GIFT: GUIDED IMPORTANCE-AWARE FINE-TUNING FOR DIFFUSION LANGUAGE MODELS

Guowei Xu†, Wenxin Xu†, Jiawang Zhao, Kaisheng Ma*
Institute for Interdisciplinary Information Sciences, Tsinghua University
†Equal Contribution. *Corresponding Author: kaisheng@tsinghua.edu.cn

ABSTRACT

Diffusion models have recently shown strong potential in language modeling, offering faster generation compared to traditional autoregressive approaches. While the diffusion mechanism enables the model to reason over entire sequences, it also makes the generation process less predictable and often inconsistent. This highlights the importance of controlling key tokens that guide the direction of generation. To address this issue, we propose GIFT, an importance-aware fine-tuning method for diffusion language models, where tokens are assigned different importance weights based on their entropy. Derived from diffusion theory, GIFT delivers substantial gains: across diverse settings including different mainstream training datasets ranging from 1k to 10k in size, utilizing LoRA or full parameter fine-tuning, and training on base or instruct models with different architectures, GIFT consistently achieves superior overall performance compared to standard SFT on four widely used reasoning benchmarks (Sudoku, Countdown, GSM8K, and MATH-500).

1 INTRODUCTION

Recently, the landscape of large language modeling has been reshaped by the emergence of diffusion-based approaches (Nie et al., 2025; Ou et al., 2025; Yang et al., 2025), which offer a fundamentally different perspective on sequence generation compared to autoregressive (AR) methods Radford et al. (2018; 2019); Brown et al. (2020). Rather than relying on strictly sequential decoding, diffusion large language models (dLLMs) employ iterative refinement procedures that enable efficient parallel generation and open new possibilities for scaling language models.

Currently, similar to AR models, the training paradigm of dLLMs is generally divided into three stages: pre-training, instruction tuning, and reinforcement learning (RL). Among them, the pre-training and instruction tuning stages adopt an SFT loss specifically derived for dLLMs (Nie et al., 2025), while the RL stage employs a loss estimated on the basis of the PPO algorithm (Schulman et al., 2017) and tailored to the characteristics of dLLMs (Zhu et al., 2025).

Specifically, when designing the SFT objective, Nie et al., 2025 model the generation process of dLLMs as a continuous-time diffusion process, where the diffusion timestep t ranges within $[0, 1]$. Here, $t = 0$ denotes a completely original, non-masked sequence, and $t = 1$ denotes a fully masked sequence. As illustrated in Figure 1(a), let the original response of the model be x_0. At a given timestep t, each token is masked with probability t, resulting in a partially masked response x_t. The training objective of dLLMs is then to enable the model to reconstruct the masked parts of x_0 given the partially masked input x_t. Based on this formulation, the loss is defined as follows:

$$L = -\sum_i \mathbb{E}_t \left[\mathbf{1}[x_t^i = \mathbf{M}] \, \frac{1}{t} \log p\big(x_0^i \mid x_t\big) \right] \tag{1}$$

This design of the SFT loss is intuitive and aligns well with the underlying rationale of diffusion theory. However, it presents several critical limitations. Among them, one of the most prominent issues is that the loss implicitly assumes a uniform masking rate across all tokens, thereby treating each token as equally important throughout the diffusion process. This assumption overlooks the

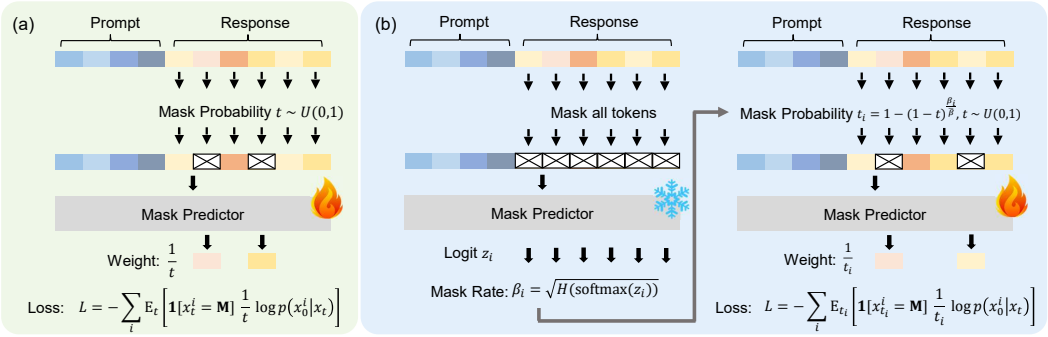


Figure 1: (a) The SFT pipeline. A timestep t is uniformly sampled from $[0, 1]$, and each token is masked independently with probability t. The training objective is to predict the masked tokens accurately based on the unmasked ones. (b) The GIFT pipeline. In each training step, we perform two forward passes. During the first forward pass, we mask the entire answer and estimate the masking rate β_i for each token by computing its predictive entropy. In the second forward pass, the i-th token is masked with probability t_i (computed from β_i), and its training weight is set to $\frac{1}{t_i}$. Tokens with higher entropy are more likely to be masked and thus receive stronger training signals.

inherent heterogeneity in token significance: tokens that play a central role in planning and reasoning should arguably receive greater emphasis during training than other tokens. While the Dream series (Ye et al., 2025) employs Context-Adaptive Token-Level Noise Rescheduling to account for varying token weights, its loss formulation remains heuristic and lacks a theoretical foundation, with its effectiveness proved to be limited in our experiments.

To address this discrepancy, we introduce GIFT, a guided importance-aware finetuning mechanism based on predictive uncertainty. Specifically, we find that the entropy of the model's predictive distribution over tokens serves as a reliable metric for token importance at the current training stage. Tokens associated with higher entropy correspond to positions where the model exhibits greater uncertainty in generation. By prioritizing these high-entropy tokens during training via an adaptive importance weighting strategy applied to the SFT loss, we can concentrate computation and importance on tokens where the model is currently more uncertain, thereby improving the overall training effectiveness.

Based on the fundamental principles of diffusion models, we derive an importance-weighted SFT loss formulation that is theoretically consistent with diffusion dynamics. Empirically, we evaluate this approach on four widely used benchmarks, Sudoku (Arel, 2025), Countdown (Pan et al., 2025), GSM8K (Cobbe et al., 2021), and MATH500 (Lightman et al., 2024), by comparing models fine-tuned with the standard SFT algorithm against those fine-tuned with GIFT under identical training data. Specifically, our method is trained using LoRA (Hu et al., 2022) on s1K, s1K-1.1 (Muennighoff et al., 2025), and 3k samples drawn from the Mixture-of-Thoughts dataset of open-r1 (HuggingFace, 2025), and using full parameter fine-tuning on 10k samples drawn from Tulu3 (Lambert et al., 2025). Across these diverse training settings, our approach consistently achieves superior overall performance compared to standard SFT.

In addition, ablation studies further confirm that both the theoretically derived importance-weighted SFT loss and the entropy-based importance-weighting scheme are indispensable for achieving the observed performance improvements. Time efficiency analysis shows that GIFT incurs only minimal overhead during LoRA fine-tuning, and imposes no computational overhead during full-parameter fine-tuning, even achieving faster speeds than standard SFT. Finally, experiments on few-shot learning and alternative model architectures further verify the generalizability and superiority of GIFT.

In summary, our core contributions are as follows:

- We derive a theoretically grounded formulation of the importance-weighted SFT loss from the perspective of diffusion processes, ensuring consistency with the underlying principles of diffusion-based generation.

- We identify token-level predictive entropy as a reasonable and effective metric for capturing token importance, and we demonstrate both theoretically and empirically that tokens with higher entropy should be emphasized during training to enhance the effectiveness and stability of the training process.
- Building upon these insights, we propose GIFT, an importance-weighted finetuning algorithm that consistently achieves superior overall performance compared to the standard SFT approach across multiple training sets and benchmarks. Ablation studies further validate the necessity and effectiveness of each design choice in GIFT. Time efficiency analysis demonstrates that GIFT incurs minimal training time overhead compared to SFT.

2 Preliminaries: Continuous-Time Discrete Diffusion Model

Following Ou et al., 2025, we model diffusion language models as a *Continuous-Time Discrete Diffusion Model* (Sun et al., 2023). Intuitively, in each infinitesimal time interval Δt, every token has a certain probability of being masked.

Formally, this process can be described as a time-dependent Markov chain, whose transition dynamics are governed by a matrix Q_t. For a transition from state x to y within $(t, t+\Delta t)$, we have

$$p_{t+\Delta t|t}(y|x) = \begin{cases} Q_t(x,y)\Delta t + o(\Delta t), & \text{if } x \neq y \\ 1 + Q_t(x,x)\Delta t + o(\Delta t), & \text{if } x = y \end{cases} \tag{2}$$

where $Q_t(x,y) \geq 0$ for $x \neq y$ and $Q_t(x,x) < 0$. Thus, $Q_t(x,y)$ specifies the instantaneous transition rate from state x to y.

In practice, it is common to assume a *time–factorized form* (Campbell et al., 2022), namely

$$Q_t = f(t)Q, \tag{3}$$

where Q is a constant matrix and $f(t)$ is a scalar function controlling the evolution speed.

Let $P_{t|s}(x,y) = p_{t|s}(y|x)$ denote the transition probability matrix from time s to t. Then, $P_{t|s}$ satisfies the Kolmogorov's forward equation (Anderson, 2012):

$$\frac{d}{dt}P_{t|s} = f(t)P_{t|s}Q. \tag{4}$$

The solution is given by

$$P_{t|s} = \exp\left(Q\int_s^t f(u)\,du\right), \tag{5}$$

where $\exp$ denotes the matrix exponential.

For notational convenience, we define

$$\bar{f}(x) = \int_0^x f(t)\,dt. \tag{6}$$

With this shorthand, the solution can be equivalently written as

$$P_{t|s} = \exp\big((\bar{f}(t) - \bar{f}(s))Q\big). \tag{7}$$

Intuitively, s corresponds to the sequence before masking at the current step, t corresponds to the sequence after masking at the current step, and Q represents the masking rate of each token. In prior work, it is commonly assumed that the Q matrix takes the following form: (Ou et al., 2025; Nie et al., 2025)

$$Q = \begin{bmatrix} -1 & 0 & \cdots & 0 & 1 \\ 0 & -1 & \cdots & 0 & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & -1 & 1 \\ 0 & 0 & \cdots & 0 & 0 \end{bmatrix} \tag{8}$$

Under this assumption, all tokens gradually diffuse into an absorbing state [M], that is, they become masked.

3 METHOD

3.1 OVERVIEW

We propose GIFT (Guided Importance-Aware Fine-Tuning), an importance-weighted finetuning method designed to assign higher importance weights to tokens where the current model exhibits greater uncertainty. As we will explain later, in practice, we quantify token importance using the square root of token entropy.

As shown in Figure 1(a), the original SFT approach masks all tokens with the same probability t, and each token contributes equally to the loss function with a weight of $\frac{1}{t}$. In contrast, Figure 1(b) illustrates the GIFT pipeline. In each training step, we perform two forward passes. In the first pass, the entire answer is masked, and the masking rate β_i for each token is estimated based on its predictive entropy. In the second pass, each token is independently masked with probability t_i (computed from β_i), and its training weight is assigned as $\frac{1}{t_i}$. This design ensures that tokens with higher entropy are more likely to be masked and trained more frequently, thereby improving the effectiveness and stability of the training process.

The derivation of the loss function is presented in Section 3.2, while the details and motivation of the importance weighting scheme are provided in Section 3.3 and Appendix D. A comprehensive description of the implementation of GIFT can be found in Appendix B.

3.2 IMPORTANCE-AWARE LOSS DERIVATION

As noted earlier, prior work typically assumes that all tokens are masked at the same rate, leading to a Q matrix that takes the form of an absorbing matrix composed of 1 and -1. In contrast, we assign each token a distinct masking rate β, from which we derive an importance-weighted finetuning formulation. Importantly, the resulting SFT loss remains consistent with the fundamental properties of diffusion, since the Q matrix is still well-defined under this construction.

Namely, we now set Q as:

$$Q = \begin{bmatrix} -\beta_1 & 0 & \cdots & 0 & \beta_1 \\ 0 & -\beta_2 & \cdots & 0 & \beta_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & -\beta_{n-1} & \beta_{n-1} \\ 0 & 0 & \cdots & 0 & 0 \end{bmatrix} \tag{9}$$

Based on the newly defined form of the Q matrix, and following the derivation in Ou et al., 2025, we obtain the modified formulation of the SFT loss. The detailed derivation is provided in Appendix A.

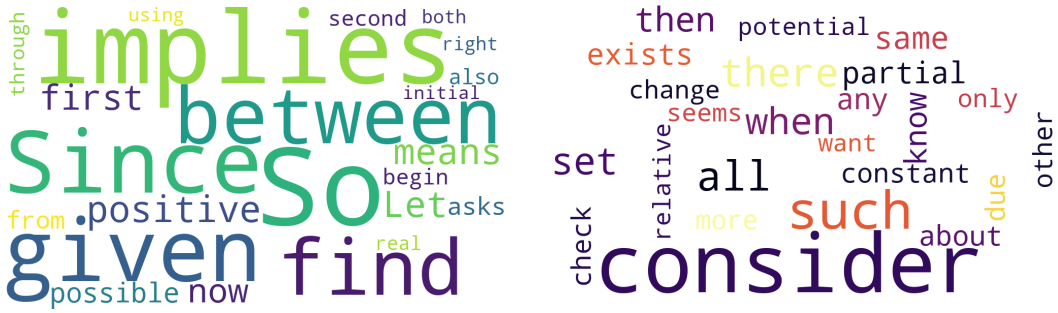
Theorem 1. *Assuming the Q matrix takes the form given in Equation 9, let the initial sequence be x_0 and the sequence at time t be x_t. Under this setting, the i-th token is masked with probability $t_i = 1-(1-t)^{\frac{\beta_{x^i}}{\beta_{ref}}}$, where β_{x^i} denotes the masking rate of the i-th token, and β_{ref} is a specified reference masking rate. Moreover, the importance-weighted supervised finetuning loss can be derived as follows:*

$$L = -\sum_i \mathbb{E}_{t_i}\left[\mathbf{1}\left[x_{t_i}^i = \mathbf{M}\right]\frac{1}{t_i}\log p\big(x_0^i \mid x_t\big)\right] \tag{10}$$

Intuitively, under the importance-weighted supervised finetuning loss, a token x_i with a larger masking rate β_{x^i} corresponds to a larger t_i, making it more likely to be masked and subsequently learned. Here, β_{ref} can be any reference value. Empirically, we set β_{ref} to the mean of all β values to enhance numerical stability.

3.3 MOTIVATION OF USING ENTROPY AS THE MASKING RATE METRIC

Having established the form of the importance-weighted supervised finetuning loss, we next investigate how to assign masking rate to tokens. First, we require that the chosen metric does not



Frequent tokens with high entropy

Frequent tokens with low entropy

Figure 2: Visualization of high-frequency tokens with different entropy levels. Tokens with higher entropy correspond to those where the model exhibits greater uncertainty, and are learned more frequently during training.

introduce excessive computational overhead. In particular, it is infeasible to compute such a metric for every token individually, and we can only afford $O(1)$ additional forward passes in total. For this reason, we consider masking all the answers and then examining the distribution of the model's output logits z_i to be a practical choice. Specifically, given a problem x and its answer y, we mask the entire answer and compute

$$z_i = \text{model}(\cdot \mid x, [M]).\text{logits}. \tag{11}$$

We then determine the masking rate β_i of the i-th token. Based on our analysis, the entropy of the distribution of z_i serves as an effective metric. First, drawing inspiration from theoretical insights, we demonstrate that using entropy as an importance weight may help reduce the variance of the loss estimation. Specifically, prior work has demonstrated that importance sampling achieves minimum variance when the sampling proposal is proportional to the L2 norm of the gradient. Through reasonable approximation, we found that token-wise entropy is approximately proportional to the L2 norm of the gradient, which motivates our choice of this metric. We provide a detailed discussion in Appendix D. Furthermore, as shown in Figure 2, we empirically visualize the entropy distribution among the 100 most frequent tokens during training, contrasting tokens with higher versus lower entropy. It can be observed that tokens with higher entropy tend to carry richer information and play a more critical role in the model's decision-making. For example, words such as "first", "second", "since", and "so" exhibit higher entropy, often signaling logical or sequential structures, whereas words like "all" and "such" show lower entropy and function primarily as auxiliary words, contributing less to semantic content. Our findings align with recent observations in LLMs (Wang et al., 2025), which note that a minority of high-entropy tokens typically serve to bridge logical connections between consecutive reasoning steps, while the majority of low-entropy tokens primarily complete current phrases or words.

Since we empirically find that using the raw entropy values may destabilize training (Section 4.3.3), we use the square root of entropy, which yields more stable optimization in practice:

$$\beta_i = \sqrt{H(\text{softmax}(z_i))}\,. \tag{12}$$

4 Experiment

In this section, we first describe the experimental setup, followed by a comparison of the performance of GIFT and the standard SFT algorithm across multiple training datasets and benchmarks. We then conduct ablation studies to validate the necessity of each design component in GIFT, and provide a runtime comparison between GIFT and SFT under identical settings. Finally, we provide a comparison between GIFT and SFT under few-shot learning settings, as well as the generalizability of GIFT across dLLM model architectures.

Model	Sudoku *0-shot*	Countdown *0-shot*	GSM8K *0-shot*	MATH500 *0-shot*	Average
Base Instruct Model					
LLaDA-8B-Instruct	5.6 ± 0.2	17.0 ± 1.0	76.8 ± 0.3	32.5 ± 0.3	33.0 ± 0.3
s1K Dataset, LoRA Tuning					
Instruct Model + SFT	4.2 ± 0.4	21.1 ± 1.4	$\mathbf{78.5 \pm 0.2}$	32.6 ± 0.9	34.1 ± 0.4
Instruct Model + GIFT	$\mathbf{6.0 \pm 0.9}$	$\mathbf{28.1 \pm 1.9}$	77.3 ± 0.3	$\mathbf{33.7 \pm 0.3}$	$\mathbf{36.3 \pm 0.5}$
s1K-1.1 Dataset, LoRA Tuning					
Instruct Model + SFT	4.9 ± 0.3	20.7 ± 0.4	$\mathbf{77.7 \pm 0.3}$	31.2 ± 0.8	33.6 ± 0.2
Instruct Model + GIFT	$\mathbf{5.9 \pm 0.4}$	$\mathbf{21.8 \pm 0.8}$	77.6 ± 0.3	$\mathbf{32.0 \pm 0.9}$	$\mathbf{34.3 \pm 0.3}$
openr1-3k Dataset, LoRA Tuning					
Instruct Model + SFT	5.9 ± 0.4	17.3 ± 0.8	77.1 ± 0.2	31.6 ± 0.6	33.0 ± 0.2
Instruct Model + GIFT	$\mathbf{6.8 \pm 0.3}$	$\mathbf{18.8 \pm 0.7}$	$\mathbf{77.7 \pm 0.3}$	$\mathbf{32.8 \pm 0.8}$	$\mathbf{34.0 \pm 0.2}$
Tulu3-10k Dataset, Full Paramter Tuning					
Base Model + SFT	$\mathbf{5.1 \pm 0.3}$	18.2 ± 0.9	$\mathbf{66.3 \pm 0.2}$	23.6 ± 0.6	28.3 ± 0.3
Base Model + GIFT	4.9 ± 0.4	$\mathbf{21.3 \pm 1.1}$	65.8 ± 0.3	$\mathbf{24.6 \pm 0.8}$	$\mathbf{29.2 \pm 0.3}$

Table 1: Comparison of the performance of GIFT and SFT across different training datasets and benchmarks. The table shows that GIFT achieves substantial percentage improvements over SFT.

4.1 Experimental Setup

We adopt LLaDA-8B-Base and LLaDA-8B-Instruct (Nie et al., 2025) as base models and apply either LoRA fine-tuning (Hu et al., 2022) or full parameter fine-tuning. For the training datasets, we use s1K, s1K-1.1, openr1-3k, and Tulu3-10k to evaluate the performance of GIFT and SFT. Among them, s1K and s1K-1.1 are reasoning datasets proposed by Muennighoff et al., 2025, while openr1-3k is constructed by sampling 3k examples from the Mixture-of-Thoughts dataset of open-r1 (HuggingFace, 2025), with sequence lengths less than 2.5k under the open-r1 encoder. Tulu3-10k consists of 10k samples drawn from Tulu3 (Lambert et al., 2025), following the setting of Zhou et al., 2025. To verify that our algorithm is framework-agnostic, we employ the `wd1` (Tang et al., 2025) framework for LoRA training, and the `dllm` (Zhou et al., 2025) framework for full-parameter fine-tuning. All experiments are conducted across three independent runs, with results reported as mean value and uncertainty.

The evaluation is conducted on the following benchmarks:

- **Sudoku** (Arel, 2025): A symbolic reasoning benchmark requiring models to solve Sudoku puzzles through step-by-step logical deduction.
- **Countdown** (Pan et al., 2025): A numerical reasoning dataset where models must combine given numbers using arithmetic operations to reach a target value.
- **GSM8K** (Cobbe et al., 2021): A widely used benchmark of grade-school math word problems designed to evaluate arithmetic and reasoning abilities.
- **MATH500** (Lightman et al., 2024): A subset of the MATH dataset containing 500 challenging competition-level math problems that test advanced mathematical reasoning.

The detailed training and evaluation hyperparameters and settings are provided in Appendix C.

4.2 Main Results

As shown in Table 1, we compare the performance of GIFT and the standard SFT method across different training datasets and benchmarks. Due to computational constraints, each of our experiments is the result of a single run. Notably, GIFT consistently achieves higher average accuracy than SFT, demonstrating that our method achieves effective training by leveraging token entropy. Experiments

Model	Sudoku *0-shot*	Countdown *0-shot*	GSM8K *0-shot*	MATH500 *0-shot*	Average
Base Model					
LLaDA-8B-Instruct	5.5	16.0	76.7	32.4	32.6
s1K Dataset					
+ SFT	4.6 (-0.9)	23.8 (+7.8)	78.8 (+2.1)	32.6 (+0.2)	34.9 (+2.3)
+ Weighted SFT (- log p)	3.0 (-2.5)	16.4 (+0.4)	75.4 (-1.3)	29.2 (-3.2)	31.0 (-1.6)
+ GIFT	7.8 (+2.3)	24.6 (+8.6)	78.0 (+1.3)	33.0 (+0.6)	35.8 (+3.2)

Table 2: Comparison between entropy-based importance weighting and NLL-based importance weighting. As shown in the table, the SFT algorithm with NLL-based importance weighting performs significantly worse than the entropy-based variant, and on many tasks even underperforms the original model.

across datasets of varying sizes, different training frameworks, multiple benchmarks, and on both base and instruct models demonstrate that the advantage of GIFT over SFT is universal rather than case-specific, indicating the strong generalizability of our method.

4.3 Ablation Study

In this subsection, we conduct ablation studies to validate the design choices of GIFT, focusing on three aspects: the necessity of using entropy as the metric, the validity of the theoretically derived loss function, and the empirical necessity of applying the square root of entropy during training.

4.3.1 Necessity of Using Entropy as the Metric

To demonstrate the necessity of entropy-based importance weighting, we compared our method with a baseline using the negative log-likelihood (NLL). Unlike entropy-based importance weighting, NLL emphasizes tokens that appear less frequently rather than those that are difficult to predict or carry more information. Assuming the logits are z_i, the masking rate β_i for the i-th token under the NLL setting is:

$$\beta_i = -\log\big(\text{softmax}(z_i)^{x_0^i}\big) \tag{13}$$

As shown in Table 2, the SFT method weighted by NLL performs poorly, in some cases even worse than the original base model. This underscores the importance of focusing on high-entropy, informative tokens for achieving effective fine-tuning performance.

4.3.2 Validity of the Theoretically Derived Loss Function

Our loss function is derived from the continuous-time discrete diffusion model combined with the Denoising Score Entropy (DSE) loss. As a result, our model preserves the inherent characteristics of the diffusion process more effectively than methods that simply add a weight to the loss function:

$$L = -\sum_i \mathbb{E}_t \left[\mathbf{1}\big[x_t^i = \mathbf{M}\big] \, \frac{w_i}{t} \log p\left(x_0^i \mid x_t\right) \right] \tag{14}$$

To illustrate this, we compare our approach with two weighted SFT methods by applying a simple token-wise weighting, namely using our square-root entropy as the weight, and using the Dream (Ye et al., 2025) method. Specifically, when using our square-root entropy, the weight of the i-th token is:

$$w_i = \sqrt{H(\text{softmax}(z_i))} \tag{15}$$

When using the weighting method from Dream, the weight of the i-th token is:

Model	Sudoku *0-shot*	Countdown *0-shot*	GSM8K *0-shot*	MATH500 *0-shot*	Average
Base Model					
LLaDA-8B-Instruct	5.5	16.0	76.7	32.4	32.6
s1K Dataset					
+ SFT	4.6 (-0.9)	23.8 (+7.8)	78.8 (+2.1)	32.6 (+0.2)	34.9 (+2.3)
+ Dream	3.3 (-2.2)	21.1 (+5.1)	77.0 (+0.3)	30.0 (-2.4)	32.8 (+0.2)
+ GIFT (SW)	5.3 (-0.2)	23.8 (+7.8)	75.9 (-0.8)	31.6 (-0.8)	34.2 (+1.6)
+ GIFT	7.8 (+2.3)	24.6 (+8.6)	78.0 (+1.3)	33.0 (+0.6)	35.8 (+3.2)

Table 3: Comparison between GIFT, GIFT (SW) (simple token-wise weighting using our square-root entropy as the weight), and Dream. Dream yields negligible improvements. GIFT (SW) provides some gains, but still performs worse than SFT, whereas GIFT significantly outperforms SFT.

$$w_i = \frac{1}{2} \sum_{j=1}^{N} \mathbf{1}\left[x_t^j \neq \mathbf{M}\right] \, \mathrm{Geo}(p, |j-i|-1) \tag{16}$$

Here, Geo denotes the geometric distribution, and p controls the sharpness of the distribution. Since the original work did not provide the corresponding parameter setting, we set it to a moderate value of 0.3.

Table 3 presents a comparison between GIFT, GIFT (SW) (i.e., simple token-wise weighting using our square-root entropy as the weight), and Dream. Dream yields only minimal improvements and, in some cases, performs worse than standard SFT. GIFT (SW) provides some gains but still performs worse than SFT, whereas GIFT significantly outperforms SFT. This demonstrates that the diffusion loss derived from theoretical principles possesses superior properties.

4.3.3 Empirical Necessity of Applying the Square Root of Entropy During Training

Below, we explain why we choose the square root of entropy rather than entropy as the metric. As shown in Table 4, we observed that directly using token weights based on raw entropy leads to large variations across tokens, which can significantly reduce training stability and even cause oscillations. To address this issue, we implement GIFT using the square root of entropy as the weighting factor. This modification effectively reduces the scale of the gradient norms, resulting in more stable training and improved overall convergence behavior.

Gradient Norm	Entropy	$\sqrt{\text{Entropy}}$
Max	646.1	46.2
Median	0.47	0.43
Average	2.02	1.03

Table 4: Comparison of gradient norms during training when using entropy versus square root entropy. Using entropy sometimes leads to extremely large gradient norms, resulting in instability, whereas using square root entropy effectively alleviates this issue.

4.4 Time Efficiency Analysis

In the time efficiency analysis, there are two counteracting factors. First, GIFT requires an additional forward pass compared to SFT, resulting in a slightly higher computational overhead. However, it is important to note that the cost of a single gradient-free forward pass is significantly lower than that of a backward pass; thus, this factor only causes a marginal decrease in speed. Second, the expected number of masked tokens per iteration may differ between GIFT and SFT, which also impacts the overall training speed.

Dataset	Framework	SFT	GIFT
s1K	wd1	46 min	57 min
Tulu3	dllm	28 min	26 min

Table 5: Time efficiency comparison between SFT and GIFT on different tasks and frameworks.

As shown in Table 5, we compare the training speed of GIFT and SFT under two distinct settings: (1) LoRA fine-tuning on the `s1K` dataset for 20 epochs using the `wd1` training framework on 4 H100 GPUs; and (2) full fine-tuning on the `Tulu3-10k` dataset for 5 epochs using the `dllm` training framework on 8 H100 GPUs. These scenarios represent representative results selected from our main experiments. The results indicate that while GIFT is merely slightly slower than SFT during LoRA fine-tuning, it is marginally faster than SFT during full fine-tuning. We attribute this to the fact that the additional forward pass incurs only a minor overhead. Conversely, since GIFT prioritizes more significant tokens, the expected total number of masked tokens may be lower than that of SFT, thereby reducing the computational load during backpropagation. This reduction in computation has a more pronounced impact during full fine-tuning compared to LoRA, leading to the observation that our method achieves faster speeds than SFT in the full fine-tuning setting.

4.5 Evaluation with Few-Shot Learning

To further strengthen our empirical results and address the assessment of in-context learning capabilities, we include performance metrics for 5-shot evaluation. As shown in Table 6, GIFT consistently demonstrates robust performance in the few-shot setting across different datasets.

Model	Sudoku *5-shot*	Countdown *5-shot*	GSM8K *5-shot*	MATH500 *5-shot*	Average
s1K Dataset, LoRA Tuning					
Instruct Model + SFT	6.0 ± 0.2	23.2 ± 1.0	$\mathbf{78.9 \pm 0.3}$	31.8 ± 0.6	35.0 ± 0.3
Instruct Model + GIFT	$\mathbf{7.9 \pm 0.5}$	$\mathbf{27.5 \pm 1.6}$	76.7 ± 0.2	$\mathbf{33.0 \pm 0.7}$	$\mathbf{36.3 \pm 0.5}$
s1.1K Dataset, LoRA Tuning					
Instruct Model + SFT	$\mathbf{23.3 \pm 0.3}$	21.7 ± 1.2	$\mathbf{78.7 \pm 0.1}$	30.0 ± 0.5	38.4 ± 0.3
Instruct Model + GIFT	20.1 ± 0.8	$\mathbf{26.0 \pm 1.2}$	78.2 ± 0.1	$\mathbf{31.4 \pm 0.6}$	$\mathbf{38.9 \pm 0.4}$

Table 6: 5-shot evaluation results on Sudoku, Countdown, GSM8K, and MATH500 benchmarks. GIFT consistently shows robust performance in few-shot scenarios.

4.6 Generalizability across Architectures

To verify that our method is applicable to dLLMs of various architectures and to establish generalizability beyond the LLaDA-8B backbone, we perform full-parameter fine-tuning on Dream-v0-7B-Base using the dLLM framework and the Tulu3-10k dataset.

Model	Sudoku *0-shot*	Countdown *0-shot*	GSM8K *0-shot*	MATH500 *0-shot*	Average
Tulu3-10k Dataset, Full Paramter Tuning					
Base Model + SFT	$\mathbf{3.8 \pm 0.4}$	12.3 ± 0.7	50.3 ± 0.3	17.6 ± 0.6	21.0 ± 0.3
Base Model + GIFT	2.1 ± 0.3	$\mathbf{14.1 \pm 0.8}$	$\mathbf{55.2 \pm 0.2}$	$\mathbf{22.2 \pm 0.5}$	$\mathbf{23.4 \pm 0.3}$

Table 7: Full-parameter fine-tuning results on Dream-v0-7B-Base.

In Table 7, the results further demonstrate the superiority of the GIFT algorithm over standard SFT and verify the generalizability of our approach across different model architectures.

5 Related Work

5.1 Diffusion Large Language Models

Diffusion models, initially developed for continuous data, have recently been adapted to discrete sequences, including natural language. Early work such as Structured Denoising Diffusion Models (D3PMs) (Austin et al., 2021) extended discrete diffusion to text, showing competitive performance in character-level generation and large vocabulary language modeling. DiffusionBERT (He et al., 2023) combined masked language modeling with diffusion, improving generative quality over conventional masked LMs. Later, masked diffusion frameworks (Sahoo et al., 2024; Shi et al., 2024) introduced simplified and generalized training objectives that reduce complexity while maintaining performance, and support state-dependent masking strategies. Diffusion Large Language Models (dLLMs) including LLaDA (Nie et al., 2025) and Dream (Ye et al., 2025) demonstrated that diffusion-based models can scale to billions of parameters, achieving strong context learning and instruction-following capabilities. Techniques such as variance-reduced preference optimization (Zhu et al., 2025) and reinforcement of lateral thought chains (Huang et al., 2025) further enhance generative stability and reasoning ability, while efficient perplexity bounds and ratio matching (Haxholli et al., 2025) improve sampling efficiency. More recently, works such as d1 (Zhao et al., 2025) and wd1 (Tang et al., 2025) have explored the possibility of applying reinforcement learning on top of existing dLLMs. LLaDA 2.0 (Bie et al., 2025) pushed the boundaries of model capacity by scaling diffusion language models to 100B parameters, offering a comprehensive analysis of scaling laws.

5.2 Weighted Methods in Language Model Learning

A complementary line of research focuses on improving large language models through weighted supervised fine-tuning (SFT) and reinforcement learning. Weighted SFT techniques aim to prioritize informative tokens. DFT (Wu et al., 2025) addresses the limited generalization of SFT by dynamically rescaling token-level objectives to stabilize gradient updates. Rho-1 (Lin et al., 2024) challenges the conventional uniform next-token prediction objective by selectively training on useful tokens identified via a reference model. Similarly, Luo et al., 2023 propose a re-weighting strategy for active learning in NER, assigning dynamic smoothed weights to tokens to boost performance. In reinforcement learning, applying weighting to improve learning effectiveness is also a common strategy. Cui et al., 2025 investigate the entropy dynamics of reinforcement learning for reasoning LLMs, and propose entropy-based weighting methods to prevent entropy collapse and encourage exploration. Wang et al., 2025 highlight the critical role of high-entropy minority tokens in RLVR, demonstrating that restricting updates to these tokens significantly enhances reasoning capabilities. Together, these approaches demonstrate that token-level weighting is an important and effective way to improve the optimization of language models.

Despite these advancements, most weighting methods explored for classical LLMs cannot be directly transferred to the dLLM setting. For instance, methods like DFT and Rho-1 rely on the specific properties of auto-regressive models. While the work by Wang et al., 2025 is related to ours in emphasizing the importance of high-entropy tokens, they merely apply a hard threshold to select tokens for gradient computation. In contrast, our GIFT weighting method is tailored to the unique characteristics of dLLMs. It determines importance weights for all tokens through a single forward pass, making our approach fundamentally distinct from these previous weighting mechanisms.

6 Conclusion

We introduced GIFT, a theoretically grounded importance-aware fine-tuning algorithm for diffusion language models. By leveraging token-level entropy as a measure of importance, GIFT prioritizes high-uncertainty tokens and preserves diffusion process properties. Experiments across varying datasets, different training frameworks, multiple benchmarks, and on both base and instruct models demonstrate the universal advantage of the GIFT method over SFT. These results highlight entropy-based importance weighting as a simple yet powerful strategy for enhancing training effectiveness and stability in diffusion language models.

ACKNOWLEDGMENTS

This work was supported in part by the Natural Science Foundation of China. (No. 625B1021)

REFERENCES

Guillaume Alain, Alex Lamb, Chinnadhurai Sankar, Aaron Courville, and Yoshua Bengio. Variance reduction in sgd by distributed importance sampling, 2016. URL https://arxiv.org/abs/1511.06481.

William J Anderson. *Continuous-time Markov chains: An applications-oriented approach*. Springer Science & Business Media, 2012.

Arel. Arel's sudoku generator. https://www.ocf.berkeley.edu/~arel/sudoku/main.html, 2025. Accessed: 2025-04-08.

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

Tiwei Bie, Maosong Cao, Kun Chen, Lun Du, Mingliang Gong, Zhuochen Gong, Yanmei Gu, Jiaqi Hu, Zenan Huang, Zhenzhong Lan, Chengxi Li, Chongxuan Li, Jianguo Li, Zehuan Li, Huabin Liu, Lin Liu, Guoshan Lu, Xiaocheng Lu, Yuxin Ma, Jianfeng Tan, Lanning Wei, Ji-Rong Wen, Yipeng Xing, Xiaolu Zhang, Junbo Zhao, Da Zheng, Jun Zhou, Junlin Zhou, Zhanchao Zhou, Liwang Zhu, and Yihong Zhuang. Llada2.0: Scaling up diffusion language models to 100b, 2025. URL https://arxiv.org/abs/2512.15745.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020. URL https://arxiv.org/abs/2005.14165.

Andrew Campbell, Joe Benton, Valentin De Bortoli, Tom Rainforth, George Deligiannidis, and Arnaud Doucet. A continuous time framework for discrete denoising models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=DmT862YAieY.

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.

Ganqu Cui, Yuchen Zhang, Jiacheng Chen, Lifan Yuan, Zhi Wang, Yuxin Zuo, Haozhan Li, Yuchen Fan, Huayu Chen, Weize Chen, Zhiyuan Liu, Hao Peng, Lei Bai, Wanli Ouyang, Yu Cheng, Bowen Zhou, and Ning Ding. The entropy mechanism of reinforcement learning for reasoning language models, 2025. URL https://arxiv.org/abs/2505.22617.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Etrit Haxholli, Yeti Z. Gurbuz, Oğul Can, and Eli Waxman. Efficient perplexity bound and ratio matching in discrete diffusion language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=Mri9WIfxSm.

Zhengfu He, Tianxiang Sun, Qiong Tang, Kuanning Wang, Xuanjing Huang, and Xipeng Qiu. Diffusionbert: Improving generative masked language models with diffusion models. In *ACL (1)*, pp. 4521–4534, 2023. URL https://doi.org/10.18653/v1/2023.acl-long.248.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Zemin Huang, Zhiyang Chen, Zijun Wang, Tiancheng Li, and Guo-Jun Qi. Reinforcing the diffusion chain of lateral thought with diffusion language models. *CoRR*, abs/2505.10446, May 2025. URL https://doi.org/10.48550/arXiv.2505.10446.

Open R1 HuggingFace. Mixture-of-thoughts. https://huggingface.co/datasets/open-r1/Mixture-of-Thoughts, 2025.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tulu 3: Pushing frontiers in open language model post-training, 2025. URL https://arxiv.org/abs/2411.15124.

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.

Zhenghao Lin, Zhibin Gou, Yeyun Gong, Xiao Liu, Yelong Shen, Ruochen Xu, Chen Lin, Yujiu Yang, Jian Jiao, Nan Duan, and Weizhu Chen. Rho-1: Not all tokens are what you need. *CoRR*, abs/2404.07965, 2024. URL https://doi.org/10.48550/arXiv.2404.07965.

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution, 2024. URL https://arxiv.org/abs/2310.16834.

Haocheng Luo, Wei Tan, Ngoc Dang Nguyen, and Lan Du. Re-weighting tokens: A simple and effective active learning strategy for named entity recognition. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023. URL https://openreview.net/forum?id=CihCvXPiEG.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candes, and Tatsunori Hashimoto. s1: Simple test-time scaling. In *Workshop on Reasoning and Planning for Large Language Models*, 2025. URL https://openreview.net/forum?id=LdH0vrgAHm.

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 *ICLR 2025 Workshop on Deep Generative Model in Machine Learning: Theory, Principle and Efficacy*, 2025. URL https://openreview.net/forum?id=wzl61tIUj6.

Jingyang Ou, Shen Nie, Kaiwen Xue, Fengqi Zhu, Jiacheng Sun, Zhenguo Li, and Chongxuan Li. Your absorbing discrete diffusion secretly models the conditional distributions of clean data. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=sMyXP8Tanm.

Jiayi Pan, Junjie Zhang, Xingyao Wang, Lifan Yuan, Hao Peng, and Alane Suhr. Tinyzero. https://github.com/Jiayi-Pan/TinyZero, 2025. Accessed: 2025-01-24.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *OpenAI technical report*, 2018. URL https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI technical report*, 2019. URL https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

Subham Sekhar Sahoo, Marianne Arriola, Aaron Gokaslan, Edgar Mariano Marroquin, Alexander M Rush, Yair Schiff, Justin T Chiu, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=L4uaAR4ArM.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

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

Petru Soviany, Radu Tudor Ionescu, Paolo Rota, and Nicu Sebe. Curriculum learning: A survey, 2022. URL https://arxiv.org/abs/2101.10382.

Haoran Sun, Lijun Yu, Bo Dai, Dale Schuurmans, and Hanjun Dai. Score-based continuous-time discrete diffusion models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=BYWWwSY2G5s.

Xiaohang Tang, Rares Dolga, Sangwoong Yoon, and Ilija Bogunovic. wd1: Weighted policy optimization for reasoning in diffusion language models, 2025. URL https://arxiv.org/abs/2507.08838.

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xionghui Chen, Jianxin Yang, Zhenru Zhang, Yuqiong Liu, An Yang, Andrew Zhao, Yang Yue, Shiji Song, Bowen Yu, Gao Huang, and Junyang Lin. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning, 2025. URL https://arxiv.org/abs/2506.01939.

Yongliang Wu, Yizhou Zhou, Zhou Ziheng, Yingzhe Peng, Xinyu Ye, Xinting Hu, Wenbo Zhu, Lu Qi, Ming-Hsuan Yang, and Xu Yang. On the generalization of sft: A reinforcement learning perspective with reward rectification, 2025. URL https://arxiv.org/abs/2508.05629.

Ling Yang, Ye Tian, Bowen Li, Xinchen Zhang, Ke Shen, Yunhai Tong, and Mengdi Wang. Mmada: Multimodal large diffusion language models, 2025. URL https://arxiv.org/abs/2505.15809.

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.

Siyan Zhao, Devaansh Gupta, Qinqing Zheng, and Aditya Grover. d1: Scaling reasoning in diffusion large language models via reinforcement learning, 2025. URL https://arxiv.org/abs/2504.12216.

Zhanhui Zhou, Lingjie Chen, Hanghang Tong, and Dawn Song. dllm: Simple diffusion language modeling. https://github.com/ZHZisZZ/dllm, 2025.

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

A PROOF OF THE WEIGHTED SFT LOSS

In order to prove this theorem, we will first prove two lemmas, and then proceed to prove the main theorem. Our proof follows that of Ou et al., 2025, with the key difference that our Q matrix incorporates varying masking rates β, whereas Ou et al., 2025 assumes all masking rates are identical.

Here, we assume a fixed matrix Q during the derivation; however, in practical entropy-driven masking, Q functions more as a heuristic variance-control scheme. We encourage future research to extend the theoretical framework to account for a dynamic Q.

A.1 PROOF OF LEMMAS

Let the original response of the model be x_0. At a given timestep t, let the partially masked response be x_t, and $[M]$ denotes the mask. $\beta_{x_t^i}$ denotes the masking rate corresponding to the i-th token. The definition of $\bar{f}$ can be found in Equation 6.

Lemma 1. *Let $x_t^i = [M]$ and ${x'}_t^i \neq [M]$. Then we have*

$$\frac{p_t(x_t^1, \ldots, {x'}_t^i, \ldots, x_t^d)}{p_t(x_t^1, \ldots, x_t^i, \ldots, x_t^d)} = \frac{\exp(-\beta_{x_t^i}\,\bar{f}(t))}{1-\exp(-\beta_{x_t^i}\,\bar{f}(t))}\; p_0({x'}_t^i \mid x_t^{\text{unmasked}}).$$

On the other hand, if $x_t^i \neq [M]$, then

$$\frac{p_t(x_t^1, \ldots, {x'}_t^i, \ldots, x_t^d)}{p_t(x_t^1, \ldots, x_t^i, \ldots, x_t^d)} = 0; \tag{17}$$

Intuitively, this theorem characterizes the exact transition probabilities during the training process. It also establishes a connection between the concrete score function and the clean data distribution via a scalar factor.

Proof. We first calculate $p_{t|0}(x_t|x_0)$:

Based on the definition of Q_t, we can easily see that $p_{t|0}(x_t|x_0) = 0$ if $x_t \neq x_0$ and $x_t \neq [M]$

Now we divide the time interval $(0, t)$ into n intervals, namely $(0, s_1), (s_1, s_2), \cdots, (s_{n-1}, t)$. Here we denote $s_0 = 0, s_n = t$

then the total probability of $p_{t|0}(x_t = x_0|x_0)$ is

$$\begin{aligned}
&\prod_{i=0}^{n-1} p_{s_{i+1}|s_i}(x_{s_{i+1}} = x_{s_i}|x_{s_i}) \\
=&\prod_{i=0}^{n-1} \Big(1 + Q_{s_i}(x_{s_i}, x_{s_i})(s_{i+1} - s_i) + o(s_{i+1} - s_i)\Big) \\
=&\exp\Big(\sum_{i=0}^{n-1} \ln\Big(1 + Q_{s_i}(x_{s_i}, x_{s_i}) \times (s_{i+1} - s_i) + o(s_{i+1} - s_i)\Big)\Big) \\
=&\exp\Big(\sum_{i=0}^{n-1} [f(s_i)Q(x_{s_i}, x_{s_i})(s_{i+1} - s_i) + o(s_{i+1} - s_i)]\Big)
\end{aligned}$$

Now we select a series of appropriate n and $s_1, s_2, \ldots, s_{n-1}$ such that while $n \to +\infty$, $\max(s_{i+1} - s_i) \to 0$. Now by recalling the definition of integral, we obtain:

$$\exp\Big(\sum_{i=0}^{n-1} [f(s_i)Q(x_{s_i}, x_{s_i})(s_{i+1} - s_i) + o(s_{i+1} - s_i)]\Big) \to \exp(-\beta_{x_0}\bar{f}(t))$$

Thus, we have

$$p_{t|0}(x_t|x_0) = \begin{cases} \exp(-\beta_{x_0}\bar{f}(t)) & \text{if } x_t = x_0 \\ 1 - \exp(-\beta_{x_0}\bar{f}(t)) & \text{if } x_t = [M] \\ 0 & \text{otherwise} \end{cases}$$

Now, based on the above result, recalling the independence of different dimensions in the diffusion process, we can get a stronger result:

Lemma 2. *Suppose that we have x_t in which the tokens $a_1, a_2, ..., a_k$ are masked and $b_1, b_2, ..., b_{n-k}$ are unmasked; then:*

$$p_t(x_t) = \prod_{i=1}^{k}(1 - \exp(-\beta_{a_i}\bar{f}(t))) \times \prod_{j=1}^{n-k} \exp(-\beta_{b_j}\bar{f}(t))p_0(x_t^{unmasked})$$

Proof. Exploiting the independence across dimensions in the diffusion process, we may, without loss of generality, set $a_i = i$ and $b_i = k + i$. Equivalently, this corresponds to masking the first k tokens while leaving the remaining $n - k$ tokens unmasked.

Then we obtain

$$\begin{aligned}
&p_t([M]\dots[M]x_t^{k+1}\dots x_t^n) \\
=&\sum_{x_0} p_{t|0}([M]\dots[M]x_t^{k+1}\dots x_t^n|x_0)p_0(x_0) \\
=&\sum_{x_0^1\dots x_0^n}\prod_{i=1}^{k} p_{t|0}^k([M]|x_0^k) \times \prod_{i=k+1}^{n} p_{t|0}(x_t^k|x_0^k)p_0(x_0^1,\dots x_0^n) \\
=&\sum_{x_0^1\dots x_0^n}\prod_{i=1}^{k}(1-\exp(-\beta_i\bar{f}(t))) \times \prod_{i=k+1}^{n} \exp(-\beta_i\bar{f}(t))p_0(x_0^1,\dots x_0^n) \\
=&\sum_{x_0^{k+1}\dots x_0^n}\prod_{i=1}^{k}(1-\exp(-\beta_i\bar{f}(t))) \times \prod_{i=k+1}^{n} \exp(-\beta_i\bar{f}(t))p_0(x_0^{k+1},\dots x_0^n)
\end{aligned}$$

Recalling the defination, $p_0(x_0^{k+1}, \dots, x_0^n)$ can be expressed as $p_0(x_t^{\text{unmasked}})$, the desired result follows.

□

Now we append x_t' and x_t to the above result and we obtain the result of Lemma 1.

□

A.2 Proof of the Main Theorem

Theorem 1. *Assuming the Q matrix takes the form given in Equation 9, let the initial sequence be x_0 and the sequence at time t be x_t. Under this setting, the i-th token is masked with probability $t_i = 1-(1-t)^{\frac{\beta_{x^i}}{\beta_{ref}}}$, where β_{x^i} denotes the masking rate of the i-th token, and β_{ref} is a specified reference masking rate. Moreover, the importance-weighted supervised finetuning loss can be derived as follows:*

$$L = -\sum_i \mathbb{E}_{t_i}\left[\mathbf{1}\left[x_{t_i}^i = \mathbf{M}\right]\frac{1}{t_i}\log p\big(x_0^i \mid x_t\big)\right] \tag{10}$$

Proof. Now we consider the Denoising Score Entropy, introduced in Lou et al. (2024). DSE loss provides a principled training objective for diffusion models by measuring the discrepancy between the model score function and the true score of the perturbed data distribution.

The DSE loss can be written as:

$$\int_0^T \mathbb{E}_{x_t\sim p_{t|0}(x_t|x_0)} \times \sum_{x_t'\neq x_t} Q_t(x_t', x_t)\Big(s_\theta(x,t)_{x_t'} - \frac{p_{t|0}(x_t'|x_0)}{p_{t|0}(x_t|x_0)}\log s_\theta(x_t,t)_{x_t'} + C\Big)dt$$

$$C = K\left(\frac{p_{t|0}(x'_t|x_0)}{p_{t|0}(x_t|x_0)}\right), \quad K(a) = a\log a - a,$$

Here C is a constant irrelevant to the loss. The function $s_\theta(x_t, t)$ denotes a score network used to estimate the ratio$\frac{p_t(x'_t)}{p_t(x_t)}$.

In practice, we often estimate $s_\theta(x,t)_{x'_t}$ using

$$s_\theta(x,t)_{x'_t} \approx p_t(x_t^1 \dots {x'}_t^i \dots x_t^d) \times \frac{1}{p_t(x_t^1 \dots x_t^i \dots x_t^d)}$$

We denote $\tilde{Q}$ as the reverse transition matrix corresponding to Q. Then, the reverse process satisfies

$$\tilde{Q}_t(x_t, x'_t) = \begin{cases} \frac{p_t(x'_t)}{p_t(x_t)} Q_t(x'_t, x_t) & \text{if } x'_t \neq x_t \\ -\sum_{k \neq x_t} \tilde{Q}_t(x_t, k) & \text{if } x'_t = x_t \end{cases}$$

Now we use Lemma 1 and calculate the three parts of the loss separately:

Using the fact that

$$\sum p_0({x'}_t^i | x_t^{\text{unmasked}}) = 1$$

we obtain:

$$\begin{aligned} &\int_0^T \mathbb{E}_{x_t \sim p_{t|0}(x_t|x_0)} \sum_{x'_t \neq x_t} Q_t(x'_t, x_t) s_\theta(x,t)_{x'_t} dt \\ =&\int_0^T \mathbb{E}_{x_t \sim p_{t|0}(x_t|x_0)} \sum_{x_t^i = [M]} \beta_i f(t) \times \frac{\exp(-\beta_{x_t^i} \bar{f}(t))}{1 - \exp(-\beta_{x_t^i} \bar{f}(t))} dt \end{aligned}$$

Using the properties of $-\frac{p_{t|0}(x'_t|x_0)}{p_{t|0}(x_t|x_0)}$ we obtain:

$$\begin{aligned} &\int_0^T \mathbb{E}_{x_t \sim p_{t|0}(x_t|x_0)} \sum_{x_t^i \neq x_t} Q_t(x'_t, x_t) \times \left(-\frac{p_{t|0}(x'_t|x_0)}{p_{t|0}(x_t|x_0)} \log s_\theta(x_t, t)_{x'_t}\right) dt \\ =&\int_0^T \mathbb{E}_{x_t \sim p_{t|0}(x_t|x_0)} \sum_{x_t^i = [M], j \neq [M]} -f(t)\beta_i \times \mathbf{1}[x_0^i = j] \log s_\theta(x_t, t)_{x'_t} dt \end{aligned}$$

Recalling that $K(a) = a\log a - a$ with $K(0) = 0$, we can safely ignore the terms $\mathbf{1}[x_0^i = j]$ and $j \neq [M]$.

$$\begin{aligned} &\int_0^T \mathbb{E}_{x_t \sim p_{t|0}(x_t|x_0)} \sum_{x'_t \neq x_t} Q_t(x'_t, x_t) \times K\left(\frac{p_{t|0}(x'_t \mid x_0)}{p_{t|0}(x_t \mid x_0)}\right) dt \\ =&\int_0^T \mathbb{E}_{x_t \sim p_{t|0}(x_t|x_0)} \sum_{x_t^i = [M], j \neq [M]} \beta_i f(t) \times K\left(\frac{\exp(-\beta_{x_t^i} \bar{f}(t))}{1 - \exp(-\beta_{x_t^i}) \bar{f}(t)} \mathbf{1}[x_0^i = j]\right) dt \\ =&\int_0^T \mathbb{E}_{x_t \sim p_{t|0}(x_t|x_0)} \sum_{x_t^i = [M]} \beta_i f(t) \times \frac{\exp(-\beta_{x_t^i} \bar{f}(t))}{1 - \exp(-\beta_{x_t^i} \bar{f}(t))} \times \left(\log \frac{\exp(-\beta_{x_t^i} \bar{f}(t))}{1 - \exp(-\beta_{x_t^i} \bar{f}(t))} - 1\right) dt \end{aligned}$$

By adding them all together, especially combining the first and third term, we get

$$\int_0^T \mathbb{E}_{x_t \sim p_{t|0}(x_t|x_0)} \sum_{x_t^i=[M]} -\frac{\beta_i f(t)\exp(-\beta_i \bar{f}(t))}{1-\exp(-\beta_i \bar{f}(t))} \times \log\left(\frac{\exp(-\beta_i \bar{f}(t))}{1-\exp(-\beta_i \bar{f}(t))} q_\theta(x_0^i | x_t^{\text{unmasked}})\right) dt$$

Now we apply a change of variables. By defining

$$t_i = 1 - \exp\big(-\beta_{x_t^i}\, \bar{f}(t)\big),$$

we obtain

$$dt_i = \beta_{x_t^i} f(t) \exp\big(-\beta_{x_t^i}\, \bar{f}(t)\big) dt$$

and the loss can be rewritten as:

$$\sum_i \int_0^1 \mathbb{E}_{x \sim p_{t|0}(x_t|x_0)} \Big[-\frac{1}{t_i} \mathbf{1}[x_t^i = [M]] \times \log(q_\theta(x_0^i | x_t^{\text{unmasked}}))\Big] dt_i + C$$

Here, x_{t_i} denotes the distribution in which x^i is masked with probability t_i.

Recall that

$$t_i = 1 - \exp(-\beta_{x^i}\, \bar{f}(t)).$$

Suppose we have a reference value β_{ref} and define

$$t = 1 - \exp(-\beta_{\text{ref}}\, \bar{f}(t)).$$

Then we can express t_i as

$$\begin{aligned} t_i =& 1 - \exp(-\beta_{x^i}\, \bar{f}(t)) \\ =& 1 - \exp(-\beta_{x^i} (\frac{1}{\beta_{ref}} \log(1-t))) \\ =& 1 - (1-t)^{\frac{\beta_{x^i}}{\beta_{\text{ref}}}} \end{aligned}$$

In practice, one can sample t and calculate t_i using the above formula. □

B Algorithm Pipeline

In Algorithm 1 and Algorithm 2, we provide a detailed description of the implementation of GIFT.

Algorithm 1: Creating Betas from Entropy

Input: model $\mathcal{M}$, input_ids $\mathcal{I}$, mask_token_id m
Output: Masking parameter β

Function `EstimateRate(`$\mathcal{M}, \mathcal{I}, M, m$`):`
  `// mask out the answer`
  for $i \leftarrow 1$ **to** $|\mathcal{I}|$ **do**
    $\mathcal{I}'^i \leftarrow (1 - M^i) \cdot \mathcal{I}^i + M^i \cdot m$;
  $z \leftarrow \mathcal{M}(\mathcal{I}')$.logits;
  `// calculate entropy`
  for $i \leftarrow 1$ **to** $|\mathcal{I}|$ **do**
    $H_i \leftarrow -\sum_j \text{softmax}(z_{ij}) \cdot \text{log_softmax}(z_{ij})$;
    $\beta_i \leftarrow M^i \sqrt{H_i}$;
  return β;

Algorithm 2: Weighted Entropy-driven Fine-Tuning

Input: model $\mathcal{M}$, input_ids $\mathcal{I}$, labels l, mask_token_id m, prompt lengths L
Output: Loss $\mathcal{L}$

Function `ComputeLoss(`$\mathcal{M}, \mathcal{I}, m, L$`)`:
  $t \sim \mathcal{U}(0, 1)$;
  `// mask the model's answer`
  for $i \leftarrow 1$ **to** $|\mathcal{I}|$ **do**
    $M^i = \mathbf{1}_{\{i > L\}}$
  $\beta \leftarrow$ `EstimateRate`$(\mathcal{M}, \mathcal{I}, M, m)$;
  $\beta_{ref} \leftarrow \frac{\sum_i \beta_i \cdot M_i}{\sum_i M_i}$;
  `// Compute the masking probability for each token`
  for $i \leftarrow 1$ **to** $|\mathcal{I}|$ **do**
    $t_i \leftarrow 1 - (1 - t)^{\frac{\beta_i}{\beta_{ref}}}$;
    $M'^i \sim$ Bernoulli(t_i) ;
  $\mathcal{I}' \leftarrow (1 - M') \cdot \mathcal{I} + M' \cdot m$;
  $z \leftarrow \mathcal{M}(\mathcal{I}')$.logits;
    $S \leftarrow \sum_i \mathbf{1}_{\{M'^i = 1\}} \frac{1}{t_i} \cdot$ CrossEntropy(z_i, l_i);
  $N \leftarrow \sum_i \mathbf{1}_{\{M'^i = 1\}}$;
  $\mathcal{L} \leftarrow S/N$;
  return $\mathcal{L}$;

C HYPERPARAMETERS AND DETAILED SETTINGS

C.1 TRAINING HYPERPARAMETERS AND SETTINGS

Our training is divided into two settings: LoRA fine-tuning and full parameter fine-tuning. LoRA fine-tuning is conducted on 4 H100 GPUs, while full parameter fine-tuning is conducted on 8 H100 GPUs.

For LoRA fine-tuning, experiments are conducted with a learning rate of 1×10^{-5} and a total of 20 training epochs. We apply gradient accumulation with 4 steps and used an effective per-device training batch size of 1. To reduce memory cost, we employ LoRA with rank $r = 128$, scaling factor $\alpha = 256$, and a dropout rate of 0.05. All hyperparameters, including model architecture, optimization, LoRA configuration, and training settings, are listed in Table 8.

Category	**Hyperparameter = Value**

Optimization	optimizer = adamw_torch; weight_decay = 0.1;
	learning_rate = 1e-5; lr_scheduler_type = linear;
	adam_beta1 = 0.9; adam_beta2 = 0.999; adam_epsilon = 1e-8;
	init_fn = mitchell; init_std = 0.02; max_grad_norm = 1

LoRA Configuration	LoRA = `True`; $r = 128$; lora_alpha = 256; lora_dropout = 0.05; target_modules = ["q_proj", "k_proj", "v_proj"]
Training	num_train_epochs = 20; per_device_train_batch_size = 1; gradient_accumulation_steps = 4; precision = bf16; max_sequence_length = 4096; seed = 42

Table 8: Hyperparameters used in LoRA training.

For full parameter fine-tuning, we train for 5 epochs with a learning rate of 2×10^{-5}. Other settings are detailed in Table 9.

Category	Hyperparameter = Value

Optimization	optimizer = adamw_torch; weight_decay = 0;
	learning_rate = 2e-5; lr_scheduler_type = cosine;
	adam_beta1 = 0.9; adam_beta2 = 0.999; adam_epsilon = 1e-8;
	init_fn = mitchell; init_std = 0.02; max_grad_norm = 1

Training	num_train_epochs = 5; per_device_train_batch_size = 4;
	gradient_accumulation_steps = 1; precision = bf16;
	max_sequence_length = 4096; seed = 42

Table 9: Hyperparameters used in full parameter training.

C.2 Evaluation Hyperparameters and Settings

In the evaluation, we set the block length to 32. For the two smaller datasets, Sudoku and Countdown, the generation length is set to 512; for the two larger datasets, MATH500 and GSM8K, the generation length is set to 256. The number of diffusion steps is always set to half of the generation length, and we use the low-confidence remasking method. Regarding the specific partitioning of validation sets for these four datasets, we follow the settings in `wd1` (Tang et al., 2025).

D Motivation for Using Entropy-based Methods as the Weight Coefficient

In this section, we provide an intuitive perspective to justify the choice of *entropy* as the coefficient β, as this section is intended to provide motivation rather than a rigorous proof.

In brief, our approach of using entropy as an importance weight helps reduce the variance of the loss estimation. Specifically, prior work has demonstrated that importance sampling achieves minimum variance when the sampling proposal is proportional to the L2 norm of the gradient. Through reasonable approximation, we found that token-wise entropy is approximately proportional to the L2 norm of the gradient, which motivates our choice of this metric. A detailed discussion is provided below.

In Alain et al. (2016), the authors proved that importance sampling has minimum variance when the sampling proposal is proportional to the L_2-norm of the gradient.

Recall that in a model, p is generated by applying softmax to the output logits. More specifically,

$$p_k = \frac{e^{z_k}}{\sum_j e^{z_j}}$$

.

Here z is the logits generated by the model.

Let c be the label and p_c be the probability of this token. Then we have

$$\frac{\partial \log p_c}{\partial z_k} = \begin{cases} -p_k & \text{if } k \neq c \\ 1 - p_c & \text{if } k = c \end{cases}$$

In the following discussion, we assume that the base model generates the correct token with relatively high probability and other tokens with relatively low probability. Therefore, L_2-norm of $\frac{\partial \log p_c}{\partial \mathbf{z}}$ can be approximated as $1 - p_c$. Since $\frac{\partial \log p_c}{\partial \mathbf{z}}$ constitutes a key component of the loss gradient, theoretical insights suggest that selecting an importance sampling distribution that is positively correlated with this term might help reduce the variance of the loss, thereby enhancing training stability and effectiveness. We now briefly analyze why entropy can serve this purpose.

The entropy function $H(\mathbf{p})$ admits the following lower bound:

$$H(\mathbf{p}) \geq -p_c \log p_c + (1 - p_c)(\log(n-1) - \log(1 - p_c))$$

where n denotes the vocabulary size. Since the term $-p_c \log p_c$ satisfies

$$-p_c \log p_c < p_c(1 - p_c), \tag{18}$$

it's negligible compared to $(1-p_c)\log(n-1)$. We may safely ignore it for approximation purposes.

Moreover, we empirically observe that the variation in incorrect predictions across different samples is relatively small. Under this assumption, $H(\mathbf{p})$ is positively correlated with $(1-p_c)$.

On the other hand, using entropy $H(\mathbf{p})$ is superior to directly using $(1-p_c)$ in that it preserves richer information about the underlying distribution. Due to the prevalence of synonyms and semantically similar tokens, the true probability distribution is unlikely to be unimodal. Instead, it is more reasonable to expect the probability mass to be concentrated on a small subset of tokens. In this respect, entropy $H(\mathbf{p})$ captures such structural information significantly better than $(1-p_c)$. In pratice, we employ a power function based on $\sqrt{H(\mathbf{p})}$ as the importance weighting function.

E SFT Role in Subsequent RL

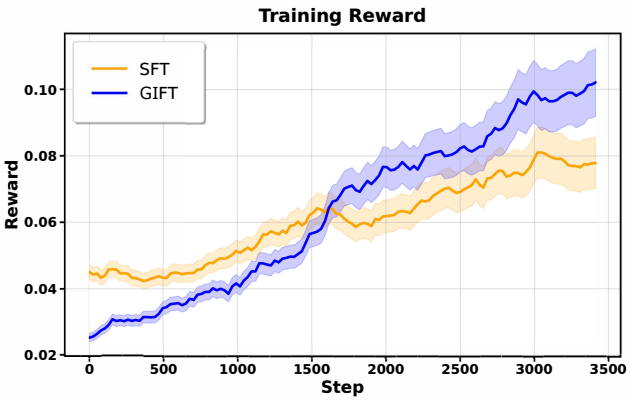


Figure 3: Reward curves of models cold-started with GIFT and SFT during subsequent reinforcement learning training. The curves are smoothed using a time-weighted EMA. As shown, the model initialized with GIFT achieves higher rewards.

Reinforcement learning (RL) post-training has become a crucial stage in the training of large language models (DeepSeek-AI, 2025). Supervised fine-tuning (SFT) cold-start plays an important role in preparing the model for subsequent RL, as SFT establishes a reliable initial policy which can then be further refined by RL objectives. Therefore, it is essential to verify whether models trained with GIFT can maintain their advantages during subsequent RL training. To this end, we fine-tune models on the s1K dataset using both GIFT and SFT, and then further train them with RL on the Sudoku dataset for comparison.

As shown in Figure 3, models initialized with GIFT achieve higher rewards throughout RL training. Moreover, as reported in Table 10, the GIFT-trained model achieves significantly better evaluation results. These findings demonstrate that models cold-started with GIFT are effective in guiding subsequent RL learning.

	Base	**SFT**	**GIFT**	**SFT+RL**	**GIFT+RL**
Sudoku	5.5	4.6	**7.8**	8.9	**13.3**

Table 10: Evaluation results of models cold-started with GIFT and SFT during subsequent RL training.

F Discussion on the Rigor of the Theoretically Derived Loss

Our theoretical derivation relies on certain necessary simplifying assumptions. Specifically, we assume a fixed matrix Q during the derivation; however, in practical entropy-driven masking, Q functions more as a heuristic variance-control scheme.

To maintain theoretical rigor, we can also construct a training method that strictly adheres to our theoretical derivation by consistently using the initial, non-finetuned model as a reference model to calculate the importance weight β. In this setting, the transition matrix Q remains fixed during training, making our theoretical derivation fully applicable. We denote this strictly theoretically-aligned version as GIFT*. We compare the performance of SFT and GIFT* using LoRA fine-tuning on the s1K dataset:

	Sudoku	**Countdown**	**GSM8K**	**MATH500**	**Average**
SFT	4.2 ± 0.4	21.1 ± 1.4	$\mathbf{78.5 \pm 0.2}$	32.6 ± 0.9	34.1 ± 0.4
GIFT*	$\mathbf{5.4 \pm 0.4}$	$\mathbf{24.3 \pm 0.3}$	77.2 ± 0.3	$\mathbf{32.8 \pm 0.4}$	$\mathbf{34.9 \pm 0.2}$

Table 11: Comparison between SFT and the strictly theoretically-aligned GIFT* on the s1K dataset. Even with a fixed reference model, GIFT* consistently outperforms SFT across most tasks.

As shown in Table 11, the theoretically-aligned GIFT* method still achieves statistically significant improvements over SFT. One primary reason we choose to calculate the importance weight β using the model being trained, rather than a fixed reference model is efficiency. This approach only requires loading the active model into GPU memory without the additional overhead of a reference model.

G Discussion on the Connection to Curriculum Learning

Curriculum Learning (Soviany et al., 2022) shares a similar high-level intuition with our approach, as both aim to improve training effectiveness by allocating different importance or tasks to the model at various stages of training. However, our method, GIFT, differs from traditional Curriculum Learning in two fundamental aspects.

Granularity and Model-Adaptivity. In Curriculum Learning, the difficulty of a problem is typically defined at the task or dataset level, and these difficulty measures are often static and independent of the specific model state. In contrast, the entropy-based weighting we employ reflects the difficulty of each individual token with respect to the current model state. This ensures that the training signal is inherently model-adaptive.

Training Paradigm. Curriculum Learning generally follows an easy-to-hard paradigm, where the model is first exposed to simpler tasks before gradually transitioning to more complex ones. Conversely, GIFT consistently focuses on the most challenging portions of the data (those that the current model finds most uncertain or confusing) to maximize learning efficiency.