PADRE: Pseudo-Likelihood Training for Reasoning Diffusion Language Models

Shiv Shankar
University of Massachusetts
sshankar@cs.umass.edu

Abstract

Policy-gradient reinforcement learning (PGRL) forms the backbone of current methods used to enhance alignment and reasoning in Large Language Models (LLMs). However, these methods are incompatible with diffusion based language models (dLLMs). Most attempts to apply PGRL to dLLMs, are either not scalable or use unprincipled approximations. This work, introduces PADRE a framework that uses a novel pseudo-likelihood based objective for alignment of dLLMs. Our objective has the same optima as PGRL based optimization, but does not need to evaluate exact likelihood from dLLMs. Experiments on various coding and mathematical reasoning benchmarks show that our method matches or surpasses the performance of recent dLLM training baselines such as diffu-GRPO/d1. Our approach provides a stable and practical alternative for RL-based fine-tuning of reasoning-focused dLLMs.

1 Introduction

Large Language Models (LLMs) have become the backbone of modern natural language processing and is powering applications ranging from code generation (Gehring et al., 2024) to robotic control (Wang et al., 2024b) to autonomous agents (Deng et al., 2023) and many other language based tasks (Ouyang et al., 2022). Extensive pre-training on massive text corpora induces impressive language generation capacity (Ouyang et al., 2022). These capabilities are then channeled towards high-value downstream tasks (such as solving mathematical problems) by post-training (or fine-tuning) LLMs to improve their complex reasoning capabilities. Most of the methods in these applications are based on reinforcement learning (RL) to enhance the' reasoning and generation abilities of LLMs (Luong et al., 2024). The core principle of these RL based is to give positive rewards when the model generates correct outputs while penalizing wrong answers, and optimizing for the average reward.

RL methods based on the policy gradient (Sutton et al., 1999) methods (PGRL), have been the dominant approaches for post-training LLMs in reasoning-heavy tasks such as solving mathematical problems (Xu et al., 2025). PGRL estimators require the ability to compute likelihoods of the generations. Most current LLMs are based on autoregressive (AR) transformer models which have a naturally efficient way to compute the requisite likelihoods for the gradient update.

While most current LLMS are based on autoregressive models, recently diffusion based language models (dLLMs) have emerged as an equally powerful way to train language models (Nie et al., 2025; Shi et al., 2024). dLLMs (also sometimes called Masked Diffusion Language Models) model sequence generation as an iterative denoising process, which allows them to break the inherent sequentiality of autoregressive models. This allows dLLMs to significantly outperform autoregressive (AR) models, especially when generating long sequences. Unfortunately, the policy-gradient methods which underpin the success of standard LLMs, cannot directly be applied to dLLMs. This is because the token-level likelihoods used in PGRL are computationally expensive to compute for dLLMs . This is because computing likelihoods of a sentence is difficult for dLLMs.

Interestingly, the standard KL-regularized RL perspective of LLMs training naturally connects to probabilistic inference (Jaynes, 1979; Khalifa et al., 2020; Ziebart et al., 2008)). Building on this insight, we propose a novel objective for alignment of dLLMs that does not rely on monte-carlo estimation of probabilities. Our approach is inspired by pseudo-likelihood (Besag, 1975, 2001) and achieves the same optimality conditions as RLHF and KL-regularized RL; but crucially avoids the inefficiencies of PGRL with dLLMs. This enables stable, and practical alignment for reasoning-intensive tasks in dLLMs.

Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics
Volume 1: Long Papers, pages 5514–5529

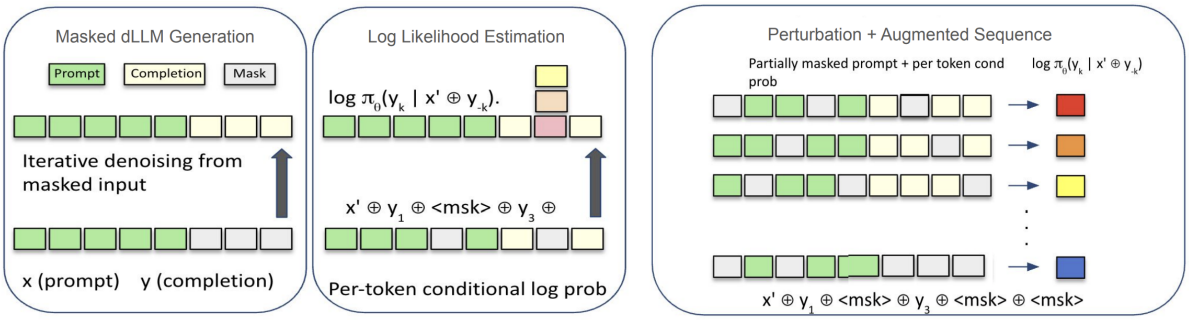

Figure 1: Visual depiction of probability estimation from dLLMs (based upon Figure 2 in Zhao et al. (2025). After generating completion $y^{1:L}$ from prompt x using full diffusion denoising (left), we compute conditional log probabilities from the model. This is achieved by running the model on the augmented sequence $x \oplus y^{1:L} \oplus$ mask sequence. The additional masked sequence is provided with shared positions embedddings as compared to y. Additionally perturbation can be applied to increase robustness of the model.

2 Preliminaries

Proximal Policy Optimization (PPO) PPO (Schulman et al., 2017b) is a classic on-policy algorithm for policy gradient based optimization. While the vanilla Reinforce (Williams, 1992) and other similar gradient methods yield an unbiased gradient, taking large steps often leads to instability in training. Improving upon the TRPO (Schulman et al., 2017a) method, PPO uses a clipped surrogate objective to prevent large charge updates. Each iteration of PPO can be written as optimizing:

$$J(\pi) = \mathbb{E}_{y \in D_k}\Bigg[\min\Bigg(\frac{\pi_\theta(y)}{\pi_{\theta_k}(y)} \cdot A(y), \text{clip}\Big(\frac{\pi(y)}{\pi_{\theta_k}(y)}, \varepsilon\Big) \cdot A(y)\Big)\Bigg]$$

$\widehat{A}$ is an advantage function, ε is a hyperparameter, π_{θ_k} is the previous policy (parameterized via θ), clip is a function which clamps its input in the range $[1-\varepsilon, 1+\varepsilon]$, and D_k is a set of trajectories y obtained by executing π_{θ_k} on the MDP.

Note that the objective involves the likelihood of a generation from the model π. Theoretically the likelihood is of the entire trajectory or output $y^{(i)}$; however when the model allows a per term decomposition such as in auto-regressive models or in state conditioned policies in MDPs, the objective can be written in terms of such a per token likelihood. We refer the readers to Sutton et al. (1999); Sutton and Barto (1998) for details.

Group Relative Policy Optimization (GRPO) GRPO (Shao et al., 2024) is a PPO based method for finetuning LLMs. GRPO usually samples multiple responses $y^{(i)}$ for each prompt x, uses a verifier (for math-like problems) or other reward functions to rate these samples, and computes advantages by normalizing rewards within each prompt group. The advantage for the i-th response y^i is computed as:

$$\widehat{A}^i = \frac{r(x, y^i) - \text{mean}(r(x, y^1), .., r(x, o^G))}{\text{stdev}(r(x, y^1), .., r(x, y^G))}, \tag{1}$$

where $r(x, y^i)$ is the outcome for response y^i to prompt x as we defined above.

This response-level advantage $\widehat{A}^i$ is in the PPO objective $\mathcal{L}^{\text{PPO}}$, along with KL-regularization to compute the update

$$J^{\text{GRPO}}(\pi) = \sum_{i=1}^{G} \frac{1}{|y_i|} \sum_{t=1}^{|y_i|} \min\Bigg[\frac{\pi(a_t^i \mid s_t^i)}{\pi_{\theta_k}(a_t^i \mid s_t^i)}\widehat{A}^i, \text{clip}\Big(\frac{\pi(a_t^i \mid s_t^i)}{\pi_{\theta_k}(a_t^i \mid s_t^i)}, \epsilon\Big)\widehat{A}^i\Bigg] - \beta D_{\text{KL}}(\pi \| \pi_{ref}),$$

where a_t^i is the t^{th} token in the sequence y^i, and $s_t^i = (y_{<t}^i, x)$ is the concatenation of all processed tokens. Effectively instead of a per step/action reward as in PPO; by using the entire trajectory reward in the objective as given, GRPO implicitly assigns each token in the response the corresponding reward. The standardization of the reward replaces the value function estimate in standard PPO. However its overall effect is similar, to stabilize training by reducing variance. Thus GRPO is often simpler to implement than PPO for post-training LLMs (Wang et al., 2024a).

2.1 Masked Diffusion

Our proposed method crucially relies on the architectural and theoretical properties of Masked Diffusion Language Models (dLLMs). Thus, before proceeding further we give an introduction to dLLMs that is relevant for this work. Our presentation here broadly follows the description in Zhao et al. (2025).

The forward process in dLLM is a discrete noising process that gradually corrupts an input sequence x_0 (where x_0 is a sequence of one-hot token vectors) by replacing tokens with a special `[MASK]` token. Let x_t denote the sequence at timestep $t \in [0, 1]$, where $t = 0$ corresponds to the clean input and $t = 1$ corresponds to the fully masked state. The corruption is governed by a noise schedule α_t, which is strictly decreasing in t.

The model π_θ is trained to predict x_0 given x_t, similar to BERT but conditioned on the noise level or in this case the masking level t.

Training Objective Normally an LLM is trained to optimize the likelihood of the text $\pi_\theta(x_0)$, however for dLLMs this is intractable. Instead , one optimizes the Evidence Lower Bound (ELBO) which serves as a lower bound for the true probability. Often instead of a continuous time input, the temporal space $[0, 1]$ is split into T discrete chunks. In that case the ELBO loss becomes:

$$\mathbb{E}\left[\frac{\partial_t \alpha_t}{1-\alpha_t}\sum_{i=1}^{L}\mathbb{I}[x_t^i = \texttt{[MASK]}]\cdot \log \pi_\theta(x_0^i|x_t)\right],$$

where the indicator function $\mathbb{I}[x_t^i = \texttt{[MASK]}]$ ensures only masked tokens contribute to the loss.

3 Related Work

Policy Gradient Methods Policy gradient methods (Williams and Peng, 1990) have been foundational in modern RL. Recent advancements for language model training (Ouyang et al., 2022; Shao et al., 2024) have been based on PPO (Schulman et al., 2017b) and its variants (Wu et al., 2023). However these methods rely on being able to compute the log-probabilities of the generated samples. However, in the context of alignment of fine-tuning diffusion language models, the densities required for computing these are not available efficiently. Furthermore methods which use approximations of the density (Zhao et al., 2025) , are using biased gradients and hence not optimizing the expected reward.

Reinforcement Learning for LLM Reasoning Since the work of Ouyang et al. (2022), application of RL to large language models (LLMs) has seen significant progress, especially in MDP-based formulations of reasoning, as seen in OpenAI's O1 and DeepSeek's R1. While policy-based methods such as GRPO (Guo et al., 2025), and their variants (e.g., DAPO (Yu et al., 2025), Dr. GRPO (Liu et al., 2025)) dominate this space, other approaches like ReMax (Li et al., 2023) and RAFT (Dong et al., 2023) have also been explored. Building on the idea of Bellman Residuals (Schweitzer and Seidmann, 1985; Baird, 1995), recently value based methods (Jia et al., 2025) have also been proposed for post-training of language models.

Language Diffusion Models While diffusion models have revolutionized continuous data generation in the visual domain (Song et al., 2020; Ho et al., 2020), their adaptation to textual data presents unique challenges. The fundamental tension arises from the categorical nature of language tokens, which necessitates specialized approaches beyond the standard diffusion framework. The masked diffusion paradigm has emerged as a particularly successful instantiation of discrete diffusion for language (Sahoo et al., 2024; Shi et al., 2024). This approach, a special case of discrete diffusion (Austin et al., 2021a), has recently achieved significant scaling milestones. Recently, LLaDA-8B (Nie et al., 2025), has been shown to match or surpasses LLaMA-3 8B on MMLU, ARC-C and other few-shot reasoning tasks.

Alignment and Fine-tuning of Language Diffusion Models There have been a few methods proposed for alignment of dLLMs. The naive method to adopt PGRL does not work well due to intractable log-likelihoods. Nie et al. (2025) estimate the log-likelihood $\log p_\theta(y|x)$ via a Monte-Carlo estimate of the ELBO. Instead of Monte-Carlo estimates, Zhao et al. (2025) propose a mean-field variant for the likelihood of an output. This allows them to approximate the likelihood by performs a single denoising step to approximate likelihood for each token position. As mentioned before, this leads to biased optimization, and in practice they also need to randomly masking different portions of the output. However, even with such heuristics, the fundamental problem of using biased gradients persists, and hence even under idealized conditions are not guaranteed to optimize the reward. Zhu et al. (2025) have also noted the issue with the Monte-Carlo ELBO approximation of likelihood, and the issue of variance in the estimate. They propose to use an antithetic sampling based method to reduce this variance. Tang et al. (2025) also

follow the GRPO style training, but to reduce importance sampling ratios, reformulates the training as a weighted likelihood objective with a geometric interpolation between current and reference model as prior.

4 Method

Modern diffusion-based language models, such as LLaDA (Nie et al., 2025), generate sequences by iteratively denoising masked tokens toward complete text. Let x denote the prompt and $y = (y^1, \ldots, y^L)$ the completed generation. Unlike autoregressive (AR) models, which decompose log-likelihoods as $\log \pi_\theta(y|x) = \sum_{l=1}^{L} \log \pi_\theta(y^l|y^{<l}, x)$, diffusion models produce outputs in a non-sequential and non-factorized manner. As such the straightforward method of using PGRL for dLLMs, does not scale. Additionally, the estimation procedure used for computing the probability can be biased, which then leads to unstable behavior. To alleviate this we propose a new probabilistic objective for alignment of dLLMs.

4.1 RL alignment as Probabilistic Inference

We begin with a well known result regarding the KL-constrained RL, which will then motivate a different probabilistic objective that can be used with dLLMs.

Consider the unnormalized target distribution $\tilde{q}(\tau)$ given as:

$$\tilde{q}(\tau) = \pi_{\text{ref}}(\tau) \exp(r(\tau)/\beta), \tag{2}$$

which leads to the Boltzmann distribution (Jaynes, 1979):

$$q(\tau) = \frac{1}{Z} \pi_{\text{ref}}(\tau) \exp(r(\tau)/\beta), . \tag{3}$$

where $Z = \sum_\tau \pi_{\text{ref}}(\tau) \exp(\frac{r(\tau)}{\beta})$ is the normalization constant.

A classic result from Ziebart et al. (2008); Jaynes (1979) shows that maximizing J is equivalent to minimizing $D_{\text{KL}}(\pi_\theta \| q)$. This result also provides multiple pathways for preference-tuning LLMs. The goal is to align a policy distribution π_θ) with an unnormalized target $q \propto \pi_{\text{ref}}(\tau) e^{r(\tau)/\beta}$, where π_{ref} is reference policy and r the reward function. However, instead of committing to optimizing Kullback-Leibler (KL) divergence (which is implicitly what these RL methods are doing), one can consider the broader perspective of matching the model distribution π with the target q. This matching can be implemented by optimizing different divergences. Under ideal conditions (e.g., unlimited model expressivity and global optimization), the final learned policy is invariant to the choice of divergence. However, in practice, different objectives can exhibit varying empirical behaviors.

For our applications, an ideal objective should satisfy three key criteria: a) avoid requiring the partition function Z of $\tilde{q}$ (efficiency) and b) can be computed without access to full densities from π_θ (dLLM) and c) theoretical guarantees of convergence to the target distribution q. In the next section, we discuss an objective based on pseudo-likelihood matching, a candidate objective with such properties.

4.2 Pseudo-likelihood Alignment

The direct minimization of the KL divergence between joint sequence distributions $\pi_\theta(y \mid x)$ and $\tilde{q}(y \mid x)$ is generally infeasible for dLLMs. This is because dLLMs do not directly parameterize a joint distribution over complete sequences. Instead, it defines a collection of conditional distributions corresponding to local denoising steps. As a result, the implied joint distribution is either intractable, requiring summing over many denoising paths. In contrast, dLLMs explicitly trained to model conditional distributions in the denoising function. This motivates an objective defined over the conditional distributions.

Fortunately, for distributions with full support, the joint distribution is uniquely determined by its single-token conditional marginals. This follows from the classic Hammersley–Clifford factorization (Clifford and Hammersley, 1971) of positive distributions. In practice, modern LLMs define distributions with full or near-full support, and strict positivity can be enforced if needed via temperature scaling. Thus, matching the per token conditional distributions is sufficient for matching two distributions.

Theorem 1 (Conditional equality implies joint equality)**.** Let p and q be strictly positive distributions over $y = (y^1, \ldots, y^L)$. Define y^{-i} as the sequence y with the i-th token removed. If $p(y^i \mid y^{-i}) = q(y^i \mid y^{-i})$ for all i and all y^{-i}, then $p(y) = q(y)$ for all y.

Motivated by Theorem 1, we wish to align the conditional marginals of $\pi_\theta(y|x)$ and $\tilde{q}(y|x)$. A natural objective to match all the conditionals is

the sum of all the conditional KL divergences:

$$\sum_{i=1}^{|L|} \sum_{y^{-i}} \mathrm{KL}(\pi_\theta(y^i|y^{-i}) \| \tilde{q}(y^i|y^{-i})). \quad (4)$$

where we have suppressed the dependence on the prompt x for notational convenience. This objective vanishes if and only if all conditional distributions match, which by Theorem 1 implies that $\pi_\theta = \tilde{q}$. However, directly evaluating this loss is intractable due to the combinatorial size of the context space y^{-i}.

To address this computational bottleneck, we introduce a weighted variant of the objective in eq. (4). The key idea is that instead of treating all the terms in the sum equally, we score them according to their probability under the model distribution $\pi_\theta(y^{-i})$.

$$J = \sum_y \sum_i \pi_\theta(y^{-i}) \mathrm{KL}(\pi_\theta(y^i|y^{-i}) \| \tilde{q}(y^i|y^{-i})) \quad (5)$$

$$= \sum_y \sum_i \pi_\theta(y^{-i}) \mathbb{E}_{y^i \sim \pi_\theta(\cdot|y^{-i})} \left[\log \frac{\pi_\theta(y^i|y^{-i})}{\tilde{q}(y^i|y^{-i})} \right] \quad (6)$$

$$= \sum_y \sum_i \pi_\theta(y^{-i}) \pi_\theta(\cdot|y^{-i}) \left[\log \frac{\pi_\theta(y^i|y^{-i})}{\tilde{q}(y^i|y^{-i})} \right] \quad (7)$$

$$= \sum_y \sum_i \pi_\theta(y) \left[\log \frac{\pi_\theta(y^i|y^{-i})}{\tilde{q}(y^i|y^{-i})} \right] \quad (8)$$

$$= \sum_i \mathbb{E}_{y \sim \pi_\theta(y)} \left[\log \frac{\pi_\theta(y^i|y^{-i})}{\tilde{q}(y^i|y^{-i})} \right] \quad (9)$$

By multiplying each term by $\pi_\theta(y_{-i})$ and applying the law of total expectation, the weighted KL objective turns into our final tractable objective:

$$J_{PKL}(\theta) = \mathbb{E}_{y \sim \pi_\theta} \sum_{i=1}^{L} \Big[\log \pi_\theta(y^i|y^{-i}) - \log \tilde{q}(y^i|y^{-i}) \Big] \quad (10)$$

This formulation offers two main advantages. First, it eliminates the need for enumerating contexts or of explicit sampling from the conditional distributions $\pi_\theta(\cdot|y^{-i})$. Instead this objective can be estimated using samples from π_θ alone, allowing for online learning. Second, it maintains the theoretical property that the global optimum is achieved precisely when all conditional distributions match their target counterparts, provided π_θ has full support. To see this note that Eqn 5 and Eqn 4 are both sum of the same positive terms (the KL divergences). Since the weights themselves are positive, the global minimum value of either objective is achieved only when each individual term is zero.

We further note that the conditional marginal distribution of q inherits the same exponential tilted structure as the reference model:

$$\tilde{q}(y^i|y^{-i}) \propto \pi_{\text{ref}}(y^i|y^{-i}) \exp(r(y^i, y^{-i})) \quad (11)$$

This enables efficient computation when π_{ref}'s conditionals are tractable. The log-conditional of $\tilde{q}$ simply becomes:

$$\log \tilde{q}(y^i|y^{-i}) = \log \pi_{\text{ref}}(y^i|y^{-i}) + r(y^i, y^{-i}) - \log Z(y^{-i})$$

where $Z(y^{-i})$ is the conditional partition function. Importantly, the $Z(y^{-i})$ term is independent of the model π_θ. Thus it can be omitted from the objective, as it contributes an additive constant that does not affect the optimization landscape.

Relation to pseudo-likelihood Besag (1975) provides a statistically consistent approach to learning joint distributions through their local behaviour. Similar to the previous discussion, the basic idea in Besag (1975) is to represent the multivariate joint distribution using the respective conditional distributions. Specifically, for a sequence $y = (y^1, \ldots, y^L)$ modeled by π_θ, consider the product of its conditional marginals:

$$\mathrm{PL}(y) = \prod_{i=1}^{L} \pi_\theta(y^i|y^{-i}), \quad (12)$$

where y^{-i} denotes the sequence with the i-th token removed. Besag (1975) established that this product (also called pseudo-likelihood) forms a proper statistical score function i.e. optimizing PL will converge to the true underlying generative model.

Taking expectations under π_θ, the negative pseudo-log-likelihood corresponds to

$$-\mathbb{E}_{y \sim \pi_\theta} \sum_{i=1}^{L} \log \pi_\theta(y^i \mid y^{-i}),$$

which is the first term of J_{PKL} (Eqn 10). The second term in Eqn 10, replaces the empirical data distribution with a target conditional model $\tilde{q}$ (which

is fixed). Comparing this against the KL decomposition ($\int p(y)[\log p(y) - \log q(y)]dy$) and likelihood, J_{PKL} can be seen as a pseudo-KL divergence between the distributions π_θ and q.

Optimization We can use the gradient equivalence between KL optimization and entropic RL (Se 4.1), to turn Eq 10 into RL like alignment procedure (detail in the Appendix). Sepecifically, we can write the following PG style gradient rule for the objective in Eqn 10.

$$\begin{aligned}\nabla J_{PKL} &= \mathbb{E}_{y\sim\pi_\theta}\Big[\nabla_\theta \log \pi(y^i|y^{-i},x) \\ &\Big(\log \pi_\theta(y^i|y^{-i}) - \log \tilde{q}(y^i|y^{-i})\Big)\Big] \\ &= \mathbb{E}_{y\sim\pi_\theta}\Big[\nabla_\theta \log \pi(y^i|y^{-i},x) \\ &\Big(\log \pi_\theta(y^i|y^{-i}) - \log \pi_{ref}(y^i|y^{-i}) - r(y)\Big)\Big]\end{aligned} \tag{13}$$

To achieve stable training and for efficient use of multiple generations, we still follow the GRPO recipe of group normalized advantage. We train in an online fashion, i.e. as the model trains, new completions are generated and rewards computed on them to compute new gradient steps. To do this effectively, we once again need to go back to the PPO/GRPO like setup where we do clipping of the weighting ratio.

$$\begin{aligned}&\sum_{l=1}^{|L|} \min\left[\frac{(\pi_\theta(y^l \mid y_{-l},x)}{(\pi_{ref}(y^l \mid y^{-l},x)}\widehat{A}, \text{clip}\left(\frac{(\pi_\theta(y^l \mid y^{-l},x)}{(\pi_{ref}(y^l \mid y_{-l},x)}, \epsilon\right)\widehat{A}\right] \\ &- \sum_{l=1}^{|L|} \beta D_{\text{KL}}((\pi_\theta(y^l \mid y_{-l},x)||(\pi_{ref}(y^l \mid y_{-l},x)),\end{aligned} \tag{14}$$

Remark 1. *An astute observer may note that the training objective seems to replace the likelihoods in the GRPO/PGRL update with pseudo-likelihoods; and the overall approach can be considered a different way to approximate the likelihood. While practically this interpretation is reasonable, using incorrect likelihoods for PGRL methods can lead to incorrect optimization. Furthermore, the corresponding parameter update can be incompatible with any objective (Nota and Thomas, 2019); and hence it is not clear what such an optimization procedure does. We instead develop a principled argument for why replacing the likelihood by pseudolikelihoods is a mathematically reasonable and does not change the optima of the training.*

4.3 Evaluating Conditional Likelihood

To evaluate the aforementioned objective efficiently we still need to be able to compute the conditional densities. For approaches similar to LLaDA (Nie et al., 2025), which learn a masked diffusion model, one can efficiently approximate conditional token probabilities for any given token trajectory. This trick stems from the unique factorization properties of the reverse diffusion kernel in these models. While originally used for full likelihood, we use it for quickly estimating the conditional likelihood.

Consider a clean sequence $y_0 = [y_0^1, \ldots, y_0^L] \in \mathcal{V}^L$ be a token sequence. and its corrupted version y_t at timestep t, where `[MASK]` denotes the mask token. The key insight lies in the structure of the reverse process kernel, which factorizes over masked positions M_t:

$$\pi_\theta(y_{t-1} \mid y_t) = \prod_{k\in M_t} f_\theta(y_k \mid y_t) \prod_{k\notin M_t} \delta(y_t^k) \tag{15}$$

where f_θ is the denoising network and δ maintains unmasked tokens. This factorization emerges from the independent token corruption in the forward process. Here we explicitly wrote the denoiser as f to distinguish it from the implied likelihood model π_θ.

To compute $\pi_\theta(y^i \mid y_0^{-i})$, we construct a partially masked sequence $y_1 = y_0^{-i} \cup \{\texttt{[MASK]}_i\}$ where only position i is masked. This represents a valid sample from the forward process with $M = \{i\}$. Through approximate marginalization and Markovian factorization one can approximate the conditional probability via the denoiser's output as

$$p_\theta(w_i \mid y_0^{-i}) = f_\theta(w_i \mid y_1) \tag{16}$$

We note that this is a commonly used trick, and we just adapt it to the conditional likelihood term. For more detailed calculation we refer the readers to the Appendix as well as the work of Zhao et al. (2025).

4.4 Efficient multi-conditional evaluation

The previous paragraphs described evaluation of a single conditional. However the optimization objective uses conditionals for all token positions. One way to achieve this is to simply randomize over token positions, which provides unbiased gradient estimates. However this slows down the optimization process, and a more efficient process can

be desirable. To be able to achieve this, we next propose a slight tweak to the objective, which will allow us to use a single pass through the model. Another disadvantage of this trick is that we need to keep track of the sequence in which the tokens were revealed during sentence generation. We highlight this trick next.

Assume that the denoiser is modeled by a transformer. Transformers are position inveriant and one typically uses some form of position embedding to induce it in the model. However this also means, that if a two identical tokens are provided at different positions but with the same position embedding, the downstream computation of these tokens remains identical (assuming they do not attend on each other). We will use this feature, and pad the prompt with the output tokens but with shared position embeddings. Specifically, if y is the decoded sequence and x the prompt, then we construct the augmented input

$$\tilde{x} = x \oplus y^{1:L} \oplus m^{L+1:2L},$$

For the masked position m^{L+i} will use the [MASK] token paired with position i i.e. we will tie the positional encodings by setting $\text{pos}(L+i) = \text{pos}(i)$ for each i.

Define an attention mask $\mathcal{M}$ with the following constraints:
(A) Token rows of x and from y positions $1:L$ cannot attend to any mask rows $L{+}1{:}2L$.
(B) Mask row $L+i$ may attend to all tokens in x, but it cannot attend to token i (in y). It can also attend to itself $L+i$; but it may not attend to other mask rows $L+j$ ($j \neq i$).

Constraint **(A)** ensures the hidden states of the tokens $y_{1:L}$ are same to those obtained without the appended masks (no feedback from masks to tokens). Constraint **(B)** ensures each appended mask $L+i$ sees only the visible context y_{-i} (and its own mask embedding) and no other mask information. Finally having the same position embedding $L+i$ and i means that the computation at $L+i$ conditioned on other hidden embeddings is the same as if it was at i. Therefore, for a single transformer layer the computation at row $L+i$ coincides with evaluating the denoiser on the hypothetical input x_t^i (only i masked).

We would like to extend this trick to multiple layers; however note that y^j has attended to y_i in a previous layer, and the computation ahead is no longer exactly the same. We propose two solutions to this challenge:

- PKL-U Choose a subset of indices $U \subset [1, L]$, mask all attention from $j \in [1, L]$ to $j' \in U$ (in the y tokens), as well as any attention between them in mask. This effectively tries to approximate $\pi_\theta(y^i|y^{-i}, x)$ with a mean field approximation of $\pi_\theta(y^U|y^{-U}, x)$ where y^{-U} skips any positions in U. This is still stochastic and does not include all tokens in L. This is similar to the perturbation technique used in Zhao et al. (2025).

- PKL-T Use the sequence of tokens unveiled during denoising. Define for every any position i, t_i as the time step where it would have become non-mask token . Let U_i be the set of all positions before i that are unmasked i.e. $U_i = j$ such that $t_j > t_i$. During unmasking, at time t_i, the attention on any non-mask token could only have been on U_i. Now we use the same mask on the mask tokens m_{L+i} i.e. allow it to only attend on self-position , the prompt x, and the positions of y corresponding to U_i. Thus we approximate $\pi_\theta(y^i|y^{-i}, x)$ as $\pi_\theta(y_t^i|y_{t+1}^{U_i}, x)$. A similar trick is used for any-order models by Hoogeboom et al. (2021).

As a consequence, the gradient of the pseudo-divergence loss can be computed in much fewer than L forward pass by reading the L mask rows $L+1, \ldots, 2L$. Note that this construction *does not* attempt to reproduce decode-time logits or a trajectory factorization; it computes *conditionals of the joint*, so attending to "future" tokens (under any decode order) is not only allowed but required, because they belong to y_{-i}.

Prompt Perturbation Additionally, following Zhao et al. (2025), we consider masking tokens in the prompt x during likelihood computation. Formally if the prompt is $x = [x^1, x^2, .., x^K]$, each token gets independently dropped byprobability p_{drop} . So for example from x we get the pertubed prompt $x_{dr} = [x^1, \texttt{[MASK]}, .., x^K]$ with probability p_{drop}. his has a regularizing effect, as these prompts implicitly act as dropout based augmentation. While our model is able to learn normally, we found that having such perturbations helped the model generalize better.

Algorithm 1 Optimization

Initialize $\pi_\theta \leftarrow \pi_{\text{ref}}$
while not converged **do**
  Sample a prompt $x \sim \mathcal{D}$
  Sample G completions $y \sim \pi_\theta(\cdot \mid x),\ i \in [G]$
  For each y, compute reward r
  Normalize to get GRPO style reward $\hat{r}(\pi_\theta)$
  $x' \leftarrow$ randomly mask tokens of prompt x with probability p
  For π_θ, π_{ref}, get conditional log-probabilities of y^i_k given x', y^i_{-k}
  Update π_θ by gradient ascent on Equation 14
end while
  return π_θ

5 Experiments

We first show the issue with using the one-step approximation in Zhao et al. (2025). Specifically, for the LLaDA-8B model (Nie et al., 2025), we compute the KL divergence between the true probability (obtained by summing over all time steps) vs the different approximations discussed in this work. These include the one-step mean-field approximation of d1 Zhao et al. (2025), the method described in Section 4.3 (denoted PKL), and the two heuristics in Section 4.4 (labeled PKL-U and PKL-T). These were evaluated on GSM8K (Cobbe et al., 2021) and MATH500 (Hendrycks et al., 2021) datasets [1]. For d1 we get this value to be 2.87 and 3.15, respectively, showing that the one-step approximation is not great. Next, we do the same for the conditional likelihood terms (without perturbation) used in our method against the true terms. For the PKL version we found the average conditional KL to be around or less than 1. While this means our proposal is much more accurate, it still is not perfect. Additionally the two faster alternatives PKL-U and PKL-T , while worse than PKL are better than d1. On the two datasets, their relative numbers did not show any pattern between the two methods. These results are presented in fig. 2

Next for our experiments, we use the recent dLLM LLaDA-8B (Nie et al., 2025) as the baseline model, which we then fine-tune based on different alignment methods. Our experiments are based on the code and hyperparameter provided in

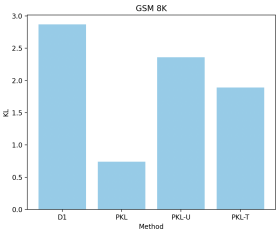

Figure 2: KL divergence comparison between different methods on the GSM8K and MATH500 datasets. d1 shows the KL divegence between true model and the mean-field approximation. The others compare the average KL divergence between the true conditional and the approximate conditionals.

Zhao et al. (2025) [2]. Experiments are conducted on multiple benchmarks including math reasoning and coding benchmarks. For math benchamrks we use: a) GSM8K (Cobbe et al., 2021), a dataset of multi-step grade school math problems, b) MATH (Hendrycks et al., 2021) a set of high-level math problems and c) MATH500(Lightman et al., 2023), a curated subset of MATH as well as coding benchmarks. Coding abilities are evaluated on: a) HumanEval (Chen, 2021) and b) MBPP (Austin et al., 2021b). We also report results from the dream model of Ye et al. (2025), VRPO of Zhu et al. (2025) and wd1 of Tang et al. (2025) as additional baseline comparisons.

Methodology For fair comparison, we follow the experimental procedure of Zhao et al. (2025). For rewards, we use the composite reward function that combines formatting and correctness rewards as recommended by Zhao et al. (2025). We test our model under 0-shot prompting with the prompts as reported in Zhao et al. (2025). The results on math reasoning tasks (reported in Table 1 demonstrate that PKL outperforms all baselines across benchmarks. The code benchmark results are presented in the Appendix.

We also perform an ablation on the use of prompt token perturbation in the prompt for pseudolikelihood computation. This version (called PKL$^+$) is compared along with baselines as well as with the heuristic methods of PKL-U and PKL-T. We see that even the simpler version is competitive with Dream on MATH and outperforms d1. On GSM8K, it achieves 87.3% accuracy, significantly surpassing competing methods. On the MATH500, PKL$^+$ reaches 42.4%, outperforming d1 by +2.2% . Additionally we see that PKL$^+$ outperforms DREAM

[1] Due to the computational cost of computing the probabilities, this was done on a small subset of test data

[2] available at https://github.com/dllm-reasoning/d1

Table 1: Performance of Diffusion Language Models on Math Benchmarks. We can see that our proposed method matches our outperforms other methods. All baseline results are from literature. The results denoted by N/A are due to the corresponding paper not having reported the results.

Method	GSM8K	MATH500	MATH
Dream 7B	81.1	N/A	42.9
VRPO	83.3	36.8	42.6
wd1	82.3	39.0	N/A
LLaDA 8B	78.3	36.2	38.9
+ SFT	81.1	34.8	N/A
+ diffu-GRPO	81.9	39.2	N/A
+ d1-LLaDA	82.1	40.2	N/A
PKL-U	83.6	41.9	43.2
PKL-T	84.8	41.2	43.0
PKL	85.6	40.9	41.6
PKL$^+$	87.3	42.4	44.3

(Ye et al., 2025) on MATH.

6 Conclusion

We have introduced a novel approach to tune dLLMs for reasoning . Our method is a version of the pseudo-likelihood training (Besag, 1975). The probabilistic objective only relies on sampling and estimating conditional distributions, both of which are efficient with dLLMs. Additionally, unlike other methods, our approach learns the same optima as standard PGRL methods. Experiments show that our method matches or outperforms other methods for finetuning dLLMs on challenging math reasoning as well as coding based tasks.

7 Limitations

The biggest limitation of our proposed method is theoretically principled scaling. While we present different heuristics to quickly approximate likelihoods, such approximation have the same theoretical problem as the methods of Nie et al. (2025) and Zhao et al. (2025). Additionally, we experimented only on limited datasets. Previous works have also experimented with tasks like simple sudoku solving, and the dLLM methods have generally been worse at solving those tasks. That not only might apply to our model, but might make it worse as sudoku like challenges require long steps of reasoning, where the conditioning approximations might be worse.

References

Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. 2021a. Structured denoising diffusion models in discrete state-spaces. *Advances in neural information processing systems*, 34:17981–17993.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. 2021b. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Mohammad Gheshlaghi Azar, Zhaohan Daniel Guo, Bilal Piot, Remi Munos, Mark Rowland, Michal Valko, and Daniele Calandriello. 2024. A general theoretical paradigm to understand learning from human preferences. In *International Conference on Artificial Intelligence and Statistics*, pages 4447–4455. PMLR.

Leemon Baird. 1995. Residual algorithms: Reinforcement learning with function approximation. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 30–37.

Julian Besag. 1975. Statistical analysis of non-lattice data. *Journal of the Royal Statistical Society Series D: The Statistician*, 24(3):179–195.

Julian Besag. 2001. [conditionally specified distributions: An introduction]: Comment. *Statistical Science*, 16(3):265–267.

Mark Chen. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Leshem Choshen, Lior Fox, Zohar Aizenbud, and Omri Abend. 2019. On the weaknesses of reinforcement learning for neural machine translation. *arXiv preprint arXiv:1907.01752*.

Peter Clifford and John M Hammersley. 1971. Markov fields on finite graphs and lattices.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. 2023. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36:28091–28114.

Hanze Dong, Wei Xiong, Deepanshu Goyal, Yihan Zhang, Winnie Chow, Rui Pan, Shizhe Diao, Jipeng Zhang, Kashun Shum, and Tong Zhang. 2023. Raft: Reward ranked finetuning for generative foundation model alignment. *arXiv preprint arXiv:2304.06767*.

Jonas Gehring, Kunhao Zheng, Jade Copet, Vegard Mella, Quentin Carbonneaux, Taco Cohen, and Gabriel Synnaeve. 2024. Rlef: Grounding code llms

in execution feedback with reinforcement learning. *arXiv preprint arXiv:2410.02089*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. 2017. Reinforcement learning with deep energy-based policies. In *International Conference on Machine Learning*, pages 1352–1361. PMLR.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pages 1861–1870. PMLR.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851.

Emiel Hoogeboom, Alexey A Gritsenko, Jasmijn Bastings, Ben Poole, Rianne van den Berg, and Tim Salimans. 2021. Autoregressive diffusion models. *arXiv preprint arXiv:2110.02037*.

Edwin T Jaynes. 1979. Concentration of distributions at entropy maxima. *ET Jaynes: Papers on probability, statistics and statistical physics*, page 315.

Zeyu Jia, Alexander Rakhlin, and Tengyang Xie. 2025. Do we need to verify step by step? rethinking process supervision from a theoretical perspective. *arXiv preprint arXiv:2502.10581*.

Muhammad Khalifa, Hady Elsahar, and Marc Dymetman. 2020. A distributional approach to controlled text generation. *arXiv preprint arXiv:2012.11635*.

Tomasz Korbak, Hady Elsahar, Germán Kruszewski, and Marc Dymetman. 2022. On reinforcement learning and distribution matching for fine-tuning language models with no catastrophic forgetting. *Advances in Neural Information Processing Systems*, 35:16203–16220.

Ziniu Li, Tian Xu, Yushun Zhang, Zhihang Lin, Yang Yu, Ruoyu Sun, and Zhi-Quan Luo. 2023. Remax: A simple, effective, and efficient reinforcement learning method for aligning large language models. *arXiv preprint arXiv:2310.10505*.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *arXiv preprint arXiv:2305.20050*.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. 2025. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*.

Trung Quoc Luong, Xinbo Zhang, Zhanming Jie, Peng Sun, Xiaoran Jin, and Hang Li. 2024. Reft: Reasoning with reinforced fine-tuning. *arXiv preprint arXiv:2401.08967*, 3.

Gergely Neu, Anders Jonsson, and Vicenç Gómez. 2017. A unified view of entropy-regularized markov decision processes. *arXiv preprint arXiv:1705.07798*.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. 2025. Large language diffusion models. *Preprint*, arXiv:2502.09992.

Chris Nota and Philip S Thomas. 2019. Is the policy gradient a gradient? *arXiv preprint arXiv:1906.07073*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36:53728–53741.

Subham Sekhar Sahoo, Marianne Arriola, Aaron Gokaslan, Edgar Mariano Marroquin, Alexander M Rush, Yair Schiff, Justin T Chiu, and Volodymyr Kuleshov. 2024. Simple and effective masked diffusion language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. 2017a. Trust region policy optimization. *Preprint*, arXiv:1502.05477.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017b. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Paul J. Schweitzer and Abraham Seidmann. 1985. Generalized polynomial approximations in markovian decision processes. *Journal of Mathematical Analysis and Applications*, 110(2):568–582.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, et al. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis Titsias. 2024. Simplified and generalized masked diffusion for discrete data. *Advances in neural information processing systems*, 37:103131–103167.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. 2020. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*.

Richard S. Sutton and Andrew G. Barto. 1998. *Reinforcement Learning: An Introduction*. MIT Press.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. 1999. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12.

Xiaohang Tang, Rares Dolga, Sangwoong Yoon, and Ilija Bogunovic. 2025. wd1: Weighted policy optimization for reasoning in diffusion language models. *Preprint*, arXiv:2507.08838.

Haoxiang Wang, Yong Lin, Wei Xiong, Rui Yang, Shizhe Diao, Shuang Qiu, Han Zhao, and Tong Zhang. 2024a. Arithmetic control of llms for diverse user preferences: Directional preference alignment with multi-objective rewards. *Preprint*, arXiv:2402.18571.

Zihan Wang, Brian Liang, Varad Dhat, Zander Brumbaugh, Nick Walker, Ranjay Krishna, and Maya Cakmak. 2024b. I can tell what i am doing: Toward real-world natural language grounding of robot experiences. *arXiv preprint arXiv:2411.12960*.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256.

Ronald J Williams and Jing Peng. 1990. An efficient gradient-based algorithm for on-line training of recurrent network trajectories. *Neural computation*, 2(4):490–501.

Tianhao Wu, Banghua Zhu, Ruoyu Zhang, Zhaojin Wen, Kannan Ramchandran, and Jiantao Jiao. 2023. Pairwise proximal policy optimization: Harnessing relative feedback for llm alignment. *arXiv preprint arXiv:2310.00212*.

Fengli Xu, Qianyue Hao, Zefang Zong, Jingwei Wang, Yunke Zhang, Jingyi Wang, Xiaochong Lan, Jiahui Gong, Tianjian Ouyang, Fanjin Meng, Chenyang Shao, Yuwei Yan, Qinglong Yang, Yiwen Song, Sijian Ren, Xinyuan Hu, Yu Li, Jie Feng, Chen Gao, and Yong Li. 2025. Towards large reasoning models: A survey of reinforced reasoning with large language models.

Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. 2025. Dream 7b.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, et al. 2025. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*.

Siyan Zhao, Devaansh Gupta, Qinqing Zheng, and Aditya Grover. 2025. d1: Scaling reasoning in diffusion large language models via reinforcement learning. *arXiv preprint arXiv:2504.12216*.

Fengqi Zhu, Rongzhen Wang, Shen Nie, Xiaolu Zhang, Chunwei Wu, Jun Hu, Jun Zhou, Jianfei Chen, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. 2025. Llada 1.5: Variance-reduced preference optimization for large language diffusion models. *Preprint*, arXiv:2505.19223.

Brian D. Ziebart. 2010. *Modeling Purposeful Adaptive Behavior with the Principle of Maximum Causal Entropy*. Ph.D. thesis, Carnegie Mellon University.

Brian D. Ziebart, Andrew L. Maas, J. Andrew Bagnell, and Anind K. Dey. 2008. Maximum entropy inverse reinforcement learning. In *AAAI Conference on Artificial Intelligence*.

8 Additional Derivations

8.1 Policy Gradient for Pseudo-Divergence

We consider, for a fixed position i and context x,

$$\mathcal{L}_i(\theta) \;:=\; \mathbb{E}^{\text{no-grad}}_{y_{-i}\sim\pi_\theta(\cdot|x)}\Big[\mathrm{KL}\Big(\pi_\theta(y_i \mid y_{-i}, x)\,\Big\|\, q(y_i \mid y_{-i}, x)\Big)\Big], \tag{17}$$

where the outer expectation is treated with a stop–gradient (i.e., no gradient flows through $\pi_\theta(y_{-i} \mid x)$), and $q(\cdot \mid y_{-i}, x)$ does *not* depend on θ.

For fixed y_{-i}, write $p_\theta(\cdot) = \pi_\theta(\cdot \mid y_{-i}, x)$ and $q(\cdot) = q(\cdot \mid y_{-i}, x)$. Then

$$\mathrm{KL}(p_\theta \| q) = \mathbb{E}_{y_i\sim p_\theta}[\log p_\theta(y_i) - \log q(y_i)]. \tag{18}$$

The score-function identity (Williams and Peng, 1990) says that

$$\nabla_\theta \mathbb{E}_{p_\theta}[f] = \mathbb{E}_{p_\theta}[\nabla_\theta \log p_\theta \, f + \nabla_\theta f]$$

Applying this, we obtain

$$\begin{aligned}
\nabla_\theta \,\mathrm{KL}(p_\theta \| q) &= \mathbb{E}_{y_i\sim p_\theta}\Big[\nabla_\theta \log p_\theta(y_i)\,(\log p_\theta(y_i) - \log q(y_i)) \;+\; \nabla_\theta \log p_\theta(y_i)\Big] \\
&\quad - \;\mathbb{E}_{y_i\sim p_\theta}\Big[\nabla_\theta \log p_\theta(y_i)\;\log q(y_i)\Big] \\
&= \mathbb{E}_{y_i\sim p_\theta}\Big[\nabla_\theta \log p_\theta(y_i)\,(\log p_\theta(y_i) - \log q(y_i))\Big],
\end{aligned} \tag{19}$$

since $\mathbb{E}_{p_\theta}[\nabla_\theta \log p_\theta] = 0$.

Putting (19) inside (17) and recalling that the outer sampling over y_{-i} is held fixed (no score term from $\pi_\theta(y_{-i} \mid x)$), we get

$$\boxed{\nabla\mathcal{L}(\theta) = \mathbb{E}^{\text{no-grad}}_{y_{-i}\sim\pi_\theta(\cdot|x)}\;\mathbb{E}_{y_i\sim\pi_\theta(\cdot|y_{-i},x)}\Big[\nabla_\theta \log \pi_\theta(y_i \mid y_{-i}, x)\,\Big(\log \pi_\theta(y_i \mid y_{-i}, x) - \log q(y_i \mid y_{-i}, x)\Big)\Big]} \tag{20}$$

8.2 Conditional Likelihood Evaluation

Consider a clean sequence $y_0 = [y_0^1, \ldots, y_0^L] \in \mathcal{V}^L$ be a token sequence. and its corrupted version y_t at timestep t, where `[MASK]` denotes the mask token. The key insight lies in the structure of the reverse process kernel, which factorizes over masked positions M_t:

$$\pi_\theta(y_{t-1} \mid y_t) = \prod_{k\in M_t} f_\theta(y_k \mid y_t) \prod_{k\notin M_t} \delta(y_t^k) \tag{21}$$

where f_θ is the denoising network and δ maintains unmasked tokens. This factorization emerges from the independent token corruption in the forward process. Here we explicitly wrote the denoiser as f to distinguish it from the implied likelihood model π_θ.

To compute $\pi_\theta(y^i \mid y_0^{-i})$, we construct a partially masked sequence $y_1 = y_0^{-i} \cup \{\texttt{[MASK]}_i\}$ where only position i is masked. This represents a valid sample from the forward process with $M = \{i\}$.

Through marginalization over latent variables $y_{1:T}$, we have:

$$\pi_\theta(w_i \mid y_{0,-i}) = \sum_{y_{1:T}} \pi_\theta(w_i \mid y_1)\pi_\theta(y_{1:T} \mid y_0^{-i}) \tag{22}$$

The Markov property of the diffusion process ensures w depends only on y_1. Crucially, y_1 is approximately deterministic given y_0^{-i}, causing the summation to collapse [3]. We note that this is a commonly

[3] More accurately a single conditioned term is used as the approximation

used trick, and we just adapt it to the conditional likelihood term. For more detailed calculation we refer the readers to Zhao et al. (2025).

$$\pi_\theta(w_i \mid y_0^{-i}) = \pi_\theta(w_i \mid y_1) \tag{23}$$

Examining the reverse kernel reveals:

$$\pi_\theta(y_0 \mid y_1) = f_\theta(w^i \mid y_1) \prod_{j \neq i} \delta(y_1^j) \tag{24}$$

Thus, the conditional probability can be quickly approximated via the denoiser's output:

$$p_\theta(w_i \mid y_0^{-i}) = f_\theta(w_i \mid y_1) \tag{25}$$

8.3 Equivalence between RLHF and Kl minimization

Consider the unnormalized target distribution $\tilde{q}(\tau)$ given as:

$$\tilde{q}(\tau) = \pi_{\text{ref}}(\tau) \exp(r(\tau)/\beta), \tag{26}$$

which leads to the Boltzmann distribution (Jaynes, 1979):

$$q(\tau) = \frac{1}{Z} \pi_{\text{ref}}(\tau) \exp(r(\tau)/\beta), . \tag{27}$$

where $Z = \sum_\tau \pi_{\text{ref}}(\tau) \exp(\frac{r(\tau)}{\beta})$ is the normalization constant.

Expanding the KL term in J_β, we can rewrite the standard policy gradient objective J as:

$$\begin{aligned} J(\theta) &= \mathbb{E}_{\tau \sim \pi_\theta}[r(\tau)] - \beta \mathbb{E}_{\tau \sim \pi_\theta}[\log \pi_\theta(\tau) - \log \pi_{\text{ref}}(\tau)] && (28)\\ &= -\beta \mathbb{E}_{\tau \sim p_\theta}[\log \pi_\theta(\tau) - \log \pi_{\text{ref}}(\tau) - r(\tau)/\beta] && (29)\\ &= -\beta (D_{\text{KL}}(\pi_\theta \| q) + \log Z). && (30) \end{aligned}$$

Hence, maximizing J is equivalent to minimizing $D_{\text{KL}}(\pi_\theta \| q)$.

9 Additional Details

9.1 dLLM Details

Masked Diffusion Language Models are a class of discrete diffusion models that generate text by gradually denoising a sequence of tokens, starting from a fully masked state. Unlike autoregressive (AR) models that generate tokens sequentially, or standard BERT-style masked language models that perform single-step infilling, dlLMs iteratively refine predictions over multiple steps, allowing for more flexible and globally coherent generation.

Forward and Reverse Process The forward process in dLLM is a discrete noising process that gradually corrupts an input sequence x_0 (where x_0 is a sequence of one-hot token vectors) by replacing tokens with a special `[MASK]` token. Let x_t denote the sequence at timestep $t \in [0, 1]$, where $t = 0$ corresponds to the clean input and $t = 1$ corresponds to the fully masked state. The corruption is governed by a noise schedule α_t, which is strictly decreasing in t.

For each token x_t^i in the sequence at time t, the forward process is defined as:

$$q(x_t^i | x_0^i) = \begin{cases} \alpha_t, & \text{if } x_t^i = x_0^i \quad \text{(token unchanged)}, \\ 1 - \alpha_t, & \text{if } x_t^i = \texttt{[MASK]} \quad \text{(token is masked)}. \end{cases}$$

This can also be written as a categorical distribution:

$$q(x_t | x_0) = \text{Cat}\left(x_t; \alpha_t x_0 + (1 - \alpha_t)\texttt{[MASK]}\right).$$

Here, α_t controls the probability of a token being preserved . Common schedule choices include: linear: $\alpha_t = 1 - t$ and cosine: $\alpha_t = \cos(\frac{\pi}{2} t)$schedules. LLada (Nie et al., 2025) propose using the linear schedule.

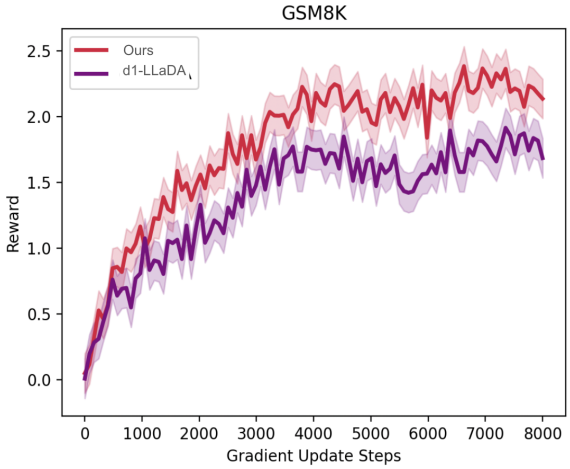

Figure 3: Training Curves for training Our models against d1

The reverse process learns to denoise x_t back to x_0. Unlike the forward process, which is fixed, the reverse process is parameterized by a neural network f_θ that predicts the original tokens given a masked sequence. The reverse transition $q(x_s|x_t)$ for $s < t$ is derived from Bayes' rule and has three cases:

1. **If $x_t^i \neq$ `[MASK]`:** The token is already unmasked and remains unchanged:

$$q(x_s^i|x_t^i) = \delta(x_s^i = x_t^i).$$

2. **If $x_t^i =$ `[MASK]` and $x_s^i =$ `[MASK]`:** The token stays masked:

$$q(x_s^i|x_t^i) = \frac{1-\alpha_s}{1-\alpha_t}.$$

3. **If $x_t^i =$ `[MASK]` and $x_s^i \neq$ `[MASK]`:** The token is unmasked, and the model predicts the original token:

$$q(x_s^i|x_t^i) = \frac{\alpha_s - \alpha_t}{1-\alpha_t} \cdot \pi_\theta(x_0^i|x_t).$$

The model π_θ is trained to predict x_0 given x_t, similar to BERT but conditioned on the noise level or in this case the masking level t.

Training Objective Normally an LLM is trained to optimize the likelihood of the text $\pi_\theta(x_0)$, however for dLLMs this is intractable. Instead , one optimizes the Evidence Lower Bound (ELBO) which serves as a lower bound for the true probability. Often instead of a continuous time input, the temporal space $[0, 1]$ is split into T discrete chunks. In that case the ELBO loss becomes:

$$\mathbb{E}_{t,x_0,x_t}\left[\frac{\partial_t \alpha_t}{1-\alpha_t}\sum_{i=1}^{L}\mathbb{1}[x_t^i = \texttt{[MASK]}]\cdot\log\pi_\theta(x_0^i|x_t)\right],$$

where the indicator function $\mathbb{1}[x_t^i = \texttt{[MASK]}]$ ensures only masked tokens contribute to the loss.

We trained on a lambda labs machine with 8 NVIDIA A100-80G GPU. We used AdamW optimizer with cosine learning rate annealing. The fine tuning was conducted with β parameters 0.9 and 0.99, and learning rate of 5e-6. For token perturbation in training we followd Zhao et al. (2025) and mask prompt tokens with p=0.15. We train for 10k steps and use validation set based early stopping for finding the best results. For all datasets, we deployed the split available from huggingface.

Model	HumanEval			MBPP		
	128	256	512	128	256	512
LLaDA-8B	21.3	32.3	32.9	40.1	39.7	41.2
+ d1	31.1	32.9	37.8	40.5	44.7	42.8
+ PKL	**32.7**	**36.2**	**38.6**	**41.9**	**47.4**	**45.4**
+ PKL-U	32.4	34.1	38.0	40.6	46.6	43.4
+ PKL-T	32.5	33.4	37.9	40.3	45.1	43.2

Table 2: Performance on HumanEval and MBPP at different sequence lengths.

9.2 Discusson on PKL-U and PKL-T and pseudo-divergence

PKL-U: Block mean-field approximation. Let $U \subseteq \{1, \dots, L\}$ denote a subset of token indices. PKL-U employs a one-step mean-field approximation in which all positions in U are unmasked simultaneously, yielding

$$\pi_\theta(y^U \mid y^{-U}, x) \;\approx\; \prod_{j \in U} \tilde{\pi}_\theta(y^j \mid y^{-U}, x), \tag{31}$$

where $\tilde{\pi}_\theta$ is computed from a single unmasking step. Thus, the block conditional $\pi_\theta(y^U \mid y^{-U}, x)$ is replaced by a mean-field approximation, resulting in a block pseudo-likelihood objective.

When $U = \{1, \dots, L\}$, PKL-U reduces exactly to the one-step fully mean-field update used in d1 / Diffu-GRPO (Zhao et al., 2025). Conversely, when $|U| = 1$ and U ranges over all positions, PKL-U recovers the full PKL objective. Therefore, PKL-U can be viewed as a computationally adjustable interpolation between the full PKL objective and the d1 (Zhao et al., 2025) update.

PKL-T: Trajectory-based approximation. PKL-T leverages the sampling order produced during generation. Let U_i denote the set of token positions that were unmasked prior to position i along the sampling trajectory. PKL-T approximates the one-token conditional as

$$\pi_\theta(y^i \mid y^{-i}, x) \;\approx\; \tilde{\pi}_\theta(y^i \mid y^{U_i}, x). \tag{32}$$

This approximation requires only a single masked-diffusion step, but restricts the attention mechanism to the positions in U_i. Substituting these trajectory-based conditionals into the pseudo-likelihood yields

$$\prod_{i=1}^{L} \pi_\theta(y^i \mid y^{U_i}, x), \tag{33}$$

which resembles an autoregressive factorization.

If the sampling order is replaced by a random permutation σ of $\{1, \dots, L\}$, this expression reduces to the random-order autoregressive decomposition used in the ELBO of Hoogeboom et al. (2021). In that formulation, conditionals of the form $\pi_\theta(y^{\sigma(i)} \mid y^{\sigma(<i)}, x)$ are evaluated for random permutations σ. Thus, PKL-T can be interpreted as a trajectory-aligned generalization of this estimator, recovering the Hoogeboom ELBO when all random orders are used.

10 Additional Experimental Results

In Table 2, we present results on the coding benchmarks HumanEval and MBPP benchmarks [4]. We see that PKL-based training consistently improves code generation performance over the LLaDA-8B baseline and the diffu-GRPO/d1 training. In particular, PKL achieves the strongest overall results, yielding the highest accuracy at all sequence lengths in both benchmarks. The PKL variants, PKL-U and PKL-T, also provide consistent improvements over the baseline and the d1 objective, though they underperform full PKL. Overall, these results indicate that optimizing conditional pseudo-KL objectives is an effective approach for improving masked diffusion language models.

[4] Results for baseline are reported from Zhao et al. (2025)

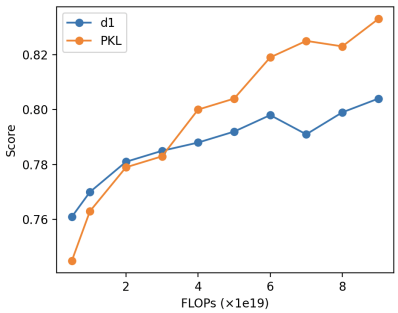

Figure 4: FLOP Adjusted performance on GSM

Compute Cost Naively PKL requires K times as much computation as 1 pass of d1/diffu-GRPO where K is the block size. However, PKL is summing up losses over token positions, and so by randomly choosing indices we can in one backward pass get an unbiased gradient estimate. We then use gradient accumulation and MC sampling to compute gradients. The same applies to PKL-U where we accumulate gradients over different random choices of U. Furthermore both PKL-U and PKL-T take for each step the same time as d1. The main reason for this is that most of the computation is similar to d1, with the additional differences having minor impact on gradient cost.

In Figure 4 we present a FLOP equalized performance for both d1 and for our approach on GSM. At very low flops, when PKL is still picking random indices, d1 outperforms. However as the budget increases and our model has updated gradients across enough token positions, we start doing better.

11 Additional Related Work

KL Regularization Commonly in RL the objective is to find a policy π that maximizes the expected cumulative reward $J(\pi) = \mathbb{E}_{\tau\sim\pi}[r(\tau)]$, where $r(\tau)$. However direct maximization is often undesired (especially in the context of LLMs), as the resulting models converge to narrow, high-reward outputs with low diversity (Choshen et al., 2019; Paulus et al., 2017). To alleviate this models are often trained with a regularization term (given by the KL divergence to a reference model). This approach inherently prevents distribution collapse. By maintaining diversity through KL regularization, the model retains its generative capabilities while learning to favor high-reward behaviors (Ziebart et al., 2008; Ziebart, 2010; Neu et al., 2017; Ouyang et al., 2022). The standard KL regularized return is defined as

$$J_\beta(\pi) = J(\pi) - \beta \cdot \mathbb{E}_{\tau\sim\pi}\left[\log \frac{\pi(\tau)}{\pi_{ref}(\tau)}\right] \tag{34}$$

where $\beta > 0$ is a regularization parameter that controls the strength of the penalty $D_{\text{KL}}(\pi\|\pi_{ref}) = \mathbb{E}_{\tau\sim\pi}\left[\log \frac{\pi(\tau)}{\pi_{ref}(\tau)}\right]$ which is the Kullback-Leibler divergence from π to π_{ref}. This is effectively equivalent to adding the log propensity term to the rewards r.

KL regularized RL KL regularized learning has its roots in maximum-entropy RL Ziebart et al. (2008); Neu et al. (2017), where a KL penalty ensures that learned policies remain close to a reference distribution. This framework has led to several influential algorithms, including Soft Q-Learning (SQL) (Haarnoja et al., 2017) and Soft Actor-Critic (SAC) (Haarnoja et al., 2018). The relation between entropy-regularized control and divergence-minimisation is known since the seminal work of Jaynes (1979). Bsed on the form of the optimal policy of such an entropy regularized optimization procedure (Ziebart et al., 2008), various direct alignment algorithms like DPO (Rafailov et al., 2023), IPO (Azar et al., 2024) etc. have been proposed. Khalifa et al. (2020); Korbak et al. (2022) proposed an alternative for fine-tuning language models, achieving results comparable to KL-regularized RL. Their GDC method minimizes $KL(q||p)$ rather than $KL(p||q)$, making it theoretically distinct from standard RL objectives (Korbak et al., 2022).