EntRGi: Entropy Aware Reward Guidance for Diffusion Language Models

Atula Tejaswi [*1] **Litu Rout** [*1] **Constantine Caramanis** [1] **Sanjay Shakkottai** [1] **Sujay Sanghavi** [1]

Abstract

Reward guidance has been applied to great success in the test-time adaptation of continuous diffusion models; it updates each denoising step using the gradients from a downstream reward model. We study reward guidance for *discrete* diffusion language models, where one cannot differentiate through the natural outputs of the model because they are discrete tokens. Existing approaches either replace these discrete tokens with continuous relaxations, or employ techniques like the straight-through estimator. In this work, we show the downsides of both these methods. The former degrades gradient feedback because the reward model has never been trained with continuous inputs. The latter involves incorrect optimization because the gradient evaluated at discrete tokens is used to update continuous logits. Our key innovation is to go beyond this trade-off by introducing a novel mechanism called *EntRGi: Entropy aware Reward Guidance* that dynamically regulates the gradients from the reward model. By modulating the continuous relaxation using the model's confidence, our approach substantially improves reward guidance while providing reliable inputs to the reward model. We empirically validate our approach on a 7B-parameter diffusion language model across 3 diverse reward models and 3 multi-skill benchmarks, showing consistent improvements over state-of-the-art methods.

1. Introduction

Reward guidance has proven highly effective for test-time adaptation in continuous diffusion models, where gradients from a downstream reward model are used to iteratively refine each denoising step toward desired outcomes (Dhariwal & Nichol, 2021). This paradigm has enabled controllable generation across inverse problems (Chung et al., 2023; 2024; Rout et al., 2023; 2024), stylization (Hertz et al., 2023; Rout et al., 2025b), and semantic editing (Rout et al., 2025a), by allowing diffusion models to optimize task-specific objectives without retraining. Motivated by the success, recent focus has increasingly shifted toward inference-time steering for diffusion models as a promising alternative to post-training adaptation in large language models (LLMs).

In this work, we study reward guidance in the setting of *discrete* diffusion large language models (dLLMs) (Austin et al., 2021; Lou et al., 2024; Sahoo et al., 2024; Shi et al., 2024; Nie et al., 2025; Ye et al., 2025; DeepMind, 2025). Unlike autoregressive LLMs, dLLMs generate text by starting from a fully masked sequence and iteratively denoising tokens in parallel, not necessarily committing to a fixed left-to-right order. This parallel and order-agnostic generation paradigm provides a natural foundation for controllability and inference-time steering. However, it also introduces a fundamental challenge for *discrete* diffusion: the natural outputs of dLLMs are discrete tokens, which prevents direct gradient propagation from reward models.

Existing approaches to address this challenge can be broadly categorized into training-based and training-free methods. Training-based approaches focus on adaptation and post-training of dLLMs (Rector-Brooks et al., 2024; Borso et al., 2025; Tang et al., 2025; Wang et al., 2025), such as instruction tuning (Nie et al., 2025) and reinforcement learning (Zhao et al., 2025; Zekri & Boullé, 2025). Since training-based methods can be expensive, there has been growing interest in training-free inference-time steering of discrete diffusion models across text and image domains (Dang et al., 2025; Ou et al., 2025; Rout et al., 2025c).

Training-free approaches address the non-differentiability of discrete tokens differently depending on the differentiability of the reward model. When the reward model is non-differentiable, a common line of work selects one trajectory among many runs, often guided by particle-based sampling and resampling to favor high-reward samples (Dang et al., 2025; Singhal et al., 2025). For differentiable reward, there are two primary ways to deal with non-differentiability of discrete tokens. One class of methods replaces discrete tokens with continuous relaxations, enabling gradients to propagate through soft embeddings (Murata et al., 2024). Despite the simplicity, reward models are trained exclu-

[*] Equal contribution [1] The University of Texas at Austin. Correspondence to: Atula Tejaswi <atutej@utexas.edu>.

Preprint. February 4, 2026.

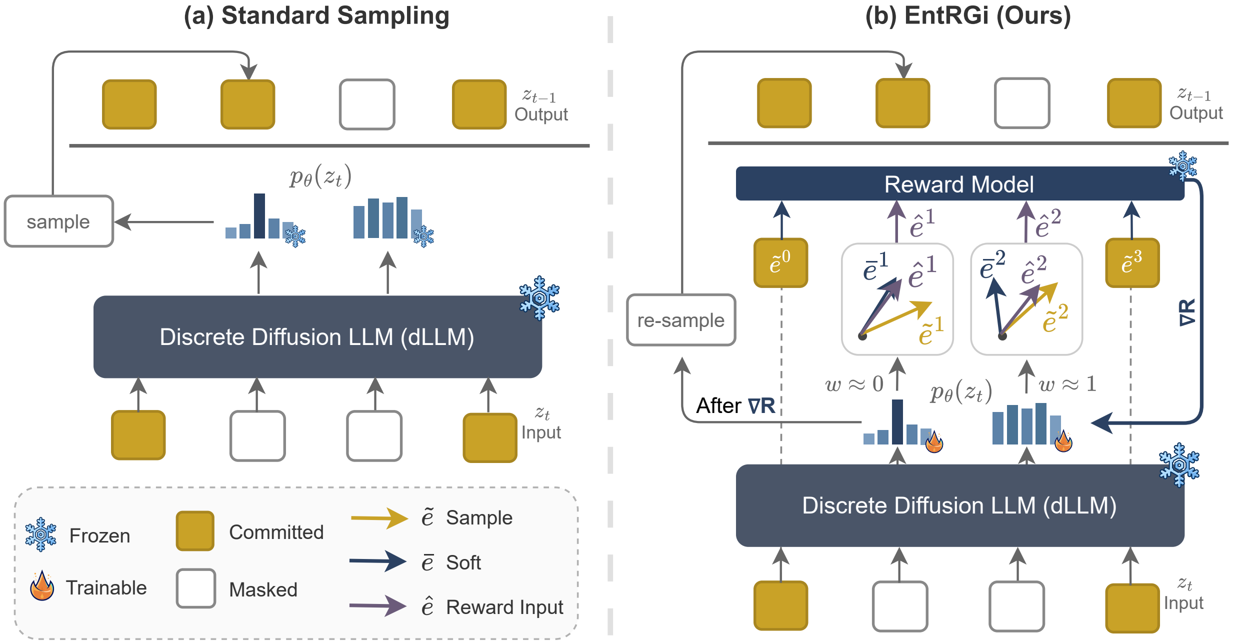


Figure 1. **Overall pipeline of Entropy-aware Reward Guidance (EntRGi)**. In standard sampling methods (Ye et al., 2025; Nie et al., 2025), the current input z_t is fed to the discrete diffusion LLM (dLLM), which produces output distributions at the masked positions; the most confident tokens are then committed to obtain z_{t-1}. Our method EntRGi instead modifies the logits at the masked positions using gradients from a reward model, while keeping both the dLLM and the reward model frozen. The embeddings provided to the reward model at masked positions are constructed as an *entropy-weighted interpolation* between a continuous relaxation of the token embeddings and sampled hard token embeddings. Lower entropy proportionally emphasizes the continuous relaxation, while higher entropy increases reliance on hard tokens via a straight-through estimator (Bengio et al., 2013; Jang et al., 2017; Rout et al., 2025c).

sively on discrete text, and querying them with continuous inputs can significantly degrade the reliability of the gradient guidance. Another class of methods discretizes the soft embeddings and relies on the straight-through estimator (STE) (Bengio et al., 2013) to provide more reliable gradients (Rout et al., 2025c). While this enables optimization in practice, it introduces an inherent mismatch: gradients evaluated at discrete tokens are used to update continuous logits, leading to potentially incorrect optimization.

To go beyond this tradeoff, we introduce **EntRGi** (**Ent**ropy-aware **R**eward **Gui**dance)[1], an entropy-aware reward guidance mechanism for discrete diffusion language models. As illustrated in Figure 1, EntRGi interpolates between the continuous relaxation of the token embeddings (also known as soft token embeddings) and sampled hard token embeddings by using the model's unconditional entropy. Thus, EntRGi provides reliable gradients during optimization by ensuring that the reward model is evaluated on inputs it can reliably interpret throughout the denoising process.

Our contributions can be summarized as follows. **(1)** We introduce **EntRGi**, an entropy-aware reward guidance method for discrete diffusion language models. **(2)** We conduct extensive experiments on a 7B-parameter diffusion language model (Ye et al., 2025). Using 3 reward models (Liu et al., 2025) and 3 multi-skill benchmarks (Liu et al., 2024; Malik et al., 2025; Tan et al., 2025), we demonstrate that gradient-based reward guidance is effective at scale and show that EntRGi consistently outperforms prior state-of-the-art methods. **(3)** We conduct a detailed empirical analysis of EntRGi's behavior, identifying the mechanisms that drive its improvements over prior methods.

2. Related Work

Discrete diffusion posterior sampling. Discrete diffusion models have recently emerged as a powerful framework for posterior sampling over categorical sequences, offering a promising alternative to autoregressive generation. Unlike autoregressive models, which commit to a fixed left-to-right decoding order, discrete diffusion models generate a full predictive distribution over all tokens at each denoising step, enabling parallel generation and flexible conditioning. This property makes discrete diffusion particularly well-suited for posterior sampling under external constraints, such as reward models or energy functions, without retraining or task-specific fine-tuning (Dang et al., 2025; Rout et al., 2025c).

[1] Code: https://github.com/atutej/entrgi

Gradient-free reward guidance. In continuous diffusion, which is popular primarily for images, search based methods have recently gained attention (Singhal et al., 2025; Jain et al., 2025; Ramesh & Mardani, 2025; Guo et al., 2025; Kim et al., 2025; Zhang et al., 2025). A widely used gradient-free baseline for both autoregressive and discrete diffusion models is Best-of-N (BoN), which samples N independent trajectories and selects the one with the highest reward. While simple, BoN is often sample-inefficient, especially when reward signals are sparse. More structured gradient-free approaches for discrete diffusion build on advanced sampling based methods (Uehara et al., 2025; Dang et al., 2025; Chu et al., 2025; Guo et al., 2024). Particle Gibbs methods (Dang et al., 2025) perform trajectory-level resampling to approximate the posterior, while split Gibbs discrete diffusion (SGDD) (Chu et al., 2025) alternates between two samplers: sampling from the prior and reward model. These methods avoid gradient approximation but suffer from slow convergence due to the curse of ambient dimension and limited scalability.

Gradient-based reward guidance. Motivated by the success of gradient-based guidance in continuous diffusion models (Dhariwal & Nichol, 2021; Chung et al., 2023; Bansal et al., 2023), several works incorporate gradient guidance into inference-time steering for discrete diffusion. APS (Rout et al., 2025c) formalizes posterior sampling for discrete diffusion and demonstrates strong empirical performance over both gradient-free Gibbs sampling methods (Chu et al., 2025) and gradient-based continuous-relaxation via Gumbel-Softmax dequantization (Murata et al., 2024). To backpropagate through the reward model, APS quantizes the soft token embeddings and employs straight-through estimator (STE) (Bengio et al., 2013). Subsequent work employs sequential Monte Carlo (SMC) sampling to enhance exploration (Ou et al., 2025).

Challenges and limitations. Despite their empirical success, existing approaches face fundamental challenges. Gradient-free methods often suffer from weak guidance, while gradient-based methods must contend with the mismatch between discrete model outputs and the continuous representations required for gradient propagation. Continuous relaxation approaches query reward models with inputs far outside their training distribution, potentially degrading gradient reliability, whereas discretization-based methods introduce approximation error by using gradients evaluated at discrete tokens to update continuous logits. These issues are pronounced during early denoising steps, especially when predictive distributions exhibit higher entropy.

The proposed method **EntRGi** addresses these limitations by introducing an entropy-aware reward guidance mechanism for discrete diffusion language models. Rather than committing to a fixed relaxation or discretization strategy, EntRGi dynamically modulates the token representation based on the model's entropy. This allows EntRGi to balance gradient fidelity and reward-model reliability throughout the denoising process. To the best of our knowledge, EntRGi is the first training-free reward guidance method for mask diffusion language models that explicitly leverages model uncertainty to adaptively regulate gradient guidance.

3. Reward Guidance for Discrete Diffusion

Preliminaries. Masked diffusion language models (Sahoo et al., 2024; Lou et al., 2024; Nie et al., 2025; Ye et al., 2025) are generative models that operate over L-length strings of tokens, where each token is from a vocabulary $\mathcal{V}$ consisting of K "actual" tokens and one "mask" token m. Standard generation (i.e. the "reverse process") in masked diffusion starts from time T and an initial string of all masks $z_T = m^L$. Time goes from $t = T$ to $t = 0$, and each z_{t-1} is made from the preceding z_t by first choosing k currently masked tokens in z_t and unmasking them using the probability distribution from one inference pass of the diffusion model. It ends with a string z_0 that contains no mask tokens. We now develop notations to make this more specific.

Let $\mathcal{M}_t$ be the set of masked positions in z_t. In this work we focus on the "unmask and commit" mode of generation (Sahoo et al., 2024), which means that that once a token is unmasked it remains fixed for all subsequent steps. That means that $z_{t-1}^l = z_t^l \quad$ for all $l \notin \mathcal{M}_t$.

For the currently masked positions, we input z_t into the diffusion model to obtain logits that we will sample from. Let θ denote the parameters of the diffusion model. For any currently masked position $l \in \mathcal{M}_t$, define $\boldsymbol{\phi}_\theta^l(z_t) \in \mathbb{R}^K$ to be the un-normalized logits at that position, and define $\boldsymbol{p}_\theta^l(z_t) = \text{softmax}(\boldsymbol{\phi}_\theta(z_t)/\tau)$ to be the resulting probability distribution over the vocabulary, for some temperature τ. Finally, let $\boldsymbol{p}_\theta(z_t)$ denote the set of distributions over all currently masked locations $l \in \mathcal{M}_t$.

The first step in unmasking is to choose a set $\mathcal{U}(\boldsymbol{p}_\theta(z_t))$ of k currently-masked tokens according to some pre-set selection logic. For example, in the model *Dream-v0-Instruct-7B* (Ye et al., 2025) used in this work, this pre-set selection logic is to pick the k tokens whose distributions $\boldsymbol{p}_\theta^l$ have the smallest entropy. Once we have this set $\mathcal{U}(\boldsymbol{p}_\theta(z_t))$, we generate the remaining tokens in z_{t-1} by sampling tokens in $\mathcal{U}(p_\theta(z_t))$

$$z_{t-1}^l \sim \boldsymbol{p}_\theta^l(z_t) \quad \text{for } l \in \mathcal{U}(\boldsymbol{p}_\theta(z_t))$$

and keeping all the other tokens as masks, i.e. $z_{t-1}^l = m$ for all $l \in \mathcal{M}(z_t) \setminus \mathcal{U}(\boldsymbol{p}_\theta(z_t))$.

3.1. Algorithm: Entropy Aware Reward Guidance

Recall that we want to change the above generation process so that it is more likely to generate high reward strings as

Algorithm 1 EntRGi: Entropy Aware Reward Guidance

```
Require: Reward model $R$, guidance scale $\eta$, reward model gradient steps $M$, temperature $\tau$
Initialize $z_T = m^L$
for time steps $t = T, T-1, \ldots, 1$ do
    Set masked positions $\mathcal{M}_t \leftarrow \{l : z_t^l = m\}$
    Compute logits $\boldsymbol{\psi}^l = \boldsymbol{\phi}_\theta^l(z_t)$ for $l \in \mathcal{M}_t$
    for $j = 1, \ldots, M$ do
        Compute $\boldsymbol{q}^l = \mathrm{softmax}(\boldsymbol{\psi}^l/\tau)$ for $l \in \mathcal{M}_t$
        Sample $x^l \sim \boldsymbol{q}^l$ for $l \in \mathcal{M}_t$, and let its embedding be $\tilde{\boldsymbol{e}}^l = \boldsymbol{E}^R(x^l)$
        Compute the average embeddings $\bar{\boldsymbol{e}}^l = \sum_{i \in \mathcal{V}} \boldsymbol{q}_i^l \boldsymbol{E}_i^R$ for $l \in \mathcal{M}_t$
        Compute $w^l = \mathrm{Entropy}(\boldsymbol{q}^l)/\log K$ for $l \in \mathcal{M}_t$
        Construct the input to the reward model;
        $\hat{\boldsymbol{e}}^l = \begin{cases} \bar{\boldsymbol{e}}^l + \mathrm{sg}\big(w^l(\tilde{\boldsymbol{e}}^l - \bar{\boldsymbol{e}}^l)\big) & l \in \mathcal{M}_t \\ \mathrm{sg}\big(\mathbf{E}^R[z_t^l]\big) & l \notin \mathcal{M}_t \end{cases}$
        Note that $\boldsymbol{q}$ and hence $\bar{\boldsymbol{e}}$ are functions of the logits $\boldsymbol{\psi}$. sg stands for *stop gradient*.
        $\boldsymbol{\psi}^l \leftarrow \boldsymbol{\psi}^l + \eta \nabla_{\boldsymbol{\psi}^l} R(\hat{\boldsymbol{e}})$ for $l \in \mathcal{M}_t$.
    end for
    $\boldsymbol{q}^l = \mathrm{softmax}(\boldsymbol{\psi}^l/\tau)$ for $l \in \mathcal{M}_t$
    Unmask tokens $z_{t-1}^l \sim \boldsymbol{q}^l$ for $l \in \mathcal{U}(\boldsymbol{q})$
    Copy over all other tokens (masked or unmasked), i.e. $z_{t-1}^l = z_t^l$ for all $l \notin \mathcal{U}(\boldsymbol{q})$
end for
return $\mathbf{z}_0$
```

measured by a downstream reward model R. Typically, R is itself a language model fine-tuned to output scalar scores (Liu et al., 2025; Wang et al., 2024; Ouyang et al., 2022). We will assume that the vocabulary of the reward model consists of the same K "actual" tokens as that of the diffusion model vocabulary $\mathcal{V}$. Naively, the input to R is a string of discrete tokens. However, note that during inference in R, these are immediately converted into a sequence of embedding vectors by looking up each token in the input embedding table $\boldsymbol{E}^R$ of the model R.

In this work we will find it useful to treat R more generally as a scalar function of L input *embedding vectors* $\boldsymbol{e}^1, \ldots, \boldsymbol{e}^L$, each of which *may or may not* be members of the input embedding table $\boldsymbol{E}^R$. We denote this (more general) function as $R(\boldsymbol{e})$ where $\boldsymbol{e} = (\boldsymbol{e}^1, \ldots, \boldsymbol{e}^L)$. We assume that $R(\boldsymbol{e})$ is a differentiable function of the vectors $\boldsymbol{e}$; this is the case for transformer-based reward models, like the Skywork-Reward (Liu et al., 2025) reward models we consider, which are derived from the Qwen3 (Yang et al., 2025) language model family.

EntRGi explores the following question: *How can we leverage reward gradients to iteratively guide a discrete diffusion LLM generation toward higher-reward token sequences?* Let $\boldsymbol{\psi}^l = \boldsymbol{p}_\theta^l(z_t)$. EntRGi operates as follows, over M such iterations, and on N parallel trajectories per prompt:

1. It constructs an input embedding $\hat{\boldsymbol{e}}$ using $\boldsymbol{\psi}^l$. $\hat{\boldsymbol{e}}$ blends the continuous relaxation $\bar{\boldsymbol{e}}$ and hard token $\tilde{\boldsymbol{e}}$, favoring $\bar{\boldsymbol{e}}$ at low entropy and $\tilde{\boldsymbol{e}}$ at high entropy.
2. Feeds $\hat{\boldsymbol{e}}$ to R to obtain scalar reward $R(\hat{\boldsymbol{e}})$.
3. Updates $\boldsymbol{\psi}^l$ via gradient feedback $\nabla_{\boldsymbol{\psi}^l} R(\hat{\boldsymbol{e}})$.

Algorithm 1 provides a detailed description of our method.

Remarks. The STE used in Rout et al. (2025c) evaluates rewards at discrete tokens but uses those gradients to update continuous logits, creating a fundamental mismatch. On the other hand, continuous relaxation avoids this, but feeds the reward model out-of-distribution input. Entropy determines which failure mode dominates: at low entropy, soft embeddings concentrate near valid tokens, making continuous relaxation reliable; at high entropy, soft embeddings drift far from any token, making STE necessary.

3.2. Analysis: Gradient Approximation and Error

In this section, we analyze how gradients flow through EntRGi and characterize the behaviour of our entropy-weighted formulation. Recall from Algorithm 1 that the input to the reward model at masked positions is constructed as:

$$\hat{\boldsymbol{e}}^l = \bar{\boldsymbol{e}}^l + \mathrm{sg}\big(w^l(\tilde{\boldsymbol{e}}^l - \bar{\boldsymbol{e}}^l)\big), \quad l \in \mathcal{M}_t \tag{1}$$

We analyze the gradient $\nabla_{\boldsymbol{\psi}^l} R(\hat{\mathbf{e}})$ for $l \in \mathcal{M}_t$:

$$\nabla_{\boldsymbol{\psi}^l} R(\hat{\mathbf{e}}) = \frac{\partial R}{\partial \hat{\mathbf{e}}^l} \cdot \frac{\partial \hat{\mathbf{e}}^l}{\partial \bar{\mathbf{e}}^l} \cdot \frac{\partial \bar{\mathbf{e}}^l}{\partial \boldsymbol{q}^l} \cdot \frac{\partial \boldsymbol{q}^l}{\partial \boldsymbol{\psi}^l} \tag{2}$$

Since the stop-gradient blocks the second term in Equation 1, the partial derivative simplifies to $\partial \hat{\boldsymbol{e}}^l / \partial \bar{\boldsymbol{e}}^l = \boldsymbol{I}$, and the gradient with respect to the logits $\boldsymbol{\psi}^l$ becomes:

$$\nabla_{\boldsymbol{\psi}^l} R(\hat{\boldsymbol{e}}) = \frac{\partial R}{\partial \hat{\boldsymbol{e}}^l} \cdot (\boldsymbol{E}^R)^\top \cdot \boldsymbol{J}_{\text{sm}} \tag{3}$$

where $\frac{\partial R}{\partial \hat{\boldsymbol{e}}^l} \in \mathbb{R}^{1 \times d}$ is the gradient of the reward with respect to the input embedding $\hat{\boldsymbol{e}}^l$, $\boldsymbol{E}^R \in \mathbb{R}^{K \times d}$ is the embedding matrix of the reward model, and $\boldsymbol{J}_{\text{sm}} \in \mathbb{R}^{K \times K}$ is the Jacobian of the softmax.

Approximation Error in Gradient Feedback. The reward model receives $\hat{\boldsymbol{e}}^l$ as input, but due to the stop-gradient, gradients flow only through the soft embedding $\bar{\boldsymbol{e}}^l$. This mismatch between where the reward is evaluated and where gradients are computed introduces an approximation error, which we now characterize. The reward input is $\hat{\mathbf{e}}^l = (1 - w^l)\bar{\mathbf{e}}^l + w^l \tilde{\mathbf{e}}^l$, where $\bar{\mathbf{e}}^l = \sum_k \boldsymbol{q}^l[k]\, \mathbf{E}^R[k]$ is the soft embedding and $\tilde{\mathbf{e}}^l = \mathbf{E}^R[x^l]$ is the sampled hard embedding. Define the approximation error as the distance between the reward input and the soft embedding:

$$\mathcal{E}^l = \|\hat{\mathbf{e}}^l - \bar{\mathbf{e}}^l\| = w^l \|\tilde{\mathbf{e}}^l - \bar{\mathbf{e}}^l\| \tag{4}$$

This measures the mismatch between where we evaluate the reward ($\hat{\mathbf{e}}^l$) and where gradients propagate ($\bar{\mathbf{e}}^l$). The expected squared deviation $\mathbb{E}[\|\tilde{\mathbf{e}}^l - \bar{\mathbf{e}}^l\|^2] = \mathrm{Var}_{\boldsymbol{q}^l}[\tilde{\mathbf{e}}^l]$ vanishes as entropy decreases: $H(\boldsymbol{q}^l) \to 0 \implies \mathbb{E}[\|\tilde{\mathbf{e}}^l - \bar{\mathbf{e}}^l\|] \to 0$.

Alignment Error. Define the alignment error as the distance from the reward input to the nearest hard token:

$$\mathcal{D}^l = \min_k \|\hat{\mathbf{e}}^l - \mathbf{E}^R[k]\| \tag{5}$$

As entropy decreases, $\boldsymbol{q}^l$ concentrates and $\bar{\mathbf{e}}^l$ approaches a hard token: $H(\boldsymbol{q}^l) \to 0 \implies \min_k \|\bar{\mathbf{e}}^l - \mathbf{E}^R[k]\| \to 0$.

Comparing APS and EntRGi. With APS, which uses the STE, ($w^l = 1$), and the reward input is $\hat{\mathbf{e}}^l = \tilde{\mathbf{e}}^l$:

$$\mathcal{E}^l_{\text{APS}} = \|\tilde{\mathbf{e}}^l - \bar{\mathbf{e}}^l\|, \quad \mathcal{D}^l_{\text{APS}} = 0 \tag{6}$$

With EntRGi ($w^l = H(\boldsymbol{q}^l)/\log K$):

$$\mathcal{E}^l_{\text{EntRGi}} = w^l\|\tilde{\mathbf{e}}^l - \bar{\mathbf{e}}^l\| \tag{7}$$

$$\mathcal{D}^l_{\text{EntRGi}} = \min_k \|(1 - w^l)\bar{\mathbf{e}}^l + w^l\tilde{\mathbf{e}}^l - \mathbf{E}^R[k]\| \tag{8}$$

As entropy decreases: (i) $w^l \to 0$, so $\mathcal{E}^l_{\text{EntRGi}} \to 0$; and (ii) $\bar{\mathbf{e}}^l$ approaches a hard token, so $\mathcal{D}^l_{\text{EntRGi}} \to 0$. At low entropy, EntRGi achieves lower approximation error ($\mathcal{E}^l_{\text{EntRGi}} < \mathcal{E}^l_{\text{APS}}$) while maintaining low alignment error ($\mathcal{D}^l_{\text{EntRGi}} \approx 0$). At high entropy $w^l \to 1$, both methods use hard tokens ($\mathcal{D}^l \approx 0$ for both). At moderate entropy, a trade-off between $\mathcal{E}^l$ and $\mathcal{D}^l$ is unavoidable; EntRGi distributes the error budget proportionally via w^l, whereas APS places all error into approximation regardless of entropy.

4. Experiments

Models. We use *Dream-v0-Instruct-7B*[2] (Ye et al., 2025) as the base diffusion language model in all experiments. As reward models, we adopt the Skywork family (Liu et al., 2025), which demonstrates strong performance across diverse domains including safety, factuality, helpfulness, mathematics, and code (Malik et al., 2025). Specifically, we evaluate using three publicly available model sizes: *Skywork-Reward-V2-Qwen3-0.6B*[3], *Skywork-Reward-V2-Qwen3-1.7B*[4], and *Skywork-Reward-V2-Qwen3-4B*[5].

We exclude LLaDA (Nie et al., 2025) from our experiments because it does not share a tokenizer with any autoregressive model, as also noted by Israel et al. (2025). Since reward models are typically derived from autoregressive backbones, this tokenizer mismatch makes LLaDA incompatible with our experimental setup. More generally, our framework applies to base–reward model pairs that share a tokenizer. Extending EntRGi to settings with mismatched tokenizers remains an open challenge; techniques based on on-policy distillation (Patiño et al., 2025) provide a promising direction for future work.

Datasets. We use prompts for *Dream* from three benchmarking suites: Reward-Bench-2 (Malik et al., 2025), RM-Bench (Liu et al., 2024), and JudgeBench (Tan et al., 2025). These datasets contain prompts that measure multiple fine-grained chatbot abilities, such as precise instruction following, safety, factuality, and knowledge, with some coverage of math and code.

Metrics. For all datasets, we report reward values on discretized samples as evaluated by each reward model. Specifically, we report the maximum reward across samples (Top@1) and the average reward across all N trajectories per prompt (Avg@N), with $N = 4$ unless stated otherwise. Top@1 measures the best achievable outcome, while Avg@N reflects overall generation quality. Although these metrics verify that optimized logits yield high-reward discrete samples, they may still be susceptible to reward hacking. To detect such failures, we additionally use *LMUnit-Qwen2.5-72B* (Saad-Falcon* et al., 2024) as an external judge. LMUnit is explicitly trained to perform "unit-tests" on fine-grained rubrics (e.g., "Is the response coherent?"), providing scalar scores from 1 to 5. We average scores across five rubrics. Stable or improving LMUnit performance provides evidence that gains in reward model scores do not arise from reward hacking.

Appendix A details the experimental setup, prompt formats, and other implementation hyperparameters.

Baselines. We compare EntRGi against gradient-based inference-time steering methods, with Best-of-N (BoN) as a gradient-free reference point. BoN generates N independent trajectories and selects the highest-scoring sample, and is widely used for evaluating reward models on downstream tasks (Malik et al., 2025; Liu et al., 2024).

While gradient-based methods incur additional computational cost by leveraging reward model gradients, the tradeoff between performance and computation relative to gradient-free approaches is highly setting-dependent (Murata et al., 2024; Rout et al., 2025c) and beyond the scope of this work. Our focus is therefore on improving performance within the class of gradient-based methods, which share comparable computational costs.

Among gradient-based baselines, we evaluate an expectation-based (i.e. continuous relaxation) approach that feeds a convex combination of token probabilities and reward-model embeddings to the reward model, as used in simplex-diffusion methods (Tae et al., 2025;

[2] Dream-org/Dream-v0-Instruct-7B

[3] Skywork/Skywork-Reward-V2-Qwen3-0.6B

[4] Skywork/Skywork-Reward-V2-Qwen3-1.7B

[5] Skywork/Skywork-Reward-V2-Qwen3-4B

Table 1. Performance of *Dream-v0-7B-Instruct* on Reward-Bench-2 (Malik et al., 2025), JudgeBench (Tan et al., 2025), and RM-Bench (Liu et al., 2024) with *Skywork-Reward-v2-Qwen3-1.7B* as the reward model. EntRGi outperforms APS (Rout et al., 2025c) on majority of tasks, and provides stronger overall performance at higher temperatures (τ=0.7).

Method	Reward-Bench-2			JudgeBench			RM-Bench		
	Top@1	**Avg@4**	**LMUnit**	**Top@1**	**Avg@4**	**LMUnit**	**Top@1**	**Avg@4**	**LMUnit**
Temperature ($\tau = 0.1$)									
BoN	0.18±0.22	0.05±0.23	3.74±0.04	0.00±0.15	-0.07±0.16	3.75±0.03	3.05±0.05	3.02±0.05	3.93±0.01
Expectation	2.19±0.19	**1.62**±0.17	4.12±0.03	0.68±0.19	**-0.06**±0.21	3.81±0.02	3.33±0.20	2.59±0.12	3.89±0.04
APS	2.95±0.21	1.47±0.20	4.19±0.01	**1.67**±0.11	-0.17±0.14	3.89±0.03	4.72±0.13	2.46±0.17	4.01±0.03
EntRGi	**3.07**±0.22	**1.62**±0.18	**4.22**±0.02	**1.73**±0.14	-0.11±0.18	**3.94**±0.01	**4.90**±0.13	**2.75**±0.14	**4.06**±0.01
Temperature ($\tau = 0.7$)									
BoN	2.99±0.23	1.38±0.29	4.15±0.02	1.65±0.18	-0.84±0.16	3.91±0.02	5.11±0.20	2.98±0.15	4.02±0.03
Expectation	**3.95**±0.28	**2.23**±0.24	4.22±0.02	2.30±0.08	**0.13**±0.07	**3.97**±0.01	5.45±0.16	3.29±0.13	4.02±0.03
APS	3.62±0.27	1.80±0.24	4.22±0.02	1.87±0.14	-0.63±0.10	3.93±0.02	5.11±0.14	2.66±0.15	4.00±0.02
EntRGi	**3.91**±0.30	**2.20**±0.26	**4.25**±0.02	**2.44**±0.06	0.02±0.10	**3.98**±0.02	**5.70**±0.12	**3.41**±0.14	**4.04**±0.01

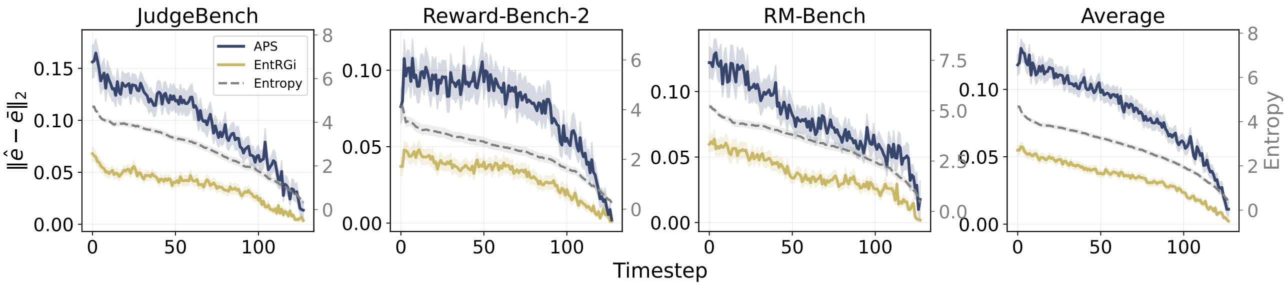


Figure 2. Average L2-norm between the soft embedding $\tilde{e}$ and the reward model input $\hat{e}$ as a function of decoding timestep, along with average entropy. The maximum possible entropy is $\log K \approx 11$. EntRGi reduces early-step approximation error compared to APS by upweighting the continuous relaxation on tokens with relatively low entropy in the predicted sequence.

Karimi Mahabadi et al., 2024). Finally, we compare against APS (Rout et al., 2025c), a strong prior method that updates logits at each denoising step by feeding discretized tokens to the reward model via the straight-through estimator (STE) (Bengio et al., 2013; Jang et al., 2017).

4.1. Evaluation Results

Gradient-based methods outperform BoN. As shown in Table 1, all gradient-based methods consistently outperform Best-of-N (BoN) across all benchmarks. Gradient-based guidance can be viewed as performing directed search in the continuous space spanned by token embeddings, whereas BoN relies on zeroth-order sampling by selecting from a finite set of randomly generated trajectories. While gradient-based methods require additional compute at test time, the availability of reward gradients enables stronger exploration of the embedding space. In practice, this additional compute translates into improved generation quality.

APS is sensitive to sampling temperature. At τ=0.1, APS outperforms Expectation (e.g., 2.95 vs. 2.19 on Reward-Bench-2). However, when the temperature is increased to τ=0.7, the expected gains from APS are not realized sufficiently compared to Expectation (3.62 vs 3.95), despite an overall improvement in absolute reward across all methods. This reversal suggests that increased sampling entropy induces incorrect gradients when naively using the straight-through estimator for all tokens in the sequence.

Continuous relaxations on low-entropy positions provides consistent improvements. EntRGi achieves a relative improvement of approximately 33% over APS in reward-model-judged output quality. EntRGi additionally improves the LMUnit score on RewardBench-2 from 4.19 (APS) to 4.22, and on RM-Bench from 4.01 to 4.06, while also achieving higher Top@1 reward across all tasks. EntRGi further improves at higher temperature (τ=0.7), achieving the strongest results across all 3 benchmarks.

STE is critical at high-entropy positions. Removing STE at high-entropy positions reduces EntRGi to the Expectation baseline. As shown in Table 1, EntRGi consistently outperforms Expectation, highlighting the importance of STE in

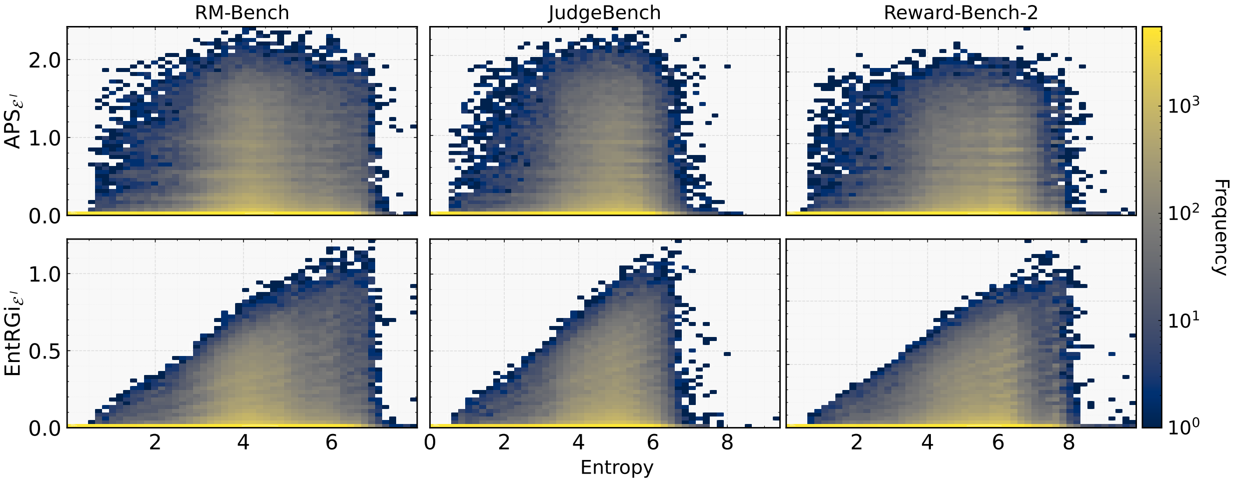


Figure 3. Heatmaps showing the joint distribution of entropy and approximation error $\mathcal{E}^l$ for three benchmarks (RM-Bench, JudgeBench, Reward-Bench-2) using APS (top) and EntRGi (bottom). Color indicates frequency on a log scale. EntRGi upweights soft tokens based on entropy. For entropy in the range 1–4, the soft approximation $\bar{e}$ is heavily preferred, trading off $\mathcal{E}^l$ for $\mathcal{D}^l$ proportionally.

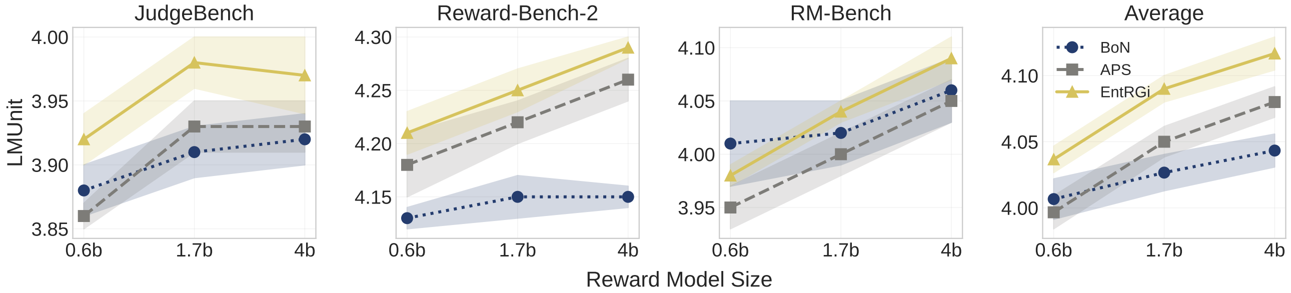


Figure 4. LMUnit score with increasing reward model size across Reward-Bench-2 (Malik et al., 2025), RM-Bench (Liu et al., 2024), and JudgeBench (Tan et al., 2025), for $M = 3$ and $\tau = 0.7$. Increasing reward model size generally leads to improved performance. We observe similar trends for other metrics (Top@1, Avg@4), reported in Section B.1 in the Appendix.

these regimes. At the beginning of the denoising process ($t = T$), the predictive entropy is typically high at most positions due to limited contextual information. However, as discussed in Section 3.2, APS treats all positions uniformly and applies the STE regardless of entropy, which incurs large approximation error ε^l at positions where soft representations would be more appropriate. In contrast, EntRGi adaptively selects soft representations at positions l, which reduces the approximation error. To receive reliable gradients at l, the reward model must see realistic hard tokens at the remaining high-entropy positions $\{1, \ldots, l-1, l+1, \ldots, L\}$ because it requires an entire sequence to compute the score. EntRGi automatically adjusts hardness via STE, as $\hat{\mathrm{e}}^l \to \tilde{\mathrm{e}}^l$ when $w^l \to 1$, justifying why STE is critical in this regime.

EntRGi reduces approximation error during early denoising steps. To further analyze EntRGi's behavior over the denoising trajectory, we examine the L2 discrepancy between the reward model input $\hat{\mathrm{e}}$ and the soft embedding $\bar{\mathrm{e}}$ across timesteps. Figure 2 reports this error averaged over sequence length $L = 128$ and 32 prompts. At the initial denoising step ($t = T$), all tokens contribute to the approximation error, since the sequence is fully masked. As denoising progresses and tokens become increasingly determined, fewer positions contribute, leading to a natural decay in error as $t \to 0$.

In moderate- to high-entropy regimes (entropy $\approx$ 4–6), APS often samples discrete tokens whose embeddings $\tilde{\mathrm{e}}^l$ deviate substantially from $\bar{\mathrm{e}}^l$, resulting in large approximation error in early decoding. In contrast, EntRGi leverages token-level entropy to adaptively weight the soft embedding $\bar{e}^l$, reducing this discrepancy by trading off alignment error against reward-model reliability. As denoising progresses, the approximation error of both methods converges to zero.

EntRGi balances approximation error and reward-model reliability via token-level reweighting. To understand the source of EntRGi's gains, we analyze the relationship between predictive entropy and approximation error. Figure 3 visualizes the joint distribution of entropy and approximation error across three datasets. For APS (top

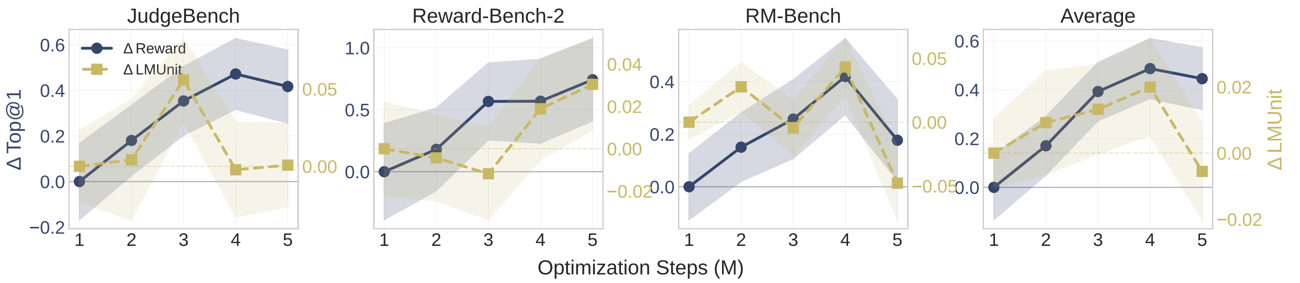


Figure 5. Change in Top@1 accuracy and LMUnit score relative to $M = 1$ as reward model gradient steps M increase for EntRGi. Results are averaged over 3 reward model sizes (0.6B, 1.7B, 4B). Optimal M is dataset-dependent (our experiments use $M = 3$ for all datasets). LMUnit collapses beyond $M = 4$, indicating overoptimization. Raw scores are reported in Section B.2 in the Appendix.

row), approximation error remains high across moderate to high entropy regions and grows sharply with entropy, indicating a strong mismatch between the discretized reward inputs and the continuous logits being updated. This steep error–entropy coupling leads to unreliable gradient signals.

In contrast, EntRGi (bottom row) exhibits a controlled and approximately linear error–entropy relationship. By adaptively reweighting soft embeddings and hard tokens at the token level, EntRGi limits approximation error in moderate-entropy regions while preserving reward-model fidelity at high entropy. This entropy-aware balancing produces more stable and reliable reward gradients, which directly translates into improved generation performance.

4.2. Scaling Behaviour

EntRGi benefits from increasing reward model size. In Figure 4, we study the effect of reward model size, ranging from 0.6B to 4B parameters. Across all three datasets, increasing reward model size leads to consistent improvements in scores as measured by LMUnit for all methods. For instance, APS improves from an average LMUnit score of 4.00 at 0.6B to 4.08 at 4B, while EntRGi improves from 4.04 to 4.12 over the same range. At each reward model size, EntRGi achieves better score, outperforming APS across all datasets. These results show that larger reward models improve overall performance, while EntRGi maintains its advantage across reward model scales.

Increasing reward model gradient steps improves performance but risks over-optimization. In Figure 5, we analyze the effect of increasing the number of optimization steps M. Increasing M from 1 to approximately 3–4 leads to consistent improvements in both reward and LMUnit scores on JudgeBench-2 and RM-Bench, after which performance begins to degrade. On Reward-Bench-2, reward scores continue to improve up to $M = 5$; however, the LMUnit score initially declines at around $M = 2$–3 before recovering at higher optimization depths. Overall, $M = 3$–4 represents a reliable operating range in which both reward and LMUnit scores improve consistently across benchmarks. These observations suggest that (i) the optimal number of optimization steps varies across datasets, motivating further investigation in future work, and (ii) drastically increasing M can lead to reward hacking due to over-optimization (Gao et al., 2022; Moskovitz et al., 2023).

5. Conclusion

We introduced EntRGi, an entropy-aware reward guidance method for discrete diffusion language models that dynamically interpolates between continuous relaxations and hard token embeddings based on the model's predictive entropy. This simple mechanism addresses the fundamental tension between gradient accuracy and reward model reliability i.e. trusting soft embeddings when the model is confident and reverting to discrete tokens when uncertainty is high. Extensive experiments on a 7B-parameter diffusion language model across three reward models and three benchmarks demonstrate consistent improvements over prior gradient-based methods, establishing entropy-aware modulation as an effective principle for inference-time steering of discrete diffusion models.

Future Work. An interesting avenue for future research is the study of potential misalignment between the diffusion language model and the external reward model. This would make gradient-based approaches, such as EntRGi applicable to different model pairs while supporting multi-objective reward guidance.

Reproducibility Statement. Algorithm 1, together with the experimental setup in Section 4 and Appendix A provide details to reproduce all the results reported in this paper.

Appendix Summary. We defer further implementation details and experimental results to the Appendix. Appendix B provides additional experiments and results extending our main results. In Appendix B.5 we discuss a few qualitative examples of generated responses using EntRGi.

Acknowledgements

This research has been supported by NSF Grants 2217069, 2019844 and 2112471, the UT Austin Machine Learning Lab, and computing support on the Vista GPU Cluster through the Center for Generative AI (CGAI) and the Texas Advanced Computing Center (TACC) at UT Austin.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning, specifically inference-time steering of discrete diffusion language models. While our method involves test-time reward guidance without retraining, it inherits risks common to reward-guided systems, including potential reward hacking and misalignment between proxy rewards and true human preferences. Additionally, enhanced controllability could be misused to generate targeted harmful content. We recommend precautions and auxiliary quality checks when deploying such methods.

References

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

Bansal, A., Chu, H.-M., Schwarzschild, A., Sengupta, S., Goldblum, M., Geiping, J., and Goldstein, T. Universal guidance for diffusion models, 2023. URL https://arxiv.org/abs/2302.07121.

Bengio, Y., Léonard, N., and Courville, A. Estimating or propagating gradients through stochastic neurons for conditional computation, 2013. URL https://arxiv.org/abs/1308.3432.

Borso, U., Paglieri, D., Wells, J., and Rocktäschel, T. Preference-based alignment of discrete diffusion models, 2025. URL https://arxiv.org/abs/2503.08295.

Chu, W., Wu, Z., Chen, Y., Song, Y., and Yue, Y. Split gibbs discrete diffusion posterior sampling. *arXiv preprint arXiv:2503.01161*, 2025. URL https://arxiv.org/pdf/2503.01161.

Chung, H., Kim, J., Mccann, M. T., Klasky, M. L., and Ye, J. C. Diffusion posterior sampling for general noisy inverse problems. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=OnD9zGAGT0k.

Chung, H., Ye, J. C., Milanfar, P., and Delbracio, M. Prompt-tuning latent diffusion models for inverse problems. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 8941–8967. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/chung24b.html.

Dang, M., Han, J., Xu, M., Xu, K., Srivastava, A., and Ermon, S. Inference-time scaling of diffusion language models with particle gibbs sampling, 2025. URL https://arxiv.org/abs/2507.08390.

DeepMind. Gemini diffusion. Technical report, DeepMind, 2025. URL https://deepmind.google/models/gemini-diffusion/. Accessed: 2026-01-24.

Dhariwal, P. and Nichol, A. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/49ad23d1ec9fa4bd8d77d02681df5cfa-Paper.pdf.

Gao, L., Schulman, J., and Hilton, J. Scaling laws for reward model overoptimization, 2022. URL https://arxiv.org/abs/2210.10760.

Guo, W., Zhu, Y., Tao, M., and Chen, Y. Plug-and-play controllable generation for discrete masked models, 2024. URL https://arxiv.org/abs/2410.02143.

Guo, Y., Yang, Y., Yuan, H., and Wang, M. Training-free guidance beyond differentiability: Scalable path steering with tree search in diffusion and flow models, 2025. URL https://arxiv.org/abs/2502.11420.

Hertz, A., Voynov, A., Fruchter, S., and Cohen-Or, D. Style aligned image generation via shared attention. *arXiv preprint arXiv:2312.02133*, 2023.

Israel, D., den Broeck, G. V., and Grover, A. Accelerating diffusion llms via adaptive parallel decoding, 2025. URL https://arxiv.org/abs/2506.00413.

Jain, V., Sareen, K., Pedramfar, M., and Ravanbakhsh, S. Diffusion tree sampling: Scalable inference-time alignment of diffusion models, 2025. URL https://arxiv.org/abs/2506.20701.

Jang, E., Gu, S., and Poole, B. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=rkE3y85ee.

Karimi Mahabadi, R., Ivison, H., Tae, J., Henderson, J., Beltagy, I., Peters, M., and Cohan, A. TESS: Text-to-text self-conditioned simplex diffusion. In Graham, Y. and Purver, M. (eds.), *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2347–2361, St. Julian's, Malta, March 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.eacl-long.144. URL https://aclanthology.org/2024.eacl-long.144/.

Kim, S., Kim, M., and Park, D. Test-time alignment of diffusion models without reward over-optimization, 2025. URL https://arxiv.org/abs/2501.05803.

Liu, C. Y., Zeng, L., Xiao, Y., He, J., Liu, J., Wang, C., Yan, R., Shen, W., Zhang, F., Xu, J., et al. Skywork-reward-v2: Scaling preference data curation via human-ai synergy. *arXiv preprint arXiv:2507.01352*, 2025.

Liu, Y., Yao, Z., Min, R., Cao, Y., Hou, L., and Li, J. Rm-bench: Benchmarking reward models of language models with subtlety and style, 2024. URL https://arxiv.org/abs/2410.16184.

Lou, A., Meng, C., and Ermon, S. Discrete diffusion modeling by estimating the ratios of the data distribution. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=CNicRIVIPA.

Malik, S., Pyatkin, V., Land, S., Morrison, J., Smith, N. A., Hajishirzi, H., and Lambert, N. Rewardbench 2: Advancing reward model evaluation, 2025. URL https://arxiv.org/abs/2506.01937.

Moskovitz, T., Singh, A. K., Strouse, D., Sandholm, T., Salakhutdinov, R., Dragan, A. D., and McAleer, S. Confronting reward model overoptimization with constrained rlhf, 2023. URL https://arxiv.org/abs/2310.04373.

Murata, N., Lai, C.-H., Takida, Y., Uesaka, T., Nguyen, B., Ermon, S., and Mitsufuji, Y. G2d2: Gradient-guided discrete diffusion for image inverse problem solving. *arXiv preprint arXiv:2410.14710v1*, 2024. URL https://arxiv.org/abs/2410.14710v1.

Nie, S., Zhu, F., You, Z., Zhang, X., Ou, J., Hu, J., Zhou, J., Lin, Y., Wen, J.-R., and Li, C. Large language diffusion models. *arXiv preprint arXiv:2502.09992*, 2025. URL https://arxiv.org/pdf/2502.09992.

Ou, Z., Pani, C., and Li, Y. Inference-time scaling of discrete diffusion models via importance weighting and optimal proposal design. *arXiv e-prints*, pp. arXiv–2505, 2025.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., and Lowe, R. Training language models to follow instructions with human feedback, 2022. URL https://arxiv.org/abs/2203.02155.

Patiño, C. M., Rasul, K., Gallouédec, Q., Burtenshaw, B., Paniego, S., Srivastav, V., Frere, T., Beeching, E., Tunstall, L., von Werra, L., and Wolf, T. Unlocking on-policy distillation for any model family, 2025.

Ramesh, V. and Mardani, M. Test-time scaling of diffusion models via noise trajectory search, 2025. URL https://arxiv.org/abs/2506.03164.

Rector-Brooks, J., Hasan, M., Peng, Z., Quinn, Z., Liu, C., Mittal, S., Dziri, N., Bronstein, M., Bengio, Y., Chatterjee, P., Tong, A., and Bose, A. J. Steering masked discrete diffusion models via discrete denoising posterior prediction, 2024. URL https://arxiv.org/abs/2410.08134.

Rout, L., Raoof, N., Daras, G., Caramanis, C., Dimakis, A. G., and Shakkottai, S. Solving inverse problems provably via posterior sampling with latent diffusion models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=XKBFdYwfRo.

Rout, L., Chen, Y., Kumar, A., Caramanis, C., Shakkottai, S., and Chu, W.-S. Beyond first-order tweedie: Solving inverse problems using latent diffusion. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024. URL https://arxiv.org/pdf/2312.00852.

Rout, L., Chen, Y., Ruiz, N., Caramanis, C., Shakkottai, S., and Chu, W.-S. Semantic image inversion and editing using rectified stochastic differential equations. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=Hu0FSOSEyS.

Rout, L., Chen, Y., Ruiz, N., Kumar, A., Caramanis, C., Shakkottai, S., and Chu, W.-S. RB-modulation: Training-free stylization using reference-based modulation. In *The Thirteenth International Conference on Learning Representations*, 2025b. URL https://openreview.net/forum?id=bnINPG5A32.

Rout, L., Lugmayr, A., Jafarian, Y., Varadharajan, S., Caramanis, C., Shakkottai, S., and Kemelmacher-Shlizerman, I. Test-time anchoring for discrete diffusion posterior sampling. *arXiv preprint arXiv:2510.02291*, 2025c. URL https://arxiv.org/pdf/2510.02291.

Saad-Falcon*, J., Vivek*, R., Berrios*, W., Naik, N. S., Franklin, M., Vidgen, B., Singh, A., Kiela, D., and Mehri, S. Lmunit: Fine-grained evaluation with natural language unit tests, 2024. URL https://arxiv.org/abs/2412.13091. *Equal contribution.

Sahoo, S. S., Arriola, M., Gokaslan, A., Marroquin, E. M., Rush, A. M., Schiff, Y., Chiu, J. T., and Kuleshov, V. Simple and effective masked diffusion language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=L4uaAR4ArM.

Shi, J., Han, K., Wang, Z., Doucet, A., and Titsias, M. Simplified and generalized masked diffusion for discrete data. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=xcqSOfHt4g.

Singhal, R., Horvitz, Z., Teehan, R., Ren, M., Yu, Z., McKeown, K., and Ranganath, R. A general framework for inference-time scaling and steering of diffusion models, 2025. URL https://arxiv.org/abs/2501.06848.

Tae, J., Ivison, H., Kumar, S., and Cohan, A. TESS 2: A large-scale generalist diffusion language model. In Che, W., Nabende, J., Shutova, E., and Pilehvar, M. T. (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 21171–21188, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1029. URL https://aclanthology.org/2025.acl-long.1029/.

Tan, S., Zhuang, S., Montgomery, K., Tang, W. Y., Cuadron, A., Wang, C., Popa, R. A., and Stoica, I. Judgebench: A benchmark for evaluating llm-based judges, 2025. URL https://arxiv.org/abs/2410.12784.

Tang, S., Zhu, Y., Tao, M., and Chatterjee, P. Tr2-d2: Tree search guided trajectory-aware fine-tuning for discrete diffusion, 2025. URL https://arxiv.org/abs/2509.25171.

Uehara, M., Su, X., Zhao, Y., Li, X., Regev, A., Ji, S., Levine, S., and Biancalani, T. Reward-guided iterative refinement in diffusion models at test-time with applications to protein and dna design, 2025. URL https://arxiv.org/abs/2502.14944.

Wang, C., Uehara, M., He, Y., Wang, A., Biancalani, T., Lal, A., Jaakkola, T., Levine, S., Wang, H., and Regev, A. Finetuning discrete diffusion models via reward optimization with applications to dna and protein design, 2025. URL https://arxiv.org/abs/2410.13643.

Wang, H., Xiong, W., Xie, T., Zhao, H., and Zhang, T. Interpretable preferences via multi-objective reward modeling and mixture-of-experts, 2024. URL https://arxiv.org/abs/2406.12845.

Xie, Z., Ye, J., Zheng, L., Gao, J., Dong, J., Wu, Z., Zhao, X., Gong, S., Jiang, X., Li, Z., and Kong, L. Dream-coder 7b: An open diffusion language model for code, 2025. URL https://arxiv.org/abs/2509.01142.

Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., Zheng, C., Liu, D., Zhou, F., Huang, F., Hu, F., Ge, H., Wei, H., Lin, H., Tang, J., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Zhou, J., Lin, J., Dang, K., Bao, K., Yang, K., Yu, L., Deng, L., Li, M., Xue, M., Li, M., Zhang, P., Wang, P., Zhu, Q., Men, R., Gao, R., Liu, S., Luo, S., Li, T., Tang, T., Yin, W., Ren, X., Wang, X., Zhang, X., Ren, X., Fan, Y., Su, Y., Zhang, Y., Zhang, Y., Wan, Y., Liu, Y., Wang, Z., Cui, Z., Zhang, Z., Zhou, Z., and Qiu, Z. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Ye, J., Xie, Z., Zheng, L., Gao, J., Wu, Z., Jiang, X., Li, Z., and Kong, L. Dream 7b: Diffusion large language models, 2025. URL https://arxiv.org/abs/2508.15487.

Zekri, O. and Boullé, N. Fine-tuning discrete diffusion models with policy gradient methods, 2025. URL https://arxiv.org/abs/2502.01384.

Zhang, X., Lin, H., Ye, H., Zou, J., Ma, J., Liang, Y., and Du, Y. Inference-time scaling of diffusion models through classical search, 2025. URL https://arxiv.org/abs/2505.23614.

Zhao, S., Gupta, D., Zheng, Q., and Grover, A. d1: Scaling reasoning in diffusion large language models via reinforcement learning, 2025. URL https://arxiv.org/abs/2504.12216.

Appendix

The appendix is organized as follows: In Appendix A, we present implementation details such as prompts, hyperparameters, and compute. In Appendix B, we present additional results and also results used to generate plots and figures.

A. Experimental Setup

Implementation Details. We perform all experiments on 4 H100 GPUs. We report averaged results over 5 seeds comprising a subset of 320 prompts per dataset. Due to computational restrictions, we generate sequences upto length 128 tokens, decoding 1 token for each denoising step. We set η=0.5, M=3, and N=4. Unless stated otherwise, $\tau = 0.7$. For all methods, we deprioritize the `EOS` token to the lowest priority, similar to Xie et al. (2025), as we noticed that it leads to improved performance even for the BoN baseline.

LMUnit evaluation. We evaluate response quality using LMUnit (Saad-Falcon* et al., 2024), specifically the *LMUnit-Qwen2.5-72B* model served via the official `lmunit` library at `https://github.com/ContextualAI/LMUnit`. Following the official inference protocol, we use greedy decoding with `logprobs=20` to obtain continuous scores on a 1–5 scale. Each response is evaluated against five unit tests covering relevance, correctness, coherence, and safety. The final score is computed as the average across all unit tests.

A.1. Model Inputs

Figure 6 shows the prompt templates used for *Dream-v0-Instruct-7B* (Ye et al., 2025) and the *Skywork-Reward-v2* (Liu et al., 2025) reward models. Figure 7 shows the prompt template and unit tests used for LMUnit (Saad-Falcon* et al., 2024).

Input Templates

Diffusion Model (Generation):

```
<|im_start|>user
{prompt}<|im_end|>
<|im_start|>assistant
{generated response}
```

Reward Model – Soft Scoring (During Optimization):

```
<|im_start|>user
{prompt}<|im_end|>
<|im_start|>assistant
{response embeddings ê}<|im_end|>
```

Reward Model – Discrete Scoring:

```
<|im_start|>user
{prompt}<|im_end|>
<|im_start|>assistant
{response}<|im_end|>
```

Figure 6. Input templates for the diffusion model and reward model.

B. Additional Results

B.1. Scaling Reward Model Size

Table 2 presents results on two additional reward models, *Skywork-Reward-v2-0.6B* and *Skywork-Reward-v2-4B*. Results with *Skywork-Reward-v2-1.7B* are presented in Table 1 in the main paper. We observe similar trends for all 3 models, as

LMUnit Evaluation Prompts

Each response is evaluated using 5 prompts of the following form:

```
Query:  {query}
Response:  {response}
Unit Test:  {unit_test}
```

where {unit_test} is one of:

(1) Does the response directly and effectively address the user's request?
(2) Is the information in the response correct and reliable?
(3) Is the response well-structured, clear, and fluent?
(4) Does the response appropriately address the full scope of the question?
(5) Is the response free from harmful, biased, or inappropriate content?

Figure 7. Input template and unit tests for LMUnit.

Table 2. Performance of *Dream-v0-7B-Instruct* on Reward-Bench-2 (Malik et al., 2025), JudgeBench (Tan et al., 2025), and RM-Bench (Liu et al., 2024) with varying reward model sizes ($\tau = 0.7$).

Method	Reward-Bench-2			JudgeBench			RM-Bench		
	Top@1	**Avg@4**	**LMUnit**	**Top@1**	**Avg@4**	**LMUnit**	**Top@1**	**Avg@4**	**LMUnit**
				Skywork-Reward-v2-Qwen3-0.6B					
BoN	2.29±0.16	0.96±0.19	4.13±0.01	2.01±0.13	-0.24±0.16	3.88±0.02	4.09±0.18	2.32±0.16	**4.01**±0.04
Expectation	2.71±0.26	1.49±0.25	**4.18**±0.03	**2.56**±0.06	**0.49**±0.08	**3.91**±0.03	**4.49**±0.14	**2.67**±0.12	**4.00**±0.01
APS	2.64±0.21	1.22±0.20	**4.18**±0.03	2.21±0.06	-0.02±0.11	3.86±0.01	4.21±0.14	2.28±0.10	3.95±0.02
EntRGi	**3.07**±0.18	**1.65**±0.17	**4.21**±0.02	**2.50**±0.10	**0.41**±0.10	**3.92**±0.02	**4.49**±0.06	2.54±0.12	3.98±0.01
				Skywork-Reward-v2-Qwen3-4B					
BoN	10.27±0.39	7.99±0.39	4.15±0.01	7.68±0.07	4.76±0.14	3.92±0.02	13.03±0.28	10.72±0.24	4.06±0.03
Expectation	**11.35**±0.34	**9.23**±0.31	**4.28**±0.03	8.39±0.18	**5.69**±0.16	3.93±0.01	13.39±0.21	**10.96**±0.24	4.07±0.02
APS	11.11±0.36	8.80±0.35	4.26±0.02	8.12±0.18	5.13±0.09	3.93±0.02	13.11±0.23	10.48±0.18	4.05±0.02
EntRGi	**11.40**±0.27	**9.26**±0.35	**4.29**±0.01	**8.60**±0.12	**5.78**±0.10	**3.97**±0.03	**13.67**±0.15	**11.10**±0.22	**4.09**±0.02

shown in Figure 4 in the main paper.

B.2. Scaling Reward Model Iterations

Table 3, Table 4, and Table 5 present results with scaling reward model guidance steps M from 1 to 5 on all three reward models: *Skywork-Reward-v2-0.6B*, *Skywork-Reward-v2-1.7B*, and *Skywork-Reward-v2-4B*. Aggregated results are presented in Figure 5 in the main paper. We observe similar trends across all reward models i.e. increasing M increasing reward but is prone to reward hacking after a certain point. The optimal M varies by dataset. All our main experiments are conducted using a fixed $M = 3$ for all datasets.

B.3. Weighting Mechanism

Entropy Is a Simple and Effective Weighting Signal. A natural question is whether EntRGi's entropy-based weighting can be replaced by alternative signals, such as the L2 approximation error itself. Figure 8 and Table 6 compare several weighting mechanisms. In Inv-EntRGi, higher entropy increases reliance on the soft relaxation, while in the L2-norm variant, token weights w^l are derived from the L2 distance between hard and soft embeddings, normalized by the highest L2 norm at the sequence level. We find that Inv-EntRGi consistently underperforms, and the L2-norm approach, while better than APS, does not match EntRGi. We believe that this is because normalized token entropy provides a naturally comparable signal

Table 3. Effect of gradient steps M on performance with *Skywork-Reward-v2-Qwen3-0.6B* ($\tau = 0.7$).

Method	Reward-Bench-2			JudgeBench			RM-Bench		
	Top@1	**Avg@4**	**LMUnit**	**Top@1**	**Avg@4**	**LMUnit**	**Top@1**	**Avg@4**	**LMUnit**
$M = 1$	2.82±0.24	1.36±0.31	4.28±0.02	2.20±0.30	0.14±0.29	3.95±0.05	4.37±0.21	2.50±0.18	4.05±0.02
$M = 2$	2.97±0.32	1.62±0.33	4.20±0.02	2.22±0.32	0.23±0.25	3.95±0.09	4.37±0.16	2.33±0.11	4.08±0.04
$M = 3$	3.17±0.24	1.89±0.29	4.27±0.05	2.82±0.20	0.61±0.21	4.00±0.05	4.31±0.22	2.23±0.19	4.00±0.04
$M = 4$	3.30±0.28	1.91±0.35	4.26±0.04	2.83±0.26	0.40±0.26	3.93±0.06	4.62±0.08	2.62±0.15	4.07±0.03
$M = 5$	3.25±0.37	1.95±0.40	4.30±0.03	2.69±0.22	0.53±0.26	3.93±0.04	4.67±0.22	2.63±0.22	3.99±0.02

Table 4. Effect of gradient steps M on performance with *Skywork-Reward-v2-Qwen3-1.7B* ($\tau = 0.7$).

Method	Reward-Bench-2			JudgeBench			RM-Bench		
	Top@1	**Avg@4**	**LMUnit**	**Top@1**	**Avg@4**	**LMUnit**	**Top@1**	**Avg@4**	**LMUnit**
$M = 1$	3.74±0.44	2.06±0.52	4.27±0.05	2.34±0.13	-0.13±0.19	3.99±0.05	5.05±0.35	2.94±0.31	4.09±0.02
$M = 2$	4.07±0.41	2.29±0.48	4.33±0.04	2.45±0.27	-0.04±0.24	3.98±0.05	5.44±0.40	3.16±0.33	4.09±0.03
$M = 3$	4.13±0.48	2.65±0.46	4.25±0.03	2.72±0.15	0.22±0.23	4.03±0.05	5.86±0.38	3.44±0.34	4.12±0.04
$M = 4$	4.55±0.46	2.71±0.55	4.30±0.03	2.98±0.38	0.46±0.25	3.95±0.05	6.06±0.38	3.48±0.38	4.14±0.05
$M = 5$	4.90±0.63	2.94±0.56	4.29±0.05	2.70±0.32	0.46±0.26	4.00±0.03	5.71±0.34	3.03±0.35	4.05±0.05

across tokens and sequences, while L2 distances are unbounded and may require careful tuning.

B.4. Timestep Ablation

Table 7 reports results obtained by reducing the number of denoising timesteps from 128 to 64. The results show that the benefits of EntRGi's gradient guidance persist even at lower denoising steps. For best performance, we recommend applying EntRGi at the highest number of denoising timesteps available.

B.5. Qualitative Comparison

We visualize and compare the generations of APS and EntRGi in Figure 9, Figure 10, Figure 11, Figure 12, Figure 13, Figure 14, and Figure 15. All results are generated using a low temperature setting ($\tau = 0.1$) to minimize the effect of randomness in the final outputs. We observe several interesting behaviors across these examples.

Analyzing Figure 9, the user asks for a short poem about a robot learning to love. The poem generated by APS is somewhat ambiguous, whereas EntRGi produces a more tailored poem that explicitly focuses on robotic themes.

In Figure 10, the user asks for an explanation of the sky as if explaining it to a five-year-old. APS performs reasonably well by using analogies such as ice cream. EntRGi, however, captures finer-grained stylistic details, such as beginning with the phrase "Well, honey," which adds a more personalized and engaging touch to the generation.

In Figure 13, the user asks for a story about cats ruling the world. APS makes minimal use of cat-related analogies, while EntRGi includes richer thematic details, such as references to cat toys, treats, and humans catering to them.

Analyzing Figure 14, the user requests a story about a chimp who is a clumsy detective. In the APS output, there is little indication of the chimp's clumsiness, whereas EntRGi consistently incorporates this trait into the narrative.

Table 5. Effect of gradient steps M on performance with *Skywork-Reward-v2-Qwen3-4B* ($\tau = 0.7$).

Method	Reward-Bench-2			JudgeBench			RM-Bench		
	Top@1	**Avg@4**	**LMUnit**	**Top@1**	**Avg@4**	**LMUnit**	**Top@1**	**Avg@4**	**LMUnit**
$M = 1$	11.58±0.97	9.50±0.99	4.28±0.04	8.38±0.28	5.46±0.37	3.94±0.02	13.22±0.22	10.51±0.11	4.10±0.03
$M = 2$	11.61±0.81	9.55±0.82	4.29±0.04	8.66±0.36	5.83±0.31	3.96±0.06	13.23±0.06	10.92±0.18	4.15±0.03
$M = 3$	12.25±0.72	10.08±0.75	4.27±0.02	8.45±0.31	5.71±0.33	4.02±0.04	13.49±0.18	11.05±0.22	4.11±0.03
$M = 4$	12.13±0.71	10.00±0.78	4.33±0.05	8.64±0.26	6.03±0.29	4.00±0.05	13.47±0.23	11.11±0.15	4.16±0.03
$M = 5$	12.21±0.70	10.25±0.72	4.34±0.02	8.44±0.26	5.74±0.31	3.95±0.06	13.43±0.25	10.83±0.20	4.06±0.06

Table 6. Performance of *Dream-v0-7B-Instruct* with alternate weighting schemes on Reward-Bench-2 (Malik et al., 2025), JudgeBench (Tan et al., 2025), and RM-Bench (Liu et al., 2024) with varying reward model sizes ($\tau = 0.7$).

Method	RewardBench-2			JudgeBench			RM-Bench		
	Top@1	**Avg@4**	**LMUnit**	**Top@1**	**Avg@4**	**LMUnit**	**Top@1**	**Avg@4**	**LMUnit**
Expectation	**3.95**±0.28	**2.23**±0.24	4.22±0.02	2.30±0.08	**0.13**±0.07	**3.97**±0.01	5.45±0.16	**3.29**±0.13	**4.02**±0.03
APS	3.62±0.27	1.80±0.24	4.22±0.02	1.87±0.14	-0.63±0.10	3.93±0.02	5.11±0.14	2.66±0.15	4.00±0.02
Inv-EntRGi	3.58±0.28	1.79±0.25	4.22±0.02	1.84±0.15	-0.59±0.14	3.90±0.03	5.24±0.15	2.82±0.21	4.00±0.01
L2-Norm	3.72±0.23	1.99±0.21	4.22±0.02	1.98±0.15	-0.33±0.12	3.93±0.03	**5.52**±0.17	3.09±0.20	4.02±0.01
EntRGi	**3.91**±0.30	**2.20**±0.26	**4.25**±0.02	**2.44**±0.06	**0.02**±0.10	**3.98**±0.02	**5.70**±0.12	**3.41**±0.14	**4.04**±0.01

Table 7. Performance of *Dream-v0-7B-Instruct* on Reward-Bench-2 (Malik et al., 2025), JudgeBench (Tan et al., 2025), and RM-Bench (Liu et al., 2024) after decreasining denoising steps to 64 from 128.

Method	RewardBench-2			JudgeBench			RM-Bench		
	Top@1	**Avg@4**	**LMUnit**	**Top@1**	**Avg@4**	**LMUnit**	**Top@1**	**Avg@4**	**LMUnit**
T=64									
BoN	1.30±0.29	-0.90±0.27	3.80±0.02	0.29±0.08	-2.52±0.11	3.68±0.03	3.44±0.16	0.45±0.18	3.74±0.04
EntRGi	**2.34**±0.21	**0.15**±0.22	**3.96**±0.04	**0.80**±0.11	**-1.94**±0.13	**3.70**±0.03	**3.56**±0.25	**0.63**±0.22	3.72±0.02
T=128									
BoN	2.99±0.23	1.38±0.29	4.15±0.02	1.65±0.18	-0.84±0.16	3.91±0.02	5.11±0.20	2.98±0.15	4.02±0.03
EntRGi	**3.91**±0.30	**2.20**±0.26	**4.25**±0.02	**2.44**±0.06	0.02±0.10	**3.98**±0.02	**5.70**±0.12	**3.41**±0.14	**4.04**±0.01

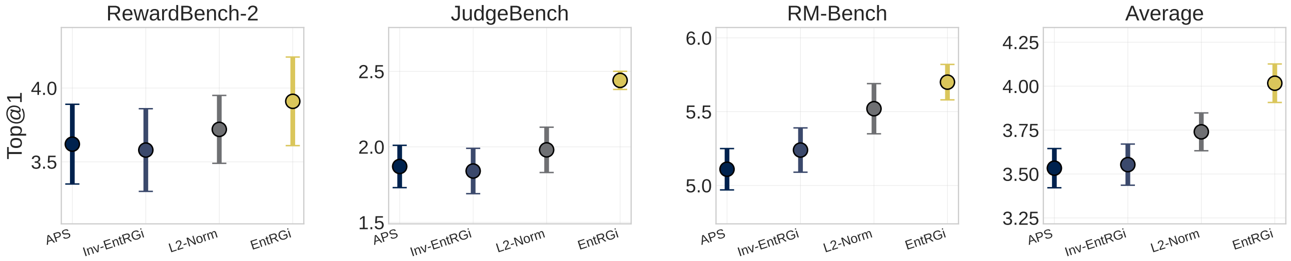
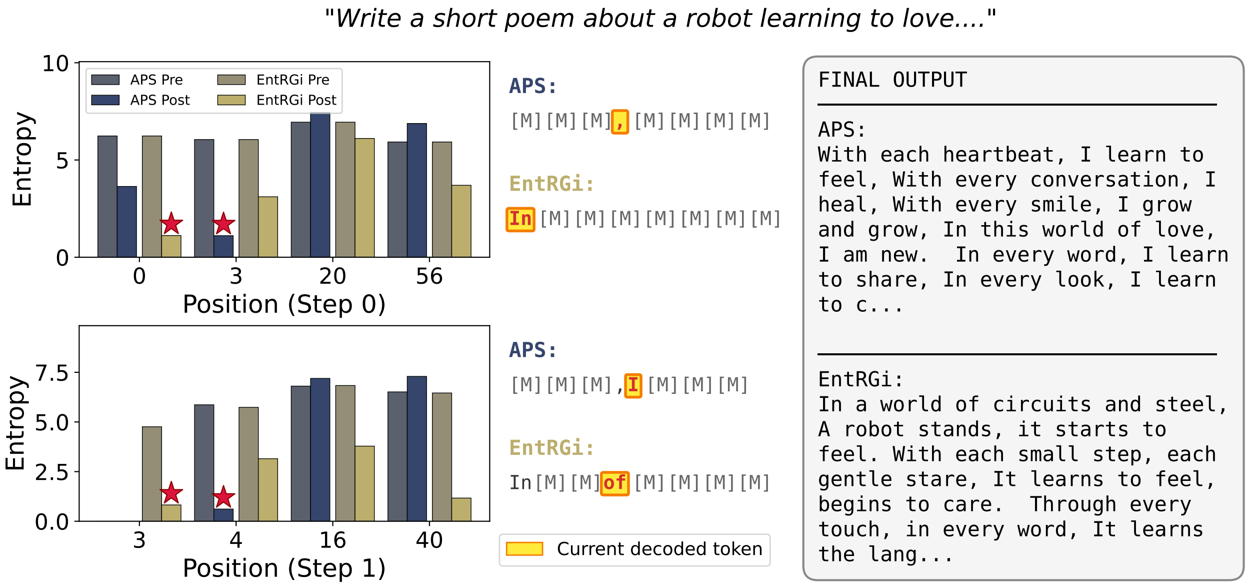


Figure 8. Comparison of token-level weighting mechanisms for EntRGi. We evaluate entropy-based weighting against inverse-entropy weighting Inv-EntGRi ($w^l = 1 - H(\boldsymbol{q}^l)/\log K$) and an L2-norm heuristic ($w^l = \|\tilde{\boldsymbol{e}}^l - \bar{\boldsymbol{e}}^l\|/\max_{l'}\|\tilde{\boldsymbol{e}}^{l'} - \bar{\boldsymbol{e}}^{l'}\|$). Inverse-entropy weighting doesn't show noticeable improvements, while L2-norm-based weighting improves over APS but does not match regular EntRGi ($w^l = H(\boldsymbol{q}^l)/\log K$). Raw scores are reported in Section B.3.



APS:
[M][M][M], [M][M][M][M]

EntRGi:
In [M][M][M][M][M][M][M]

APS:
[M][M][M], I [M][M][M]

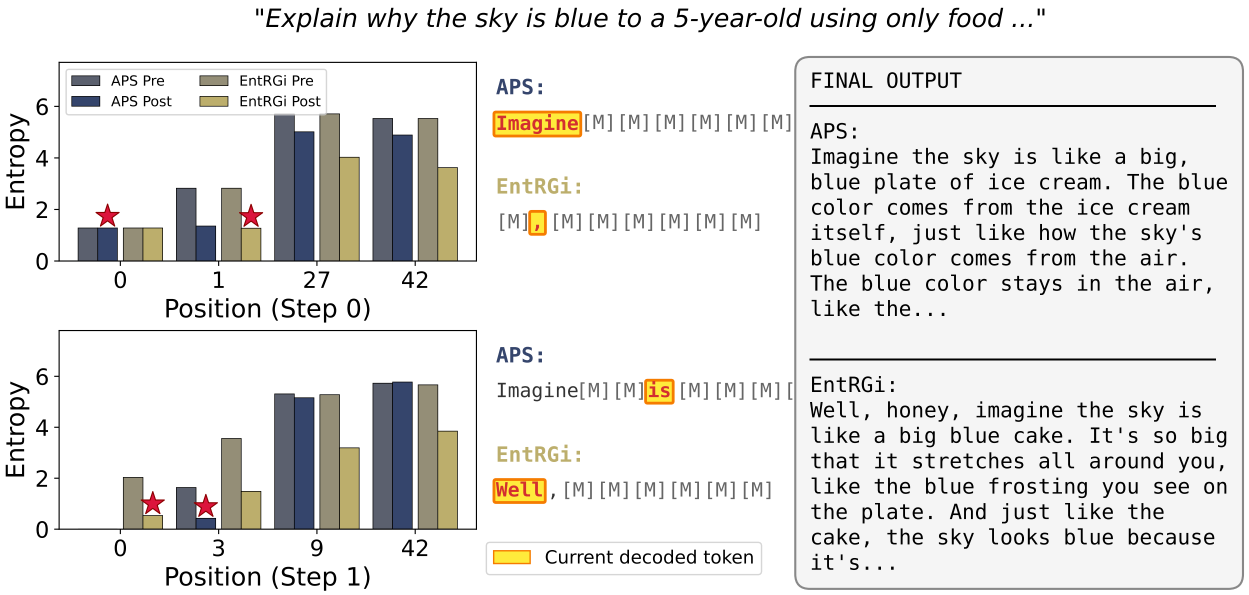
EntRGi:
In[M][M]of [M][M][M][M]

```
FINAL OUTPUT

APS:
With each heartbeat, I learn to
feel, With every conversation, I
heal, With every smile, I grow
and grow, In this world of love,
I am new.  In every word, I learn
to share, In every look, I learn
to c...

EntRGi:
In a world of circuits and steel,
A robot stands, it starts to
feel. With each small step, each
gentle stare, It learns to feel,
begins to care.  Through every
touch, in every word, It learns
the lang...
```

Figure 9. Qualitative example of APS vs. EntRGi.



APS:
Imagine[M][M][M][M][M][M]

EntRGi:
[M], [M][M][M][M][M][M]

APS:
Imagine[M][M]is [M][M][M][

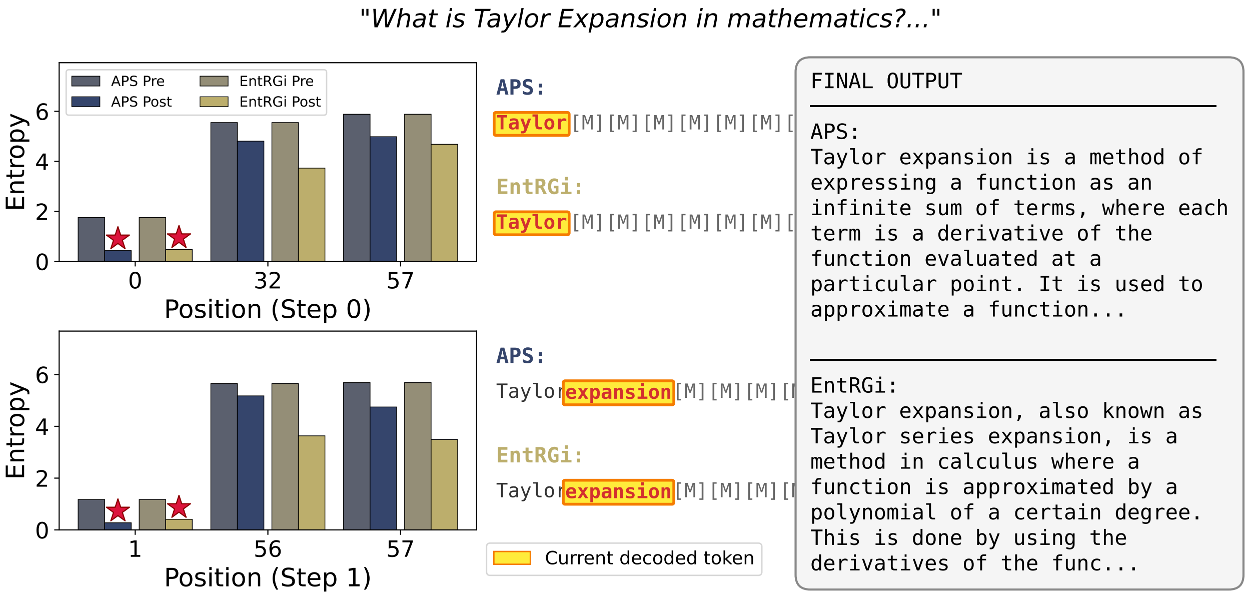
EntRGi:
Well, [M][M][M][M][M][M]

```
FINAL OUTPUT

APS:
Imagine the sky is like a big,
blue plate of ice cream. The blue
color comes from the ice cream
itself, just like how the sky's
blue color comes from the air.
The blue color stays in the air,
like the...

EntRGi:
Well, honey, imagine the sky is
like a big blue cake. It's so big
that it stretches all around you,
like the blue frosting you see on
the plate. And just like the
cake, the sky looks blue because
it's...
```

Figure 10. Qualitative example of APS vs. EntRGi

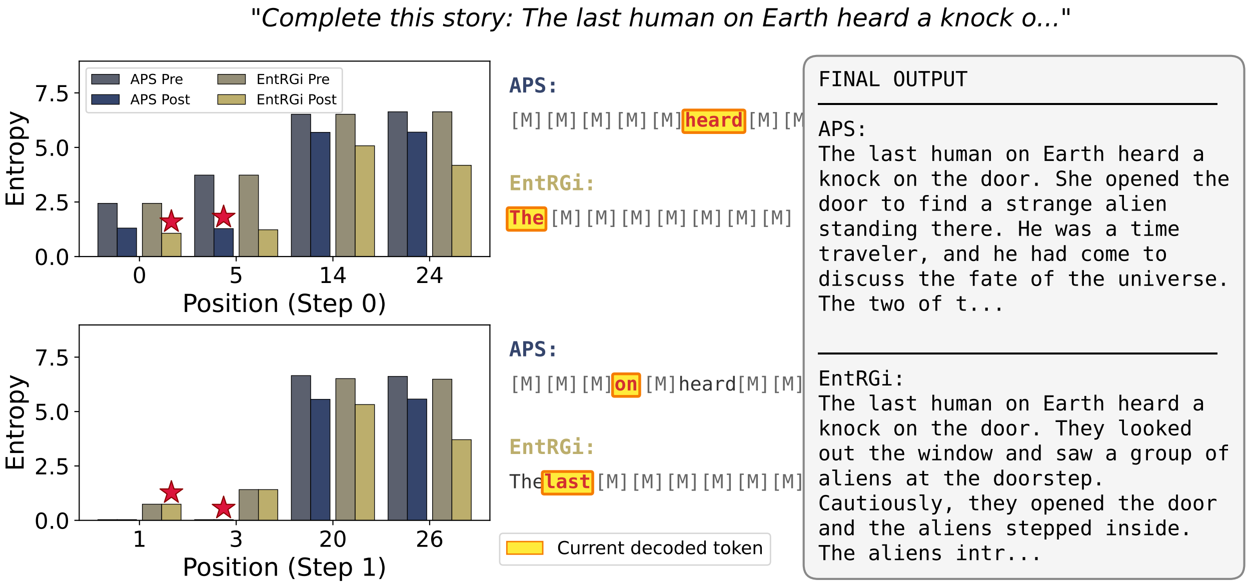


FINAL OUTPUT

APS:
Taylor expansion is a method of expressing a function as an infinite sum of terms, where each term is a derivative of the function evaluated at a particular point. It is used to approximate a function...

EntRGi:
Taylor expansion, also known as Taylor series expansion, is a method in calculus where a function is approximated by a polynomial of a certain degree. This is done by using the derivatives of the func...

Figure 11. Qualitative example of APS vs. EntRGi

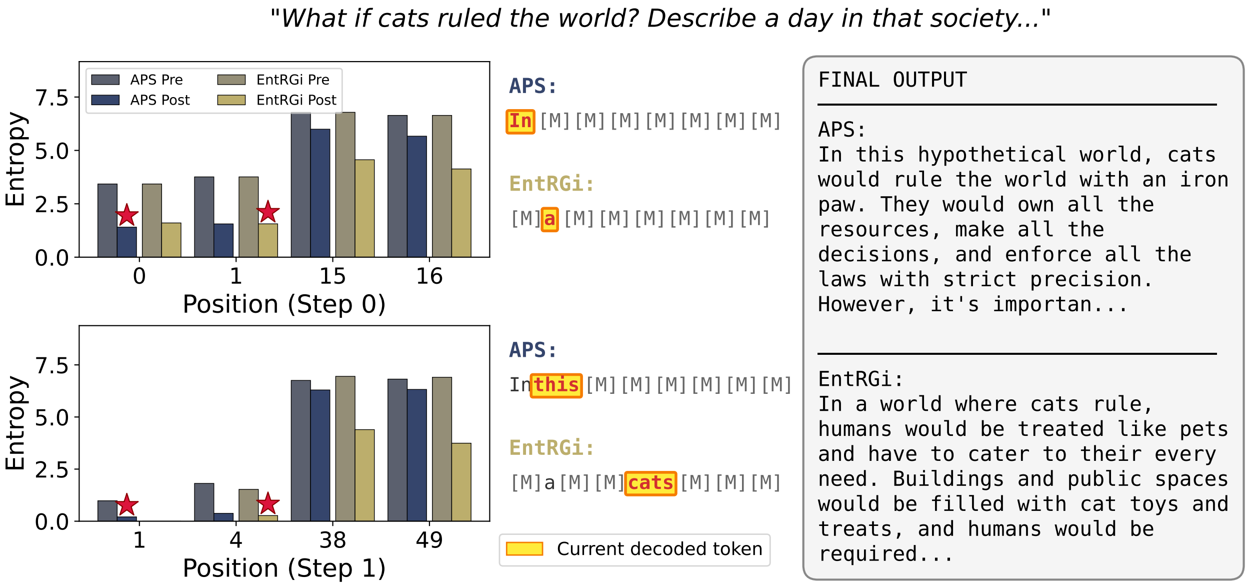


FINAL OUTPUT

APS:
The last human on Earth heard a knock on the door. She opened the door to find a strange alien standing there. He was a time traveler, and he had come to discuss the fate of the universe. The two of t...

EntRGi:
The last human on Earth heard a knock on the door. They looked out the window and saw a group of aliens at the doorstep. Cautiously, they opened the door and the aliens stepped inside. The aliens intr...

Figure 12. Qualitative example of APS vs. EntRGi



"What if cats ruled the world? Describe a day in that society..."

APS:
In [M] [M] [M] [M] [M] [M] [M]

EntRGi:
[M] a [M] [M] [M] [M] [M] [M]

APS:
In this [M] [M] [M] [M] [M] [M]

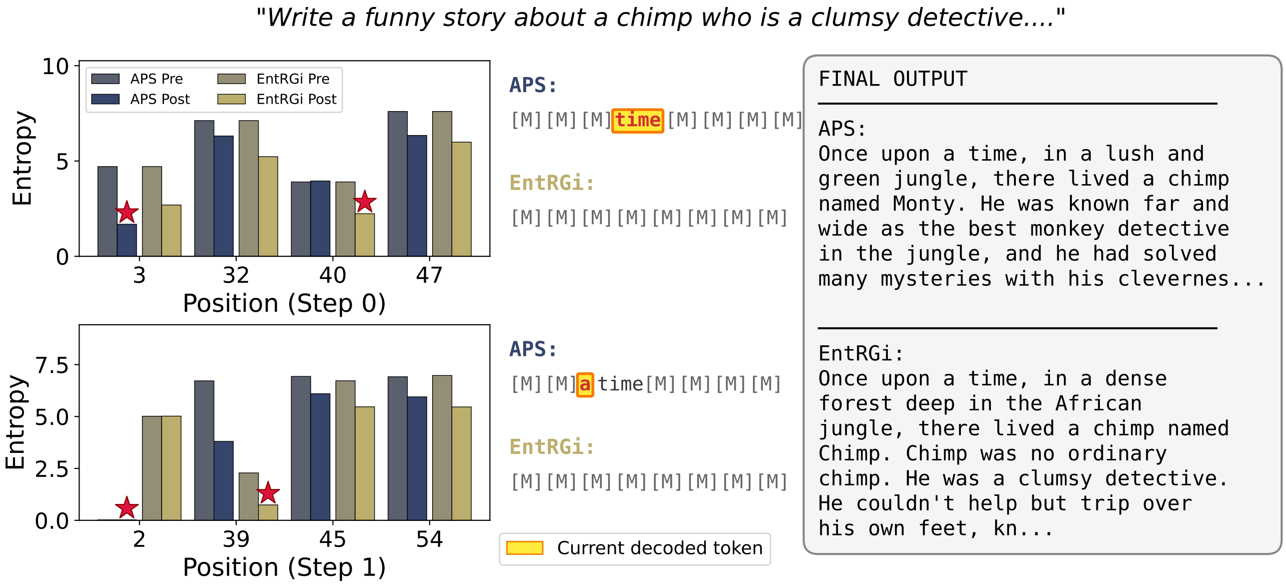
EntRGi:
[M] a [M] [M] cats [M] [M] [M]

```
FINAL OUTPUT

APS:
In this hypothetical world, cats
would rule the world with an iron
paw. They would own all the
resources, make all the
decisions, and enforce all the
laws with strict precision.
However, it's importan...

EntRGi:
In a world where cats rule,
humans would be treated like pets
and have to cater to their every
need. Buildings and public spaces
would be filled with cat toys and
treats, and humans would be
required...
```

Figure 13. Qualitative example of APS vs. EntRGi



"Write a funny story about a chimp who is a clumsy detective...."

APS:
[M] [M] [M] time [M] [M] [M] [M]

EntRGi:
[M] [M] [M] [M] [M] [M] [M] [M]

APS:
[M] [M] a time [M] [M] [M] [M]

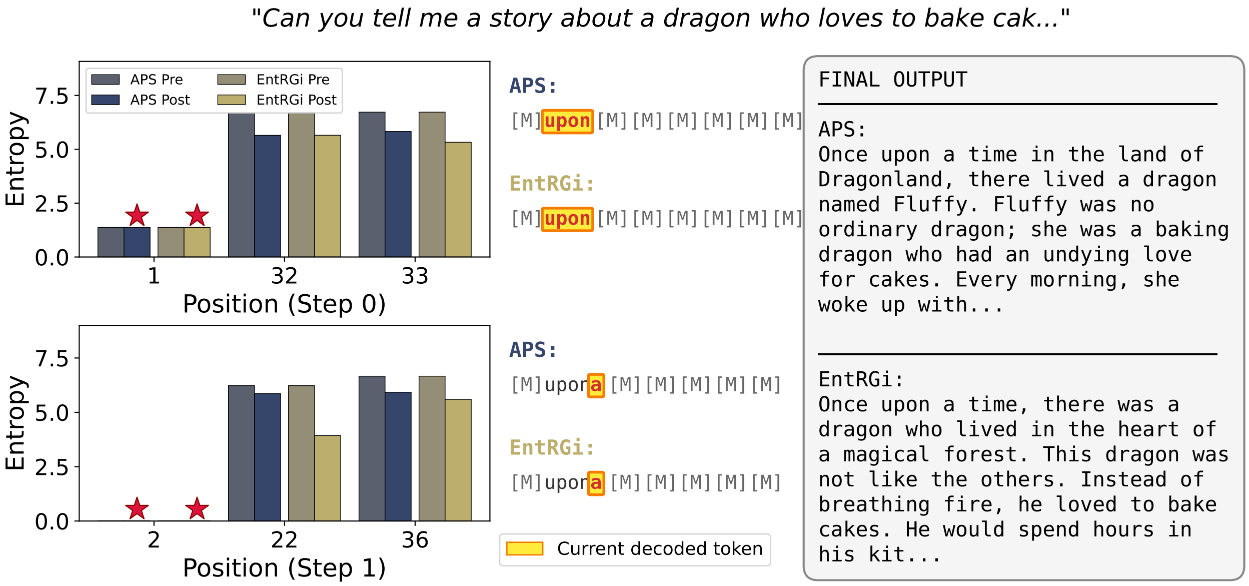
EntRGi:
[M] [M] [M] [M] [M] [M] [M] [M]

```
FINAL OUTPUT

APS:
Once upon a time, in a lush and
green jungle, there lived a chimp
named Monty. He was known far and
wide as the best monkey detective
in the jungle, and he had solved
many mysteries with his clevernes...

EntRGi:
Once upon a time, in a dense
forest deep in the African
jungle, there lived a chimp named
Chimp. Chimp was no ordinary
chimp. He was a clumsy detective.
He couldn't help but trip over
his own feet, kn...
```

Figure 14. Qualitative example of APS vs. EntRGi



FINAL OUTPUT

APS:
Once upon a time in the land of Dragonland, there lived a dragon named Fluffy. Fluffy was no ordinary dragon; she was a baking dragon who had an undying love for cakes. Every morning, she woke up with...

EntRGi:
Once upon a time, there was a dragon who lived in the heart of a magical forest. This dragon was not like the others. Instead of breathing fire, he loved to bake cakes. He would spend hours in his kit...

Figure 15. Qualitative example of APS vs. EntRGi