



TIME IS A FEATURE: EXPLOITING TEMPORAL DYNAMICS IN DIFFUSION LANGUAGE MODELS

Wen Wang^{1,2*}, Bozhen Fang^{1*}, Chenchen Jing^{1,3}, Yongliang Shen¹, Yangyi Shen⁴, Qiuyu Wang², Hao Ouyang², Hao Chen¹, Chunhua Shen^{1,3,2}

¹ Zhejiang University ² Ant Group ³ Zhejiang University of Technology ⁴ Stanford University

ABSTRACT

Diffusion large language models (dLLMs) generate text through iterative denoising, yet current decoding strategies discard rich intermediate predictions in favor of the final output. Our work here reveals a critical phenomenon, **temporal oscillation**, where correct answers often emerge in the middle process, but are overwritten in later denoising steps. To address this issue, we introduce two complementary methods that exploit temporal consistency: 1) **Temporal Self-Consistency Voting**, a training-free, test-time decoding strategy that aggregates predictions across denoising steps to select the most consistent output; and 2) a post-training method termed **Temporal Consistency Reinforcement**, which uses Temporal Semantic Entropy (TSE), a measure of semantic stability across intermediate predictions, as a reward signal to encourage stable generations. Empirical results across multiple benchmarks demonstrate the effectiveness of our approach. Using the negative TSE reward alone, we observe a remarkable average improvement of **24.7%** on the Countdown dataset over an existing dLLM. Combined with the accuracy reward, we achieve absolute gains of **2.0%** on GSM8K, **4.3%** on MATH500, **6.6%** on SVAMP, and **25.3%** on Countdown, respectively. Our findings underscore the untapped potential of temporal dynamics in dLLMs and offer two simple yet effective tools to harness them.

Project webpage: <https://aim-uofa.github.io/dLLM-MidTruth>

1 INTRODUCTION

Diffusion large language models (dLLMs) (Nie et al., 2025; Zhu et al., 2025; Ye et al., 2025) have recently emerged as a promising alternative to the auto-regressive (AR) large language models, garnering significant attention for their competitive performance and potential for faster inference. In contrast to AR models, which generate text in a strictly sequential manner by predicting one token at a time, dLLMs operate through iterative cycles of denoising and remasking, predicting all masked tokens in parallel at each step. A small subset of the predicted tokens, typically those with high confidence (Nie et al., 2025), are retained, while the remaining tokens are remasked and refined in subsequent steps. *Despite their drastic architectural differences, current dLLMs typically adopt a decoding strategy that mirrors AR models: solely relying on the sequence predicted in the final denoising step as the final answer, and discarding all the intermediate predictions.*

In this work, we challenge this convention by uncovering a critical phenomenon that we term **temporal oscillation**: correct answers often appear during intermediate denoising steps but are overwritten in later iterations. This discrepancy between the final output and intermediate correctness suggests that dLLMs possess rich temporal dynamics that are largely under-utilized.

As depicted in Fig. 1, we analyze two key metrics across four widely used benchmark datasets using two representative models: LLaDA-8B-Instruct (Nie et al., 2025) and LLaDA-1.5 (Zhu et al., 2025). The first metric, final pass rate, measures the accuracy of the final output, while the second, ever-pass rate, captures whether a correct answer appears at any point during the decoding process. In Fig. 1a,

*WW and BF contributed equally. CS is the corresponding author.

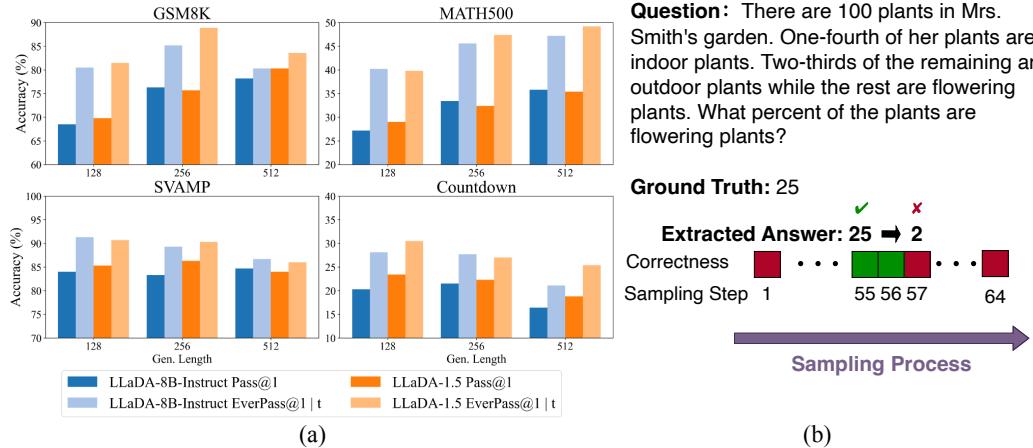


Figure 1: **Illustration of temporal oscillation during sampling.** (a) Across four datasets, a significant gap is observed between the **final answer’s pass rate** (denoted as Pass@1) and the **ever-pass rate at any intermediate step** (denoted as EverPass@1 | t). This gap reveals the phenomenon we refer to as temporal oscillation, where correct intermediate answers are sometimes overwritten as the generation proceeds. (b) Example of temporal oscillation: For a given math problem, the model initially gives the correct answer, 25, at an intermediate step (e.g., step 55), aligning with the ground truth. However, by the final step, this correct answer is replaced with an incorrect one: 2.

a consistent and significant discrepancy exists between these metrics. This gap reveals a critical phenomenon: models often generate correct answers during intermediate steps but subsequently overwrite them with incorrect ones. Fig. 1b illustrates this concretely—in a math problem, the model produces the correct answer “25” at sampling step 55, only to replace it with an incorrect “2” by the final step 64. More examples on temporal oscillation are presented in Appendix D.4.

To better understand this behavior, we analyze dLLMs from the entropy perspective and introduce a new metric: Temporal Semantic Entropy (TSE), which captures the distribution of semantic variation across intermediate outputs during decoding. Specifically, we collect the sequence of intermediate answers across denoising steps and group them into clusters based on semantic equivalence. TSE then quantifies the degree of uncertainty in the semantic content of these answers. A higher TSE indicates greater semantic fluctuation throughout the trajectory, *i.e.*, the model changes its answer frequently, while a lower TSE suggests convergence toward a stable meaning.

To harness the latent signals embedded in dLLMs decoding, we treat temporal oscillation as an informative feature and develop two complementary methods that exploit the temporal dynamics:

- **Temporal Self-Consistency Voting:** A training-free test-time decoding strategy that aggregates predictions across denoising steps and selects the most consistent output, considerably improving accuracy with negligible computational overhead.
- **Temporal Consistency Reinforcement:** A post-training method based on reinforcement learning that uses negative TSE as a reward signal to encourage stable and consistent generations. Importantly, leveraging negative TSE as the reward enables performance improvements without requiring ground-truth labels for reward computation.

Experiments across multiple datasets validate the effectiveness of both our decoding-time strategy and our RL-based post-training method. Specifically, Temporal Self-Consistency Voting brings an average improvement of **1.5%** over the LLaDA-8B-Instruct baseline with negligible overhead. In terms of Temporal Consistency Reinforcement, fine-tuning using the negative TSE reward alone, we observe a substantial average improvement of **24.7%** on the Countdown dataset. When combined with the accuracy reward derived from ground truth, our approach yields notable improvements across diverse datasets: **2.0%** on GSM8K, **4.3%** on MATH500, **6.6%** on SVAMP, and an impressive **25.3%** on Countdown, respectively. By quantifying and leveraging temporal consistency, we offer a new perspective on dLLM decoding and introduce practical tools to unlock their potential. We hope that this study inspires further research into the temporal characteristics of diffusion decoding.

2 RELATED WORK

Diffusion language models. With the success of diffusion models in image and video generation (Song et al., 2020; Ho et al., 2020; 2022), diffusion methods have been extended to text generation. Early continuous diffusion approaches (Han et al., 2022; Li et al., 2022) operate in continuous space, while others project discrete text onto the probability simplex (Avdeyev et al., 2023; Stark et al., 2024), though these often struggle with complex linguistic patterns. More recent work applies flow matching on the simplex to learn categorical distributions (Davis et al., 2024; Cheng et al., 2024), but remains limited to simpler sequences.

Discrete diffusion models, pioneered by D3PM (Austin et al., 2021), have advanced through masked token frameworks (Sahoo et al., 2024; Nie et al., 2024) and scaling efforts. Lightweight models like Plaid (Gulrajani & Hashimoto, 2023) and SEDD (Lou et al., 2023) show competitive results against GPT-2 (Radford et al., 2019), yet scalability lags behind autoregressive models. To bridge this gap, BD3-LMs (Arriola et al., 2025) and Eso-LMs (Sahoo et al., 2025) interpolate between the autoregressive and diffusion paradigms, allowing both parallel sampling and competitive performance. More recently, attempts have been made to scale up dLLMs. For example, Dream (Ye et al., 2025) converts pre-trained autoregressive models into diffusion models, while LLaDA (Nie et al., 2025) also demonstrates strong training-from-scratch results.

Another line of work explores sampling strategies in dLLMs. For example, Kim et al. (2025) examines the impact of token ordering during sampling and proposes an adaptive inference strategy that improves model performance. Similarly, ReMDM (Wang et al., 2025) introduces a remasking mechanism that leverages inference-time scaling to improve generation quality in dLLMs. Unlike prior works, we analyze and improve dLLMs from the perspective of temporal stability, a domain that remains underexplored.

Test-time Strategy. Test-time strategies (Wei et al., 2022; Madaan et al., 2023; Snell et al., 2024; Yao et al., 2023; Liu et al., 2025a) for LLMs have been extensively studied and widely adopted to enhance the accuracy, consistency, and reliability of model outputs. One of the most straightforward and effective techniques in this domain is Self-Consistency (Wang et al., 2022), which selects the most consistent answer from multiple outputs using majority voting. Our approach leverages the concept of temporal self-consistency, devising a test-time strategy specifically tailored for dLLMs. Notably, our method incurs only negligible inference overhead and can be directly integrated into existing dLLMs frameworks. Another key technique is semantic entropy (Farquhar et al., 2024; Kuhn et al., 2023), a novel uncertainty metric designed for LLMs. It addresses the “semantic equivalence” problem by first clustering sentences that share the same meaning but differ in phrasing, and then computing entropy over these semantic clusters. While semantic entropy has been successfully applied to tasks like uncertainty estimation (Kuhn et al., 2023) and hallucination detection (Farquhar et al., 2024), we introduce Temporal Semantic Entropy to characterize the stability and confidence of dLLMs during the progressive denoising process.

Post-Training using Reinforcement Learning. Group Relative Policy Optimization (Shao et al., 2024; Guo et al., 2025) (GRPO) is a variant of Proximal Policy Optimization (Schulman et al., 2017) (PPO) that directly computes the advantage from rewards within a group of responses, obviating the need for an separately trained value function. GRPO has demonstrated not only strong performance on reasoning tasks such as mathematics and code generation, but also significant potential across a broader range of modalities and downstream tasks (Huang et al., 2025; Shen et al., 2025; Qi et al., 2025; Zhong et al., 2025; Li et al., 2025; Damani et al., 2025). Building on this, related works have introduced further refinements. For instance, DAPO (Yu et al., 2025) employs a dynamic sampling strategy to prevent training batches from consisting entirely of correct or incorrect samples. Other approaches (Zhang et al., 2025; Prabhudesai et al., 2025; Cui et al., 2025; Agarwal et al., 2025) have enhanced RL strategies from an entropy perspective; EMPO (Zhang et al., 2025) estimates rewards directly based on semantic entropy, while Seed-GRPO (Chen et al., 2025) leverages entropy to improve advantage estimation. Recently, the GRPO framework has been adapted for dLLMs in methods like *diffu*-GRPO (Zhao et al., 2025), UniGRPO (Yang et al., 2025), and coupled-GRPO (Gong et al., 2025). A commonality among these methods is their reliance on ground-truth answers to formulate reward signals. In contrast, our approach is distinguished by its unsupervised nature; it enhances the model’s temporal consistency without depending on ground-truth supervision.

3 EXPLORATIONS ON DLLMs

3.1 PRELIMINARIES ON DLLMs

DLLMs formulate text generation as a process of iteratively denoising text sequences across different time steps. Let $x_0 \sim p_{\text{data}}(x_0)$ denote the original input sequence, and x_t represent its noisy version at time $t \in [0, T]$. Typically, the noise is introduced by masking a portion of tokens in the original training data. The forward noising process is defined as a Markov chain $q(x_{1:T} | x_0) = \prod_{t=1}^T q(x_t | x_{t-1})$, which progressively adds noise to x_0 over time steps. This process incrementally transforms the clean sequence x_0 into a highly noisy version x_T through a series of conditional transitions. In contrast, the reverse (generative) process is modeled as:

$$p_\theta(x_{0:T}) = p_\theta(x_T) \prod_{t=1}^T p_\theta(x_{t-1} | x_t) = \prod_{t=1}^T q(x_{t-1} | x_0) p_\theta(x_0 | x_t). \quad (1)$$

This generative process can be divided into two key steps. The first step involves $p_\theta(x_0 | x_t)$, a model trained to predict the clean text sequence based on the current noisy input. We denote the answer decoded at the t -th step as x_0^t , i.e., $x_0^t = p_\theta(x_0 | x_t)$. The second step relies on $q(x_{t-1} | x_0)$, referred to as the re-masking process. This process applies the forward procedure to the currently predicted x_0^t , yielding a sequence x_{t-1} that is less noisy compared to x_t at time t . Various strategies can be employed for this process, such as random re-masking or low-confidence re-masking, as discussed in (Nie et al., 2025).

3.2 TEMPORAL OSCILLATION

In the reverse generative process of DLLMs, the prediction of the clean sequence $p_\theta(x_0 | x_t)$ at a single denoising step is often inaccurate, especially when the noise level t is large. Existing models perform iterative denoising according to the generative process framework, where the final output is determined by the prediction $p_\theta(x_0 | x_1)$ at the last denoising step, while neglecting all intermediate predictions $\{x_0^t = p_\theta(x_0 | x_t)\}_{t=2}^T$ generated during the iterative process. In this work, we conduct an in-depth investigation into these intermediate step results, revealing a critical phenomenon in diffusion-based text generation.

To formalize our analysis, let $e_{i,k}$ denote the Pass @1 rate (Chen et al., 2021) for the prediction generated at the k -th noise step on the i -th question in the evaluation benchmark. Building on this, we introduce the ever pass rate, denoted as EverPass @1 | t , which measures the proportion of questions in the dataset for which the model produces a correct answer at *any* timestep along the sampling trajectory. Formally, it is defined as:

$$\text{EverPass @1} | t = \mathbb{E}_i \left\{ \max_{k \in \{1, \dots, t\}} e_{i,k} \right\} \quad (2)$$

This metric captures the cumulative correctness across all sampling steps, reflecting the overall fraction of questions for which the model arrives at a correct solution at least once, even if that solution is later discarded in the final output.

Experiment Setup. To validate the existence of fluctuating correctness across timesteps, we evaluate both the final pass rate, i.e., the model’s Pass @1 at the last denoising step, and the EverPass @1 | t for two representative DLLMs: LLaDA-8B-Instruct (Nie et al., 2025) and LLaDA-1.5 (Zhu et al., 2025). The evaluation is conducted on varying answer length and four representative mathematical reasoning benchmarks: GSM8K (Cobbe et al., 2021), MATH500 (Lightman et al., 2023), SVAMP (Patel et al., 2021), and Countdown (Pan et al., 2025).

Observations. As shown in Fig. 1 and Table 1 in Sec. 5.2, there exists a significant gap between the final pass rate and the ever pass rate. For example, on the GSM8K dataset with a sequence length of 128, the final pass rate of LLaDA-8B-Instruct is 68.5%, while its ever pass rate reaches 80.5%, resulting in a gap of 12.0%. This discrepancy indicates that a considerable number of questions are correctly answered in one or more intermediate steps, but the answers are subsequently modified to be incorrect during later stages of the refinement process. This phenomenon highlights a potential instability in the model’s iterative decoding, where correct intermediate paths may be overwritten or lost as generation progresses. We term this phenomenon *temporal oscillation*, with examples presented in Appendix D.4.

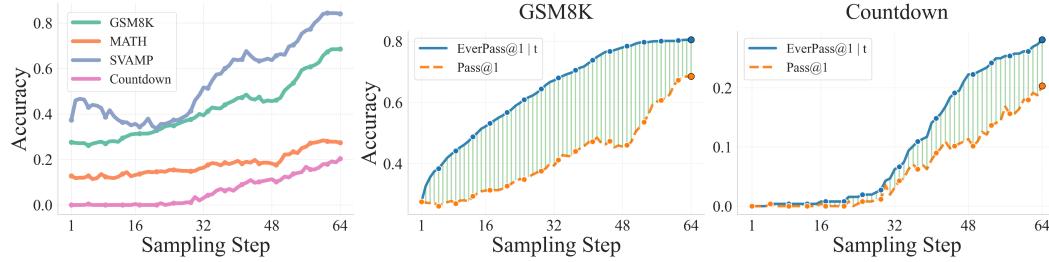


Figure 2: Patterns of accuracy evolution over diffusion sampling steps. Responses are generated with a length of 128 tokens using 64 diffusion steps, based on the LLaDA-8B-Instruct model. **Left:** Accuracy generally improves with more sampling steps across all datasets. Simpler datasets like SVAMP start high, while harder ones like Countdown begin lower but steadily improve. **Middle and Right:** We compare the final pass rate, Pass @1, with the cumulative “ever pass rate” over steps, EverPass @1 | t . A consistent gap remains between Pass @1 and EverPass @1 | t , highlighted by the green shaded area in the figure.

Takeaway 1: Correct intermediate answers are lost during sampling

During sampling, answers may oscillate between correct and incorrect states across diffusion steps. A notable portion of questions achieve correct answers in the intermediate steps, but ultimately yield incorrect results in the final step.

3.3 ANALYSES

To gain a deeper understanding of the temporal oscillation phenomenon, we conduct comprehensive analyses from multiple dimensions: the evolution of accuracy, the dynamics of entropy, and the semantic stability across the decoding process.

Accuracy Across Sampling Steps. To better understand the temporal oscillation phenomenon, we first visualize how accuracy evolves across diffusion sampling steps. As shown in Fig. 2a, overall, accuracy improves with increasing sampling steps across all datasets. Furthermore, the models exhibit distinct accuracy patterns depending on task complexity. For simpler datasets like SVAMP, accuracy is already relatively high at the initial steps. Correct answers often appear early and are gradually refined, with minor fluctuations. In contrast, for more complex datasets like Countdown, the models tend to struggle initially, *i.e.*, accuracy starts low, but they benefit significantly from iterative refinement, with performance improving steadily over subsequent steps.

To better understand these dynamics, we further analyze GSM8K and Countdown, computed Pass @1 and EverPass @1 | t for all intermediate steps. While initial correct predictions appear early in GSM8K but later in Countdown, a noticeable gap between Pass @1 and EverPass @1 | t emerges and gradually widens as the sampling process unfolds. These results suggest that early correct predictions do not guarantee stable reasoning, highlighting the importance of preserving correct intermediate states throughout the sampling process. More analysis on the SVAMP and MATH500 datasets is presented in Appendix C.1.

Entropy Across Sampling Steps. Temporal oscillations reflect the model’s uncertainty during generation. To quantify this, we analyze the average token-level entropy across sampling steps. According to Fig. 3, we observe that across all datasets, the average token-level entropy decreases steadily during the sampling process and approaches zero at the final step. Notably, GSM8K exhibits relatively low entropy from the beginning, indicating higher initial confidence in its predictions.

To understand how entropy relates to correctness, we further categorize questions into three groups: (1) *Finally-Correct*: the final answer is correct, regardless of intermediate steps; (2) *Intermediate-Correct*: at least one intermediate step is correct, but the final answer is incorrect; (3) *Always-Incorrect*: all sampling steps, including the final answer, are incorrect. We compute the average entropy of tokens associated with the predicted answers across diffusion steps for GSM8K and Countdown. The results in Fig. 3a show that questions with incorrect final answers, covering both

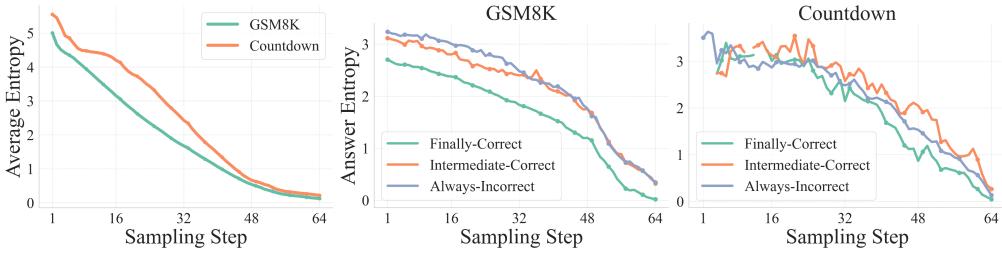


Figure 3: Patterns of entropy evolution over diffusion sampling steps. Responses are generated with a length of 128 using 64 diffusion steps, based on the LLaDA-8B-Instruct model. **Left:** The average token-level entropy decreases steadily during the sampling process. GSM8K exhibits lower entropy than Countdown, consistent with the model’s better performance on GSM8K. **Middle and Right:** Both Intermediate-Correct and Always-Incorrect questions exhibit higher overall entropy compared to Finally-Correct ones. On GSM8K, Intermediate-Correct questions display lower entropy in the early steps than Always-Incorrect, indicating initial confidence, whereas on Countdown the entropy trend is less stable. Note that some points in Right are missing due to the inability to extract answers from generated responses at early diffusion steps for Countdown.

the *Always-Incorrect* and *Intermediate-Correct* categories, tend to exhibit higher overall entropy. Notably, GSM8K questions in the *Intermediate-Correct* category show lower entropy than *Always-Incorrect* ones in the early steps, as shown in Fig. 3b, suggesting initial prediction that deteriorates during the refinement process.

Considering the use of a semi-autoregressive sampling strategy (Nie et al., 2025), we additionally measure the average entropy of the currently generated block, detailed in Appendix C.3.

Temporal Semantic Entropy. While token-level entropy measures local uncertainty, we need a metric that captures semantic consistency across the entire decoding trajectory. To this end, we introduce a metric called Temporal Semantic Entropy. It captures the distribution of semantic variations in answers generated at each step of the iterative denoising process.

During decoding, we obtain a sequence of T intermediate answers, denoted by $\{x_0^t\}_{t=1}^T$. We group these answers into clusters based on their semantic meaning, forming a set of semantic clusters $\mathcal{C} = \{C_1, C_2, \dots, C_K\}$, where each cluster $C_k = \{x_0^t : \text{meaning}(x_0^t) = k\}$ contains all answers with equivalent semantics (outcome equals k). We then define the temporal semantic entropy (TSE) of a trajectory as:

$$\text{TSE}(\{x_0^t\}_{t=1}^T) = -\sum_{C_k} \left(\left[\sum_{x_0^t \in C_k} p(x_0^t) \right] \log \left[\sum_{x_0^t \in C_k} p(x_0^t) \right] \right), \quad (3)$$

which quantifies the uncertainty in the semantic content of answers over the decoding steps. A higher value indicates more semantic variation throughout the trajectory, while a lower value suggests convergence toward a consistent meaning.

As illustrated in Fig. 4, temporal semantic entropy provides valuable insight into the model’s behavior during the generation process. In datasets like Countdown and MATH, where model performance is comparatively lower, we observe a higher temporal semantic entropy than in GSM8K and SVAMP, suggesting increased semantic instability and uncertainty throughout the generation trajectory. Moreover, questions that are ultimately answered correctly, encompassing both *Intermediate-Correct* and *Always-Incorrect*, tend to exhibit lower temporal semantic entropy than those answered incorrectly. This indicates that more stable and semantically consistent decoding trajectories are associated with better performance. Consequently, high temporal semantic

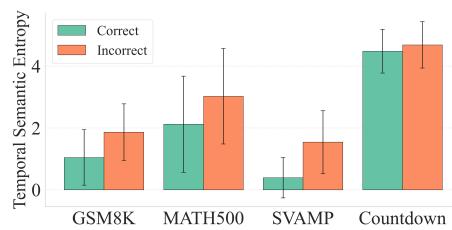


Figure 4: Temporal semantic entropy across four mathematical benchmarks. This metric measures the uncertainty in the semantic content of answers throughout the decoding steps. Statistically, correctly answered questions exhibit lower entropy.

entropy may serve as a signal for model uncertainty and could be used to prioritize samples for further training or refinement. We refer readers to Appendix C.2 for more observations on temporal semantic entropy.

Takeaway 2: Correct answers statistically exhibit lower temporal semantic entropy

Temporal semantic entropy, computed over intermediate predictions during the decoding trajectory, reflects the semantic stability of the model’s outputs. Statistically, correctly answered questions tend to have lower entropy, indicating greater consistency and confidence throughout the generation process.

4 METHOD

4.1 TEMPORAL SELF-CONSISTENCY VOTING

In this section, we propose a temporal self-consistency decoding strategy for dLLMs, which leverages intermediate predictions during sampling to produce more reliable final outputs.

As discussed in Sec. 3.3, although the final timestep in a diffusion trajectory typically yields the most accurate results overall, the correct answer to a given question may sometimes appear earlier in the trajectory. Relying solely on the final prediction thus risks discarding better intermediate outputs. To address this, we propose a method that aggregates predictions across timesteps through a weighted voting mechanism.

Formally, given a diffusion sampling trajectory $\{x_0^t\}_{t=1}^T$, our method selects the final answer a^* according to a weighted vote over all timesteps:

$$a^* = \arg \max_a \sum_{t=1}^T f(t) \cdot \mathbb{1}(\text{meaning } (x_0^t) = a). \quad (4)$$

Here, $\mathbb{1}(\cdot)$ is the indicator function that returns 1 if the decoded meaning of x_0^t matches the candidate answer a , and $f(t)$ is a weighting function over timesteps. Since the accuracy of predictions generally tends to improve as the sampling step increases (or diffusion step decreases), we design $f(t)$ to be a monotonically decreasing function of diffusion step t . In experiments, we explore three types of weighting schemes: constant, linear decay, and exponential decay, as detailed in Sec. 5.2.

Discussion. Our method is conceptually related to self-consistency decoding (Wang et al., 2022), which improves reasoning in autoregressive LLMs by sampling diverse reasoning paths and selecting the most consistent answer via majority voting. However, self-consistency requires multiple full-length forward passes, leading to significant computational overhead. In contrast, our approach requires only a single sampling trajectory from the dLLMs. By exploiting the temporal nature of diffusion inference indicated by Eq. (1), we obtain a series of intermediate predictions essentially for free, without additional model evaluations. This makes our method both efficient and effective for boosting accuracy through temporal aggregation.

4.2 TEMPORAL CONSISTENCY REINFORCEMENT

Motivated by our observation in Sec. 3.3 that correct answers generally exhibit lower Temporal Semantic Entropy (TSE) than incorrect ones, reflecting stronger semantic consistency over time, we propose a post-training approach designed to encourage temporal consistency in model outputs. Specifically, we adopt Group Relative Policy Optimization (GRPO) (Shao et al., 2024; Guo et al., 2025) as our reinforcement learning framework and use TSE as a self-supervised reward signal.

Negative TSE as the Reward. Following GRPO, for each question q sampled from the dataset \mathcal{D} , we draw a group of G responses $\{o_1, o_2, \dots, o_G\}$ from the current policy π_θ . Each response o_i receives a scalar reward $r_i = -\text{TSE}(o_i)$, where $\text{TSE}(o_i)$ is computed using Eq. (3) from Sec. 3.3. This reward encourages the model to produce responses whose intermediate predictions remain semantically consistent throughout the decoding process. Based on this, we define the unnormalized advantage (Liu et al., 2025b) for all tokens $k = 1, \dots, |o_i|$ as $A_i^k(\pi) = r_i(\pi) - \text{mean} \left(\{r_j(\pi)\}_{j=1}^G \right)$.

The training objective follows the standard GRPO formulation with our TSE-based reward:

$$\mathcal{L}_{\text{GRPO}}(\theta) = \mathbb{E}_{o_1, \dots, o_G \sim \pi_\theta(\cdot|q)} \left[\left(\frac{1}{G} \sum_{i=1}^G \frac{1}{|o_i|} \sum_{k=1}^{|o_i|} \min(\rho_i^k A_i^k, \text{clip}(\rho_i^k, 1 - \varepsilon, 1 + \varepsilon) A_i^k) \right) - \beta D_{\text{KL}}[\pi_\theta(\cdot|q) \| \pi_{\text{ref}}(\cdot|q)] \right], \quad (5)$$

where π_{ref} is the reference policy, $\rho_i^k = \frac{\pi_\theta(o_i^k|q)}{\pi_{\theta_{\text{old}}}(o_i^k|q)}$ is the importance sampling ratio, ε is the clipping threshold, and β controls the strength of the KL penalty. To compute the token-level probabilities used in ρ_i^k , we follow the *diffu*-GRPO method (Zhao et al., 2025), which estimates these probabilities by averaging outputs from multiple randomly masked versions of the prompt.

Combining TSE with Accuracy Reward. As an alternative, we can combine TSE with the accuracy reward to achieve further performance gain when ground-truth answers are available during training. We draw on insights from statistical decision theory, specifically the framework of proper scoring rules (Gneiting & Raftery, 2007; Damani et al., 2025). These scoring rules are designed to encourage honest confidence estimates: they are minimized when the predicted confidence c accurately reflects the true likelihood that the model’s output o matches the correct answer o^* .

We derive the confidence c from the TSE by normalizing it with respect to the maximum entropy achievable under uniform sampling. Specifically, we define confidence as $c(o_i) = \frac{\mathcal{H}_{\max} - \text{TSE}(o_i)}{\mathcal{H}_{\max}}$, where $\mathcal{H}_{\max} = \log T$. Here, T denotes the total number of sampling steps. The normalization ensures $c \in [0, 1]$, where higher values indicate greater confidence for correct answers.

The accuracy reward follows a binary scheme: it assigns a reward of 1 when the model’s prediction is correct (*i.e.*, $o_i = o^*$), and 0 otherwise. We propose a composite reward function grounded in the theory of proper scoring rules. Specifically, we adopt the spherical scoring rule, which empirically demonstrates superior performance across tasks. The final reward function is defined as:

$$r_i = \mathbb{1}_{o_i=o^*} + \frac{c(o_i)}{\sqrt{(c(o_i))^2 + (1 - c(o_i))^2}}. \quad (6)$$

Here, the second term incorporates the model’s confidence, rewarding high certainty when the prediction is correct while penalizing overconfidence in incorrect predictions. This design ensures the reward is sensitive not just to accuracy, but also to the reliability of the prediction. A detailed comparison of various scoring rules and their empirical performance in the Appendix D.1.

Discussion. Unlike prior reinforcement learning post-training methods for dLLMs, such as *diffu*-GRPO, which relies on ground-truth answers for reward computation, our approach operates without any labeled data. Instead, we harness the model’s internal temporal dynamics as a self-supervised signal, employing negative TSE to assess answer quality. This enables our method to be broadly applicable, particularly in unsupervised settings, and offers a novel direction for improving dLLMs. Furthermore, we show that combining the negative TSE reward with the accuracy reward based on ground-truth answers yields notably better performance than using the accuracy reward alone.

5 EXPERIMENTS

5.1 IMPLEMENTATION DETAILS

We utilize both LLaDA-8B-Instruct (Nie et al., 2025) and LLaDA-1.5 (Zhu et al., 2025), and evaluate performances on four widely recognized mathematical datasets: GSM8K (Cobbe et al., 2021), MATH500 (Hendrycks et al., 2021), SVAMP (Patel et al., 2021), and Countdown (Pan et al., 2025). In line with d1 (Zhao et al., 2025), we assess the model’s performance across various output lengths. For the temporal self-consistency voting, we employ the exponential function by default to weight the sampling steps. In terms of post-training, we follow d1 and perform supervised fine-tuning (SFT) before reinforcement fine-tuning (RFT) for LLaDA-8B-Instruct. Specifically, we conduct SFT on the s1K dataset (Muennighoff et al., 2025) for 20 epochs, using sequences with a length of 4,096. For LLaDA-1.5, we omit SFT because it often results in performance degradation, likely due to the model having already undergone sophisticated post-training. All training is executed using 8 H800 GPUs. Please refer to the Appendix B for more details.

Table 1: **Performance of temporal majority voting on mathematics benchmarks:** We evaluate three temporal majority voting strategies, including fixed weighting, linear weighting, and exponential weighting, across four math datasets and two model variants: LLaDA-8B-Instruct and LLaDA-1.5. Exponential weighting achieves the best results across datasets and sequence lengths. Bold numbers indicate the highest performance within each group, while green values represent improvements over the baseline. For reference, we include the oracle EverPass @1 | t as a performance upper bound.

		GSM8K			MATH500			SVAMP			Countdown		
Method / Seq Len		128	256	512	128	256	512	128	256	512	128	256	512
LLaDA-8B-Instruct	baseline	68.5	76.3	78.2	27.4	33.4	35.8	84.0	83.3	84.7	20.3	21.5	16.4
	Fixed Weighting	68.0	73.4	78.3	26.6	30.8	34.2	87.0	84.3	84.3	22.7	18.8	11.3
	+ Temporal Voting	70.0	78.0	78.8	28.0	34.4	34.6	87.0	84.3	84.3	24.2	21.9	16.0
	Exp. Weighting	70.1 +1.6	78.7 +2.4	78.9 +0.7	28.4 +1.0	35.6 +2.2	36.2 +0.4	86.0 +2.0	84.3 +1.0	84.7 +0.0	25.0 +4.7	23.4 +1.9	16.4 +0.0
EverPass @1 t		80.5	85.2	80.3	40.2	45.6	47.2	91.3	89.3	86.7	28.1	27.7	21.1
LLaDA-1.5	baseline	69.8	79.4	81.1	29.0	32.4	35.4	85.3	86.3	83.3	21.5	21.1	20.7
	Fixed Weighting	68.8	75.7	80.3	27.3	30.8	34.6	87.3	85.3	84.0	23.4	22.3	18.8
	+ Temporal Voting	71.0	79.8	81.0	29.2	32.8	35.8	86.0	87.0	84.0	24.2	23.4	19.1
	Exp. Weighting	70.7 +0.9	79.8 +0.4	81.1 +0.0	29.0 +0.0	33.2 +0.8	36.2 +0.8	85.7 +0.4	87.7 +1.4	84.3 +1.0	26.2 +4.7	25.0 +3.9	21.1 +0.4
EverPass @1 t		81.5	88.9	83.6	39.8	47.4	49.2	90.7	90.3	86.0	30.5	27.0	25.4

5.2 TEMPORAL SELF-CONSISTENCY VOTING

Voting Strategies. In experiments, temporal voting is performed by applying weighted voting across denoising steps using three schemes: fixed, linear, and exponential. Each uses a weighting function $f(t)$, where t is the current diffusion step. The fixed scheme assigns equal weight to all steps with $f(t) = 1$. The linear scheme emphasizes earlier diffusion steps using $f(t) = 1 - t/T$, while the exponential scheme takes $f(t) = \exp(\alpha(1 - t/T))$, where $\alpha = 5$ by default. Both linear and exponential schemes prioritize early diffusion time steps, *i.e.*, latter sampling steps.

Ablations on Varying Voting Strategies. As shown in Table 1, both linear and exponential weighting improve the model’s inference performance. For example, exponential weighting consistently enhances the performance of LLaDA-8B-Instruct across different datasets, with an average improvement of 1.6%, 1.2%, 1.0% and 2.2% accuracy on the GSM8K, MATH500, SVAMP, and Countdown datasets, respectively. The model performs slightly worse under the fixed weighting strategy, which we attribute to the inaccuracy of the answers predicted in the early sampling steps; assigning the same weight to all steps leads to performance degradation. By default, we adopt the exponential weighting strategy.

We further conducted an ablation study on the hyperparameter α in the exponential weighting. Taking LLaDA-8B-Instruct as an example, we calculated the average improvement across four datasets and various response lengths. As illustrated in Fig. 5b, α values ranging from 1 to 11 consistently yielded improvements. The optimal performance was observed at $\alpha = 5$, achieving an average improvement of 1.5% in accuracy. Based on these results, we set $\alpha = 5$ as the default value.

5.3 TEMPORAL CONSISTENCY REINFORCEMENT

Main Results. Table 2 presents the results of our training leveraging TSE in reinforcement learning across different datasets. Our observations are as follows: **(1)** With TSE reward alone, our method consistently improves model performance across diverse answer lengths and datasets. **(2)** With TSE reward alone, our method achieves performance on par with or even better than using accuracy reward, despite not relying on ground-truth answers during training. On the Countdown dataset, for example, LLaDA-8B-Instruct trained with our approach achieves a 24.7% improvement, significantly outperforming the 15.1% gain observed with d1. **(3)** By combining TSE with accuracy reward, our method significantly outperforms using accuracy reward alone, achieving absolute improvements of 0.9% on GSM8K, 0.2% on MATH500, 1.7% on SVAMP, and 10.2% on Countdown, compared to d1 (Zhao et al., 2025). In total, our approach yields average gains of 2.0%,

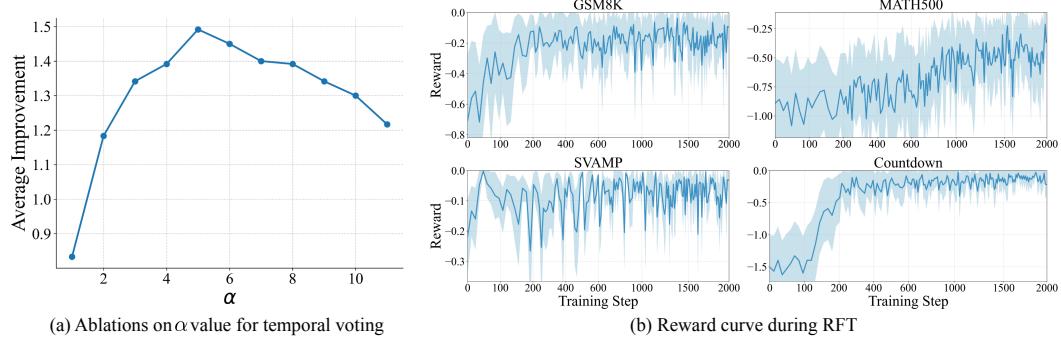


Figure 5: (a) Ablations on α value selection in temporal voting with exponential weighting. (b) Negative temporal semantic entropy reward curve during reinforcement fine-tuning.

Table 2: **Performance of reinforcement fine-tuning on mathematics benchmarks.** Unlike the competing method d1 (Zhao et al., 2025), which requires access to ground-truth answers during training, our approach leverages the negative Temporal Semantic Entropy (TSE) reward without relying on ground-truth labels. Furthermore, combining TSE with accuracy-based rewards leads to even greater performance improvements across multiple benchmarks. The LLaDA baseline is obtained by applying supervised fine-tuning to LLaDA-8B-Instruct using the s1K dataset (Sahoo et al., 2024). Green numbers show the difference between our method and the baseline.

		GSM8K			MATH500			SVAMP			Countdown		
Method / Seq Len		128	256	512	128	256	512	128	256	512	128	256	512
LLaDA	baseline	70.2	78.7	80.1	23.8	34.4	36.8	81.7	83.3	82.7	21.5	19.9	21.5
	accuracy reward (d1)	71.7	78.3	82.3	31.0	36.0	40.4	85.7	88.0	88.7	34.8	35.5	37.9
	+ RFT	72.2	78.8	80.2	30.6	34.6	38.0	84.3	89.0	88.7	38.6	53.5	44.9
	negative TSE reward (ours)	+2.0	+0.1	+0.1	+6.8	+0.2	+1.2	+2.6	+5.7	+6.0	+17.1	+33.6	+23.4
LLaDA-1.5	baseline	69.8	79.4	81.1	29.0	32.4	35.4	85.3	86.3	83.3	21.5	21.1	20.7
	accuracy reward (d1)	70.6	78.5	81.4	29.6	34.2	39.2	85.0	88.0	88.3	32.8	25.8	39.5
	+ RFT	70.3	79.5	81.4	29.2	35.6	39.0	86.0	88.0	88.7	34.0	48.8	49.5
	negative TSE reward (ours)	+0.5	+0.1	+0.3	+0.2	+3.2	+3.6	+0.7	+1.7	+5.4	+12.5	+27.7	+28.8
	combining both (ours)	70.2	79.5	81.7	30.8	34.8	41.2	88.7	89.3	89.0	53.1	43.0	55.1
		+0.4	+0.1	+0.6	+1.8	+2.4	+5.8	+3.4	+3.0	+5.7	+31.6	+21.9	+34.4

4.3%, 6.6%, and 25.3% on these benchmarks, respectively, compared to the SFT baseline. These results demonstrate the effectiveness of encouraging temporally consistent answers in post-training.

Training Dynamics. We visualize the reward curves during training using LLaDA-8B-Instruct as an example, as shown in Fig. 5. The curves demonstrate a consistent upward trend in rewards across different datasets as training progresses, indicating effective learning and stable optimization.

Model Attributes After RFT. We conduct an in-depth analysis of the LLaDA-8B-Instruct model fine-tuned with the negative TSE reward alone, with a generation length of 128 tokens, evaluating its behavior after RFT across several dimensions: temporal semantic entropy, the ever pass rate, and the number of effective tokens (defined as the average count of non-padding, non-EOS tokens per generation). Our findings reveal that (1) as shown in Fig. 6a, temporal semantic entropy consistently decreases across various datasets after RFT, reflecting enhanced temporal consistency in the model’s outputs—an anticipated result of reinforcement learning; (2) Fig. 6b demonstrates that the ever pass rate remains higher than the final pass rate after RFT, suggesting there is still potential for further improvement; and (3) as depicted in Fig. 6c, the number of effective tokens reduces following RFT, implying that the fine-tuned model generates more concise responses. We hypothesize that shorter outputs may be less prone to temporal oscillations, although this relationship requires further investigation. More analysis of the model trained after temporal consistency reinforcement is provided in the Appendix D.2.

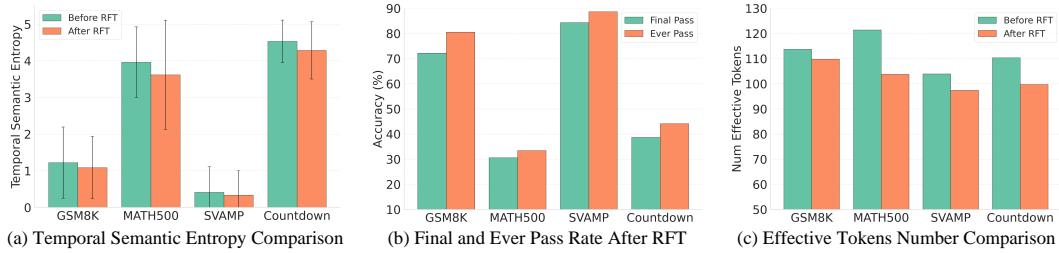


Figure 6: Model attributes after reinforcement finetuning (RFT). (a) Temporal semantic entropy decreases across datasets after RFT, indicating improved semantic consistency in model outputs. (b) The ever pass rate remains higher than the final pass rate, suggesting room for continued enhancement. (c) The number of effective tokens per generation is reduced after RFT, reflecting more concise outputs. This brevity may contribute to reduced temporal oscillations, though further analysis is needed.

6 LIMITATIONS

While our temporal self-consistency voting and post-training approach demonstrates effectiveness in many scenarios, it exhibits significant limitations when applied to tasks where the model’s intermediate predictions are consistently inaccurate. As discussed in Appendix D.3, for the Sudoku dataset, the average correctness across all intermediate generation steps remains exceedingly low (below 5%), making it difficult to reliably vote for the correct answer. Similarly, RFT relies on the model already achieving reasonably good performance to produce meaningful reward signals (Prabhudesai et al., 2025; Agarwal et al., 2025), though combining TSE with the accuracy reward may alleviate this issue to some extent. This underscores that our approach depends on the model’s inherent ability to generate correct or near-correct answers in the sampling trajectory.

7 CONCLUSION

This work uncovers a critical yet overlooked aspect of diffusion large language models: their rich temporal dynamics. By identifying temporal oscillation as a recurring issue in dLLM decoding, we challenge the convention of relying solely on final-step predictions. Our proposed methods—Temporal Self-Consistency Voting and Temporal Consistency Reinforcement—demonstrate that intermediate predictions are not noise, but signal. These strategies improve accuracy and stability without requiring additional inference passes or ground-truth supervision. Through extensive experiments, we show that temporal consistency is not just a desirable property—it’s a powerful lever for performance. We hope that this study inspires future research to *treat intermediate denoising time steps not as a nuisance, but as a feature in diffusion-based text generation*.

ACKNOWLEDGMENT

We would like to thank Muzhi Zhu, Canyu Zhao, and Linhao Zhong at Zhejiang University for their valuable discussions and insightful feedback.

REFERENCES

- Shivam Agarwal, Zimin Zhang, Lifan Yuan, Jiawei Han, and Hao Peng. The unreasonable effectiveness of entropy minimization in llm reasoning. *arXiv preprint arXiv:2505.15134*, 2025.
- Marianne Arriola, Aaron Gokaslan, Justin T Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Block diffusion: Interpolating between autoregressive and diffusion language models. *arXiv preprint arXiv:2503.09573*, 2025.
- Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. Structured denoising diffusion models in discrete state-spaces. *Advances in neural information processing systems*, 34:17981–17993, 2021.

- Pavel Avdeyev, Chenlai Shi, Yuhao Tan, Ksenia Dudnyk, and Jian Zhou. Dirichlet diffusion score model for biological sequence generation. In *International Conference on Machine Learning*, pp. 1276–1301. PMLR, 2023.
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.
- Minghan Chen, Guikun Chen, Wenguan Wang, and Yi Yang. Seed-grpo: Semantic entropy enhanced grpo for uncertainty-aware policy optimization. *arXiv preprint arXiv:2505.12346*, 2025.
- Chaoran Cheng, Jiahua Li, Jian Peng, and Ge Liu. Categorical flow matching on statistical manifolds. *Advances in Neural Information Processing Systems*, 37:54787–54819, 2024.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.
- Ganqu Cui, Yuchen Zhang, Jiacheng Chen, Lifan Yuan, Zhi Wang, Yuxin Zuo, Haozhan Li, Yuchen Fan, Huayu Chen, Weize Chen, et al. The entropy mechanism of reinforcement learning for reasoning language models. *arXiv preprint arXiv:2505.22617*, 2025.
- Mehul Damani, Isha Puri, Stewart Slocum, Idan Shenveld, Leshem Choshen, Yoon Kim, and Jacob Andreas. Beyond binary rewards: Training lms to reason about their uncertainty. *arXiv preprint arXiv:2507.16806*, 2025.
- Oscar Davis, Samuel Kessler, Mircea Petrache, Ismail Ceylan, Michael Bronstein, and Joey Bose. Fisher flow matching for generative modeling over discrete data. *Advances in Neural Information Processing Systems*, 37:139054–139084, 2024.
- Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, and Yarin Gal. Detecting hallucinations in large language models using semantic entropy. *Nature*, 630(8017):625–630, 2024.
- Tilmann Gneiting and Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American statistical Association*, 102(477):359–378, 2007.
- Shanshan Gong, Ruixiang Zhang, Huangjie Zheng, Jiatao Gu, Navdeep Jaitly, Lingpeng Kong, and Yizhe Zhang. Diffucoder: Understanding and improving masked diffusion models for code generation. *arXiv preprint arXiv:2506.20639*, 2025.
- Ishaan Gulrajani and Tatsunori B Hashimoto. Likelihood-based diffusion language models. *Advances in Neural Information Processing Systems*, 36:16693–16715, 2023.
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.
- Xiaochuang Han, Sachin Kumar, and Yulia Tsvetkov. Ssd-lm: Semi-autoregressive simplex-based diffusion language model for text generation and modular control. *arXiv preprint arXiv:2210.17432*, 2022.
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *NeurIPS*, 2021.
- Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.
- Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *Advances in neural information processing systems*, 35:8633–8646, 2022.
- Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

- Wenxuan Huang, Bohan Jia, Zijie Zhai, Shaosheng Cao, Zheyu Ye, Fei Zhao, Zhe Xu, Yao Hu, and Shaohui Lin. Vision-r1: Incentivizing reasoning capability in multimodal large language models. *arXiv preprint arXiv:2503.06749*, 2025.
- Jaeyeon Kim, Kulin Shah, Vasilis Kontonis, Sham Kakade, and Sitan Chen. Train for the worst, plan for the best: Understanding token ordering in masked diffusions. *arXiv preprint arXiv:2502.06768*, 2025.
- Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. *arXiv preprint arXiv:2302.09664*, 2023.
- Xiang Li, John Thickstun, Ishaan Gulrajani, Percy S Liang, and Tatsunori B Hashimoto. Diffusion-lm improves controllable text generation. *Advances in neural information processing systems*, 35: 4328–4343, 2022.
- Yuetai Li, Zhangchen Xu, Fengqing Jiang, Bhaskar Ramasubramanian, Luyao Niu, Bill Yuchen Lin, Xiang Yue, and Radha Poovendran. Temporal sampling for forgotten reasoning in llms. *arXiv preprint arXiv:2505.20196*, 2025.
- 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*, 2023.
- Yexiang Liu, Zekun Li, Zhi Fang, Nan Xu, Ran He, and Tieniu Tan. Rethinking the role of prompting strategies in llm test-time scaling: A perspective of probability theory. *arXiv preprint arXiv:2505.10981*, 2025a.
- Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025b.
- Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. *arXiv preprint arXiv:2310.16834*, 2023.
- Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594, 2023.
- Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*, 2025.
- Shen Nie, Fengqi Zhu, Chao Du, Tianyu Pang, Qian Liu, Guangtao Zeng, Min Lin, and Chongxuan Li. Scaling up masked diffusion models on text. *arXiv preprint arXiv:2410.18514*, 2024.
- 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. *arXiv preprint arXiv:2502.09992*, 2025.
- 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.
- Arkil Patel, Satwik Bhattacharya, and Navin Goyal. Are nlp models really able to solve simple math word problems? *arXiv preprint arXiv:2103.07191*, 2021.
- Mihir Prabhudesai, Lili Chen, Alex Ippoliti, Katerina Fragkiadaki, Hao Liu, and Deepak Pathak. Maximizing confidence alone improves reasoning. *arXiv preprint arXiv:2505.22660*, 2025.
- Zhangyang Qi, Zhixiong Zhang, Yizhou Yu, Jiaqi Wang, and Hengshuang Zhao. Vln-r1: Vision-language navigation via reinforcement fine-tuning. *arXiv preprint arXiv:2506.17221*, 2025.
- Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

- Subham Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. *Advances in Neural Information Processing Systems*, 37:130136–130184, 2024.
- Subham Sekhar Sahoo, Zhihan Yang, Yash Akhauri, Johnna Liu, Deepansha Singh, Zhoujun Cheng, Zhengzhong Liu, Eric Xing, John Thickstun, and Arash Vahdat. Esoteric language models. *arXiv preprint arXiv:2506.01928*, 2025.
- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.
- Haozhan Shen, Peng Liu, Jingcheng Li, Chunxin Fang, Yibo Ma, Jiajia Liao, Qiaoli Shen, Zilun Zhang, Kangjia Zhao, Qianqian Zhang, et al. Vlm-r1: A stable and generalizable r1-style large vision-language model. *arXiv preprint arXiv:2504.07615*, 2025.
- Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.
- Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.
- Hannes Stark, Bowen Jing, Chenyu Wang, Gabriele Corso, Bonnie Berger, Regina Barzilay, and Tommi Jaakkola. Dirichlet flow matching with applications to dna sequence design. *arXiv preprint arXiv:2402.05841*, 2024.
- Guanghan Wang, Yair Schiff, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Remasking discrete diffusion models with inference-time scaling. *arXiv preprint arXiv:2503.00307*, 2025.
- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.
- Ling Yang, Ye Tian, Bowen Li, Xinchen Zhang, Ke Shen, Yunhai Tong, and Mengdi Wang. Mmada: Multimodal large diffusion language models. *arXiv preprint arXiv:2505.15809*, 2025.
- Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.
- Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Dream 7b, 2025. URL <https://hkunlp.github.io/blog/2025/dream>.
- Qiyng Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.
- Qingyang Zhang, Haitao Wu, Changqing Zhang, Peilin Zhao, and Yatao Bian. Right question is already half the answer: Fully unsupervised llm reasoning incentivization. *arXiv preprint arXiv:2504.05812*, 2025.
- Siyuan Zhao, Devaansh Gupta, Qinqing Zheng, and Aditya Grover. d1: Scaling reasoning in diffusion large language models via reinforcement learning. *arXiv preprint arXiv:2504.12216*, 2025.
- Hao Zhong, Muzhi Zhu, Zongze Du, Zheng Huang, Canyu Zhao, Mingyu Liu, Wen Wang, Hao Chen, and Chunhua Shen. Omni-r1: Reinforcement learning for omnimodal reasoning via two-system collaboration. *arXiv preprint arXiv:2505.20256*, 2025.

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

APPENDIX

A APPENDIX OVERVIEW

This appendix provides additional implementation details, empirical analysis, and extended results to supplement the main paper. It is organized as follows:

- **Appendix B: More Implementation Details**

Provides further implementation information, including:

- Appendix B.1: Detailed descriptions of the datasets used
- Appendix B.2: Training configurations and hyperparameters
- Appendix B.3: Sampling strategies and evaluation procedures

- **Appendix C: More Analysis**

Presents extended analyses, including:

- Appendix C.1: Accuracy and entropy analysis on the MATH500 and SVAMP datasets
- Appendix C.2: Temporal semantic entropy across varying generated lengths
- Appendix C.3: Block-level token entropy analysis

- **Appendix D: More Experimental Results**

Includes additional experimental findings, such as:

- Appendix D.1: Ablation studies on different scoring rules for combining TSE with accuracy reward
- Appendix D.2: Performance of temporal self-consistency voting on reinforcement fine-tuned models
- Appendix D.3: Analysis of failure cases on the Sudoku dataset
- Appendix D.4: Detailed examples illustrating temporal oscillation

B MORE IMPLEMENTATION DETAILS

B.1 DATASETS

We provided detailed descriptions of the datasets as follows:

- GSM8K ([Cobbe et al., 2021](#)) comprises 8.5K linguistically diverse grade school math word problems (7.5K training, 1K test), solvable by bright middle school students via 2–8 steps of basic arithmetic, suited for multi-step mathematical reasoning.
- MATH500 ([Lightman et al., 2023](#)) is a curated subset of 500 problems selected from the broader MATH dataset ([Hendrycks et al., 2021](#)), featuring high-school-level competition math problems.
- SVAMP ([Patel et al., 2021](#)) serves as a benchmark for elementary-level Math Word Problems (MWPs), where each MWP is a short natural language narrative describing a scenario and asking questions about unknown quantities.
- Countdown ([Pan et al., 2025](#)) involves a combinatorial arithmetic game with three numbers, requiring models to reach target numbers using basic arithmetic operations on a given set of numbers.

B.2 TRAINING

During reinforcement fine-tuning, we train our model using sequences of 256 tokens, with a batch size of 6 per GPU and gradient accumulation over 2 steps. Low-Rank Adaptation (LoRA) ([Hu et al., 2022](#)) is applied with a rank of 128 and a scaling factor of 64. During reward computation, answers are parsed from generated text sequences for semantic clustering. When answer parsing fails due to an inaccurate format, we simply discard the answer for temporal semantic entropy computation. Moreover, since answers generated in the first half of the sampling steps tend to be rough and less reliable, we exclude them from consideration. Only answers from the second half of the sampling steps are used to calculate the temporal semantic entropy.

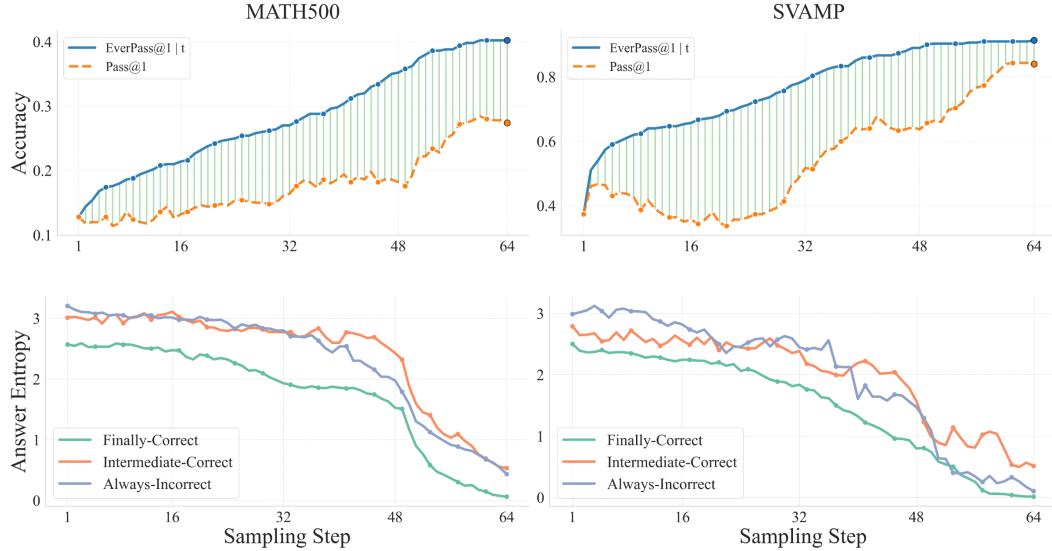


Figure S1: **Top Row:** Pass @1 and Pass @1 | t for MATH500 and SVAMP are provided as supplementary results to Fig. 2, using the same experimental settings. A noticeable gap similar to that observed in GSM8K and Countdown between Pass @1 and Pass @1 | t is present across all sampling steps. **Bottom Row:** Answer Entropy for MATH500 and SVAMP are provided as supplementary results to Fig. 3, using the same experimental setups. MATH500 and SVAMP exhibit answer entropy patterns similar to those of GSM8K and Countdown.

B.3 SAMPLING AND EVALUATION

During sampling, we adopt the semi-autoregressive sampling approach following LLaDA (Nie et al., 2025). Specifically, the sequence is split into multiple blocks, which are generated in a left-to-right manner. For each individual block, we employ the low-confidence remasking strategy during the sampling process. Following the practice in d1 (Zhao et al., 2025), we evaluate the model every 100 steps, starting from step 600 to 8,000 steps, and report the best results.

C MORE ANALYSIS

C.1 ANALYSIS ON MATH500 AND SVAMP DATASETS

Accuracy Analysis. As shown in the first row in Fig. S1, MATH500 and SVAMP exhibit a similar pattern to that observed Fig. 2 in Sec. 3.3, where a noticeable gap emerges between Pass @1 and Pass @1 | t . In MATH500 and SVAMP, correct answers start to appear early in the sampling process (near 4% and 39% at first step, respectively), and continue to improve over subsequent iterations. Interestingly, on the SVAMP dataset, a distinct pattern emerged in the model’s Pass @1 accuracy across different sampling steps. Between steps 3 and 20, performance declined noticeably, followed by a recovery after step 20. This distinctive fluctuation trajectory represents another manifestation of the temporal oscillation phenomenon.

Entropy Analysis. As shown in the second row in Fig. S1, for MATH500 and SVAMP, the answer entropy of *Finally-Correct* questions remains the lowest throughout the sampling process. *Intermediate-Correct* questions consistently exhibit lower entropy in the early sampling steps compared to *Always-Incorrect* ones, a pattern observed across all four datasets. However, unlike GSM8K, MATH500, and Countdown, where the final entropy of *Intermediate-Correct* and *Always-Incorrect* questions is similar, SVAMP displays relatively high entropy in the final sampling step for *Intermediate-Correct* questions.

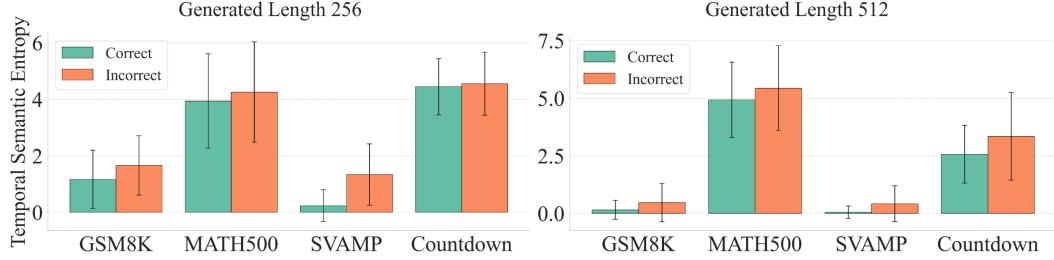


Figure S2: Temporal semantic entropy with varying generation lengths. Average temporal semantic entropy across datasets with generation lengths of 256 and 512, showing consistent patterns with Fig. 4. Higher entropy generally correlates with lower accuracy.

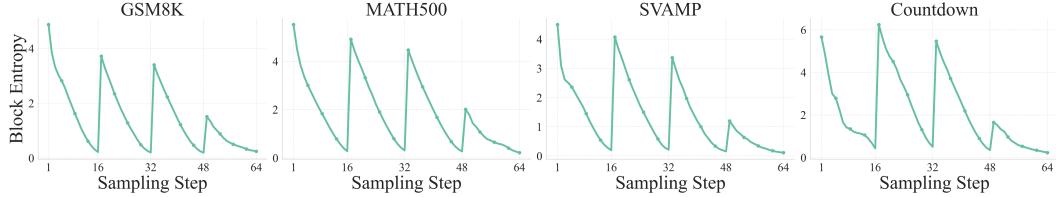


Figure S3: Block-level entropy dynamics under semi-autoregressive sampling. During sampling, sequences are partitioned into fixed-length blocks, processed in a left-to-right order, with remasking and unmasking operations restricted to the current block. Average token entropy within a block decreases with more sampling steps. A sharp entropy spike occurs when shifting to a new block, likely due to simultaneous decoding of multiple masked tokens increasing initial uncertainty.

C.2 TEMPORAL SEMANTIC ENTROPY OVER VARYING GENERATED LENGTH

To further validate the generalizability of our findings regarding temporal semantic entropy, we extended the experiments beyond those presented in Fig. 4, which used a generation length of 128 and 64 diffusion steps. Specifically, we tested generation lengths of 256 and 512, with diffusion steps set to half the generation length.

As shown in Fig. S2, the pattern of temporal semantic entropy observed here aligns with the conclusions drawn in Sec. 3.3: questions that ultimately receive incorrect answers consistently exhibit relatively high temporal semantic entropy throughout the sampling process. This high entropy reflects greater instability and uncertainty in the model’s intermediate predictions.

C.3 ANALYSIS ON THE AVERAGE ENTROPY IN BLOCKS

In addition to the token-level entropy considered in Sec. 3.3 and the answer-level entropy considered in temporal semantic entropy, we additionally introduce a block-level entropy. This is motivated by the fact that dLLMs typically adopts a semi-autoregressive sampling strategy. In this sampling strategy, the entire generated sequence is divided into multiple fixed-length blocks, with each block allocated a specific number of sampling steps. During these steps, remasking and unmasking operations are performed exclusively on the current block. Starting with the first block, the process steps to the next block only after all tokens in the current block have been unmasked.

We denote the current block as containing tokens indexed from a start s to an end $e - 1$ (inclusive), such that the block spans indices $i \in \{s, s + 1, \dots, e - 1\}$. The block entropy is then calculated as:

$$H_{\text{block}} = \frac{1}{e - s} \sum_{i=s}^{e-1} H(i)$$

where $H(i)$ represents the entropy of the i -th generated token, and $e - s$ denotes the total number of tokens in the block.

Table S1: **Ablations on scoring rules for combining TSE with accuracy reward.** We compare 4 different scoring rules, including entropy scoring, quadratic scoring, spherical scoring, and logistic scoring.

Model / Dataset		GSM8K			MATH500			SVAMP			Countdown			
		Method / Seq Len	128	256	512	128	256	512	128	256	512	128	256	512
LLaDA	baseline		70.2	78.7	80.1	23.8	34.4	36.8	81.7	83.3	82.7	21.5	19.9	21.5
	entropy		71.7	78.5	82.3	31.6	38.2	39.2	88.7	89.3	89.3	47.6	50.0	53.1
	quadratic		71.3	79.9	82.0	31.0	37.6	40.0	88.0	8.3	89.3	50.0	34.4	48.1
	logistic		73.0	79.5	81.1	31.6	36.8	39.4	87.7	87.3	90.7	46.5	37.9	51.2
LLaDA-1.5	baseline		69.8	79.4	81.1	29.0	32.4	35.4	85.3	86.3	83.3	21.5	21.1	20.7
	entropy		72.6	77.9	80.8	31.8	36.2	38.6	88.0	88.7	90.0	41.4	44.5	53.1
	quadratic		69.8	78.6	81.8	30.0	35.0	40.8	88.3	88.7	88.3	45.3	40.6	54.3
	logistic		70.1	78.5	81.7	30.6	37.2	39.0	89.0	87.7	89.3	42.2	41.1	52.7
	spherical		70.2	79.5	81.7	30.8	34.8	41.2	88.7	89.3	89.0	53.1	43.0	55.1

As illustrated in Fig. S3, the average token entropy within a single block exhibits a consistent downward trend as sampling steps accumulate. This pattern is intuitive: as more tokens in the current block are decoded, they collectively form a richer contextual foundation, thereby mitigating the model’s uncertainty about subsequent tokens in the block. Interestingly, when decoding shifts to a new block, the entropy rises sharply. This phenomenon is likely due to the need to decode multiple masked tokens simultaneously at the start of a new block, which increases the model’s uncertainty.

D MORE EXPERIMENTAL RESULTS

D.1 ABLATIONS ON SCORING RULES FOR COMBINING TSE WITH ACCURACY REWARD

In Sec. 4.2, we combine TSE with the accuracy reward using the proper scoring rule (Gneiting & Raftery, 2007). The purpose of these scoring rules is to promote truthful confidence assessments: they attain their minimum value when the predicted confidence c precisely mirrors the actual probability that the model’s output o corresponds to the correct answer o^* .

Here, we conduct an ablation study on different scoring rules. Specifically, we consider the following four forms:

$$\text{Entropy scoring: } r_i^{\text{ent}} = \mathbb{1}_{o_i=o^*} \cdot c(o_i)$$

$$\text{Quadratic scoring: } r_i^{\text{quad}} = \mathbb{1}_{o_i=o^*} - (c(o_i) - \mathbb{1}_{o_i=o^*})^2$$

$$\text{Logistic scoring: } r_i^{\text{log}} = \mathbb{1}_{o_i=o^*} + \mathbb{1}_{o_i=o^*} \log(c(o_i)) + (1 - \mathbb{1}_{o_i=o^*}) \log(1 - c(o_i))$$

$$\text{Spherical scoring: } r_i^{\text{sph}} = \mathbb{1}_{o_i=o^*} + \frac{c(o_i)}{\sqrt{(c(o_i))^2 + (1 - c(o_i))^2}}$$

All four proposed reward functions aim to jointly encourage correctness and temporal self-consistency. Notably, the r_i^{ent} function gives a reward of 0 for incorrect answers, while for correct answers, the reward is given by $c(o_i) = \frac{\mathcal{H}_{\max} - \text{TSE}(o_i)}{\mathcal{H}_{\max}}$, where $\mathcal{H}_{\max} = \log T$. This reward reaches a maximum value of 1 when all sampling steps yield the correct answer. The remaining three functions, r_i^{quad} , r_i^{log} , and r_i^{sph} , correspond to the commonly used quadratic scoring, logarithmic scoring, and spherical scoring in proper scoring rules (Gneiting & Raftery, 2007), respectively. We report the performance of all four reward combination methods in Table S1. By default, we use the spherical scoring rule because it demonstrates more superior results compared to the alternatives.

D.2 TEMPORAL SELF-CONSISTENCY VOTING AFTER RFT

It is worthwhile to investigate whether Temporal Self-consistency Voting continues to provide performance benefits after Temporal Consistency Reinforcement. To this end, we conducted

Table S2: **Performance of temporal majority voting after reinforcement learning.** Temporal self-consistency voting was applied to the model fine-tuned via temporal consistency reinforcement. The upper part is derived from the LLaDA-8B-Instruct model trained using the Negative TSE reward, whereas the lower part is based on the model trained with a combination of TSE and accuracy reward. For reference, we include the oracle EverPass @1 | t as a performance upper bound.

	GSM8K			MATH500			SVAMP			Countdown			
Method / Seq Len	128	256	512	128	256	512	128	256	512	128	256	512	
Negative TSE reward	After RFT	72.2	78.8	80.2	30.6	34.6	38.0	84.3	89.0	88.7	38.6	53.5	44.9
	Fixed Weighting	70.4	77.6	80.2	30.8	34.4	37.6	85.0	88.3	88.7	39.5	53.5	45.7
+ Temporal Voting	Linear Weighting	72.3	78.8	80.6	30.8	35.4	38.0	85.3	88.3	88.7	39.1	53.5	45.7
	Exp. Weighting	72.6	79.2	80.6	30.8	35.0	38.0	85.3	89.0	88.7	39.5	53.9	45.7
		+0.4	+0.4	+0.4	+0.2	+0.4	+0.0	+1.0	+0.0	+0.0	+0.9	+0.4	+0.8
	EverPass @1 t	80.5	81.8	81.3	33.4	37.4	40.0	88.7	90.7	89.7	44.1	59.8	55.5
TSE & accuracy reward	After RFT	72.1	80.0	83.0	31.2	35.4	41.4	85.0	90.3	92.3	41.5	42.6	54.7
	Fixed Weighting	71.2	79.6	82.8	31.2	35.6	41.2	85.3	90.7	92.3	40.6	41.5	53.2
+ Temporal Voting	Linear Weighting	73.0	81.9	83.0	31.2	36.0	41.0	86.7	90.3	92.7	40.9	42.8	54.2
	Exp. Weighting	72.6	81.1	83.3	31.6	35.8	41.4	86.3	90.7	92.3	41.5	42.7	55.1
		+0.5	+1.1	+0.3	+0.4	+0.4	+0.0	+1.3	+0.4	+0.0	+0.0	+0.1	+0.4
	EverPass @1 t	84.0	91.4	87.9	43.6	49.2	52.0	90.7	91.7	92.7	54.3	68.0	70.3

Table S3: **Performance of temporal self-consistency voting and temporal consistency reinforcement on the Sudoku dataset.** The baseline model is obtained by applying supervised fine-tuning to LLaDA-8B-Instruct using the s1K dataset.

Model	Sudoku			
	Method / Seq Len	128	256	512
baseline		12.2	6.7	5.5
+ temporal voting		12.5	6.1	2.8
accuracy reward (d1)		23.2	17.8	12.7
+ RFT	negative TSE reward (ours)	15.2	9.4	3.3
	combining both (ours)	27.5	27.8	16.6

experiments using two models: the first is derived from the LLaDA-8B-Instruct model trained with the negative TSE reward, and the second is trained using the accuracy reward combined with TSE.

We applied Temporal Self-Consistency Voting to both models, and the results are summarized in Table S2. The findings indicate that both models still exhibit performance improvements when temporal voting is applied, even after undergoing reinforcement learning. This suggests that Temporal Self-Consistency Voting and Temporal Consistency Reinforcement are complementary techniques, and can be effectively combined to further enhance model performance.

D.3 FAILURE CASE ANALYSIS

As discussed in Sec. 6, our method may rely on the model’s initial performance to achieve further improvements. To investigate this limitation, we take the challenging Sudoku dataset as a case study. As shown in Table S3, directly applying our proposed temporal self-consistency voting and temporal consistency reinforcement with the negative TSE reward alone results in a noticeable performance drop on the Sudoku dataset. For example, the original accuracy of the base model for the generation length of 512 is 5.5%, while the voting method achieves 2.8%, reflecting a decline of 2.7%.

To better understand this problem, we conduct a deeper analysis of the model’s behavior during generation. We define a metric called **Temporal Accuracy**, which quantifies the average correctness across all intermediate sampling steps in the generation process. Formally, let $e_{i,t}$ represent the correctness indicator for the i -th question at the t -th sampling step, where $e_{i,t} = 1$ if the answer is correct, and 0 otherwise. Then, the **Temporal Accuracy** for a dataset with N examples and T

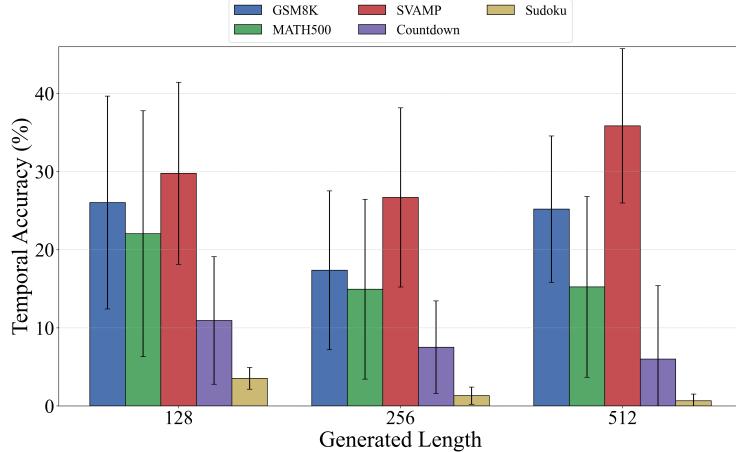


Figure S4: **Temporal accuracy across datasets.** Temporal Accuracy on Sudoku remains consistently low (all below 5%) across different settings, indicating that the model rarely generates correct answers during the sampling process. This scarcity of valid candidates severely limits the effectiveness of voting mechanisms, as there are insufficient correct outputs to reliably converge on the correct answer.

sampling steps is computed as:

$$\text{TemporalAccuracy} = \frac{1}{N \cdot T} \sum_{i=1}^N \sum_{t=1}^T e_{i,t}. \quad (\text{S1})$$

As shown in Fig. S4, the **Temporal Accuracy** of the Sudoku is much lower than the other 4 datasets, where temporal self-consistency voting proved effective. On the Sudoku dataset, we observe that the Temporal Accuracy remains exceedingly low—below 5% on average—across all intermediate steps. This low signal makes it difficult for temporal voting mechanisms to reliably identify the correct answer and for reinforcement learning signals to guide the model effectively.

Interestingly, while RFT with only negative TSE reward leads to poorer results, combining TSE with accuracy reward can achieve better performance than using accuracy reward alone. We hypothesize that this is because the integration of TSE allows the model to receive more fine-grained rewards, rather than just binary outcomes of correct or incorrect.

D.4 EXAMPLES OF TIME OSCILLATION

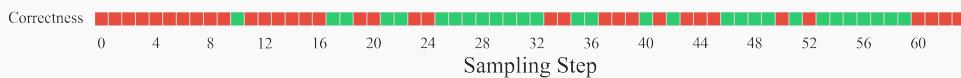
We present representative examples of temporal oscillation from GSM8K, using LLaDA-8B-Instruct as the backbone model. Note that all examples shown below were ultimately incorrect but were classified as correct answers by our voting method, as described in Sec. 4.1. Correct answers are in green, incorrect answers in red. We use blue to highlight the key segments in each example.

We observe that many questions become incorrect during temporal oscillation due to flawed reasoning paths. Although the model may initially provide the correct answer and follow a logical reasoning path, an incorrect trajectory can ultimately lead to an erroneous response.

A notable example is seen in Example 3 below. Initially, the model correctly calculates the total cups, per-hour revenue, and the necessary per-hour cost deduction to derive the right profit per hour (step 54). However, during the reasoning process, it introduces erroneous masking regarding key calculation elements—the per-hour cost to subtract. Even though the answer remains correct at first (step 61), the flawed reasoning trajectory eventually results in miscalculations and an incorrect final response (step 62).

Example 1

Problem: Ruby is 6 times older than Sam. In 9 years, Ruby will be 3 times as old as Sam. How old is Sam now?

**Step 59:**

<reasoning>

Let Sam's age be x years. Then Ruby's age is $6x$ years. In 9 years, Ruby's age will be $6x + 9$ and Sam's age will be $x + 9$. According to the problem, in 9 years, Ruby will be 3 times as old as Sam, so we have the equation $6x + 9 = 3(x + 9)$. Simplifying this equation, we get $3x = 0$ so $x = 3$.

</reasoning>

<answer>

6

</answer>

Step 60:

<reasoning>

Let Sam's age be x years. Then Ruby's age is $6x$ years. In 9 years, Ruby's age will be $6x + 9$ and Sam's age will be $x + 9$. According to the problem, in 9 years, Ruby will be 3 times as old as Sam, so we have the equation $6x + 9 = 3(x + 9)$. Simplifying this equation, we get $3x = 0 \Rightarrow x = 0$

</reasoning>

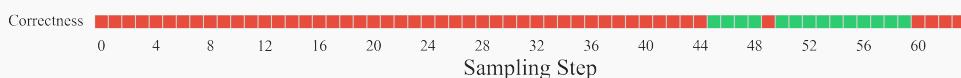
<answer>

0

</answer>

Example 2

Problem: Shiela bought five cell phones for \$150 each for a 3-month installment. A 2% interest will be charged for each unit. How much will Shiela pay each month for 3 months

**Step 59:**

<reasoning>

Shiela bought five cell phones for \$150 each, so the total cost is $5 \times 150 = \$750$. The interest for each unit is 2%, so the total interest is $\$750 \times 0.02 = \15 . Since Shiela will pay this amount over 3 months, the monthly payment is $\$750 + \$15 = \$750 / 3 = \250 .

</reasoning>

<answer>

255

</answer>

Step 60:

<reasoning>

Shiela bought five cell phones for \$150 each, so the total cost is $5 \times 150 = \$750$. The interest for each unit is 2%, so the total interest is $\$750 \times 0.02 = \15 . Since Shiela will pay this amount over 3 months, the monthly payment is $\$750 + \$15 = \$750 / 3 = \250 .

</reasoning>

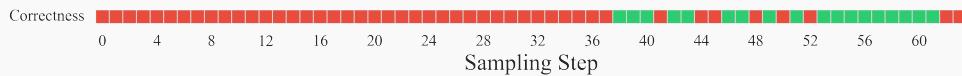
<answer>

250

</answer>

Example 3

Problem: Millie decides to open a lemonade stand. She spends \$18 to buy enough supplies to make 3 pitchers of lemonade. Each pitcher holds 12 cups of lemonade. She sells each cup of lemonade for \$1. She sells an average of 4 cups per hour that her lemonade stand is open. If Millie sells all of the lemonade, how much profit will she make per hour that she spends running the lemonade stand?

**Step 54:**

<reasoning>

Millie spends \$18 on supplies to make 3 pitchers of lemonade. Each pitcher holds 12 cups, so she has a total of $3 * 12 = 36$ cups of lemonade. She sells each cup for \$1 and sells an average of 4 cups per hour. Therefore, her revenue per hour is $4 * \$1 = \4 . To find per hour profit, we subtract the cost of supplies from the revenue: $\$4 - \$2 = \$2$ per hour.

</reasoning>

<answer>

2

</answer>

...

Step 61:

<reasoning>

Millie spends \$18 on supplies to make 3 pitchers of lemonade. Each pitcher holds 12 cups, so she has a total of $3 * 12 = 36$ cups of lemonade. She sells each cup for \$1 and sells an average of 4 cups per hour. Therefore, her revenue per hour is $4 * \$1 = \4 . To find per hour profit, we subtract the cost of supplies from the revenue: $\$4 - \$3 = \$2$ per hour.

</reasoning>

<answer>

2

</answer>

Step 62:

<reasoning>

Millie spends \$18 on supplies to make 3 pitchers of lemonade. Each pitcher holds 12 cups, so she has a total of $3 * 12 = 36$ cups of lemonade. She sells each cup for \$1 and sells an average of 4 cups per hour. Therefore, her revenue per hour is $4 * \$1 = \4 . To find per hour profit, we subtract the cost of supplies from the revenue: $\$4 - \$3 = \$1$ per hour.

</reasoning>

<answer>

1

</answer>