ENHANCING DIFFUSION LANGUAGE MODELS WITH AUTOREGRESSIVE POST-TRAINING WEIGHTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion language models (dLLMs) have emerged as a promising alternative to autoregressive (AR) language models, offering flexible token-update orders and parallel decoding. Recent dLLMs are often initialized from pretrained AR models before diffusion conversion to inherit their learned representations. After the conversion, however, they typically discard the extensive post-training ecosystem of their AR ancestors. In this work, we show that these existing AR post-training weight updates can instead be effectively recycled to enhance diffusion models, despite the changes by AR-to-diffusion conversion. Directly adding an AR post-training weight update to a diffusion base model remains effective, bringing its performance close to that achieved by direct diffusion post-training. Notably, AR and diffusion post-training updates are nearly orthogonal in weight space, yet induce substantially more aligned representation changes in the diffusion model. Their distinct updates are also complementary: composing their weights can retain gains from both and further improve the diffusion-post-trained model. Based on these findings, we propose **A2D**, a simple training-free framework for enhancing diffusion models with existing AR post-training resources. A2D can transfer capabilities from AR post-trained models to diffusion base models, and further improve already post-trained diffusion models by composing AR and diffusion post-training updates. Across Dream, DreamReasoner, DiffuCoder, Nemotron-Labs-Diffusion, and DiffusionGemma, A2D reliably improves instruction following, mathematical reasoning, and coding with both supervised fine-tuning and reinforcement learning updates, without additional training, or inference-time computation.

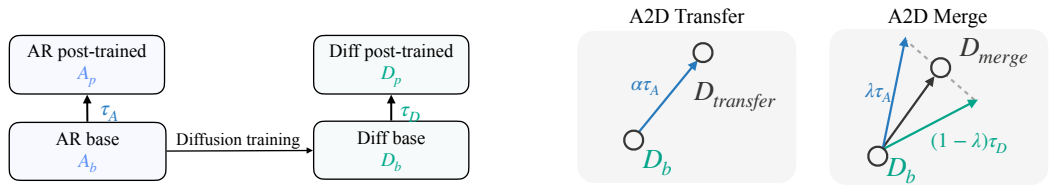
1 INTRODUCTION

Diffusion large language models (dLLMs) have recently emerged as a competitive alternative to autoregressive (AR) models, achieving strong performance across instruction following, code generation, and mathematical reasoning (Zhu et al., 2025; Ye et al., 2025; Gong et al., 2026; Fu et al., 2026; Team et al., 2026). By casting sequence generation as iterative denoising rather than fixed left-to-right factorization, dLLMs support bidirectional context and flexible token-update orders, as well as parallel decoding, making them naturally amenable to scalable generation.

Recent dLLMs commonly initialize from pretrained AR models before diffusion conversion (Ye et al., 2025; Fu et al., 2026; Wu et al., 2026b; Team et al., 2026), reusing language representations learned during large-scale AR pretraining and substantially improving training efficiency over training dLLMs from scratch. This pipeline, however, creates a notable asymmetry: dLLMs routinely inherit AR pretraining, yet largely discard the extensive AR post-training investment built upon it. This is especially notable as the AR ecosystem (Yang et al., 2025; Team, 2026; Liu et al., 2026) often provides a rich collection of post-trained checkpoints across diverse capabilities, which are often stronger than their diffusion counterparts. Diffusion models instead typically develop these capabilities through separate diffusion-specific post-training, which introduces additional optimization challenges arising from learning over partially corrupted states (Parashar et al., 2026; Lin et al., 2026).

This raises a natural question: *can the knowledge acquired with AR post-training be incorporated into diffusion models?* Such a transfer could reuse readily available, often stronger AR post-trained checkpoints while retaining the parallel-decoding benefits of diffusion. Since AR post-training

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



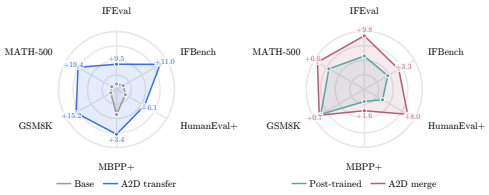
(a) Illustration of notations. (b) Illustration of A2D.

Figure 1: **Illustration of notations and method**. A2D-Transfer directly adds the AR post-training vector to diffusion base to create a post-trained diffusion model. A2D-Merge interpolates this model and the diffusion model's own post-trained checkpoint to further improve downstream capabilities.

and diffusion conversion originate from the same pretrained AR model, their shared initialization and architecture naturally motivate a weight-space approach: *model merging*. Within the AR ecosystem, model merging has proven effective at integrating capabilities from independently fine-tuned checkpoints without requiring their training data (Wortsman et al., 2022; Yang et al., 2026). In particular, task arithmetic represents post-training weight updates as *task vectors* that can be transferred or composed across compatible models (Ilharco et al., 2022). Extending model merging across the AR–diffusion boundary, however, is nontrivial: diffusion conversion moves the model away from its AR initialization while changing generation from causal next-token prediction to bidirectional denoising.

Nevertheless, we show that AR post-training exhibits *cross-paradigm transferability*: knowledge learned in the AR regime can remain effective to a diffusion model without further adaptation. In a controlled setting, directly transferring the AR post-training task vector brings the diffusion base close to the performance of direct diffusion post-training. This cross-paradigm transfer of AR task vector provides a practical way to improve diffusion models when no corresponding diffusion post-trained checkpoint is available. Moreover, we show that AR and diffusion post-training yield nearly orthogonal task vectors in parameter space, yet induce much more aligned representation changes, suggesting that related capabilities arise through distinct parameter directions. Motivated by prior findings that less-correlated fine-tuning directions can combine favorably through interpolation (Wortsman et al., 2022), we further interpolate the AR-transferred and diffusion-post-trained solutions and lead to performance surpassing both endpoints.

We introduce **A2D** (*AR-to-Diffusion Transfer and Composition*), a simple framework that exploits the cross-paradigm transferability of AR post-training updates. `A2D` operates in two regimes, as shown in Figure 1b: (a) in the absence of a diffusion post-training vector, it transfers an AR task vector directly to the diffusion base model; (b) when both AR and diffusion vectors are available, it interpolates the corresponding diffusion-space solutions to further improve the post-trained diffusion model. Both operations are training-free given the available checkpoints and incur no additional trainable parameters. Across multiple diffusion model families and diverse post-training settings, including instruction tuning, reasoning, mathematical reasoning, and coding, both modes of `A2D` reliably improves downstream performance, as shown in Figure 2. We summarize our contributions as follows:



(a) Direct transfer (b) Update merge

Figure 2: **`A2D` on Dream.** `A2D` improves diffusion base and post-trained models **without training** through (a) direct AR update transfer and (b) AR–diffusion update merging.

- We study *cross-paradigm transferability*: AR post-training vectors remain effective after diffusion conversion, inducing aligned representation changes despite near-orthogonal parameter directions, which further motivates their composition with diffusion post-training.
- Building on this observation, we introduce A2D, a simple weight-space framework for reusing AR post-training in diffusion models. `A2D` supports both direct transfer to diffusion base models and training-free composition with existing diffusion post-training updates.
- Across Dream, DreamCoder, DiffuCoder, Nemotron-Labs-Diffusion, DreamReasoner, and DiffusionGemma, we validate `A2D` and diverse post-training settings, consistently improving downstream performance without additional training or inference-time computation.

2 CROSS-PARADIGM POST-TRAINING TRANSFER AND COMPOSITION

2.1 BACKGROUND AND NOTATIONS

Autoregressive and diffusion language models. AR and diffusion models differ in both their training objectives and generation processes, yet modern dLLMs are often adapted from pretrained AR checkpoints to inherit the learned language representations (Ye et al., 2025; Fu et al., 2026; Wu et al., 2026b; Team et al., 2026). We denote the weights of this AR source model by $\boldsymbol{A}_b$, which is trained with an autoregressive objective and generates tokens from left to right. Starting from $\boldsymbol{A}_b$, further diffusion training adapts the model for diffusion generation to obtain $\boldsymbol{D}_b$. Depending on the model family, this adaptation uses either a diffusion objective alone (Ye et al., 2025) or a combination of AR and diffusion objectives (Fu et al., 2026), and may involve hundreds of billions of tokens of training. Consequently, despite sharing an initialization, $\boldsymbol{D}_b$ can differ substantially from $\boldsymbol{A}_b$ in both parameters and generation behavior. At inference time, it generates through iterative denoising with flexible token-update orders.

Starting from their respective base checkpoints, the AR and diffusion models can be post-trained independently for the same downstream capability, yielding $\boldsymbol{A}_p$ from $\boldsymbol{A}_b$ and $\boldsymbol{D}_p$ from $\boldsymbol{D}_b$, as illustrated in Figure 1a. Although $\boldsymbol{A}_p$ and $\boldsymbol{D}_p$ target the same capability, their post-training updates are learned under different objectives and from different base checkpoints. This raises our central question: can an update learned on the AR branch remain effective when applied to the diffusion branch?

Model merging and task arithmetic. Model merging combines knowledge from multiple trained models directly in weight space without additional training (Yang et al., 2026; Wortsman et al., 2022). Task arithmetic represents the effect of post-training as a task vector, defined as the parameter difference between a post-trained model and its base (Ilharco et al., 2022). These vectors can be scaled and composed in weight space to transfer or combine their learned capabilities. Accordingly, we define the AR and diffusion post-training vectors as $\boldsymbol{\tau}_A = \boldsymbol{A}_p - \boldsymbol{A}_b$ and $\boldsymbol{\tau}_D = \boldsymbol{D}_p - \boldsymbol{D}_b$, respectively. Notably, task arithmetic typically composes task vectors defined relative to a shared base model or initialization, whereas $\boldsymbol{\tau}_A$ and $\boldsymbol{\tau}_D$ originate from different base checkpoints.

2.2 CROSS-PARADIGM TRANSFER OF AR POST-TRAINING TO DIFFUSION

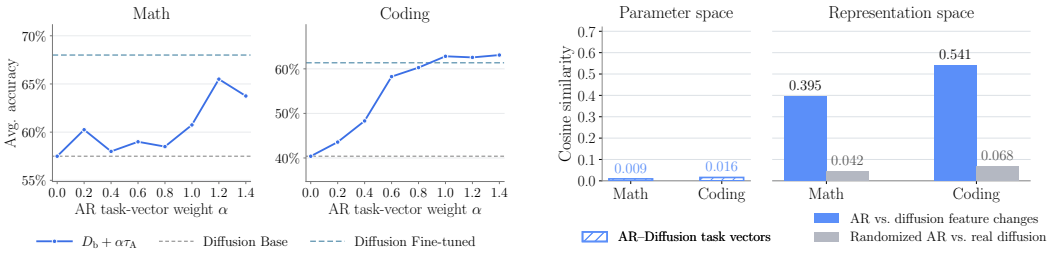
We begin our analysis by testing whether post-training knowledge learned in the AR regime survives AR-to-diffusion conversion. In this case, $\boldsymbol{\tau}_A$ is learned around $\boldsymbol{A}_b$ but applied to its diffusion descendant $\boldsymbol{D}_b$, whose parameters and generation mechanism have been substantially reshaped by diffusion continued training. To isolate the effect of this paradigm shift, we post-train the AR and diffusion base models on identical data for the same downstream capability in a controlled setting.

Experimental setting. We consider mathematical reasoning and coding. For math, we use Qwen2.5-7B and its diffusion counterpart Dream-7B-Base, obtained from Qwen2.5-7B through AR-to-diffusion conversion, and fine-tune both on the same 100K NuminaMath examples (LI et al., 2024). For coding, we use Qwen2.5-Coder-7B and Dream-Coder-v0-Base-7B, its corresponding diffusion counterpart, and fine-tune both on the same 100K EpiCoder examples (Wang et al., 2025b). This construction yields matched AR and diffusion task vectors targeting the same capability, allowing us to isolate whether AR post-training transfers across the two generative paradigms. Full training and evaluation details are provided in Appendix B.3.

Cross-paradigm transfer. We test this directly by applying the scaled AR task vector $\boldsymbol{\tau}_A$ to the diffusion base model: $\boldsymbol{D}_T = \boldsymbol{D}_b + \alpha\boldsymbol{\tau}_A,\ \alpha \geq 0$, where α controls the transfer strength. This modifies only the model parameters while preserving the diffusion inference procedure.

We evaluate mathematical reasoning on GSM8K (Cobbe et al., 2021) and MATH-500 (Hendrycks et al., 2021) [1], and coding on HumanEval+ (Chen et al., 2021) and MBPP+ (Austin et al., 2021b), respectively for the models fine-tuned on math and coding datasets. As shown in Figure 3a, transferring $\boldsymbol{\tau}_A$ substantially improves $\boldsymbol{D}_b$ in both domains and recovers a large fraction of the gains obtained by direct diffusion post-training, or even surpasses them. This shows that an update learned under causal AR computation can remain effective after the model has been converted to iterative diffusion denoising.

[1] We subsample first 200 samples for GSM8K and MATH-500 for the analysis in section 2



(a) AR post-training updates transfer effectively to diffusion.

(b) Distinct parameter updates induce aligned representation changes.

Figure 3: **Cross-paradigm transfer of AR post-training. (a)** AR task vectors remain effective on the diffusion base. **(b)** Despite weak parameter alignment, AR and diffusion post-training induce more aligned representation changes than randomized controls.

Distinct parameters, aligned effects in representation space. The successful transfer raises a question: do AR and diffusion post-training simply learn similar parameter updates? Despite being trained toward the same capability, $\boldsymbol{\tau}_A$ and $\boldsymbol{\tau}_D$ are nearly orthogonal in parameter space with a cosine similarity of only 0.016 for Coding and 0.009 for Math. Parameter-space similarity, however, does not reflect similarity in the induced model behavior (Xu et al., 2024). We therefore compare the representation changes produced by the two task vectors within the diffusion model. We generate trajectories with $\boldsymbol{D}_b$ and collect intermediate pre-commit masked states m. For each state, we evaluate $\boldsymbol{D}_b$, $\boldsymbol{D}_b + \boldsymbol{\tau}_A$, and $\boldsymbol{D}_b + \boldsymbol{\tau}_D$ on the same input, and isolate the effect of each task vector by subtracting the representation of the diffusion base:

$$\Delta \boldsymbol{h}_A(m) = \boldsymbol{h}(\boldsymbol{D}_b + \boldsymbol{\tau}_A; m) - \boldsymbol{h}(\boldsymbol{D}_b; m), \qquad \Delta \boldsymbol{h}_D(m) = \boldsymbol{h}(\boldsymbol{D}_b + \boldsymbol{\tau}_D; m) - \boldsymbol{h}(\boldsymbol{D}_b; m),$$

where $\boldsymbol{h}$ denotes the final-layer representation at the prediction position. We then measure the cosine similarity between $\Delta \boldsymbol{h}_A(m)$ and $\Delta \boldsymbol{h}_D(m)$. As shown in Figure 3b, the induced representation changes are substantially more aligned than the task vectors themselves and well above randomized-vector controls [2]. Together, these results show that AR and diffusion post-training induce related representation changes, despite following substantially different directions in parameter space.

2.3 Composing Autoregressive and Diffusion Post-Training

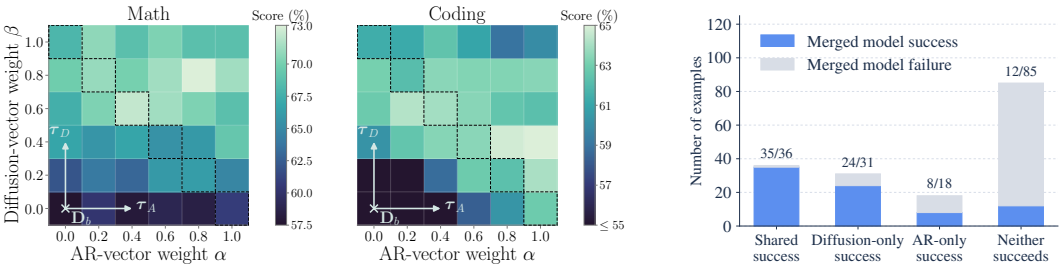
The preceding results reveal an interesting structure: AR and diffusion post-training induce related representation changes through largely distinct parameter directions. Prior work has suggested that less-correlated fine-tuning directions can admit favorable linear interpolations (Wortsman et al., 2022), motivating the composition of these two post-training models. We therefore examine whether their composition can yield a stronger model than either alone.

To analyze this composition in weight space, we consider the two-dimensional task-vector plane $\boldsymbol{D}(\alpha, \beta) = \boldsymbol{D}_b + \alpha\boldsymbol{\tau}_A + \beta\boldsymbol{\tau}_D$, $\alpha, \beta \geq 0$, where α and β controls the contribution of the AR and diffusion task vector, respectively. The AR-transferred and diffusion-post-trained models correspond to $\boldsymbol{D}_T = \boldsymbol{D}(1, 0)$ and $\boldsymbol{D}_p = \boldsymbol{D}(0, 1)$, respectively, while $\alpha + \beta = 1$ traces the linear interpolation between them.

As shown in Figure 4a, both mathematical reasoning and coding exhibit high-performing regions along this interpolation path, and interior points along their interpolation path outperform either endpoint. This indicates that interpolating the AR and diffusion post-training directions captures complementary gains beyond either endpoint.

Complementarity between AR and diffusion post-training. Why can an intermediate model outperform both endpoints? To better understand these gains, we characterize the behavioral overlap and differences between the two endpoints and the interpolated model.

[2]The randomized vector is built to share the same norm as the AR task vector and similarity to the diffusion vector.



(a) High-performing solutions lie between the AR and diffusion endpoints.

(b) Interpolation preserves shared gains and endpoint-specific successes.

Figure 4: **Why AR and diffusion post-training can be composed.** **(a)** Performance over $\boldsymbol{D}_b + \alpha\boldsymbol{\tau}_A + \beta\boldsymbol{\tau}_D$. The interpolation path between the AR-transferred and diffusion-post-trained endpoints passes through high-performing regions and contains solutions that outperform both endpoints. **(b)** Among examples failed by $\boldsymbol{D}_b$, the two endpoints exhibit both shared and endpoint-specific successes. The best interpolated model preserves most shared successes while retaining gains unique to either endpoint, indicating limited functional interference under composition.

Among examples failed by the base model $\boldsymbol{D}_b$ in mathematical reasoning, we partition problems into four mutually exclusive subsets: those solved only by the AR-transfer model $\boldsymbol{D}_T$, only by the post-trained model $\boldsymbol{D}_p$, by both, or by neither. We then evaluate the best interpolated model on each subset. As shown in Figure 4b, the two endpoints solve overlapping but also distinct subsets of problems. Importantly, the interpolated model preserves nearly all their shared successes while retaining large portions of the successes unique to either endpoint. Overall, we show that AR and diffusion post-training provide partially complementary behavioral gains that can be jointly retained through interpolation, explaining how an intermediate model can outperform both endpoints. This observation is also consistent with prior findings that task-vector composition can preserve distinct effects of different parameter-space directions (Ortiz-Jimenez et al., 2023).

2.4 A2D: Cross-Paradigm Transfer and Composition

Motivated by the preceding observations, we propose **A2D** (*AR-to-Diffusion Transfer and Composition*), a simple weight-space framework for reusing AR post-training updates in diffusion models. A2D addresses two practical settings.

A2D Transfer. For many downstream capabilities, strong AR post-trained checkpoints are readily available, while corresponding diffusion checkpoints may not yet exist. In this setting, A2D transfers the AR task vector directly to the diffusion base to equip it with the target capability:

$$\boldsymbol{D}_{\text{Transfer}}(\alpha) = \boldsymbol{D}_b + \alpha\boldsymbol{\tau}_A, \qquad \alpha \geq 0. \tag{1}$$

This directly reuses existing AR post-training knowledge, avoiding a separate round of diffusion-specific post-training.

A2D Merge. For scenarios where both AR and diffusion post-trained models are available, A2D interpolates their post-training updates to exploit their complementary gains. We restrict the composition to this one-dimensional path, which already traverses a high-performing region in Figure 4a:

$$\boldsymbol{D}_{\text{Merge}}(\lambda) = \boldsymbol{D}_b + (1-\lambda)\boldsymbol{\tau}_{\boldsymbol{D}} + \lambda\boldsymbol{\tau}_A = \boldsymbol{D}_p + \lambda(\boldsymbol{\tau}_A - \boldsymbol{\tau}_D), \qquad \lambda \in [0,1]. \tag{2}$$

Both A2D variants preserve the diffusion architecture and require no additional training or inference-time hyperparameter tuning. We select α and λ on a single anchor benchmark for each model family and keep them fixed across all remaining evaluations; see Appendix B.

3 A2D Across Models and Post-Training Settings

Beyond the controlled setting, we now evaluate `A2D` on independently released checkpoints, where the AR and diffusion branches may differ substantially in their post-training data, objectives, and

Table 1: **Evaluation of A2D across tasks and model families.** Headers denote AR donor→diffusion recipient. A2D (transfer) applies the AR donor's task vector to the diffusion recipient, while A2D (merge) composes it with diffusion post-training. Anchor tasks for scaling coefficient search are highlighted. All results are in percentages.

(a) Instruction following vectors.

	Qwen2.5-7B → Dream-7B			Qwen2.5-Coder-7B → Dream-Coder			Qwen2.5-Coder-7B → DiffuCoder		
	IFEval	MATH-500	MBPP+	HumanEval+	MBPP+	BCB-Full	HumanEval+	MBPP+	BCB-Full
AR donor	75.3	56.0	69.0	86.6	72.5	41.8	86.6	72.5	49.6
Base	49.7	20.8	59.3	57.9	62.4	15.4	**61.6**	52.9	39.7
A2D (transfer)*	**59.2**	**40.2**	**62.7**	**67.1**	**62.7**	**34.2**	60.4	**54.0**	**40.9**
Instruct	63.0	38.4	57.1	77.4	68.5	36.0	65.9	65.9	38.1
A2D (merge)	**72.8**	**45.0**	**58.7**	**79.3**	**69.3**	**38.4**	**75.0**	**66.4**	**40.8**

(b) Reasoning, SFT and RL post-training vectors. DR abbreviates DreamReasoner.

	Qwen3-8B → DR-8B, thinking on			Gemma-4 → DiffusionGemma			AceCoder-Ins-Rule → DiffuCoder, RL		
	MATH-500	AIME24	AIME25	VQA-RAD	SLAKE	VQA-Med	HumanEval+	MBPP+	BCB-Full
AR donor	96.3	72.1	60.4	75.8	66.5	44.0	85.4	70.6	50.8
Base	68.0	5.8	12.1	57.4	63.2	43.2	**65.9**	65.9	38.1
A2D (transfer)	**76.0**	**21.2**	**16.2**	**74.1**	**67.3**	**45.2**	64.6	**66.9**	**38.5**
Post-trained	93.6	71.7	56.7	71.6	62.2	38.6	63.4	66.9	41.3
A2D (merge)	**94.8**	**72.9**	**67.1**	**74.5**	**63.8**	**38.8**	**70.1**	**69.0**	**44.3**

(c) Ministral-3-3B → Nemotron-LD-3B, dual-mode AR-diffusion post-training vectors. The model supports both diffusion and AR decoding, and we report both results. The mean excludes the anchor task IFEval.

	Mode	GSM8K	MATH-500	HumanEval	MBPP	IFEval	MMLU	GPQA	AIME24	AIME25	LCB-CPP	Mean
AR donor	AR	91.3	83.0	80.5	75.9	65.1	67.9	40.9	30.0	13.3	27.1	56.7
Base	DLM	59.1	40.0	64.6	70.9	23.2	62.0	**35.4**	0.0	0.0	9.9	38.0
A2D (transfer)	DLM	**88.1**	**52.6**	**72.6**	**74.3**	**60.6**	**62.8**	33.3	**3.3**	**3.3**	**14.1**	**44.9**
Instruct	DLM	89.2	76.8	73.2	73.0	68.0	72.1	35.4	10.0	**13.3**	16.3	51.0
A2D (merge)	DLM	**90.1**	**79.6**	**75.0**	**75.7**	**72.9**	**72.9**	**43.4**	**16.7**	**13.3**	**18.9**	**54.0**
Instruct	AR	87.6	76.0	73.8	69.8	68.8	71.9	**41.9**	**23.3**	**13.3**	17.6	52.8
A2D (merge)	AR	**89.3**	**81.0**	**79.9**	**75.1**	**72.8**	**72.5**	38.4	20.0	**13.3**	**18.7**	**54.2**

optimization pipelines. Our evaluation spans general instruction following, coding, mathematical reasoning, and medical question answering, and covers both supervised and reinforcement-learning-based post-training. We organize the experiments by post-training type and evaluate both A2D regimes: A2D-Transfer improves upon the diffusion base by directly applying an AR post-training update, while A2D-Merge further improves the diffusion post-trained model by composing it with the corresponding AR update.

For each model family, the A2D coefficient is selected on a single anchor benchmark and kept fixed across the evaluations of the remaining tasks. We follow the native inference protocol of each diffusion model and reproduce the parent checkpoints before evaluating the transferred and merged models. Full evaluation, merging, and coefficient-selection details are provided in Appendix B.

3.1 Instruction-Tuned Updates for General and Code Models

We first evaluate whether instruction-tuning updates transfer from AR to diffusion across both general-purpose and code-specialized model families. For the general-purpose setting, we construct the AR instruction-tuning vector by subtracting Qwen2.5-7B-Base from Qwen2.5-7B-Instruct (Yang et al., 2024a), and transfer this update to Dream-7B-Base (Ye et al., 2025). We evaluate the resulting model on instruction following, mathematical reasoning, and coding.

For the code-specialized setting, we similarly construct the instruction-tuning vector from Qwen2.5-Coder-7B-Base and Qwen2.5-Coder-7B-Instruct (Hui et al., 2024), and transfer it to two independently developed diffusion code models, Dream-Coder (Xie et al., 2025) and DiffuCoder (Gong

et al., 2026). In all three settings, the diffusion models originate from the corresponding AR base family, while their instruction-tuned checkpoints are obtained through separate diffusion-specific post-training pipelines.

As shown in Table 1a, A2D is broadly effective across different types of instruction-following post-training, exhibiting two complementary regimes. Direct transfer can recover substantial capability missing from the diffusion base, with particularly large gains on Dream-7B MATH-500 ($+19.4$) and IfEval ($+9.5$), and on Dream-Coder BCB-Full ($+18.8$). When diffusion instruction tuning is already available, A2D-Merge further provides consistent gains, improving all $9/9$ benchmarks over the corresponding post-trained baselines. Moreover, coefficients selected on a single anchor task generalize to the remaining benchmarks, yielding average non-anchor gains of 7.4 and 2.4 points for transfer and merge, respectively.

3.2 Specialized Post-Training: Reasoning, Medical VQA, and Code RL

We next test whether A2D extends beyond general instruction tuning to more specialized capabilities and later-stage post-training updates. For mathematical reasoning, we use the post-training update from Qwen3-8B-Base to its reasoning checkpoint (Yang et al., 2025) and transfer it to DreamReasoner-8B (Wu et al., 2026b), evaluating on MATH and competition-level math benchmarks. For medical visual question answering (VQA), we use matched Gemma-4 (Team, 2026) and DiffusionGemma medical-SFT updates, obtained by fine-tuning their respective instruction-tuned checkpoints on VQA-RAD (Van Puyvelde et al., 2026), and evaluate on 3 medical VQA tasks. For code, we transfer and merge the RL updates from the instruction-tuned checkpoints of Ace-Coder (Zeng et al., 2025) and DiffuCoder (Gong et al., 2026). Together, these settings cover both specialization from a base model and continued SFT or RL applied after instruction tuning.

As shown in Table 1b, A2D remains effective across reasoning, medical SFT, and code RL. A2D-Transfer yields substantial gains in the reasoning and medical settings, including $+15.4$ on AIME24 and $+16.7$ on VQA-RAD, while improving $8/9$ benchmarks overall. A2D-Merge is more consistently effective, improving all $9/9$ post-trained baselines, with gains of $+10.4$ on AIME25 and $+6.7$ on HumanEval+. These gains extend beyond the anchor tasks, averaging 4.1 and 3.9 points on non-anchor benchmarks for transfer and merge, respectively. Together, these results extend A2D to specialized domain adaptation and different stages of post-training, while showing robust performance across model scales, as demonstrated by results for DiffusionGemma with 26B parameters.

3.3 A2D on Dual-Mode AR-Diffusion Models

Finally, we evaluate A2D on Nemotron-Labs-Diffusion-3B (Nemotron-LD-3B) (Fu et al., 2026), a dual-mode model initialized from Ministral-3-3B (Liu et al., 2026) that supports diffusion, AR, and self-speculative decoding. We evaluate A2D-Transfer and A2D-Merge across ten benchmarks covering math, coding, instruction following, and general reasoning under diffusion decoding, and additionally test the merged model under AR decoding. This setting also probes transfer with imperfect parameter correspondence, since Nemotron-LD contains diffusion-specific parameters with no AR counterpart. We keep these unmatched parameters from the diffusion recipient and apply the AR update only to aligned parameters. Full parameter-mapping details are provided in App. B.2.

As shown in Table 1c, A2D-Transfer improves the diffusion base on $9/10$ benchmarks, with large gains on GSM8K ($+29.0$), MATH-500 ($+12.6$), and IFEval ($+37.4$), with a performance improvement of 6.9 points averaged on the non-anchor tasks. A2D-Merge further improves or matches all $10/10$ benchmarks under diffusion decoding, increasing the non-anchor mean from 51.0 to 54.0. Notably, the same merged weights also remain effective under AR decoding, improving the mean from 52.8 to 54.2, with notable gains on MATH-500 (+5.0), HumanEval (+6.1), and MBPP (+5.3).

Additionally, we also provide the A2D-Transfer's performance on AR mode in Table 17, results with Nemotron-LD-8B in Appendix C.4, which shows A2D-Merge leads to notable gains when merging the reasoning post-training vector from Ministral-3-8B donor. Furthermore, we show that A2D-merge generally improves the decoding efficiency for self-speculative decoding with Nemotron-LD in Table 20. Together, these results show that A2D demonstrates reliable performance improvement across a broad task suite, tolerates imperfect parameter correspondence, and transfers its gains across both diffusion and autoregressive decoding modes.

Table 2: A2D improves both denoising scheduling and token prediction.

Schedule from	Tokens from Parent	Tokens from A2D
Parent	38.4	42.8
A2D	42.2	**45.0**

Table 3: The transfer is asymmetric: it transfers from AR to diffusion, but not in reverse.

	GSM8K	MATH-500
D_b	66.5	20.8
D_b + A2D transfer	**81.7**	**40.2**
A_b	**62.5**	**17.4**
A_b + D2A transfer	56.0	16.2

Table 4: A2D benefits from both adding the AR vector and downscaling the diffusion vector.

Method	Avg. Acc.	Avg. rank
Diffusion parent D_p	72.3	3.7
$+\tau_A$ only	75.8	2.4
$-\tau_D$ only	74.1	2.4
A2D merge	**76.8**	**1.4**

4 Analysis and Ablations

We further analyze the behavior and scope of cross-paradigm transfer and identify the sources of A2D's gains. Unless otherwise stated, experiments use Dream-7B on MATH-500, where we transfer the instruction-tuned vector from Qwen2.5-7B, and all scores are reported in percentages.

4.1 Understanding Cross-Paradigm Transfer

How does A2D alter diffusion generation? We examine how A2D affects the diffusion generation process to decompose its gains.

A2D improves both denoising schedule and token prediction. As shown in Table 2, using A2D-merged model token predictions with the denoising schedule of Dream-7B-Instruct which we call here the *diffusion parent*, improves accuracy from 38.4 to 42.8, while using the merged model's denoising schedule with diffusion parent's predictions reaches 42.2. Combining both yields the full A2D performance of 45.0, showing that prediction and scheduling provide complementary gains. These changes are nevertheless highly localized: when evaluated on the parent's intermediate states, A2D selects different positions at only 5.3% of steps and commits different tokens at only 3.4% of positions compared to the diffusion parent.

A2D does not systematically shift denoising toward causal generation. Despite modifying the denoising schedule, A2D does not induce a systematic shift toward left-to-right decoding. Following Gong et al. (2026), we measure global and local AR-ness over the response up to the first EOS. Global@k measures whether the selected position is among the k leftmost masked tokens, Local@k measures whether the current update continues a left-to-right run over the previous k updates. Both equal 1 under strict autoregressive decoding. As shown in Figure 5(a) of the appendix, the merged model closely tracks the diffusion parent on both metrics.

A2D selectively revises uncertain predictions. The few changed tokens by A2D are not spread evenly. As shown in Figure 5(b), they concentrate at positions where the diffusion parent has relatively low confidence. Thus, A2D largely preserves the parent's predictions while selectively revising uncertain decisions.

Directional asymmetry in cross-paradigm transfer. We test reversing the transfer direction and apply the diffusion task vector to the corresponding AR base. As shown in Table 3, while applying AR task vector to diffusion base model substantially improves it, applying the diffusion task vector to the AR base model can degrade its performance, revealing a clear directional asymmetry.

This asymmetry provides insight into what enables A2D transfer. The AR update $\boldsymbol{\tau}_A$ is learned around $\boldsymbol{A}_b$, from which $\boldsymbol{D}_b$ is subsequently derived through diffusion continued pretraining. Thus, $\boldsymbol{D}_b$ inherits much of the parameter and representation context on which $\boldsymbol{\tau}_A$ acts. In contrast, $\boldsymbol{\tau}_D$ is learned around the already converted model $\boldsymbol{D}_b$ and may depend on changes introduced during diffusion pretraining that are absent from $\boldsymbol{A}_b$. These results suggest that A2D works in part because the diffusion recipient shares the training lineage of the model on which the transferred AR update was learned, whereas this compatibility is absent in the reverse direction.

Scope of cross-paradigm transfer. Our main experiments consider task vectors induced by downstream fine-tuning, for which A2D broadly enables cross-paradigm transfer. We additionally test vectors induced by continued pretraining, respectively from Qwen2.5-7B to Qwen2.5-Math-7B and Qwen2.5-Coder-7B, and find that neither transfers effectively to the corresponding diffusion model. These vectors have 70–82$\times$ larger norms than the post-training vectors used in our main experi-

ments, reflecting substantially broader parameter changes that may reorganize the representations on which the updates rely. These results delineate A2D's scope to relatively local post-training updates, rather than pretraining-scale model transformations.

4.2 Ablation Studies

Decomposition of the A2D gain. A2D-Merge can be written as $\boldsymbol{D}_p + \lambda(\boldsymbol{\tau}_A - \boldsymbol{\tau}_D)$, combining two effects: adding the AR update and partially removing the diffusion update. We isolate these effects by retaining only $+\boldsymbol{\tau}_A$ or $-\boldsymbol{\tau}_D$ with a corresponding scaling coefficient. As shown in Table 4 (full results in Table 12), both components individually improve over the diffusion parent, but neither matches A2D-Merge. Thus, the full gain benefits from jointly balancing the AR and diffusion post-training updates.

Module-wise contributions to A2D. We ablate the contribution of attention and MLP modules from $\boldsymbol{\tau}_A$. As shown in Table 14, while A2D-Transfer depends mostly on updates in MLP modules, A2D-Merge requires both MLP and attention modules to achieve optimal merging performance.

5 Related Work

Diffusion large language models. Diffusion language models (dLLMs) have emerged as competitive alternatives to autoregressive (AR) models (Nie et al., 2025a;b; Ye et al., 2025; Song et al., 2025; Gong et al., 2026; Wu et al., 2026b). Many recent dLLMs are initialized from pretrained AR checkpoints and converted through continued diffusion training (Gong et al., 2025; Ye et al., 2025; Xie et al., 2025; Wu et al., 2026a; Cheng et al., 2026; Fu et al., 2026; Team et al., 2026), suggesting substantial shared knowledge across the two paradigms. Post-training dLLMs, however, poses distinct optimization challenges as learning must span partially corrupted states across varying corruption levels (Parashar et al., 2026; Deng et al., 2026), with additional challenges in likelihood estimation for RL (Gong et al., 2026; Tang et al., 2026). Recent work further shows that AR models can guide diffusion adaptation and that AR-like biases can remain useful under flexible generation orders (Peng et al., 2026; Su et al., 2026; Xue et al., 2025; Ni et al., 2026). We study this connection in *weight space*, asking whether AR post-training updates remain effective after diffusion conversion and can be composed with diffusion updates.

Model merging. Model merging combines independently trained models directly in weight space without additional training (Yang et al., 2026). Prior work includes parameter averaging (Wortsman et al., 2022; Matena & Raffel, 2022; Tam et al., 2023) and task arithmetic, which represents fine-tuning updates as composable task vectors (Ilharco et al., 2022). Subsequent methods improve merging by reducing interference through conflict resolution, sparsification, or layer-wise rescaling (Ortiz-Jimenez et al., 2023; Yadav et al., 2023; Yu et al., 2024; Wang et al., 2024; 2025a; Yang et al., 2024b). Prior work on LLMs primarily focused on model merging within the AR paradigm (Ren et al., 2026; Zhu et al., 2026), whereas we study *cross-paradigm transfer* from AR to diffusion. Closest to our setting, Lin et al. (2026) transfers AR preference-alignment updates as an alternative to diffusion-based alignment, while we study a broader post-training setting where AR knowledge is used to further improve already post-trained diffusion models across diverse objectives, capabilities, and model families.

6 Conclusion

We show that AR post-training knowledge exhibits *cross-paradigm transfer*: it remains effective after diffusion conversion and can be reused directly in diffusion models through weight-space transfer. Building on this cross-paradigm transferability, A2D transfers AR post-training updates to diffusion models and, when diffusion post-training is available, combines the two to exploit complementary gains. Across diverse model families, tasks, and post-training settings, A2D consistently improves diffusion models without additional training or inference-time computation.

Practically, A2D provides a simple training-free framework for reusing existing AR post-trained checkpoints as resources for diffusion models, supporting both capability transfer to diffusion base models and further improvement of already post-trained diffusion models through composition.

ETHICS STATEMENT

This work develops methods for diffusion language models and does not introduce new ethical considerations beyond those generally associated with large language models and generative AI. The proposed methods may inherit the broader benefits and risks of the underlying models. We encourage responsible use of the methods and models described in this work.

REPRODUCIBILITY STATEMENT

Our inference setup closely follows the official inference procedures of the diffusion models used in this work. Before applying our method, we first verify that our local evaluation approximately reproduces the reported performance of the corresponding checkpoints, and then perform all merging experiments under the same evaluation setup. The model merging procedure is simple, and detailed implementation, inference, evaluation, and merging details are provided in the appendix.

AI USE STATEMENT

In this work, we used generative AI tools for:

1. Implement methods. We use coding agents to assist with implementation. We check all code and take full responsibility for its correctness.

We have not used generative AI tools for:

1. Design or provide feedback on research methodology or experiments.
2. Generate synthetic datasets.
3. Clean and reformat datasets.
4. Interpret results.

These required disclosure tasks are not applicable to this work:

1. Help develop theoretical models or conceptual frameworks.
2. Formulate mathematical claims.
3. Provide critical ingredients for proving mathematical claims.
4. Assist in the writing of proofs.
5. Propose or refine hypotheses.
6. Assist with translation.
7. Support qualitative and thematic data analysis.

Additionally, we used generative AI tools for:

1. Editing the research paper to improve readability. We use agents to find and correct grammar mistakes and point out sections which require more human editing.
2. Identifying relevant literature. We use agents to perform an additional literature search on top of our human search.
3. Creating or modifying scientific figures and visualizations. We use agents to help with making plots and visualizations.

We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. Structured denoising diffusion models in discrete state-spaces. *Advances in neural information processing systems*, 34, 2021a. 17

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021b. 3

Asma Ben Abacha, Sadid A. Hasan, Vivek V. Datla, Joey Liu, Dina Demner-Fushman, and Henning Müller. VQA-Med: Overview of the medical visual question answering task at ImageCLEF 2019. In *Working Notes of CLEF 2019*, volume 2380 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2019. URL https://ceur-ws.org/Vol-2380/paper_272.pdf. 19

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. 2021. 3

Shuang Cheng, Yihan Bian, Dawei Liu, Yuhua Jiang, Yihao Liu, Linfeng Zhang, Qian Yao, Zhongbo Tian, Wenhai Wang, Qipeng Guo, et al. Sdar: A synergistic diffusion-autoregression paradigm for scalable sequence generation. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 22058–22075, 2026. 9, 17

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. 3

Jia Deng, Junyi Li, Wayne Xin Zhao, Jinpeng Wang, Hongyu Lu, and Ji-Rong Wen. Beyond fully random masking: Attention-guided denoising and optimization for diffusion language models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 44502–44514, 2026. 9, 17

Yonggan Fu, Lexington Whalen, Abhinav Garg, Chengyue Wu, Maksim Khadkevich, Nicolai Oswald, Enze Xie, Daniel Egert, Sharath Turuvekere Sreenivas, Shizhe Diao, Chenhan Yu, Ye Yu, Weijia Chen, Sajad Norouzi, Jingyu Liu, Shiyi Lan, Ligeng Zhu, Jin Wang, Jindong Jiang, Morteza Mardani, Mehran Maghoumi, Song Han, Ante Jukic, Nima Tajbakhsh, Jan Kautz, and Pavlo Molchanov. Nemotron-labs-diffusion: A tri-mode language model unifying autoregressive, diffusion, and self-speculation decoding. Technical report, NVIDIA, 2026. Technical report. 1, 3, 7, 9, 17, 19

Shansan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and Lingpeng Kong. DiffuSeq: Sequence to sequence text generation with diffusion models. In *International Conference on Learning Representations, ICLR*, 2023. 17

Shansan Gong, Shivam Agarwal, Yizhe Zhang, Jiacheng Ye, Lin Zheng, Mukai Li, Chenxin An, Peilin Zhao, Wei Bi, Jiawei Han, et al. Scaling diffusion language models via adaptation from autoregressive models. In *International Conference on Learning Representations*, volume 2025, pp. 5046–5073, 2025. 9, 17

Shansan Gong, Ruixiang Zhang, Huangjie Zheng, Jiatao Gu, Navdeep Jaitly, Lingpeng Kong, and Yizhe Zhang. Diffucoder: Understanding and improving masked diffusion models for code generation. In *International Conference on Learning Representations*, volume 2026, pp. 15723–15750, 2026. 1, 6, 7, 8, 9, 17, 18

Xiaochuang Han, Sachin Kumar, and Yulia Tsvetkov. Ssd-lm: Semi-autoregressive simplex-based diffusion language model for text generation and modular control. *arXiv preprint arXiv:2210.17432*, 2022. 17

Zhengfu He, Tianxiang Sun, Qiong Tang, Kuanning Wang, Xuan-Jing Huang, and Xipeng Qiu. Diffusionbert: Improving generative masked language models with diffusion models. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 4521–4534, 2023. 17

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021. 3

Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax flows and multinomial diffusion: Learning categorical distributions. *Advances in neural information processing systems*, 34, 2021. 17

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Kai Dang, et al. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*, 2024. 6, 18

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*, 2022. 2, 3, 9, 17

Jason J. Lau, Soumya Gayen, Asma Ben Abacha, and Dina Demner-Fushman. A dataset of clinically generated visual questions and answers about radiology images. *Scientific Data*, 2018. doi: 10.1038/sdata.2018.251. URL https://www.nature.com/articles/sdata2018251. 19

Jia LI, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Costa Huang, Kashif Rasul, Longhui Yu, Albert Jiang, Ziju Shen, Zihan Qin, Bin Dong, Li Zhou, Yann Fleureau, Guillaume Lample, and Stanislas Polu. Numinamath. [https://huggingface.co/AI-MO/NuminaMath-CoT](https://github.com/project-numina/aimo-progress-prize/blob/main/report/numina_dataset.pdf), 2024. 3, 21

Liang Lin, Feng Xiong, Zengbin Wang, Kun Wang, Junhao Dong, Xuecai Hu, Yong Wang, and Xiangxiang Chu. Ar-map: Are autoregressive large language models implicit teachers for diffusion large language models? *arXiv preprint arXiv:2602.02178*, 2026. 1, 9, 17

Alexander H. Liu, Kartik Khandelwal, Sandeep Subramanian, Victor Jouault, Abhinav Rastogi, Adrien Sadé, Alan Jeffares, Albert Jiang, Alexandre Cahill, Alexandre Gavaudan, Alexandre Sablayrolles, Amélie Héliou, Amos You, Andy Ehrenberg, Andy Lo, Anton Eliseev, Antonia Calvi, Avinash Sooriyarachchi, Baptiste Bout, Baptiste Rozière, Baudouin De Monicault, Clémence Lanfranchi, Corentin Barreau, Cyprien Courtot, Daniele Grattarola, Darius Dabert, Diego de las Casas, Elliot Chane-Sane, Faruk Ahmed, Gabrielle Berrada, Gaëtan Ecrepont, Gauthier Guinet, Georgii Novikov, Guillaume Kunsch, Guillaume Lample, Guillaume Martin, Gunshi Gupta, Jan Ludziejewski, Jason Rute, Joachim Studnia, Jonas Amar, Joséphine Delas, Josselin Somerville Roberts, Karmesh Yadav, Khyathi Chandu, Kush Jain, Laurence Aitchison, Laurent Fainsin, Léonard Blier, Lingxiao Zhao, Louis Martin, Lucile Saulnier, Luyu Gao, Maarten Buyl, Margaret Jennings, Marie Pellat, Mark Prins, Mathieu Poirée, Mathilde Guillaumin, Matthieu Dinot, Matthieu Futeral, Maxime Darrin, Maximilian Augustin, Mia Chiquier, Michel Schimpf, Nathan Grinsztajn, Neha Gupta, Nikhil Raghuraman, Olivier Bousquet, Olivier Duchenne, Patricia Wang, Patrick von Platen, Paul Jacob, Paul Wambergue, Paula Kurylowicz, Pavankumar Reddy Muddireddy, Philomène Chagniot, Pierre Stock, Pravesh Agrawal, Quentin Torroba, Romain Sauvestre, Roman Soletskyi, Rupert Menneer, Sagar Vaze, Samuel Barry, Sanchit Gandhi, Siddhant Waghjale, Siddharth Gandhi, Soham Ghosh, Srijan Mishra, Sumukh Aithal, Szymon Antoniak, Teven Le Scao, Théo Cachet, Theo Simon Sorg, Thibaut Lavril, Thiziri Nait Saada, Thomas Chabal, Thomas Foubert, Thomas Robert, Thomas Wang, Tim Lawson, Tom Bewley, Tom Edwards, Umar Jamil, Umberto Tomasini, Valeriia Nemychnikova, Van Phung, Vincent Maladière, Virgile Richard, Wassim Bouaziz, Wen-Ding Li, William Marshall, Xinghui

Li, Xinyu Yang, Yassine El Ouahidi, Yihan Wang, Yunhao Tang, and Zaccharie Ramzi. Ministral 3, 2026. URL https://arxiv.org/abs/2601.08584. 1, 7, 19

Bo Liu, Li-Ming Zhan, Li Xu, Lin Ma, Yan Yang, and Xiao-Ming Wu. SLAKE: A semantically-labeled knowledge-enhanced dataset for medical visual question answering. In *2021 IEEE 18th International Symposium on Biomedical Imaging (ISBI)*, 2021. doi: 10.1109/ISBI48211.2021.9434010. URL https://arxiv.org/abs/2102.09542. 19

Michael S Matena and Colin Raffel. Merging models with fisher-weighted averaging. *Advances in Neural Information Processing Systems*, 35:17703–17716, 2022. 9, 17

Zanlin Ni, Shenzhi Wang, Yang Yue, Tianyu Yu, Weilin Zhao, Yeguo Hua, Tianyi Chen, Jun Song, Cheng Yu, Bo Zheng, et al. The flexibility trap: Rethinking the value of arbitrary order in diffusion language models. *arXiv preprint arXiv:2601.15165*, 2026. 9, 17

Shen Nie, Fengqi Zhu, Chao Du, Tianyu Pang, Qian Liu, Guangtao Zeng, Min Lin, and Chongxuan Li. Scaling up masked diffusion models on text. In *International Conference on Learning Representations*, volume 2025, pp. 82974–82997, 2025a. 9, 17

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*, 2025b. 9, 17

Guillermo Ortiz-Jimenez, Alessandro Favero, and Pascal Frossard. Task arithmetic in the tangent space: Improved editing of pre-trained models. *Advances in Neural Information Processing Systems*, 36:66727–66754, 2023. 5, 9, 17

Shubham Parashar, Atharv Chagi, Jacob Helwig, Lakshmi Jotsna, Sushil Vemuri, James Caverlee, Dileep Kalathil, and Shuiwang Ji. Learnability-informed fine-tuning of diffusion language models. *arXiv preprint arXiv:2605.22939*, 2026. 1, 9, 17

Fred Zhangzhi Peng, Alexis Fox, Anru R Zhang, and Alexander Tong. Don't retrain, align: Adapting autoregressive lms to diffusion lms via representation alignment. *arXiv preprint arXiv:2605.06885*, 2026. 9, 17

Valentina Pyatkin, Saumya Malik, Victoria Graf, Hamish Ivison, Shengyi Huang, Pradeep Dasigi, Nathan Lambert, and Hanna Hajishirzi. Generalizing verifiable instruction following. *Advances in Neural Information Processing Systems*, 38, 2026. 18

Qwen Team. Qwen3.8-Max: A new bar for coding and cowork, August 2026. URL https://qwen.ai/blog?id=qwen3.8. 20

Zixuan Ren, Jinliang Lu, Junhong Wu, Yang Zhao, Dai Dai, Haifeng Wang, Chengqing Zong, et al. Enough is as good as a feast: A comprehensive analysis of how reinforcement learning mitigates task conflicts in llms. In *International Conference on Learning Representations*, volume 2026, pp. 25142–25161, 2026. 9, 17

Subham Sekhar Sahoo, Marianne Arriola, Aaron Gokaslan, Edgar Mariano Marroquin, Alexander M Rush, Yair Schiff, Justin T Chiu, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=L4uaAR4ArM. 17

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, 2024. 17

Yuxuan Song, Zheng Zhang, Cheng Luo, Pengyang Gao, Fan Xia, Hao Luo, Zheng Li, Yuehang Yang, Hongli Yu, Xingwei Qu, et al. Seed diffusion: A large-scale diffusion language model with high-speed inference. *arXiv preprint arXiv:2508.02193*, 2025. 9, 17

Xingyu Su, Jacob Helwig, Shubham Parashar, Atharv Chagi, Lakshmi Jotsna, Degui Zhi, James Caverlee, Dileep Kalathil, and Shuiwang Ji. Data-efficient autoregressive-to-diffusion language models via on-policy distillation. *arXiv preprint arXiv:2606.06712*, 2026. 9, 17

Derek Tam, Mohit Bansal, and Colin Raffel. Merging by matching models in task parameter subspaces. *arXiv preprint arXiv:2312.04339*, 2023. 9, 17

Xiaohang Tang, Keyue Jiang, Che Liu, Qifang Zhao, Xiaoxiao Xu, Sangwoong Yoon, and Ilija Bogunovic. Gdsd: Reinforcement learning as guided denoiser self-distillation for diffusion language models. *arXiv preprint arXiv:2605.29398*, 2026. 9, 17

DiffusionGemma Team, Adrien Ali Taïga, James Assiene, Daniele Calandriello, Rahma Chaabouni, João Gante, Tamara von Glehn, Nate Keating, Chris Knutsen, Martin Kukla, Tianlin Liu, Ivan Lobov, Ofir Nabati, João Gabriel Oliveira, Nicolas Perez-Nieves, Nastasia Prutianova, Bobak Shahriari, Jean Tarbouriech, Pavel Tyletski, Çağlar Ünlü, Cindy Wu, Glenn Cameron, Jerome Connor, Sertan Girgin, Maarten Grootendorst, Alon Levkovitch, Eliya Nachmani, Omar Sanseviero, Piotr Stanczyk, Quentin Berthet, Andrew Campbell, Clément Crepy, Valentin De Bortoli, Arnaud Doucet, Romuald Elie, Alexandre Galashov, Klaus Greff, Alexis Jacq, David Ruhe, Yu-Han Wu, Sebastian Flennerhag, Brendan O'Donoghue, George Scrivener, and Shantanu Thakoor. Diffusiongemma technical report, 2026. URL https://arxiv.org/abs/2608.00146. 1, 3, 9, 17, 19

Gemma Team. Gemma 4 technical report, 2026. URL https://arxiv.org/abs/2607.02770. 1, 7, 19

Max Van Puyvelde, Halil Ibrahim Gulluk, Wim Van Criekinge, and Olivier Gevaert. Discrete diffusion language models for interactive radiology report drafting. *arXiv preprint arXiv:2607.01436*, 2026. 7, 19

Ke Wang, Nikolaos Dimitriadis, Guillermo Ortiz-Jimenez, François Fleuret, and Pascal Frossard. Localizing task information for improved model merging and compression. *arXiv preprint arXiv:2405.07813*, 2024. 9, 17

Ke Wang, Nikos Dimitriadis, Alessandro Favero, Guillermo Ortiz-Jimenez, Francois Fleuret, and Pascal Frossard. Lines: Post-training layer scaling prevents forgetting and enhances model merging. In *International Conference on Learning Representations*, volume 2025, pp. 27100–27133, 2025a. 9, 17

Yaoxiang Wang, Haoling Li, Xin Zhang, Jie Wu, Xiao Liu, Wenxiang Hu, Zhongxin Guo, Yangyu Huang, Ying Xin, Yujiu Yang, et al. Epicoder: Encompassing diversity and complexity in code generation. *arXiv preprint arXiv:2501.04694*, 2025b. 3, 21

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International conference on machine learning*, pp. 23965–23998. Pmlr, 2022. 2, 3, 4, 9, 17

Chengyue Wu, Hao Zhang, Shuchen Xue, Shizhe Diao, Yonggan Fu, Zhijian Liu, Pavlo Molchanov, Ping Luo, Song Han, and Enze Xie. Fast-dllm v2: Efficient block-diffusion llm. In *International Conference on Learning Representations*, volume 2026, pp. 128353–128370, 2026a. 9, 17

Zirui Wu, Lin Zheng, Jiacheng Ye, Shansan Gong, Xueliang Zhao, Yansong Feng, Wei Bi, and Lingpeng Kong. Dreamreasoner-8b: Block-size curriculum learning for diffusion reasoning models, 2026b. URL https://arxiv.org/abs/2606.19257. 1, 3, 7, 9, 17, 19

Zhihui Xie, Jiacheng Ye, Lin Zheng, Jiahui Gao, Jingwei Dong, Zirui Wu, Xueliang Zhao, Shansan Gong, Xin Jiang, Zhenguo Li, et al. Dream-coder 7b: An open diffusion language model for code. *arXiv preprint arXiv:2509.01142*, 2025. 6, 9, 17, 18

Zhengqi Xu, Ke Yuan, Huiqiong Wang, Yong Wang, Mingli Song, and Jie Song. Training-free pretrained model merging. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5915–5925. IEEE, 2024. 4

Shuchen Xue, Tianyu Xie, Tianyang Hu, Zijin Feng, Jiacheng Sun, Kenji Kawaguchi, Zhenguo Li, and Zhi-Ming Ma. Any-order gpt as masked diffusion model: Decoupling formulation and architecture. *arXiv preprint arXiv:2506.19935*, 2025. 9, 17

Prateek Yadav, Derek Tam, Leshem Choshen, Colin A Raffel, and Mohit Bansal. Ties-merging: Resolving interference when merging models. *Advances in neural information processing systems*, 36:7093–7115, 2023. 9, 17

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen25 technical report. *arXiv preprint arXiv:2412.15115*, 2024a. 6, 18

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. 1, 7, 19

Enneng Yang, Zhenyi Wang, Li Shen, Shiwei Liu, Guibing Guo, Xingwei Wang, and Dacheng Tao. Adamerging: Adaptive model merging for multi-task learning. In *International Conference on Learning Representations*, volume 2024, pp. 22743–22763, 2024b. 9, 17

Enneng Yang, Li Shen, Guibing Guo, Xingwei Wang, Xiaochun Cao, Jie Zhang, and Dacheng Tao. Model merging in llms, mllms, and beyond: Methods, theories, applications, and opportunities. *ACM Computing Surveys*, 58(8):1–41, 2026. 2, 3, 9, 17

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. 1, 3, 6, 9, 17, 18

Linyan Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. Language models are super mario: Absorbing abilities from homologous models as a free lunch. In *Icml*, volume 2, pp. 21, 2024. 9, 17

Huaye Zeng, Dongfu Jiang, Haozhe Wang, Ping Nie, Xiaotong Chen, and Wenhu Chen. Acecoder: Acing coder rl via automated test-case synthesis. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12023–12040, 2025. 7, 18

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

Kejian Zhu, Zhuoran Jin, Shangqing Tu, Hongbang Yuan, Yushi Bai, Kang Liu, Juanzi Li, and Jun Zhao. Sft conflicts, rl coexists: A theoretical and empirical analysis of multi-task learning for llms. *arXiv preprint arXiv:2608.03573*, 2026. 9, 17

Appendix

Table of Contents

A Expanded Related Work **17**

B Experimental Details **18**
B.1 Models and the evaluation pipeline 18
B.2 Details for model merging 20
B.3 Experimental details for section 2 21
B.4 GPU and training time 21
B.5 Hyperparameter search 21

C Additional Results **24**
C.1 Full results for analyses and ablations 24
C.2 Full results of Dream-7B 24
C.3 Full results of Nemontron-LD-3B 25
C.4 Results of Nemontron-LD-8B 25
C.5 DiffusionGemma with SLAKE Post-training 26
C.6 Self-speculative decoding of Nemotron 26
C.7 Post-training types and norm ratios 26

D Analysis **30**
D.1 Correction and error introduction along the A2D direction 30
D.2 Interpolation preserves complementary gains on released checkpoints 30
D.3 Aligned representation changes on released checkpoints 30

E Limitations and Future Work **32**

A Expanded Related Work

Diffusion large language models Extending diffusion-based generative modeling to discrete sequences has been widely studied (Austin et al., 2021a; Hoogeboom et al., 2021; Han et al., 2022; Gong et al., 2023), with masked diffusion emerging as a particularly effective formulation for language modeling (He et al., 2023; Sahoo et al., 2024; Shi et al., 2024). Subsequent scaling efforts established diffusion language models (dLLMs) as competitive alternatives to autoregressive (AR) models (Nie et al., 2025a;b; Ye et al., 2025; Song et al., 2025; Gong et al., 2026; Wu et al., 2026b). Rather than training from scratch, many recent dLLMs build on pretrained AR checkpoints and introduce diffusion objectives through continued training or adaptation (Gong et al., 2025; Ye et al., 2025; Xie et al., 2025; Wu et al., 2026a; Cheng et al., 2026; Fu et al., 2026; Team et al., 2026), reusing pretrained linguistic knowledge while substantially reducing the cost of learning diffusion-based generation. Post-training dLLMs, however, poses distinct optimization challenges as learning must span partially corrupted states across varying corruption levels (Parashar et al., 2026; Deng et al., 2026), with additional challenges in likelihood estimation for RL (Gong et al., 2026; Tang et al., 2026).

Recent work further reveals shared structure between the two paradigms: AR models can guide diffusion adaptation (Peng et al., 2026; Su et al., 2026), while left-to-right biases can remain beneficial even with flexible generation orders (Xue et al., 2025; Ni et al., 2026). Together, these results suggest that AR and diffusion models share reusable knowledge while retaining distinct inductive biases. We study this connection in *weight space*, asking whether AR post-training updates remain effective after diffusion conversion and can be composed with diffusion updates.

Model merging. Model merging aims to combine the capabilities of independently trained models directly in weight space, avoiding additional training (Yang et al., 2026). Early work showed that models fine-tuned from a common initialization often remain connected in weight space for parameter averaging to improve performance, motivating both simple averaging (Wortsman et al., 2022) and importance-weighted variants (Matena & Raffel, 2022; Tam et al., 2023). Task arithmetic formalized fine-tuning updates as task vectors that can be composed in weight space to manipulate model behavior (Ilharco et al., 2022). Its effectiveness has been linked to functional disentanglement among task vectors (Ortiz-Jimenez et al., 2023), with subsequent work mitigating interference through conflict resolution (Yadav et al., 2023), sparsification (Yu et al., 2024; Wang et al., 2024), and layer-wise rescaling (Wang et al., 2025a; Yang et al., 2024b).

Prior work on LLMs primarily focused on model merging within the AR paradigm (Ren et al., 2026; Zhu et al., 2026), whereas we study *cross-paradigm transfer* from AR to diffusion. Closest to our setting, Lin et al. (2026) transfers AR preference-alignment updates as an alternative to diffusion-based alignment, while we study a broader post-training setting where AR knowledge is used to further improve already post-trained diffusion models across diverse objectives, capabilities, and model families.

B EXPERIMENTAL DETAILS

This section describes how each model family is evaluated, how the merges are built, and how the coefficients are chosen.

B.1 MODELS AND THE EVALUATION PIPELINE

Every diffusion model in this work is initialized from an AR model, so the four-checkpoint construction of Sec. 2 is well defined. Each family has an AR base and its post-trained model, and a diffusion base $\boldsymbol{D}_b$ and its post-trained model $\boldsymbol{D}_p$. We call the AR post-trained model the *AR donor*, because its task vector $\boldsymbol{\tau}_A$ is the one we transfer, and we call $\boldsymbol{D}_p$ the *diffusion parent*, as in section 4, because it is the model A2D is built on and compared against. We take these relationships from each release's paper and model card. Diffusion models and their A2D variants use the diffusion model's official evaluation code and decoding settings, and we reproduce each parent's published numbers before evaluating any merge. Each family below gives its evaluation protocol and, specifically, every point at which we depart from an official one.

Dream-7B (Ye et al., 2025) / Qwen2.5-7B (Yang et al., 2024a). Dream-7B continues Qwen2.5-7B with a diffusion objective, and Dream-Instruct adds SFT on instruction pairs. IFEval, GSM8K, MATH-500 and HumanEval run in the Dream fork of `lm-evaluation-harness` at its released decoding settings and per-task budgets, with the chat template and one denoising step per generated token. IFEval is the mean of the four lm-eval metrics, and GSM8K and MATH-500 are 0-shot. MATH-500 replaces the release's full MATH task at its 512-token budget. HumanEval+ scores the generations of the fork's `humaneval_instruct` task with the EvalPlus tests, and MBPP+ uses the Dream-Coder EvalPlus harness at Dream's own 1,024-token budget. IFBench is included to provide a more challenging instruction following benchmark (Pyatkin et al., 2026), generated with the same decoding settings at 1,280 tokens and scored with the official IFBench code at prompt-level loose accuracy.

GSM8K prompt. On GSM8K we add the system prompt *"Solve the math problem with brief, direct reasoning. Do NOT restate the problem, do NOT use bullet points, and use as few steps as possible. On the final line, write the answer as '#### <number>' and nothing else."* Without the prompt the model often runs out of the 256-token budget before it reaches the final-answer format. The prompt helps the parent as well, and we use it for every Dream-7B GSM8K evaluation. The AR donor is evaluated with the same prompt for faireness.

Dream-Coder (Xie et al., 2025) / Qwen2.5-Coder (Hui et al., 2024). Dream-Coder-7B is built from Qwen2.5-Coder-7B, and Dream-Coder-Instruct applies SFT followed by GRPO, so its task vector carries both. HumanEval, MBPP and BigCodeBench(BCB) use the EvalPlus fork released with Dream-Coder at its documented settings. We report HumanEval+ and MBPP+, and BCB runs on the Instruct split.

DiffuCoder (Gong et al., 2026) / Qwen2.5-Coder (Hui et al., 2024) / AceCoder-Rule (Zeng et al., 2025). DiffuCoder-7B continues Qwen2.5-Coder-7B with diffusion pre-training on code, DiffuCoder-Instruct adds SFT, and DiffuCoder-cpGRPO adds coupled-GRPO. This gives two task vectors on one architecture: an SFT vector from Base to Instruct, paired with Qwen2.5-Coder-7B to Qwen2.5-Coder-7B-Instruct, and an RL vector from Instruct to cpGRPO, paired with Qwen2.5-Coder-7B-Instruct to AceCoder-Rule. We use the Dream-style EvalPlus harness with DiffuCoder's own chat template and decoding settings. DiffuCoder reports the best of $T \in \{0.2, 0.3, 0.4\}$ and a different temperature per checkpoint, whereas we compare every variant at a fixed T=0.2 and use the best-of-three only to check our reproduction of the parents. BCB runs on the Complete split, and we use Dream-Coder's, with 1,024 steps for 1,024 tokens and top-p 0.9.

Base evaluation. The BCB harness does not prefill ```` ```python ````, without which DiffuCoder-Base often degenerates into repeated filler tokens and performs substantially worse. We therefore evaluate it using `direct_completion`, which directly continues the function body. We use this prompting setup for the Base and transfer rows, since the transfer coefficient is only 0.1 and their behavior remains close to the base model, while the post-trained and merged rows use the chat template. All results are reported on the Complete split.

DreamReasoner (Wu et al., 2026b) / Qwen3 (Yang et al., 2025). DreamReasoner-8B is initialized from Qwen3-8B-Base. We use the release's block-diffusion decoder at its documented settings, with thinking enabled unless stated otherwise. AIME24/25 follow the paper, which averages eight runs at a 24,576-token budget because the output varies between runs even at $T{=}0$, and we report that mean as avg@8. MATH-500 uses a single greedy sample.

Base evaluation. The DreamReasoner chat template wraps the answer in ¡think¿ markers that the base model has never seen, and it repeats them until the budget runs out. We therefore give the base model the plain problem with the same $\square$ instruction. The transfer model reads the template normally, and we evaluate it in both modes.

Nemotron-Labs-Diffusion (Fu et al., 2026) / Ministral 3 (Liu et al., 2026). The 3B and 8B Nemotron-LD models are initialized from Ministral 3 Base. The four checkpoints are Nemotron-LD Base/Instruct and Ministral 3 Base-2512/Instruct-2512, and merges into the Instruct models retain the recipient's tokenizer and chat template. Base-transfer models use the interface described below.

Base evaluation. Ideally the diffusion Base would use the same instruction-following interface as the post-trained models. Nemotron-LD-8B-Base does not have that interface yet. Its embedding rows for the chat markers <|im_start|> and <|im_end|> are exactly zero, because those ids sit on Ministral's [IMG] and <pad> slots and are trained only at the Instruct stage. With no signal on the turn markers the Base cannot tell where the user turn ends, so it writes the role labels out as plain text and repeats them. It still answers math, but drops sharply on code, GPQA, IFEval and LiveCodeBench (Table 18). That score reflects a missing interface rather than the Base's ability, and using it as a baseline would inflate the gains we report. We therefore use the raw prompt, which the Base can read, in the main results and give both settings in the appendix. We evaluate it on the raw problem text at its native positional setting, with EOS set to ID 2 (</s>) as in its generation configuration.

Transfer evaluation. The transfer model cannot use the Nemotron chat template either, for the same reason. Those two rows are among the mismatched rows frozen at Base (App. B.2), and the AR vector does not rewrite them. What the AR vector does bring is the donor's own interface. [INST], [/INST] and </s> are IDs 3, 4 and 2, and these ids carry the same strings in both tokenizers, so they do receive the AR update. We therefore prompt the transfer rows in the donor's format, <s>[INST]{problem}[/INST], with no system prompt and no thinking marker. The difference is large. On the same 20 IFEval prompts at $\beta{=}1.0$, the Nemotron template leaves 12 of the 20 generations looping and scores 17.9, while [INST] leaves 1 and scores 60.8. The 3B transfer uses the same donor format, although its Base chat-marker embeddings are not zero.

Reasoner model for Nemontron-LD-8B For the 8B model we also consider the Ministral-3-8B-Reasoning donor, because the 8B diffusion model reasons on its own even when thinking is disabled. On LiveCodeBench (LCB) it emits </think> by itself in 15.0% of non-thinking generations, against 0% for the 3B model. A further 60.8% of its generations carry a </think> that the server inserts at generated token 6,000, a length artefact rather than a behaviour. The thinking-off comparison with an Instruct donor is therefore not fully matched at 8B. Unless otherwise stated, evaluations use 8,192 output tokens and the Instruct positional settings.

DiffusionGemma (Team et al., 2026) / Gemma 4 (Team, 2026). DiffusionGemma is initialized from Gemma 4-26B-A4B-Instruct and trained by supervised diffusion adaptation, sampler distillation, and reinforcement learning. It has no public base checkpoint, so for this pair we study domain-specific fine-tuning instead of the Base/Instruct setup, using the Gemma 4 and DiffusionGemma fine-tunes released by Van Puyvelde et al. (2026) for SLAKE (Liu et al., 2021), VQA-RAD (Lau et al., 2018), and VQA-Med-2019 (Ben Abacha et al., 2019), with the two instruction-tuned checkpoints as the pre-task references.

We use the Transformers implementation of the native block-diffusion decoder with block length 256, at most 48 denoising steps per block, entropy bound 0.1, and temperature annealed from 0.8 to 0.4, stopping early once the predictions stop changing between steps. Thinking is disabled. All three datasets are evaluated 0-shot with the image followed by the question and a 256-token budget, on all 1,061 English SLAKE test questions, all 451 public VQA-RAD test questions, and all 500 VQA-

Table 5: **Special-token mismatches across the four 3B checkpoints.** All listed input-embedding rows are masked during task-vector composition.

ID	Ministral Base	Ministral Instruct	Nemotron Base	Nemotron Instruct
10	[IMG]	[IMG]	<|im_start|>	<|im_start|>
11	<pad>	<pad>	<|im_end|>	<|im_end|>
12	[IMG_BREAK]	[IMG_BREAK]	<think>	<think>
13	[IMG_END]	[IMG_END]	</think>	</think>
14	[PREFIX]	[PREFIX]	<tool_call>	<tool_call>
15	[MIDDLE]	[MIDDLE]	</tool_call>	</tool_call>
16	[SUFFIX]	[SUFFIX]	<tool_response>	<tool_response>
17	[SYSTEM_PROMPT]	[SYSTEM_PROMPT]	</tool_response>	</tool_response>
18	[/SYSTEM_PROMPT]	[/SYSTEM_PROMPT]	<SPECIAL_18>	<SPECIAL_18>
19	[TOOL_CONTENT]	[TOOL_CONTENT]	<SPECIAL_19>	<SPECIAL_19>
24	[AUDIO]	[AUDIO]	<SPECIAL_24>	<SPECIAL_24>
25	[BEGIN_AUDIO]	[BEGIN_AUDIO]	<SPECIAL_25>	<SPECIAL_25>
32	[ARGS]	[ARGS]	<SPECIAL_32>	<SPECIAL_32>
33	[CALL_ID]	[CALL_ID]	<SPECIAL_33>	<SPECIAL_33>
34	[THINK]	<SPECIAL_34>	<SPECIAL_34>	<SPECIAL_34>
35	[/THINK]	<SPECIAL_35>	<SPECIAL_35>	<SPECIAL_35>

Med-2019 test questions. We report semantic answer accuracy judged by Qwen3.8-27B (Qwen Team, 2026) with greedy decoding and thinking disabled.

B.2 Details for model merging

Qwen-derived models. For Dream-7B, Dream-Coder, DiffuCoder, and DreamReasoner the diffusion model keeps its AR parent's architecture, parameter names, and tensor shapes, so the merge is a direct element-wise combination of all parameters, with no remapping, freezing, or masking.

Tokens that exist only in the diffusion tokenizer need no special treatment. The mask token and the other ids beyond the AR vocabulary never occur in the AR model's training data, so τ_A is exactly zero on their input-embedding rows for the Qwen2.5-based vectors and small for Qwen3. We merge them with the same formula as every other row.

Nemotron / Ministral. Nemotron needs an explicit mapping from Ministral's language-model parameters to the diffusion backbone. The vocabulary sizes agree, but 14 token IDs denote different tokens in the two families, and IDs 34–35 additionally differ between Ministral Base and Instruct. Some of these are generation controls: ID 11 is padding in Ministral but end-of-turn in Nemotron, and IDs 12–13 are image delimiters in Ministral but thinking delimiters in Nemotron (Table 5). For protected Instruct merges, we keep these 16 input-embedding rows at their values in the Nemotron Instruct recipient, excluding them from both the AR-vector addition and the diffusion-vector subtraction. For corrected Base-transfer experiments, we additionally freeze the diffusion mask-token embedding (ID 100), keeping all 17 protected rows at their Nemotron Base values.

The diffusion output head is likewise kept at its value in the recipient checkpoint (Base for transfer into Base, Instruct for merges into Instruct), so donor updates enter through the mapped backbone and the compatible input embeddings without touching the diffusion readout. Instruct merges retain the Nemotron Instruct evaluation interface, whereas Base-transfer models use the donor-format [INST] interface. Prompt formats, positional settings, and generation budgets for these models and their Base baselines are specified in App. B.1.

DiffusionGemma / Gemma 4. We map the AR updates to the corresponding DiffusionGemma language-model projections. The diffusion encoder and decoder share their base weights, so each AR update is applied once to the shared parameter and affects both branches. The diffusion fine-tuning adapters, which hold separate encoder and decoder updates, stay unmerged and are scaled according to the experimental formula. Diffusion-specific adapter updates such as the self-conditioning projections have no AR counterpart. The vision encoder, embeddings, normalization

parameters, and output head are unchanged, and all merged models keep the diffusion recipient's tokenizer and chat template.

B.3 Experimental details for section 2

Training. For mathematical reasoning we fine-tune Qwen2.5-7B and Dream-Base-7B on 100K NuminaMath-CoT (LI et al., 2024) examples; for coding we fine-tune Qwen2.5-Coder-7B and Dream-Coder-Base-7B on 100K EpiCoder-func (Wang et al., 2025b) examples. Training uses AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.95$, weight decay 0.01), learning rates of 5×10^{-6} for the AR models and 2×10^{-6} for Dream, cosine decay with 10% warmup, batch size 256, and one epoch (390 updates), with sequences right-truncated to 2,048 tokens. Prompts use each model's single-user chat template and the loss covers only the response tokens. AR models use causal next-token cross-entropy. Dream uses the bidirectional masked-denoising loss with CART time reweighting and a random per-batch sequence cutoff.

Evaluation. We evaluate 0-shot on GSM8K, MATH-500, HumanEval, and MBPP, reporting answer accuracy for math and pass@1 for code, including the extended EvalPlus tests. Dream uses the same chat template as in training, with no extra system instructions or assistant prefill. Because fine-tuning lengthens responses, the budget is 512 tokens for GSM8K and coding and 1,024 for MATH-500. Dream uses as many diffusion steps as generated tokens, entropy-based unmasking, temperature 0.1, and top-p 0.9. GSM8K additionally asks for concise reasoning and a final #### `<number>` answer, MATH-500 uses no extra instruction. Math sweeps use 200 examples per benchmark unless stated otherwise, and coding evaluations use all 164 HumanEval and 378 MBPP problems.

B.4 GPU and training time

All evaluations run on a single H100, H200, or B300 node. Because these pools do not give identical numbers, every run in a hyperparameter sweep uses the same hardware, and so do directly compared variants (base and transfer, post-trained and merged) on each benchmark. The four fine-tuning runs of section 2 take 30.8 GPU-hours in total, 6.1–9.2 GPU-hours each on four H100/H200 GPUs.

B.5 Hyperparameter search

For each family and construction, we fix an anchor benchmark (Table 7). For example, for instruction following the anchor is IFEval. The coefficient is searched on the anchor only and then kept fixed on every other benchmark, so the main table reports the anchor alongside tasks that played no role in the selection. For DiffuCoder, we use HumanEval+ as the anchor for instruction tuning, consistent with the other coding models, and MBPP+ for RL because we do not observe a gain from the instruction-tuned model to the RL model on HumanEval+ in our setup.

Table 6: **Coefficient selection for each transfer family.** Each coefficient is selected on its anchor benchmark and kept fixed on all other benchmarks. For the Nemotron Reasoning-donor row, the anchors are listed in transfer / merge order: IFEval without thinking and MATH-500 with thinking, respectively. Full sweeps are reported in Tables 8 and 7.

AR → Diffusion	vector type	anchor	transfer α	merge λ
Qwen2.5-7B → Dream-7B	instruct (SFT)	IFEval	0.9	0.6
Qwen2.5-Coder → Dream-Coder	code (SFT)	HumanEval+	0.6	0.2
Qwen2.5-Coder → DiffuCoder	code (SFT)	HumanEval+	0.1	0.2
AceCoder-Rule → DiffuCoder	code (RL)	MBPP+	0.2	0.5
Qwen3-8B → DreamReasoner	reasoning	MATH-500	0.5	0.2
Ministral-3-3B → Nemotron-LD-3B	instruct (SFT)	IFEval	0.9	0.3
Ministral-3-8B → Nemotron-LD-8B	instruct (SFT)	IFEval	1.4	0.4
Ministral-3-8B-Reasoning → Nemotron-LD-8B	reasoning	IFEval / MATH-500	0.8	0.2
Gemma 4 → DiffusionGemma	SLAKE (SFT)	SLAKE	1.0	0.1
Gemma 4 → DiffusionGemma	VQA-RAD (SFT)	VQA-RAD	0.7	0.3

Full sweeps for the main results and the key ablations are in Table 7, 9, 10 and 8.

Table 7: **Merge**, $D_b + (1-\lambda)\tau_D + \lambda\tau_A$, swept on each family's anchor benchmark. For DiffusionGemma, D_b denotes the instruction-tuned reference checkpoint. Shaded = selected λ, **bold** = best score. All results are in percentages.

AR → Diffusion	Anchor	λ									
		0 (parent)	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	1.0
Qwen2.5-7B → Dream-7B	IFEval	63.0	65.0	67.6	—	69.8	70.3	**72.8**	72.6	71.5	56.7
Qwen2.5-Coder → Dream-Coder	HumanEval+	77.4	76.2	**79.3**	76.2	77.4	—	—	—	—	—
Qwen2.5-Coder → DiffuCoder	HumanEval+	65.9	70.7	**74.4**	72.6	71.3	—	—	—	—	—
AceCoder-Rule → DiffuCoder (RL)	MBPP+	66.9	—	66.4	**69.0**	68.3	**69.0**	66.9	—	—	—
Qwen3-8B → DreamReasoner	MATH-500	93.6	93.2	**94.8**	94.0	94.2	—	—	—	—	—
Ministral-3-3B-Instruct → Nemotron-LD-3B	IFEval	68.0	—	71.5	**72.9**	71.9	72.5	69.1	—	51.2	—
Ministral-3-8B-Instruct → Nemotron-LD-8B	IFEval	69.8	—	74.4	74.1	**78.8**	76.2	—	—	—	—
Ministral-3-8B-Reasoning → Nemotron-LD-8B	MATH-500	87.2	90.8	**92.6**	90.4	90.2	—	39.4	—	11.4	—
Gemma 4 → DiffusionGemma	SLAKE	83.1	**83.3**	82.9	—	82.3	—	81.8	—	81.2	77.1
Gemma 4 → DiffusionGemma	VQA-RAD	71.6	72.7	73.8	**74.5**	73.6	—	68.7	—	71.8	70.5

Table 8: **Transfer**, $D_b + \lambda\tau_A$: the AR vector on the diffusion base. For DiffusionGemma, D_b denotes the instruction-tuned reference checkpoint. Shading and bold as above; λ=0 is the base. All results are in percentages.

AR → Diffusion	Anchor	α													
		0 (base)	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0	1.1	1.3	1.4	1.5
Qwen2.5-7B → Dream-7B	IFEval	50.5	—	54.1	—	—	58.4	58.7	59.1	58.4	**59.3**	56.7	—	—	—	—
Qwen2.5-Coder → Dream-Coder	HumanEval+	57.9	—	—	—	65.9	65.9	**67.1**	61.0	59.1	—	—	—	—	—	—
Qwen2.5-Coder → DiffuCoder	HumanEval+	61.6	**60.4**	57.3	—	47.6	—	—	—	—	—	—	—	—	—	—
AceCoder-Rule → DiffuCoder (RL)	MBPP+	65.9	66.1	**66.9**	65.3	65.3	—	—	—	—	—	—	—	—	—	—
Qwen3-8B → DreamReasoner	MATH-500	—	—	64.8	66.6	69.4	**70.4**	67.4	—	—	—	—	—	—	—	—
Ministral-3-3B-Instruct → Nemotron-LD-3B	IFEval	25.2	—	—	—	—	—	—	53.4	56.2	**59.6**	58.1	—	—	—	—
Ministral-3-8B-Instruct → Nemotron-LD-8B	IFEval	28.2	—	—	—	—	—	—	—	—	59.0	61.6	61.0	62.2	**64.9**	63.2
Ministral-3-8B-Reasoning → Nemotron-LD-8B	IFEval	28.2	—	—	—	53.5	—	55.1	56.9	**57.6**	57.6	56.8	—	—	—	—
Gemma 4 → DiffusionGemma	SLAKE	63.2	—	68.0	—	71.6	—	74.9	—	76.8	76.9	**77.1**	75.5	—	—	—
Gemma 4 → DiffusionGemma	VQA-RAD	57.4	—	65.0	—	68.5	72.9	72.3	**74.1**	71.8	—	70.5	—	—	—	—

Table 9: $D_p + \alpha\tau_A$: the AR vector added to the post-trained diffusion model. Shading and bold as above. All results are in percentages.

AR → Diffusion	Anchor	α												
		0 (parent)	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0	1.1	1.2
Qwen2.5-7B → Dream-7B	IFEval	63.0	—	67.7	—	70.9	—	71.0	73.3	73.5	74.5	74.9	**77.1**	75.5
Qwen2.5-Coder → Dream-Coder	HumanEval+	77.4	—	75.0	76.8	**77.4**	**77.4**	75.6	—	—	—	—	—	—
Qwen2.5-Coder → DiffuCoder	HumanEval+	65.9	—	66.5	—	68.9	**69.5**	67.7	60.4	—	—	—	—	—
AceCoder-Rule → DiffuCoder (RL)	MBPP+	66.9	—	—	—	66.9	66.9	66.4	—	66.9	—	—	—	—
Qwen3-8B → DreamReasoner	MATH-500	89.2	89.8	90.4	**91.4**	90.2	—	—	—	—	—	—	—	—
Ministral-3-3B-Instruct → Nemotron-LD-3B	IFEval	68.0	68.1	69.1	69.2	**71.0**	70.0	69.8	69.4	66.5	64.6	—	—	—
Ministral-3-8B-Instruct → Nemotron-LD-8B	IFEval	69.8	69.6	71.3	73.5	74.3	**75.4**	72.9	73.2	73.6	71.4	—	—	—
Ministral-3-8B-Reasoning → Nemotron-LD-8B	MATH-500	87.2	88.4	90.8	**91.6**	89.2	—	79.6	—	35.4	—	—	—	—
Gemma 4 → DiffusionGemma	SLAKE	83.1	**83.4**	82.8	—	82.4	—	81.6	—	79.3	—	77.9	—	—
Gemma 4 → DiffusionGemma	VQA-RAD	71.6	—	73.4	73.4	**74.7**	74.3	73.2	—	74.3	—	71.4	—	—

Efficiency of hyperparameter search Evaluating the transfer hyperparameter sweeps for the Nemotron-LD variants in Table 8 on all 541 IFEval prompts with the full 8,192-token generation budget was too expensive, so we used a 1,280-token budget while retaining the full dataset for coefficient selection. This choice was motivated by a separate budget comparison on the Nemotron-LD-8B protected merge (Table 11): the shorter budget preserved the ranking and selected coefficient as well as median output lengths across the tested points.

Table 10: $D_b + \alpha\tau_D$: the model's own post-training vector rescaled, no AR term. Shading and bold as above. All results are in percentages.

Diffusion (no AR term)	Anchor	α									
		0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0 (parent)
Qwen2.5-7B → Dream-7B	IFEval	—	—	—	—	—	61.5	—	61.3	**63.4**	63.0
Qwen2.5-Coder → Dream-Coder	HumanEval+	—	—	—	—	—	—	—	72.0	**78.7**	77.4
Qwen2.5-Coder → DiffuCoder	HumanEval+	—	—	—	68.3	—	**70.1**	65.9	66.5	65.9	65.9
AceCoder-Rule → DiffuCoder (RL)	MBPP+	—	65.6	—	66.7	69.3	68.8	**69.6**	67.5	—	66.9
Qwen3-8B → DreamReasoner	MATH-500	—	—	—	—	—	—	—	89.2	**90.2**	89.2
Ministral-3-3B-Instruct → Nemotron-LD-3B	IFEval	28.4	49.7	60.9	67.3	70.1	70.3	**72.0**	71.3	70.5	68.0
Ministral-3-8B-Instruct → Nemotron-LD-8B	IFEval	20.8	21.4	24.2	56.4	70.1	**71.7**	70.8	70.5	71.5	69.8
Ministral-3-8B-Reasoning → Nemotron-LD-8B	MATH-500	—	0.0	—	41.4	—	83.6	87.4	90.2	**90.8**	87.2
Gemma 4 → DiffusionGemma	SLAKE	—	68.7	—	75.6	—	80.3	82.6	**84.0**	83.4	83.1
Gemma 4 → DiffusionGemma	VQA-RAD	—	66.7	—	69.0	—	71.2	—	71.4	**72.5**	71.6

Table 11: **Short-budget IFEval comparison on the 8B `A2D` merge.** Both budgets evaluate the same 541 prompts with thinking off and select $\lambda = 0.4$. Scores are percentages; median lengths are generated token counts. Bold marks the best score per budget.

	1,280-token budget		8,192-token budget	
λ	Score	Median length	Score	Median length
0.3	74.03	314	74.07	308
0.4	**78.00**	332	**78.76**	336
0.5	75.59	323	76.20	315

C Additional Results

In this section, we report additional experimental results.

C.1 Full results for analyses and ablations

Here we put the full results for the analyses and ablations conducted in section 4.

- Table 12 presents the full results for the decomposition of the contributions by adding the AR task vector and downscaling the diffusion task vector.
- Table 13 presents the full results for transfering the diffusion vector back to the AR base model.
- Figure 5 (a) compares the "AR-ness" between the diffusion parent model and the A2D merged model, showing that A2D does not shift the model systematically toward autoregressive generation.
- Figure 5 (b) shows that A2D mainly changes the diffusion parent's token prediction at positions where the diffusion parent has a low confidence.
- Table 14 presents the module-wise ablation of the transferred AR update. For `A2D`-Transfer, removing the MLP component causes a substantial performance drop, while removing attention has little effect. For `A2D`-Merge, reverting the MLP modules to the diffusion parent nearly preserves full performance, whereas reverting attention causes a larger drop, suggesting that MLPs mainly carry standalone AR transfer while attention provides more complementary gains during merging.
- Table 15 shows that continued-pretraining vectors do not transfer effectively, suggesting that `A2D` is most suitable for relatively local post-training updates rather than pretraining-scale model transformations.

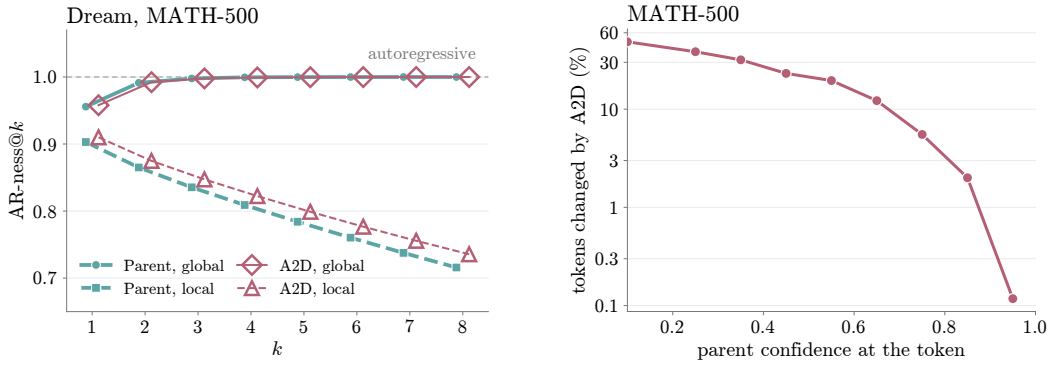


Figure 5: **How A2D alters diffusion generation. Left:** A2D largely preserves the parent's decoding order. **Right:** Prediction changes concentrate on low-confidence tokens.

Table 12: **Sources of the A2D gain.** Best-performing point for each construction, with coefficients selected on the anchor task. Avg. rank is the mean within-pair rank (1 = best) and Avg. Acc. is the mean over the seven pairs. All results are in percentages.

	Dream IFEval	DreamCoder HumanEval+	DiffuCoder HumanEval+	DiffuCoder-RL MBPP+	DreamReasoner MATH-500	Nemotron-3B IFEval	DiffusionGemma VQA-RAD	Avg. rank↓	Avg. Acc.
Diffusion parent D_p	63.0	77.4	65.9	66.9	93.6	68.0	71.6	3.7	72.3
$+\tau_A$ only $D_p + \lambda\tau_A$	**77.1**	77.4	69.5	66.9	94.2	71.0	**74.7**	2.4	75.8
$-\tau_D$ only $D_p - \lambda\tau_D$	63.4	78.7	70.1	**69.6**	92.4	72.0	72.5	2.4	74.1
`A2D` merge $D_p + \lambda(\tau_A - \tau_D)$	72.8	**79.3**	**74.4**	69.0	**94.8**	**72.9**	74.5	**1.4**	**76.8**

C.2 Full results of Dream-7B

Here, we report the results used in Figure 2a and in Figure 2b in the Table 16.

Table 13: **Reverse direction: diffusion post-training vector into the AR model.** All results are in percentages.

	Dream-7B←Qwen2.5-7B				DiffuCoder→Qwen2.5-Coder	
	IFEval	IFBench	GSM8K	MATH-500	HumanEval+	MBPP+
Diffusion donor	63.0	19.0	83.0	38.4	65.9	65.9
AR base	40.5	16.3	**62.5**	**17.4**	74.4	62.2
A2D (transfer, α=1.0 / 0.1)	**46.7**	**19.0**	56.0	16.2	74.4	**64.8**
AR instruct	75.3	**31.7**	**83.0**	56.0	86.6	71.7
A2D (merge, λ=0.1 / 0.2)	**77.3**	30.7	82.9	**56.4**	86.6	71.7

Table 14: **Module-wise ablation of the transferred AR update.** MATH-500 accuracy on Dream-7B. For A2D-Merge, the ablated module is reverted to the diffusion parent D_p, while all remaining modules use the merged weights. All results are in percentages.

	A2D-Transfer	A2D-Merge
Full model	**40.8**	**45.8**
w/o AR attention	40.8	43.0
w/o AR MLP	30.4	45.6

C.3 Full results of Nemotron-LD-3B

Table 17 reports the full Nemotron-LD-3B results under both diffusion and autoregressive decoding. For the diffusion Base, we evaluate two prompting protocols. The main paper reports the native raw-prompt result, which provides the stronger overall Base performance. For completeness, we additionally evaluate the Base using the same instruction prompt format as A2D-Transfer, providing a prompt-matched comparison.

Under either protocol, A2D-Transfer substantially strengthens the Base checkpoint, bringing it close to or beyond the Instruct model on several benchmarks. A2D-Merge then provides further improvements over the Instruct checkpoint under both diffusion and autoregressive decoding.

C.4 Results of Nemotron-LD-8B

We provide in the main paper the full results of Nemotron-LD-3B as official results. Here, we provide the full results for Nemotron-LD-8B in Table 18. At the 8B scale, we evaluate A2D with both instruction-tuning and reasoning-oriented AR updates.

Instruction-tuning updates. With the Ministral-3-8B-Instruct donor, A2D-Transfer broadly improves the diffusion base, while A2D-Merge provides gains on several benchmarks under both diffusion and AR decoding. The merging gains are less consistent than for the 3B model. Importantly, this setting exhibits a mismatch in generation behavior: Nemotron-LD-8B produces long, reasoning-like responses on reasoning-intensive tasks even with thinking disabled, whereas merging the Instruct update substantially shortens these generations (Figure 6). Thus, comparisons on these tasks are partially confounded by changes in effective reasoning behavior in addition to the weight-space composition itself.

Reasoning updates. Separately, we evaluate A2D using the Ministral-3-8B-Reasoning donor, with thinking enabled for both the diffusion parent and the merged model. Unlike the instruction-tuning setting above, both models use the same explicit thinking mode, avoiding the generation-behavior mismatch. Under the official generation budget, A2D-Merge improves the mean score from 59.9 to 64.3, with substantial gains on reasoning-related benchmarks such as MATH-500 (87.2 → 92.6), GPQA (36.4 → 44.4), AIME24 (33.3 → 46.7), and LiveCodeBench (26.9 → 38.1). The gains remain strong when doubling the generation budget, where the mean further increases from 60.4 to 65.6. These results show that reasoning-oriented AR post-training updates can be effectively composed with diffusion models at the 8B scale under matched reasoning behavior.

Table 15: **Continued pre-training vectors do not transfer.** Adding the AR continued-pretraining vector to the diffusion base does not improve performance, even at small coefficients. These vectors are 70–82× larger than the post-training vectors used in our main experiments. **Bold** denotes the best $\alpha>0$ result; all results are percentages.

	Recipient	Benchmark	α in $D_b + \alpha\tau_{\text{CPT}}$			
			0	0.01	0.05	1.0
Math (82×), Qwen2.5-Math − Qwen2.5	Dream-7B-Base	GSM8K	76.6	**76.6**	74.8	0.0
Code (70×), Qwen2.5-Coder − Qwen2.5	Dream-Coder-Base	HumanEval	63.4	**61.6**	57.3	0.0

Table 16: **Dream-7B ← Qwen2.5-7B on six tasks.** One transfer model (α=0.9) and one merged model (λ=0.6), both selected on IFEval, improve every task over the corresponding baseline. All results are in percentages.

	Dream-7B ← Qwen2.5-7B					
	IFEval	IFBench	GSM8K	MATH-500	HumanEval	MBPP
AR donor	75.3	31.7	83.0	56.0	83.5	81.2
Base	50.1	14.0	54.7	20.8	51.8	57.4
A2D (transfer)	**59.3**	**25.0**	**81.7**	**40.2**	**59.8**	**59.4**
Instruct	63.0	19.0	83.0	38.4	56.1	58.2
A2D (merge)	**72.8**	**22.3**	**83.7**	**45.0**	**64.0**	**61.8**

C.5 DiffusionGemma with SLAKE Post-training

In the main paper, we report results for merging with DiffusionGemma using checkpoints fine-tuned on VQA-RAD. In Table 19, we additionally report the same setup using checkpoints fine-tuned on SLAKE. Unlike on VQA-Med, where the AR and diffusion models achieve comparable performance, on SLAKE the AR model is substantially weaker than its diffusion counterpart, providing a complementary setting where the donor itself is less capable than the recipient.

Despite this gap, the AR SLAKE task vector transfers effectively to the diffusion base and further improves the diffusion post-trained model across all three benchmarks when transferred or merged. This shows that AR updates can remain complementary even when the donor itself is weaker than the recipient.

C.6 Self-speculative decoding of Nemotron

Table 20 evaluates whether the benefits of A2D extend to self-speculative decoding. We compare each merged checkpoint with its corresponding Instruct parent under matched decoding settings, reporting accuracy, tokens per forward evaluation (TPF), and generation time per token. We use NVIDIA's native linear self-speculative decoding, with diffusion drafting and autoregressive verification, a block size of 32, threshold 0, and no LoRA adapter. All runs use greedy decoding, with the same generation and thinking budgets specified in Appendix B.1.

Across the evaluated configurations, A2D consistently increases TPF and generally improves accuracy. The non-thinking merges improve both accuracy and observed time per token on both benchmarks and at both model sizes. These benefits also extend to MATH-500 with the reasoning donor and thinking enabled.

C.7 Post-training types and norm ratios

As shown in Table 21, we concatenate parameter differences over backbone tensors (attention, MLP, and normalization layers) and compute their ℓ_2 norms and cosine similarities. The Nemotron measurements additionally include compatible input-embedding rows after key mapping, excluding protected rows and unmatched parameters. All measurements exclude output heads, the other

Table 17: **Ministral-3-3B-Instruct → Nemotron-LD-3B.** DLM and AR denote diffusion and autoregressive decoding of the same checkpoint. Mean excludes IFEval. **Bold** marks the better result within each Instruct-merge pair. All results are in percentages.

	Mode	GSM8K	MATH-500	HumanEval	MBPP	IFEval	MMLU	GPQA	AIME24	AIME25	LCB-CPP	Mean
AR donor	AR	91.3	83.0	80.5	75.9	65.1	67.9	40.9	30.0	13.3	27.1	56.7
Base (raw protocal)	DLM	59.1	40.0	64.6	70.9	23.2	62.0	**35.4**	0.0	0.0	9.9	38.0
Base (Instruct protocol)	DLM	78.0	34.6	39.6	38.6	25.8	55.1	26.3	0.0	0.0	4.0	30.7
A2D (transfer)	DLM	**88.1**	**52.6**	**72.6**	**74.3**	**60.6**	**62.8**	33.3	**3.3**	**3.3**	**14.1**	**44.9**
Base (raw protocal)	AR	52.1	50.4	71.3	72.8	21.1	58.0	32.8	**10.0**	**13.3**	**14.5**	41.7
A2D (transfer)	AR	**89.2**	**60.4**	**73.8**	**74.6**	**59.0**	**62.6**	**33.3**	6.7	0.0	**14.5**	**46.1**
Instruct	DLM	89.2	76.8	73.2	73.0	68.0	72.1	35.4	10.0	**13.3**	16.3	51.0
A2D (merge)	DLM	**90.1**	**79.6**	**75.0**	**75.7**	**72.9**	**72.9**	**43.4**	**16.7**	**13.3**	**18.9**	**54.0**
Instruct	AR	87.6	76.0	73.8	69.8	68.8	71.9	**41.9**	**23.3**	**13.3**	17.6	52.8
A2D (merge)	AR	**89.3**	**81.0**	**79.9**	**75.1**	**72.8**	**72.5**	38.4	20.0	**13.3**	**18.7**	**54.3**

Table 18: **Ministral-3-8B → Nemotron-LD-8B.** DLM and AR denote diffusion and autoregressive decoding of the same checkpoint. Mean excludes IFEval and, when unavailable, MMLU, which under some settings exhibits repetitive generation and requires over four days to complete. **Bold** marks the better result within each corresponding pair. All results are in percentages.

	Mode	GSM8K	MATH-500	HumanEval	MBPP	IFEval	MMLU	GPQA	AIME24	AIME25	LCB-CPP	Mean
AR donor	AR	92.4	83.6	71.0	79.0	64.3	73.9	42.9	27.7	24.6	20.8	57.3
Base (raw protocal)	DLM	64.7	51.2	**77.4**	**82.0**	28.2	38.4	34.3	13.3	**20.0**	**17.0**	44.3
Base (Instruct protocol)	DLM	78.2	59.8	4.9	13.2	21.5	—	9.1	6.7	10.0	2.2	*23.0*
A2D (transfer)	DLM	**89.5**	**71.6**	75.0	81.7	**65.7**	72.4	**37.4**	13.3	6.7	11.2	**51.0**
Instruct	DLM	93.7	**89.2**	**73.8**	62.2	69.8	**79.1**	**44.4**	30.0	**30.0**	**29.3**	**59.1**
A2D (merge)	DLM	**94.5**	89.0	71.3	**65.3**	**78.8**	78.6	42.9	**33.3**	26.7	28.0	58.8
Instruct	AR	93.5	88.6	78.0	65.1	70.8	79.8	**48.0**	**33.3**	26.7	**35.5**	**60.9**
A2D (merge)	AR	**93.9**	**89.4**	**79.3**	**72.2**	**74.9**	78.8	47.0	30.0	26.7	31.1	**60.9**
thinking, official budget (8,192 total / 6,000 thinking), reasoning donor												
Instruct	DLM	92.4	87.2	**86.0**	**87.3**	**75.5**	—	36.4	33.3	30.0	26.9	59.9
A2D (merge)	DLM	**94.1**	**92.6**	79.3	86.2	74.7	—	**44.4**	**46.7**	**33.3**	**38.1**	**64.3**
thinking, double budget (16,384 total / 12,000 thinking), reasoning donor												
Nemotron-LD-8B-Instruct	DLM	92.2	87.0	**85.4**	**87.3**	**75.6**	—	40.4	33.3	30.0	27.3	60.4
A2D (merge)	DLM	**94.0**	**92.2**	79.3	86.5	74.6	—	**46.0**	**53.3**	**33.3**	**40.5**	**65.6**

model families also exclude all input embeddings. We define $\Delta_b = \boldsymbol{D}_b - \boldsymbol{A}_b$, $\Delta_p = \boldsymbol{D}_p - \boldsymbol{A}_p$, $\boldsymbol{\tau}_A = \boldsymbol{A}_p - \boldsymbol{A}_b$, and $\boldsymbol{\tau}_D = \boldsymbol{D}_p - \boldsymbol{D}_b$.

For most SFT pairs, the AR-to-diffusion conversion direction remains highly stable after post-training, with $\cos(\Delta_b, \Delta_p) \geq 0.98$, while the AR and diffusion post-training vectors have low cosine similarity, indicating largely distinct updates. DreamReasoner shows a larger change in the conversion direction and also has a substantially larger AR post-training update.

The update scale is also markedly different between post-training and continued pre-training. Post-training vectors have relatively small norms, whereas continued-pretraining vectors are substantially larger, potentially moving the resulting checkpoints beyond the local low-loss region in which weight-space composition is most effective. In addition, the code continued-pretraining vectors are strongly aligned across the AR and diffusion paradigms, in contrast to the near-orthogonality observed for post-training vectors.

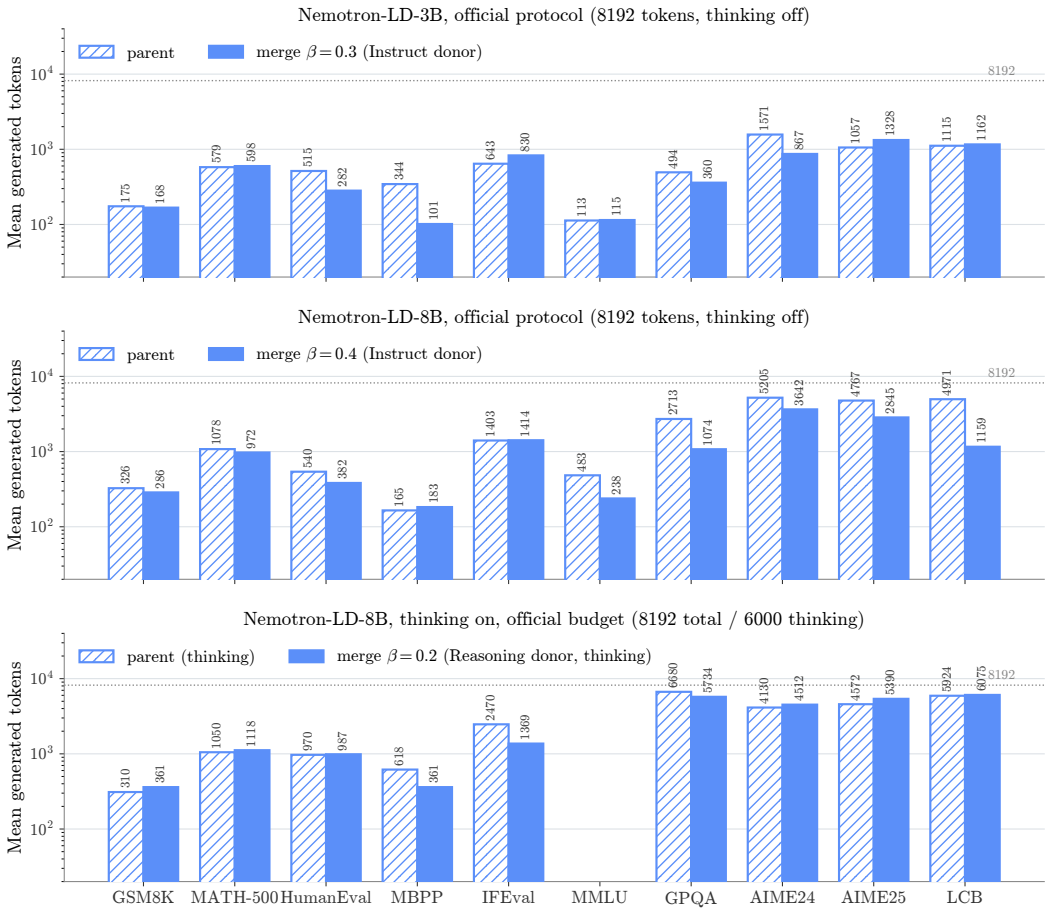


Figure 6: **Generation-length mismatch in Nemotron-LD.** The 8B model, unlike the 3B one, exhibits longer, reasoning-like generations even with thinking disabled. Enabling thinking substantially reduces the length mismatch between the parent and merged models.

Table 19: **AR SLAKE post-training transfer from Gemma to DiffusionGemma.** AR donor denotes the AR checkpoint providing the task vector. Base denotes the instruction-tuned diffusion checkpoint. Anchor task used for hyperparameter selection is highlighted. **Bold** marks the best result compared to the corresponding baseline. All results are in percentages.

	Gemma 4→ DiffusionGemma		
	VQA-RAD	SLAKE	VQA-Med
AR donor	58.09	80.68	37.20
Base	**57.43**	63.24	**43.20**
A2D (transfer)	56.54	**77.10**	37.20
Post-trained	60.75	83.13	39.60
A2D (merge)	**61.64**	**83.32**	**40.00**

Table 20: **Accuracy and efficiency of self-speculative decoding.** TPF is $\sum_i T_i / \sum_i \mathrm{NFE}_i$, including thinking tokens. Relative time is normalized generation time per token within each dataset, model size, and thinking setting; lower is better. Timings are observational and exclude model loading and final grading. Accuracy results are percentages (MATH-500 accuracy; MBPP base-test pass@1), while TPF and relative time are ratios. Results are in percentages.

	MATH-500			MBPP		
Model	Acc. ↑	TPF ↑	Rel. time ↓	Acc. ↑	TPF ↑	Rel. time ↓
Ministral-3-3B-Instruct → Nemotron-LD-3B, thinking off						
Instruct parent	76.6	4.96	1.00	69.8	3.71	1.00
A2D (merge)	**80.8**	**5.04**	**0.87**	**74.9**	**3.90**	**0.80**
Ministral-3-8B-Instruct → Nemotron-LD-8B, thinking off						
Instruct parent	87.0	5.34	1.00	68.0	3.42	1.00
A2D (merge)	**88.6**	**6.07**	**0.88**	**73.0**	**3.55**	**0.77**
Ministral-3-8B-Reasoning → Nemotron-LD-8B, thinking on						
Instruct parent	87.6	3.70	1.00	**87.8**	3.77	**1.00**
A2D (merge)	**91.4**	**4.04**	**0.81**	84.4	**3.84**	1.53

Table 21: **Geometry of post-training and paradigm-conversion directions across model pairs.** Measurements use backbone parameters and, for Nemotron, compatible input embeddings after key mapping, excluding protected rows. Output heads are excluded throughout.

diffusion ← AR	post-training	$\|\tau_A\|$	$\|\tau_D\|$	$\|\Delta_b\|$	$\|\Delta_p\|$	$\cos(\Delta_b, \Delta_p)$	$\cos(\tau_A, \tau_D)$
Qwen2.5-7B → Dream-7B	instruct (SFT)	23.0	31.5	729	732	+0.999	+0.013
Qwen2.5-Coder→ Dream-Coder	code (SFT)	87.0	84.6	589	602	+0.980	+0.023
Qwen3-8B → DreamReasoner	reasoning (SFT / SFT+RL)	189.2	295.8	459	556	+0.803	+0.118
Ministral-3 → Nemotron-LD-3B	instruct (SFT), cross-arch.	43.1	17.1	280.9	271.4	+0.987	+0.075
Ministral-3 → Nemotron-LD-8B	instruct (SFT), cross-arch.	42.2	16.3	387.0	380.4	+0.994	+0.070
Qwen2.5-Coder → DiffuCoder	code (SFT)	87.0	33.3	627	630	+0.989	+0.005
AceCoder-Rule → DiffuCoder	code (RL)	0.68	4.10	630	630	+1.000	+0.004
Continued pre-training (knowledge) vectors, for contrast		$\|v_A\|$	$\|v_D\|$				$\cos(v_A, v_D)$
code knowledge	continued pre-training	1603.6	1460.9	—	—	—	+0.869
math knowledge	continued pre-training	1876.9	—	—	—	—	—

D ANALYSIS

D.1 CORRECTION AND ERROR INTRODUCTION ALONG THE A2D DIRECTION

We further characterize how A2D changes model predictions by tracking beneficial and adverse prediction transitions as the AR contribution increases.

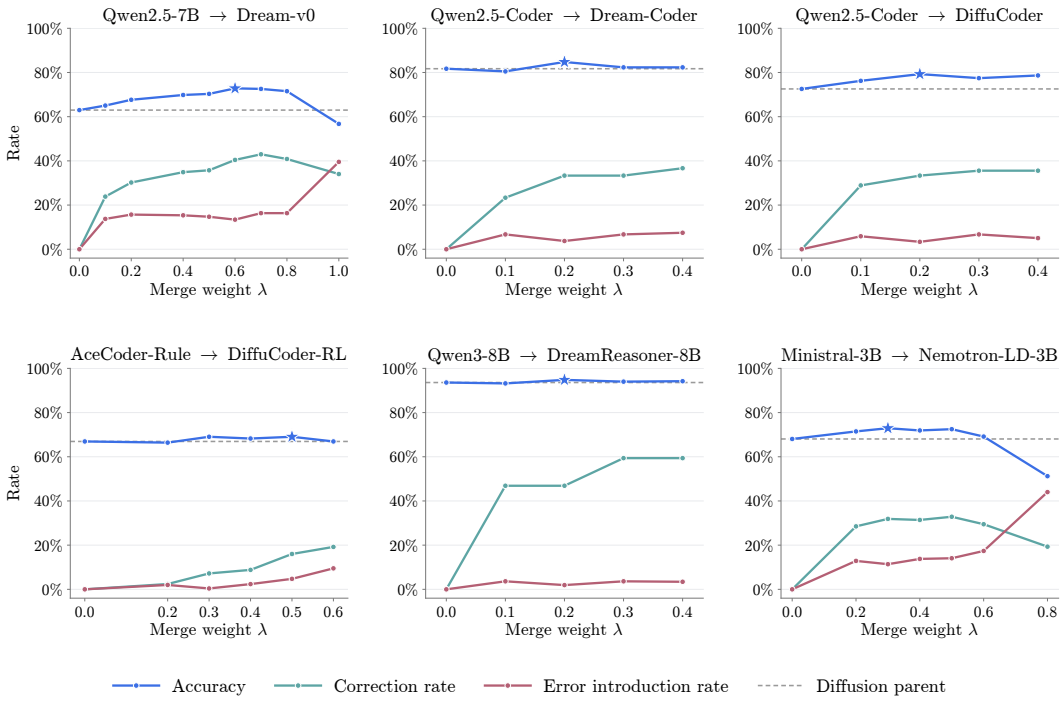


Figure 7: **Correction increases with the AR contribution while error introduction remains limited over the effective interpolation range.** We report accuracy, correction rate (parent wrong → A2D correct), and error introduction rate (parent correct → A2D wrong) along the A2D interpolation path. Dashed lines denote the diffusion parent, and stars mark the best-performing coefficient.

Across the six settings in Figure 7, increasing the AR contribution generally increases correction while introducing few new errors around the optimal region. Some models remain robust over a broad range of λ, while Dream-7B and Nemotron-LD show a clearer trade-off at larger values. Overall, A2D improves performance by correcting parent errors while largely preserving already-correct predictions.

D.2 INTERPOLATION PRESERVES COMPLEMENTARY GAINS ON RELEASED CHECKPOINTS

As shown in Figure 8, the same pattern holds for released checkpoints: the diffusion parent and AR donor solve different uqestions, while the merged model preserves most successes from both.

D.3 ALIGNED REPRESENTATION CHANGES ON RELEASED CHECKPOINTS

The same pattern extends to released checkpoints. As shown in Figure 9, for both Dream-Coder and DiffuCoder-RL, AR and diffusion post-training updates remain nearly orthogonal in parameter space while inducing substantially more aligned changes in the diffusion model's representations. Notably, this pattern also holds for the RL-derived updates of DiffuCoder-RL.



Figure 8: **A2D keeps the successes of both endpoints.** For every item of the benchmark, the inner ring records which of the diffusion parent and the AR donor solves it. The outer ring records whether the merged model solves it.

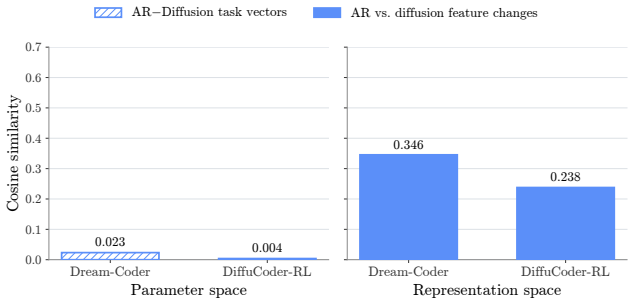


Figure 9: Distinct parameter updates induce aligned representation changes for released checkpoints.

E LIMITATIONS AND FUTURE WORK

Our results reveal a weight-space connection between AR and diffusion models: AR post-training updates remain meaningful after transfer to diffusion models and can complement diffusion-specific updates. This extends existing connections between the two paradigms beyond training and decoding mechanisms. However, several questions remain open.

A2D currently relies on a compatible training lineage between the AR donor and diffusion recipient. All model pairs studied here originate from the same AR family, although our Nemotron experiments show that exact architectural correspondence is not required. Whether cross-paradigm transfer extends to more distant or independently trained model pairs remains unclear.

Moreover, the transferability we observe is primarily limited to relatively local post-training updates. Continued-pretraining updates are substantially larger and do not transfer effectively in our experiments, suggesting that pretraining-scale transformations may require additional alignment or decomposition.

Future work could further characterize what determines cross-paradigm compatibility, develop more adaptive composition strategies, and study A2D as an initialization for subsequent SFT or RL.