbol{s}_{h+1}, \boldsymbol{a}_{h+1}) \sim \mathcal{D}}\Big[\big(r_h + \gamma(1-d_h)\, Q_{\bar{\phi}}(\boldsymbol{s}_{h+1}, \boldsymbol{a}_{h+1}, h+1) - Q_\phi(\boldsymbol{s}_h, \boldsymbol{a}_h, h)\big)^2\Big], \tag{7}$$

where $\bar{\phi}$ denotes the parameters of a Polyak-averaged target network, $d_h \in \{0, 1\}$ indicates whether the transition is terminal, and the continuation action $(\boldsymbol{s}_{h+1}, \boldsymbol{a}_{h+1})$ is generated by the same frozen denoiser policy. Thus, $\mathcal{L}_{\text{td}}$ performs on-policy evaluation rather than a control update. Equation (7) is the mean-squared one-step TD error, with the target network providing a slowly varying bootstrapping target for more stable training.

Regularization of Q-function landscape. While the TD loss enforces temporal consistency along observed denoising trajectories, it does not fully constrain the Q-function landscape needed for

selection at inference time. Specifically, there are two technical issues: (i) Q values at nearby counterfactual actions remain underdetermined. (ii) Accurate pointwise estimation does not necessarily induce the correct relative ordering across latent trajectories. To address these issues, we further introduce two complementary regularization terms as follows.

- **Q-margin ranking loss.** The TD loss constrains the Q values primarily along the observed denoising trajectories and hence can possibly assign unreliable or spuriously high values to nearby counterfactual actions. Motivated by the principle of conservative Q-learning (CQL) in offline RL (Kumar et al., 2020), we introduce a local margin ranking loss that encourages the observed denoising action to score above perturbed alternatives by a *horizon-adjusted margin*. This regularizes the local geometry of the learned Q function and thereby mitigates off-trajectory overestimation. Hence, this design makes the learned Q values more reliable for comparing latent trajectories at inference time. Specifically, for each observed $(\boldsymbol{s}_h, \boldsymbol{a}_h)$, let $\mathcal{A}_{\text{perturb}}(\boldsymbol{a}_h)$ denote a set of locally perturbed actions with respect to $\boldsymbol{a}_h$. We define the Q-margin ranking loss as

$$\mathcal{L}_{\text{margin}}(\phi) := \mathbb{E}_{(\boldsymbol{s}_h, \boldsymbol{a}_h, h)\sim\mathcal{D}}\left[\frac{1}{|\mathcal{A}_{\text{perturb}}(\boldsymbol{a}_h)|}\sum_{\boldsymbol{a}'\in\mathcal{A}_{\text{perturb}}}\left[\delta\gamma^{T-h} - \big(Q_\phi(\boldsymbol{s}_h, \boldsymbol{a}_h, h) - Q_\phi(\boldsymbol{s}_h, \boldsymbol{a}', h)\big)\right]^+\right], \tag{8}$$

  where $[x]^+ := \max(0, x)$ and $\delta > 0$ is the base margin. The factor γ^{T-h} scales the margin according to the remaining denoising horizon, consistent with the discounted scale of the terminal return. The construction of the perturbed actions and the associated hyperparameters are provided in Section A.2.

- **Cross-trajectory Q-selection loss.** While the TD objective estimates the return of each individual latent trajectory, multi-trajectory latent generation requires comparing trajectories and determining which ones to retain, as described in Section 3. We therefore introduce a cross-trajectory Q-selection loss that encourages the Q-function to assign the highest intermediate score to the trajectory that ultimately achieves the highest terminal utility. Consider a group of N latent trajectories evaluated at the same decision index h, and let $i^\star := \arg\max_{i\in[N]} R(\text{text}^{(i)})$ denote the trajectory with the highest terminal utility. We define the Q-selection loss as

$$\mathcal{L}_{\text{selection}}(\phi) := -\mathbb{E}\left[\frac{1}{T}\sum_{h=0}^{T-1}\left(\frac{1}{\alpha}Q_\phi(\boldsymbol{s}_h^{(i^\star)}, \boldsymbol{a}_h^{(i^\star)}, h) - \log\sum_{i=1}^{N}\exp\Big(\frac{1}{\alpha}Q_\phi(\boldsymbol{s}_h^{(i)}, \boldsymbol{a}_h^{(i)}, h)\Big)\right)\right], \tag{9}$$

  where $\{(\boldsymbol{s}_h^{(i)}, \boldsymbol{a}_h^{(i)})\}_{i=1}^N$ are the observed state-action pairs from N distinct trajectories at decision index h, and α is a fixed logit temperature. By forming each comparison across trajectories at the same denoising step, this objective directly aligns the learned Q-scores with the trajectory selection performed at inference time, which we formalize in Section 4.3.

Putting everything together. To sum up, we train Q_ϕ on latent reverse-diffusion trajectories generated by the frozen denoiser using the following objective function, which addresses both temporal credit assignment and regularization of the latent landscape:

$$\mathcal{L}(\phi) = \lambda_{\text{td}}\,\mathcal{L}_{\text{td}}(\phi) + \lambda_{\text{margin}}\,\mathcal{L}_{\text{margin}}(\phi) + \lambda_{\text{selection}}\,\mathcal{L}_{\text{selection}}(\phi). \tag{10}$$

Here, $\mathcal{L}_{\text{td}}(\phi)$ propagates the terminal utility backward along each denoising trajectory, $\mathcal{L}_{\text{margin}}(\phi)$ regularizes the local Q-landscape around the observed actions, and $\mathcal{L}_{\text{selection}}(\phi)$ leverages relative comparisons across trajectories as auxiliary supervision. Together, these complementary objectives produce intermediate Q-scores that are both temporally consistent and suitable for inference-time trajectory selection.

Remark 2 (Policy evaluation) The TD learning considered here is used solely for evaluating the frozen denoiser policy π_θ, *i.e.,* policy evaluation or critic learning in the context of RL. Given a trained LDLM, we freeze its denoiser and regard it as a fixed policy π_θ, and there is no policy improvement step performed. Indeed, the TD target in Equation (7) contains neither a maximization over actions (*e.g.,* Q-learning) nor a gradient propagated into the denoiser (*e.g.,* policy gradient methods). The learning of Q_ϕ does not change the underlying denoising policy.

Remark 3 (Diffusion samplers) Our formulation presumes the stochastic DDPM update (Ho et al., 2020). Notably, the TD loss and auxiliary regularization losses are *not* specific to DDPM and extend

directly to stochastic samplers, such as DDIM (Song et al., 2021a), reverse-time SDE samplers (Song et al., 2021b), and related diffusion sampling schemes (Karras et al., 2022). Under the deterministic DDIM, each state-action pair would uniquely determine the next latent. With a stochastic sampler, the transition kernel integrates over newly sampled noise, and the learned Q-function predicts the expected terminal utility over the future randomness. Therefore, the same TD update and ranking objectives are applicable for both deterministic and stochastic samplers.

4.3 Latent Q-Guided Generation and Trajectory Pruning

We now describe how LQG uses the learned latent Q-function to reduce the inference cost of multi-trajectory generation in LDLMs.

n**Latent Q-guided generation without pruning.** Consider the multi-trajectory generation setting introduced in Section 3. We initialize N latent trajectories with $\mathbf{z}_T^{(1)}, \ldots, \mathbf{z}_T^{(N)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and denoise them in parallel for T steps, yielding terminal latents $\mathbf{z}_0^{(1)}, \ldots, \mathbf{z}_0^{(N)}$. Since Q_ϕ is trained to predict terminal utility from intermediate state-action pairs, we select the trajectory with the largest Q-score at the final decision step:

$$i^\star \leftarrow \arg\max_i Q_\phi\big(\mathbf{z}_0^{(i)}, \boldsymbol{a}^{(i)}, T\big), \qquad (11)$$

where $\boldsymbol{a}^{(i)} \leftarrow \pi_\theta(\mathbf{z}_0^{(i)}, 0)$. Only the selected terminal latent $\mathbf{z}_0^{(i^\star)}$ is then forwarded to the autoregressive decoder. Consequently, LQG can exploit multiple denoising trajectories while incurring the decoding cost of only a single candidate, rather than decoding all N candidates before selection. The overall procedure is summarized in Algorithm 1.

Algorithm 1 Latent Q-Guided Generation

Require: Frozen denoiser π_θ, latent Q-function Q_ϕ, branching factor N, and horizon T

for $i = 1, \ldots, N$ **do**
  $\boldsymbol{z}^{(i)} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
  for $h = 0, \ldots, T-1$ **do**
    $\boldsymbol{a}^{(i)} \leftarrow \pi_\theta(\boldsymbol{z}^{(i)}, T-h)$
    $\boldsymbol{z}^{(i)} \leftarrow \textsc{DDPMStep}(\boldsymbol{z}^{(i)}, \boldsymbol{a}^{(i)}, T-h)$
  end for
  $\boldsymbol{a}^{(i)} \leftarrow \pi_\theta(\boldsymbol{z}^{(i)}, 0)$
end for
$i^\star \leftarrow \arg\max_{i \in [N]} Q_\phi(\boldsymbol{z}^{(i)}, \boldsymbol{a}^{(i)}, T)$
return $\textsc{Decoder}(\boldsymbol{z}^{(i^\star)})$

Latent Q-guided generation with pruning. Since Q_ϕ estimates terminal utility throughout the denoising process, it can also be used to identify and discard unpromising trajectories before they are fully denoised. Specifically, at a prescribed decision index $h^\star < T$, LQG selects $i^\star \leftarrow \arg\max_i Q_\phi\big(\mathbf{z}_{T-h}^{(i)}, \boldsymbol{a}^{(i)}, h^*\big)$, where $\boldsymbol{a}^{(i)} \leftarrow \pi_\theta(\mathbf{z}_{T-h^*}^{(i)}, T-h^*)$. The remaining $N-1$ trajectories are pruned, and only the selected trajectory $i^\star$ is continued for the remaining $T-h^\star$ denoising steps to obtain $\mathbf{z}_0^{(i^\star)}$. The resulting latent is then decoded into text. In this way, LQG reduces both denoising and decoding costs. In practice, one can choose $h^\star$ based on the following tradeoff: earlier pruning yields greater computational savings, while later pruning provides the Q-function with more informative, less noisy latent states for trajectory selection.

Remark 4 (Computational speedup) With N trajectories, a denoising horizon of T, and pruning after decision index h^*, LQG requires $C_{\text{LQG}}(T, N, h^*) = N(h^*+1) + (T-h^*-1) = T + (N-1)(h^*+1)$ denoiser calls, compared with NT calls for terminal Best-of-N. For example, under $T = 250$, $N = 4$, and $h^* = 48$, LQG empirically achieves a 60.3% reduction in denoiser calls. In practice, compared to vanilla terminal Best-of-N sampling, this yields a 53.7% reduction (*i.e.,* 2.16× speedup) in sampling-and-scoring time and a 46.7% reduction (*i.e.,* 1.88× speedup) in end-to-end latency, including text decoding.

5 Experiments

To evaluate the proposed LQG, we empirically compare it against state-of-the-art latent-space and output-dependent guidance baselines. We test whether LQG effectively guides test-time search over denoised LDLM trajectories to improve generation quality over the base sampler, outperform supervised latent critics and existing test-time methods, and reduce overall denoising cost.

Method	ROCStories MAUVE↑	ROCStories Ppl↓	ROCStories Div↑	ROCStories Mem↓	AG News MAUVE↑	AG News Ppl↓	AG News Div↑	AG News Mem↓
Base denoiser (N=1)	.731$_{\pm.006}$	29.8$_{\pm0.7}$	.352$_{\pm.003}$	.433$_{\pm.002}$	.892$_{\pm.014}$	**79.2**$_{\pm3.1}$	.514$_{\pm.007}$	.341$_{\pm.005}$
Output-dependent								
SWIFT	.852$_{\pm.017}$	30.2$_{\pm.9}$	.349$_{\pm.0003}$	**.432**$_{\pm.001}$	.858$_{\pm.015}$	91.9$_{\pm6.8}$	.558$_{\pm.006}$	.295$_{\pm.005}$
TTSnap	.667$_{\pm.079}$	29.2$_{\pm0.5}$	.326$_{\pm.001}$	.460$_{\pm.009}$	.396$_{\pm.051}$	124.1$_{\pm3.4}$	.580$_{\pm.010}$	**.243**$_{\pm.005}$
TrajSelector	.634$_{\pm.020}$	**28.6**$_{\pm0.24}$	**.433**$_{\pm.002}$	.438$_{\pm.001}$	.768$_{\pm.01}$	90.7$_{\pm2.3}$	**.674**$_{\pm.005}$	.291$_{\pm.001}$
Latent-space								
latent-SL BoN	.886$_{\pm.011}$	28.7$_{\pm0.6}$	.336$_{\pm.005}$	.454$_{\pm.002}$	.818$_{\pm.027}$	103.1$_{\pm4.4}$	.556$_{\pm.014}$	.295$_{\pm.006}$
latent-SL PSP	.885$_{\pm.008}$	28.9$_{\pm0.2}$	.344$_{\pm.003}$	.439$_{\pm.005}$	.861$_{\pm.002}$	80.3$_{\pm3.8}$	.525$_{\pm.001}$	.321$_{\pm.005}$
latent-SL FK	.873$_{\pm.013}$	**28.6**$_{\pm0.2}$	.335$_{\pm.001}$	.453$_{\pm.003}$	.887$_{\pm.017}$	85.5$_{\pm1.1}$	.540$_{\pm.010}$	.312$_{\pm.011}$
LQG (Ours)	**.893**$_{\pm.011}$	29.7$_{\pm0.06}$	.348$_{\pm.003}$	.440$_{\pm.001}$	**.919**$_{\pm.006}$	91.3$_{\pm5.6}$	.560$_{\pm.006}$	.307$_{\pm.007}$

Method	XSum R-1↑	XSum R-2↑	XSum R-L↑	XSum BERTScore↑	QQP R-1↑	QQP R-2↑	QQP R-L↑	QQP BERTScore↑
Base denoiser (N=1)	35.55$_{\pm0.14}$	13.85$_{\pm0.12}$	28.72$_{\pm0.05}$	39.85$_{\pm0.08}$	63.97$_{\pm0.13}$	40.25$_{\pm0.18}$	61.29$_{\pm0.13}$	68.86$_{\pm0.10}$
Output-dependent								
SWIFT	34.21$_{\pm0.24}$	12.62$_{\pm0.11}$	27.39$_{\pm0.12}$	37.52$_{\pm0.25}$	62.84$_{\pm0.10}$	39.60$_{\pm0.20}$	60.35$_{\pm0.10}$	68.10$_{\pm0.07}$
TTSnap	35.87$_{\pm0.16}$	13.83$_{\pm0.09}$	28.86$_{\pm0.14}$	40.32$_{\pm0.11}$	63.86$_{\pm0.16}$	39.80$_{\pm0.37}$	61.25$_{\pm0.25}$	69.01$_{\pm0.12}$
TrajSelector	34.46$_{\pm0.03}$	13.61$_{\pm0.05}$	28.01$_{\pm0.07}$	43.33$_{\pm0.06}$	60.93$_{\pm0.35}$	37.47$_{\pm0.41}$	58.14$_{\pm0.36}$	**69.24**$_{\pm0.20}$
Latent-space								
latent-SL BoN	37.91$_{\pm0.17}$	15.19$_{\pm0.13}$	30.37$_{\pm0.07}$	42.27$_{\pm0.12}$	64.02$_{\pm0.17}$	40.46$_{\pm0.19}$	61.37$_{\pm0.19}$	69.05$_{\pm0.04}$
latent-SL PSP	37.86$_{\pm0.22}$	15.11$_{\pm0.19}$	30.30$_{\pm0.14}$	42.17$_{\pm0.11}$	63.97$_{\pm0.12}$	40.13$_{\pm0.22}$	61.32$_{\pm0.10}$	69.11$_{\pm0.10}$
latent-SL FK	37.79$_{\pm0.29}$	15.14$_{\pm0.04}$	30.33$_{\pm0.25}$	42.16$_{\pm0.14}$	**64.29**$_{\pm0.17}$	**40.76**$_{\pm0.15}$	**61.60**$_{\pm0.12}$	69.21$_{\pm0.14}$
LQG (Ours)	**38.45**$_{\pm0.09}$	**15.34**$_{\pm0.16}$	**31.08**$_{\pm0.08}$	**43.40**$_{\pm0.12}$	64.04$_{\pm0.14}$	40.47$_{\pm0.11}$	61.37$_{\pm0.19}$	69.00$_{\pm0.05}$

Table 1: Main results on two unconditional (top) and two conditional (bottom) tasks. All methods share the same frozen denoiser, 250-step sampler, and N=4 candidates (except the N=1 base). LQG uses hierarchical selection with W=16, M=4, and β=2, except on AG News, where W=8 and M=2. Entries are mean$_{\pm\text{std}}$ over three generation seeds of 1,000 outputs; bold marks the best value in each column; LQG rows are ours. ROUGE (R-1/2/L) and BERTScore are on a 0–100 scale. Calls is the number of denoiser forward passes per output, including the classifier-free-guidance pass on the conditional tasks.

5.1 Experimental Setup

Tasks and Base Models. Following LD4LG (Lovelace et al., 2023), we consider unconditional generation on **ROCStories** (Mostafazadeh et al., 2016) and **AG News** (Zhang et al., 2015), and conditional generation on **XSum** (Narayan et al., 2018) and **QQP** (Chen et al., 2017). All tasks use L=32 latent tokens of dimension D=32 and a frozen denoiser sampled for T=250 DDPM steps. Conditional models use a frozen BART-base context encoder and classifier-free guidance with scale 2. Within each task, all methods share the same denoiser, autoencoder, and decoder, differing only in candidate selection.

Terminal Utility. For unconditional generation, R is the density-ratio utility in Equation (6), estimated over 50 k-means clusters of GPT-2-large features using held-out reference texts and 6,000 frozen-policy samples. For conditional generation, R averages ROUGE-1, ROUGE-2, ROUGE-L, and BERTScore-F1 against the training target. Rewards are standardized using training-set statistics.

Search Operators. For unconditional generation, we report MAUVE (Pillutla et al., 2021), GPT-2-large perplexity, diversity, and training-set 4-gram memorization following LD4LG (Lovelace et al., 2023). For conditional generation, we report ROUGE (Lin, 2004) and BERTScore (Zhang et al., 2020) on a 0–100 scale. Each run produces 1,000 outputs, and we report the mean and sample standard deviation over three generation seeds. Cost is measured by denoiser forward passes per output, including the additional classifier-free-guidance pass for conditional tasks.

Baselines. The *base denoiser* generates one sample. *latent-SL* replaces Q_ϕ with an MLP trained by Monte-Carlo regression on the same terminal utility and is combined with BoN, PSP, and FK. We further port three output-dependent methods to our latent denoisers: *SWIFT* (Guo et al., 2026b) and *TrajSelector* (Yu et al., 2025a) select among completed candidates using internal representations, whereas *TTSnap* (Yu et al., 2025b) prunes candidates using a reward model trained on noisy intermediate states.

Metrics and Protocol. For the unconditional tasks, we report MAUVE (Pillutla et al., 2021) against the first 1,000 validation texts (GPT-2-large features, averaged over 10 clustering seeds), GPT-2-large perplexity (Ppl), diversity (Div), and 4-gram memorization against the training set (Mem), following LD4LG (Lovelace et al., 2023). For the conditional tasks, we report ROUGE (Lin, 2004) and BERTScore (Zhang et al., 2020) against the validation references, on a 0–100 scale. Each run generates 1,000 outputs; we report mean$_{\pm\text{std}}$ over three generation seeds (sample standard deviation). Cost is the number of denoiser forward passes per output. For the conditional tasks, it includes the extra classifier-free-guidance pass, which doubles the per-step count.

5.2 Experimental results

Unconditional Generation. On ROCStories, all pre-decoding methods substantially improve over the base sampler (MAUVE .731), with LQG achieving the highest score (.893). Its advantage over latent-SL BoN (.886) is within one standard deviation, but latent-SL attains its gains with lower diversity and higher memorization, suggesting a preference for safer, more training-like text. In contrast, LQG remains close to the base on perplexity, diversity, and memorization. On AG News, where the base is already strong (.892), LQG is the only method to improve MAUVE (.919), while also increasing diversity and reducing memorization. SWIFT is competitive only on ROCStories, whereas TTSnap and TrajSelector underperform the base on both tasks. The secondary metrics like perplexity and memorization is expected, as the terminal utility in our Q-learning objective is implicitly designed to optimize MAUVE rather than targeting individual lexical statistics.

Conditional Generation. On XSum, all pre-decoding methods improve substantially over the base, with LQG performing best across ROUGE and BERTScore (38.45/15.34/31.08/43.40). LQG consistently outperforms latent-SL BoN, despite that their ROUGE-2 difference is within one standard deviation. TrajSelector attains a comparable BERTScore but falls below the base on all ROUGE metrics. On the near-saturated QQP task, LQG and latent-SL BoN perform nearly identically and slightly outperform the base, while SWIFT degrades performance and TTSnap remains roughly comparable to the base.

Stage	MAUVE↑	Div.↑
TD only	$.7214_{\pm.0148}$	$.3497_{\pm.0005}$
$+\mathcal{L}_{\text{selection}}$	$.8730_{\pm.0231}$	$.3518_{\pm.0035}$
$+\mathcal{L}_{\text{margin}}$	$\mathbf{.8935}_{\pm.0074}$	$\mathbf{.3567}_{\pm.0036}$

Table 2: Cumulative objective ablation on ROCStories. Each stage adds one term to the previous stage, and all stages rank the same $N{=}4$ candidate pools with hierarchical selection over the last 16 steps ($M{=}4$, $\beta{=}2$). Mean$_{\pm\text{std}}$ over three generation seeds.

Ablations on Training Loss. Table 2 adds the objective terms one at a time and ranks the same candidate pools with each resulting Q-function, so every difference is due to the Q-function alone. QT trained with TD loss alone achieves MAUVE .721, which is comparable to the base denoiser (.731). Adding $\mathcal{L}_{\text{selection}}$ increases MAUVE to .873, because it directly trains the cross-trajectory comparisons required by test-time selection. Adding $\mathcal{L}_{\text{margin}}$ further improves MAUVE to .894 and reduces variation across seeds. On XSum (Table 3), the $\mathcal{L}_{\text{selection}}$ term has a negligible effect, whereas the margin term $\mathcal{L}_{\text{margin}}$ improves ROUGE-1, ROUGE-L, and BERTScore. More experimental results on the end-to-end inference cost and examples of output responses are in Appendix A.

Stage	R-1↑	R-2↑	R-L↑	BERTScore↑
TD only	$36.52_{\pm0.05}$	$15.02_{\pm0.22}$	$29.69_{\pm0.09}$	$42.20_{\pm0.10}$
$+\mathcal{L}_{\text{selection}}$	$36.52_{\pm0.06}$	$15.06_{\pm0.22}$	$29.67_{\pm0.12}$	$42.20_{\pm0.10}$
$+\mathcal{L}_{\text{margin}}$	$\mathbf{38.37}_{\pm0.09}$	$\mathbf{15.20}_{\pm0.21}$	$\mathbf{31.03}_{\pm0.18}$	$\mathbf{43.36}_{\pm0.18}$

Table 3: Cumulative objective ablation on XSum (ROUGE and BERTScore on a 0–100 scale).

6 Concluding Remarks

We introduced Latent Q-Guided Generation (LQG), which formulates latent denoising as a finite-horizon MDP and learns a latent Q-function through TD policy evaluation of a frozen denoiser. LQG improves terminal utility while reducing multi-candidate inference cost through latent-space selection and pruning. Our results highlight the importance of cross-trajectory and margin regularization and motivate adaptive pruning, early-horizon prediction, and extensions to other diffusion models.

AI USE STATEMENT

We used a code editor with generative AI functionalities during code development and paper writing. In writing, these tools were limited to LaTeX code completion and minor text editing, without generating substantive technical content. We also used AI assistants to create single-use data processing scripts and to assist in generating figures. All AI-generated artifacts were carefully reviewed, verified, and approved by the authors prior to inclusion.

ETHICS STATEMENT

This work improves the efficiency of inference in latent diffusion language models and does not introduce new capabilities beyond those of the underlying pretrained models. Nevertheless, generated text may inherit biases, inaccuracies, or harmful content from these models and their training data. Therefore, deployment should include appropriate safeguards, monitoring, and human oversight.

REPRODUCIBILITY STATEMENT

We describe the LQG formulation, training objectives, and inference procedure in Section 4, and provide the experimental setup and evaluation protocol in Section 5.1. Appendices A detail the terminal utilities, critic training objective and architecture, trajectory-pruning procedure, latent representation, and hyperparameters. Appendices B provide the dataset configurations and implementation details for all external and supervised baselines. We will make the source code and experiment configurations publicly available upon acceptance.

REFERENCES

Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training Diffusion Models with Reinforcement Learning. In *International Conference on Learning Representations (ICLR)*, 2024.

Zihang Chen, Hongbo Zhang, Xiaoji Zhang, and Le Zhao. Quora question pairs. 2017. URL https://api.semanticscholar.org/CorpusID:233225749.

Yilun Du, Shuang Li, Joshua Tenenbaum, and Igor Mordatch. Learning Iterative Reasoning through Energy Minimization. In *International Conference on Machine Learning (ICML)*, 2022.

Yilun Du, Jiayuan Mao, and Joshua B. Tenenbaum. Learning Iterative Reasoning through Energy Diffusion. In *International Conference on Machine Learning (ICML)*, 2024.

Ying Fan, Olivia Watkins, Yuqing Du, Hao Liu, Moonkyung Ryu, Craig Boutilier, Pieter Abbeel, Mohammad Ghavamzadeh, Kangwook Lee, and Kimin Lee. DPOK: Reinforcement Learning for Fine-tuning Text-to-Image Diffusion Models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

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 *Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2024.

Shansan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and Lingpeng Kong. Diffuseq-v2: Bridging discrete and continuous text spaces for accelerated seq2seq diffusion models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 9868–9875, 2023.

Rogerio Guimaraes and Pietro Perona. Inference-time scaling of diffusion models via progressive seed pruning. *arXiv preprint arXiv:2607.21591*, 2026a.

Rogerio Guimaraes and Pietro Perona. Inference-time scaling of diffusion models via progressive seed pruning. 2026b.

Hongcan Guo, Qinyu Zhao, Yian Zhao, Shen Nie, Rui Zhu, Qiushan Guo, Feng Wang, Tao Yang, Hengshuang Zhao, Guoqiang Wei, and Yan Zeng. Cola: Continuous latent diffusion language model. *arXiv preprint arXiv:2605.06548*, 2026a.

Jizhou Guo, Zhaomin Wu, Hanchen Yang, and Philip S Yu. Mining intrinsic rewards from llm hidden states for efficient best-of-n sampling. In *Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 371–382, 2026b.

Seungwook Han, Idan Shenfeld, Akash Srivastava, Yoon Kim, and Pulkit Agrawal. Value augmented sampling: Predict your rewards to align language models. In *ICLR 2024 Workshop on Reliable and Responsible Foundation Models*.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

Haoqiang Kang, Yizhe Zhang, Nikki Kuang, Nicklas Majamaki, Navdeep Jaitly, Yian Ma, and Lianhui Qin. LaDiR: Latent diffusion enhances LLMs for text reasoning. In *International Conference on Learning Representations*, volume 2026, pp. 81705–81733, 2026.

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.

Theodoros Kouzelis, Ioannis Kakogeorgiou, Spyros Gidaris, and Nikos Komodakis. EQ-VAE: Equivariance regularized latent space for improved generative image modeling. In *International Conference on Machine Learning*, pp. 31648–31666, 2025.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative Q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191, 2020.

Hengli Li, Chenxi Li, Tong Wu, Xuekai Zhu, Yuxuan Wang, Zhaoxin Yu, Eric Hanchen Jiang, Song-Chun Zhu, Zixia Jia, Ying Nian Wu, et al. Seek in the dark: Reasoning via test-time instance-level policy gradient in latent space. *arXiv preprint arXiv:2505.13308*, 2025.

Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori B. Hashimoto. Diffusion-LM Improves Controllable Text Generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Xiner Li, Yulai Zhao, Chenyu Wang, Gabriele Scalia, Gokcen Eraslan, Surag Nair, Tommaso Biancalani, Aviv Regev, Sergey Levine, and Masatoshi Uehara. Diffusion model alignment via reward-guided tree search. *arXiv preprint arXiv:2408.08252*, 2024.

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

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 *International Conference on Machine Learning (ICML)*, 2023.

Zhenhua Liu, Lijun Li, Ruizhe Chen, Yuxian Jiang, Tong Zhu, Zhaochen Su, Wenliang Chen, and Jing Shao. Evolutionary guided decoding: Iterative value refinement for llms. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*, pp. 3266–3283, 2026.

Justin Lovelace, Varsha Kishore, Chao Wan, Eliot Shekhtman, and Kilian Q. Weinberger. Latent Diffusion for Language Generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Justin Lovelace, Varsha Kishore, Yiwei Chen, and Kilian Q. Weinberger. Diffusion guided language modeling. *arXiv preprint arXiv:2408.04220*, 2024.

Viacheslav Meshchaninov, Egor Chimbulatov, Alexander Shabalin, Aleksandr Abramov, and Dmitry Vetrov. COSMOS: Compressed and smooth latent space for text diffusion modeling. *Advances in Neural Information Processing Systems*, 38:14271–14299, 2025.

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.

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 *Proceedings of the 2018 conference on empirical methods in natural language processing*, pp. 1797–1807, 2018.

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. *Advances in Neural Information Processing Systems*, 34:4816–4828, 2021.

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. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2024.

Raghav Singhal, Zachary Horvitz, Ryan Teehan, Mengye Ren, Zhou Yu, Kathleen McKeown, and Rajesh Ranganath. A general framework for inference-time scaling and steering of diffusion models. *arXiv preprint arXiv:2501.06848*, 2025.

Ivan Skorokhodov, Sharath Girish, Benran Hu, Willi Menapace, Yanyu Li, Rameen Abdal, Sergey Tulyakov, and Aliaksandr Siarohin. Improving the diffusability of autoencoders. In *International Conference on Machine Learning*, pp. 55876–55905, 2025.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations (ICLR)*, 2021a.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-Based Generative Modeling through Stochastic Differential Equations. In *International Conference on Learning Representations (ICLR)*, 2021b.

Robin Strudel, Corentin Tallec, Florent Altché, Yilun Du, Yaroslav Ganin, Arthur Mensch, Will Grathwohl, Nikolay Savinov, Sander Dieleman, Laurent Sifre, and Rémi Leblond. Self-conditioned embedding diffusion for text generation. *arXiv preprint arXiv:2211.04236*, 2022.

Shaobo Wang, Guo Chen, Ziyue Wang, Zhengyang Tang, Qingyang Liu, Xingzhang Ren, Dayiheng Liu, and Linfeng Zhang. The missing piece in pre-trained model evaluation: Reward-guided decoding unlocks task-oriented behavior without parameter updates. *arXiv preprint arXiv:2605.28020*, 2026.

Minkai Xu, Tomas Geffner, Karsten Kreis, Weili Nie, Yilun Xu, Jure Leskovec, Stefano Ermon, and Arash Vahdat. Energy-Based Diffusion Language Models for Text Generation. In *International Conference on Learning Representations (ICLR)*, 2025.

Bin Yu, Xinming Wang, Shijie Lian, Haotian Li, Changti Wu, Ruina Hu, Bailing Wang, Yuliang Wei, and Kai Chen. TrajSelector: Harnessing latent representations for efficient and effective best-of-N in large reasoning model. *arXiv preprint arXiv:2510.16449*, 2025a.

Qingtao Yu, Changlin Song, Minghao Sun, Zhengyang Yu, Vinay Kumar Verma, Soumya Roy, Sumit Negi, Hongdong Li, and Dylan Campbell. TTSnap: Test-time scaling of diffusion models via noise-aware pruning. *arXiv preprint arXiv:2511.22242*, 2025b.

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.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, volume 28, pp. 649–657, 2015.

Yizhe Zhang, Jiatao Gu, Zhuofeng Wu, Shuangfei Zhai, Joshua M. Susskind, and Navdeep Jaitly. PLANNER: Generating Diversified Paragraph via Latent Language Diffusion Model. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Xiaochen Zhu, Georgi Karadzhov, Chenxi Whitehouse, and Andreas Vlachos. Segment-level diffusion: A framework for controllable long-form generation with diffusion language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, pp. 4163–4183, 2025.

Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. Texygen: A benchmarking platform for text generation models. In *ACM SIGIR Conference on Research & Development in Information Retrieval*, pp. 1097–1100, 2018.

A Detailed Implementation and Additional Experimental Results

A.1 Terminal utilities

All utilities are assigned only at the terminal transition ($r_h = 0$ for every non-terminal transition) and are z-scored with training-population statistics before entering the bootstrap and the selection targets. The unconditional tasks use a reference-free density-ratio score; the conditional tasks use a reference-based task metric.

Density-ratio utility (unconditional tasks) To construct the terminal reward in Equation (6), we embed each decoded text using GPT-2-large and assign it to one of $K = 50$ semantic clusters through a fixed mapping $c(\cdot)$. The cluster centroids are obtained by fitting k-means once on embeddings from the reference corpus. We then estimate two categorical distributions over these clusters: $\hat{p}_{\mathrm{ref}}$, using texts from the reference corpus, and $\hat{p}_{\mathrm{model}}$, using candidates generated by the frozen denoising policy, and define

$$S(x) = \log \hat{p}_{\mathrm{ref}}\big(c(x)\big) - \log \hat{p}_{\mathrm{model}}\big(c(x)\big). \tag{12}$$

Both histograms are Laplace-smoothed and are frozen before Q-function training. Consequently, computing the reward of a generated text requires only feature extraction, cluster assignment, and a table lookup; neither distribution is re-estimated during training.

In our experiments, the centroids and $\hat{p}_{\mathrm{ref}}$ are estimated from the 500 texts of the ROCStories validation split. We estimate $\hat{p}_{\mathrm{model}}$ from 6,000 frozen-policy candidates, comprising three generation seeds, 500 candidate groups per seed, and $N = 4$ trajectories per group.

Reference-based utility (conditional tasks). For the conditional tasks (QQP and XSum), each candidate is scored against the reference target of its source:

$$R_{\mathrm{cond}}(\hat{y}, y) = \tfrac{1}{4}\Big(\text{ROUGE-1}(\hat{y}, y) + \text{ROUGE-2}(\hat{y}, y) + \text{ROUGE-L}(\hat{y}, y) + \text{BERTScore-F1}(\hat{y}, y)\Big), \tag{13}$$

where BERTScore uses `roberta-base` without baseline rescaling. Because the score depends on the (prediction, reference) pair, rewards are computed per source: the $N = 4$ candidates generated for one source are all scored against that source's reference.

A.2 Training objective

In Equation (10) we use $(\lambda_{\mathrm{td}}, \lambda_{\mathrm{margin}}, \lambda_{\mathrm{selection}}) = (1,\ 0.03,\ 0.1)$.

Fixed-policy TD. We first generate complete reverse-diffusion trajectories using the frozen denoiser π_θ and store transitions $(\boldsymbol{s}_h, \boldsymbol{a}_h, r_h, \boldsymbol{s}_{h+1}, \boldsymbol{a}_{h+1}, d_h)$ in the replay buffer. Both $\boldsymbol{a}_h$ and the continuation action $\boldsymbol{a}_{h+1} = \pi_\theta(\boldsymbol{s}_{h+1}, h+1)$ are actions actually produced by the frozen denoiser. Accordingly, the bootstrap target is

$$y_h = r_h + \gamma(1 - d_h) Q_{\bar{\phi}}\left(\boldsymbol{s}_{h+1}, \boldsymbol{a}_{h+1}, h+1\right), \tag{14}$$

which is a one-step SARSA-style policy-evaluation target and contains no $\max_{\boldsymbol{a}'} Q_{\bar{\phi}}(\boldsymbol{s}_{h+1}, \boldsymbol{a}', h+1)$ operation. During optimization, gradients update only the critic parameters ϕ, while the denoiser parameters θ, stored actions, and bootstrap targets are held fixed; the target parameters $\bar{\phi}$ are updated through Polyak averaging. Transitions are drawn from a timestep-balanced prioritized experience replay (PER) buffer that first samples a timestep uniformly among active timesteps and then samples by priority within it. The margin and selection losses likewise shape only the critic: they do not update the denoiser or propose replacement actions.

Implementation of the Q-selection loss. Each group in Equation (9) consists of $N = 4$ distinct trajectories observed at the same decision index h (for conditional tasks, the N candidates generated for one source), with z-scored terminal utilities $R^{(1)}, \dots, R^{(N)}$. In the deployed configuration the one-hot target on $i^\star$ is replaced by its soft counterpart,

$$p_i = \frac{\exp(R^{(i)}/\tau_r)}{\sum_{j=1}^{N} \exp(R^{(j)}/\tau_r)}, \qquad \mathcal{L}_{\text{selection}} = -\mathbb{E}\left[\sum_{i=1}^{N} p_i \log \frac{\exp\big(Q_\phi(\boldsymbol{s}_h^{(i)}, \boldsymbol{a}_h^{(i)}, h)/\alpha\big)}{\sum_{j=1}^{N} \exp\big(Q_\phi(\boldsymbol{s}_h^{(j)}, \boldsymbol{a}_h^{(j)}, h)/\alpha\big)}\right], \tag{15}$$

which recovers Equation (9) as $\tau_r \to 0$. We use $\alpha = \tau_r = 1$ and 16 groups per update, sampled uniformly over decision indices. All candidates of a group are scored in one forward pass.

Implementation of the Q-margin loss. For every observed state–action pair $(\boldsymbol{s}_h, \boldsymbol{a}_h)$, the perturbation set $\mathcal{A}_{\text{perturb}}(\boldsymbol{a}_h)$ in Equation (8) contains $M = 5$ counterfactual actions in a small neighborhood of $\boldsymbol{a}_h$: one isotropic perturbation, three orthogonal perturbations at scales 0.2, 0.5, and 1.0, and one perturbation along a randomly sampled direction,

$$\mathcal{A}_{\text{perturb}}(\boldsymbol{a}_h) = \left\{\boldsymbol{a}'_{\text{iso}}, \boldsymbol{a}'_{\text{orth},0.2}, \boldsymbol{a}'_{\text{orth},0.5}, \boldsymbol{a}'_{\text{orth},1.0}, \boldsymbol{a}'_{\text{rand}}\right\}. \tag{16}$$

Each perturbed action is rescaled token-wise to the norm of $\boldsymbol{a}_h$, so that it lies on the same product of spheres and differs from the observed action only in direction. We use base margin $\delta = 0.25$, and all five perturbations share the same margin.

A.3 Network architecture of the latent Q function.

Q_ϕ is a Transformer encoder over the concatenation of state tokens, action tokens, and a single time token,

$$\left[s_1, \dots, s_L,\ a_1, \dots, a_L,\ t_{\text{tok}}\right] \in \mathbb{R}^{(2L+1)\times d_{\text{model}}}, \qquad t_{\text{tok}} = \text{MLP}(\text{FourierEmb}(t)),$$

with a learned input projection $\mathbb{R}^D \to \mathbb{R}^{d_{\text{model}}}$ that decouples critic width from the latent dimension. Learned type embeddings distinguish state, action, and time tokens, and a positional embedding is shared between s_i and a_i so the network can exploit the token-aligned $s_i \to a_i$ correspondence. The output is a LayerNorm-ed pool biased toward action tokens, projected to a scalar. We use d_{model}=256, 8 layers, 8 attention heads, and no dropout.

Conditional critic. For the conditional tasks, the source sequence is encoded by a frozen BART-base encoder; its hidden states are mean-pooled into a single prompt token, linearly projected to d_{model}, tagged with a fourth type embedding, and appended to the sequence, giving $2L+2$ input tokens. The encoder receives no gradient.

Numerical precision. Scoring and ranking are performed in FP32, since within-pool Q differences can be of order 10^{-4}.

A.4 Inference: trajectory pruning

At deployment we draw $N = 4$ independent trajectories under the frozen denoiser with the ancestral (DDPM) sampler, advance all of them through decision index $h^\star = 48$, score each state–action pair once with Q_ϕ in FP32, keep the $\arg\max$ trajectory, and complete only that trajectory. The critic is read only as a scalar that orders realized trajectories; it does not modify any latent.

A.5 Latent world

Text is mapped to a compressed sentence latent by a frozen Perceiver autoencoder: $\mathbf{x}_0 \in \mathbb{R}^{L\times D}$ with L=32 latent tokens and D=32, for every task. Latents are ℓ_2-normalized per token, so every token lies exactly on a sphere of radius $\sqrt{D}$ and the action space is a product of spheres $(\mathcal{S}^{D-1})^L$ with zero norm variance.

A.6 HYPERPARAMETERS

We present the training details of the latent Q-function Q_ϕ in Table 4. All four critics share the same architecture and optimization settings. Apart from their task-specific terminal utility, the conditional critics for QQP and XSum differ only in that they must process the input context: they receive one additional token that encodes the source, obtained by mean-pooling the outputs of a frozen BART-base encoder and projecting them to the model width, and each selection group consists of rollouts of the same source.

Each critic is trained offline on 4,096 frozen-policy trajectories of 250 decisions (1,024,000 transitions), stored in a timestep-balanced prioritized replay buffer. For the conditional tasks, these trajectories are sampled without classifier-free guidance (scale 2.0). Terminal utilities are z-scored with training-set statistics, which lets all tasks share the same selection temperatures. We train each critic on a single NVIDIA H200 GPU for a fixed number of updates without early stopping and evaluate the final checkpoint.

A.7 INFERENCE TIME

Table 5 reports the end-to-end inference cost on ROCStories: the wall-clock time to produce $1{,}000$ outputs on one NVIDIA H200, covering latent denoising, the method's own candidate scoring, and decoding of the returned text. Model loading and evaluation are excluded. All methods run in batches of 50 outputs with the settings of Table 1 ($N{=}4$, 250 DDPM steps; TTSnap starts from 8 candidates and prunes at six intermediate points). We time one batch (the first 50 outputs of one generation seed) as the mean of two repetitions after a warm-up batch and scale it to $1{,}000$ outputs; for the base denoiser and SWIFT, this estimate is within 5% of timing all $1{,}000$ outputs directly. Each method uses one trained model. LQG completes all four trajectories and scores the full state history with Q_ϕ, exactly as in the main evaluation.

Denoising dominates the cost. In a separate breakdown of SWIFT, denoising the four candidates takes $229\,\mathrm{s}$ per $1{,}000$ outputs, greedily decoding all four candidates and scoring them with the reward head takes $28\,\mathrm{s}$, and beam-decoding the selected candidate takes $42\,\mathrm{s}$. Decoding runs at most 64 autoregressive steps and processes the four candidates in parallel, whereas the denoiser runs 250 sequential steps, so decoding every candidate adds only about 10% at $N{=}4$. On ROCStories, LQG therefore matches the cost of the output-dependent selector SWIFT ($2.47\times$ vs. $2.53\times$ the base denoiser) while reaching higher MAUVE. TTSnap, which also evaluates a reward model at its pruning points, costs $10.7\times$ the base denoiser.

A.8 CASE STUDY

B TRAINING DATASETS AND BENCHMARK DETAILS

B.1 TRAINING DATASETS

We use the four English datasets of LD4LG (Lovelace et al., 2023) and follow its splits and length limits: two unconditional generation tasks, ROCStories and AG News, and two sequence-to-sequence tasks, QQP and XSum. Table 8 summarizes the datasets, Table 9 lists the data used by each component of our pipeline, and Tables 10 and 11 show training examples.

ROCStories (Mostafazadeh et al., 2016). ROCStories is a crowd-sourced corpus of five-sentence stories about everyday life that capture causal and temporal commonsense relations between events. We use the combined 2016 and 2017 releases (98,161 stories) and treat the five sentences of a story as a single passage.

AG News (Zhang et al., 2015). AG News is a topic classification dataset of news articles, each with a title, a short description, and one of four topics (World, Sports, Business, Sci/Tech) in equal proportions. Following LD4LG, we discard the titles and generate only the descriptions. The descriptions contain HTML-escape artifacts from the original crawl (e.g., `#39;` for an apostrophe), and we train on them unchanged.

Table 4: Training details for the latent Q-function Q_ϕ across different datasets.

	ROCStories	AG News	QQP	XSum
Trainable Params	6.4M		6.6M	
Architecture	Transformer encoder			
Input Tokens	65 ($2L+1$)		66 ($2L+2$)	
Source Encoder	n/a	n/a	BART-base (frozen)	
Max Seq Length (Source)	n/a	n/a	64	256
Transformer Layers	8			
Transformer Dimension	256			
Self-Attention Heads	8			
Feed-Forward Dimension	1024			
Activation Function	ReLU			
Dropout	0.0			
Time Embedding	Learnable Fourier features (128-d) + MLP			
Pooling Weights (Action, State)	(0.7, 0.3)			
Output Head Initialization	Xavier uniform (gain 0.1)			
λ_{td}	1.0			
λ_{margin}	0.03			
$\lambda_{\text{selection}}$	0.1			
Discount Factor γ	0.995			
TD Target	One-step SARSA			
Target Network	Polyak ($\tau = 0.005$)			
Target Update Interval (Steps)	1	1	1	1
Base Margin δ	0.25			
Perturbed Actions M	5 (per-token norm preserved)			
Isotropic Noise Std	$0.5\,\|a_i\|/\sqrt{D}$			
Orthogonal Step Sizes	$\{0.2, 0.5, 1.0\} \times \|a_i\|$			
Margin Multipliers	(1, 1, 1, 1, 1)			
Selection Candidates N	4			
Selection Groups per Update	16			
Selection Temp. (τ_R, α)	(1.0, 1.0)			
Optimizer	AdamW			
Learning Rate	2e-4			
(β_1, β_2)	(0.9, 0.999)			
Batch Size	256			
Warmup Steps	0			
Learning Rate Schedule	Constant			
Weight Decay	1e-6			
Gradient Clipping	1.0			
Precision	FP32			
Training Steps	30k			
Training Trajectories	4,096 (8×512)		4,096 ($1{,}024 \times 4$)	
Sampling Timesteps T	250			
Trajectory Sampler	DDPM			
Guidance Scale (Trajectories)	n/a	n/a	1.0	1.0
Replay Sampling	Timestep-balanced PER			
PER (α_{PER}, ϵ)	(0.6, 0.01)			
PER β_{PER}	$0.4 \rightarrow 1.0$ (+0.001 per update)			

QQP (Chen et al., 2017). Quora Question Pairs contains about 400k question pairs from Quora, each annotated for whether the two questions ask the same thing. Following LD4LG, we use the paraphrase-generation version released with DiffuSeq (Gong et al., 2023), in which the pairs annotated as duplicates serve as source–target paraphrase pairs.

XSum (Narayan et al., 2018). XSum consists of BBC news articles published from 2010 to 2017 on a wide range of topics. The professionally written, single-sentence introduction of each article is removed from the article and serves as its summary, so a summary may contain information that is absent from the remaining article. Articles are long (489 tokens on average), and 71.4% of the

Table 5: End-to-end inference cost on AG News (latent denoising, candidate scoring, and decoding; one NVIDIA H200). Time is in seconds per 1,000 outputs. Calls is the number of denoiser forward passes per output, and Rel. is the time relative to the base denoiser. MAUVE and Div are taken from Table 1. LQG is timed with $(W, M)=(16, 4)$; the $(8, 2)$ setting used in Table 1 performs the same computation, since both score the full state history.

Method	Calls	Time (s)	Rel.	MAUVE↑	Div↑
Base denoiser ($N=1$)	250	116	1.00×	$.892_{\pm.014}$	$.514_{\pm.007}$
Post-decoding					
SWIFT	1,000	296	2.56×	$.858_{\pm.015}$	$.558_{\pm.006}$
TTSnap	1,089	1,143	9.86×	$.396_{\pm.051}$	$\mathbf{.580}_{\pm.010}$
Pre-decoding					
latent-SL BoN	1,000	270	2.33×	$.818_{\pm.027}$	$.556_{\pm.014}$
LQG (Ours)	1,000	298	2.57×	$\mathbf{.919}_{\pm.006}$	$.560_{\pm.006}$

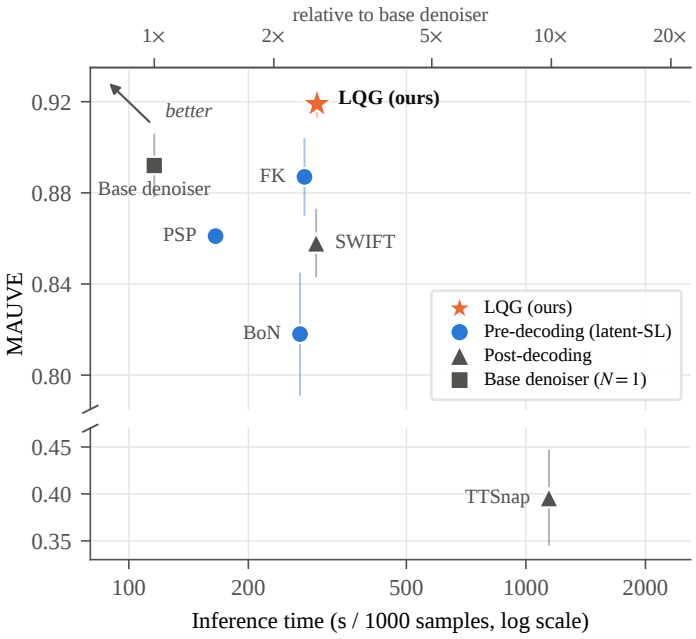


Figure 2: MAUVE versus end-to-end inference time (denoising, scoring, and decoding; 1,000 outputs on one H200) on AG News. LQG attains the highest MAUVE and is the only method that improves over the base denoiser, at the same cost as SWIFT (2.6× the base denoiser), while TTSnap costs 9.9×. The y-axis is broken; see Appendix A.7.

training articles exceed the 256-token source limit, so the denoiser sees only the beginning of most articles.

Preprocessing. Apart from minor whitespace cleanup, we use all texts as released, without filtering or deduplication. We tokenize all texts with the BART-base tokenizer. The target text, i.e., the text that the autoencoder compresses and the denoiser generates, is truncated to 64 tokens for every task; this truncates 13.6% of the ROCStories and 6.0% of the AG News training texts (Table 8). For the conditional tasks, the source is truncated to 64 tokens for QQP and 256 tokens for XSum.

B.2 Porting for External Baselines

We compare QT with five baselines. TTSnap was developed for image diffusion models, and SWIFT and TrajSelector for autoregressive LLMs; none of them has a reference implementation for latent diffusion language models. We port each of them to the frozen autoencoder, denoiser, and BART

Table 6: Case study on ROCStories. Each row is one group of $N=4$ candidate trajectories: all methods share the same denoiser and candidate noise, and the base denoiser's sample is the first candidate. Outputs are shown verbatim.

Base denoiser	SWIFT	latent-SL BoN	**LQG**
Jack wasn't going to propose to his girlfriend. He was going to buy her a ring. He made a beautiful bouquet. Finally, she got down on one knee. He proposed and she said yes to him.	On June 20th, we had a jambalaya shop. We ordered takeout and meat. We smoked ribs and quimboni and meat meat. He was very popular and listened to us. We went back and everyone drank it.	Jason was walking down the street when he saw a man. He saw him up and asked if he lived there. The man looked suspicious, so Jason stopped. He offered the man a twenty dollar bill. Then Jason dropped the man in his wallet.	I was driving on the highway the other day. I was going over the speed faster than normal. Suddenly, I was pulled over by a cop. The cop gave me a ticket for speeding. From then on, I always vowed to never drive again.
One day Peter took a trip to Boston, MA. First he went to the NBA Park. Then he goes to baseball Stadium. Finally he wentto Ordway Park. He had a fun day.	Jackson was walking home from school. Suddenly, he tripped on a puddle. Suddenly,. He tripped and fell. The rock hurt him profusely. Jackson promised to be more careful from then on.	James loved being in the woods. He went hiking every weekend. However, this weekend he got lost in the forests. The family spent an entire day searching for James. They finally found James after two hours in the forest.	Sam was playing soccer. It was the championship. His team was down by one point. Sam was sure he made it to the last shot. He missed the shot and lost the game.
My nephew's niece visited from NYC last week. She and her mother-in-law have two kids. Her 10 year old son posted photos on FB. My nephew took photos but he did not. His kid laughed though, as he knew he was asleep.	Fisher heard that all travel would be very slow and extensive. The first thing he did was do lipwriting in his novel. Soon the end result was Christmas. He rewaved his video to explore ideas and ideas. And the activities of the trip were like magic.	Martha went to eat at a fast food place. She ordered a hamburger. It came up 105 degrees for the temperature temperature. Her manager complained. She said there was nothing wrong with her order.	I decided to make toast toast for my boyfriend. I let the toast toast tack too hard in the toast. I tried to toast the toast but it still burnt. I realized he wouldn't like toast Toast. I followed the recipe again.

decoder shared by all rows of Table 1, keeping its selection rule and as much of its released training recipe as our setting allows; every deviation is stated in its subsection. Our tasks have neither pretrained reward models nor binary correctness labels, so all learned components are supervised with the terminal utility R that also defines the reward of Q_ϕ (Section 5.1). After selection, the chosen latent is decoded with the same beam search as the outputs of all other rows. LD4LG MBR and latent-SL need no porting and are described in Sections B.6 and B.7.

B.3 TTSnap

Original method. TTSnap (Yu et al., 2025b) runs Best-of-N over initial noises and prunes candidates before they are fully denoised. At each verification step, it decodes the Tweedie estimate of every surviving candidate with the VAE and scores the resulting image with a noise-aware reward model. It keeps a fixed fraction of the candidates and returns the survivor with the highest reward at the final step (Algorithm 1 of Yu et al., 2025b). The noise-aware reward models are obtained by fine-tuning off-the-shelf image reward models (ImageReward, HPS, and PickScore) by self-distillation.

Table 7: Case study on XSum. Each block is one validation document (truncated to its first 60 words), its reference summary, and the summary returned by each method. Documents are drawn uniformly at random from those evaluated by all three methods on which the three methods return three different summaries (78 of 79 documents). Blocks are matched by source document only; unlike Table 6, the outputs do not come from a shared candidate pool. Outputs are shown verbatim.

Document	Ritzy Cinema workers in Brixton get paid a minimum of £8.80 an hour. Bectu is calling for an increase to £9.40 an hour. Union bosses said Picturehouse Entertainment is being "very hard faced" and has refused to negotiate. Picturehouse said pay would rise to £9.10 an hour on 2 September as part of a three-year agreement with Bectu. Balloting began ...
Reference	Staff at a cinema are being balloted by a union calling for the London living wage to be paid.
Base denoiser	Workers at an East Midlands cinema are to vote next week in a long dispute over pay.
latent-SL BoN	Workers at one of England's biggest cinema are to be balloted on strike action in a dispute over pay.
LQG	Workers at a cinema in south London are to vote whether to take part in next month's strike in a dispute over pay.
Document	The Office for National Statistics said there were 3.5 million such households in the UK between April and June this year, about 17.1% of all households including a working age adult. This was down from 3.7 million, or 17.9%, a year earlier. The highest percentage of workless households was in north-east England. Some 23% of households including a working age ...
Reference	The proportion of UK households where no adult aged 16 to 64 is in work is at its lowest since comparable records began in 1996, official figures show.
Base denoiser	The number of households identified as workless in the UK has fallen to its lowest level, figures show.
latent-SL BoN	The number of teenagers living in workless households in the UK has fallen to its lowest level, official figures show.
LQG	The proportion of households aged 16 to 64 in the UK has fallen to its lowest level since 2008, official figures show.

Table 8: Dataset statistics. Lengths are measured in BART-base tokens, including the two special tokens, on the training split. "Truncated" is the fraction of training texts that exceed the length limit.

	ROCStories	AG News	QQP	XSum
Task	Unconditional	Unconditional	Paraphrasing	Summarization
Source	n/a	n/a	Question	News article
Target	Story	News description	Question	Summary
Training instances	93,161	119,000	144,715	204,045
Validation instances	1,000	1,000	2,048	11,332
Test instances	4,000	7,600	2,500	11,334
Max target length		64		
Mean target length	53.3	43.7	13.8	28.1
Truncated targets	13.6%	6.0%	<0.1%	0.2%
Max source length	n/a	n/a	64	256
Mean source length	n/a	n/a	13.8	488.6
Truncated sources	n/a	n/a	<0.1%	71.4%

Table 9: Data used by each component of our pipeline.

Component	ROCStories, AG News	QQP, XSum
Autoencoder, denoiser	Training split	Training split (autoencoder: targets only)
Critic trajectories	4,096 rollouts from independent Gaussian latents	4 rollouts for each of 1,024 training sources
Evaluation	MAUVE: first 1,000 validation texts; memorization: training split	First 1,000 validation sources and their references

The fine-tuning follows a curriculum from the cleanest to the noisiest timestep, trains for one epoch per timestep, and saves one checkpoint per timestep.

Reward model. TTSnap scores the decoded image of the Tweedie estimate, not the latent itself. The counterpart in an LDLM is to decode each candidate's current clean-latent estimate $\boldsymbol{a}_h = \hat{\mathbf{x}}_0$ into text and score that text. We keep this decode-then-score design. In our tasks, there is no pretrained per-sample reward model to fine-tune, and MAUVE is a corpus-level statistic. We therefore build the reward model from a frozen GPT-2-large encoder and a small trainable head. The encoder is the featurizer already used by the density-ratio utility and by MAUVE: it truncates the input to 256 tokens and returns the final-layer hidden state of the last token, a 1280-dimensional feature. The head follows the design of ImageReward's MLP head. As in TTSnap's default configuration, the noise level is not an input to the network; noise awareness comes from keeping a separate checkpoint for each verification step.

Training. TTSnap trains noise-aware checkpoints for the first 15 of its 20 sampling steps and scores the final step with the clean reward model. We use seven checkpoints spaced evenly over the last 75% of our 250-step chain, at $h \in \{62, 93, 124, 155, 187, 218, 249\}$. For each training trajectory, we decode the clean-latent estimate $\hat{\mathbf{x}}_0$ at each of the seven steps and extract its GPT-2-large feature. All seven examples of a trajectory share one target, the utility R of its final text, so the model is therefore trained by outcome regression (self-distillation) rather than by bootstrapping.

Curriculum training. As in TTSnap, a single reward model is trained through the checkpoints from clean to noisy ($h = 249, 218, \ldots, 62$). Each stage runs one epoch of MSE regression with a new optimizer, starting from the weights of the previous stage. The weights at the end of a stage become the checkpoint for that noise level. TTSnap starts this relay from a pretrained reward model. No such model exists for our utilities, so the first stage ($h = 249$) trains the head from random initialization on terminal texts; the resulting checkpoint plays the role of the off-the-shelf clean-sample reward model. For the same reason, we train only the head, with a much larger learning rate than TTSnap's

Table 10: Training examples from the unconditional datasets. For AG News, we show one description per topic; the topic labels are not used. HTML-escape artifacts in AG News are decoded for readability.

ROCStories	AG News
Leighton was helping her friend pull off her dream wedding. Things were going well until it began to rain. Leighton was freaking out before she had a bright idea. She moved everything under a nice awning with a good view. The wedding proceeded near the gentle rain and it was lovely.	*(World)* Leaders of China's Communist Party have opened a four-day plenum amid reports of internal jockeying between supporters of President Hu Jintao and former President Jiang Zemin.
Samantha was playing the recorder in the car. Tycho started to tease her. Samantha hit her brother with the recorder. Samantha would not say she was sorry. Samantha's dad broke the recorder.	*(Sports)* There was little doubt what the focus of Tuesday night at Old Trafford would be as soon as the Manchester United team had been named: Wayne Rooney.
Miranda decided to go to Dublin. It was spur of the moment. When she got there she wanted to go clubbing. Luckily her friends agreed. Miranda had so much fun.	*(Business)* Consumer confidence in the US economy fell in August for the first time since February as job growth faded and oil prices rose to a record.
Kay had never been stung by a bee. She did not think she was allergic, since no one in her family was. But at a picnic one summer at age 18, she was stung. Her arm swelled and became red and very painful. It turned out Kay was allergic to bees after all!	*(Sci/Tech)* For a theoretically free operating system, Linux is—and will continue to be—a cash cow. A research firm sees a bright financial future for the OS.

full-network fine-tuning rate of 3×10^{-6}. We also keep the learning rate constant, whereas TTSnap uses a linear schedule. The remaining settings are listed in Table 12.

Online pruning. We implement Algorithm 1 of Yu et al. (2025b) inside the DDPM loop. Sampling starts from $K_0 = 8$ candidates, which keeps the total cost close to that of Best-of-4. TTSnap's retention ratios were tuned for a 20-step image sampler with at most five verification steps and do not transfer directly to our six. We therefore use a single constant ratio, $\alpha = 0.75$. The population shrinks as $8 \to 6 \to 4 \to 3 \to 2$ by $h = 155$, and the verification steps at $h = 187$ and $h = 218$ keep both survivors. At the final step, both survivors are scored with the $h = 249$ checkpoint, and the beam-decoded text of the higher-scoring one is returned.

B.4 SWIFT

Original method. SWIFT (Guo et al., 2026b) reranks N completed reasoning paths of an LLM with a linear reward head on the LLM's own hidden states. For each generated token t, it concatenates the hidden states of all layers into $\boldsymbol{h}_t$. A single linear map $W\boldsymbol{h}_t + \boldsymbol{b}$ produces a gate logit $\tilde{g}_t$ and a token reward r_t. The path score is the gated average $\hat{R} = \sum_t \sigma(\tilde{g}_t)\, r_t / \max\big(\sum_t \sigma(\tilde{g}_t), \epsilon\big)$. The head is trained with binary cross-entropy on answer correctness; hinge, DPO, InfoNCA, and NCA losses are reported as alternatives. At test time, the highest-scoring path is returned.

Hidden states. In an LDLM, the tokens are emitted by the frozen BART decoder conditioned on the terminal latent, and the denoiser produces no token-level states. We therefore attach SWIFT's head to the BART decoder, whose hidden states correspond to those of SWIFT's task LLM. Attaching the head to the diffusion latent instead would turn SWIFT into another latent value regressor. As in the original method, every candidate must be fully denoised and decoded before it can be scored, so we list SWIFT as a post-decoding baseline. SWIFT scores a text from one hidden-state vector per generated token. To obtain these vectors, we decode each candidate's terminal latent greedily, up to 64 tokens, and record the top-layer decoder state (768 dimensions) at every generation step. We skip the first step, whose output is the beginning-of-sequence token that BART always forces, so it reflects no choice by the model. We also mask the padding steps that follow the end-of-sequence token. We use only the top layer rather than concatenating all six decoder layers, since SWIFT's layer ablation locates most of the discriminative information in the final layers. The scoring decode

Table 11: Training examples from the conditional datasets. Parts of the XSum articles are omitted for brevity.

QQP	
Source question	Target question
Can you buy ballsyic gel/targets in Australia?	Can you buy ballistic gel/targets in Australia?
What is the superstition regarding your eye twitching? Is there any validity to it?	What are the signs and omens for eye twitching?
What is the U.S.A. doing to prevent China from becoming a superpower?	Short of a war, how will the USA prevent China from becoming an economic and a military superpower?

XSum	
Article (source)	Summary (target)
Thameslink and East Midlands train services are suffering delays after damage to overhead wires at Luton on Tuesday caused signalling problems. Both operators are running amended services on Wednesday. Govia Thameslink said Network Rail "still can't tell us when they expect to be able to repair it". Spokesman Roger Perkins said: "That's understandable, they really are doing their very best, but it means that although we're saying [disruption] until at least 2pm, this could well continue right through to the end of the day." ...	Disruption on the line between the east Midlands, Bedfordshire, Hertfordshire and London could last all day, a rail operator has said.
The 25-year-old has scored one goal in five appearances for the Championship club this season. "He had a little problem and it was determined he would have to have an operation on his meniscus," Albion boss Chris Hughton told BBC Sussex. "It is a big blow as he had been really good in training and was excellent in the two cup games."	Brighton & Hove Albion winger Kazenga LuaLua will be out for six weeks after having surgery on a knee injury.

is greedy because beam search interleaves hidden states across beams. After selection, the winning latent is decoded again with the standard beam search, so SWIFT's outputs are decoded like those of every other row.

Reward head and training. The head is SWIFT's linear layer, $W \in \mathbb{R}^{2\times 768}$ plus a bias (1,538 parameters), followed by the gated average above. Our tasks have no binary correctness label, so we supervise the head with the terminal utility R (Section B.2) of the greedy text from which the hidden states are extracted. SWIFT's default binary cross-entropy would require thresholding R. A threshold computed within each group, however, would make a candidate's label depend on the other candidates in its group. We therefore use the InfoNCA loss from SWIFT's loss study with the same temperature, $\alpha = 0.01$, over the $N = 4$ candidates of a group:

$$\mathcal{L}_{\text{InfoNCA}} = -\sum_{i=1}^{N} \text{softmax}_i\big(R^{(i)}/\alpha\big)\ \log \text{softmax}_i\big(\hat{R}^{(i)}\big), \tag{17}$$

where $R^{(i)}$ and $\hat{R}^{(i)}$ are the label and the predicted score of candidate i. We train with Adam at learning rate 10^{-3} for 20 epochs, one group per update. Since no held-out split is reserved, there is no early stopping. The original uses AdamW at 10^{-4} with batch size 16 and early stopping.

B.5 TRAJSELECTOR

Original method. TrajSelector (Yu et al., 2025a) performs Best-of-N selection for large reasoning models with a small process verifier that reads the hidden states of the sampling model. Each sampled reasoning trace is split into steps at the delimiter "\n\n", and the last-layer hidden state of the sampler at the final token of each step represents that step. A linear layer projects these vectors into the input space of Qwen3-0.6B-Base, which reads them as a sequence of input embeddings. A score head, two linear layers with a ReLU, then maps each step to probabilities over three classes: wrong,

Table 12: Implementation details of the ported TTSnap and SWIFT baselines. The step index h counts denoiser calls of the 250-step sampler, from pure noise ($h = 0$) to the final call ($h = 249$). R is the terminal utility that also defines the reward of Q_ϕ (Section 5.1).

	ROCStories	AG News	QQP	XSum
TTSnap				
Scored input	Beam-decoded $\hat{\mathbf{x}}_0$ of each survivor			
Text encoder (frozen)	GPT-2-large, last-token state (1280-d)			
Source input	n/a	n/a	None	
Reward head	Linear 1280→256→64→16→1			
Dropout	(0.2, 0.2, 0.1)			
Trainable params	345K per checkpoint			
Checkpoint steps h	62, 93, 124, 155, 187, 218, 249			
Curriculum	$h = 249 \rightarrow 62$, 1 epoch per stage			
Label	Terminal R (density ratio)		Terminal R (ROUGE+BERTScore)	
Loss	MSE			
Optimizer	AdamW, re-initialized every stage			
(β_1, β_2)	(0.9, 0.999)			
Weight decay	1e-2			
Learning rate (constant)	1e-3	1e-4	1e-3	1e-3
Batch size	32	32	8	8
Guidance scale (data)	n/a	n/a	2.0	2.0
Initial candidates K_0	8			
Verification steps h	62, 93, 124, 155, 187, 218			
Retention ratio α	0.75 (survivors 8→6→4→3→2→2→2)			
Final selection	argmax of the $h = 249$ checkpoint			
Denoiser calls / output	1,089	1,089	2,178	2,178
Decodes / output	23 (beam search)			
SWIFT				
Hidden states	BART-base decoder, top layer (768-d)			
Scoring decode	Greedy, at most 64 tokens			
Reward head	Linear 768→2 (gate, token reward)			
Trainable params	1,538			
Label	R (density ratio)		R (ROUGE+BERTScore)	
Loss	InfoNCA ($\alpha = 0.01$) over groups of $N = 4$			
Optimizer	Adam			
Learning rate	1e-3			
Epochs	20 (no early stopping)			
Batch size	1 group (4 candidates)			
Candidates N	4			
Guidance scale (candidates)	n/a	n/a	1.0	1.0
Denoiser calls / output	1,000	1,000	1,000	1,000
Decodes / output	4 greedy + 1 beam search			
Training seeds (both)	42, 142, 242			

buffer, and right. A step's score is its probability of the right class, a trace's score is the mean of its step scores, and the highest-scoring candidate is returned. Following FreePRM, the correctness of the final answer is copied to every step as a pseudo-label, and the buffer class lets the verifier avoid committing on steps whose pseudo-label may be wrong. The sampler stays frozen, while the projection, the 0.6B model, and the head are trained.

Steps and hidden states. In our implementation, each candidate's terminal latent is decoded greedily, up to 64 tokens, and the top-layer states of the frozen BART decoder are recorded. Our outputs contain no "\n\n" delimiters, so we use sentences as steps and split them directly on BART token ids. A token ends a sentence if it is one of the 74 sentence-final punctuation tokens of the BART vocabulary and it is either the last token of the text or followed by a new word that starts with an uppercase letter, a digit, a quotation mark, or an opening parenthesis. Each sentence is represented by the decoder state at its final token, that is, the state after the decoder has read the punctuation.

Table 13: Implementation details of the ported TrajSelector baseline, for the tuned variant reported in Table 1. It differs from the released recipe in the learning rate (released: 10^{-4}), a 5% warmup (released: none), and the zero-initialized output layer (released: Xavier uniform).

	ROCStories	AG News	QQP	XSum
Hidden states	BART-base decoder, top layer (768-d)			
Scoring decode	Greedy, at most 64 tokens			
Step	Sentence, at most 16 per candidate			
Mean steps per candidate	4.9	1.2	1.0	1.0
Step representation	Decoder state at the sentence-final token			
Projection	Linear 768→1024			
Verifier	Qwen3-0.6B-Base (all parameters trained)			
Score head	Linear 1024→1024, ReLU, Linear 1024→3			
Trainable params	0.6B			
Label	R (density ratio)	R (ROUGE+BERTScore)		
Binarization	$R >$ training-set median			
Loss	Released FreePRM loss (last-step weight 3)			
Optimizer	AdamW, linear decay			
Weight decay	1e-2			
Gradient clipping	1.0			
Epochs	3			
Effective batch size	8 candidates (2×4 accumulation)			
Precision	FP32 master weights, BF16 autocast			
Verifier seeds	42, 142, 242			
Tuned variant				
Learning rate grid	{1e-4, 3e-5, 1e-5}			
Selected learning rate	1e-5	3e-5	3e-5	3e-5
Val. pairwise accuracy	0.59	0.71	0.52	0.51
Warmup	5% of updates			
Loss computation	Log space (no ϵ)			
Head initialization	Xavier, zero output layer			
Tuning split	20% of training groups, 2 tuning seeds			
Denoiser calls / output	1,000	1,000	2,000	2,000

Labels. The labels come from the same terminal utility R as for SWIFT (Section 5.1), computed on the greedy text from which the states are extracted. TrajSelector's loss needs a binary outcome, so we binarize R with one threshold: the median of R over all training candidates, computed once before training. Candidates above the median are labeled positive and the rest negative, including ties. As in the original, the label of a candidate is copied to all of its steps.

Tuned training. Verifiers trained with the released recipe (four tasks, three seeds each) give all four candidates the same score, so the selection always returns the first candidate and the output is identical to that of the base sampler. We therefore also train a tuned variant, which changes only what is needed for training to converge: (i) the output layer of the score head is initialized to zero, so the three classes start equally likely; (ii) the learning rate warms up linearly over the first 5% of updates; (iii) the learning rate is selected from $\{10^{-4}, 3 \times 10^{-5}, 10^{-5}\}$. For each learning rate, we average the validation pairwise ranking accuracy of two tuning seeds. Learning rates within 0.02 of the best average count as tied, since differences of this size are within the seed-to-seed noise of the small validation split, and among them we pick the largest, which is closest to the released 10^{-4}. The rule selects 10^{-5} on ROCStories and 3×10^{-5} on the other three tasks. The final verifiers are then trained with seeds 42, 142, and 242.

B.6 LD4LG MBR

Implementation. LD4LG (Lovelace et al., 2023) draws several outputs for each source from independent initial latents and selects one of them by minimum Bayes risk (MBR) decoding, with the negative ROUGE as the loss. It thus returns the candidate with the highest total ROUGE against the other candidates, that is, the candidate that agrees most with the rest. The released implementation uses ROUGE-2, and the LD4LG paper reports this rule with five candidates and guidance scale

2.0. The rule involves no training and never sees the reference target, so it applies to our LDLMs without porting. We keep it unchanged and only reduce the number of candidates to $N = 4$, as in the other rows of Table 1. Candidate i receives the score $s_i = \sum_{j \neq i} \mathrm{R2}(y_i, y_j)$, where $\mathrm{R2}(y_i, y_j)$ is the ROUGE-2 F1 of y_i with the other candidate y_j as the reference, computed with the Hugging Face `evaluate` implementation without stemming.

B.7 LATENT-SL

Purpose. The latent-SL rows replace Q_ϕ with a latent value function trained by supervised regression onto observed returns and keep everything else fixed. Comparing a latent-SL row with the QT row of the same operator therefore isolates the effect of the critic.

Table 14: Implementation details of the latent-SL critic. Its training trajectories, reward standardization, optimizer, and replay settings are identical to those of Q_ϕ (Table 4).

	ROCStories	AG News	QQP	XSum
Inputs	$\mathbf{z}_t$, a_t, t		$\mathbf{z}_t$, a_t, t, source	
Source vector	n/a	n/a	BART-base encoder mean (768-d)	
Time embedding	Learned Fourier features (128-d)			
Input dimension d_{in}	2,176		2,944	
Network	Linear $d_{\text{in}} \to 256 \to 64 \to 16 \to 1$			
Dropout	0.0			
Trainable params	575K		771K	
Initialization	PyTorch default			
Training				
Training trajectories	Those of Q_ϕ (4,096 per task)			
Regression target	$\gamma^{249-t}\tilde{R}$ (no bootstrapping)			
Reward standardization	Training-set mean and std			
Discount factor γ	0.995			
Auxiliary losses	Margin 0.3, InfoNCE 1, path 1, gradient 50			
Optimizer	AdamW			
Learning rate	2e-4 (constant, no warmup)			
(β_1, β_2)	(0.9, 0.999)			
Weight decay	1e-6			
Batch size	256			
Gradient clipping	1.0			
Precision	FP32			
Replay sampling	Timestep-balanced PER			
PER $(\alpha_{\text{PER}}, \epsilon)$	(0.6, 0.01)			
PER β_{PER}	0.4 $\to$ 1.0 (+0.001 per update)			
Decisions sampled	Last 16, last 64, then all 250		All 250	
Updates	10k per stage (30k)		30k	
Critics per task	3 (initialization seeds of Q_ϕ)			
Denoiser calls / output				
BoN	1,000	1,000	2,000	2,000
PSP	511	511	1,010	1,010
FK	1,032	1,032	2,032	2,032

Value network. The network reads the same inputs as Q_ϕ: the noisy latent $\mathbf{z}_t$, the action $a_t = \hat{\mathbf{x}}_0$, the time t, and, on the conditional tasks, the source. It flattens $\mathbf{z}_t$ and a_t into $2 \times 32 \times 32 = 2{,}048$ values and appends a 128-dimensional learned Fourier embedding of t. On the conditional tasks, it also appends the average of the frozen BART-base encoder states of the source over positions, a 768-dimensional vector. It has four linear layers ($d_{\text{in}} \to 256 \to 64 \to 16 \to 1$), 575K trainable parameters on the unconditional tasks and 771K on the conditional tasks.

Training. Latent-SL regresses each decision's observed discounted return $\gamma^{249-t}\tilde{R}$ with an importance-weighted squared error, without bootstrapping; the remaining settings are listed in Table 14. The loss also includes four auxiliary terms computed on the perturbed actions of the margin loss of Q_ϕ: the margin loss itself, an InfoNCE loss, a hinge that requires the value to increase along a

spherical path from a random direction to the observed action ("path"), and a term that aligns the action gradient of the value with that path ("gradient"); Q_ϕ uses only the margin loss, with weight 0.03, together with its selection loss.

C Additional Related Work

Energy-based and iterative-reasoning views of diffusion. Casting each diffusion step as an optimization problem rather than a fixed forward pass is the central idea of energy-based diffusion. EDLM (Xu et al., 2025) augments a pretrained discrete diffusion model with a residual EBM $p_{\text{EDLM}}(x_{t-1} \mid x_t) \propto p_{\text{diff}}(x_{t-1} \mid x_t) \exp(-E(x_{t-1}, x_t))$ trained by noise contrastive estimation, correcting the token-independence assumption that causes accumulated decoding error in masked diffusion LMs. IRED (Du et al., 2024) learns a *sequence* of annealed energy landscapes $E^k_\phi(x, y)$ over which test-time inference performs gradient descent, showing that the number of optimization steps is itself a continuous knob that lets a single trained model generalize to harder reasoning instances. Earlier implicit-optimization EBMs (Du et al., 2022) share this gradient-based-inference view but differentiate through the inner loop at training time, which is slow and unstable. Like EDLM, we keep a pretrained generator and add a learned scalar on top of it; unlike EDLM and IRED, the scalar is not used to move samples. Our critic Q_ϕ is trained with a TD objective against a terminal utility, which lets it be shaped by a non-differentiable sequence-level reward rather than by paired clean targets, and at inference it only *ranks* trajectories that the frozen denoiser has already produced.

Reinforcement learning for diffusion models. A separate strand of work treats diffusion sampling as an MDP and uses reinforcement learning to align a pretrained denoiser. DDPO (Black et al., 2024) and DPOK (Fan et al., 2023) apply policy gradient to fine-tune a diffusion model against a reward (e.g., CLIP score or human preference), and reward-weighted and preference-based fine-tuning variants follow the same template. These methods update the policy parameters θ; the per-step action is still produced by a single forward pass of the fine-tuned denoiser. We share the MDP view but not the object being learned: π_θ stays frozen, and the only trained component is a value function. Because the critic is fit by fixed-policy TD on offline rollouts of π_θ, it performs *policy evaluation*, not control (§4.2); the practical consequence is that the method adds no fine-tuning cost to the generator, leaves its weights untouched, and can be attached to any existing checkpoint. The closest RL analogue is a learned value/verifier used to rerank rollouts of a fixed policy, rather than reward fine-tuning.

Inference-time search and test-time compute for diffusion. Best-of-N sampling, sequential Monte Carlo and Feynman–Kac particle steering (FK Steering (Singhal et al., 2025)), tree search over diffusion trajectories (Li et al., 2024), and Langevin/annealed-MCMC correctors (Song et al., 2021b) all trade additional inference compute for sample quality. Most of these methods score candidates with an external reward evaluated on (a prediction of) the clean sample, and therefore either complete every trajectory before selecting or decode intermediate predictions to text at every resampling step. Our critic plugs into the same selection templates — we evaluate it inside BoN, particle (FK), and PSP-style samplers, and compare against the same samplers driven by a direct scorer (SWIFT (Guo et al., 2026b), TTSnap (Yu et al., 2025b), PSP (Guimaraes & Perona, 2026a)). Unlike these direct scorers, however, it scores the *noisy latent* state (s_t, a_t, t) directly. This allows a decision at an intermediate index $t^\star$, after which only the selected trajectory is completed. The resulting saving is measured against running the same N-candidate pool to completion, not against a single sample (§4.3).

Latent geometry of text autoencoders. LD4LG (Lovelace et al., 2023) observed that BART encoder latents occupy a narrow, anisotropic norm band and introduced self-conditioning to keep denoiser predictions consistent across steps; subsequent work investigated flow-matching priors (Lovelace et al., 2024) and density penalties for keeping samples on-manifold. This geometry matters to any method that edits latents directly. We sidestep it by working in a compressed autoencoder whose latents are ℓ_2-normalized per token, so that the action space is an exact product of spheres with zero norm variance (Appendix A.5); since the method only selects among trajectories generated by π_θ, it never leaves the support of the denoiser and requires no manifold projection.