LATENT-DARM: BRIDGING DISCRETE DIFFUSION AND AUTOREGRESSIVE MODELS FOR REASONING

Lina Berrayana[1†‡], **Ahmed Heakl**[2†], **Muhammad Abdullah Sohail**[2], **Thomas Hofmann**[3], **Salman Khan**[2], **Wei Chen**[4‡]
[1]EPFL [2]MBZUAI [3]ETH Zürich [4]Microsoft Research Asia

ABSTRACT

Most multi-agent systems rely exclusively on autoregressive language models (ARMs) that are based on sequential generation. Although effective for fluent text, ARMs limit global reasoning and plan revision. On the other hand, Discrete Diffusion Language Models (DDLMs) enable non-sequential, globally revisable generation and have shown strong planning capabilities, but their limited text fluency hinders direct collaboration with ARMs. We introduce **Latent-DARM**, a latent-space communication framework bridging DDLM (planners) and ARM (executors), maximizing collaborative benefits. Across mathematical, scientific, and commonsense reasoning benchmarks, Latent-DARM outperforms text-based interfaces on average, improving accuracy from 27.0% to 36.0% on DART-5 and from 0.0% to 14.0% on AIME 2024. Latent-DARM approaches the results of state-of-the-art reasoning models while using less than 2.2% of its token budget. This work advances multi-agent collaboration among agents with heterogeneous models.

1 INTRODUCTION

Collaborative interactions between models are increasingly recognized as a fundamental driver of system-level intelligence in the era of agentic AI (Acharya et al., 2025; Guo et al., 2024). Advances in multi-agent systems (MAS) research (Hong et al., 2023; Chen et al., 2023; Wu et al., 2024) have shifted the prevailing paradigm away from isolated, single-model reasoning toward intelligence that emerges through coordination among multiple agents, often with complementary and specialized roles. Within this setting, MAS built on large language models (LLMs) have demonstrated strong performance in a wide range of applications, such as collaborative reasoning in mathematics and scientific problem solving (Karbasi et al., 2025; Zhang & Xiong, 2025), as well as common-sense reasoning (Panzarasa et al., 2002).

Despite this progress, most existing MAS rely exclusively on autoregressive language models (ARMs), which generate output token by token in a strictly sequential manner. As a consequence, every stage of the agentic workflow remains inherently autoregressive. In contrast, discrete diffusion language models (DDLMs) (Sahoo et al., 2024; Gat et al., 2024; Shi et al., 2024) have recently attracted increasing attention, driven by evidence that they can outperform ARMs on complex reasoning and planning tasks (Ye et al., 2024b). Therefore, relying solely on ARMs may constrain the full potential of MAS, particularly for tasks that require flexible planning or global reasoning.

However, a key limitation remains: DDLMs still lag behind ARMs in terms of text fluency. This gap can hinder effective communication between the two agents, particularly when the DDLM-generated output lacks sufficient linguistic coherence. This observation naturally raises the following question:

> *How to take advantage of DDLMs and ARMs properties while optimizing communication between the two models ?*

This work presents a preliminary empirical investigation of this question, as illustrated in Fig. 1. We introduce **Latent-DARM** (**Latent**- Discrete **D**iffusion and **A**uto**R**egressive **M**odel Communication), a communication framework that bridges DDLMs and ARMs operating within the latent space. For

[†] Equal contribution.
[‡] Corresponding authors: weic@microsoft.com, lina.berrayana@epfl.ch

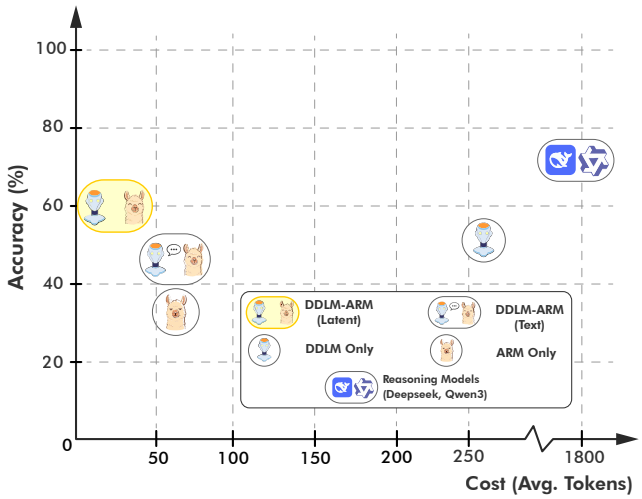

Figure 1: Accuracy–cost trade-offs across planner–executor configurations. Here DDLM refers to a LLada-8B-Instruct and ARM to a Llama-3.2-3B-Instruct model. DDLM→ARM, particularly with latent-space exchange, achieves higher reasoning accuracy at lower token budgets compared to ARM-only models.

that, we study a planner–executor framework in which a DDLM generates a solution plan for a given problem and an autoregressive model (ARM) executes the plan to produce the final answer. We empirically evaluated the effectiveness of this latent-space communication against traditional text-based collaboration across benchmarks in mathematical and scientific reasoning, as well as commonsense understanding.

Why consider a planner–executor framework? First, this setting is quite common in multi-agent systems (He et al., 2025). Additionally, key advantage of DDLMs is their ability to generate tokens in arbitrary orders, enabling non-sequential and bidirectional planning that more closely aligns with human reasoning processes. In mathematical problem solving, for instance, humans often reason backward from the desired solution, iteratively refining intermediate steps before presenting a fully sequential proof. Similarly, commonsense reasoning proceeds non-linearly, with salient considerations identified prior to explicit verbalization. Once such a plan is formed, it is subsequently articulated in a sequential linguistic form. Motivated by this analogy, we adopt a planner–executor multi-agent framework as our experimental setting to investigate the central research question.

Contributions. Our main contributions are twofold:

- We provide empirical insights into collaborations between DDLMs and ARMs in a MAS context.
- We introduce a latent-based communication framework to improve DDLMs and ARMs integration performance. To our knowledge, we introduce the first latent-space communication solution designed to bridge models with fundamentally different architectures and latent representations.

Organization. We first review the relevant definitions and preliminaries and motivate our work in Section 2. In Section 3, we introduce Latent-DARM and describe how it addresses the challenge of bridging DDLMs and ARMs. In Section 4, we present the experimental details. Finally, we discuss the results in Section 5 and show that the observed accuracy improvements stem from a better processing of the plan.

2 PRELIMINARIES

2.1 DEFINITIONS

In our framework, we define two complementary roles: the *planner* and the *executor*.

Planner: Planner is a language model that is responsible for generating intermediate outputs—such as plans, hints, or key facts—that guide the reasoning process without directly producing the final answer. This phase corresponds to the "thinking" stage, distinct from the final response generation. Following the *Plan-and-Solve* prompt method (Wang et al., 2023), which improves reasoning performance, the planner formulates the solution plan.

Executor: Executor is a language model that generates the final answer leveraging both the original query and the planner's output, without engaging in further explicit reasoning.

In the multi-agent system (MAS) context, we refer to the planner as *Agent 1* and the executor as *Agent 2*.

2.2 MOTIVATION

In this work, we focus on DDLM, specifically Masked Discrete Diffusion Language Models (Sahoo et al., 2024) as *Agent 1* and ARMs as *Agent 2*.

DDLMs for planning. Through their iterative denoising process, DDLMs enable flexible, non-sequential token generation, allowing the model to condition on global context when constructing sequences. This property makes diffusion-based models particularly attractive for planning and structured reasoning tasks (Ye et al., 2024a). In contrast, ARMs, trained to generate sequences via prediction of next-tokens from left-to-right, force decisions to be made based on prefixes generated previously, restricting the model's ability to review previous choices or reason over the global structure. As shown by Bachmann & Nagarajan (2024), ARMs can learn spurious heuristics that exploit prefix-level shortcuts rather than internalizing the underlying planning dynamics, thereby limiting their effectiveness in certain planning tasks.

Recent work on hybrid approaches provides encouraging evidence for the collaboration between these paradigms. Arriola et al. (2025) introduce Block Diffusion models that interpolate between autoregressive and diffusion generation, achieving state-of-the-art performance among diffusion models while supporting flexible-length generation. Their results demonstrate that combining the strengths of both approaches –sequential coherence from ARMs and parallel generation from diffusion—can overcome the limitations of either method alone.

Fluency Problem. However, a key limitation of DDLMs that must be addressed in such collaborative frameworks is output fluency, as disfluent outputs can degrade the quality of the messages sent to the next agent. For example, perplexity, which is often used as a proxy for fluency Feng et al. (2025), is generally higher for DDLMs, indicating worse fluency, compared to state-of-the-art ARMs Sahoo et al. (2024). Let $\mathbf{x} \in \{1, \ldots, V\}^L$ denote a sequence of length L in vocabulary V. During training, DDLMs learn to model the **unmasking posterior** $p_\theta(x_i = \cdot \mid z)$ for each masked position i in a partially masked sequence z. The masking process samples $t \sim \text{Unif}[0, 1]$ and replaces each token x_i with a mask token m independently of probability t, creating a corrupted sequence z. The model p_θ is trained to minimize:

$$\mathbb{E}_{\mathbf{x} \sim q_{\text{data}}, t, z} \left[\sum_{i : z_i = m} - \log p_\theta(x_i \mid z) \right], \tag{1}$$

where the expectation is over data sequences $\mathbf{x} \sim q_{\text{data}}$, mask ratios t, and masking patterns z (Sahoo et al., 2024; Shi et al., 2024). Generation reverses this corruption: starting from fully masked input, the model iteratively unmasks tokens through multiple denoising steps, until producing clean output. However, tokens revealed simultaneously at a given denoising step are predicted **independently** given only the current unmasked context, which can undermine fluency Feng et al. (2025). Recent work has sought to address this issue by deriving optimal unmasking schedules that minimize sampling error through connections to function approximation theory Chen et al. (2025), or by

introducing self-correction mechanisms for diffusion-based generation von Rütte et al. (2025); Kim et al. (2025). Our work takes a **complementary approach**: rather than solely addressing fluency through improved diffusion techniques, we propose a multi-agent framework where DDLMs generate high-level plans leveraging their global reasoning capabilities, while ARMs execute these plans with sequential fluency.

Remark. This division of labor mirrors **human cognitive processes** –internal flexible thinking followed by sequential articulation—and allows each model to operate within its strengths while mitigating its weaknesses.

2.3 Problem Formulation

Latent-Space Communication. Latent-space reasoning has recently gained attention as a promising alternative to text-space reasoning (Hao et al., 2024; Zhu et al., 2025b), which is inherently limited by the constraints of natural language. In text space, the reliance on discrete tokens and natural language syntax restricts the expressive bandwidth of models. Recent work (Zhu et al., 2025a) demonstrates that latent representations encode richer information than discrete token outputs. For instance, the Chain of Continuous Thought framework (Hao et al., 2024), which performs multi-step inference within the continuous hidden state space, achieves improved accuracy by enabling more informative intermediate representations. Motivated by these advances, we investigate latent representations as a communication medium between agents. Specifically, we explore continuous latent communication to enhance inter-agent collaboration between DDLMs and ARMs.

Formally, consider two agents in a MAS, where the output of *Agent 1* is fed as input to *Agent 2*. Let the hidden state in step t in *Agent 1* be denoted by $\mathbf{h}_t^{(1)} \in \mathbb{R}^d$, where d is the embedding dimension. In traditional ARMs, recent approaches (Hao et al., 2024) use the final hidden state $\mathbf{h}_T^{(1)}$ as input embedding to predict the $(T+1)$-th token. In a MAS setting, this would be equivalent to passing $\mathbf{h}_T^{(1)}$ as an input embedding to *Agent 2*, i.e.,

$$\mathbf{x}_0^{(2)} = \mathbf{h}_T^{(1)},$$

where $\mathbf{x}_0^{(2)}$ is the initial input embedding for *Agent 2*. However, as we explain next, this straightforward approach is not feasible.

Challenge : Embedding Space Mismatch. A key challenge arises in directly passing $\mathbf{h}_T^{(1)}$ from the DDLM (*Agent 1*), which is trained bidirectionally through masked denoising, to the ARM (*Agent 2*), which is trained unidirectionally in an autoregressive manner. These fundamentally different training paradigms result in different embedding spaces. Specifically, the hidden representations $\mathbf{h}_T^{(1)}$ and $\mathbf{h}_0^{(2)}$ lie in separate latent manifolds:

$$\mathbf{h}_T^{(1)} \in \mathcal{H}_{\text{DDLM}} \subseteq \mathbb{R}^{d_1}, \quad \mathbf{h}_0^{(2)} \in \mathcal{H}_{\text{ARM}} \subseteq \mathbb{R}^{d_2},$$

with typically $d_1 \neq d_2$ and differing geometric and statistical properties. Consequently, a direct assignment $\mathbf{x}_0^{(2)} = \mathbf{h}_T^{(1)}$ is not feasible, motivating the need for a learned or engineered mapping between these heterogeneous latent spaces to enable effective inter-agent communication.

To address this, we propose learning a dedicated projection network (Figure 2) that maps latent representations from the DDLM space to the ARM space, thereby allowing meaningful translation and collaboration despite architectural and representational heterogeneity.

3 Latent-DARM

3.1 System Architecture

We compare the following two planner–executor integration schemes.

Text-Space Interface (Baseline). The conventional planner–executor interface operates in text space, where the planner's latent representation is first decoded into a discrete sequence and then

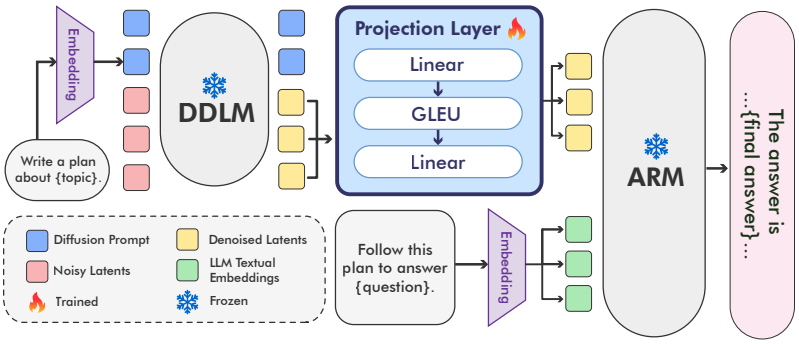

Figure 2: Overview of the latent-space collaboration pipeline. A discrete diffusion language model (DDLM) generates a latent plan. The plan is projected directly into the autoregressive model (ARM) embedding space through a learned projector (in blue). The ARM then conditions on the plan and the question to produce the final answer.

re-encoded by the executor:

$$h_{\text{DDLM}} \xrightarrow{\pi_{\text{decode}}} T \xrightarrow{\phi_{\text{encode}}} h_{\text{ARM}}. \tag{2}$$

Latent-DARM (Proposed). In contrast, we propose a direct latent-space interface that bypasses explicit text generation:

$$h_{\text{DDLM}} \xrightarrow{f_\theta} h_{\text{ARM}}, \tag{3}$$

where $f_\theta : \mathcal{H}_{\text{DDLM}} \to \mathcal{H}_{\text{ARM}}$ is a learned projection module implemented as a Linear–GELU–Linear network, as shown in Figure 2. The projection is trained to align the planner and executor representation spaces, enabling information transfer without intermediate discretizations.

Remark. Here, the only trainable component is the projector, which learns to map latents from the last hidden layer after the final denoising step of the DDLM (*Agent 1*) to an input embedding. This embedding is then concatenated with the prompt embedding of the *Agent 2* ARM. It is important to note that we do *not* perform any fine-tuning of the agents themselves.

3.2 Projector Training

We train the projection module f_θ while keeping both the DDLM planner and the ARM executor frozen. A natural but problematic approach would be to directly align the projected planner representation with an "ideal" executor representation via a distance-based objective,

$$\min_\theta \ \mathbb{E}\big[\|f_\theta(h_{\text{DDLM}}) - h^*_{\text{ARM}}\|_2^2\big], \tag{4}$$

where h^*_{ARM} denotes a target embedding in the executor space. However, such an objective is ill-defined in practice: the executor does not admit a unique canonical hidden state corresponding to a correct solution. Furthermore, defining h^*_{ARM} via text-based execution would reintroduce the very discretization bottleneck that our latent interface is designed to avoid.

As an alternative to this approach, which is one of the key contributions of this work, we adopt a training objective task-based that optimizes the projection indirectly through downstream performance. Concretely, given a dataset of reasoning tasks comprising question-answer pairs, for each input question q with ground-truth answer a, we extract the latent representation of the planner $h_{\text{DDLM}}(q)$ from the final denoising step. This latent state is then projected through f_θ and concatenated with the embedding of the encoded question to condition the frozen ARM executor. The projector is trained by minimizing the negative log-likelihood of the correct answer:

$$\min_\theta \ \mathbb{E}_{(q,a)\sim\mathcal{D}}\big[-\log p_{\text{ARM}}\big(a \mid f_\theta(h_{\text{DDLM}}(q)), q\big)\big]. \tag{5}$$

Rather than imposing geometric proximity to an ill-defined target embedding, it encourages f_θ to map planner latents into regions of the executor's representation space that induce similar downstream behavior. In other words, the projection is optimized to preserve functional equivalence with

respect to the task, rather than geometric similarity in the embedding space. Through backpropagation, the gradients in the executor's output implicitly shape the projection to achieve optimal task alignment, without requiring intermediate text generation or oracle supervision. The training procedure for the projector is described in Algorithm 1.

Algorithm 1 Latent-DARM Training

Input: data $\mathcal{D}$, planner θ_{DDLM}, executor θ_{ARM}, projection f_θ, rate η, epochs E, size B

Initialize θ randomly
for each epoch **do**
  for each batch **do**
    for i in batch **do**
      $h_i^{\text{DDLM}} = \text{DDLM}_{\text{planner}}(q_i)$
      $h_i^{\text{proj}} = f_\theta(h_i^{\text{DDLM}})$
      $h_i^{\text{ARM}} = [h_i^{\text{proj}}; \text{embed}_{\text{ARM}}(q_i)]$
    $\mathcal{L} = -\frac{1}{B}\sum_{i=1}^{B} \log p_{\text{ARM}}(a_i \mid h_i^{\text{ARM}})$
    $\theta = \theta - \eta \nabla_\theta \mathcal{L}$

3.3 Projector Inference and Evaluation

The projector's inference procedure is detailed in Algorithm 2. We conduct experiments on the benchmarks described in Section 4.

Algorithm 2 Latent-DARM Inference

Input: query q, projection f_{θ^*}, planner, executor

$h^{\text{DDLM}} = \text{DDLM}_{\text{planner}}(q)$
$h^{\text{proj}} = f_{\theta^*}(h^{\text{DDLM}})$
$h^{\text{input}} = [h^{\text{proj}}; \text{embed}_{\text{ARM}}(q)]$
$a = \text{ARM}_{\text{executor}}(h^{\text{input}})$
return a

4 Experimental Evaluation

4.1 Models and Benchmarks

DDLMs. We use two DDLMs, LLada-8B-Instruct (Nie et al., 2025) and Dream-v0-Instruct-7B (Ye et al., 2025). The default sequence length is set to 128 tokens, which provides sufficient capacity for plan generation while reducing repetition errors (see Appendix A).

ARMs. Non-reasoning models: We consider two ARMs of comparable scale—Qwen2.5-7B-Instruct (Yang et al., 2024) and Llama-3.1-8B-Instruct (Touvron et al., 2023; Dubey et al., 2024)—for fair comparison. We also include smaller variants, Llama-3.2-3B-Instruct and Qwen2.5-3B-Instruct (Yang et al., 2024), to assess performance in lower-capacity regimes. **Reasoning models:** We also evaluate Qwen3-1.7B (Yang et al., 2025) and DeepSeek-R1-Distill-Qwen-7B (Yang et al., 2024), a distilled variant that compresses the reasoning capabilities of DeepSeek-R1 into a Qwen-7B backbone. For simplicity, we refer to this model as the *DeepSeek* model. Our goal is not to match specialized reasoning systems, but to contextualize hybrid ARM–DDLM collaboration against strong reasoning baselines.

Benchmarks. We evaluate on a diverse suite of reasoning benchmarks: ARC-E and ARC-C (Clark et al., 2018), science exam questions at Easy and Challenging difficulty; MMLU (Hendrycks et al., 2021b;a), which spans mathematics, history, computer science, and law; and AIME 2024 (Veeraboina, 2024; Jia, 2024), a high-school mathematics competition. We also include DART-1 through DART-5 (Tong et al., 2024), a large-scale mathematical reasoning benchmark covering five difficulty levels. These benchmarks are standard in the literature and together provide broad coverage of

Table 1: Evaluation of DeepSeek-R1-Distill-Qwen-7B and Qwen3-1.7B on reasoning benchmarks, including text-space vs. latent-space collaboration.

Model / Setting	ARC-E	ARC-C	DART-1	DART-2	DART-3	DART-4	DART-5	AIME24	MMLU
Accuracy									
DeepSeek-R1 (ARM only)	94.0	88.0	89.0	81.5	85.5	75.0	61.5	28.5	60.0
Qwen3-1.7B (ARM only)	92.0	86.0	89.5	86.0	83.0	73.0	49.0	8.5	68.5
Llama-3.2-3B (ARM only)	87.5	79.5	44.5	35.5	28.0	34.5	36.5	3.5	54.0
LLaDA-8B → Llama-3.2-3B (text-space)	90.5	82.5	53.5	43.0	35.5	30.0	27.0	0.0	52.5
LLaDA-8B → Llama-3.2-3B (latent-space)									
64 tokens plan	85.0	78.5	78.5	62.5	57.0	63.0	54.0	12.5	52.0
128 tokens plan	87.5	76.5	70.5	43.0	43.0	49.0	36.0	14.0	44.0
256 tokens plan	85.0	81.0	70.0	62.0	50.0	62.0	52.5	14.0	15.5
Number of Tokens in average									
DeepSeek-R1 (ARM only)	398	504	1420	1833	2076	2418	3068	3832	760
Qwen3-1.7B (ARM only)	282	397	1024	1669	2112	2550	3106	4036	984
Llama-3.2-3B (ARM only)	4	4	17	16	22	28	41	462	4
LLaDA-8B → Llama-3.2-3B (text-space)	4	4	10	14	16	12	20	503	4
LLaDA-8B → Llama-3.2-3B (latent-space)									
plan (64, 128, 256 tokens) + executor :[†]	2	2	4	5	5	5	6	14	2

[†] *Executor token averages; planner tokens (64, 128, 256) are to be added implicitly.*

reasoning domains. For each evaluation, we used the largest of 200 samples or the full benchmark size.

Projector Training Setup. We employ Latent-DARM equipped with a custom latent projector illustrated in Figure 2, consisting of three linear layers interleaved with two GELU activations. We train three projector variants, each using a different DDLM latent sequence length (64, 128, and 256). The projector maps a 4096-dimensional input to a 1024-dimensional bottleneck and subsequently projects to LLaMA's hidden dimension, followed by LlamaRMSNorm normalization. The training data comprises latent representations of shape $\{64, 128, 256\} \times 4096$ tokens, extracted from the last hidden layer of the DDLM LLaDA-8B-Instruct model after the final denoising step. This dataset contains 35,000 samples uniformly drawn from seven datasets: ARC_Easy, ARC_Challenge, and DART1–DART5 (5,000 samples each). The ARM model used for training is Llama-3.2-3B-Instruct using bfloat16 precision. Training utilizes the AdamW optimizer (PyTorch) with a learning rate of 5×10^{-4}, weight decay of 0.001, and 300 warmup steps, combined with cosine learning rate scheduling. We use a batch size of 4 per device, with gradient accumulation over 2 steps (effective batch size 8), training for 10 epochs. LoRA adapters (rank 8, alpha 32) are applied to the executor's attention and feed-forward network (FFN) projection layers for computational efficiency, while the LLaMA backbone and language modeling head remain frozen.

5 Results and Discussion

5.1 Results

Accuracy. The results of Latent-DARM compared to the text-space baseline are reported in Table 1 and illustrated in Figure 3. While performance on ARC-E (85.0 vs. 90.5) and ARC-C (81.0 vs. 82.5) remains comparable across the two settings, the latent space consistently yields substantially higher accuracy on the DART benchmarks: e.g., DART-1 (78.5 vs. 53.5), DART-2 (62.5 vs. 43.0), DART-3 (57.0 vs. 35.5), DART-4 (63.0 vs. 30.0), and DART-5 (54.0 vs. 27.0). On AIME, the latent approach performance reaches 12.5% with the projector trained on 64 tokens and 14% with the projector trained on 128 or 256 tokens, compared to 0.0% for the text space. Significantly,this improvement is obtained even though the projector between DDLM and ARM was trained without using any data from AIME or MMLU, yet it still generalizes to deliver markedly stronger results on these challenging evaluations. The latent approach underperforms on MMLU (44.0% vs. 52.5% text-space), likely because our projector was trained exclusively on reasoning benchmarks (DART, ARC) where planning provides structural guidance. MMLU emphasizes factual recall across diverse domains, where the planning bottleneck may lose fine-grained knowledge while preserving reasoning patterns—a trade-off that benefits multi-step tasks but hinders broad knowledge retrieval.

Remark. *At first glance, a result of 0.0% may seem surprising. However, when compared to publicly evaluated models such as Mistral-Small-3.1, a 24B parameter model (Vals.ai, 2026) achieving 3.54% on Aime, our result with a 3B model is reasonable.*

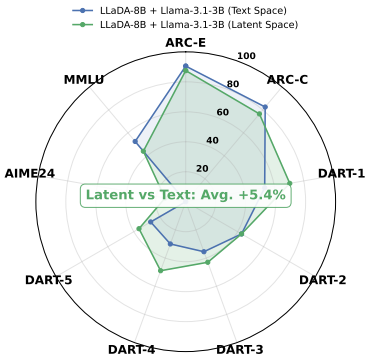

Figure 3: **Accuracy comparison of text-space vs. latent-space collaboration.**

Token Budget and Efficiency. We explicitly control the length of diffusion-generated plans (64, 128, 256 tokens). Results demonstrate that longer is not always better: Notably, the 64-token configuration offers the best overall trade-off between accuracy and efficiency on average. One plausible explanation for this superior performance in MMLU for instance, is its lower degree of redundancy. As shown in Appendix A, the repetition rate of diffusion outputs with 64 tokens in LLaDA-8B is comparable to that of Qwen2.5-7B and remains similarly low, whereas configurations with 128 and 256 tokens exhibit slightly higher repetition.

Most importantly, Latent-DARM is markedly more efficient than both baselines (DDLM→ARM in text space and ARM-only) as well as reasoning models. Remarkably, with only 64 planner tokens and an average of 5 executor tokens, it surpasses QWEN3 on DART-5 while using merely **2.2%** of the tokens, and outperforms it on AIME with just **1.9%**. Although it does not yet reach the raw accuracy of DEEPSEEK-R1, it achieves highly competitive performance at a fraction of the computational cost, as reflected in the average token usage reported in Table 1.

5.2 ARE WE IMPROVING THE PLANNER ?

To determine whether the gains from latent-space collaboration stem from improved plan communication—our central motivation—we perform a diagnostic analysis that disentangles planner and executor failures in DDLM→ARM collaboration. For each interface (text-space and latent-space), we consider two subsampled setups. **Setup X** consists of questions where DDLM→ARM fails but ARM→ARM succeeds, indicating a failure of the DDLM planner, since the executor is capable of solving the task when provided with a coherent plan. **Setup Y** consists of questions where

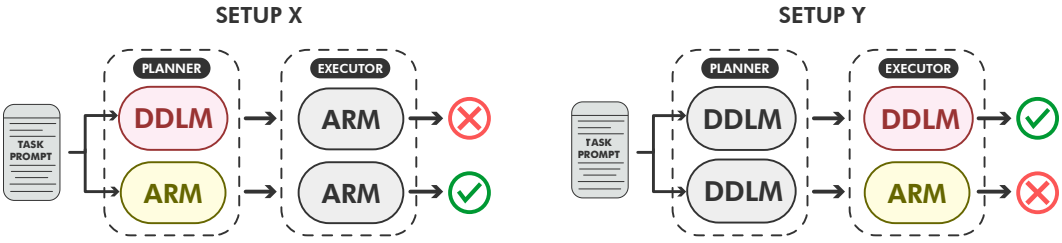

Figure 4: Diagnostic configurations for attributing errors to planner or executor. **Setup X** tests whether failures stem from the planner: if replacing the diffusion planner (DDLM) with an autoregressive planner (ARM) fixes the output, the error is attributed to the DDLM. **Setup Y** tests executor reliability: if a diffusion executor succeeds where an ARM executor fails, the limitation lies in the executor.

DDLM→DDLM succeeds but DDLM→ARM fails, indicating a limitation of the executor, as the planner output is sufficient and the executor is the only component that changes. We define a failure as a question for which the accuracy is zero. This diagnostic allows us to attribute DDLM→ARM failures to either the planner or the executor and to examine how this attribution shifts when moving from text-space to latent-space collaboration.

We quantify these effects using $\textbf{Percentage}_X$, the fraction of DDLM→ARM failures attributable to the planner (Setup X), and $\textbf{Percentage}_Y$, the fraction attributable to the executor (Setup Y), defined as

$$\textbf{Percentage}_i = \frac{\#\{\text{Setup } i \text{ samples}\}}{\#\{\text{Incorrect samples in DDLM} \rightarrow \text{ARM}\}} \quad \text{for } i \in \{X, Y\}.$$

Figure 4 illustrates these two setups. In both setups, ARM→ARM and DDLM→DDLM are always evaluated in text space; only DDLM→ARM is varied between text and latent communication, isolating the effect of the interface. As shown in Figure 5, under text-space collaboration, most failures fall into Setup X, indicating that performance is primarily limited by planning degradation induced by textual decoding. In contrast, under Latent-DARM, failures shift predominantly to Setup Y, demonstrating that latent-space communication substantially improves planning fidelity, with the executor emerging as the dominant bottleneck. Exact failure distributions are reported in Appendix C.

Remark. This observation does *not* imply that the executor makes more mistakes. The reported quantities are relative percentages over failures: the shift indicates that, under Latent-DARM, a smaller fraction of errors is attributable to the planner. Consequently, the observed accuracy gains can be attributed to improved planner communication.

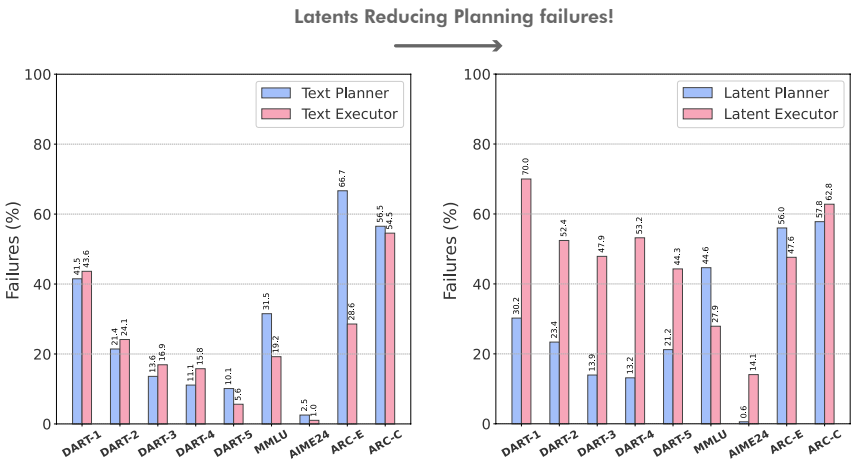

Figure 5: **Planner vs. executor failures in text- vs. latent-space collaboration.** Results for LLaDA-8B-Instruct and Llama-3.2-3B-Instruct. Latent-space collaboration substantially reduces planning errors compared to text-space.

6 Conclusion

We introduced **Latent-DARM**, a latent-space collaboration framework that enables effective cooperation between discrete diffusion language models (DDLMs) and autoregressive language models (ARMs) in multi-agent reasoning systems. By replacing text-based interfaces with a learned latent projection, our approach allows diffusion planners to transmit globally structured planning information to autoregressive executors without being constrained by fluency limitations.

Empirically, latent-space communication outperforms text-based collaboration on average, particularly on planning-intensive benchmarks such as DART and AIME. Diagnostic analyses show that these gains primarily arise from a reduction in planning failures, indicating that latent exchange

preserves high-level reasoning structure that is otherwise degraded by textual decoding. Moreover, Latent-DARM achieves competitive performance with state-of-the-art reasoning models while using orders of magnitude fewer tokens, demonstrating that strong reasoning does not require long textual chains of thought.

Overall, this work suggests that text does not need to be the sole medium of inter-agent communication. Latent-space interfaces provide a high-bandwidth, task-aligned alternative that enables efficient collaboration between heterogeneous models, opening new directions for scalable and budget-aware reasoning systems.

Future Work. Our results suggest that latent-space communication represents an underexplored but promising paradigm for multi-agent systems. The dramatic efficiency gains and improvements on planning-intensive tasks indicate this approach deserves further systematic investigation. Key directions include: developing adaptive architectures that route between latent and text modes based on task characteristics, scaling to diverse domains to understand generalization limits, extending to bidirectional and multi-hop agent collaboration, and establishing theoretical foundations for when and why latent communication succeeds. More broadly, this work challenges the assumption that natural language is the optimal inter-agent medium, opening questions about what other structured representations might enable even tighter model integration.

REFERENCES

Deepak Bhaskar Acharya, Karthigeyan Kuppan, and B Divya. Agentic ai: Autonomous intelligence for complex goals–a comprehensive survey. *IEEe Access*, 2025.

Naoto Aoki, Naoki Mori, and Makoto OKada. Analysis of llm-based narrative generation using the agent-based simulation. In *2023 15th International Congress on Advanced Applied Informatics Winter (IIAI-AAI-Winter)*, pp. 284–289, 2023. doi: 10.1109/IIAI-AAI-Winter61682.2023.00059.

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

Gregor Bachmann and Vaishnavh Nagarajan. The pitfalls of next-token prediction. *arXiv preprint arXiv:2403.06963*, 2024.

Guangyao Chen, Siwei Dong, Yu Shu, Ge Zhang, Jaward Sesay, Börje F Karlsson, Jie Fu, and Yemin Shi. Autoagents: A framework for automatic agent generation. *arXiv preprint arXiv:2309.17288*, 2023.

Sitan Chen, Kevin Cong, and Jerry Li. Optimal inference schedules for masked diffusion models. *arXiv preprint arXiv:2511.04647*, 2025.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *ArXiv*, abs/1803.05457, 2018.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv e-prints*, pp. arXiv–2407, 2024.

Guhao Feng, Yihan Geng, Jian Guan, Wei Wu, Liwei Wang, and Di He. Theoretical benefit and limitation of diffusion language model. *arXiv preprint arXiv:2502.09622*, 2025.

Itai Gat, Tal Remez, Neta Shaul, Felix Kreuk, Ricky TQ Chen, Gabriel Synnaeve, Yossi Adi, and Yaron Lipman. Discrete flow matching. *Advances in Neural Information Processing Systems*, 37: 133345–133385, 2024.

T Guo, X Chen, Y Wang, R Chang, S Pei, NV Chawla, O Wiest, and X Zhang. Large language model based multi-agents: A survey of progress and challenges. arxiv 2024. *arXiv preprint arXiv:2402.01680*, 10, 2024.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*, 2024.

Gaole He, Gianluca Demartini, and Ujwal Gadiraju. Plan-then-execute: An empirical study of user trust and team performance when using llm agents as a daily assistant. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, pp. 1–22, 2025.

Dan Hendrycks, Collin Burns, Steven Basart, Andrew Critch, Jerry Li, Dawn Song, and Jacob Steinhardt. Aligning ai with shared human values. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021a.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021b.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, et al. Metagpt: Meta programming for a multi-agent collaborative framework. In *The twelfth international conference on learning representations*, 2023.

Maxwell Jia. Aime problem set 2024, 2024. URL https://huggingface.co/datasets/Maxwell-Jia/AIME_2024.

Kia Karbasi, Kevin Hong, Mohammad Amin Samadi, and Gregory Pottie. Multi-agent collaborative framework for math problem generation. *arXiv preprint arXiv:2511.03958*, 2025.

Jaeyeon Kim, Seunggeun Kim, Taekyun Lee, David Z Pan, Hyeji Kim, Sham Kakade, and Sitan Chen. Fine-tuning masked diffusion for provable self-correction. *arXiv preprint arXiv:2510.01384*, 2025.

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

Pietro Panzarasa, Nicholas R Jennings, and Timothy J Norman. Formalizing collaborative decision-making and practical reasoning in multi-agent systems. *Journal of logic and computation*, 12(1): 55–117, 2002.

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

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

Yuxuan Tong, Xiwen Zhang, Rui Wang, Ruidong Wu, and Junxian He. Dart-math: Difficulty-aware rejection tuning for mathematical problem-solving. *Advances in Neural Information Processing Systems*, 37:7821–7846, 2024.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Vals.ai. Mistral small 3.1 (model id mistral-small-2503). https://www.vals.ai/models/mistralai_mistral-small-2503, 2026. Accessed: 2026-02-09.

Hemish Veeraboina. Aime problem set 1983-2024, 2024. URL https://www.kaggle.com/datasets/hemishveeraboina/aime-problem-set-1983-2024.

Dimitri von Rütte, Janis Fluri, Yuhui Ding, Antonio Orvieto, Bernhard Schölkopf, and Thomas Hofmann. Generalized interpolating discrete diffusion. *arXiv preprint arXiv:2503.04482*, 2025.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. *arXiv preprint arXiv:2305.04091*, 2023.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, et al. Autogen: Enabling next-gen llm applications via multi-agent conversations. In *First Conference on Language Modeling*, 2024.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2. 5 technical report. *arXiv e-prints*, pp. arXiv–2412, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Jiacheng Ye, Jiahui Gao, Shansan Gong, Lin Zheng, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Beyond autoregression: Discrete diffusion for complex reasoning and planning. *arXiv preprint arXiv:2410.14157*, 2024a.

Jiacheng Ye, Shansan Gong, Liheng Chen, Lin Zheng, Jiahui Gao, Han Shi, Chuan Wu, Xin Jiang, Zhenguo Li, Wei Bi, et al. Diffusion of thought: Chain-of-thought reasoning in diffusion language models. *Advances in Neural Information Processing Systems*, 37:105345–105374, 2024b.

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

Shaowei Zhang and Deyi Xiong. Debate4math: Multi-agent debate for fine-grained reasoning in math. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 16810–16824, 2025.

Hanlin Zhu, Shibo Hao, Zhiting Hu, Jiantao Jiao, Stuart Russell, and Yuandong Tian. Reasoning by superposition: A theoretical perspective on chain of continuous thought. *arXiv preprint arXiv:2505.12514*, 2025a.

Rui-Jie Zhu, Tianhao Peng, Tianhao Cheng, Xingwei Qu, Jinfa Huang, Dawei Zhu, Hao Wang, Kaiwen Xue, Xuanliang Zhang, Yong Shan, et al. A survey on latent reasoning. *arXiv preprint arXiv:2507.06203*, 2025b.

APPENDIX

A MORE INSIGHTS ON PLANNER REPETITION OF TOKENS

We will perform a qualitative assessment to identify prompt repetition errors in the planner text in the setup DDLM $\rightarrow$ ARM in the text space, using 256, 128 and 64 tokens for the LlaDa-8B-Instruct model, as well as Qwen2.5-7B-Instruct and Dream-v0-7B-Instruct for comparison. This analysis examines how increasing the number of planning tokens affects repetition, relative to an autoregressive model baseline.

We use the following metrics proposed in (Aoki et al., 2023):

- Distinct-3 (D-3)
- Repetition-4 (R-4)
- Lexical Repetition (LR-n)

Distinct-3 (D-3) calculates the percentage of unique 3-grams over all 3-grams. The value of Distinct-3 takes values between 0 and 1, with the closer to 1 indicating that the text is more diverse at the 3-gram level. Let D_3 be the number of unique 3-grams in the text and T_3 be the total number of 3-grams in the text. Distinct-3 is then computed by the following formula:

$$\text{Distinct-3} = \frac{D_3}{T_3} \times 100$$

Repetition-4(R-4) Let T be the total number of sentences in the text, R_t be the number of 4-grams repeated in a sentence t, and $I(x)$ be an indicator function (1 if x is true, 0 if x is false). Then Repetition-4 is calculated as follows:

$$\text{Repetition-4} = \frac{1}{T} \sum_{t=1}^{T} I(R_t > 1) \times 100$$

Lexical Repetition (LR-n) computes the average percentage of 4-grams that occur at least n times in the generated text. Let G be the total number of possible 4-grams in all texts and L_g be the number of repetitions of G, then Lexical Repetition (LR-n) is calculated by the following formula:

$$\text{Lexical Repetition} = \frac{1}{G} \sum_{g=1}^{G} I(L_g \geq n) \times 100$$

Table 2: Repetition Evaluation

Simulation	**D-3**	**R-4**	**LR-n**
	↑	↓	↓
`Qwen2.5-7B-Instruct (Baseline)`	98.47	3.05	0.64
`LLaDA-8B-Instruct (256 tokens)`	83.05	10.52	6.74
`LLaDA-8B-Instruct (128 tokens)`	93.15	5.33	3.43
`LLaDA-8B-Instruct (64 tokens)`	96.85	3.37	1.33
`Dream-v0-7B-Instruct (256 tokens)`	62.02	3.26	2.15

Results In table 2, the ↑ indicates that a larger value corresponds to better performance, while the ↓ indicates that a smaller value corresponds to better performance.

We observe a tendency for greater repetition in the plans generated by LLADA as the number of tokens increases.The number of repetitions in diffusion with 64 tokens in LLADA is comparable to that of Qwen2.5-7B and remains similarly low.

B Prompts for LLM Experiments

B.1 Planner Prompt

```
You are a careful problem-solving planner.

Task: Produce ONLY a short list of HINTS that help
solve the question.
Do NOT state or imply the final answer.
Do NOT mention any option letter (A, B, C, or D).
Do NOT quote any option text verbatim.
If you find yourself about to reveal a specific
option or an answer, replace it with \[HIDDEN]".

Format:
- Key facts to recall (2{4 bullets)
- Reasoning steps or elimination rules (2{5 bullets)
```

```
- Useful equations or definitions (if relevant)
- Edge cases or common traps (optional)

Be concise (<=120 words). No \Answer:" line.
No letters A{D.

Question (stem only):
{question}
```

B.2 Executor Prompt

```
You are an expert in solving multiple-choice questions.
Given the following plan or reasoning, please solve the question.
If the plan contains any explicit answer or option letter,
ignore it and solve from the hints + question only.

Plan:
{plan}
{question}
```

C Performance Comparison of Planner vs. Executor Issues

Table 3: Performance comparison of planner vs. executor issues for LLaDA-8B-Instruct and Llama-3.2-3B-Instruct under **Text-Space** vs. **Latent-Space** collaboration.

Benchmark	Planning Failures (%)	Execution Failures (%)	Error Gap (%)
LLaDA-8B + Llama-3.2-3B (Text-Space Pipeline)			
DART-1	41.50	**43.64**	2.14
DART-2	21.43	**24.14**	2.71
DART-3	13.60	**16.92**	3.32
DART-4	11.11	**15.79**	4.68
DART-5	**10.12**	5.63	4.49
MMLU	**31.52**	19.23	12.29
AIME24	**2.54**	1.03	1.51
ARC-E	**66.67**	28.57	38.10
ARC-C	**56.52**	54.55	1.97
LLaDA-8B + Llama-3.2-3B (Latent-Space Pipeline)			
DART-1	30.23	**70.00**	39.77
DART-2	23.37	**52.42**	29.05
DART-3	13.95	**47.88**	33.93
DART-4	13.15	**53.19**	40.04
DART-5	21.21	**44.29**	23.08
MMLU	**44.64**	27.90	16.74
AIME24	0.58	**14.07**	13.49
ARC-E	**56.00**	47.61	8.39
ARC-C	57.80	**62.79**	4.99

D DETAILED RESULTS FROM EXPERIMENTS

Table 4: Consolidated evaluation results on text-space collaboration across all model combinations and reasoning benchmarks. † Evaluated with `enable_thinking=True`, ‡ with `enable_thinking=False`.

Model / Combination	Setup	ARC-E	ARC-C	DART-1	DART-2	DART-3	DART-4	DART-5	AIME	MMLU
ARMs only										
Qwen2.5-3B	ARM	91.0	86.5	58.0	34.0	29.5	17.5	11.5	1.0	61.0
	ARM → ARM	89.5	81.5	64.0	49.0	30.5	24.0	16.0	0.0	59.0
Dream-v0-7B + Qwen2.5-3B	DDLM → ARM	92.0	86.0	65.5	40.5	31.0	25.0	12.5	0.5	60.5
LLaDA-8B + Qwen2.5-3B	DDLM → ARM	88.5	85.0	68.0	44.5	29.0	23.0	14.0	0.5	55.5
Qwen2.5-7B	ARM	94.5	90.5	68.0	48.5	35.0	35.0	19.5	2.5	67.0
	ARM → ARM	96.5	89.0	72.5	56.5	35.0	33.5	20.0	3.0	61.0
Dream-v0-7B + Qwen2.5-7B	DDLM → ARM	95.5	88.5	60.5	42.0	34.0	28.5	16.0	1.0	64.5
LLaDA-8B + Qwen2.5-7B	DDLM → ARM	92.5	88.5	73.5	58.0	37.5	37.0	16.0	1.5	54.0
Llama-3.2-3B	ARM	87.5	79.5	44.5	35.5	28.0	34.5	36.5	3.5	54.0
	ARM → ARM	91.0	80.0	47.5	37.0	31.0	30.5	30.0	0.0	56.5
Dream-v0-7B + Llama-3.2-3B	DDLM → ARM	91.0	79.0	54.0	39.5	32.0	31.5	27.5	0.0	59.5
LLaDA-8B + Llama-3.2-3B	DDLM → ARM	90.5	82.5	53.5	43.0	35.5	30.0	27.0	0.0	52.5
LLaDA-8B → Llama-3.2-3B (latent-space)		87.5	76.5	70.5	43.0	43.0	49.0	36.0	14.0	44.0
Llama-3.1-8B	ARM	87.5	79.5	68.5	50.5	36.0	36.0	20.5	2.5	63.5
	ARM → ARM	91.0	82.0	74.0	58.5	35.5	35.0	20.5	3.0	64.0
Dream-v0-7B + Llama-3.1-8B	DDLM → ARM	94.0	86.5	55.5	33.0	37.0	28.0	26.0	0.5	64.0
LLaDA-8B + Llama-3.1-8B	DDLM → ARM	91.5	82.5	74.0	60.0	38.5	37.0	16.5	1.5	56.5
DeepSeek-R1	ARM	94.0	88.0	89.0	81.5	85.5	75.0	61.5	28.5	60.0
LLaDA-8B + DeepSeek-R1	DDLM → ARM	92.5	88.5	73.5	58.0	37.5	37.0	16.0	1.5	54.0
Qwen3.1-7B	ARM†	92.0	86.0	89.5	86.0	83.0	73.0	49.0	8.5	68.5
LLaDA-8B + Qwen3.1-7B	DDLM → ARM‡	91.0	82.5	87.0	70.0	58.5	46.0	27.0	1.0	52.5