Who Should Lead Decoding Now? Tracking Reliable Trajectories for Ensembling Masked Diffusion Language Models

Anonymous ACL submission

Abstract

Masked Diffusion Language Models (MDLMs) have emerged as a distinct paradigm for sequence generation. As MDLMs become diverse in capabilities and knowledge coverage, an important question is how to combine their knowledge. Toward this, we first investigate the unique decoding dynamics of MDLMs. We find that successful generations exhibit stable confidence dynamics over answer-relevant positions, while unreliable trajectories can often be corrected by injecting promising intermediate states from other models. Guided by this observation, we propose **TIE** (**T**rajectory-based **I**terative **E**nsembling), a knowledge fusion framework in which MDLMs iteratively identify reliable decoding trajectories and relay them across models. TIE tracks confidence dynamics over answer-relevant positions to determine which model currently follows a more reliable trajectory and selectively transfers partially denoised sequences across models. As the model on the more promising trajectory often changes across denoising steps, TIE allows different models to contribute complementary strengths at different stages of generation. Strong performance across diverse reasoning tasks, along with our analyses, suggests that TIE offers a practical approach to the underexplored problem of MDLM ensembling.

1 Introduction

Masked Diffusion Language Models (MDLMs) are becoming increasingly compelling alternatives to the autoregressive paradigm. By denoising masked sequences in parallel through an iterative remasking process, MDLMs show competitive sequence generation capabilities across a broad range of domains (Sahoo et al., 2024; Ye et al., 2025b; Nie et al., 2026; Ye et al., 2025a).

As the family of MDLMs continues to diversify with models exhibiting different strengths, training distributions, and decoding dynamics, one question naturally comes up: *how can we effectively orchestrate or fuse knowledge from heterogeneous MDLMs?* This question has grown important in recent years, as users often try different models jointly on their own tasks in search of the best possible results. However, such ensembling strategies for MDLMs remain largely underexplored.

One straightforward approach would be to extend conventional ensemble approaches in autoregressive language models: taking into account the next-token probability distributions, then averaging them (Yu et al., 2024; Xu et al., 2024) or routing toward the more confident model (Shen et al., 2024; Wang et al., 2025b). These approaches, however, are not directly applicable to MDLMs due to their unique decoding dynamics. Since sequences are generated in a flexible, non-left-to-right order, each model may operate on different partially denoised sequences at each step, which makes it difficult to define a shared next-token across models. Such disparities call for knowledge orchestration frameworks specifically designed for MDLMs.

Toward this, we first scrutinize the decoding dynamics of MDLMs to gain insights that guide the design of a knowledge ensemble framework for generating quality-enhanced responses. Specifically, we focus on two perspectives: (i) identifying the more confident model that is likely to produce a correct answer before the full response is generated, allowing it to lead the ensemble process. Our study uncovers that answer-related tokens, even while still masked, tend to follow more stable denoising trajectories (*i.e.*, less fluctuating confidence) when they eventually converge to correct answers. Then, (ii) examining whether the relatively less confident models in the early decoding phase can recover toward correct responses after receiving promising partially denoised sequences from more confident models, thereby allowing them to re-enter subsequent knowledge exchange.

Building upon our findings, we propose **TIE** (**T**rajectory-based **I**terative **E**nsembling), a knowl-

edge fusion framework in which trajectories from more confident models are iteratively relayed to other models so that complementary knowledge from different models can be naturally integrated. Each model monitors the confidence dynamics of answer-related tokens, allowing the framework to identify which model is currently following a more reliable trajectory toward the correct response. Models whose confidence trajectories become unstable are provided with partially denoised sequences from more reliable counterparts and continue generation from those intermediate states.

Through this process, models that deviate from the correct trajectory can be guided back onto promising generation paths when provided with sufficiently reliable intermediate responses, as observed in our analysis. This knowledge transfer process is repeated periodically throughout generation. Interestingly, the model producing the more reliable response frequently changes across denoising steps, suggesting that different models contribute distinct strengths at different stages of generation. Consequently, all participating models collaboratively contribute to refining the final response.

Extensive experiments across diverse domains, including general reasoning, mathematics, coding, and planning, demonstrate that TIE consistently improves over individual MDLMs, highlighting the effectiveness of continual knowledge transfer guided by confidence dynamics over answer-related tokens. Our in-depth analyses further reveal that TIE is most effective when constituent models exhibit both comparable and strong individual capabilities. Overall, our findings provide practical guidelines for effective MDLM ensembling.

2 Preliminaries

Masked diffusion language models. Assume that $\mathbf{x}$ is a clean sequence consisting of a single token, $\mathbf{m}$ is the one-hot representation of the mask index, and z_t denotes the token state at an intermediate noise level t. The forward process in Masked Diffusion Language Models (MDLMs) is defined as follows (Austin et al., 2021a):

$$q(\mathbf{z}_t|\mathbf{x}) = \mathrm{Cat}(\mathbf{z}_t; \alpha_t \mathbf{x} + (1-\alpha_t)\mathbf{m}), \quad (1)$$

where α_t is a predefined noise schedule. For an earlier level $s < t$, posterior distribution $q(\mathbf{z}_s|\mathbf{z}_t)$ can be analytically expressed. If $\mathbf{z}_t \neq \mathbf{m}$, the posterior is deterministic and satisfies

$$q(\mathbf{z}_s \mid \mathbf{z}_t, \mathbf{x}) = \mathrm{Cat}(\mathbf{z}_s; \mathbf{z}_t). \quad (2)$$

Otherwise, when $\mathbf{z}_t = \mathbf{m}$, the posterior becomes

$$q(\mathbf{z}_s \mid \mathbf{z}_t, \mathbf{x}) = \mathrm{Cat}\left(\mathbf{z}_s; \frac{(1-\alpha_s)\mathbf{m} + (\alpha_s - \alpha_t)\mathbf{x}}{1-\alpha_t}\right). \quad (3)$$

Following prior MDLM formulations, the model learns the reverse process by approximating the posterior distribution only on masked positions while unmasked tokens remain unchanged. Accordingly, the posterior transition is parameterized by a neural network f_θ: $p_\theta(\mathbf{z}_s|\mathbf{z}_t) := q(\mathbf{z}_s|\mathbf{z}_t, f_\theta(\mathbf{z}_t, t))$, where f_θ estimates the clean-token distribution conditioned on the noisy state $\mathbf{z}_t$ and diffusion time t. The training objective for a sequence of length L is formulated as the negative evidence lower bound (ELBO):

$$\mathcal{L}_\infty^{\text{ELBO}} = \int_0^1 \frac{\partial_t \alpha_t}{1-\alpha_t} \mathbb{E}_{\mathbf{x}\sim q_0,\, \mathbf{z}_t \sim q_t(\mathbf{z}_t|\mathbf{x})} \left[\sum_{l:\mathbf{z}_t^{(l)}=\mathbf{m}}^{L} \mathbf{x}^{(l)} \cdot \log f_\theta^{(l)}(\mathbf{z}_t, t) \right] dt, \quad (4)$$

where q_0 represents the data distribution, and the summation is taken over masked positions.

Ancestral sampling and unmasking. During the inference phase, the diffusion process is discretized into T denoising steps, with the sequence initialized as fully masked. The model then iteratively denoises the sequence by sampling from the reverse process: $\mathbf{x}_{t-1} \sim p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ for $t = T, \ldots, 1$. Various policies for determining which tokens to unmask have been introduced in prior work, including confidence-based (Kim et al., 2025), thresholding (Wu et al., 2026), and KL-divergence criteria (Kim et al., 2026).

3 Towards Effective MDLM Ensembling

We begin by presenting two observations on MDLM decoding dynamics that provide the key insights underlying our ensembling framework. Motivated by the goal of MDLM ensembling, we specifically focus on *what characterizes high-quality MDLM decoding trajectories* by contrasting correct and incorrect samples. Our analysis reveals two key insights: (i) correct decoding trajectories show more stable and confident answer-token dynamics, and (ii) sharing even a partial portion of such trajectories can steer weaker models toward correct answers.

Analysis setup. The analysis is conducted on MMLU (Hendrycks et al., 2021) and

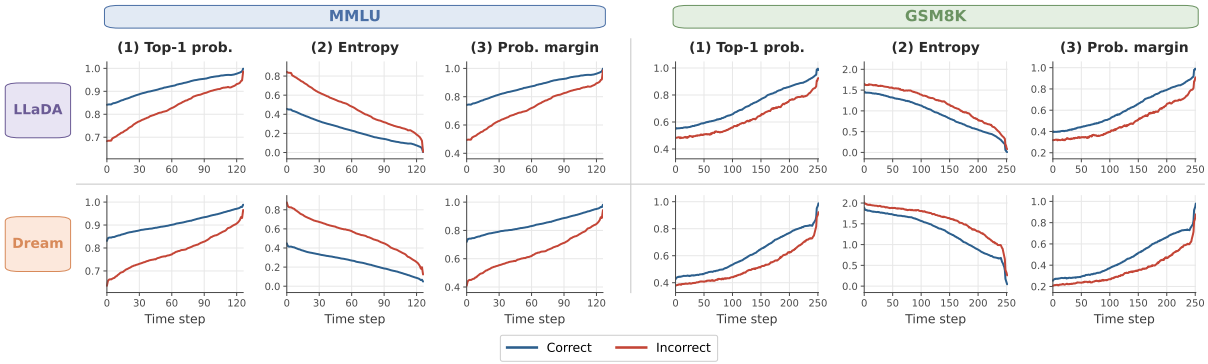


Figure 1: Evolution of three confidence metrics on answer tokens across decoding steps, averaged over answer positions and grouped by correct and incorrect samples. Correct samples consistently show higher confidence across all three metrics (higher top-1 probability/probability margin, lower entropy).

Table 1: Comparison of token change counts $\mathcal{C}^{(T)}$ between correct and incorrect samples.

Models	MMLU		GSM8K	
	Correct	Incorrect	Correct	Incorrect
LLaDA	1.81	4.27	32.29	51.48
Dream	2.32	6.19	40.88	58.07

Table 2: Correction rate (%) under different injection ratios. Each model continues decoding from a partial decoding trajectory generated by another model.

Dataset	Model	Injection ratio	Correction rate
MMLU	LLaDA	33%	56.43
	Dream		68.84
	LLaDA	50%	65.22
	Dream		76.57
GSM8K	LLaDA	33%	74.66
	Dream		78.63
	LLaDA	50%	72.60
	Dream		80.92

GSM8K (Cobbe et al., 2021) using LLaDA-1.5 (Zhu et al., 2025) and Dream-7B-Instruct (Ye et al., 2025b). We employ semi-autoregressive generation with a block size of 16 and low-confidence remasking. The generation length is set to 128 tokens for MMLU and 256 tokens for GSM8K.

3.1 Correct Samples Are More Stable and Confident in Their Answers

Our first observation investigates how the decoding dynamics of answer tokens differ between correct and incorrect samples. Our key question is: *Do correctly answered samples show greater consistency and confidence in their answers during decoding?*

To analyze this, we follow the generation setting of Li et al. (2026), which divides tokens into reasoning and answer tokens, where answer tokens are defined as the tokens appearing after the "Answer:" suffix. We then quantify the stability of answer tokens via the *Token Change Count* ($\mathcal{C}^{(n)}$), defined as the total number of top-1 (*i.e.*, highest-probability) token changes between consecutive decoding steps, accumulated over n decoding steps and the masked answer-token positions at each step t:

$$\mathcal{C}^{(n)} = \sum_{t=T-n+1}^{T-1} \sum_{a \in \mathcal{A}^{(t)}} \mathbf{1}\left[\arg\max \mathbf{p}_a^{(t)} \neq \arg\max \mathbf{p}_a^{(t+1)}\right], \tag{5}$$

where $\mathcal{A}^{(t)}$ denotes the set of masked answer-token positions at step t, T denotes the total number of decoding steps, and $\mathbf{p}_a^{(t)}$ denotes the predicted probability distribution at position a and step t. A lower $\mathcal{C}^{(n)}$ indicates that answer tokens change less frequently throughout decoding, suggesting a more stable decoding behavior.

Table 1 shows that $\mathcal{C}^{(T)}$ of incorrect samples is roughly twice that of correct samples, indicating that incorrect samples exhibit substantially less stability in their answers. For a detailed analysis, we examine how the confidence of answer tokens evolves during decoding using three metrics: (1) top-1 probability, (2) entropy, and (3) probability margin (the gap between the top-1 and top-2 probabilities). As shown in Figure 1, correct samples consistently exhibit higher confidence on answer tokens throughout decoding across all three metrics. These results imply that confidence-based signals over answer tokens may serve as reliable indicators for identifying promising decoding trajectories.

3.2 Reliable Partial Decoding Trajectories Guide Models Toward Correct Answers

Given that promising decoding trajectories can be identified at early stages (Section 3.1), we next ex-

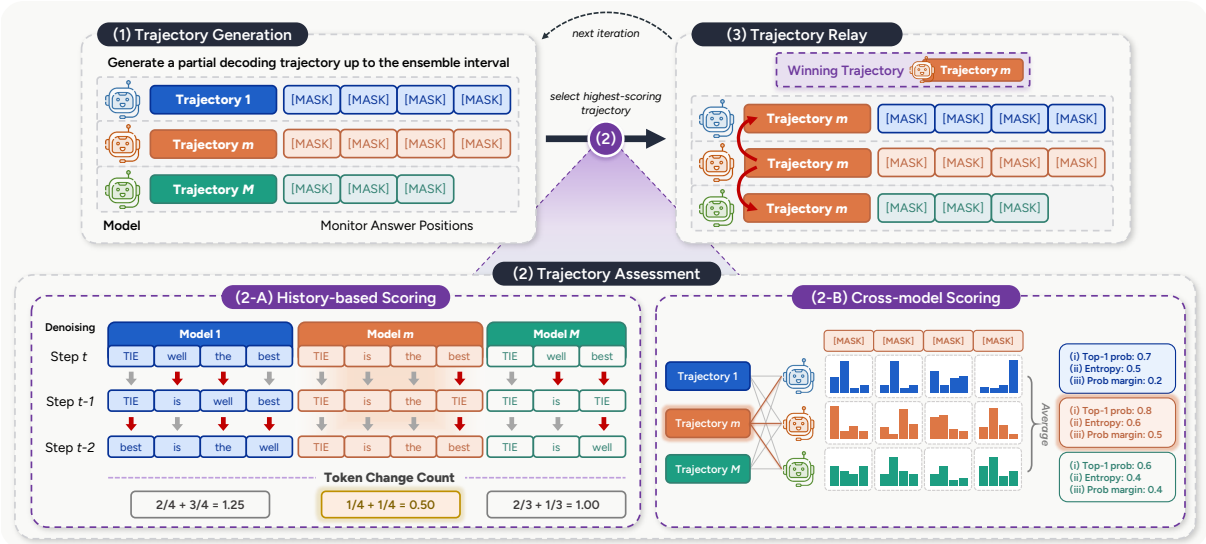


Figure 2: **Overview of TIE.** Each MDLM first independently generates a partial decoding trajectory from its current state. TIE then evaluates these trajectories using confidence-based scoring over answer-token positions, relays the most reliable trajectory across models, and continues decoding from the resulting more reliable intermediate state. In (2-A), the answer-token positions are originally masked, but we display their top-1 tokens for visualization.

amine whether such reliable decoding trajectories from one model can guide subsequent decoding of another toward a correct answer, which could be key to cross-model collaboration during decoding. Specifically, we investigate how often a model that initially generates an incorrect answer can be corrected when given an early portion of a decoding trajectory from another model that produced the correct answer. As shown in Table 2, we observe that a substantial portion of incorrect samples can be corrected even when only one-third of the correct decoding trajectory is provided. These results demonstrate that sharing reliable decoding trajectories among models can rectify generations that would otherwise produce incorrect answers. This motivates us to develop a trajectory-based ensemble framework in which models iteratively correct one another, enabling weaker models to recover rather than be discarded. As a result, different models can contribute their distinct strengths at different stages of generation.

4 TIE: Trajectory-based Iterative Ensembling

Guided by our two observations, we propose **TIE** (**T**rajectory-based **I**terative **E**nsembling), an MDLM-specific ensemble method that combines complementary strengths guided by confidence dynamics at answer-relevant positions. Given M constituent MDLMs, TIE proceeds in a recurring three-step cycle: (i) **Trajectory Generation** (Section 4.1) - each model independently performs unmasking for n decoding steps; (ii) **Trajectory Assessment** (Section 4.2) - each trajectory is scored using confidence-based metrics; (iii) **Trajectory Relay** (Section 4.3) - the highest-scoring trajectory is relayed to all constituent models, replacing their current decoding states to recover models from erroneous or suboptimal trajectories for more effective collaboration in subsequent ensemble steps. This cycle repeats until generation terminates, progressively steering the ensemble toward a more accurate final response.

Notation. Throughout this section, subscripts m index constituent models and superscripts (t) index decoding steps. For example, $\mathcal{T}_m^{(n)}$ denotes the trajectory $\mathcal{T}$ of model m after n steps.

4.1 Trajectory Generation

In each round, every constituent model $m \in [M]$ independently decodes from its current state for n steps, where n denotes the *ensemble interval*, producing a partial decoding trajectory $\mathcal{T}_m^{(n)}$. During decoding, answer-token positions are unmasked only after all reasoning-token positions have been fully unmasked. The effect of ensemble interval is studied in Section 5.3.

4.2 Trajectory Assessment

Given the M partial trajectories $\{\mathcal{T}_m^{(n)}\}_{m=1}^{M}$ from each model, this step assigns a confidence-based score to each trajectory in order to identify the most

reliable one. Following Section 3.1, we consider four scoring metrics: (i) negative token change count, (ii) top-1 probability, (iii) negative entropy, and (iv) probability margin, where higher values indicate greater confidence. All metrics are computed only over masked answer-token positions $\mathcal{A}_m^{(t)}$ to identify the trajectory with the most stable and confident decoding behavior over answer tokens. Each scoring function is applied independently, and their effectiveness is compared in Section 5.2.

History-based scoring for the token change count. The token change count $\mathcal{C}_m^{(n)}$ reflects the history of answer-token stability throughout the generation of $\mathcal{T}_m^{(n)}$. To compute $\mathcal{C}_m^{(n)}$, we track how many masked answer-token positions $\mathcal{A}_m^{(t)}$ change their top-1 tokens across decoding steps during trajectory generation (see Equation (5)). Since $|\mathcal{A}_m^{(t)}|$ may differ across models and decoding steps, we normalize $\mathcal{C}_m^{(n)}$ by $|\mathcal{A}_m^{(t)}|$ to prevent the token change count from inflating simply due to having more answer tokens:

$$\mathrm{Score}\big(\mathcal{T}_m^{(n)}\big) \;=\; -\tilde{\mathcal{C}}_m^{(n)}, \tag{6}$$

where

$$\tilde{\mathcal{C}}_m^{(n)} = \sum_{t=T-n+1}^{T-1} \frac{1}{|\mathcal{A}_m^{(t)}|} \sum_{a \in \mathcal{A}_m^{(t)}} \mathbf{1}\Big[\arg\max \mathbf{p}_a^{(t)} \neq \arg\max \mathbf{p}_a^{(t+1)}\Big]. \tag{7}$$

Cross-model scoring for logit-based metrics. Logit-based scoring functions (*i.e.*, top-1 probability, entropy, and probability margin) directly reflect the confidence over answer tokens for each $\mathcal{T}_m^{(n)}$. Since models can differ in their confidence calibration, we employ cross-model scoring. We forward each $\mathcal{T}_m^{(n)}$ through all constituent models and use the averaged score across models as its final confidence score:

$$\mathrm{Score}\big(\mathcal{T}_m^{(n)}\big) \;=\; \frac{1}{M} \sum_{m'=1}^{M} f\big(\mathcal{T}_m^{(n)}; m'\big), \tag{8}$$

where $f(\cdot; m')$ denotes the scoring function evaluated under the m'-th model. This procedure ensures the selection of a trajectory that is consistently confident across models rather than merely the one favored by its source model.

4.3 Trajectory Relay

The trajectory with the highest score is relayed to all models to recover those that previously produced suboptimal trajectories:

$$m^* \;=\; \arg\max_{m \in [M]} \mathrm{Score}\big(\mathcal{T}_m^{(n)}\big), \tag{9}$$

by replacing the current partial trajectory of each model with $\mathcal{T}_{m^*}^{(n)}$:

$$\mathcal{T}_m^{(n)} \leftarrow \mathcal{T}_{m^*}^{(n)} \quad \forall m \in [M]. \tag{10}$$

After the relay, each model resumes independent decoding for the next n steps. Since the prior decoding histories have been replaced, the trajectory scores are reset accordingly. This cycle of generation, assessment, and relay repeats until any of the constituent models completes decoding.

4.4 Final Response Selection

After decoding terminates, the constituent models produce M candidate responses. We select the final answer as the response that exhibited the most stable answer-token dynamics throughout decoding, *i.e.*, the one with the lowest $\tilde{\mathcal{C}}_m^{(T)}$, breaking ties using the highest top-1 probability. We ablate different selection strategies in Appendix D.

5 Experiments

In this section, we demonstrate the effectiveness of TIE through extensive experiments. We first describe our experimental setup (Section 5.1), then present three key analyses (Section 5.2), and conclude with an ablation study (Section 5.3).

5.1 Experimental Settings

Models. We evaluate our method using four widely-used MDLMs: LLaDA-1.5 (Zhu et al., 2025), Dream-7B-Instruct (Ye et al., 2025b), DreamCoder-7B-Instruct (Xie et al., 2025), and DiffuCoder-7B-Instruct (Gong et al., 2026).

Benchmarks. To assess generalization across diverse domains, we evaluate TIE on eight benchmarks spanning four task categories: (i) *general reasoning*: MMLU (Hendrycks et al., 2021), ARC-Challenge (Clark et al., 2018), and Wino-Grande (Sakaguchi et al., 2021); (ii) *mathematical reasoning*: GSM8K (Cobbe et al., 2021) and MATH500 (Lightman et al., 2024); (iii) *coding*: HumanEval (Chen et al., 2021) and MBPP-sanitized (Austin et al., 2021b); and (iv) *planning*: Countdown[1].

Baselines. Since existing ensemble methods are not directly compatible with MDLMs, we primarily compare our method against the performance of individual models. We further categorize our

[1] https://huggingface.co/datasets/predibase/countdown

Table 3: Benchmark results for ensembling LLaDA and Dream under different scoring functions. Post-generation ensembling corresponds to a special case of TIE where the ensemble interval equals the generation length.

	General				Math		Coding		Planning
Method	MMLU	MMLU* (high-perf.)	ARC-C	WinoGrande	GSM8K	MATH500	HumanEval	MBPP	Countdown
Single Models									
LLaDA	61.23	71.45	85.15	71.59	78.77	37.4	45.73	53.16	13.4
Dream	67.46	77.37	86.69	72.22	78.39	48.0	**61.59**	63.23	16.4
Post-generation Ensemble									
LLaDA + Dream	67.26	77.92	88.57	73.72	80.29	43.0	55.49	62.53	19.2
Intermediate-generation Ensemble									
LLaDA + Dream (Token change count)	**67.55**	**78.12**	**89.16**	72.85	83.47	**48.6**	54.27	**64.17**	18.8
LLaDA + Dream (Top-1 probability)	67.25	77.69	88.82	73.95	82.71	47.0	57.32	62.06	18.8
LLaDA + Dream (Entropy)	67.53	**78.12**	88.57	71.90	**83.62**	45.2	57.93	62.76	**19.4**
LLaDA + Dream (Probability margin)	67.34	77.61	88.74	**73.88**	82.56	48.4	57.32	62.06	18.6

Table 4: Results for ensembling DreamCoder and DiffuCoder on coding benchmarks.

Method	HumanEval	MBPP
Single Models		
DreamCoder-7B-Instruct	**72.56**	75.88
DiffuCoder-7B-Instruct	70.12	72.60
Intermediate-generation Ensemble		
DreamCoder + DiffuCoder (Token change count)	**72.56**	76.11
DreamCoder + DiffuCoder (Top-1 probability)	**72.56**	**76.58**

method according to when ensembling occurs: *post-generation ensemble* and *intermediate-generation ensemble*. Post-generation ensemble selects the final answer from independently generated responses following Section 4.4. This corresponds to a special case of TIE where the ensemble interval equals the total generation length. Intermediate-generation ensemble, in contrast, enables models to collaborate during decoding. We compare four scoring functions for trajectory assessment: token change count, top-1 probability, entropy, and probability margin.

Implementation. We adopt semi-autoregressive generation with a block size of 16 tokens, low-confidence remasking, and greedy decoding. The generation length is set to 128 tokens for general-reasoning tasks, 256 tokens for GSM8K and Countdown, and 512 tokens for MATH500 and coding benchmarks. Except for coding tasks, all models are prompted to provide reasoning before the final answer (Wei et al., 2022). The ensemble interval is set to 16 decoding steps by default, and set to 32 steps when using token change count on general reasoning, coding, and planning. During ensembling, all constituent models perform generation in parallel, resulting in generation latency comparable to that of a single model.

5.2 Main Analysis

(i) TIE selects better decoding trajectories across domains. Table 3 shows that TIE improves over individual models across a wide range of domains, demonstrating that confidence-based scoring over answer tokens can effectively identify high-quality decoding trajectories. Among the four scoring functions, token change count achieves the most robust performance, yielding the best results on four out of eight benchmarks. We attribute this to its ability to track the full decoding history, allowing it to better capture the stability of answer tokens throughout decoding. The benefits of TIE extend beyond LLaDA and Dream: as shown in Table 4, TIE also outperforms individual models when applied to code-specialized MDLMs such as DreamCoder and DiffuCoder.

(ii) TIE is most effective with comparable and strong constituent models. Although TIE generally outperforms individual models, the gains are not uniform across all benchmarks. In particular, when one model substantially underperforms the others, the ensemble becomes less ef-

Table 5: Model change rate (%), defined as the percentage of trajectory-relay steps at which the selected (highest-scoring) model differs from the one selected at the previous step. A high change rate indicates that no single model dominates throughout decoding; instead, different models contribute at different stages.

	General			Math		Coding		Planning
Method	MMLU	ARC-C	WinoGrande	GSM8K	MATH500	HumanEval	MBPP	Countdown
Token change count	12.28	10.55	12.59	23.74	27.80	27.36	21.60	32.04
Top-1 probability	34.25	30.31	40.99	44.80	47.32	28.29	22.83	44.43

fective, as weaker models may introduce noisy signals during trajectory aggregation. For example, on HumanEval, where LLaDA and Dream exhibit a performance gap above 15%, TIE underperforms Dream. We also observe that the gains from TIE increase as the performance of the constituent models improves. TIE achieves especially large improvements on ARC-C and GSM8K, where the individual models already perform well. To further examine this, we define MMLU* as the subset of MMLU subjects on which both LLaDA and Dream achieve over 60% accuracy. On this subset, TIE yields larger improvements than on the full MMLU. We attribute this to the more reliable confidence dynamics of stronger models - they are more accurate in knowing what they know and what they do not - which TIE directly leverages for trajectory assessment. These results suggest that ensembling becomes more effective when constituent models have comparable and strong individual capabilities.

(iii) TIE allows different models to contribute at different decoding stages. One interesting finding is that the model producing the highest-scoring trajectory changes dynamically throughout the ensembling process, rather than a single model consistently leading the generation, as shown in Table 5. Consistent with Section 3.2, this confirms that models initially heading toward incorrect answers can recover after receiving reliable trajectories from other models and contribute high-quality trajectories in later aggregation steps. Consequently, repeated trajectory aggregation during decoding allows TIE to leverage the strengths of different models at different stages, progressively converging toward better final answers. This explains why intermediate-generation ensembling outperforms post-generation ensembling in Table 3.

5.3 Ablation Study

We provide two ablations in this section: (i) the effect of varying the ensemble interval n, and (ii) the compatibility of TIE with existing MDLM decoding acceleration methods. Additional ablations are provided in Appendices B to D.

Robustness across ensemble intervals. Table 6 shows the results under different ensemble intervals n. Across all settings, TIE consistently outperforms individual models on most domains. Nevertheless, setting an appropriate ensemble interval is important. When the ensemble interval is too small, partially decoded trajectories may not contain sufficient information for reliable assessment. Conversely, too large intervals reduce the frequency of trajectory aggregation, limiting knowledge fusion across models. We find that using an interval of 16 steps generally yields the best overall performance.

Compatibility with decoding acceleration. Since MDLMs are commonly combined with acceleration techniques for efficient inference, we further examine whether TIE remains effective when combined with such methods. We consider two standard acceleration techniques that serve as the basis of many existing MDLM inference acceleration methods: (i) *thresholding*, which unmasks all tokens whose top-1 probability exceeds a predefined threshold at each step, and (ii) *top-k unmasking* ($k > 1$), which unmasks the top-k confident tokens per step. As shown in Table 7, TIE remains consistently effective even when combined with these acceleration strategies, continuing to outperform individual models across multiple domains. These results demonstrate that TIE can be readily combined with existing acceleration techniques for more practical deployment.

6 Related Work

Autoregressive Language Models Ensemble. Owing to the prevailing success of autoregressive language models, prior work on LLM ensembling has been developed specifically for the autoregressive generation setting. These approaches can be broadly categorized according to the granularity at which model outputs are aggregated.

Output-level Ensemble methods first elicit com-

Table 6: Ablation on the ensemble interval n where trajectory assessment is based on the token change count.

Method	Ensemble interval	General				Math	
		MMLU	MMLU* (high-perf.)	ARC-C	WinoGrande	GSM8K	MATH500
LLaDA	–	61.23	71.45	85.15	71.59	78.77	37.4
Dream	–	67.46	77.37	86.69	72.22	78.39	48.0
Intermediate-generation Ensemble							
LLaDA + Dream	8	67.50	78.21	89.33	71.90	82.34	46.0
	16	67.43	78.17	89.51	73.32	83.47	48.6
	32	67.55	78.12	89.16	72.85	82.49	46.0

Table 7: Benchmark results under different inference acceleration strategies for MDLMs. (a) *Threshold* unmasks all tokens with confidence above a fixed threshold τ. (b) *Top-k* unmasks the k most confident tokens per step.

(a) Threshold (τ=0.9)

Method	General		Math	Coding
	MMLU*	ARC-C	GSM8K	MBPP
Single Models				
LLaDA	71.34	84.47	80.44	53.63
Dream	77.14	86.69	78.39	**63.00**
Intermediate-generation Ensemble				
LLaDA + Dream (Token change count)	77.56	**89.33**	82.26	**63.00**
LLaDA + Dream (Top-1 probability)	**77.75**	88.23	**83.24**	62.76

(b) Top-k (k=2)

Method	General		Math	Coding
	MMLU*	ARC-C	GSM8K	MBPP
Single Models				
LLaDA	71.25	84.81	78.92	46.37
Dream	76.11	86.26	73.77	53.86
Intermediate-generation Ensemble				
LLaDA + Dream (Token change count)	77.32	89.16	80.14	**58.31**
LLaDA + Dream (Top-1 probability)	**77.88**	**89.93**	**81.58**	55.27

plete responses from each model independently and then aggregate them into a single answer. Early studies explored iterative multi-agent debate (Du et al., 2024; Chen et al., 2024), whereas more recent approaches instead directly fuse independently generated responses. For example, LLM-Blender (Jiang et al., 2023) trains a dedicated fuser to synthesize a final answer, while Si et al. (2023) trains a classifier to select the optimal response. MoA (Wang et al., 2025a) designates one constituent model as an aggregator that consolidates the outputs of the others. Although effective, these methods incur additional inference costs for the fusion stage, cannot integrate models' knowledge during the generation process itself, and typically rely on a large pool of candidate responses.

Span-/Token-level Ensemble methods perform aggregation during generation at finer granularities. Span-level methods (Liu et al., 2025; Xu et al., 2025) iteratively construct the final response by selecting the most promising span (*e.g.*, a sequence of words or tokens) among candidate spans proposed by multiple models, often based on perplexity from other models. Token-level methods (Yu et al., 2024; Xu et al., 2024; Yao et al., 2025; Yun et al., 2026) further refine the aggregation granularity to individual tokens, aggregating next-token probability distributions from multiple models and sampling from the aggregated distribution.

Both span-level and token-level ensemble methods require the next span or token to be at the same position across participating models in order to aggregate them. Consequently, they are not directly applicable to settings in which responses are not generated autoregressively or in which the token generation order varies across models, where the notion of a "next-token" is not well-defined.

7 Conclusion

We introduced *TIE*, a knowledge fusion framework that rethinks how heterogeneous Masked Diffusion Language Models can collaborate. Through continual intermediate exchange guided by confidence dynamics over answer-related tokens, *TIE* allows models to recover from suboptimal trajectories and contribute complementary strengths throughout denoising. Without additional training, *TIE* consistently improves performance across diverse reasoning tasks, highlighting the promise of collaborative inference for diffusion language models. We believe this work takes a meaningful step toward more effective MDLM orchestration.

Limitations

Although TIE has been shown to be effective across various domains and generation settings, several aspects remain open for improvement. First, as discussed in Section 5.2, TIE becomes less effective when the performance gap between constituent models is excessively large (*e.g.*, greater than 15%). This is a common challenge in LLM ensembling (Yao et al., 2025; Yun et al., 2026), and our method could be further strengthened by incorporating mechanisms such as model routing, which selects suitable constituent models prior to ensembling. Second, our experiments are limited to ensembles of two models. Investigating how the effectiveness of MDLM ensembling scales with a larger number of constituent models would be a valuable direction for future work.

References

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. 2021a. Structured denoising diffusion models in discrete state-spaces. In *Advances in Neural Information Processing Systems*.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and 1 others. 2021b. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Justin Chen, Swarnadeep Saha, and Mohit Bansal. 2024. ReConcile: Round-table conference improves reasoning via consensus among diverse LLMs. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7066–7085, Bangkok, Thailand. Association for Computational Linguistics.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the AI2 reasoning challenge. *CoRR*, abs/1803.05457.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. 2024. Improving factuality and reasoning in language models through multiagent debate. In *Forty-first International Conference on Machine Learning*.

Shansan Gong, Ruixiang ZHANG, Huangjie Zheng, Jiatao Gu, Navdeep Jaitly, Lingpeng Kong, and Yizhe Zhang. 2026. Diffucoder: Understanding and improving masked diffusion models for code generation. In *The Fourteenth International Conference on Learning Representations*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.

Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. 2023. LLM-blender: Ensembling large language models with pairwise ranking and generative fusion. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14165–14178, Toronto, Canada. Association for Computational Linguistics.

Jaeyeon Kim, Kulin Shah, Vasilis Kontonis, Sham M. Kakade, and Sitan Chen. 2025. Train for the worst, plan for the best: Understanding token ordering in masked diffusions. In *Forty-second International Conference on Machine Learning*.

Seo Hyun Kim, Sunwoo Hong, Hojung Jung, Youngrok Park, and Se-Young Yun. 2026. KLASS: KL-guided fast inference in masked diffusion models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Pengxiang Li, Yefan Zhou, Dilxat Muhtar, Lu Yin, Shilin Yan, Li Shen, Yi Liang, Soroush Vosoughi, and Shiwei Liu. 2026. Diffusion language model knows the answer before it decodes. In *The Fourteenth International Conference on Learning Representations*.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

Cong Liu, Xiaojun Quan, Yan Pan, Weigang Wu, Xu Chen, and Liang Lin. 2025. Cool-fusion: Fuse large language models without training. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10617–10627, Vienna, Austria. Association for Computational Linguistics.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, JUN ZHOU, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. 2026. Large language diffusion models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

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

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106.

Zejiang Shen, Hunter Lang, Bailin Wang, Yoon Kim, and David Sontag. 2024. Learning to decode collaboratively with multiple language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12974–12990, Bangkok, Thailand. Association for Computational Linguistics.

Chenglei Si, Weijia Shi, Chen Zhao, Luke Zettlemoyer, and Jordan Boyd-Graber. 2023. Getting MoRE out of mixture of language model reasoning experts. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 8234–8249, Singapore. Association for Computational Linguistics.

Chenyu Wang, Paria Rashidinejad, DiJia Su, Song Jiang, Sid Wang, Siyan Zhao, Cai Zhou, Shannon Zejiang Shen, Feiyu Chen, Tommi Jaakkola, Yuandong Tian, and Bo Liu. 2026. SPG: Sandwiched policy gradient for masked diffusion language models. In *The Fourteenth International Conference on Learning Representations*.

Junlin Wang, Jue WANG, Ben Athiwaratkun, Ce Zhang, and James Zou. 2025a. Mixture-of-agents enhances large language model capabilities. In *The Thirteenth International Conference on Learning Representations*.

Ziyao Wang, Muneeza Azmat, Ang Li, Raya Horesh, and Mikhail Yurochkin. 2025b. Speculate, then collaborate: Fusing knowledge of language models during decoding. In *Forty-second International Conference on Machine Learning*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.

Chengyue Wu, Hao Zhang, Shuchen Xue, Zhijian Liu, Shizhe Diao, Ligeng Zhu, Ping Luo, Song Han, and Enze Xie. 2026. Fast-dLLM: Training-free acceleration of diffusion LLM by enabling KV cache and parallel decoding. In *The Fourteenth International Conference on Learning Representations*.

Zhihui Xie, Jiacheng Ye, Lin Zheng, Jiahui Gao, Jingwei Dong, Zirui Wu, Xueliang Zhao, Shansan Gong, Xin Jiang, Zhenguo Li, and 1 others. 2025. Dreamcoder 7b: An open diffusion language model for code. *arXiv preprint arXiv:2509.01142*.

Yangyifan Xu, Jianghao Chen, Junhong Wu, and Jiajun Zhang. 2025. Hit the sweet spot! span-level ensemble for large language models. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 8314–8325, Abu Dhabi, UAE. Association for Computational Linguistics.

Yangyifan Xu, Jinliang Lu, and Jiajun Zhang. 2024. Bridging the gap between different vocabularies for LLM ensemble. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7140–7152, Mexico City, Mexico. Association for Computational Linguistics.

Yuxuan Yao, Han Wu, Mingyang LIU, Sichun Luo, Xiongwei Han, Jie Liu, Zhijiang Guo, and Linqi Song. 2025. Determine-then-ensemble: Necessity of top-k union for large language model ensembling. In *The Thirteenth International Conference on Learning Representations*.

Jiacheng Ye, Jiahui Gao, Shansan Gong, Lin Zheng, Xin Jiang, Zhenguo Li, and Lingpeng Kong. 2025a. Beyond autoregression: Discrete diffusion for complex reasoning and planning. In *The Thirteenth International Conference on Learning Representations*.

Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. 2025b. Dream 7b: Diffusion large language models. *arXiv preprint arXiv:2508.15487*.

Yao-Ching Yu, Chun Chih Kuo, Ye Ziqi, Chang Yucheng, and Yueh-Se Li. 2024. Breaking the ceiling of the LLM community by treating token generation as a classification for ensembling. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 1826–1839, Miami, Florida, USA. Association for Computational Linguistics.

Heecheol Yun, Kwangmin Ki, Jung Hyun Lee, and Eunho Yang. 2026. When to ensemble: Identifying token-level points for stable and fast LLM ensembling. In *The Fourteenth International Conference on Learning Representations*.

Fengqi Zhu, Rongzhen Wang, Shen Nie, Xiaolu Zhang, Chunwei Wu, Jun Hu, Jun Zhou, Jianfei Chen, Yankai Lin, Ji-Rong Wen, and 1 others. 2025. Llada 1.5: Variance-reduced preference optimization for large language diffusion models. *arXiv preprint arXiv:2505.19223*.

A Experimental Details

A.1 Dataset Details

Selected MMLU subjects. In Section 5.2, to examine whether ensembling stronger models yields larger gains, we define MMLU* as the subset of MMLU subjects on which both LLaDA and Dream achieve over 60% accuracy. This subset comprises the following 35 subjects: astronomy, business ethics, clinical knowledge, college biology, college medicine, computer security, conceptual physics, elementary mathematics, high school biology, high school computer science, high school European history, high school geography, high school government and politics, high school macroeconomics, high school microeconomics, high school psychology, high school US history, high school world history, human aging, human sexuality, international law, jurisprudence, logical fallacies, management, marketing, medical genetics, miscellaneous, nutrition, philosophy, prehistory, public relations, security studies, sociology, US foreign policy, and world religions.

Dataset splits. When ground-truth answers are available for the test split, we evaluate on the test split; otherwise, we use the validation split. For MBPP, we use the sanitized version, which filters out low-quality samples.

License. All datasets and models used in the experiments, when accompanied by a license, permit their use for research purposes. Detailed information is provided in their respective references.

A.2 Hardware

When ensembling models with TIE, each model is loaded onto a separate RTX 3090 GPU with bfloat16 precision.

A.3 Prompts

We present the prompt templates used in our experiments. For the multiple-choice and math domains, we adopt the prompt format from `simple-evals`[2]. For Countdown, we follow the template used in Wang et al. (2026). The prompts for HumanEval and MBPP are shown below.

HumanEval.

```
Read the following function signature and
docstring, and fully implement the function
described. Return only the Python function,
no explanation.

{Code}
```

MBPP.

```
{Question}

Your code should satisfy these tests:

{Tests}

Return only the Python function, no explanation.
```

[2] https://github.com/openai/simple-evals

Table 8: Effect of normalizing the token change count by the number of masked answer-token positions.

Method	GSM8K	MATH500
Single Models		
LLaDA	78.77	37.4
Dream	78.39	48.0
Intermediate-generation Ensemble		
TIE w/o norm.	82.64	47.8
TIE	**83.47**	**48.6**

A.4 Answer-token positions

We detail how answer-token positions are defined in our method. We append an `Answer:` suffix to the token sequence, and define the positions preceding the suffix as reasoning positions and those following it as answer positions. For coding tasks, we replace the `Answer:` suffix with ```` ```python ```. During decoding, the answer positions are unmasked only after all reasoning positions have been fully unmasked.

To obtain more reliable confidence dynamics over answer positions, we exclude any answer position whose top-1 token is the end-of-sequence (EOS) token, as such positions are unrelated to the model's actual answer. Moreover, in domains such as code generation, answer sequences can become substantially longer, causing later answer tokens to become noisy and less accurate. We therefore evaluate confidence dynamics only on the first eight masked answer positions in these domains. Consequently, the set of answer-token positions is updated dynamically at each decoding step for each model, as it may vary across both decoding steps and models.

B Effect of Token Change Count Normalization

In the history-based trajectory assessment (Section 4.2), we use the normalized token change count $\tilde{\mathcal{C}}_m^{(n)}$, which normalizes $\mathcal{C}_m^{(n)}$ by the number

Table 9: Comparison between TIE with and without cross-model scoring. *TIE w/o cross-model scoring* (*i.e.*, source-model-only scoring) evaluates trajectories solely using confidence scores from their source models.

	General		Math	
Method	MMLU*	ARC-C	GSM8K	MATH500
TIE w/o cross-model scoring	77.28	88.23	81.35	42.4
TIE	**77.69**	**88.82**	**82.71**	**47.0**

of masked answer-token positions $|\mathcal{A}_m^{(t)}|$, instead of directly using $\mathcal{C}_m^{(n)}$. This normalization is important because $|\mathcal{A}_m^{(t)}|$ can differ across models, and using $\mathcal{C}_m^{(n)}$ without accounting for this difference may fail to reflect answer-token stability accurately. In particular, a model with a larger $|\mathcal{A}_m^{(t)}|$ may naturally exhibit a higher $\mathcal{C}_m^{(n)}$ simply due to having more answer-token positions, rather than due to genuinely unstable decoding dynamics. Table 8 shows the effect of normalizing $\mathcal{C}_m^{(n)}$. As shown in the table, compensating for differences in $|\mathcal{A}_m^{(t)}|$ leads to improved performance.

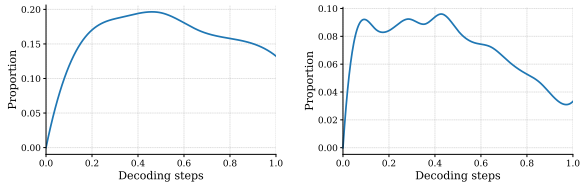
C Effect of Cross-Model Scoring

In this section, we discuss why cross-model scoring is important when using logit-based scoring functions (*i.e.*, top-1 probability, entropy, and probability margin). Since different models are calibrated differently, directly comparing their logits may lead to biased trajectory comparisons. For example, if one model tends to be overly confident in its answers, logit-based scoring functions may disproportionately favor its trajectories regardless of their actual reliability. Therefore, accounting for calibration differences across models is essential for robust trajectory comparison.

To address this, we employ cross-model scoring, which evaluates a given trajectory not only under its source model but across all constituent models, selecting the trajectory that exhibits the highest confidence on average. As shown in Table 9, cross-model scoring outperforms source-model-only scoring by favoring trajectories that are consistently supported by all constituent models. These results suggest that accounting for calibration differences is important when scoring trajectories using logit-based scoring functions.

D Ablation on Final Response Selection Strategies

There is no trajectory relay after the final trajectory generation step since all answer-token positions have been unmasked. Therefore, an additional strategy is required to select the final response among the M candidate responses. We explore three final response selection strategies: (i) *lowest TCC*, which selects the response with the lowest $\tilde{\mathcal{C}}_m^{(T)}$; (ii) *best model*, which selects the response from the best-performing individual model (Dream in our setup); and (iii) *most selected*, which selects the response from the model that produced the highest-scoring trajectory most frequently during ensembling. As shown in Table 10, all three strategies achieve strong performance. However, in domains with a large performance gap between models, the *best model* strategy performs particularly well. This indicates that when a clearly superior model exists, simply selecting its final response is effective. In contrast, when constituent models exhibit comparable performance, selecting the response with the lowest $\tilde{\mathcal{C}}_m^{(T)}$ produces better results. Overall, these results suggest that the optimal final response selection strategy may depend on the relative performance gap between constituent models.



(a) MMLU (b) GSM8K

Figure 3: Visualization of when the ensemble-leading model changes during decoding. The y-axis represents the proportion (%) of decoding steps in which the highest-scoring model changes.

E When Does the Ensemble-Leading Model Change?

Table 5 shows that the model producing the highest-scoring trajectory changes frequently throughout generation. One natural question is at which stages of decoding these changes primarily occur. As shown in Figure 3, we observe that the ensemble-leading model changes more frequently during the

Table 10: Ablation on the final response selection strategy. Token change count is used for trajectory assessment. We compare three strategies: (i) *lowest TCC*, selecting the response with the lowest $\tilde{\mathcal{C}}_m^{(T)}$ (Section 4.4); (ii) *best model*, selecting the response from the best-performing individual model; and (iii) *most selected*, selecting the response from the model chosen most frequently during ensembling.

	General		**Math**		**Coding**
Method	MMLU*	ARC-C	GSM8K	MATH500	MBPP
Single Models					
LLaDA	71.45	85.15	78.77	37.4	53.16
Dream	77.37	86.69	78.39	48.0	63.23
Intermediate-generation Ensemble					
(i) Lowest TCC	78.12	**89.16**	**83.47**	48.6	64.17
(ii) Best model	**78.64**	88.23	81.65	**49.0**	**66.04**
(iii) Most selected	78.06	88.82	83.17	48.2	63.00

early decoding stages. This enables knowledge exchange among models to occur primarily during the early decoding steps, before a clear answer trajectory has fully emerged, allowing TIE to progressively converge toward higher-quality responses.

F Qualitative examples

This section presents qualitative examples of TIE. Tables 11 and 12 compare the responses generated by individual models with those produced by ensembling the same models via TIE.

Question	A merchant wants to make a choice of purchase between 2 purchase plans: jewelry worth $5,000 or electronic gadgets worth $8,000. His financial advisor speculates that the jewelry market will go up 2.5% while the electronic gadgets market will rise 1.2% within the same month. If the merchant is looking to maximize profit at the end of this month by making a choice, how much profit would this be?
Answer	125
LLaDA	To determine the potential profit for the merchant, we need to calculate the future value of the jewelry and electronic gadgets after the market increases and then find the difference. 1. Calculate the future value of the jewelry: - Current value of jewelry: $5,000 - Market increase: 2.5% - Future value of jewelry: $5,000 * (1 + 0.025) = $5,000 * 1.025 = $5,125 2. Calculate the future value of the electronic gadgets: - Current value of electronic gadgets: $8,000 - Market increase: 1.2% - Future value of electronic gadgets: $8,000 * (1 + 0.012) = $8,000 * 1.012 = $8,096 3. Calculate the profit: - Profit: Future value of electronic gadgets - Future value of jewelry = $8,096 - $5,125 = $71 Answer: 71
Dream	1. Calculate the potential increase for the jewelry market: - Initial value of jewelry: $5,000 - Increase percentage: 2.5% - Increase: $5,000 * 0.025 = $125 - New value: $5,000 + $125 = $5,125 2. Calculate the potential increase for the electronic gadgets market: - Initial value of electronic gadgets: $8,000 - Increase percentage: 1.2% - Increase: $8,000 * 0.012 = $96 - New value: $8,000 + $96 = $8,096 3. Compare the new values to determine which purchase plan yields a higher profit: - Jewelry market: $5,125 - Electronic gadgets market: $8,096 4. Calculate the profit for the electronic gadgets market: - Profit: New value - Cost = $8,096 - $8,000 = $96 Answer: 96
TIE (LLaDA + Dream)	1. Calculate the potential increase for the jewelry market: - Initial value of jewelry: $5,000 - Increase percentage: 2.5% - Increase: $5,000 * 0.025 = $125 - New value: $5,000 + $125 = $5,125 2. Calculate the potential increase for the electronic gadgets market: - Initial value of electronic gadgets: $8,000 - Increase percentage: 1.2% - Increase: $8,000 * 0.012 = $96 - New value: $8,000 + $96 = $8,096 3. Determine the profit for each option: - Profit from jewelry: $5,125 - $5,000 = $125 - Profit from electronic gadgets: $8,096 - $8,000 = $96 4. Compare the profits: - Profit from jewelry is higher than electronic gadgets. Answer: 125

Table 11: Qualitative examples of TIE.

Question	The vending machines sell chips for 40 cents and candy bars for 75 cents. George spent $5 and got 3 bags of chips and had 1% of his money left. How many candy bars did he buy?
Answer	5
LLaDA	1. George spent $5 in total. 2. He bought 3 bags of chips at 40 cents each, which is 3 * 40 = 120 cents. 3. He had 1% of his money left, which is 1% of $5, or 0.01 * 500 = 5 cents. 4. Therefore, the total amount spent on chips and candy bars is 500 - 5 = 385 cents. 5. The amount spent on chips is 120 cents, so the amount spent on candy bars is 385 - 120 = 265 cents. 6. Each candy bar costs 75 cents, so the number of candy bars George bought is 265 / 75 = 3.5333... candy bars. 7. Since George can't buy a fraction of a candy bar, we need to find the whole number of candy bars he bought. 8. Rounding 3.5333... to the nearest whole number, we get that George bought 3 candy bars. Answer: $\boxed{3}$
Dream	1. George spent $5 and got 3 bags of chips. Each bag of chips costs 40 cents, so 3 bags of chips cost 3 * 40 = 120 cents. 2. George had 1% of his money left after buying the chips. To find out how much money he had left, we need to find 1% of $5. 1% of $5 is 0.01 * 5 = 0.05 dollars. 3. George spent $5 - 0.05 = $4.95 on candy bars. 4. Each candy bar costs 75 cents. To find out how many candy bars George bought, we need to divide the amount he spent on candy bars by the cost of one candy bar. $4.95 / 0.75 = 6.6. 5. Since George can't buy a fraction of a candy bar, we need to round down to the nearest whole number. So, George bought 6 candy bars. 6. However, we need to find the number of candy bars he bought, not the amount he spent. So, the answer is 6. Answer: $\boxed{6}$
TIE (LLaDA + Dream)	1. George spent $5 and got 3 bags of chips. Each bag of chips costs 40 cents, so the total cost for 3 bags is 3 * 40 = 120 cents. 2. George had 1% of his money left. Since $5 is equivalent to 500 cents, 1% of 500 cents is 0.01 * 500 = 5 cents. So, George had 5 cents left. 3. The total amount George spent is $5, which is equivalent to 500 cents. He spent 120 cents on chips and had 5 cents left, so the amount spent on candy bars is 500 - 120 - 5 = 375 cents. 4. Each candy bar costs 75 cents. To find out how many candy bars George bought, divide the total amount spent on candy bars by the cost of one candy bar: 375 / 75 = 5. 5. Therefore, George bought 5 candy bars. The final answer is 5, so the format of the final answer would be: Answer: $\boxed{5}$

Table 12: Qualitative examples of TIE.