

000 001 002 003 004 005 TIME IS A FEATURE: EXPLOITING TEMPORAL DY- 006 NAMICS IN DIFFUSION LANGUAGE MODELS 007 008 009

010 **Anonymous authors**
011 Paper under double-blind review
012
013
014
015
016
017
018
019
020
021
022
023
024
025
026
027

ABSTRACT

028 Diffusion large language models (dLLMs) generate text through iterative denoising,
029 yet current decoding strategies discard rich intermediate predictions in favor
030 of the final output. Our work here reveals a critical phenomenon, **temporal**
031 **oscillation**, where correct answers often emerge in the middle process, but are
032 overwritten in later denoising steps. To address this issue, we introduce two
033 complementary methods that exploit temporal consistency: 1) **Temporal Self-**
034 **Consistency Voting**, a training-free, test-time decoding strategy that aggregates
035 predictions across denoising steps to select the most consistent output; and 2)
036 a post-training method termed **Temporal Consistency Reinforcement**, which
037 uses Temporal Semantic Entropy (TSE), a measure of semantic stability across
038 intermediate predictions, as a reward signal to encourage stable generations.
039 Empirical results across multiple benchmarks demonstrate the effectiveness of our
040 approach. Using the negative TSE reward alone, we observe a remarkable average
041 improvement of **24.7%** on the Countdown dataset over an existing dLLM. Com-
042 bined with the accuracy reward, we achieve absolute gains of **2.0%** on GSM8K,
043 **4.3%** on MATH500, **6.6%** on SVAMP, and **25.3%** on Countdown, respectively.
044 Our findings underscore the untapped potential of temporal dynamics in dLLMs
045 and offer two simple yet effective tools to harness them.
046
047
048
049
050
051
052
053

1 INTRODUCTION

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

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

068 As depicted in Fig. 1, we analyze two key metrics across four widely used benchmark datasets using
069 two representative models: LLaDA-8B-Instruct (Nie et al., 2025) and LLaDA-1.5 (Zhu et al., 2025).
070 The first metric, final pass rate, measures the accuracy of the final output, while the second, ever-pass
071 rate, captures whether a correct answer appears at any point during the decoding process. In Fig. 1a,
072 a consistent and significant discrepancy exists between these metrics. This gap reveals a critical
073 phenomenon: models often generate correct answers during intermediate steps but subsequently
074 overwrite them with incorrect ones. Fig. 1b illustrates this concretely—in a math problem, the
075 model produces the correct answer “25” at sampling step 55, only to replace it with an incorrect “2”
076 by the final step 64. More examples on temporal oscillation are presented in Appendix E.6.

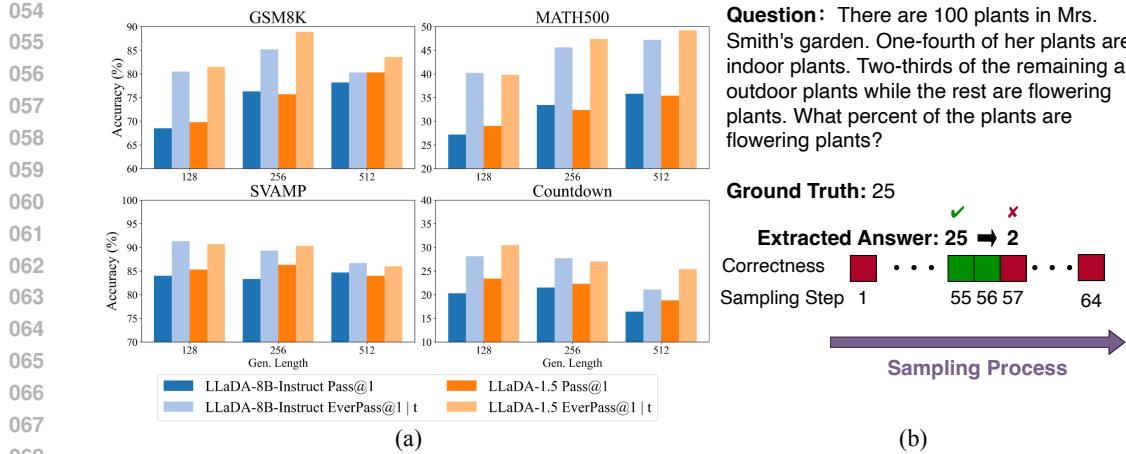


Figure 1: **Illustration of temporal oscillation during sampling.** (a) Across four datasets, a significant gap is observed between the **final answer’s pass rate** (denoted as Pass@1) and the **ever-pass rate at any intermediate step** (denoted as EverPass@1 | t). This gap reveals the phenomenon we 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. More examples with detailed outputs can be found in Appendix E.6.

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 multiple denoising steps and selects the most temporally consistent output. This simple yet effective method improves accuracy while introducing only negligible computational overhead, making it practical for real-world deployment.
- **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 explicitly. Notably, leveraging negative TSE as the reward enables reliable performance improvements without requiring ground-truth labels for reward computation. Furthermore, when ground-truth labels are available, combining them with TSE-based rewards can provide richer and more complementary supervision, ultimately leading to even greater and more robust improvements in generation quality.

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.

108

2 RELATED WORK

109

2.1 DIFFUSION LANGUAGE MODELS

110 Building on the success of diffusion in image and video generation (Song et al., 2020; Ho et al., 2020; 2022), diffusion methods have been extended to text. Early continuous approaches (Han et al., 2022; Li et al., 2022) operate in continuous space, while others map text to the probability simplex (Avdeyev et al., 2023; Stark et al., 2024). 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.

111 Discrete diffusion models, pioneered by D3PM (Austin et al., 2021), advanced through masked token frameworks (Shi et al., 2024; Sahoo et al., 2024; Nie et al., 2024) and scaling efforts. 112 Lightweight variants like Plaid (Gulrajani & Hashimoto, 2023) and SEDD (Lou et al., 2023) rival 113 GPT-2 (Radford et al., 2019), yet lag autoregressive models in scalability. To bridge this gap, BD3- 114 LMs (Arriola et al., 2025) and Eso-LMs (Sahoo et al., 2025) interpolate between autoregressive 115 and diffusion paradigms, enabling parallel sampling with competitive performance. Recent efforts 116 scale up dLLMs: Dream (Ye et al., 2025) converts pretrained autoregressive models into diffusion 117 models, while LLaDA (Nie et al., 2025) trains strong models from scratch. Another line of work 118 studies sampling in dLLMs. For instance, Kim et al. (2025) show token ordering affects performance 119 and propose adaptive inference, while ReMDM (Wang et al., 2025) uses inference-time remasking 120 to boost generation. **Rather than modifying local sampling heuristics, we study dLLMs through the** 121 **underexplored lens of temporal stability across the entire denoising trajectory.**

122 Several concurrent work also investigates temporal behaviors in diffusion language models. Li et al. 123 (2025a) show that dLLMs often generate correct intermediate answers that are later overwritten, and 124 propose stopping generation early when such signals appear. He et al. (2025) further demonstrate 125 that temporal cues along the denoising trajectory can be directly exploited to improve reasoning. 126 Xie et al. (2025) introduce a step-aware RL method that aligns denoising steps with hierarchical 127 reasoning to avoid unstructured refinement. While these studies highlight the value of early- 128 step signals or step-specific structure, our work takes a complementary direction by developing 129 both a lightweight test-time voting method and an unsupervised reinforcement signal grounded in 130 trajectory-level temporal consistency.

131

2.2 TEST-TIME STRATEGY

132 Test-time strategies (Wei et al., 2022; Madaan et al., 2023; Snell et al., 2024; Yao et al., 2023; Liu et al., 2025a) are widely used to improve LLM accuracy, consistency, and reliability. A simple 133 yet effective method is Self-Consistency (Wang et al., 2022), which selects the most consistent 134 answer from multiple outputs via majority voting. Building on this idea, we propose a temporal self- 135 consistency strategy tailored for dLLMs, which adds negligible inference overhead and integrates 136 seamlessly into existing frameworks. Another important technique is semantic entropy (Farquhar 137 et al., 2024; Kuhn et al., 2023), an uncertainty metric that clusters semantically equivalent outputs 138 before computing entropy. While previously applied to uncertainty estimation and hallucination 139 detection, we extend it to dLLMs by introducing Temporal Semantic Entropy, capturing stability 140 and confidence throughout the denoising process.

141

2.3 POST-TRAINING USING REINFORCEMENT LEARNING

142 Group Relative Policy Optimization (Shao et al., 2024; Guo et al., 2025) (GRPO), a variant of 143 Proximal Policy Optimization (Schulman et al., 2017), computes advantages directly from group 144 rewards, removing the need for a separately trained value function. GRPO has shown strong 145 performance in reasoning tasks like mathematics and code generation, and promising results across 146 broader modalities (Huang et al., 2025; Shen et al., 2025; Qi et al., 2025; Zhong et al., 2025; Li et al., 147 2025b; Damani et al., 2025). Building on this, refinements such as DAPO (Yu et al., 2025) introduce 148 dynamic sampling to balance training batches, while entropy-based methods (Zhang et al., 2025; 149 Prabhudesai et al., 2025; Cui et al., 2025; Agarwal et al., 2025) further enhance RL. For example, 150 EMPO (Zhang et al., 2025) derives rewards from semantic entropy, and Seed-GRPO (Chen et al., 151 2025) improves advantage estimation. Recent adaptations to dLLMs include *diffu*-GRPO (Zhao 152 et al., 2025), UniGRPO (Yang et al., 2025), and coupled-GRPO (Gong et al., 2025), which still 153

162 rely on ground-truth rewards. By contrast, our method is fully unsupervised, enhancing temporal
 163 consistency without ground-truth supervision.
 164

165 3 EXPLORATIONS ON DLLMs

166 3.1 PRELIMINARIES ON DLLMs

169 DLLMs formulate text generation as a process of iteratively denoising text sequences across
 170 different time steps. Let $\mathbf{x}_0 \sim p_{\text{data}}(\mathbf{x}_0)$ denote the original clean input sequence. For each diffusion
 171 step $t \in [0, T]$, let $\mathbf{x}_t \in \mathcal{V}^L$ denote the corresponding noisy token-sequence tensor of length L . The
 172 noisy sequence \mathbf{x}_t is generated via a masking-based forward corruption process, in which a subset
 173 of tokens is stochastically masked at each step. The forward noising process is defined as a Markov
 174 chain $q(\mathbf{x}_{1:T} \mid \mathbf{x}_0) = \prod_{t=1}^T q(\mathbf{x}_t \mid \mathbf{x}_{t-1})$, which progressively adds noise to \mathbf{x}_0 over time steps.
 175 This process incrementally transforms the clean sequence \mathbf{x}_0 into a highly noisy version \mathbf{x}_T through
 176 a series of conditional transitions. In contrast, the reverse (generative) process is modeled as:

$$177 \quad p_{\theta}(\mathbf{x}_{0:T}) = p_{\theta}(\mathbf{x}_T) \prod_{t=1}^T p_{\theta}(\mathbf{x}_{t-1} \mid \mathbf{x}_t) = p_{\theta}(\mathbf{x}_T) \prod_{t=1}^T [\sum_{\mathbf{x}_0^t} q(\mathbf{x}_{t-1} \mid \mathbf{x}_0^t) p_{\theta}(\mathbf{x}_0^t \mid \mathbf{x}_t)], \quad (1)$$

179 where the summation $\sum_{\mathbf{x}_0^t}$ enumerates all possible decoded clean sequences \mathbf{x}_0^t at step t , making
 180 explicit that the sampling procedure first predicts a clean intermediate sequence and then re-applies
 181 noise via $q(\mathbf{x}_{t-1} \mid \mathbf{x}_0^t)$.
 182

183 This decomposition reveals that each reverse step consists of two sub-operations aligned with the
 184 marginalization in Eq. (1). First, the model predicts a clean latent sequence through $p_{\theta}(\mathbf{x}_0^t \mid \mathbf{x}_t)$,
 185 yielding an intermediate hypothesis \mathbf{x}_0^t . Second, the next state \mathbf{x}_{t-1} is generated by sampling from
 186 $q(\mathbf{x}_{t-1} \mid \mathbf{x}_0^t)$, which re-applies noise to \mathbf{x}_0^t following the forward corruption process. This remasking
 187 mechanism is realized in practice through strategies such as random or low-confidence re-masking,
 188 as adopted in (Nie et al., 2025).
 189

190 3.2 TEMPORAL OSCILLATION

191 In the reverse process of DLLMs, predictions $p_{\theta}(\mathbf{x}_0^t \mid \mathbf{x}_t)$ at a single step are often inaccurate,
 192 especially under high noise when t is large. Existing models perform iterative denoising according
 193 to the generative process framework, where the final output is determined by the prediction
 194 $\mathbf{x}_0^1 \sim p_{\theta}(\mathbf{x}_0^1 \mid \mathbf{x}_1)$ at the last denoising step, while neglecting all intermediate predictions
 195 $\{\mathbf{x}_0^t \sim p_{\theta}(\mathbf{x}_0^t \mid \mathbf{x}_t)\}_{t=2}^T$ generated during the iterative process. In this work, we conduct an in-depth
 196 investigation into these intermediate-step results, revealing a critical phenomenon in diffusion-based
 197 text generation.
 198

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

$$204 \quad \text{EverPass@1} | t = \mathbb{E}_i \left\{ \max_{k \in \{1, \dots, t\}} e_{i,k} \right\} \quad (2)$$

205 This metric captures the cumulative correctness across all sampling steps, reflecting the overall
 206 fraction of questions for which the model arrives at a correct solution at least once, even if that
 207 solution is later discarded in the final output.
 208

209 **Experiment Setup.** We compare the final pass rate, *i.e.*, Pass@1 at the last step with EverPass@1 |
 210 t on two representative DLLMs: LLaDA-8B-Instruct and LLaDA-1.5, evaluated across different
 211 answer lengths and four reasoning benchmarks: GSM8K (Cobbe et al., 2021), MATH500 (Lightman
 212 et al., 2023), SVAMP (Patel et al., 2021), and Countdown (Pan et al., 2025).
 213

214 **Observations.** As shown in Fig. 1 and Table 1, there is a notable gap between the final pass rate and
 215 the ever pass rate. For instance, on GSM8K with length 128, LLaDA-8B-Instruct achieves 68.5%
 216 final pass rate versus 80.5% ever pass rate, a gap of 12.0%. This gap shows that many questions
 217 are correctly solved at intermediate steps but later revised to incorrect answers during refinement.
 218

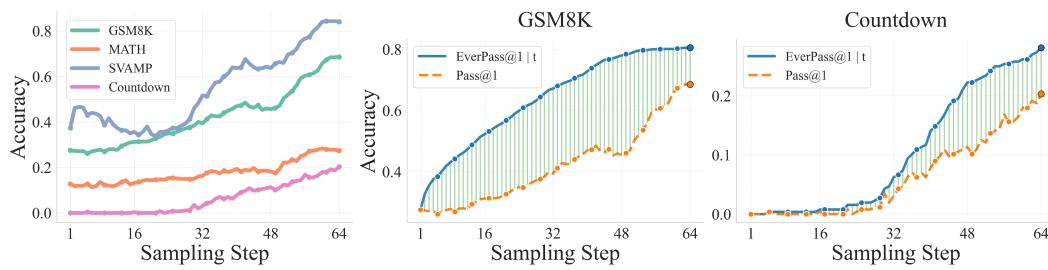


Figure 2: **Patterns of accuracy evolution over diffusion sampling steps.** Responses of length 128 are generated with 64 steps using LLaDA-8B-Instruct. **Left:** Accuracy generally rises with more steps across datasets; SVAMP starts high, while harder ones like Countdown start low but improve steadily. **Middle/Right:** We compare the final pass rate, Pass @1, with cumulative EverPass @1 | t over steps. A clear gap persists between them, shown by the green shaded area.

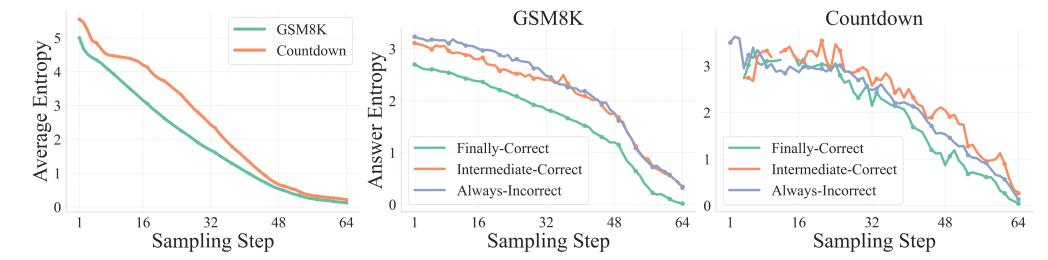


Figure 3: **Patterns of entropy evolution over diffusion sampling steps.** Responses are generated with length 128 using 64 diffusion steps from the LLaDA-8B-Instruct model. **Left:** Average token-level entropy decreases steadily during sampling. GSM8K shows lower entropy than Countdown, aligning with its higher accuracy. **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.

It reveals an instability in iterative decoding, where correct paths can be overwritten as generation proceeds. We term this phenomenon *temporal oscillation*, with examples in Appendix E.6.

Takeaway 1: Correct intermediate answers may be 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: accuracy, entropy, and semantic stability across decoding steps.

Accuracy Across Sampling Steps. As shown in Fig. 2a, accuracy generally improves with more sampling steps. Simpler datasets like SVAMP start high and remain stable, while harder ones like Countdown begin low but benefit from iterative refinement. To probe further, we compare Pass @1 and EverPass @1 | t on GSM8K and Countdown. Early correct predictions appear sooner on GSM8K but later on Countdown, and a growing gap between the two metrics reveals that early correctness does not ensure stable reasoning. This underscores the importance of preserving correct intermediate states. Additional results for SVAMP and MATH500 are in Appendix C.1.

Entropy Across Sampling Steps. Temporal oscillations reflect model uncertainty. To quantify this, we analyze the average token-level entropy. As shown in Fig. 3, entropy decreases steadily and approaches zero by the final step, with GSM8K starting lower, indicating higher initial

confidence. We categorize questions into three groups: Finally-Correct, Always-Incorrect, and Intermediate-Correct, where Intermediate-Correct means at least one intermediate step is correct but the final answer is incorrect. For GSM8K and Countdown, incorrect final answers show consistently higher entropy. On GSM8K, Intermediate-Correct questions begin with lower entropy than Always-Incorrect, suggesting initially confident but unstable predictions. Considering the semi-autoregressive sampling strategy (Nie et al., 2025), we also measure the average entropy of the currently generated block, detailed in Appendix C.3.

Temporal Semantic Entropy. **Temporal Semantic Entropy.** Token-level entropy reflects local uncertainty, but we also need a measure of semantic consistency across the decoding trajectory. We therefore introduce *Temporal Semantic Entropy* (TSE), which captures the semantic variations of answers during sampling. During decoding, we obtain a sequence of T intermediate answers, denoted by $\{\mathbf{x}_0^t\}_{t=1}^T$. We cluster them by semantic meaning into $\mathcal{C} = \{C_1, \dots, C_K\}$, where each cluster C_k groups answers with equivalent semantics. We define the probability mass of a semantic cluster as $P(C_k) = \sum_{\mathbf{x}_0^t} p(\mathbf{x}_0^t \in C_k) = |C_k|/T$, where $|C_k|$ counts how many intermediate answers fall into cluster C_k . Based on this, the TSE of a sampling trajectory is defined as:

$$\text{TSE}(\{\mathbf{x}_0^t\}_{t=1}^T) = - \sum_{k=1}^K P(C_k) \log P(C_k), \quad (3)$$

which quantifies the uncertainty in semantic content across steps: higher TSE indicates more semantic variation, while lower TSE implies convergence to a consistent meaning.

As shown in Fig. 4, TSE offers insight into model behavior during generation. In datasets like Countdown and MATH, where performance is weaker, we observe higher entropy than in GSM8K and SVAMP, reflecting greater semantic instability. Moreover, questions ultimately answered correctly generally exhibit lower entropy than incorrect ones, including both “finally correct” and “intermediate correct” cases, indicating that stable, semantically consistent trajectories align with better performance. Thus, high TSE may signal model uncertainty and highlight samples for further improvement. More results on TSE are provided in Appendix C.2.

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

We propose a temporal self-consistency decoding strategy for dLLMs, which leverages intermediate predictions to improve final outputs. As discussed in Sec. 3.3, while the last timestep usually yields the best result, the correct answer may also appear earlier, so relying only on the final prediction risks discarding better outputs. To address this, we aggregate predictions across timesteps using a weighted voting mechanism. Formally, given a diffusion sampling trajectory $\{\mathbf{x}_0^t\}_{t=1}^T$, our method selects the final answer a^* according to a weighted vote over all timesteps:

$$a^* = \arg \max_a \sum_{t=1}^T f(t) \cdot \mathbb{1}(\text{meaning } (\mathbf{x}_0^t) = a). \quad (4)$$

324 Here, $\mathbb{1}(\cdot)$ indicates whether \mathbf{x}_0^t decodes to a , and $f(t)$ is a timestep weighting function. Since
 325 accuracy generally increases with later steps, we design $f(t)$ as a monotonically decreasing function
 326 of diffusion step. We experiment with constant, linear, and exponential weighting, as in Sec. 5.2.
 327

328 **Discussion.** Our method is conceptually related to self-consistency decoding (Wang et al.,
 329 2022), which improves reasoning in autoregressive LLMs by sampling diverse reasoning paths
 330 and selecting the most consistent answer via majority voting. However, self-consistency requires
 331 multiple full-length forward passes with high cost. In contrast, our approach requires only a single
 332 sampling trajectory. By exploiting the temporal nature of diffusion inference indicated by Eq. (1),
 333 we obtain a series of intermediate predictions without additional model evaluations. This makes our
 334 method both efficient and effective for boosting accuracy through temporal aggregation.
 335

336 4.2 TEMPORAL CONSISTENCY REINFORCEMENT

337 Motivated by our observation in Sec. 3.3 that correct answers generally exhibit lower Temporal
 338 Semantic Entropy (TSE) than incorrect ones, reflecting stronger semantic consistency over time,
 339 we propose a post-training approach designed to encourage temporal consistency in model outputs.
 340 Specifically, we adopt Group Relative Policy Optimization (GRPO) (Shao et al., 2024; Guo et al.,
 341 2025) as our reinforcement learning framework and use TSE as a self-supervised reward signal.
 342

343 **Negative TSE as the Reward.** Following GRPO, for each question q sampled from the dataset
 344 \mathcal{D} , we draw a group of G responses $\{\mathbf{o}_1, \mathbf{o}_2, \dots, \mathbf{o}_G\}$ from the old policy $\pi_{\theta_{\text{old}}}$. Each response
 345 \mathbf{o}_i receives a scalar reward $r_i = -\text{TSE}(\mathbf{o}_i)$, where $\text{TSE}(\mathbf{o}_i)$ is computed using Eq. (3) from
 346 Sec. 3.3. This reward encourages the model to produce responses whose intermediate predictions
 347 remain semantically consistent throughout the decoding process. Based on this, we define the
 348 unnormalized advantage (Liu et al., 2025b) for all tokens $k = 1, \dots, |\mathbf{o}_i|$ as $A_i^k(\pi) = r_i(\pi) -$
 349 $\text{mean}(\{r_j(\pi)\}_{j=1}^G)$. The training objective follows the standard GRPO formulation with our TSE-
 350 based reward. **We optimize the policy model by maximizing the following objective:**

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{\mathbf{o}_1, \dots, \mathbf{o}_G \sim \pi_{\theta_{\text{old}}}(\cdot|q)} \left[\left(\frac{1}{G} \sum_{i=1}^G \frac{1}{|\mathbf{o}_i|} \sum_{k=1}^{|\mathbf{o}_i|} \min(\rho_i^k A_i^k, \text{clip}(\rho_i^k, 1 - \varepsilon, 1 + \varepsilon) A_i^k) \right) - \beta D_{\text{KL}}[\pi_{\theta}(\cdot|q) \parallel \pi_{\text{ref}}(\cdot|q)] \right], \quad (5)$$

351 where π_{ref} is the reference policy, $\rho_i^k = \frac{\pi_{\theta}(\mathbf{o}_i^k|q)}{\pi_{\theta_{\text{old}}}(\mathbf{o}_i^k|q)}$ is the importance sampling ratio, ε is the clipping
 352 threshold, and β controls the strength of the KL penalty. To compute the token-level probabilities
 353 used in ρ_i^k , we follow the *diffu*-GRPO method (Zhao et al., 2025), which estimates these probabilities
 354 by averaging outputs from multiple randomly masked versions of the prompt.
 355

356 **Combining TSE with Accuracy Reward.** When ground-truth answers are available during
 357 training, we combine TSE with an accuracy reward to further enhance performance. Specifically,
 358 We design a composite reward function that integrates correctness and temporal consistency.
 359

360 The accuracy reward follows a binary scheme: it assigns 1 when the model’s prediction is correct
 361 ($\mathbf{o}_i = \mathbf{o}^*$), and 0 otherwise. To measure temporal consistency, we derive a normalized confidence
 362 score from TSE: $c(\mathbf{o}_i) = \frac{\mathcal{H}_{\text{max}} - \text{TSE}(\mathbf{o}_i)}{\mathcal{H}_{\text{max}}}$, $\mathcal{H}_{\text{max}} = \log T$. Here, T denotes the total sampling steps.
 363 This normalization yields $c \in [0, 1]$, with larger values indicating stronger consistency.
 364

365 We further transform $c(\mathbf{o}_i)$ using the spherical scoring rule (Gneiting & Raftery, 2007), which has
 366 shown strong empirical performance across tasks. The final reward is defined as:
 367

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

368 Here, the first term enforces correctness while the second term encourages higher temporal
 369 consistency. This formulation makes the reward sensitive not only to prediction accuracy, but also
 370 to the stability of the underlying generation process. A detailed comparison of various scoring rules
 371 and their empirical performance in the Appendix E.4.
 372

373 **Discussion.** Unlike prior reinforcement learning post-training methods for dLLMs, such as *diffu*-
 374 GRPO, which relies on ground-truth answers for reward computation, our approach operates without
 375 any labeled data. Instead, we harness the model’s internal temporal dynamics as a self-supervised
 376 signal, employing negative TSE to assess answer quality. This enables our method to be broadly
 377

378
379
380
381
382Table 1: **Performance of temporal majority voting.** We compare three strategies: fixed, linear, and exponential weighting, on four datasets using LLaDA-8B-Instruct and LLaDA-1.5. Bold numbers mark group bests, and green values show gains over baseline. For reference, we report the oracle EverPass @1 | t as an upper bound.

Method / Seq Len	GSM8K			MATH500			SVAMP			Countdown			
	128	256	512	128	256	512	128	256	512	128	256	512	
LLaDA-8B-Instruct	baseline	68.3 ± 0.3	76.4 ± 0.3	78.1 ± 0.2	27.6 ± 0.4	33.2 ± 0.3	35.8 ± 0.4	84.3 ± 0.2	83.0 ± 0.3	84.3 ± 0.2	20.4 ± 0.4	21.4 ± 0.4	16.5 ± 0.5
Fixed Weighting	68.0 ± 0.2	73.4 ± 0.2	78.3 ± 0.2	26.6 ± 0.2	30.8 ± 0.2	34.2 ± 0.2	87.0 ± 0.2	84.3 ± 0.2	84.3 ± 0.2	22.7 ± 0.2	18.8 ± 0.2	11.3 ± 0.2	
+ Temporal Voting	Linear Weighting	70.0 ± 0.2	78.0 ± 0.2	78.8 ± 0.2	28.0 ± 0.2	34.4 ± 0.2	34.6 ± 0.2	87.0 ± 0.2	84.3 ± 0.2	84.3 ± 0.2	24.2 ± 0.2	21.9 ± 0.2	16.0 ± 0.2
	Exp. Weighting	70.1 ± 0.2	78.8 ± 0.2	78.9 ± 0.2	28.6 ± 0.2	35.2 ± 0.2	36.2 ± 0.2	86.3 ± 0.2	84.0 ± 0.2	84.7 ± 0.2	24.9 ± 0.2	23.3 ± 0.2	16.5 ± 0.2
		+1.8	+2.4	+0.8	+1.0	+2.0	+0.4	+2.0	+1.0	+0.4	+4.5	+1.9	+0.0
	EverPass @1 t	80.5 ± 0.3	85.2 ± 0.3	80.3 ± 0.3	40.2 ± 0.4	45.6 ± 0.4	47.2 ± 0.4	91.3 ± 0.2	89.3 ± 0.3	86.7 ± 0.3	28.1 ± 0.4	27.7 ± 0.4	21.1 ± 0.4
LLaDA-1.5	baseline	69.9 ± 0.3	79.2 ± 0.3	81.0 ± 0.2	29.2 ± 0.4	32.2 ± 0.3	35.6 ± 0.4	85.3 ± 0.2	86.3 ± 0.2	83.0 ± 0.3	21.6 ± 0.4	21.0 ± 0.4	20.8 ± 0.4
Fixed Weighting	68.8 ± 0.2	75.7 ± 0.2	80.3 ± 0.2	27.4 ± 0.2	30.8 ± 0.2	34.6 ± 0.2	87.3 ± 0.2	85.3 ± 0.2	84.0 ± 0.2	23.4 ± 0.2	22.3 ± 0.2	18.8 ± 0.2	
+ Temporal Voting	Linear Weighting	71.0 ± 0.2	79.8 ± 0.2	81.0 ± 0.2	29.2 ± 0.2	32.8 ± 0.2	35.8 ± 0.2	86.0 ± 0.2	87.0 ± 0.2	84.0 ± 0.2	24.2 ± 0.2	23.4 ± 0.2	19.1 ± 0.2
	Exp. Weighting	70.8 ± 0.2	79.6 ± 0.2	81.1 ± 0.2	29.2 ± 0.2	33.0 ± 0.2	36.4 ± 0.2	85.7 ± 0.2	87.7 ± 0.2	84.0 ± 0.2	26.3 ± 0.2	24.9 ± 0.2	21.2 ± 0.2
		+0.9	+0.4	+0.1	+0.0	+0.8	+0.8	+0.4	+1.4	+1.0	+4.7	+3.9	+0.4
	EverPass @1 t	81.5 ± 0.3	88.9 ± 0.3	83.6 ± 0.3	39.8 ± 0.4	47.4 ± 0.4	49.2 ± 0.4	90.7 ± 0.2	90.3 ± 0.2	86.0 ± 0.3	30.5 ± 0.4	27.0 ± 0.4	25.4 ± 0.4

392
393394 applicable, particularly in unsupervised settings, and offers a novel direction for improving dLLMs.
395 Furthermore, we show that combining the negative TSE reward with the accuracy reward based on
396 ground-truth answers yields notably better performance than using the accuracy reward alone.

397

398

5 EXPERIMENTS

400
401

5.1 IMPLEMENTATION DETAILS

402
403
404
405
406
407
408
409
410
411
412

Experimental Setup. We use LLaDA-8B-Instruct (Nie et al., 2025) and LLaDA-1.5 (Zhu et al., 2025), evaluating on four math benchmarks: GSM8K (Cobbe et al., 2021), MATH500 (Hendrycks et al., 2021), SVAMP (Patel et al., 2021), and Countdown (Pan et al., 2025). Following d1 (Zhao et al., 2025), we report performance under different output lengths. Temporal self-consistency voting applies exponential weights to sampling steps. All of our experiments adopt semi-autoregressive decoding with low-confidence remasking, using a block size of 32. The diffusion step is set to half of the target output length. For post-training, LLaDA-8B-Instruct undergoes supervised fine-tuning (SFT) on s1K (Muennighoff et al., 2025) for 20 epochs with 4,096 token sequences, followed by reinforcement fine-tuning (RFT). 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 uses 8 H800 GPUs. Further details are in Appendix B.

413
414
415
416
417
418
419
420
421

Answer Extraction for Voting and Semantic Clustering. At every diffusion step, LLaDA first generates a fully decoded token sequence *before* applying the remasking operation. Although some positions may be remasked and re-predicted in later steps, this pre-remask sequence is always a complete, unmasked prediction. We use these fully decoded sequences as the intermediate outputs for all temporal analyses. This design guarantees that each intermediate prediction provides a syntactically complete candidate answer. Answer extraction itself is performed by parsing the `<answer>...</answer>` block. If a pre-remask sequence does not contain a valid answer span at a particular step, that step is excluded from temporal self-consistency voting or semantic clustering. Only steps containing valid, fully parsed answers contribute to our temporal statistics.

422
423

5.2 TEMPORAL SELF-CONSISTENCY VOTING

424
425
426
427
428

Voting Strategies. We apply 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 weighting takes the form $f(t) = 1 - t/T$, and exponential uses $f(t) = \exp(\alpha(1 - t/T))$ with $\alpha = 5$. Both linear and exponential schemes prioritize early diffusion time steps, *i.e.*, latter sampling steps.

429
430
431

Ablations on Voting Strategies. As shown in Table 1, linear and exponential weighting improve inference performance, with exponential yielding the largest gains, *e.g.*, LLaDA-8B-Instruct improves by 1.6%, 1.2%, 1.0%, and 2.2% on GSM8K, MATH500, SVAMP, and Countdown, respectively. Fixed weighting performs slightly worse, likely because equal weights amplify

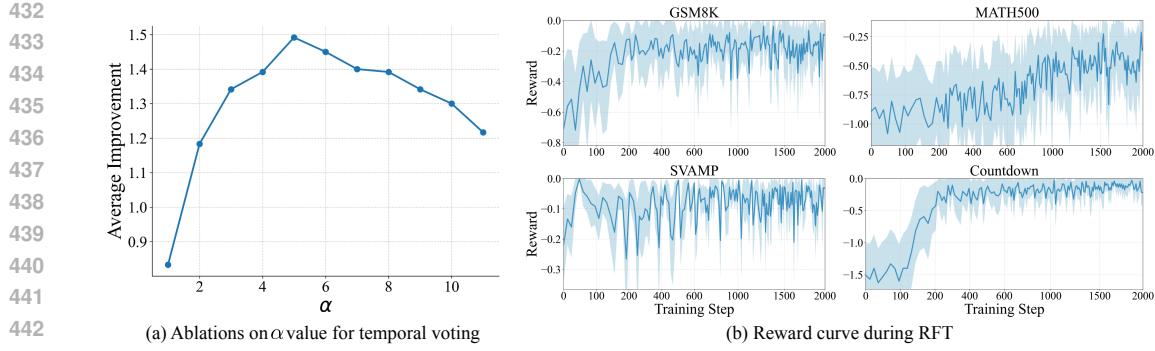


Figure 5: (a) Ablations on α value for temporal voting with exponential weighting. (b) Negative temporal semantic entropy reward curve during reinforcement fine-tuning.

Table 2: **Performance of reinforcement fine-tuning.** Unlike d1 (Zhao et al., 2025), which requires ground-truth answers, our method uses negative Temporal Semantic Entropy (TSE) as the reward without labels. Combining TSE with accuracy-based rewards yields further gains across benchmarks. **Green numbers** denote improvements over baseline.

Method / Seq Len	GSM8K			MATH500			SVAMP			Countdown			
	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	negative TSE reward (ours)	72.2	78.8	80.2	30.6	34.6	38.0	84.3	89.0	88.7	38.6	53.5	44.9
	combining both (ours)	72.1	80.0	83.0	31.2	35.4	41.4	85.0	90.3	92.3	41.5	42.6	54.7
		+1.9	+1.3	+2.9	+7.4	+1.0	+4.6	+3.3	+7.0	+9.6	+20.0	+22.7	+33.2
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
	accuracy reward (d1)	73.0	78.9	83.1	29.8	36.2	40.2	84.7	89.3	88.0	38.7	29.7	39.1
+ RFT	negative TSE reward (ours)	72.0	80.8	82.6	30.2	35.0	40.0	86.3	88.3	87.3	50.0	55.9	53.1
	combining both (ours)	73.2	80.5	84.0	29.6	35.4	41.4	86.3	90.3	89.0	44.5	46.9	63.3
		+3.4	+1.1	+2.9	+0.6	+3.0	+6.0	+1.0	+4.0	+5.7	+23.0	+25.8	+42.6

inaccurate early predictions. We therefore adopt exponential weighting by default. We further ablate the exponential hyperparameter α . As shown in Fig. 5a, α values ranging from 1 to 11 consistently improve accuracy, peaking at $\alpha = 5$ with an average gain of 1.5%. Thus, we set $\alpha = 5$ by default.

5.3 TEMPORAL CONSISTENCY REINFORCEMENT

Main Results. Table 2 reports results of incorporating temporal consistency into RFT. We have the following observations. (1) Using TSE reward alone consistently improves performance across lengths and datasets. (2) TSE reward matches or surpasses accuracy reward despite not using ground truth, e.g., on Countdown, LLaDA-8B-Instruct improves 24.7% vs. 15.1% with d1. (3) Combining TSE with accuracy reward further boosts results, with absolute gains of 0.9% (GSM8K), 0.2% (MATH500), 1.7% (SVAMP), and 10.2% (Countdown) over d1. Overall, our method achieves average improvements of 2.0%, 4.3%, 6.6%, and 25.3% over the SFT baseline, confirming the benefit of encouraging temporal consistency.

Training Dynamics. We visualize the reward curves during training using LLaDA-8B-Instruct as an example, as shown in Fig. 5b. 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 analyze LLaDA-8B-Instruct fine-tuned with the negative TSE reward, with generation length 128, evaluating its behavior across several dimensions: TSE, the ever pass rate, and the number of effective tokens (defined as the average count of non-padding, non-EOS tokens per generation). We have the following observations. (1) As shown in Fig. 6a, temporal semantic entropy consistently decreases across various datasets after RFT, reflecting enhanced

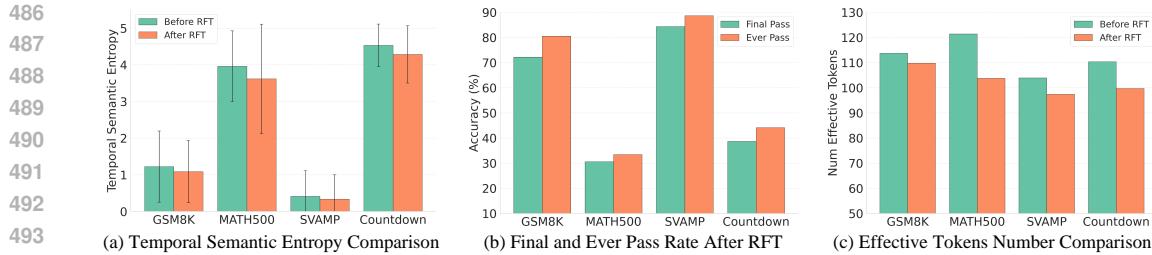


Figure 6: **Model attributes after reinforcement finetuning (RFT).** (a) Temporal semantic entropy decreases after RFT, showing improved semantic consistency in outputs. (b) The ever pass rate remains above the final pass rate, leaving room for further gains. (c) Effective tokens per generation drop after RFT, yielding more concise outputs.

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. (3) In Fig. 6c, effective tokens decline after RFT, implying more concise outputs. We suspect shorter generations may reduce temporal oscillations, though this requires further study. Further analysis of model attributes after RFT and a detailed discussion of limitations are provided in Appendix E.5 and Appendix D.

6 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*.

ETHICS STATEMENT

This work does not involve human subjects, personal data, or sensitive information. All datasets used in our experiments (GSM8K, MATH500, SVAMP, and Countdown) are publicly available benchmark datasets designed for evaluating mathematical reasoning in large language models. We strictly adhered to ethical research practices and did not conduct any data collection that could raise privacy, security, or fairness concerns. Our methods—Temporal Self-Consistency Voting and Temporal Consistency Reinforcement—focus on improving the robustness and accuracy of diffusion language models, without introducing risks of harmful applications. To the best of our knowledge, this research complies with the ICLR Code of Ethics and poses no foreseeable ethical concerns.

REPRODUCIBILITY STATEMENT

We have made extensive efforts to ensure the reproducibility of our work. Detailed dataset descriptions are provided in Appendix B.1, and training configurations and hyperparameters are reported in Appendix B.2. The sampling and evaluation procedures are outlined in Appendix B.3. To further promote transparency, we provide mathematical formulations and thorough descriptions of our proposed algorithms—Temporal Self-Consistency Voting and Temporal Consistency Reinforcement—directly in the main paper. Upon acceptance, we will release our models, together with training and inference code, to facilitate replication and further research.

540 REFERENCES
541

542 Shivam Agarwal, Zimin Zhang, Lifan Yuan, Jiawei Han, and Hao Peng. The unreasonable
543 effectiveness of entropy minimization in llm reasoning. *arXiv preprint arXiv:2505.15134*, 2025.

544 Marianne Arriola, Aaron Gokaslan, Justin T Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han,
545 Subham Sekhar Sahoo, and Volodymyr Kuleshov. Block diffusion: Interpolating between
546 autoregressive and diffusion language models. *arXiv preprint arXiv:2503.09573*, 2025.

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

551 Pavel Avdeyev, Chenlai Shi, Yuhao Tan, Kseniia Dudnyk, and Jian Zhou. Dirichlet diffusion score
552 model for biological sequence generation. In *International Conference on Machine Learning*, pp.
553 1276–1301. PMLR, 2023.

555 Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared
556 Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large
557 language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

558 Minghan Chen, Guikun Chen, Wenguan Wang, and Yi Yang. Seed-grpo: Semantic entropy enhanced
559 grpo for uncertainty-aware policy optimization. *arXiv preprint arXiv:2505.12346*, 2025.

561 Chaoran Cheng, Jiahua Li, Jian Peng, and Ge Liu. Categorical flow matching on statistical
562 manifolds. *Advances in Neural Information Processing Systems*, 37:54787–54819, 2024.

563 Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and
564 Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge.
565 *arXiv:1803.05457v1*, 2018.

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

570 Ganqu Cui, Yuchen Zhang, Jiacheng Chen, Lifan Yuan, Zhi Wang, Yuxin Zuo, Haozhan Li, Yuchen
571 Fan, Huayu Chen, Weize Chen, et al. The entropy mechanism of reinforcement learning for
572 reasoning language models. *arXiv preprint arXiv:2505.22617*, 2025.

574 Mehul Damani, Isha Puri, Stewart Slocum, Idan Shenfeld, Leshem Choshen, Yoon Kim, and Jacob
575 Andreas. Beyond binary rewards: Training lms to reason about their uncertainty. *arXiv preprint
576 arXiv:2507.16806*, 2025.

577 Oscar Davis, Samuel Kessler, Mircea Petrache, Ismail Ceylan, Michael Bronstein, and Joey Bose.
578 Fisher flow matching for generative modeling over discrete data. *Advances in Neural Information
579 Processing Systems*, 37:139054–139084, 2024.

581 Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, and Yarin Gal. Detecting hallucinations in large
582 language models using semantic entropy. *Nature*, 630(8017):625–630, 2024.

583 Tilmann Gneiting and Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation.
584 *Journal of the American statistical Association*, 102(477):359–378, 2007.

586 Shanshan Gong, Ruixiang Zhang, Huangjie Zheng, Jiatao Gu, Navdeep Jaitly, Lingpeng Kong, and
587 Yizhe Zhang. Diffucoder: Understanding and improving masked diffusion models for code
588 generation. *arXiv preprint arXiv:2506.20639*, 2025.

589 Ishaan Gulrajani and Tatsunori B Hashimoto. Likelihood-based diffusion language models.
590 *Advances in Neural Information Processing Systems*, 36:16693–16715, 2023.

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

594 Xiaochuang Han, Sachin Kumar, and Yulia Tsvetkov. Ssd-lm: Semi-autoregressive simplex-
 595 based diffusion language model for text generation and modular control. *arXiv preprint*
 596 *arXiv:2210.17432*, 2022.

597

598 Haoyu He, Katrin Renz, Yong Cao, and Andreas Geiger. Mdpo: Overcoming the training-inference
 599 divide of masked diffusion language models. *arXiv preprint arXiv:2508.13148*, 2025.

600 Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song,
 601 and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *NeurIPS*,
 602 2021.

603

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

606 Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J
 607 Fleet. Video diffusion models. *Advances in neural information processing systems*, 35:8633–
 608 8646, 2022.

609

610 Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang,
 611 Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

612

613 Wenzuan Huang, Bohan Jia, Zijie Zhai, Shaosheng Cao, Zheyu Ye, Fei Zhao, Zhe Xu, Yao Hu, and
 614 Shaohui Lin. Vision-r1: Incentivizing reasoning capability in multimodal large language models.
arXiv preprint arXiv:2503.06749, 2025.

615

616 Sakaguchi Keisuke, Le Bras Ronan, Bhagavatula Chandra, and Choi Yejin. Winogrande: An
 617 adversarial winograd schema challenge at scale. 2019.

618

619 Jaeyeon Kim, Kulin Shah, Vasilis Kontonis, Sham Kakade, and Sitan Chen. Train for the
 620 worst, plan for the best: Understanding token ordering in masked diffusions. *arXiv preprint*
arXiv:2502.06768, 2025.

621

622 Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Semantic uncertainty: Linguistic invariances for
 623 uncertainty estimation in natural language generation. *arXiv preprint arXiv:2302.09664*, 2023.

624

625 Pengxiang Li, Yefan Zhou, Dilxat Muhtar, Lu Yin, Shilin Yan, Li Shen, Yi Liang, Soroush Vosoughi,
 626 and Shiwei Liu. Diffusion language models know the answer before decoding. *arXiv preprint*
arXiv:2508.19982, 2025a.

627

628 Xiang Li, John Thickstun, Ishaan Gulrajani, Percy S Liang, and Tatsunori B Hashimoto. Diffusion-
 629 lm improves controllable text generation. *Advances in neural information processing systems*, 35:
 4328–4343, 2022.

630

631 Yuetai Li, Zhangchen Xu, Fengqing Jiang, Bhaskar Ramasubramanian, Luyao Niu, Bill Yuchen Lin,
 632 Xiang Yue, and Radha Poovendran. Temporal sampling for forgotten reasoning in llms. *arXiv*
preprint arXiv:2505.20196, 2025b.

633

634 Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan
 635 Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let’s verify step by step. In *The Twelfth*
636 International Conference on Learning Representations, 2023.

637

638 Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching
 639 for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

640

641 Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and
 642 transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

643

644 Yexiang Liu, Zekun Li, Zhi Fang, Nan Xu, Ran He, and Tieniu Tan. Rethinking the role of
 645 prompting strategies in llm test-time scaling: A perspective of probability theory. *arXiv preprint*
arXiv:2505.10981, 2025a.

646

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

648 Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios
 649 of the data distribution. *arXiv preprint arXiv:2310.16834*, 2023.

650

651 Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri
 652 Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement
 653 with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594, 2023.

654 Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke
 655 Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time
 656 scaling. *arXiv preprint arXiv:2501.19393*, 2025.

657

658 Shen Nie, Fengqi Zhu, Chao Du, Tianyu Pang, Qian Liu, Guangtao Zeng, Min Lin, and Chongxuan
 659 Li. Scaling up masked diffusion models on text. *arXiv preprint arXiv:2410.18514*, 2024.

660

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

663

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

666

667 Arkil Patel, Satwik Bhattacharya, and Navin Goyal. Are nlp models really able to solve simple math
 668 word problems? *arXiv preprint arXiv:2103.07191*, 2021.

669

670 Mihir Prabhudesai, Lili Chen, Alex Ippoliti, Katerina Fragkiadaki, Hao Liu, and Deepak Pathak.
 671 Maximizing confidence alone improves reasoning. *arXiv preprint arXiv:2505.22660*, 2025.

672

673 Zhangyang Qi, Zhixiong Zhang, Yizhou Yu, Jiaqi Wang, and Hengshuang Zhao. Vln-r1: Vision-
 674 language navigation via reinforcement fine-tuning. *arXiv preprint arXiv:2506.17221*, 2025.

675

676 Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language
 677 models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

678

679 Subham Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin Chiu,
 680 Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language
 681 models. *Advances in Neural Information Processing Systems*, 37:130136–130184, 2024.

682

683 Subham Sekhar Sahoo, Zhihan Yang, Yash Akhauri, Johnna Liu, Deepansha Singh, Zhoujun Cheng,
 684 Zhengzhong Liu, Eric Xing, John Thickstun, and Arash Vahdat. Esoteric language models. *arXiv
 preprint arXiv:2506.01928*, 2025.

685

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

688

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

692

693 Haozhan Shen, Peng Liu, Jingcheng Li, Chunxin Fang, Yibo Ma, Jiajia Liao, Qiaoli Shen, Zilun
 694 Zhang, Kangjia Zhao, Qianqian Zhang, et al. Vlm-r1: A stable and generalizable r1-style large
 695 vision-language model. *arXiv preprint arXiv:2504.07615*, 2025.

696

697 Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis K. Titsias. Simplified and
 698 generalized masked diffusion for discrete data. In *Advances in Neural Information Processing
 699 Systems*, 2024.

700

701 Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally
 702 can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

703

704 Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv
 705 preprint arXiv:2010.02502*, 2020.

706

707 Hannes Stark, Bowen Jing, Chenyu Wang, Gabriele Corso, Bonnie Berger, Regina Barzilay, and
 708 Tommi Jaakkola. Dirichlet flow matching with applications to dna sequence design. *arXiv
 709 preprint arXiv:2402.05841*, 2024.

702 Guanghan Wang, Yair Schiff, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Remasking discrete
 703 diffusion models with inference-time scaling. *arXiv preprint arXiv:2503.00307*, 2025.
 704

705 Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha
 706 Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language
 707 models. *arXiv preprint arXiv:2203.11171*, 2022.

708 Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny
 709 Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in
 710 neural information processing systems*, 35:24824–24837, 2022.

711 Shaoan Xie, Lingjing Kong, Xiangchen Song, Xinshuai Dong, Guangyi Chen, Eric P Xing, and Kun
 712 Zhang. Step-aware policy optimization for reasoning in diffusion large language models. *arXiv
 713 preprint arXiv:2510.01544*, 2025.

714

715 Ling Yang, Ye Tian, Bowen Li, Xinchen Zhang, Ke Shen, Yunhai Tong, and Mengdi Wang. Mmada:
 716 Multimodal large diffusion language models. *arXiv preprint arXiv:2505.15809*, 2025.

717

718 Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik
 719 Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances
 720 in neural information processing systems*, 36:11809–11822, 2023.

721 Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng
 722 Kong. Dream 7b, 2025. URL <https://hkunlp.github.io/blog/2025/dream>.

723

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

727 Qingyang Zhang, Haitao Wu, Changqing Zhang, Peilin Zhao, and Yatao Bian. Right question
 728 is already half the answer: Fully unsupervised llm reasoning incentivization. *arXiv preprint
 729 arXiv:2504.05812*, 2025.

730

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

733

734 Hao Zhong, Muzhi Zhu, Zongze Du, Zheng Huang, Canyu Zhao, Mingyu Liu, Wen Wang, Hao
 735 Chen, and Chunhua Shen. Omni-r1: Reinforcement learning for omnimodal reasoning via two-
 736 system collaboration. *arXiv preprint arXiv:2505.20256*, 2025.

737

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

741

742

743

744

745

746

747

748

749

750

751

752

753

754

755

756 **LLM USAGE**
757758 In this section, we clarify the role of large language models (LLMs) in preparing this work. The
759 model was used exclusively for language polishing, such as refining grammar, style, and readability,
760 without contributing to the research design, analysis, or conclusions.
761762 **A APPENDIX OVERVIEW**
763764 This appendix provides additional implementation details, empirical analysis, and extended results
765 to supplement the main paper. It is organized as follows:
766767 • **Appendix B: More Implementation Details**
768769 Provides further implementation information, including:
770771

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

775 • **Appendix C: More Analysis**
776777 Presents extended analyses, including:
778779

- 780 – Appendix C.1: Accuracy and entropy analysis on the MATH500 and SVAMP datasets
- 781 – Appendix C.2: Temporal semantic entropy across varying generated lengths
- 782 – Appendix C.3: Block-level token entropy analysis
- 783 – Appendix C.4: Time analysis for temporal consistency voting
- 784 – Appendix C.5: Investigating the Causes of Time Oscillation

785 • **Appendix D: Limitations**
786787 Discuss limitations and analyze failure cases, including:
788789

- 790 – Appendix D.1: Discuss potential limitations of our method
- 791 – Appendix D.2: Analysis of failure cases on the Sudoku dataset

792 • **Appendix E: More Experimental Results**
793794 Includes additional experimental findings, such as:
795796

- 797 – Appendix E.2: Training a unified model across multiple tasks
- 798 – Appendix E.3: Possible alternatives to temporal semantic entropy
- 799 – Appendix E.4: Ablation studies on different scoring rules for combining TSE with
800 accuracy reward
- 801 – Appendix E.5: Performance of temporal self-consistency voting on reinforcement
802 fine-tuned models
- 803 – Appendix E.6: Detailed examples illustrating temporal oscillation

804 **B MORE IMPLEMENTATION DETAILS**
805806 **B.1 DATASETS**
807808 We provided detailed descriptions of the datasets as follows:
809810

- 811 • GSM8K (Cobbe et al., 2021) comprises 8.5K linguistically diverse grade school math word
812 problems (7.5K training, 1K test), solvable by bright middle school students via 2–8 steps
813 of basic arithmetic, suited for multi-step mathematical reasoning.
- 814 • MATH500 (Lightman et al., 2023) is a curated subset of 500 problems selected from the
815 broader MATH dataset (Hendrycks et al., 2021), featuring high-school-level competition
816 math problems.
- 817 • SVAMP (Patel et al., 2021) serves as a benchmark for elementary-level Math Word
818 Problems (MWPs), where each MWP is a short natural language narrative describing a
819 scenario and asking questions about unknown quantities.

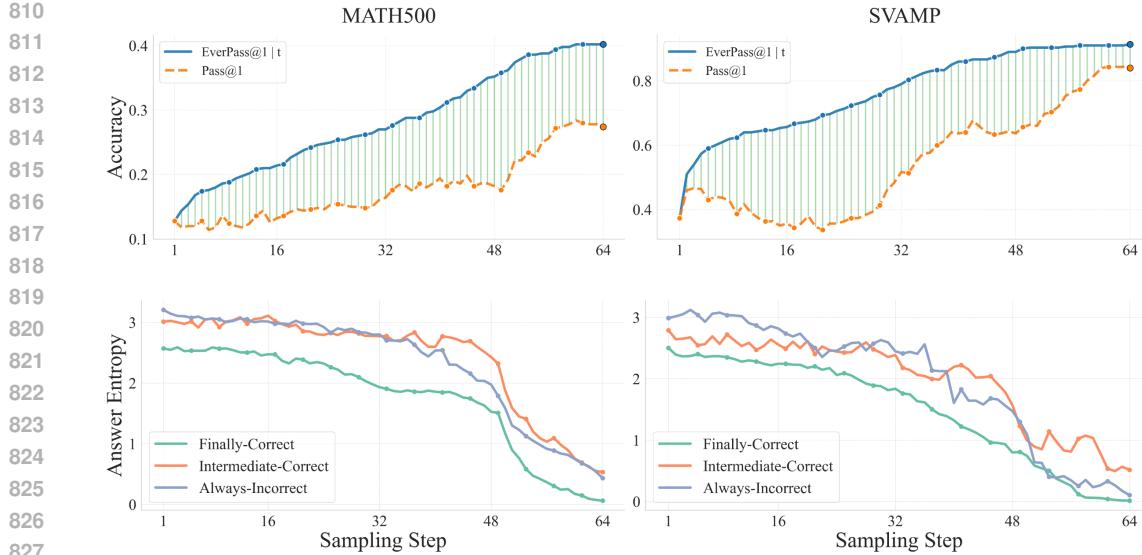


Figure S1: **Top Row:** Pass @1 and Pass @1 | t for MATH500 and SVAMP are provided as supplementary results to Fig. 2, using the same experimental settings. A noticeable gap similar to that observed in GSM8K and Countdown between Pass @1 and Pass @1 | t is present across all sampling steps. **Bottom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

- 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.

B.3 SAMPLING AND EVALUATION

Unified Decoding Configuration. 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. **We use a block size of 32, following LLaDA (Nie et al., 2025) and d1 (Zhao et al., 2025).** 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.

Generation Lengths. (1) *Evaluation.* For test-time self-consistency voting and all RL evaluations, we evaluate three output lengths: 128, 256, and 512 tokens, corresponding to 64, 128, and 256 diffusion steps, respectively (again following d1 (Zhao et al., 2025)). (2) *Analysis experiments.* In the exploratory study in Sec. 3.2 / Fig. 1, we analyze output lengths of 128, 256, and 512, and observe temporal oscillation under all settings. Sec. 3.3 uses the default 128-token / 64-step configuration.

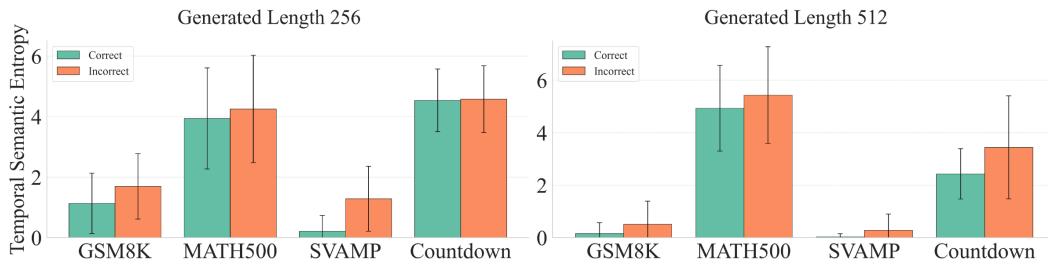


Figure S2: **Temporal semantic entropy with varying generation lengths.** Average temporal semantic entropy across datasets with generation lengths of 256 and 512, showing consistent patterns with Fig. 4. Higher entropy generally correlates with lower accuracy.

Appendix Sec. C.2 / Fig. S2 further reports temporal semantic entropy under 256- and 512-length generations to demonstrate robustness.

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.

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 are eventually answered correctly tend to display consistently low temporal semantic entropy, indicating that their intermediate predictions remain stable and semantically coherent throughout the decoding process.

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

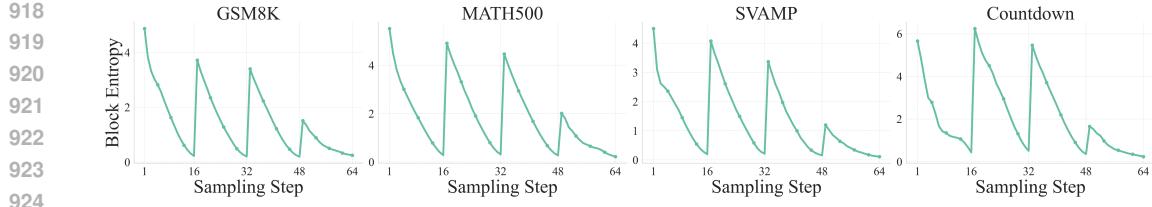


Figure S3: **Block-level entropy dynamics under semi-autoregressive sampling.** During sampling, sequences are partitioned into fixed-length blocks, processed in a left-to-right order, with remasking and unmasking operations restricted to the current block. Average token entropy within a block decreases 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.

Table S1: **Runtime comparison (in seconds) for temporal self-consistency voting.** For the voting setting, we report end-to-end latency from receiving the input to producing the final voted output. For the non-voting setting, we measure the time required to generate an output of the same final length. Across all datasets, the additional cost introduced by voting is minimal.

Method / Seq Len	GSM8K			MATH500			SVAMP			Countdown		
	128	256	512	128	256	512	128	256	512	128	256	512
w/o voting (baseline)	4.48	10.76	29.24	4.47	10.87	29.45	4.23	10.31	28.45	4.59	10.99	29.58
w/ voting	4.53	10.86	29.76	4.54	10.91	29.61	4.29	10.41	28.59	4.64	11.12	29.81

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.

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.

C.4 TIME ANALYSIS FOR TEMPORAL SELF-CONSISTENCY VOTING

To assess the efficiency of our temporal self-consistency voting approach, we compare the per-sample runtime with and without voting. We use LLaDA-8B-Instruct with semi-autoregressive decoding (block size 32), a diffusion step set to half of the target output length, and a batch size of 1. All timing experiments are performed on four NVIDIA 4090-24GB GPUs.

For the voting setting, we measure the end-to-end runtime from receiving the input to producing the final voted answer. For the non-voting setting, we measure the time to generate an output of the same final length. The difference reflects the additional operations performed by voting—namely answer parsing, clustering, and selecting the final answer. We report the average time to generate one response of the fixed output length for each dataset.

As shown in Table S1, temporal self-consistency voting introduces essentially no extra runtime. Across all datasets, the overhead is about 1% per sample—negligible relative to the total computation time.

Table S2: **Evaluation results under different randomness levels and remasking strategies.** Black numbers denote Pass@1, while gray numbers denote EverPass@1| t . As shown, the gap between Pass@1 and EverPass@1| t persists across all seeds, temperature settings, and remasking strategies, indicating that the phenomenon of time oscillation is not sensitive to these configuration choices.

Setting			GSM8K			MATH500			SVAMP			Countdown		
Seed	Temp.	Remasking	128	256	512	128	256	512	128	256	512	128	256	512
42	0.0	Low Confidence	67.9	76.5	79.2	27.4	33.4	35.8	83.3	84.0	84.7	18.8	20.3	15.6
			80.3	84.8	81.0	40.2	45.6	47.2	90.7	89.7	87.3	25.8	27.7	19.9
218	0.0	Low Confidence	68.2	76.5	79.2	27.2	33.4	35.8	83.0	84.0	84.7	20.3	20.3	16.0
			81.5	84.8	81.0	40.8	45.6	47.2	91.0	89.7	87.3	29.3	27.0	20.7
66	0.0	Low Confidence	69.6	76.3	78.2	27.2	33.4	35.8	83.3	83.3	84.7	20.3	22.3	18.8
			80.3	85.2	80.3	40.8	45.6	47.2	90.7	89.3	86.7	27.7	29.3	22.7
42	0.6	Low Confidence	67.4	76.6	78.5	25.6	33.8	33.8	84.7	85.0	84.7	20.3	22.3	18.8
			82.6	86.7	81.5	43.2	49.0	50.8	91.7	90.7	87.0	29.3	20.7	22.7
42	1.0	Low Confidence	66.1	75.2	77.0	23.8	30.6	30.8	85.0	84.3	85.0	19.5	17.6	16.8
			78.8	81.9	79.2	37.6	42.0	42.2	89.7	87.0	87.3	22.3	19.5	21.1
42	0.0	Random	57.2	64.5	62.1	19.8	24.4	23.4	73.0	73.3	72.7	10.9	8.2	10.5
			76.9	76.6	66.8	35.8	38.6	33.4	86.0	82.0	78.0	22.3	12.5	14.5
218	0.0	Random	59.4	61.6	62.1	19.8	24.6	21.2	75.0	73.3	72.7	12.1	9.8	10.5
			75.3	75.7	67.2	35.2	42.2	33.6	87.3	79.0	77.7	23.8	14.1	15.6

C.5 INVESTIGATING THE CAUSES OF TIME OSCILLATION

First, we investigate the effects of randomness and temperature on time oscillation. As shown in Table S2, although both Pass @1 and EverPass @1| t exhibit slight fluctuations across different seeds and temperature settings, the gap between them consistently persists. We further examine the impact of applying a random remasking strategy. While this strategy noticeably reduces both Pass @1 and EverPass @1| t , the gap remains. These observations suggest that time oscillation is an inherent and pervasive phenomenon originating from the model itself.

We hypothesize that time oscillation may arise from the training dynamics, where the same input signal can be associated with multiple valid outputs, which is analogous to diffusion models in the vision domain (Liu et al., 2022; Lipman et al., 2022). Specifically, consider a sequence $seq_1 = "abcd"$, possible that another sequence $seq_2 = "abce"$ also appears in the training data with the same masked pattern, and the model is likewise supervised to predict from " $<\text{MASK}>bc <\text{MASK}>$ ". As a result, the model receives conflicting supervision signals, causing it to oscillate between " $abcd$ " and " $abce$ " during sampling.

Moreover, even if no alternative sequence like *seq2* exists, the model may still suffer from underfitting under certain masking patterns. In such cases, given the prompt “<MASK>bc <MASK>”, the model might incorrectly predict “abcx” instead of the intended sequence. This behavior can likewise induce time oscillation during sampling.

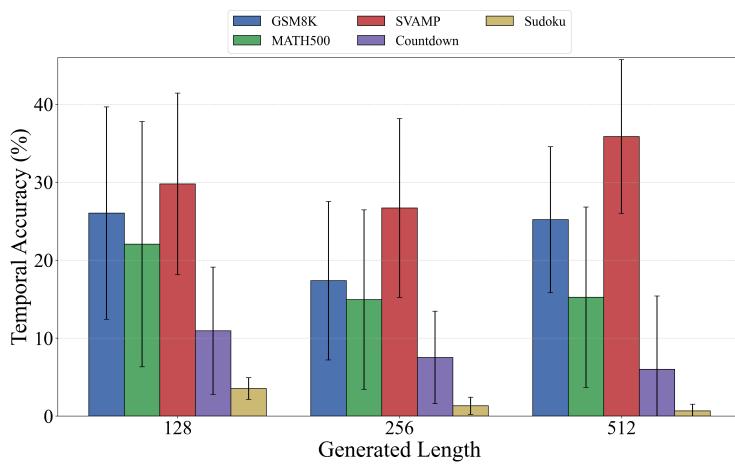
D LIMITATIONS

D.1 DISCUSSIONS ON 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.2, 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.

1026
 1027 **Table S3: Performance of temporal self-consistency voting and temporal consistency**
 1028 **reinforcement on the Sudoku dataset.** The baseline model is obtained by applying supervised
 1029 fine-tuning to LLaDA-8B-Instruct using the s1K dataset.

Model	Method / Seq Len	Sudoku		
		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



1040
 1041
 1042
 1043
 1044
 1045
 1046
 1047
 1048
 1049
 1050
 1051
 1052
 1053 **Figure S4: Temporal accuracy across datasets.** Temporal Accuracy on Sudoku remains
 1054 consistently low (all below 5%) across different settings, indicating that the model rarely generates
 1055 correct answers during the sampling process. This scarcity of valid candidates severely limits the
 1056 effectiveness of voting mechanisms, as there are insufficient correct outputs to reliably converge on
 1057 the correct answer.

1058 D.2 FAILURE CASE ANALYSIS

1059
 1060 As discussed in Appendix D.1, our method may rely on the model’s initial performance to achieve
 1061 further improvements. To investigate this limitation, we take the challenging Sudoku dataset as a
 1062 case study. As shown in Table S3, directly applying our proposed temporal self-consistency voting
 1063 and temporal consistency reinforcement with the negative TSE reward alone results in a noticeable
 1064 performance drop on the Sudoku dataset. For example, the original accuracy of the base model for
 1065 the generation length of 512 is 5.5%, while the voting method achieves 2.8%, reflecting a decline of
 1066 2.7%.

1067 To better understand this problem, we conduct a deeper analysis of the model’s behavior during
 1068 generation. We define a metric called **Temporal Accuracy**, which quantifies the average correctness
 1069 across all intermediate sampling steps in the generation process. Formally, let $e_{i,t}$ represent the
 1070 correctness indicator for the i -th question at the t -th sampling step, where $e_{i,t} = 1$ if the answer
 1071 is correct, and 0 otherwise. Then, the **Temporal Accuracy** for a dataset with N examples and T
 1072 sampling steps is computed as:

$$1073 \text{TemporalAccuracy} = \frac{1}{N \cdot T} \sum_{i=1}^N \sum_{t=1}^T e_{i,t}. \quad (S1)$$

1074
 1075 As shown in Fig. S4, the **Temporal Accuracy** of the Sudoku is much lower than the other 4 datasets,
 1076 where temporal self-consistency voting proved effective. On the Sudoku dataset, we observe that
 1077 the Temporal Accuracy remains exceedingly low—below 5% on average—across all intermediate

1080
1081 **Table S4: Temporal self-consistency voting results on ARC-C and Winogrande using the**
1082 **semi-autoregressive setup (block size 32).** Exponential and linear weighting schemes yield
1083 consistent accuracy improvements without QA-specific tuning, demonstrating robust gains across
1084 both datasets.

	Method / Seq Len	ARC-C			Winogrande		
		128	256	512	128	256	512
LLaDA-8B-Instruct	baseline	81.4	83.3	83.1	53.1	53.6	52.5
	Fixed Weighting	82.6	83.8	83.4	54.4	55.1	55.0
	Linear Weighting	82.6	83.8	83.4	56.1	57.7	58.4
	Exp. Weighting	82.6	83.8	83.4	54.4	55.1	55.0
		+1.2	+0.5	+0.3	+1.3	+1.5	+2.5
	EverPass @1 t	86.1	86.9	85.5	61.5	61.6	62.0

1093
1094 **Table S5: Temporal Consistency Reinforcement results on ARC-C and Winogrande.** Both
1095 TSE-only and combined rewards consistently outperform the baseline across sequence lengths, with
1096 the combined reward yielding the largest gains.

	Method / Seq Len	ARC-C			Winogrande		
		128	256	512	128	256	512
LLaDA	baseline	81.6	82.8	83.4	54.5	56.6	56.5
	negative TSE reward (ours)	84.5	84.8	84.8	59.1	60.6	61.3
		+2.9	+2.0	+1.4	+4.6	+4.0	+4.8
+ RFT	combining both (ours)	84.5	85.8	86.3	66.6	69.0	68.5
		+2.9	+3.0	+2.9	+12.1	+12.4	+12.0

1104
1105 steps. This low signal makes it difficult for temporal voting mechanisms to reliably identify the
1106 correct answer and for reinforcement learning signals to guide the model effectively.

1107
1108 Interestingly, while RFT with only negative TSE reward leads to poorer results, combining TSE with
1109 accuracy reward can achieve better performance than using accuracy reward alone. We hypothesize
1110 that this is because the integration of TSE allows the model to receive more fine-grained rewards,
1111 rather than just binary outcomes of correct or incorrect.

1112 E MORE EXPERIMENTAL RESULTS

1117 E.1 EXPERIMENTS ON NATURAL LANGUAGE QUESTION-ANSWERING DATASETS

1119 To verify that our methods generalize beyond math reasoning, we added experiments on two natural
1120 language QA benchmarks, ARC-C (Clark et al., 2018) and Winogrande (Keisuke et al., 2019),
1121 evaluating both Temporal Self-Consistency Voting and Temporal Consistency Reinforcement.

1122 **Temporal Self-Consistency Voting.** We adopt exactly the same semi-autoregressive decoding
1123 configuration as described in the main paper, using a block size of 32. For the exponential weighting
1124 scheme, we directly reuse the hyperparameter $\alpha = 1.5$, which was originally tuned on mathematical
1125 reasoning datasets, and perform no additional tuning on QA tasks. As shown in Table S4, on ARC-
1126 C, exponential temporal voting yields an average improvement of approximately +0.7 accuracy
1127 points over the baseline. On Winogrande, the untuned exponential weighting achieves an average
1128 gain of +1.8 points, and applying linear weighting leads to further improvements, reaching 58.4
1129 accuracy (compared to a 52.5 baseline at length 512). These findings demonstrate that temporal
1130 voting provides consistent benefits on natural-language QA tasks and remains robust under different
1131 weighting schemes.

1132 **Temporal Consistency Reinforcement.** We train on ARC-C and Winogrande using the same
1133 GRPO configuration as in the main experiments, adopting either (i) a negative TSE reward
1134 alone or (ii) a combined TSE + accuracy reward. As shown in Table S7, both configurations

1134
1135
1136Table S6: **Unified Model Performance Across Multiple Tasks.** A single model is trained jointly on GSM8K, MATH500, SVAMP, and Countdown.

	Method / Seq Len	GSM8K			MATH500			SVAMP			Countdown		
		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)	71.3	77.3	80.7	28.0	33.6	39.2	84.0	84.0	85.0	25.8	22.3	37.1
	+ RFT	71.5	77.2	81.2	29.0	33.8	39.0	86.3	84.7	87.0	36.7	25.4	46.1
	combined reward (ours)	72.4	78.8	80.4	28.8	35.2	40.2	88.7	87.0	86.0	37.1	27.7	46.9
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)	72.5	79.2	81.7	28.6	35.0	39.0	89.0	87.7	85.3	30.5	24.6	41.8
	+ RFT	72.5	79.2	81.4	29.0	32.6	38.2	89.3	85.3	86.7	37.3	44.5	34.0
	combined reward (ours)	73.7	79.5	82.5	28.2	35.0	41.6	88.3	90.0	88.0	37.9	29.7	50.4

1147
1148

1149 consistently outperform the baseline across all sequence lengths on ARC-C, yielding improvements
1150 of approximately 2–3 accuracy points (e.g., 81.6 to 84.5 at 128 tokens and 82.8 to 85.8 at 256
1151 tokens). On Winogrande, the combined reward yields the largest gains, with improvements of up to
1152 +12.4 points (e.g., 56.6 to 69.0 at 256 tokens).

1153
1154
1155
1156
1157
1158

1153 Overall, these additional experiments on ARC-C and Winogrande demonstrate that both our test-
1154 time voting and reinforcement learning approaches generalize beyond mathematical reasoning to
1155 natural-language QA. This further supports our claim that leveraging temporal dynamics constitutes
1156 a broadly applicable principle for improving diffusion language models.

1158
1159
1160

E.2 TRAINING A UNIFIED MODEL ACROSS MULTIPLE TASKS

1161
1162
1163
1164

1161 To evaluate the generalization ability of our approach, we validate the proposed method in a multi-
1162 task setting. Specifically, we jointly train a single model on four datasets: GSM8K, MATH500,
1163 SVAMP, and Countdown. To ensure balanced training across tasks, we subsample each dataset so
1164 that the number of training examples is equal for all tasks.

1165
1166
1167
1168
1169
1170
1171
1172

1165 Table S6 summarizes the results. Across all four benchmarks, our combined reward method
1166 consistently outperforms the supervised fine-tuning baselines. On GSM8K, we observe clear gains
1167 at shorter sequence lengths, for example, with LLaDA-1.5 on length 128, accuracy improves from
1168 69.8 to 73.7, yielding +3.9 points. On SVAMP, similar short-sequence gains are observed, from 81.7
1169 to 88.7, a +7.0 points increase. For the more challenging MATH500, the best improvement occurs
1170 at LLaDA-1.5, length 512, where performance increases from 35.4 to 41.6 (+6.2 points). Finally, on
1171 Countdown, our method achieves dramatic improvements, for instance, with LLaDA-1.5 at length
1172 512, accuracy climbs from 20.7 to 50.4, a striking +29.7 points.

1173
1174
1175

1173 These results demonstrate that our method not only scales well across different reasoning domains
1174 but also enhances robustness under varying sequence lengths, confirming its effectiveness as a
1175 unified reward design for multi-task learning.

1176
1177
1178

E.3 POSSIBLE ALTERNATIVES TO TEMPORAL SEMANTIC ENTROPY

1179
1180
1181
1182

1179 Our method primarily adopts Temporal Semantic Entropy(TSE) as the underlying mechanism for
1180 temporal consistency modeling. However, TSE is not the only viable choice. To contextualize our
1181 design and clarify the landscape of alternatives, we examine two additional possible approaches:
1182 pairwise agreement and token entropy.

1183
1184
1185
1186

1183 To measure temporal consistency, a natural idea is to quantify how frequently the predicted answer
1184 changes along the sampling trajectory. Motivated by this intuition, we introduce ChangeRate as a
1185 metric for assessing the model’s confidence and stability.

1187

1187 During decoding, we obtain a sequence of T intermediate predictions, denoted as $\{\mathbf{x}_0^t\}_{t=1}^T$. Based on
this sequence, we define ChangeRate as the fraction of consecutive steps that produce the different

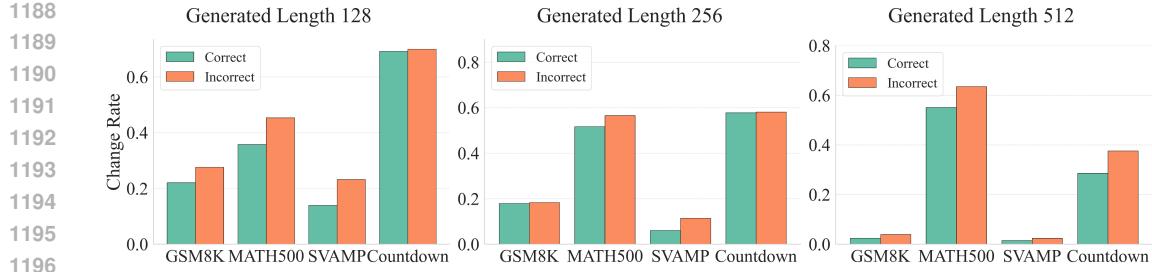


Figure S5: **ChangeRate of correct answers across different output lengths for each dataset.**
Lower ChangeRate indicates more stable predictions over the sampling trajectory.

Table S7: **Ablation results replacing TSE with alternative metrics, including negative average token entropy (ATE) and pairwise agreement (PA) between consecutive diffusion steps. **Bolded** values indicate the best performance among the three methods.**

Method / Seq Len	GSM8K			MATH500			SVAMP			Countdown		
	128	256	512	128	256	512	128	256	512	128	256	512
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
negative TSE reward	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 ATE reward	69.1	77.7	79.5	27.4	32.2	37.2	82.7	85.3	85.0	22.9	23.4	30.5
PA reward	71.3	78.9	79.7	31.0	33.0	37.6	83.0	87.3	86.7	32.0	44.7	41.3
TSE & Accuracy reward	72.1	80.0	83.0	31.2	35.4	41.4	85.0	90.3	92.3	41.5	42.6	54.7
ATE & Accuracy reward	70.2	77.2	80.2	30.8	34.8	38.4	84.7	86.3	85.3	28.1	29.7	35.9
PA & Accuracy reward	71.0	79.2	81.4	32.0	33.8	39.6	85.0	89.7	88.7	36.9	43.4	50.5

answer:

$$\text{ChangeRate} = \frac{1}{T-1} \sum_{t=1}^{T-1} \mathbf{1}[\mathbf{x}_0^t = \mathbf{x}_0^{t+1}] \quad (\text{S2})$$

A higher ChangeRate indicates that the model’s predictions vary more frequently across successive steps, signaling lower temporal consistency. For completeness, we define pairwise agreement (PA) as the complement of ChangeRate, $\text{PA} = 1 - \text{ChangeRate}$, and employ it as an RL reward to promote temporally consistent predictions.

Similar to TSE, we compute the ChangeRate for correct answers across different output lengths, with the results shown in Fig. S5. We observe that, for correctly answered questions, the ChangeRate tends to be lower. However, unlike TSE, ChangeRate does not exhibit significant variation across datasets. That is, whether the model performs well or poorly on a given dataset, the ChangeRate does not show a clear pattern. In contrast, TSE is noticeably lower on easier datasets such as GSM8K and SVAMP compared to more challenging ones like MATH500 and Countdown, as shown in Sec. 3.3.

In addition, we propose using the answer entropy, which is computed from the token distribution of the predicted answer, as a supervisory signal. Although this entropy measure cannot directly supervise temporal consistency in the answer trajectory, it still provides a meaningful indication of how confident the model is in its prediction. As shown in Fig. 3, higher entropy corresponds to lower confidence, while lower entropy suggests that the model is more certain about the generated answer.

As shown in Table S7, we observe that: (1) Using negative token entropy as the reward yields moderate improvement, indicating that local confidence is useful but insufficient. Gains are consistently smaller than with TSE; (2) The pairwise agreement reward performs better than token entropy and comparable to TSE.

Intuitively, PA captures dynamic stability; for example, it can tell that "010101" is less stable than "000111," but it cannot differentiate between semantically diverse outputs such as "010203" and

1242
1243 **Table S8: Ablations on scoring rules for combining TSE with accuracy reward.** We compare 4
1244 different scoring rules, including entropy scoring, quadratic scoring, spherical scoring, and logistic
1245 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
	spherical	72.1	80.0	83.0	31.2	35.4	41.4	85.0	90.3	92.3	41.5	42.6	54.7
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	71.2	79.2	83.4	27.8	36.6	41.2	86.7	90.3	89.0	44.1	45.3	58.6
	quadratic	73.4	79.1	83.2	29.2	33.2	42.4	86.0	86.7	90.0	43.4	45.3	57.4
	logistic	71.2	79.2	83.1	30.2	33.8	41.0	88.3	89.0	91.0	48.8	46.9	61.3
	spherical	73.2	80.5	84.0	29.6	35.4	41.4	86.3	90.3	89.0	44.5	46.9	63.3

1253
1254 ”010101.” In contrast, TSE measures the semantic dispersion of intermediate predictions, capturing
1255 how the meaning of partial answers drifts along the trajectory, but it is less sensitive to fine-grained
1256 dynamical fluctuations among semantically similar states.

1257
1258 We believe an ideal temporal-consistency metric should integrate the complementary strengths of
1259 PA and TSE, leveraging PA’s sensitivity to dynamics and TSE’s sensitivity to semantic variation,
1260 and we plan to explore such hybrid rewards in future work.

1261 E.4 ABLATIONS ON SCORING RULES FOR COMBINING TSE WITH ACCURACY REWARD

1262
1263 In Sec. 4.2, we combine TSE with the accuracy reward using the proper scoring rule (Gneiting &
1264 Raftery, 2007). The purpose of these scoring rules is to promote truthful confidence assessments:
1265 they attain their minimum value when the predicted confidence c precisely mirrors the actual
1266 probability that the model’s output o corresponds to the correct answer o^* . An exception is the
1267 spherical version we use, which consistently encourages both correctness and higher certainty,
1268 without penalizing overconfidence in incorrect predictions. Here, we conduct an ablation study
1269 on different scoring rules. Specifically, we consider the following four forms:

1270 **Entropy scoring:** $r_i^{\text{ent}} = \mathbb{1}_{o_i=o^*} \cdot c(o_i)$

1271 **Quadratic scoring:** $r_i^{\text{quad}} = \mathbb{1}_{o_i=o^*} - (c(o_i) - \mathbb{1}_{o_i=o^*})^2$

1272 **Logistic scoring:** $r_i^{\text{log}} = \mathbb{1}_{o_i=o^*} + \mathbb{1}_{o_i=o^*} \log(c(o_i)) + (1 - \mathbb{1}_{o_i=o^*}) \log(1 - c(o_i))$

1273 **Spherical scoring:** $r_i^{\text{sphe}} = \mathbb{1}_{o_i=o^*} + \frac{c(o_i)}{\sqrt{(c(o_i))^2 + (1 - c(o_i))^2}}$

1274
1275 All four proposed reward functions aim to jointly encourage correctness and temporal self-
1276 consistency. Notably, the r_i^{ent} function gives a reward of 0 for incorrect answers, while for correct
1277 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
1278 a maximum value of 1 when all sampling steps yield the correct answer. The remaining three
1279 functions, r_i^{quad} , r_i^{log} , and r_i^{sphe} , correspond to the commonly used quadratic scoring, logarithmic
1280 scoring, and spherical scoring in proper scoring rules (Gneiting & Raftery, 2007), respectively. We
1281 report the performance of all four reward combination methods in Table S8. By default, we use the
1282 spherical scoring rule because it demonstrates more superior results compared to the alternatives.

1283 E.5 TEMPORAL SELF-CONSISTENCY VOTING AFTER RFT

1284
1285 It is worthwhile to investigate whether Temporal Self-consistency Voting continues to provide
1286 performance benefits after Temporal Consistency Reinforcement. To this end, we conducted
1287 experiments using two models: the first is derived from the LLaDA-8B-Instruct model trained with
1288 the negative TSE reward, and the second is trained using the accuracy reward combined with TSE.

1296

1297 **Table S9: Performance of temporal majority voting after reinforcement learning.** Temporal
 1298 self-consistency voting was applied to the model fine-tuned via temporal consistency reinforcement.
 1299 The upper part is derived from the LLaDA-8B-Instruct model trained using the Negative TSE
 1300 reward, whereas the lower part is based on the model trained with a combination of TSE and
 1301 accuracy reward. For reference, we include the oracle EverPass @1 | t as a performance upper
 1302 bound.

Method / Seq Len	GSM8K			MATH500			SVAMP			Countdown			
	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

1309

1310

1311

1312

1313

1314

1315

1316

1317

1318

1319

1320

1321

1322

We applied Temporal Self-Consistency Voting to both models, and the results are summarized in Table S9. 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.

1327

1328

1329

1330

1331

1332

1333

1334

E.6 EXAMPLES OF TIME OSCILLATION

1335

1336

1337

1338

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.

1342

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.

1346

A notable case is seen in Example 3. Initially, the model correctly calculates the total cups, per-hour revenue, and the per-hour cost deduction to derive the right profit per hour (step 54). However, during reasoning, 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).

1350
1351

Example 1

1352

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?

1353

1354

1355

1356

1357

1358

1359

1360

1361

1362

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**

$$3x = 0 \text{ so } x = 3.$$

</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>

1377

1378

1379

1380

Example 2

1381

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

1382

1383

1384

1385

1386

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 IELa 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 IELa will pay this amount over 3 months, the monthly payment is $\$750 + \$15 = \$750 / 3 = \250 .

</reasoning>

<answer>

250

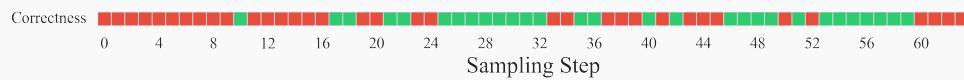
</answer>

1400

1401

1402

1403

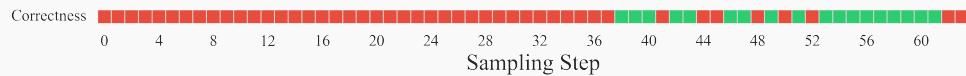


1404
1405

Example 3

1406
1407
1408
1409
1410

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?

1411
1412
1413

1414

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>

...

1424
1425
1426**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>

1435
1436
1437**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>

1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457