DACA-GRPO: Denoising-Aware Credit Assignment for Reinforcement Learning in Diffusion Language Models

Amin Karimi Monsefi Apple
Dominic Culver Apple
Nikhil Bhendawade Apple
Lokesh Boominathan Apple
Manuel R. Ciosici Apple
Yizhe Zhang* Meta
Irina Belousova Apple

Abstract

Diffusion large language models are a compelling alternative to autoregressive models, and recent work adapts GRPO-style reinforcement learning to train them. The resulting methods share two weaknesses: they lack *temporal credit assignment*, treating all denoising steps as equally important, and they optimize the policy with biased mean-field likelihood estimates. To address these, we propose Denoising-Aware Credit Assignment for GRPO (DACA-GRPO), a lightweight, plug-and-play enhancement for any GRPO-style trainer. DACA-GRPO introduces two complementary mechanisms: *Denoising Progress Scores*, which extract per-token importance weights from intermediate predictions at no additional forward cost, and *Stratified Masking Likelihood*, which partitions token positions into strata so that each token is predicted with most of the sequence as context, reducing the mean-field bias. Applied on top of three GRPO-style base methods, DACA-GRPO improves results on seven benchmarks in mathematical reasoning, code generation, constraint satisfaction (DPS only), and constrained generation. The largest improvements are 4.8 percentage points (pp) on mathematical reasoning, 7.4pp on code generation, 36.3pp on constraint satisfaction, and 14.1pp on JSON schema adherence. These results show that the intermediate predictions existing methods discard carry a useful training signal.

1 Introduction

Diffusion large language models (dLLMs) are an alternative to autoregressive models that support high-throughput, any-order parallel text generation: starting from a sequence of mask or uniform tokens, they iteratively denoise it into the final output [Sahoo et al., 2024, Nie et al., 2025, Monsefi et al., 2026, Inception Labs et al., 2025, Ye et al., 2025]. As with autoregressive LLMs, reinforcement learning (RL) can improve their reasoning, and Group Relative Policy Optimization (GRPO; Shao et al., 2024), widely adopted for autoregressive LLMs [Guo et al., 2025, Yu et al., 2025, Liu et al., 2025], has recently been adapted to dLLMs [Zhao et al., 2025, Tang et al., 2026, Rojas et al., 2026, Gong et al., 2026].

These adaptations have seen significant success, but they ignore the *denoising trajectory* itself and score completions with mean-field likelihood estimates. During each denoising step, dLLMs compute distributions over all masked positions but unmask only a few. The remaining predictions, which encode the model's evolving understanding, are discarded. Not all denoising steps contribute equally: some involve decisive structural commitments, while others are routine. Yet these methods assign

*Work done while at Apple.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

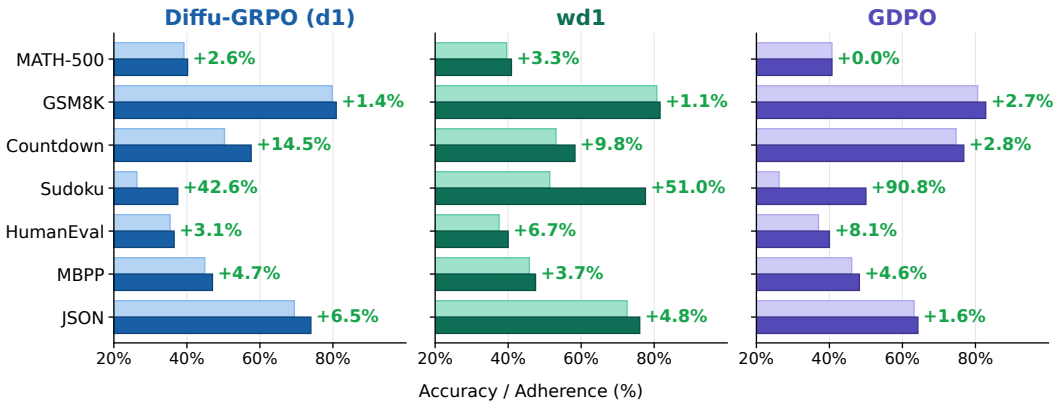

Figure 1: **DPS improves all three base methods on nearly every benchmark.** Best accuracy (%) across generation lengths {128, 256, 512}. Light bars: reproduced baselines. Dark bars: with DPS applied. Green percentages show relative improvement over the baseline. DPS delivers consistent gains across mathematical reasoning, code generation, constraint satisfaction, and constrained generation tasks, with the largest relative gains on Sudoku—up to $+51\%$ for wd1 and $+91\%$ for GDPO.

equal credit to every token regardless of when or how confidently it was generated. Moreover, unlike an autoregressive model, a dLLM cannot compute the completion log-likelihood $\log p_\theta(\mathbf{x} \mid q)$ that GRPO's objective requires. Existing methods approximate it, or replace it with an evidence lower bound (ELBO), by predicting the masked tokens independently of one another. The resulting mean-field bias corrupts the policy gradient. We make the following contributions:

1. We identify two fundamental weaknesses in existing dLLM RL methods: the **absence of temporal credit assignment** across denoising steps, and the **systematic mean-field bias** of the likelihood estimates used for policy optimization.

2. We introduce **Denoising Progress Scores (DPS)**, which track the model's belief evolution across denoising steps, converting these signals into per-token importance weights that modulate the RL loss. DPS requires no additional forward passes—it repurposes logits that the model would otherwise discard.

3. To mitigate the mean-field bias, we introduce **Stratified Masking Likelihood (SML)**, a likelihood estimator that splits the completion's token positions into a few groups and masks one group at a time, so each token is predicted with most of the completion visible. Unlike the standard estimate, which masks the whole completion, SML thus captures dependencies between tokens.

4. **DACA-GRPO** combines DPS and SML into a unified, plug-and-play framework that is agnostic to the underlying GRPO variant.

We evaluate DACA-GRPO on top of three popular RL methods—Diffu-GRPO (d1; Zhao et al., 2025)[2], wd1 [Tang et al., 2026], and GDPO [Rojas et al., 2026]—across seven benchmarks spanning mathematical reasoning (MATH-500, GSM8K), code generation (MBPP, HumanEval), constraint satisfaction (Countdown, Sudoku), and constrained generation (JSON). We apply the full method to math, code, and JSON generation, and DPS alone to constraint satisfaction. DACA-GRPO improves results across benchmarks and base methods, with gains of up to 4.8pp on math reasoning, 7.4pp on code generation, 36.3pp on constraint satisfaction, and 14.1pp on JSON schema adherence (Tables 1–3). The two components are complementary: DPS is most impactful on structured constraint tasks (Figure 1), while SML provides its largest gains on JSON and code generation.

[2]Diffu-GRPO is the RL stage of the two-stage d1 recipe [Zhao et al., 2025]; we apply it to LLaDA-8B-Instruct without d1's SFT stage and, following Tang et al. [2026], refer to it as d1.

2 Related work

Diffusion language models such as SEDD [Lou et al., 2024], MDLM [Sahoo et al., 2024], and LLaDA [Nie et al., 2025] established discrete diffusion as a viable paradigm for text generation, and Dream-Coder [Xie et al., 2025] extended it to code. Our work focuses on the RL training signal and does not change the diffusion architecture.

RL has been widely applied to autoregressive LLMs through methods like Reinforcement Learning from Human Feedback [Ouyang et al., 2022] and GRPO [Shao et al., 2024]. Recent work extends GRPO to diffusion models: Diffu-GRPO [Zhao et al., 2025] adapts PPO-clipped objectives, wd1 [Tang et al., 2026] proposes a ratio-free alternative, GDPO [Rojas et al., 2026] uses a sequence-level ELBO estimated by deterministic quadrature, and DiffuCoder [Gong et al., 2026] introduces coupled sampling for variance reduction. All four assign the same advantage to every token in a completion. TraceRL [Wang et al., 2026] assigns step-wise credit with a learned value model. To our knowledge, DPS is the first method to derive credit from the intermediate predictions that the model already computes during denoising, and it requires no value model or additional forward passes.

Temporal credit assignment is a classical challenge in RL [Sutton and Barto, 2018]. In the autoregressive setting, per-token reward models [Lightman et al., 2024, Zhu et al., 2025] and process reward models [Uesato et al., 2022] provide step-level supervision but require training additional reward models or collecting extra annotations. DPS obtains a step-level signal without either, because it reuses the model's own intermediate predictions. Concurrently, DyLLM [Lee et al., 2026] observes that not all token positions contribute equally across denoising steps and identifies salient positions via attention-context similarity. Both methods mine signals from the denoising trajectory, but to different ends: DyLLM uses attention-based saliency to accelerate inference, while DPS uses token-level prediction confidence to improve training via credit assignment.

SML differs from prior variance-reduction schemes for masked-token likelihood estimation: instead of pairing two complementary masks at random masking ratios (as in DiffuCoder's coupled sampling [Gong et al., 2026]) or using deterministic quadrature over time (as in GDPO's SDMC [Rojas et al., 2026]), it stratifies token *positions* into K strata and masks one stratum at a time. Each token then sees exactly $(K-1)/K$ of the sequence as context, which targets the mean-field bias rather than sampling variance.

3 Background

Masked diffusion language models [Sahoo et al., 2024, Nie et al., 2025] define a forward process that progressively masks tokens and a reverse process that denoises them. Given a clean sequence $\mathbf{x}_0 = (x_1, \ldots, x_L)$, the forward process produces a noisy sequence $\mathbf{x}_t$ at timestep $t \in [0, 1]$ by independently replacing each token with `[MASK]` with probability t. The reverse (generative) process starts with a fully masked sequence $\mathbf{x}_1$ and iteratively predicts and reveals tokens until reaching $\mathbf{x}_0$. At each denoising step t, the model p_θ predicts a distribution $p_\theta(x_i \mid \mathbf{x}_t)$ over all vocabulary tokens for every masked position $i \in \mathcal{M}_t$, where $\mathcal{M}_t = \{i : x_{t,i} = \texttt{[MASK]}\}$. The reverse process then unmasks only the few positions where the model is most confident, filling each with the predicted token. The predictions for all other masked positions are discarded. These "wasted" predictions form the basis of our DPS mechanism.

GRPO [Shao et al., 2024] generates a group of G candidate responses $\{\mathbf{x}^{(g)}\}_{g=1}^{G}$ for each prompt, computes rewards $\{r^{(g)}\}$, and defines group-relative advantages using the group mean μ_r and standard deviation σ_r:

$$A^{(g)} = \frac{r^{(g)} - \mu_r}{\sigma_r}. \tag{1}$$

We build on three adaptations of GRPO to dLLMs, which differ in their objectives and in how they estimate the likelihood of a completion.

Diffu-GRPO [Zhao et al., 2025] adapts GRPO to dLLMs using a per-token PPO-clipped surrogate objective with per-token likelihood ratio $\rho_i(\theta) = p_\theta(x_i \mid \mathbf{m})/p_{\theta_{\text{old}}}(x_i \mid \mathbf{m})$, computed with all completion tokens masked ($\mathbf{m}$):

$$\mathcal{L}_{\text{d1}} = -\mathbb{E}\left[\min\left(\rho_i(\theta)A,\ \text{clip}(\rho_i(\theta), 1-\epsilon, 1+\epsilon)A\right)\right]. \tag{2}$$

wd1 [Tang et al., 2026] proposes a ratio-free alternative with explicit positive and negative sample reinforcement, using normalized advantage weights $\hat{A}_+^{(g)} = \text{softmax}(A^{(g)})$ and $\hat{A}_-^{(g)} = \text{softmax}(-A^{(g)})$, computed over the group and applied to every sample:

$$\mathcal{L}_{\text{wd1}} = \underbrace{-\sum_{g=1}^{G} \hat{A}_+^{(g)} \log p_\theta(\mathbf{x}^{(g)})}_{\text{amplify correct}} \underbrace{+\sum_{g=1}^{G} \hat{A}_-^{(g)} \log p_\theta(\mathbf{x}^{(g)})}_{\text{suppress incorrect}}. \tag{3}$$

GDPO [Rojas et al., 2026] operates at the *sequence level*, using the evidence lower bound (ELBO) as a surrogate for $\log p_\theta(\mathbf{x})$:

$$\mathcal{L}_{\text{ELBO}}(\mathbf{x} \mid q) = \mathbb{E}_{t\sim\mathcal{U}[0,1]}\, \mathbb{E}_{\mathbf{x}_t\sim\pi_t(\cdot|\mathbf{x})} \left[\frac{1}{t} \sum_{i=1}^{L} \mathbf{1}[x_{t,i} = \texttt{[MASK]}] \log \pi_\theta(x_i \mid y_t, q) \right]. \tag{4}$$

To reduce variance, GDPO introduces Semi-deterministic Monte Carlo (SDMC): a small number of fixed quadrature points over t with random mask sampling. The loss uses sequence-level importance ratios $\rho_g = \exp\big((\mathcal{L}_{\text{ELBO}}^{\theta}(\mathbf{x}^{(g)} \mid q) - \mathcal{L}_{\text{ELBO}}^{\theta_{\text{old}}}(\mathbf{x}^{(g)} \mid q))/|\mathbf{x}^{(g)}|\big)$ and unnormalized advantages $A_g = r^{(g)} - \text{mean}(r^{(1)}, \ldots, r^{(G)})$:

$$\mathcal{L}_{\text{GDPO}}(\theta) = -\frac{1}{G} \sum_{g=1}^{G} \min(\rho_g \cdot A_g,\ \text{clip}(\rho_g, 1-\epsilon, 1+\epsilon) \cdot A_g) + \beta\, \text{KL}(\pi_\theta \| \pi_{\text{ref}}). \tag{5}$$

Masked-token log-likelihood. All three methods require estimating $\log p_\theta(\mathbf{x})$ for each completion. In dLLMs, this is typically approximated by sampling a random masking ratio $t \sim \mathcal{U}(0, 1)$, constructing a masked version $\mathbf{x}_t$, and computing:

$$\log p_\theta(\mathbf{x}) \approx \frac{1}{|\mathcal{M}_t|} \sum_{i \in \mathcal{M}_t} \log p_\theta(x_i \mid \mathbf{x}_t). \tag{6}$$

Here $p_\theta(x_i \mid \mathbf{x}_t)$ is obtained from the model's output logits at position i via softmax—each per-token log-probability is exact; the approximation arises from the mean-field factorization into independent per-token terms. d1 and wd1 use a single sample of this estimate with the completion fully masked (t=1), so every token is predicted with zero inter-token context (mean-field bias), and the only randomness comes from masking the prompt. GDPO averages over several masking ratios via deterministic quadrature over t, which reduces variance, but at each quadrature point the masked tokens are still scored independently under a random partial mask.

4 Weaknesses of current approaches

Weakness 1: No Temporal Credit Assignment. Diffusion models generate a completion through a trajectory of denoising steps, predicting all masked tokens in parallel at each step and committing only a few. Yet the RL losses of all three methods assign a single scalar advantage to the *entire* completion (Figure 3), ignoring when and how confidently each token was committed along this trajectory. We hypothesize that this uniform weighting under-credits decisive steps, where the model commits to key tokens such as operators in math or control flow in code, and over-credits routine fill-in steps with little change in belief. The result is a training signal that does not concentrate learning on the steps that matter most.

Weakness 2: Biased and High-Variance Likelihood Estimates. All three base methods require estimating $\log p_\theta(\mathbf{x})$ for each completion. The estimate used by d1 and wd1 (Equation (6) with t=1) masks *all* completion tokens and predicts them from zero context, treating tokens as independent; GDPO's ELBO uses partial masks but still scores masked tokens independently. This introduces *systematic bias*: predicting without any context is fundamentally harder, adding errors to the log-likelihood estimates. It also introduces *high variance*: a single random masking pattern (of the prompt in d1 and wd1, or of the completion in ELBO-based estimates) may over- or under-represent the typical difficulty of predicting $\mathbf{x}$, producing noisy gradients.

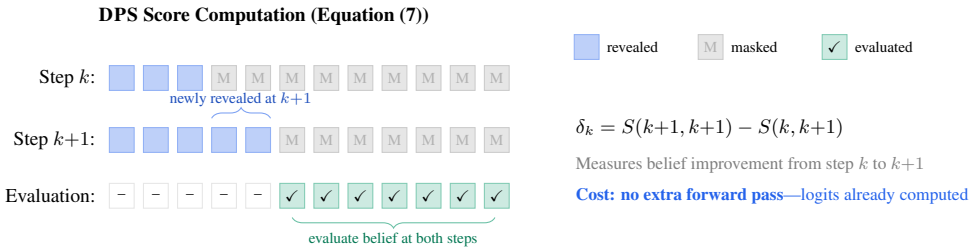

Figure 2: **DPS score computation.** We quantify the change in model beliefs by computing the difference δ_k in model predictions on positions masked in the current and the following step. DPS uses the logits computed naturally during generation and requires no extra forward passes.

5 Method

DACA-GRPO addresses the two weaknesses of Section 4 with two mechanisms that change different inputs to the loss. To address Weakness 1, DPS (Section 5.1) weights each token's advantage by the progress made at the denoising step that committed it. To address Weakness 2, SML (Section 5.2) estimates the completion likelihood with less mean-field bias. Section 5.3 shows how SML enters the loss of each base method, and Section 5.4 combines the two.

5.1 Denoising Progress Scores (DPS)

At each denoising step, the model predicts token distributions for all masked positions, but only unmasks a few. The predictions at still-masked positions reveal how well the model "understands" the emerging sequence, and the amount by which that understanding changes between steps provides a natural measure of step importance.

Formally, let $\mathbf{x}_{t_0}, \mathbf{x}_{t_1}, \ldots, \mathbf{x}_{t_T}$ denote the recorded snapshots of the denoising trajectory at noise levels $1 = t_0 > t_1 > \cdots > t_T$, so the snapshot index k counts up as the noise level t_k decreases, and let $\mathcal{M}_T \subset \mathcal{M}_{T-1} \subset \cdots \subset \mathcal{M}_0$ be the corresponding sets of masked positions ($\mathbf{x}_{t_0}$ is fully masked and $\mathbf{x}_{t_T}$ is the last recorded snapshot). We define $S(k, j)$ as the average log-probability under the model at step k over the positions in $\mathcal{M}_j$:

$$S(k, j) = \frac{1}{|\mathcal{M}_j|} \sum_{i \in \mathcal{M}_j} \log p_\theta(x_i \mid \mathbf{x}_{t_k}), \tag{7}$$

where x_i is the token at position i of the sampled completion, that is, the token the model itself selected during the rollout rather than a reference answer. The *denoising progress delta* at step k measures the belief improvement when transitioning from step k to $k+1$:

$$\delta_k := S(k{+}1,\ k{+}1) - S(k,\ k{+}1). \tag{8}$$

Both scores are evaluated on the same set of positions $\mathcal{M}_{k+1}$ (i.e., those still masked at the next step), so the difference isolates the effect of the newly revealed tokens (Figure 2). A large positive δ indicates that revealing tokens at step $k{+}1$ substantially improved the model's understanding (i.e., an important denoising step). A small or negative δ indicates minimal or confused progress (Figure 3).

DPS requires no additional forward passes. DPS reuses the log-probabilities that the model already computes at still-masked positions and would otherwise discard. The overhead consists of caching the logits for each recorded snapshot and gathering the log-probabilities of the final tokens once generation ends: under 1% wall-clock overhead for strides $s \geq 4$ and 3.2% at $s{=}1$ (Appendix C.5).

Raw deltas vary in scale across denoising steps due to the non-stationary nature of the denoising process: the amount of visible context and the number of still-masked positions change systematically along the trajectory. Therefore, we apply *per-step* z-score normalization within the group of completions for each prompt:

$$\bar{\delta}_k = \frac{\delta_k - \mu_k}{\sigma_k + \epsilon}, \tag{9}$$

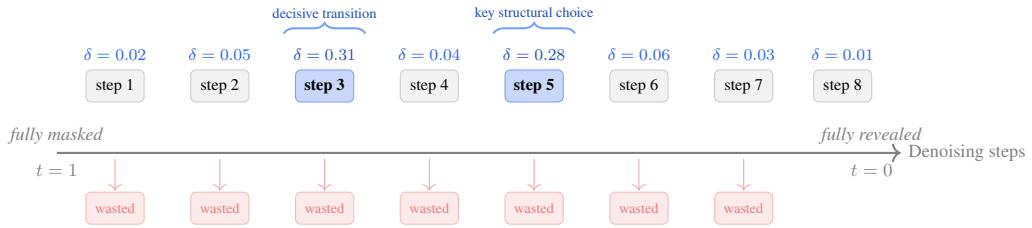

Figure 3: **Illustration of Weakness 1.** dLLMs perform many denoising steps of varying importance. Steps 3 and 5 involve decisive transitions (large δ), while others are routine. Standard methods assign equal credit to all steps and discard intermediate predictions. The δ values are illustrative.

where μ_k and σ_k are the mean and standard deviation of δ_k over the G completions of a prompt. This ensures fair credit assignment across the entire denoising timeline: an "impressive" early-step delta and an "impressive" late-step delta receive comparable normalized values, despite different magnitudes. Among four normalization modes, per-step is the only one that matches or exceeds the baseline in every stride configuration (Appendix C.2).

Per-token weight modulation. Each token in the final sequence is "born" at the denoising step where it was unmasked. We track each token's birth step, mapped to the index $\text{birth}(i)$ of the recorded snapshot interval in which it was unmasked, and define the DPS-*modulated per-token advantage*:

$$\tilde{A}_i^{(g)} = A^{(g)} \cdot \left(1 + \lambda \cdot \bar{\delta}_{\text{birth}(i)}\right), \tag{10}$$

where $\lambda > 0$ controls modulation strength. The factor $\omega_i = 1 + \lambda \cdot \bar{\delta}_{\text{birth}(i)}$ amplifies or attenuates the learning signal based on step importance. Since a z-score over G=6 samples satisfies $|\bar{\delta}_k| \leq (G-1)/\sqrt{G} \approx 2.04$, the default λ=0.1 keeps $\omega_i \in [0.79, 1.21]$, so the weights are always positive. For negative-advantage samples, the sign of $\bar{\delta}$ provides directional credit: positive values increase the penalty (the step made the model more confident in the wrong completion it produced); negative values reduce it (the step made it less confident).

The DPS implementation requires two design choices: (1) tokens assigned to the last recorded snapshot T have no following snapshot to compute a delta against, so we extrapolate $\delta_T = \delta_{T-1}$; (2) we record snapshots with a stride s, trading credit granularity for signal-to-noise ratio. Both choices are robust: extrapolation achieves the highest average accuracy among five last-step strategies, and every stride $s \in \{1, 2, 4, 8, 16, 32\}$ matches or exceeds the baseline under our default settings. We describe and ablate them in Appendix C.1.

5.2 Stratified Masking Likelihood (SML)

As Weakness 2 explains, existing estimates score masked tokens independently of one another, which biases the likelihood (the mean-field bias). To reduce this bias, we propose a stratified estimator that uses K forward passes. We partition the L output token positions $x_1, \ldots, x_L$ of completion $\mathbf{x}$ into K disjoint strata $\mathcal{S}_1, \ldots, \mathcal{S}_K$ of roughly equal size. For each stratum $\mathcal{S}_k$, we mask *only* the positions in $\mathcal{S}_k$ while keeping all other output tokens visible, and compute the per-token log-probability for each masked position $n \in \mathcal{S}_k$. Summing the token-level log-probabilities across all strata gives the sequence-level SML estimate:

$$\log p_\theta^{\text{SML}}(\mathbf{x}|q) = \sum_{k=1}^{K} \sum_{n \in \mathcal{S}_k} \log p_\theta(x_n \mid \mathbf{x}_{\setminus \mathcal{S}_k}, q), \tag{11}$$

where $\mathbf{x}_{\setminus \mathcal{S}_k}$ denotes the output sequence with only stratum $\mathcal{S}_k$ replaced by mask tokens. Each token is predicted with $(K-1)/K$ of the sequence as visible context, forming a tunable spectrum: K=1 places all tokens in a single stratum, recovering the fully masked estimate of Equation (6) (with t=1, up to normalization); K=L gives each token its own stratum, recovering the pseudo-log-likelihood where every token sees all others. Figure 4 illustrates SML. We hypothesize that this approximation of the log-probabilities reduces the estimation error of mean-field-based methods such as d1, and we evaluate this hypothesis through downstream accuracy in Section 6. SML targets the mean-field bias—complementary to variance-reduction methods like GDPO's SDMC.

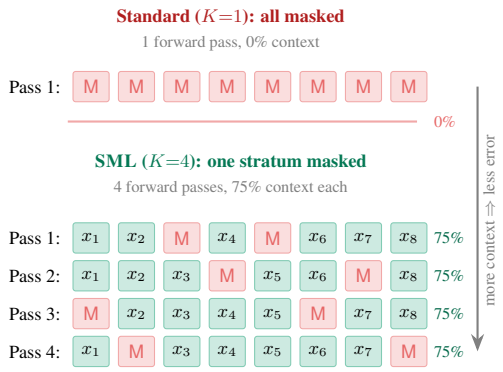

Figure 4: **Stratified Masking Likelihood.** Standard estimation (K=1) masks all tokens with zero context. SML (K=4) masks one stratum per pass, giving each token 75% context.

5.3 SML integration

All three integrations use the same *enriched* per-token log-probability. For each output position i, let $k(i)$ denote the stratum containing position i. We average two views—the fully masked prediction and the stratum prediction:

$$\hat{\ell}_{\theta,i} = \frac{1}{2}\Big[\log p_\theta(x_i \mid \mathbf{m}, q) + \log p_\theta(x_i \mid \mathbf{x}_{\setminus \mathcal{S}_{k(i)}}, q)\Big], \tag{12}$$

where $\mathbf{m}$ denotes the fully masked completion (the standard estimate from Equation (6)) and $\mathbf{x}_{\setminus \mathcal{S}_{k(i)}}$ denotes the completion with only stratum $\mathcal{S}_{k(i)}$ masked. The fully masked view keeps the estimate aligned with the generation regime (Appendix B.1), while the stratified view is intended to reduce the mean-field bias. How $\hat{\ell}_{\theta,i}$ enters the loss depends on whether the base method uses a clipped importance ratio.

Ratio-free methods. Methods like wd1 optimize log-probabilities directly without importance ratios or trust regions, so we simply replace each per-token log-probability in Equation (3) with its enriched counterpart:

$$\mathcal{L}_{\text{wd1+SML}} = \underbrace{-\sum_{g=1}^{G} \hat{A}_+^{(g)} \sum_i \hat{\ell}_{\theta,i}^{(g)}}_{\text{amplify correct}} \underbrace{+\sum_{g=1}^{G} \hat{A}_-^{(g)} \sum_i \hat{\ell}_{\theta,i}^{(g)}}_{\text{suppress incorrect}}. \tag{13}$$

The loss mirrors the original wd1 loss and introduces no additional hyperparameters.

Ratio-based methods. d1 and GDPO (Equations (2) and (5)) cannot accept an additive SML loss, as it destabilizes training: its gradient bypasses the PPO clip, resulting in training collapse. Instead, we incorporate SML *inside* the importance ratio at the token level:

$$\rho_i^{\text{SML}} := \exp\Big(\hat{\ell}_{\theta,i} - \hat{\ell}_{\theta_{\text{old}},i}\Big), \tag{14}$$

which replaces the per-token ratio of d1 and, for GDPO, the sequence-level ELBO ratio, in the clipped surrogate. At $\theta = \theta_{\text{old}}$ the ratio equals 1, provided the same strata and masks are used for both terms; $\hat{\ell}_{\theta_{\text{old}},i}$ is computed once per rollout and cached. No hyperparameters beyond K are introduced. SML adds K forward passes during loss computation, a moderate cost (+11.7% wall-clock at K=4; Appendix C.5) compared to the generation phase, which dominates each training step.

5.4 DACA-GRPO: combining DPS and SML

DPS and SML can be applied separately or jointly, as they modify two independent inputs to the loss: DPS modulates the per-token advantage while SML improves the likelihood estimate. For ratio-free methods, the softmax weights are computed from the sequence-level advantages $A^{(g)}$, so DPS enters through the per-token factor ω_i of Equation (10), and the combined loss becomes:

$$\mathcal{L}_{\text{DACA}} = -\sum_{g=1}^{G} \hat{A}_+^{(g)} \sum_i \omega_i^{(g)}\, \hat{\ell}_{\theta,i}^{(g)} + \sum_{g=1}^{G} \hat{A}_-^{(g)} \sum_i \omega_i^{(g)}\, \hat{\ell}_{\theta,i}^{(g)}. \tag{15}$$

Algorithm 1 DACA-GRPO

Require: Policy π_θ, reward R, stride s, modulation strength λ, number of strata K

for each training step **do**
  Generate G completions; cache logits at still-masked positions every s steps
  Compute rewards and group-relative advantages $A^{(g)}$
  DPS: compute progress deltas δ_k from cached logits — Equations (7) and (8)
  DPS: per-token advantages $\tilde{A}_i^{(g)} \leftarrow A^{(g)} \cdot \left(1 + \lambda \cdot \bar{\delta}_{\text{birth}(i)}\right)$ — Equations (9) and (10)
  for each inner iteration **do**
    SML: compute enriched per-token log-probabilities $\hat{\ell}_{\theta,i}$ with K strata — Equations (11) and (12)
    Compute $\mathcal{L}_{\text{DACA}}$ using $\tilde{A}_i^{(g)}$ and SML likelihood — Equation (15) or Equation (16)
    Update θ via gradient descent
  end for
end for

With DPS alone, $\hat{\ell}_{\theta,i}$ is replaced by the standard per-token log-probability. For ratio-based methods, SML enters inside the importance ratio and DPS modulates the per-token advantages:

$$\mathcal{L}_{\text{DACA}} = -\mathbb{E}\left[\min\left(\rho_i^{\text{SML}} \cdot \tilde{A}_i^{(g)},\ \text{clip}(\rho_i^{\text{SML}}, 1-\epsilon, 1+\epsilon) \cdot \tilde{A}_i^{(g)}\right)\right]. \tag{16}$$

For GDPO, whose ratio is sequence-level, DPS alone scales the per-sample loss by the average weight $\frac{1}{|\mathbf{x}^{(g)}|}\sum_i \omega_i^{(g)}$. Algorithm 1 gives the complete procedure. The key structural property is that DPS weights are computed once per training step during generation and reused across all inner iterations, while SML forward passes occur within each inner iteration during loss computation.

6 Experiments

6.1 Setup

We use LLaDA-8B-Instruct [Nie et al., 2025] as the base dLLM, consistent with prior work [Zhao et al., 2025, Tang et al., 2026], and the three GRPO variants described in Section 3 as base RL methods: Diffu-GRPO (PPO-clipped surrogate with importance ratios), wd1 (ratio-free weighted log-likelihood), and GDPO (sequence-level, ELBO-based policy optimization).

We evaluate on seven benchmarks: MATH-500 [Hendrycks et al., 2021] and GSM8K [Cobbe et al., 2021] for mathematical reasoning; MBPP [Austin et al., 2021] and HumanEval [Chen et al., 2021] for code generation; Countdown and Sudoku for constraint satisfaction; and JSON schema adherence for constrained generation. We apply the full DACA-GRPO framework (DPS + SML) to math, code, and JSON generation (Tables 1 and 3) and only DPS to constraint satisfaction (Table 2). We evaluate at generation lengths 128, 256, and 512 with proportionally scaled diffusion steps to maintain consistency with Zhao et al. [2025], Tang et al. [2026]. For Sudoku, we report per-empty-cell accuracy, following Tang et al. [2026]. Appendices A and C contain more details and analyses.

6.2 Math and code

Table 1 presents mathematical reasoning and code generation results. Two patterns emerge.

DPS and SML are complementary. DACA-GRPO achieves the best or tied-best average improvement in 10 of 12 method–benchmark combinations. Both DPS and SML lead to increases, and combining them (DACA-GRPO) usually improves over either alone, in some cases nearly additively. For example, on wd1 MBPP, DPS alone adds +1.4pp and SML alone adds +2.4pp, but DACA-GRPO adds +3.7pp—more than either component and close to their sum. This confirms that DACA-GRPO is usually the best choice. Each component on its own improves 33 (DPS) and 32 (SML) of the 36 method–benchmark–length cells of Table 1, for both ratio-based (d1, GDPO) and ratio-free (wd1) base methods.

Effect of generation length. On math, averaged over the three base methods and both benchmarks, DPS improves accuracy by 1.6, 1.4, and 1.2pp and SML by 2.1, 1.6, and 1.0pp at L=128, 256, and 512, so both improvements shrink as outputs get longer. On code, DPS improves accuracy by 2.9,

Table 1: Mathematical reasoning accuracy and code generation pass@1 (%) by generation length.

	MATH-500			GSM8K			MBPP			HumanEval		
Method	**128**	**256**	**512**	**128**	**256**	**512**	**128**	**256**	**512**	**128**	**256**	**512**
LLaDA-8B-Inst.	26.0	32.4	36.2	68.7	76.7	78.2	35.8	42.6	45.2	17.1	26.2	34.2
Diffu-GRPO	31.8	37.1	39.2	72.1	79.2	79.8	41.7	44.9	44.2	19.8	32.3	35.4
+ DPS	33.5	37.5	40.2	74.0	80.8	80.9	**47.0**	46.5	45.6	22.9	32.8	36.5
+ SML	34.3	38.3	**40.5**	**74.6**	**81.6**	**81.5**	44.4	47.5	48.6	**26.6**	**36.5**	**40.6**
+ DACA-GRPO	**35.4**	**38.8**	**40.5**	**74.6**	**81.6**	**81.5**	45.4	**48.1**	**50.5**	25.5	35.4	40.1
wd1	32.6	36.7	39.6	70.5	80.6	80.7	45.4	45.8	45.8	21.9	34.9	37.5
+ DPS	33.5	39.8	40.9	72.8	81.6	81.6	**46.1**	47.5	47.5	25.0	36.5	40.0
+ SML	**35.6**	39.4	**41.7**	74.3	81.8	81.5	45.1	49.3	49.8	24.0	36.5	40.6
+ DACA-GRPO	**35.6**	**40.2**	41.3	**75.3**	**81.9**	**82.3**	**46.1**	**50.0**	**51.9**	**27.1**	**38.0**	**41.2**
GDPO	33.9	36.9	40.7	73.6	80.6	80.0	38.9	45.1	46.1	18.8	37.0	37.0
+ DPS	34.9	**39.0**	40.7	**75.1**	80.6	**82.8**	40.7	47.7	48.2	21.9	35.9	40.0
+ SML	34.5	**39.0**	40.2	74.0	80.6	80.6	45.8	47.5	**50.0**	**25.5**	36.5	40.6
+ DACA-GRPO	**35.2**	38.6	**41.7**	74.3	**81.6**	82.0	**46.3**	**48.1**	49.5	25.0	**37.0**	**41.1**

Table 3: **JSON generation.** Schema adherence rate (%) on GITHUB-MEDIUM; Diffu-GRPO and wd1 at L=512, GDPO at L=256 (LLaDA-8B-Inst. at the same L). Best over record strides (DPS) or strata counts and stratification strategies (SML). Green: absolute change over the RL baseline.

Method (L)	**LLaDA-8B-Inst.**	**RL baseline**	**+ DPS**	**+ SML**	**+ DACA-GRPO**
Diffu-GRPO (512)	60.55	69.44	73.96 +4.52	79.20 +9.76	**81.10 +11.66**
wd1 (512)	60.55	72.57	76.04 +3.47	**85.71 +13.14**	85.18 +12.61
GDPO (256)	48.83	63.19	64.23 +1.04	76.25 +13.06	**77.29 +14.10**

1.2, and 2.0pp and SML by 4.2, 2.3, and 4.0pp, with the largest improvements at the shortest and longest lengths. DACA-GRPO achieves the highest MBPP accuracy at L=512 on Diffu-GRPO and wd1 (50.5 and 51.9); on GDPO, SML alone is higher (50.0 vs. 49.5).

6.3 Constraint satisfaction

SML Applicability: Task Structure and Output Length. We apply SML only to long-form tasks (MATH-500, GSM8K, MBPP, HumanEval, JSON) and exclude it from constraint satisfaction (Countdown, Sudoku) due to a distributional mismatch. SML evaluates each token with $(K-1)/K$ visible context (75% for K=4), whereas generation starts fully masked. For short, tightly constrained outputs ($\sim$7 empty Sudoku cells, 10–15 Countdown tokens), 75% context nearly determines the answer via constraint propagation, driving the model toward trivial gap-filling that fails at generation time.

Results. DPS delivers the most dramatic improvements on Sudoku, where wd1 + DPS improves per-cell accuracy by 26–36pp (Table 2)—consistent with the task's tight inter-cell dependencies, which make decisive denoising steps especially impactful. The jump reflects a qualitative shift: wd1's per-cell accuracy at L=512 rises from 34.6% to 70.9%, crossing from unreliable to reliable. On Countdown, DPS improves d1 and wd1 consistently (+1.2–10.4pp) but shows a small degradation on GDPO at L=128 ($-$2.4pp), likely because GDPO's stronger baseline leaves less headroom.

Table 2: **Constraint satisfaction (DPS only).** Countdown accuracy and Sudoku per-empty-cell accuracy (%).

	Countdown			Sudoku		
Method	**128**	**256**	**512**	**128**	**256**	**512**
LLaDA-8B-Inst.	20.7	19.5	16.0	11.7	6.7	5.5
Diffu-GRPO	50.0	50.3	47.9	25.9	26.2	26.3
+ DPS	**57.6**	**55.2**	**53.8**	**26.8**	**29.2**	**37.5**
wd1	53.1	53.1	47.9	51.4	40.5	34.6
+ DPS	**54.3**	**56.9**	**58.3**	**77.6**	**76.1**	**70.9**
GDPO	**62.5**	69.4	74.7	26.1	26.2	25.8
+ DPS	60.1	**70.2**	**76.8**	**38.9**	**50.0**	**37.5**

6.4 JSON generation

Existing JSON benchmarks such as json-mode-eval are saturated: LLaDA-8B-Instruct already reaches 88–96% schema adherence, and nearly 100% after RL, because their schemas are shallow [Mündler et al., 2026]. We therefore build a synthetic benchmark from JSONSchemaBench [Geng et al., 2025]: for each schema, GPT-5 writes three scenarios and the matching JSON, and we keep only outputs that parse and conform to the schema. We report schema adherence, the fraction of outputs that conform to the target schema, on the GITHUB-MEDIUM test split; GDPO runs at $L{=}256$ because its implementation runs out of memory at $L{=}512$.

The task is non-trivial even after RL post-training: the strongest base method (wd1 at $L{=}512$) reaches only 72.57%, reflecting the difficulty of generating nested structures that fully conform to a target schema. DPS improves schema adherence for all three base methods (Table 3), and combining both components amplifies the effect: DACA-GRPO achieves the highest schema adherence on Diffu-GRPO and GDPO (up to +14.10pp). wd1 is the exception: SML alone contributes most of the gain and narrowly beats DACA-GRPO, suggesting mild interference between the two components. Despite the rigid schema structure, SML is effective here—unlike on Sudoku and Countdown—because JSON answers contain many more content tokens than Sudoku or Countdown answers, so partial-mask context does not trivially determine the remaining tokens. This indicates that SML's applicability is limited by output length rather than by structure (reward and training details in Appendix A.2).

7 Discussion and conclusion

We presented DACA-GRPO, a plug-and-play framework that brings temporal credit assignment and improved likelihood estimation to any GRPO-style dLLM trainer, with improvements across seven benchmarks and three base methods.

The central finding is that intermediate predictions, which existing methods compute and discard at every denoising step, contain enough signal to improve training: DPS uses this signal at near-zero cost, and its largest improvements, on Sudoku, suggest that tasks with tight inter-token dependencies are especially under-served by uniform credit assignment. The same per-step measure improves ratio-based and ratio-free losses across math, code, and constraint satisfaction and across generation lengths, which indicates that the missing credit-assignment structure is a shared bottleneck in current dLLM RL pipelines—not an artifact of any single loss formulation.

SML complements DPS on math, code, and JSON generation, with the largest improvements on JSON and code. This holds even for GDPO, whose SDMC already diversifies likelihood views along the time axis, although its math improvements on GDPO are small. This points to a practical guideline: DPS is a broadly applicable add-on, while SML should be prioritized for methods and tasks where likelihood quality is the constraint.

Limitations. SML is not applicable to short, tightly constrained outputs, where the distributional mismatch between partial-mask training and fully masked generation causes mode collapse; developing a likelihood estimator effective across all output lengths remains an open problem. Our experiments also use a single base model (LLaDA-8B-Instruct); validation on other dLLM architectures and scales would strengthen the generality claims.

Future directions. Adaptive stride selection—adjusting recording frequency based on observed delta magnitudes—could improve the signal-to-noise trade-off without manual tuning. In addition, SML's stratified estimation principle may extend beyond RL to the pre-training objective itself, where the same mean-field bias affects the masked diffusion loss.

Acknowledgments and Disclosure of Funding

We thank Barry Theobald and Mehrdad Farajtabar for feedback on early versions of this paper.

References

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models, 2021. URL https://arxiv.org/abs/2108.07732.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021. URL https://arxiv.org/abs/2107.03374.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

Di Feng, Kaixin Ma, Feng Nan, Haofeng Chen, Bohan Zhai, David Griffiths, Mingfei Gao, Zhe Gan, Eshan Verma, Yinfei Yang, Zhifeng Chen, and Afshin Dehghan. SO-Bench: A structural output evaluation of multimodal LLM. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 37694–37704, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Feng_SO-Bench_A_Structural_Output_Evaluation_of_Multimodal_LLM_CVPR_2026_paper.html.

Saibo Geng, Hudson Cooper, Michał Moskal, Samuel Jenkins, Julian Berman, Nathan Ranchin, Robert West, Eric Horvitz, and Harsha Nori. JSONSchemaBench: A rigorous benchmark of structured outputs for language models, 2025. URL https://arxiv.org/abs/2501.10868.

Shansan Gong, Ruixiang Zhang, Huangjie Zheng, Jiatao Gu, Navdeep Jaitly, Lingpeng Kong, and Yizhe Zhang. DiffuCoder: Understanding and improving masked diffusion models for code generation. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/1a5b00e59818e4be1c89855a8b921903-Abstract-Conference.html.

Daya Guo, Dejian Yang, Haowei Zhang, et al. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081):633–638, September 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-09422-z. URL https://doi.org/10.1038/s41586-025-09422-z.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *The Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. URL https://openreview.net/forum?id=7Bywt2mQsCe.

Inception Labs, Samar Khanna, Siddhant Kharbanda, Shufan Li, Harshit Varma, Eric Wang, Sawyer Birnbaum, Ziyang Luo, Yanis Miraoui, Akash Palrecha, Stefano Ermon, Aditya Grover, and Volodymyr Kuleshov. Mercury: Ultra-fast language models based on diffusion, 2025. URL https://arxiv.org/abs/2506.17298.

Younjoo Lee, Seungkyun Dan, Junghoo Lee, Jaiyoung Park, and Jung Ho Ahn. DyLLM: Efficient diffusion LLM inference via saliency-based token selection and partial attention. In *The Forty-third International Conference on Machine Learning*, pages 63635–63655, 2026. URL https://proceedings.mlr.press/v306/lee26i.html.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding R1-Zero-like training: A critical perspective. In *The Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=5PAF7PAY2Y.

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. In *The Forty-first International Conference on Machine Learning*, 2024.

Amin Karimi Monsefi, Nikhil Bhendawade, Manuel Rafael Ciosici, Dominic Culver, Yizhe Zhang, and Irina Belousova. FS-DFM: Fast and accurate long text generation with few-step diffusion language models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=ue1zFeD275.

Niels Mündler, Jasper Dekoninck, and Martin Vechev. Constrained decoding of diffusion LLMs with context-free grammars. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=7Sph4KyeYO.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=KnqiC0znVF.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *The Thirty-sixth Annual Conference on Neural Information Processing Systems*, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/b1efde53be364a73914f58805a001731-Paper-Conference.pdf.

Kevin Rojas, Jiahe Lin, Kashif Rasul, Anderson Schneider, Yuriy Nevmyvaka, Molei Tao, and Wei Deng. Improving reasoning for diffusion language models via group diffusion policy optimization. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/0428b72cab14bc38db5b2bad183e1987-Abstract-Conference.html.

Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. doi: 10.52202/079017-4135. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/eb0b13cc515724ab8015bc978fdde0ad-Paper-Conference.pdf.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2018.

Xiaohang Tang, Rares Dolga, Sangwoong Yoon, and Ilija Bogunovic. wd1: Weighted policy optimization for reasoning in diffusion language models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=L2rfd2Czbj.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process- and outcome-based feedback, 2022. URL https://arxiv.org/abs/2211.14275.

Yinjie Wang, Ling Yang, Bowen Li, Ye Tian, Ke Shen, and Mengdi Wang. Revolutionizing reinforcement learning framework for diffusion large language models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/a61023ce36d21010f1423304f8ec49af-Abstract-Conference.html.

Zhihui Xie, Jiacheng Ye, Lin Zheng, Jiahui Gao, Jingwei Dong, Zirui Wu, Xueliang Zhao, Shansan Gong, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Dream-Coder 7B: An open diffusion language model for code, 2025. URL https://arxiv.org/abs/2509.01142.

Zhangchen Xu, Yang Liu, Yueqin Yin, Mingyuan Zhou, and Radha Poovendran. KodCode: A diverse, challenging, and verifiable synthetic dataset for coding. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 6980–7008, 2025. doi: 10.18653/v1/2025.findings-acl.365. URL https://aclanthology.org/2025.findings-acl.365/.

Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Dream 7B: Diffusion large language models, 2025. URL https://arxiv.org/abs/2508.15487.

Qiying Yu, Zheng Zhang, Ruofei Zhu, et al. DAPO: An open-source LLM reinforcement learning system at scale. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. doi: 10.52202/085713-3775. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/a4277440d50f1f15d2cb4c14f7e0c0d2-Abstract-Conference.html.

Siyan Zhao, Devaansh Gupta, Qinqing Zheng, and Aditya Grover. d1: Scaling reasoning in diffusion large language models via reinforcement learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=7ZVRlBFuEv.

Xinyu Zhu, Mengzhou Xia, Zhepei Wei, Wei-Lin Chen, Danqi Chen, and Yu Meng. The surprising effectiveness of negative reinforcement in LLM reasoning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=ftVlLG9cks.

Appendix

A Experimental details and reproducibility

A.1 Hyperparameters

Table 4 lists the training hyperparameters, which are the same for all three base methods and our extensions. Table 5 lists the hyperparameters specific to DPS and SML.

Table 4: **Training hyperparameters.** Settings shared across all base methods (Diffu-GRPO, wd1, GDPO) and our extensions.

Hyperparameter	Value
Model	
Base model	LLaDA-8B-Instruct
Precision	bfloat16
Attention implementation	FlashAttention-2
Quantization	4-bit (QLoRA)
LoRA	
Rank (r)	128
Alpha (α)	64
Dropout	0.05
Generation	
Group size (G)	6
Max completion length	256 tokens
Max prompt length	200 tokens
Block length	32
Diffusion steps	128
Remasking strategy	Low confidence
Random masking	True
Evaluation	
Generation lengths	128, 256, 512
Diffusion steps at eval	64, 128, 256
Eval frequency	Every 100 steps

Hyperparameter	Value
Optimization	
Optimizer	AdamW
Learning rate	3e-6
LR scheduler	Constant with warmup
Warmup ratio	0.0001
Adam β_1 / β_2	0.9 / 0.99
Weight decay	0.1
Max gradient norm	0.2
Per-device batch size	6
Gradient accumulation steps	2
Inner iterations	12
GRPO-specific	
PPO clip range (ϵ)	0.5
KL coefficient (β)	0.0
Prompt masking probability	0.15
Hardware and reproducibility	
GPUs	8 $\times$ NVIDIA H100 80GB
Random seed	42

Table 5: **DACA-GRPO hyperparameters.** Settings specific to DPS and SML. For the main results, the record stride (DPS) and the number of strata and stratification strategy (SML) are selected per setting from the explored ranges (Section 6.1); all other hyperparameters are fixed at their default values. Ranges are explored in the ablation studies of Appendix C.

Component	Hyperparameter	Default	Range explored
DPS *(Denoising Progress Scores)*			
	Modulation strength (λ)	0.1	{0.05, 0.1, 0.2}
	Record stride (s)	–	{1, 2, 4, 8, 16, 32}
	Normalization mode	Per-step	{Per-step, Trajectory, Group, None}
	Last-step mode	Extrapolate	{Raw, Neutral, Mean, Measured, Extrapolate}
SML *(Stratified Masking Likelihood)*			
	Number of strata (K)	4	{2, 3, 4, 5, 6, 7, 8}
	Stratification strategy	Random	{Random, Confidence-based}
	Gradient checkpointing	Function-level	—

A.2 Benchmark details

Table 6 summarizes the benchmarks, training configuration, and which DACA-GRPO components are applied. "Steps" refers to the number of optimizer updates (each step processes one batch across all GPUs with gradient accumulation). For mathematical reasoning and constraint satisfaction tasks, we follow the dataset splits and training protocols of Tang et al. [2026]. For code generation tasks, we

follow Rojas et al. [2026]; HumanEval and MBPP are evaluation-only, and we train on KodCode [Xu et al., 2025]. We construct a synthetic JSON dataset from JSONSchemaBench [Geng et al., 2025].

Table 6: **Training configuration.** Steps denotes optimizer updates. SML is excluded from Countdown and Sudoku due to the distributional mismatch discussed in Section 6.3.

Training set	Task type	Reward type	Steps	DPS	SML
MATH-500	Math reasoning	Format + correctness	6,000	✓	✓
GSM8K	Math reasoning	Format + correctness	7,500	✓	✓
KodCode	Code generation	Format + unit tests	6,000	✓	✓
Countdown	Constraint sat.	Three-level	7,500	✓	✗
Sudoku	Constraint sat.	Per-cell fraction	12,500	✓	✗
JSONSchemaBench (our synthetic data)	Constrained gen.	Multi-component	2,500	✓	✓

Reward functions. Following Tang et al. [2026], we use multi-component reward functions for mathematical reasoning tasks. For MATH-500, the reward combines a format component (+1.0 for an answer wrapped in `\boxed{}` with reasoning, scaled down for partial formatting) and a correctness component (+2.0 if the boxed answer matches the ground truth), yielding scores in $[0.25, 3.0]$. For GSM8K, the reward combines XML structure compliance (+0.125 per correct tag), soft and strict format matching (+0.5 each), integer answer validation (+0.5), and correctness (+2.0 for exact match), yielding scores in $[0, 3.625]$. For constraint satisfaction tasks, Countdown uses a three-level reward: +1.0 if the arithmetic expression reaches the target using exactly the given numbers, +0.1 if the correct numbers are used but the target is missed, and 0 otherwise. Sudoku uses the fraction of correctly filled empty cells as the reward, focusing on solving accuracy rather than copying given values. For code generation, the KodCode training reward has two components: a binary formatting reward (+1.0 if the output is wrapped in ```` ```python ```` markdown fences) and a continuous reward equal to the fraction of passed unit tests. For JSON generation, the reward combines three components, inspired by SO-Bench [Feng et al., 2026]: (1) a *format reward* (+1.0 if the output uses ```` ```json ```` markdown fences, +0.5 if the output is valid JSON without fences, 0 otherwise); (2) a *strict schema reward* (binary +1.0 if the fenced JSON is schema-compliant, 0 otherwise); and (3) a *character-level content reward* using hierarchical field-matching accuracy based on normalized Levenshtein distance. The content reward computes field-matching accuracy between the predicted and ground-truth JSON after recursive normalization (sorting dictionary keys, aligning list elements), squares the score for sharper gradients, and applies a $0.8\times$ penalty multiplier when the prediction violates the schema.

B Additional method details

This section provides an extended discussion of design choices summarized briefly in the main text.

B.1 SML: retaining the fully masked estimate in ratio-based methods

For ratio-based methods (d1, GDPO), the enriched per-token log-probability (Equation (12)) averages each token's fully masked prediction with its stratified prediction rather than using the stratified estimate alone. Including the fully masked term is important for two reasons.

First, it prevents a distributional mismatch between training and generation. During generation, the model starts from a fully masked state with zero inter-token context. If the importance ratio is computed entirely from stratified estimates (where each token sees $(K-1)/K$ context), the model is optimized under a regime that never occurs during actual inference. The fully masked term anchors the ratio to the generation-time distribution.

Second, we empirically observe that using only stratified estimates degrades performance. Training becomes unstable at early stages when the model's predictions under partial context diverge significantly from its predictions under no context. The $1/2$ weighting provides a conservative blend that benefits from bias reduction while maintaining alignment with the generation regime.

Table 7: **DPS record stride, last-step mode, and normalization mode ablation** on MATH-500 with wd1 + DPS ($\lambda = 0.1$). Each mode is run with every record stride $s \in \{1, 2, 4, 8, 16, 32\}$. Last-step columns use per-step normalization; normalization columns use the Extrapolate last-step mode. The shaded columns are the default configuration.

	Last-step mode								*Normalization mode*						*Default*	
	Raw		**Neutral**		**Mean**		**Measured**		**Trajectory**		**Group**		**None**		**Extrapolate + Per-step**	
Stride	**256**	**512**	**256**	**512**	**256**	**512**	**256**	**512**	**256**	**512**	**256**	**512**	**256**	**512**	**256**	**512**
1	**39.2**	35.6	38.5	36.6	36.0	**40.0**	37.3	39.2	36.4	38.8	36.0	38.3	36.4	39.0	38.8	40.3
2	**39.2**	36.0	38.3	35.4	37.7	39.0	**38.1**	39.6	36.7	38.5	37.9	39.8	37.1	38.8	37.3	39.6
4	37.5	36.2	37.7	36.6	**37.9**	39.4	36.7	40.0	37.5	39.6	37.5	**41.0**	**39.0**	39.4	38.8	40.5
8	36.7	33.9	36.9	36.2	36.0	38.6	37.9	**40.2**	**37.9**	39.4	38.3	40.3	37.5	39.4	**39.8**	40.5
16	38.8	**36.6**	**39.7**	**37.1**	37.8	39.0	36.7	39.6	36.7	39.4	36.4	40.9	37.9	39.8	38.5	39.6
32	38.1	35.2	37.9	36.0	37.1	39.4	37.7	39.6	36.2	**40.2**	**39.4**	39.2	36.9	**40.0**	38.6	**40.9**
wd1 baseline	36.7 / 39.6															
Avg.	38.3	35.6	38.2	36.3	37.1	39.2	37.4	39.7	36.9	39.3	37.6	39.9	37.5	39.4	**38.6**	**40.2**
$\geq$ baseline	6/6	0/6	6/6	0/6	4/6	1/6	6/6	5/6	4/6	2/6	4/6	4/6	5/6	2/6	6/6	6/6

C Ablation studies and sensitivity analysis

This section ablates DACA-GRPO design choices on MATH-500, with wd1 as the base method unless otherwise noted, and reports the computational overhead of DPS and SML and the training dynamics of DPS.

C.1 DPS record stride and last-step mode

Table 7 ablates two coupled design choices of DPS: the record stride s and how the final delta δ_T is computed.

Record stride. Computing deltas at every denoising step is possible, but we record a snapshot every s steps and compute each delta between consecutive snapshots (Equation (8)). A stride of s=1 gives the finest credit assignment, but each step unmasks only a few tokens, so each delta reflects a small and potentially unrepresentative change in the model's predictions. Larger strides aggregate more newly unmasked tokens per interval, which gives a higher signal-to-noise ratio at the cost of coarser credit: all tokens born within the same window share a single delta.

Last-step handling. Tokens born at the final recorded snapshot T have no subsequent snapshot $S(T{+}1, \cdot)$ to compute a delta against (Equation (8)). We evaluate five ways to set δ_T:

- **Raw** sets $\delta_T = S(T, T)$. All other deltas are differences of scores, so this substitutes an absolute log-probability for a difference and puts δ_T on a different scale, which distorts the per-step normalization (Equation (9)).
- **Neutral** sets $\delta_T = 0$, so tokens born at the last step receive no extra credit.
- **Mean** sets $\delta_T = \frac{1}{T}\sum_{k<T} \delta_k$, the average of the earlier deltas. This smooths step-level noise but assumes roughly uniform progress along the trajectory, which is at odds with the non-stationary dynamics that motivate per-step normalization.
- **Measured** computes $\delta_T = S_{\text{final}} - S(T, T)$, where S_{final} comes from one additional forward pass on the fully denoised output. This gives the true last-step delta but gives up the zero-cost property of DPS.
- **Extrapolate** (our default) sets $\delta_T = \delta_{T-1}$, assuming that adjacent transitions make comparable progress. It keeps δ_T on the same scale as the other deltas (unlike Raw), keeps credit for the last step (unlike Neutral), and needs no extra computation (unlike Measured).

Extrapolate is the *only* mode that achieves the highest average accuracy at both sequence lengths (38.6 at 256, 40.2 at 512) while matching or exceeding the baseline in every stride configuration (6/6 at both lengths). Raw and Neutral both fail at length 512 (0/6 each), suggesting that scale corruption and information suppression worsen with longer sequences. Measured performs well (6/6 at 256, 5/6

Table 8: **DPS modulation strength ablation** on MATH-500 with wd1 + DPS. Best accuracy (%) across record strides $s \in \{1, 2, 4, 8, 16, 32\}$. Shaded row indicates our default.

λ	**128**	**256**	**512**
0 (baseline)	32.6	36.7	39.6
0.05	33.4	39.2	**40.9**
0.1	33.5	**39.8**	**40.9**
0.2	**35.0**	39.0	40.2

at 512) but sacrifices the zero-cost property of DPS by requiring an extra forward pass. We therefore adopt Extrapolate as the default last-step mode across all experiments.

Under Extrapolate, every stride $s \in \{1, 2, 4, 8, 16, 32\}$ matches or exceeds the wd1 baseline at both lengths, confirming that the DPS signal is reliable across a wide range of recording granularities. The optimal stride varies slightly, but the consistent improvement over the baseline demonstrates that the attribution signal is not an artifact of a particular stride choice.

C.2 DPS normalization mode

Raw deltas δ_k differ in scale across denoising steps, so DPS z-scores them before they enter the weight ω_i (Equation (10)). The modes differ in which deltas share a mean and standard deviation, that is, in what each delta is compared against:

- **Per-step** (our default, Equation (9)) compares a completion's delta at step k with the deltas of the other completions of the same prompt at that step.
- **Trajectory** compares a completion's delta at a step with its deltas at its other steps. Completions are not compared with each other.
- **Group** compares a delta with the mean over all completions in the batch and all steps. Because steps differ systematically in magnitude, the steps with the largest deltas dominate this mean.
- **None** uses the raw deltas without any comparison, so the right λ depends on the scale of the log-probabilities.

The normalization columns of Table 7 vary the mode jointly with the record stride on MATH-500 with wd1 + DPS ($\lambda = 0.1$) at $L \in \{256, 512\}$. Per-step is the only mode that matches or exceeds the baseline in all 12 stride–length configurations, and it has the highest average accuracy (38.6 / 40.2). Trajectory's average at L=512 (39.3) falls below the baseline, consistent with losing the comparison between completions, and Group and None fall below the baseline in 2 and 4 of the six strides at L=512. We therefore use per-step normalization by default.

C.3 DPS modulation strength

The modulation strength λ in Equation (10) sets how strongly DPS reweights per-token advantages; $\lambda = 0$ recovers the base method. Table 8 reports the best MATH-500 accuracy of wd1 + DPS over record strides $s \in \{1, 2, 4, 8, 16, 32\}$ for $\lambda \in \{0.05, 0.1, 0.2\}$, with per-step normalization and the Extrapolate last-step mode. All three values improve over the baseline at every generation length; λ=0.2 is best at L=128 (+2.4pp), and λ=0.1, our default, is best at L=256 and tied-best at L=512.

C.4 SML sensitivity: number of strata and stratification strategy

Figure 5 ablates the number of strata $K \in \{2, \ldots, 8\}$ and the stratification strategy (random vs. confidence-based) on MATH-500 for both Diffu-GRPO and wd1.

SML improves wd1 and Diffu-GRPO robustly. On both methods, the average accuracy across all seven K values exceeds the respective baseline at nearly every generation length. A practitioner choosing K blindly can still expect to see improvements, making SML a low-risk addition. That the pattern holds for both a ratio-free method (wd1) and a ratio-based method (Diffu-GRPO) confirms that the benefit originates from better likelihood estimation rather than from interaction with a specific loss structure.

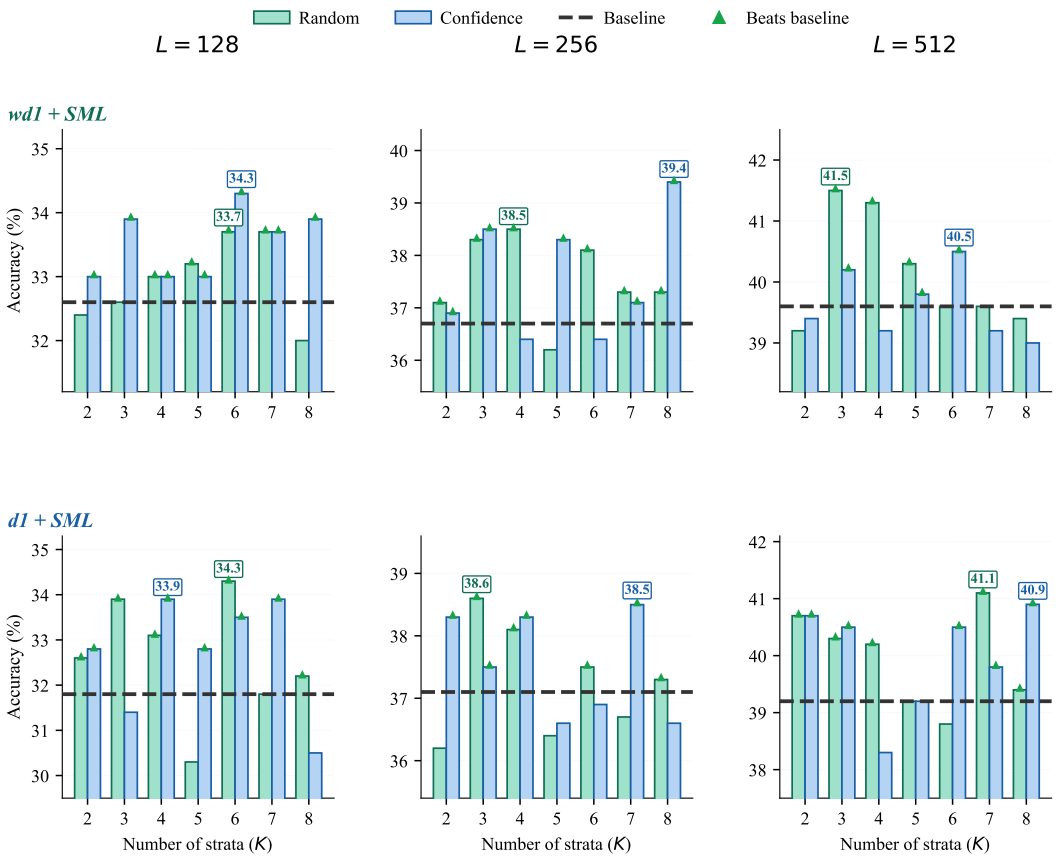

Figure 5: **SML sensitivity on MATH-500 for Diffu-GRPO and wd1.** Grouped bars show accuracy (%) with random (teal) and confidence-based (blue) stratification for $K \in \{2, \ldots, 8\}$ at three generation lengths. Dashed lines: respective baselines without SML. Green triangles (▲) mark configurations that exceed the baseline. SML consistently improves wd1 and Diffu-GRPO across most K values, with random stratification showing a smoother profile.

Random stratification dominates confidence-based stratification. Confidence-based stratification occasionally produces the single highest accuracy at a given K (e.g., K=6 on wd1 at L=128; K=7 on d1 at L=128), but its variance across K is substantially higher. This erratic behavior has a principled explanation: confidence-based sorting relies on prediction entropy from a fully masked forward pass—the same biased regime that SML exists to correct. When entropy estimates are noisy, the resulting strata can be adversarial: hard-to-predict tokens cluster together within a stratum, depriving each other of the contextual benefit that SML is supposed to provide. Random stratification avoids this by ensuring each stratum is a representative sample of the full difficulty spectrum, yielding a smoother profile that requires less per-task tuning.

C.5 Computational overhead

Table 9 reports the wall-clock time of DPS and SML, each added to wd1 and measured over 100 training steps with the default configuration and hardware (Table 4). DPS costs at most 3.2% (at s=1) and under 1% for $s \geq 4$, because it only reads predictions that generation already computes; SML costs 6.5%, 11.7%, and 21.8% at $K = 2$, 4 (our default), and 8, because it adds K forward passes to each loss computation.

C.6 Training dynamics of DPS

Figure 6 compares the training reward of wd1 and wd1 + DPS on Countdown, MATH-500, and Sudoku; each pair of runs shares the seed, batch size, generation length, and reward functions, so the

Table 9: **Computational overhead of DPS and SML.** Wall-clock time measured over 100 training steps on wd1 with LLaDA-8B-Instruct. Overhead (%) is relative to the wd1 baseline.

Configuration	Wall-clock (s)	Overhead (%)
Baseline		
wd1	3 768	—
DPS *(varying record stride s)*		
$s = 1$	3 887	+3.2
$s = 4$	3 797	+0.8
$s = 16$	3 785	+0.5
$s = 32$	3 771	+0.1
SML *(varying number of strata K)*		
$K = 2$	4 014	+6.5
$K = 4$	4 208	+11.7
$K = 8$	4 590	+21.8

only difference is the DPS modulation of the per-token advantage. On Countdown and Sudoku, the wd1 + DPS curves separate from the baseline after roughly 2,000 steps and end $+0.302$ and $+0.356$ above it in mean reward, while on MATH-500 the two curves largely overlap and end $+0.058$ apart.

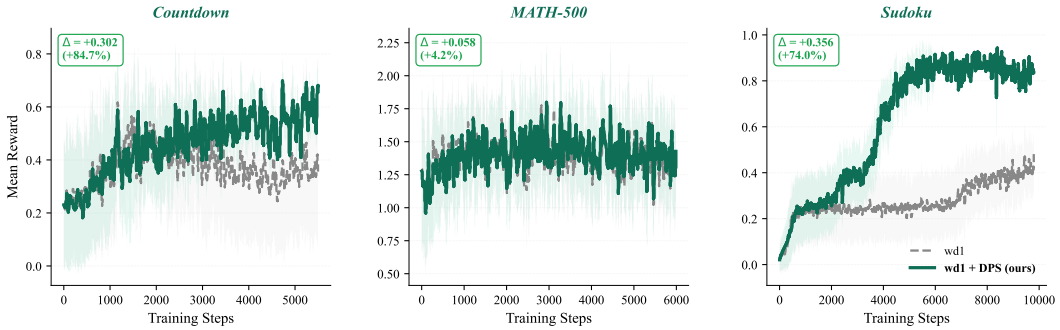

Figure 6: **Training dynamics: wd1 vs. wd1 + DPS.** Mean reward (solid lines) with ± 1 group standard deviation (shaded bands), smoothed over a 20-step rolling window. The Δ box reports the absolute and relative reward improvement at the last plotted step. All runs use identical hyperparameters except for the DPS modulation.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: All claims are backed by experimental results.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Included with Conclusion

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: The paper does not include theoretical results.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The training setup, including hyperparameters, GPUs, optimizers, and data sets, is included in the main paper and the appendices.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: The paper does not promise open code and data at this point.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Provided in the main paper and appendices.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: The paper follows established result reporting for diffusion models.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Hardware is listed in Table 4 and wall-clock time per 100 training steps in Appendix C.5.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conforms to the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No]

    Justification: The paper focuses on improving LLM training and makes no other changes that would require discussion of impacts.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not claim to release code or data.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All methods and datasets are cited.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The paper describes how the synthetic JSON dataset was created. The dataset is not published.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: [N/A]

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: [N/A]

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: [N/A]

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.