MASK-ON: COMPACT ACTIVATION-CALIBRATED MERGING FOR AUTOREGRESSIVE AND DIFFUSION LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Autoregressive (AR) and diffusion language models offer complementary generation behaviors, but retaining both pretrained endpoints requires storing two full parameter sets. We propose MASK-ON, a gradient-free method that represents both modes using an unchanged AR backbone and a compact AR-to-diffusion update. Building on support/sign encoding, MASK-ON compresses AR-to-diffusion weight changes into ternary masks and output-channel scales, calibrated by matching the diffusion endpoint's linear-layer outputs on collected activations. Closed-form scale fitting reduces the search to a single pruning parameter without gradient-based optimization or downstream task-specific tuning. Across four AR–diffusion model pairs and four benchmarks spanning diverse domains, we demonstrate that direct parameter merging does not consistently preserve both generation modes, whereas compact mode-specific updates support more consistent diffusion retention alongside an unchanged AR checkpoint. MASK-ON achieves the highest average diffusion-mode accuracy among the compared merging methods on all four models, with an update substantially smaller than a full diffusion checkpoint. Experiments on Fast dLLM v2 show a similar quality–throughput trend to the diffusion endpoint across six decoding configurations. A simple training-free experiment using the original AR checkpoint further improves diffusion-draft accuracy by up to 3.26 percentage points on GSM8K, illustrating how access to both modes can support complementary generation.

1 INTRODUCTION

Autoregressive (AR) generation remains the predominant paradigm for language models (Xu et al., 2026; Yamada et al., 2025; OpenAI, 2026; Zeng et al., 2026). By conditioning each token on the prefix, AR decoding explicitly incorporates preceding token predictions, but this sequential dependence limits token-level parallelism. Diffusion offers an alternative through parallel token refinement, predicting multiple positions per step to reduce the number of sequential decoding steps (Wu et al., 2026a; Cheng et al., 2026; Bie et al., 2025). However, more aggressive parallel generation can come at the cost of output quality, creating a trade-off between generation quality and speed (Liu et al., 2025; Wu et al., 2026a).

Practical systems may therefore benefit from retaining both generation modes, selecting between sequentially conditioned AR decoding and parallel diffusion refinement according to each request's requirements. The two modes can also cooperate: diffusion's parallelism can support efficient generation of candidate content, while AR's sequential conditioning can support contextual assessment and refinement (Christopher et al., 2025; Chen et al., 2026).

A straightforward approach is to retain both complete checkpoints to preserve access to their respective behaviors, but this incurs the storage overhead of maintaining two full parameter sets. Model merging offers efficient tools for combining checkpoint capabilities through shared parameters and task-specific representations (Ilharco et al., 2022; Yadav et al., 2023; Huang et al., 2024; Qi et al., 2025). This perspective is particularly relevant to AR-to-diffusion adaptation, which develops diffusion models by leveraging the pretrained capabilities of an AR model rather than training them entirely from scratch (Wu et al., 2026a; Cheng et al., 2026; Bie et al., 2025). For checkpoints with

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.

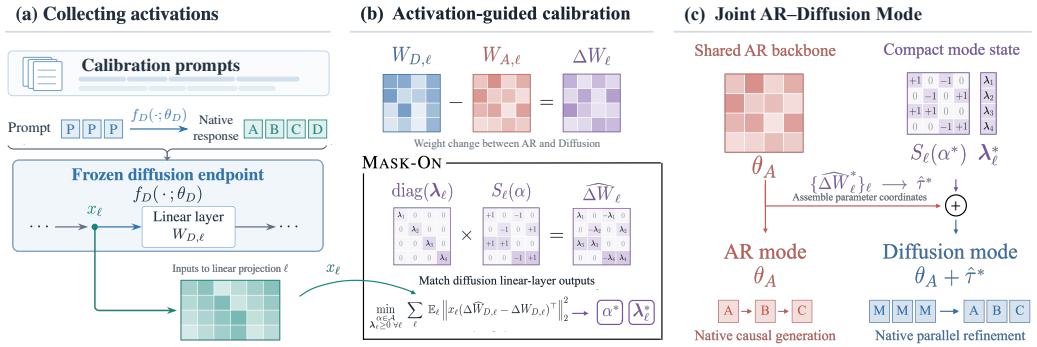

Figure 1: **Overview of MASK-ON.** (a) The diffusion endpoint generates responses to calibration prompts using its native inference procedure. Each sequence is then replayed in a single forward pass to collect calibration activations. (b) These activations are used to fit output-channel scales and sparsify the AR-to-diffusion weight update by minimizing reconstruction error. (c) The resulting compact update $\hat{\tau}$ enables switchable generation: AR uses the original checkpoint θ_A unchanged, while diffusion uses $\theta_A + \hat{\tau}$.

aligned parameters, this relationship motivates retaining the AR model as a shared backbone and storing the AR-to-diffusion weight changes as a compressed increment (Qi et al., 2025).

However, AR-to-diffusion adaptation changes the generation objective and conditioning patterns through substantial training, involving billions to over a hundred billion tokens (Wu et al., 2026a; Cheng et al., 2026; Liu et al., 2025). While existing merging methods combine adapted weights by exploiting compatibility or redundancy (Ilharco et al., 2022; Yang et al., 2024b; Davari & Belilovsky, 2024; Yu et al., 2024), combining AR and diffusion checkpoints into a single parameter state does not by itself ensure preservation of both generation modes. A mode-specific increment avoids changing the AR checkpoint, but introduces a separate challenge: compressing the adaptation can modify the computations needed for diffusion generation. Since the effect of a weight approximation depends on its input activations, weight-space approximation alone may not preserve these computations. The remaining question is how effectively a compact increment can retain diffusion performance while leaving the AR checkpoint unchanged.

To address this question, we propose MASK-ON, a gradient-free merging method that retains the AR checkpoint unchanged and compresses the AR-to-diffusion weight changes into a mode-specific update (Figure 1). Following T-Switch (Qi et al., 2025), we encode selected changes through their support and signs. Using activations from the diffusion endpoint as a functional reference (Heyrani Nobari et al., 2025; Sun et al., 2025), we fit output-channel scales by closed-form least squares and search only over pruning levels to minimize linear-layer output reconstruction error, without gradient-based optimization or downstream task-specific tuning. AR generation uses the original checkpoint, while diffusion generation uses the reconstructed weights under its native inference procedure.

Across four AR–diffusion model pairs and four benchmarks spanning diverse domains, direct parameter merging exhibits inconsistent performance, with strong results in one generation mode not necessarily indicating preservation of the other. Among the compared merging methods, MASK-ON achieves the highest average diffusion-mode accuracy on all four models while retaining the AR checkpoint unchanged, with additional checkpoint storage comparable in scale to T-Switch's. On Fast dLLM v2, MASK-ON also exhibits similar quality–throughput trends to the diffusion endpoint across six decoding configurations. A simple, training-free *draft-and-review* using the original AR checkpoint improves diffusion-draft accuracy and requires less generation time than reviewing the same drafts with diffusion. These results support compact storage of both generation modes and illustrate a use of their complementary behaviors.

Our contributions are threefold:

- We propose MASK-ON, which compresses AR-to-diffusion updates through closed-form scale fitting and pruning-level selection guided by linear-layer output matching on endpoint activations, without gradient-based optimization or downstream task-specific tuning.

- Across four AR–diffusion model pairs and four benchmarks spanning distinct domains, we demonstrate that direct parameter merging does not consistently preserve both generation modes, whereas compact mode-specific updates support more consistent diffusion retention alongside an unchanged AR checkpoint. MASK-ON achieves the highest average diffusion-mode accuracy among the compared merging methods on all four models, including 68.18% on LLaDA2.0-mini—19.88 points above the best parameter-combination baseline and 10.91 points above T-Switch—while adding only 14.71–25.48% storage relative to the AR checkpoint.
- We show that MASK-ON exhibits similar quality–throughput behavior to the diffusion endpoint across six decoding configurations on Fast dLLM v2. A simple, training-free *draft-and-review* procedure illustrates their complementary use, improving GSM8K accuracy over diffusion-only drafts with less generation time than diffusion review.

2 RELATED WORK

AR-to-Diffusion Adaptation Diffusion language models generate text through iterative denoising, refining multiple positions within a sequence or block in parallel (Wu et al., 2026a; Cheng et al., 2026; Bie et al., 2025). A practical development strategy initializes these models from pretrained AR checkpoints and adapts them to denoising objectives, leveraging their capabilities. Development strategies include complementary attention masks for blockwise bidirectional modeling (Wu et al., 2026a), autoregressive generation across diffusion blocks (Cheng et al., 2026), and staged block-size adaptation (Bie et al., 2025). Token reordering offers an alternative that supports parallel denoising while retaining standard causal attention (Liu et al., 2025).

AR–Diffusion Hybrid The complementary properties of AR and diffusion—sequential conditioning and parallel refinement—have motivated efforts to combine the two paradigms. One line of work uses diffusion to generate proposals rapidly in parallel, followed by AR-based assessment and refinement (Christopher et al., 2025; Chen et al., 2026). Shao et al. (2026) use diffusion to explore multiple candidate reasoning paths, with AR evaluating and selecting among them. A further direction jointly trains a single model to support both AR and diffusion generation (Liu et al., 2026; Fu et al., 2026). In contrast, we retain both modes from an existing AR–diffusion pair without jointly training a new hybrid model.

Model Merging Parameter-combination methods aggregate weight changes through Task Arithmetic (Ilharco et al., 2022), resolve sign conflicts with TIES (Yadav et al., 2023), or learn layer-wise coefficients (Yang et al., 2024b). Yu et al. (2024) randomly drops and rescales weight changes before applying a merging rule. Activation-based methods match linear-layer outputs (Jin et al., 2022), correct merged representations (Yang et al., 2024a), or protect activation-important base weights (Heyrani Nobari et al., 2025). For compact task-specific representations, Qi et al. (2025) encodes task vectors using support and sign masks with rescaling factors. Our setting differs from prior merging work in that the two checkpoints use different generation modes, and the goal is to preserve both AR and diffusion behavior.

3 METHODOLOGY

Building on these ideas, MASK-ON combines an unchanged AR checkpoint with a compact AR-to-diffusion update. The representation encodes selected weight changes through their locations, signs, and output-channel scales. Using *activations*—the hidden-state vectors entering the diffusion endpoint's linear layers—we fit the scales by closed-form least squares and select the pruning level by the resulting linear-layer output reconstruction error.

3.1 PROBLEM FORMULATION

We consider AR and diffusion generation with pretrained models $f_A(\cdot;\theta_A)$ and $f_D(\cdot;\theta_D)$, respectively. The difference between their parameters defines the AR-to-diffusion task vector $\tau_{A\to D}$, which we approximate with a compact representation $\hat{\tau}_{A\to D}$:

$$\tau = \theta_D - \theta_A, \qquad \hat{\theta}_D = \theta_A + \hat{\tau}. \tag{1}$$

We write $\tau_{A\to D}$ and $\hat{\tau}_{A\to D}$ as τ and $\hat{\tau}$ for brevity. Our goal is to replace the full task vector τ with a compressed approximation $\hat{\tau}$ while preserving the diffusion endpoint's behavior. AR generation uses θ_A unchanged, whereas diffusion generation uses $\hat{\theta}_D$ under its native inference procedure.

3.2 Compact AR-to-Diffusion Vector with a Ternary Mask

Let $W_{A,\ell}, W_{D,\ell} \in \mathbb{R}^{d_{\text{out}}\times d_{\text{in}}}$ denote corresponding linear-projection weight matrices in the two checkpoints. Their difference $\Delta W_\ell = W_{D,\ell} - W_{A,\ell}$ is the block of τ corresponding to projection ℓ, where r and j index its output and input channels, respectively. Here ℓ indexes linear projections, such as query, key, value, attention output, or an MLP projection.

Following T-Switch (Qi et al., 2025), we prune an α fraction of the smallest-magnitude weight changes separately within the positive and negative groups of ΔW_ℓ. Let $\gamma_\ell^+(\alpha) > 0$ and $\gamma_\ell^-(\alpha) < 0$ denote the corresponding sign-specific cutoffs. We encode the retained changes in a *ternary mask* $S_\ell(\alpha) \in \{-1, 0, +1\}^{d_{\text{out}}\times d_{\text{in}}}$:

$$S_{\ell,rj}(\alpha) = \begin{cases} +1, & \Delta W_{\ell,rj} \geq \gamma_\ell^+(\alpha), \\ -1, & \Delta W_{\ell,rj} \leq \gamma_\ell^-(\alpha), \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

The values $+1$ and -1 mark retained positive and negative changes, respectively, while 0 omits the change and leaves the corresponding AR weight unchanged. The mask admits bit-packed storage of support and sign information, rather than a floating-point value for each entry.

With nonnegative output-channel scales $\boldsymbol{\lambda}_\ell \in \mathbb{R}^{d_{\text{out}}}_{\geq 0}$, the reconstructed weight change is $\widehat{\Delta W}_\ell = \text{diag}(\boldsymbol{\lambda}_\ell) S_\ell(\alpha)$, giving the reconstructed diffusion weight

$$\widehat{W}_{D,\ell} = W_{A,\ell} + \text{diag}(\boldsymbol{\lambda}_\ell) S_\ell(\alpha). \tag{3}$$

Across calibrated projections, these changes form the corresponding components of the compact update $\hat{\tau}$. Each projection stores the ternary mask $S_\ell(\alpha)$ as packed support and sign bits and d_{out} FP32 scales, instead of $d_{\text{out}} d_{\text{in}}$ full-precision update values. The packed code length depends on the retained support, rather than assigning a fixed two-bit slot to every entry. The diffusion endpoint's token embeddings and output head are retained separately without compression.

3.3 Collecting Calibration Activations

To calibrate the compressed update against the endpoint's computations, we collect activations from the frozen diffusion model. We use the diffusion endpoint $f_D(\cdot; \theta_D)$ to generate responses under its native inference procedure, forming a calibration dataset $\mathcal{D}$ of completed prompt–response pairs. We then process each sequence in one additional forward pass through the endpoint using its native attention pattern, without masking, adding noise, or generating further tokens. We collect activations entering each calibrated linear projection ℓ, sampling from both prompt and response positions.

Let $a_\ell(s, i; \theta_D) \in \mathbb{R}^{1\times d_{\text{in}}}$ denote the activation entering projection ℓ at token position i of calibration sequence $s \in \mathcal{D}$. The retained observations define an empirical activation distribution

$$x_\ell = a_\ell(s, i; \theta_D), \qquad x_\ell \sim \mathcal{P}_\ell(\mathcal{D}; \theta_D). \tag{4}$$

Using this, we select the pruning level α and fit the output-channel scales $\boldsymbol{\lambda}_\ell$.

3.4 Mask-On: Diffusion-Native Activation-Based Merging

Reconstruction objective. We calibrate the compressed AR-to-diffusion task vector by matching the diffusion endpoint's linear-layer outputs on the collected diffusion-native activations. Let $\mathbb{E}_\ell[\cdot]$ denote expectation under the empirical activation distribution $\mathcal{P}_\ell(\mathcal{D}; \theta_D)$. Using the reconstructed weights in Equation 3, we measure each projection's output difference on the same activation:

$$J_\ell(\alpha, \boldsymbol{\lambda}_\ell) = \mathbb{E}_\ell \left\| x_\ell \left(\widehat{W}_{D,\ell} - W_{D,\ell} \right)^\top \right\|_2^2. \tag{5}$$

For a predefined nonempty finite set $\mathcal{A} = \{0.05, 0.10, \ldots, 0.95\}$ of candidate pruning levels, with one α shared across calibrated projections, the joint calibration problem is

$$\min_{\substack{\alpha \in \mathcal{A} \\ \boldsymbol{\lambda}_\ell \geq 0 \ \forall \ell}} \sum_{\ell} J_\ell(\alpha, \boldsymbol{\lambda}_\ell). \tag{6}$$

Fitting the scales $\boldsymbol{\lambda}_\ell$. For a fixed α and activation x_ℓ, let $u_{\ell,r}(x_\ell; \alpha) = [x_\ell S_\ell(\alpha)^\top]_r$ and $v_{\ell,r}(x_\ell) = [x_\ell \Delta W_\ell^\top]_r$ denote the output-channel contributions of the ternary mask and the original weight change. Suppressing these arguments for brevity, each channel fits $\lambda_{\ell,r} u_{\ell,r}$ to $v_{\ell,r}$. The objective in Equation 5 separates across output channels, yielding the closed-form solution

$$\lambda^*_{\ell,r}(\alpha) = \max\left(0, \frac{\mathbb{E}_\ell[u_{\ell,r} v_{\ell,r}]}{\mathbb{E}_\ell[u^2_{\ell,r}]}\right). \tag{7}$$

A zero denominator is assigned scale zero.

Selecting the pruning level α. For each pruning level α, the fitted scales attain $\sum_\ell J_\ell(\alpha, \boldsymbol{\lambda}^*_\ell(\alpha))$, the smallest reconstruction error in Equation 6 achievable at that level. We compare candidates using this error normalized by the original adaptation vector's output energy on the same calibration activations:

$$E(\alpha) = \frac{\sum_\ell J_\ell(\alpha, \boldsymbol{\lambda}^*_\ell(\alpha))}{\sum_\ell \mathbb{E}_\ell \|x_\ell \Delta W_\ell^\top\|_2^2}. \tag{8}$$

The denominator equals the reconstruction error obtained by omitting the adaptation entirely on these activations. Assuming this energy is positive, its independence from α makes minimizing $E(\alpha)$ equivalent to minimizing the original reconstruction objective after scale fitting. We therefore select $\alpha^* \in \arg\min_{\alpha \in \mathcal{A}} E(\alpha)$ and reconstruct each diffusion weight as

$$\widehat{W}^*_{D,\ell} = W_{A,\ell} + \mathrm{diag}\big(\boldsymbol{\lambda}^*_\ell(\alpha^*)\big) S_\ell(\alpha^*).$$

This procedure selects α and fits the compact update without gradient-based optimization or downstream task-specific tuning. The resulting update is calibrated to retain diffusion generation behavior under the endpoint's native inference procedure, while the AR checkpoint remains unchanged. Appendix A provides the complete derivation and the optimality of calibration.

4 Results

4.1 Experimental Settings

Models We consider four AR-to-diffusion adapted models: Fast dLLM v2 (Wu et al., 2026a), WeDLM-8B-Instruct (Liu et al., 2025), SDAR-8B-Chat (Cheng et al., 2026), and LLaDA2.0-mini (Bie et al., 2025). As AR anchors, we use Qwen2.5-7B-Instruct (Qwen et al., 2025) for Fast dLLM v2, the post-trained Qwen3-8B (Yang et al., 2025) checkpoint for both WeDLM and SDAR, and Ling-mini-2.0 (Team et al., 2025) for LLaDA2.0-mini.

Baselines We compare three groups of baselines: *parameter-combination* methods—Task Arithmetic (Ilharco et al., 2022), TIES-Merging (Yadav et al., 2023), layer-wise AdaMerging and AdaMerging++ (Yang et al., 2024b), and DARE (Yu et al., 2024); *activation-based* methods—RegMean (Jin et al., 2022), Representation Surgery (RS) (Yang et al., 2024a), and AIM (Heyrani Nobari et al., 2025); and the *task-specific representation* T-Switch (Qi et al., 2025). We use $\beta = 0.6$ for standalone Task Arithmetic, with coefficient sensitivity (See Appendix D.2). For T-Switch, we retain the diffusion endpoint's token embeddings and output head without compression, as in Mask-On. Unless otherwise specified, we follow the original method formulations and reference hyperparameters. Detailed baseline settings are provided in Appendix B.

Calibration data We randomly sample 128 calibration prompts from the Nemotron post-training v2 dataset (Nathawani et al., 2025): 32 each from code, math, STEM, and chat. The dataset contains user–assistant conversations. For Mask-On, we use the user messages as input prompts and generate new responses with the original diffusion endpoint, rather than using the dataset-provided assistant responses. We collect activations from the resulting prompt–response sequences. All data-dependent baselines share the same prompt set and calibration-example budget.

Table 1: **Diffusion-mode accuracy (%) on four benchmarks.** Bold and underlined values mark the best and second-best directly evaluated merging scores for each model and benchmark, respectively. TA and RS denote Task Arithmetic and Representation Surgery, respectively; standalone TA uses $\beta = 0.6$.

	Fast dLLM v2 Qwen2.5-7B-Instruct				WeDLM-8B-Instruct Qwen3-8B				SDAR-8B-Chat Qwen3-8B				LLaDA2.0-mini Ling-mini-2.0			
Method	**GSM8K**	**MBPP**	**ARC-C**	**IFEval**	**GSM8K**	**MBPP**	**ARC-C**	**IFEval**	**GSM8K**	**MBPP**	**ARC-C**	**IFEval**	**GSM8K**	**MBPP**	**ARC-C**	**IFEval**
Endpoint	83.24	50.20	89.33	63.59	85.37	63.40	92.41	69.32	91.28	65.60	91.89	61.92	91.81	66.80	89.76	80.59
Parameter-combination methods																
Task Arithmetic	67.78	43.80	88.65	59.33	82.71	39.80	86.95	70.24	86.73	64.00	**91.89**	**68.58**	52.54	5.80	89.76	45.10
TIES	78.85	44.60	**89.85**	56.01	0.08	0.00	8.62	6.47	2.05	34.60	84.39	29.57	0.76	0.00	63.23	21.26
AdaMerging	13.87	7.40	83.11	38.63	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.08	0.00	25.60	11.28
AdaMerging++	0.08	1.40	77.22	18.85	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.17	2.77
DARE + TA	46.55	35.40	87.88	55.64	0.00	0.00	0.00	0.18	27.52	29.80	87.03	57.49	3.94	0.20	80.63	25.32
Activation-based methods																
RegMean	46.32	40.20	86.69	56.19	75.51	19.40	76.28	49.35	84.61	52.20	87.46	67.10	1.06	0.00	88.23	32.35
RS	19.41	30.40	87.29	50.46	0.99	0.00	77.73	20.15	1.67	4.40	82.00	36.78	0.99	0.00	18.26	24.95
AIM + TA	23.81	25.00	86.77	48.80	0.00	0.00	0.00	0.00	1.14	0.40	2.39	19.04	1.74	0.00	74.83	23.48
Task-specific representations																
T-Switch	75.89	48.60	88.99	60.44	89.08	**65.20**	92.41	72.64	72.86	55.80	91.72	44.36	70.05	32.20	87.46	39.37
MASK-ON	**84.99**	**51.80**	89.16	**64.14**	**90.52**	64.40	**92.83**	**74.86**	**90.45**	**67.40**	**91.89**	65.06	**85.37**	**41.60**	**89.93**	**55.82**

Evaluation protocol We evaluate mathematical reasoning on GSM8K (Cobbe et al., 2021); code generation on MBPP (Austin et al., 2021); science reasoning on ARC-Challenge (Clark et al., 2018); and instruction following on IFEval (Zhou et al., 2023). We use zero-shot prompts, a maximum of 1,024 newly generated tokens, and deterministic token selection under each model's native decoding implementation. AR evaluation uses vLLM with causal attention, whereas diffusion evaluation uses each endpoint's native inference implementation. Details of model-specific decoding, benchmark scoring, and calibration are provided in Appendix C.

4.2 JOINT AR–DIFFUSION PERFORMANCE

Direct parameter merging does not consistently preserve diffusion performance. Table 1 shows that parameter-combination methods preserve diffusion performance unevenly across the models and evaluated domains. For example, TIES slightly exceeds the Fast dLLM v2 endpoint on ARC-C (89.85% versus 89.33%), but reduces MBPP accuracy from 50.20% to 44.60% and IFEval from 63.59% to 56.01%. Activation-based baselines show similar limitations: RegMean and AIM still combine endpoint parameters, while the evaluated RS configuration does not reliably restore diffusion performance across tasks by correcting final-layer activations. This indicates that a shared merged parameter state can preserve performance in one generation mode without reliably maintaining it in the other.

MASK-ON improves diffusion retention. T-Switch and MASK-ON generally show stronger diffusion retention across the evaluated domains while leaving the AR anchor unchanged (Table 2). Among the evaluated merging methods, MASK-ON achieves the highest four-benchmark average diffusion-mode accuracy on all four models: Fast dLLM v2 (72.52%), WeDLM (80.65%), SDAR (78.70%), and LLaDA2.0-mini (68.18%). Compared with T-Switch, MASK-ON improves average diffusion accuracy across all four models, with particularly large gains on SDAR and LLaDA2.0-mini. Although LLaDA2.0-mini remains 14.06 percentage points below its diffusion endpoint (68.18% versus 82.24%), MASK-ON leads the compared merging methods and improves over T-Switch's 57.27% by 10.91 points. These gains support the effectiveness of its reconstruction design beyond adopting a compact representation alone, even where endpoint recovery remains incomplete.

Beyond endpoint performance. We find that some directly evaluated merges even outperform their AR endpoints on individual benchmarks. This observation parallels the findings of Yu et al. (2024), who report that merging fine-tuning updates can improve performance beyond that of individual source models. Because diffusion adaptation includes continued language-model training and instruction tuning in addition to changing the generation objective (Cheng et al., 2026; Liu et al., 2025), parts of the resulting update may remain useful under causal generation.

Table 2: **Average accuracy (%) on four benchmarks.** Each cell reports AR / Diffusion, with the unweighted mean of GSM8K, MBPP, ARC-C, and IFEval computed separately for each mode. Bold and underlined values mark the best and second-best merging scores for each model and mode, respectively. Standalone Task Arithmetic uses $\beta = 0.6$. † denotes AR scores inherited from the unchanged AR endpoint.

Method	Fast dLLM v2 Qwen2.5-7B-Instruct	WeDLM-8B-Instruct Qwen3-8B	SDAR-8B-Chat Qwen3-8B	LLaDA2.0-mini Ling-mini-2.0
Endpoint	79.89 / 71.59	83.47 / 77.63	82.69 / 77.67	81.04 / 82.24
Parameter-combination methods				
Task Arithmetic	79.38 / 64.89	83.30 / 69.92	81.05 / <u>77.80</u>	70.62 / 48.30
TIES	78.37 / 67.33	82.89 / 3.79	73.21 / 37.65	72.71 / 21.31
AdaMerging	79.80 / 35.75	83.94 / 0.00	**84.41** / 0.00	73.66 / 9.24
AdaMerging++	79.07 / 24.39	**84.55** / 0.00	<u>84.35</u> / 0.00	**82.58** / 0.74
DARE + TA	<u>79.94</u> / 56.37	84.01 / 0.05	82.56 / 50.46	70.01 / 27.52
Activation-based methods				
RegMean	79.65 / 57.35	83.78 / 55.14	83.69 / 72.84	74.96 / 30.41
RS	79.89†/ 46.89	83.47†/ 24.72	82.69†/ 31.21	<u>81.04</u>†/ 11.05
AIM + TA	**80.13** / 46.10	<u>84.48</u> / 0.00	84.12 / 5.74	78.63 / 25.01
Task-specific representations				
T-Switch	79.89†/ <u>68.48</u>	83.47†/ <u>79.83</u>	82.69†/ 66.19	<u>81.04</u>†/ <u>57.27</u>
MASK-ON	79.89†/ **72.52**	83.47†/ **80.65**	82.69†/ **78.70**	<u>81.04</u>†/ **68.18**

Similarly, T-Switch and MASK-ON exceed their original diffusion endpoints on individual benchmarks. For MASK-ON, most such gains occur on benchmarks where the AR anchor also outperforms the diffusion endpoint, consistent with retaining AR weights at omitted update coordinates. Prior work on delta compression has also associated above-source performance with regularization from restricted updates (Wang et al., 2026), although our experiments do not isolate these mechanisms. Together, these results show that compact reconstruction can alter downstream task behavior rather than simply interpolate between endpoint accuracies.

Diffusion recovery with modest additional storage. MASK-ON obtains this diffusion performance with a compact mode-specific update rather than a second full checkpoint. Table 3 accounts for required additional bytes relative to the AR weight files, using measured code and scale lengths, uncompressed native tensors, and metadata; unused embedding/head codes and scales are excluded for both methods. MASK-ON adds 14.71–25.48%, which is 0.62–1.41 percentage points more than T-Switch but remains comparable in scale and substantially smaller than a full diffusion checkpoint. Appendix D.5 details the packing format, accounting scope, and component-wise byte counts.

Table 3: Required additional representation storage relative to AR weight files (%).

Method	Fast dLLM v2	WeDLM 8B	SDAR 8B	LLaDA2.0 mini
T-Switch	+23.24%	+24.06%	+24.07%	+14.09%
MASK-ON	+23.98%	+25.16%	+25.48%	+14.71%

5 ANALYSIS

5.1 RECONSTRUCTION ABLATION STUDY

We build on T-Switch to examine the effects of scale fitting, scale granularity, calibration responses, and pruning selection (Table 4). Each row adds (+) or replaces (↔) one component, keeping $\alpha = 0.5$ until the final pruning-selection step. The increments are conditional on preceding choices rather than independent contributions; model-wise results are provided in Appendix D.4.

Weight-space scale fitting. Starting from T-Switch's tensor-wise root mean square (RMS) scales, we apply weight-space least squares (weight-LS) to non-projection backbone tensors and then introduce independent RMS scales for projection output channels. The latter raises average accuracy from 68.17% to 72.89%, with gains across all four benchmarks, highlighting the importance of scale granularity. We also include channel-wise weight-LS as a control that changes the fitting criterion without introducing activations or additional scales. It lowers average accuracy to 69.91%, indicating that the gain from channel-wise scales does not come from a closer weight-space fit.

Table 4: **Reconstruction ablation.** Diffusion accuracy (%), averaged over four models per benchmark; Avg. averages the four benchmarks. Rows marked + add and ↔ replace one component, at $\alpha = 0.5$ until pruning selection. Parentheses show changes from the preceding row.

Reconstruction	GSM8K	MBPP	ARC-C	IFEval	Avg.
T-Switch	76.97	50.45	90.15	54.21	67.94
+ weight-LS for remaining tensors	77.24 (+0.27)	50.45 (+0.00)	90.32 (+0.17)	54.67 (+0.46)	68.17 (+0.22)
+ channel-wise RMS rescaling	86.18 (+8.95)	53.90 (+3.45)	90.81 (+0.49)	60.67 (+6.01)	72.89 (+4.72)
↔ channel-wise weight-LS	82.07 (-4.11)	51.05 (-2.85)	91.21 (+0.41)	55.31 (-5.36)	69.91 (-2.98)
↔ output-LS, dataset responses	87.11 (+5.04)	53.90 (+2.85)	90.85 (-0.36)	64.00 (+8.69)	73.97 (+4.05)
↔ endpoint-generated responses	86.92 (-0.19)	54.25 (+0.35)	90.89 (+0.04)	64.88 (+0.88)	74.24 (+0.27)
+ pruning-level selection (MASK-ON)	87.83 (+0.91)	56.30 (+2.05)	90.96 (+0.06)	64.97 (+0.09)	75.01 (+0.78)

Function-space calibration. Compared with channel-wise RMS, fitting scales by least squares on projection outputs (output-LS; Section 3) with dataset-provided responses raises average accuracy to 73.97%, primarily through gains on IFEval and GSM8K. Relative to the channel-wise weight-LS control, which differs only in fitting projection outputs instead of weights, output-LS raises the average by 4.05 points, isolating the effect of activation weighting. Replacing dataset responses with endpoint-generated responses further raises the average to 74.24%, although individual gains remain model- and task-dependent.

Table 5: **Calibration response sources.** Four-benchmark mean accuracy (%); green values show gains.

Model (α)	Dataset → Endpoint
Fast dLLM v2 (0.40)	72.20 → 72.52 (+0.32)
WeDLM-8B (0.35)	79.73 → 80.65 (+0.92)
SDAR-8B (0.30)	78.38 → 78.70 (+0.32)
LLaDA2.0-mini (0.45)	66.54 → 68.18 (+1.64)
Average	74.21 → 75.01 (+0.80)

Pruning selection without task-specific tuning. Selecting $\alpha \in \{0.05, 0.10, \ldots, 0.95\}$ by calibration reconstruction error raises average accuracy from 74.24% to 75.01%, improving all four benchmark averages. Scales are refitted in closed form, and one selected configuration serves all tasks without gradient-based optimization or downstream benchmark tuning.

Endpoint-generated responses provide a useful calibration reference. Under the final calibration recipe, replacing dataset-provided responses with endpoint-generated responses raises average accuracy from 74.21% to 75.01%, with gains in each model's four-benchmark average (Table 5). Both sources independently select the same pruning level within each model, so the gains do not arise from different pruning choices.

5.2 DIFFUSION RETENTION AND AR–DIFFUSION COOPERATION

To assess whether MASK-ON retains the diffusion endpoint's response to changes in decoding, we compare accuracy and generation throughput (TPS; tokens/s) under six decoding configurations on Fast dLLM v2 (Figure 2): *Top-1 confidence* (Chang et al., 2022), *Top-2 margin* (Kim et al., 2025a), confidence-threshold decoding (Wu et al., 2026b), *KLASS* (Kim et al., 2025b), and *SlowFast* (Wei et al., 2026), which vary how unresolved positions are selected and how aggressively tokens are revealed in parallel. We additionally evaluate a simple, training-free *draft-and-review* composition: MASK-ON diffusion ($\theta_A + \hat{\tau}$) generates a draft at confidence threshold 0.9 or 0.8, after which MASK-ON AR (θ_A) reviews the original question and complete draft, identifies and repairs any error, and produces a corrected answer. As a control, we replace the AR reviewer with the reconstructed diffusion model ($\theta_A + \hat{\tau}$), using the same drafts and review prompts. Both modes here run in Hugging Face transformers (details in Appendix C.4).

MASK-ON retains a similar quality–throughput response. Figure 2 shows that the diffusion endpoint and MASK-ON exhibit similar quality–throughput trends across the six decoding configurations. Both achieve their highest measured accuracy with Top-1 confidence and their highest throughput with confidence threshold 0.8. Although absolute scores differ across configurations, these shared trends suggest that MASK-ON retains useful decoding flexibility rather than preserving performance only under a single fixed strategy.

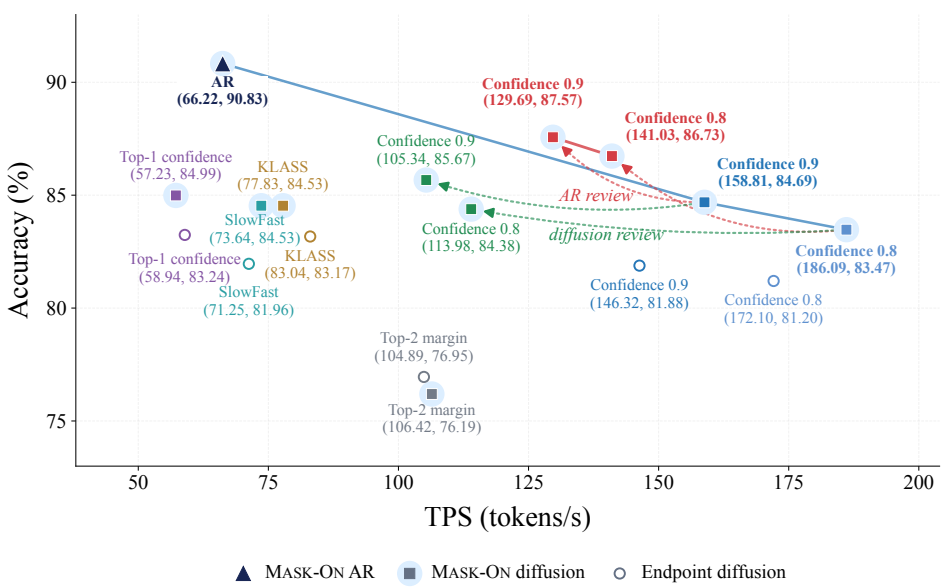

Figure 2: **Accuracy and generation throughput (TPS; tokens/s) on GSM8K with Fast dLLM v2.** The blue line connects its AR and high-throughput diffusion settings; AR-reviewed drafts (red line) add non-dominated points to the accuracy–TPS trade-off. Diffusion-reviewed drafts achieve both lower accuracy and lower throughput than their AR-reviewed counterparts, highlighting the benefit of accessing both modes. Draft-and-review TPS uses total generated tokens divided by the sum of separately measured draft and review generation times, excluding mode switching and loading.

Table 6: **Generation cost of draft-and-review on GSM8K with Fast dLLM v2.** Tokens and seconds are per-question means; the total sums separately measured draft and review generation times. The same saved drafts are supplied to both reviewers.

Draft threshold	Reviewer	Draft tokens	Review tokens	Draft (s)	Review (s)	Total (s)
0.9	Diffusion	297.71	141.92	1.87	2.30	4.17
0.9	AR	297.71	52.53	1.87	**0.83**	**2.70**
0.8	Diffusion	295.63	140.48	1.59	2.24	3.83
0.8	AR	295.63	59.13	1.59	**0.93**	**2.52**

AR review improves diffusion drafts with less generation time than diffusion review. At draft thresholds 0.9 and 0.8, AR review raises GSM8K accuracy from 84.69%/83.47% to 87.57%/86.73%, exceeding diffusion review's 85.67%/84.38% (See example in Appendix D.7, Figure 4). AR review corrects 24.75–25.69% of initially incorrect drafts, compared with 6.88–9.41% for diffusion review, while changing only 1.07–1.18% of initially correct drafts into incorrect answers (Appendix D.6). It also requires less generation time: combined draft-and-review times are 2.70/2.52 seconds per question, compared with 4.17/3.83 seconds for diffusion review (Table 6). Although the separately measured stage times do not establish an end-to-end speedup or runtime-memory savings, these results illustrate a useful form of AR–diffusion cooperation: AR review improves draft quality with less generation time than diffusion-only review.

6 CONCLUSION

We presented MASK-ON, a gradient-free method supporting AR and diffusion generation through an unchanged AR checkpoint and a compact update calibrated to the diffusion's linear-layer outputs. Experiments show competitive diffusion recovery with reduced additional storage, while component effects vary across models and tasks. Although endpoint gaps remain on LLaDA2.0-mini, MASK-ON substantially improves over the compared merging methods. Training-free draft-and-review illustrates complementary use of both modes, improving diffusion drafts with less generation time than diffusion-only review. Memory-efficient execution and end-to-end serving gains remain open.

AI USE STATEMENT

We used generative AI tools to assist with manuscript drafting, organization, and language polishing, as well as literature retrieval and discovery. These tools also supported code implementation and debugging and suggested presentation choices for figures, tables, and experimental descriptions. The authors reviewed and revised AI-assisted content and verified scientific claims, numerical results, experimental settings, and citations against experimental records and original sources. All reported numerical results were computed from experimental outputs using the evaluation procedures described in the paper. The authors take full responsibility for the final content of the paper.

ETHICS STATEMENT

Our work aims to reduce the parameter storage required to support autoregressive and diffusion generation while preserving access to their respective capabilities. Because MASK-ON builds on existing checkpoints, the resulting models may inherit their biases, factual errors, harmful generation behaviors, and risks associated with their training data. Compressing weight changes does not remove these risks or resolve training-data contamination, and may alter safety-relevant behaviors not covered by our evaluations. We encourage application-specific safety evaluation before deployment and responsible use of the released artifacts.

REPRODUCIBILITY STATEMENT

We describe the calibration data and activation collection procedures, and provide model-specific generation settings, benchmark scoring procedures, and computational resources in Appendix C. Baseline implementations are detailed in Appendix B, and complete mathematical derivations are provided in Appendix A. We will release code in supplementary material to enable direct reproducibility.

REFERENCES

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*, 2021.

Tiwei Bie, Maosong Cao, Kun Chen, Lun Du, Mingliang Gong, Zhuochen Gong, Yanmei Gu, Jiaqi Hu, Zenan Huang, Zhenzhong Lan, et al. Llada2. 0: Scaling up diffusion language models to 100b. *arXiv preprint arXiv:2512.15745*, 2025.

Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. Maskgit: Masked generative image transformer. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11305–11315. IEEE, 2022.

Jian Chen, Yesheng Liang, and Zhijian Liu. Dflash: Block diffusion for flash speculative decoding. *arXiv preprint arXiv:2602.06036*, 2026.

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.

Jacob K Christopher, Brian R Bartoldson, Tal Ben-Nun, Michael Cardei, Bhavya Kailkhura, and Ferdinando Fioretto. Speculative diffusion decoding: Accelerating language generation through diffusion. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 12042–12059, 2025.

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

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.

MohammadReza Davari and Eugene Belilovsky. Model breadcrumbs: Scaling multi-task model merging with sparse masks. In *European Conference on Computer Vision*, pp. 270–287. Springer, 2024.

Yonggan Fu, Lexington Whalen, Abhinav Garg, Chengyue Wu, Maksim Khadkevich, Nicolai Oswald, Enze Xie, Daniel Egert, Sharath Turuvekere Sreenivas, Shizhe Diao, et al. Nemotron-labs-diffusion: A tri-mode language model unifying autoregressive, diffusion, and self-speculation decoding. *arXiv preprint arXiv:2607.05722*, 2026.

Amin Heyrani Nobari, Kaveh Alimohammadi, Ali ArjomandBigdeli, Akash Srivastava, Faez Ahmed, and Navid Azizan. Activation-informed merging of large language models. *Advances in neural information processing systems*, 38:72975–72998, 2025.

Chenyu Huang, Peng Ye, Tao Chen, Tong He, Xiangyu Yue, and Wanli Ouyang. Emr-merging: Tuning-free high-performance model merging. *Advances in Neural Information Processing Systems*, 37:122741–122769, 2024.

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.

Xisen Jin, Xiang Ren, Daniel Preotiuc-Pietro, and Pengxiang Cheng. Dataless knowledge fusion by merging weights of language models. *arXiv preprint arXiv:2212.09849*, 2022.

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

Seo Hyun Kim, Sunwoo Hong, Hojung Jung, Youngrok Park, and Se-Young Yun. Klass: Kl-guided fast inference in masked diffusion models. *Advances in Neural Information Processing Systems*, 38:92267–92301, 2025b.

Aiwei Liu, Minghua He, Shaoxun Zeng, Sijun Zhang, Linhao Zhang, Chuhan Wu, Wei Jia, Yuan Liu, Xiao Zhou, and Jie Zhou. Wedlm: Reconciling diffusion language models with standard causal attention for fast inference. *arXiv preprint arXiv:2512.22737*, 2025.

Jingyu Liu, Xin Dong, Zhifan Ye, Rishabh Mehta, Yonggan Fu, Vartika Singh, Ce Zhang, and Pavlo Molchanov. Tidar: Think in diffusion, talk in autoregression. *Proceedings of Machine Learning and Systems*, 8:748–762, 2026.

Dhruv Nathawani, Shuoyang Ding, Vitaly Lavrukhin, Igor Gitman, Somshubra Majumdar, Evelina Bakhturina, Boris Ginsburg, and Jane Polak Scowcroft. Nemotron-Post-Training-Dataset-v2, August 2025. URL https://huggingface.co/datasets/nvidia/Nemotron-Post-Training-Dataset-v2.

OpenAI. Gpt-6 astra system card. Technical report, OpenAI, 2026. URL https://deploymentsafety.openai.com/gpt-6-astra/gpt-6-astra.pdf.

Biqing Qi, Fangyuan Li, Zhen Wang, Junqi Gao, Dong Li, Peng Ye, and Bowen Zhou. Less is more: Efficient model merging with binary task switch. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15265–15274. IEEE, 2025.

Qwen, :, 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, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Chenyang Shao, Sijian Ren, Fengli Xu, and Yong Li. Diffuse thinking: Exploring diffusion language models as efficient thought proposers for reasoning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 26739–26757, 2026.

Wenju Sun, Qingyong Li, Wen Wang, Yang Liu, Yangliao Geng, and Boyang Li. Towards minimizing feature drift in model merging: Layer-wise task vector fusion for adaptive knowledge integration. *Advances in Neural Information Processing Systems*, 38:172635–172663, 2025.

Ling Team, Ang Li, Ben Liu, Binbin Hu, Bing Li, Bingwei Zeng, Borui Ye, Caizhi Tang, Changxin Tian, Chao Huang, et al. Every activation boosted: Scaling general reasoner to 1 trillion open language foundation. *arXiv preprint arXiv:2510.22115*, 2025.

Xiaohui Wang, Peng Ye, Chenyu Huang, Shenghe Zheng, Bo Zhang, Lei Bai, Wanli Ouyang, and Tao Chen. Breaking the compression ceiling: Data-free pipeline for ultra-efficient delta compression. *Advances in Neural Information Processing Systems*, 38:23167–23195, 2026.

Qingyan Wei, Yaojie Zhang, Zhiyuan Liu, Puyu Zeng, Yuxuan Wang, Biqing Qi, Dongrui Liu, and Linfeng Zhang. Accelerating diffusion large language models with slowfast sampling: The three golden principles. In *International Conference on Learning Representations*, volume 2026, pp. 4340–4354, 2026.

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.

Chengyue Wu, Hao Zhang, Shuchen Xue, Zhijian Liu, Shizhe Diao, Ligeng Zhu, Ping Luo, Song Han, and Enze Xie. Fast-dllm: Training-free acceleration of diffusion llm by enabling kv cache and parallel decoding. In *International Conference on Learning Representations*, volume 2026, pp. 57027–57051, 2026b.

Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chenchen Ling, et al. Deepseek-v4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026.

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.

Yutaro Yamada, Robert Tjarko Lange, Cong Lu, Shengran Hu, Chris Lu, Jakob Foerster, Jeff Clune, and David Ha. The ai scientist-v2: Workshop-level automated scientific discovery via agentic tree search. *arXiv preprint arXiv:2504.08066*, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Enneng Yang, Li Shen, Zhenyi Wang, Guibing Guo, Xiaojun Chen, Xingwei Wang, and Dacheng Tao. Representation surgery for multi-task model merging. *arXiv preprint arXiv:2402.02705*, 2024a.

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.

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.

Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, et al. Glm-5: from vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*, 2026.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023.

Supplementary Material

CONTENTS

A Complete Derivation **14**

A.1 Empirical Reconstruction Objective 14

A.2 Fitting the Scales 14

A.3 Selecting the Pruning Level 15

A.4 Proof of Proposition 1 16

B Baseline Details **16**

B.1 Common comparison setup 16

B.2 Parameter-combination methods 17

B.3 Activation-based methods 17

B.4 Task-specific representations 18

C Experimental Details **18**

C.1 Generation Configuration 18

C.2 Benchmarks and Scoring 19

C.3 Calibration Data and Activation Collection 19

C.4 Decoding and Draft-and-Review Settings 20

D Additional Results **21**

D.1 Full Results 21

D.2 Task Arithmetic Coefficient Sensitivity under GSM8K 22

D.3 Activation Collection Ablation 23

D.4 Reconstruction Breakdown 24

D.5 Storage Accounting 24

D.6 Answer Transitions in Draft-and-Review 25

D.7 A Qualitative Example of Draft-and-Review 26

E Limitation and Future Work **28**

A Complete Derivation

We derive the reconstruction objective and the two optimization steps in Section 3.4, then prove Proposition 1. Throughout, ℓ indexes a calibrated projection, r an output channel, and j an input channel. The empirical activation distributions $\mathcal{P}_\ell(\mathcal{D};\theta_D)$, linear-weight changes ΔW_ℓ, and ternary masks $S_\ell(\alpha)$ for $\alpha \in \mathcal{A}$ are fixed. We use $\mathbb{E}_\ell$ for expectation under $\mathcal{P}_\ell$, as in the Section 3.4.

A.1 Empirical Reconstruction Objective

Let $x_{\ell,i} \in \mathbb{R}^{1\times d_{\text{in}}}$ denote retained activation i and let $n_\ell > 0$ be the number retained for projection ℓ. The empirical distribution is

$$\mathcal{P}_\ell(\mathcal{D};\theta_D) = \frac{1}{n_\ell}\sum_{i=1}^{n_\ell} \delta_{x_{\ell,i}}, \qquad \mathbb{E}_\ell[\varphi(x_\ell)] = \frac{1}{n_\ell}\sum_{i=1}^{n_\ell} \varphi(x_{\ell,i}), \tag{9}$$

where δ_x is a point mass at activation x. Because the collection retains the same number of fitting activations per calibration domain, this row average also gives equal weight to each domain.

For the same activation x_ℓ, the difference between the reconstructed and original linear-layer outputs is expressed by canceling out the common AR contribution as

$$\begin{aligned} x_\ell \widehat{W}_{D,\ell}^\top - x_\ell W_{D,\ell}^\top &= x_\ell\big[(W_{A,\ell} + \widehat{\Delta W}_\ell) - (W_{A,\ell} + \Delta W_\ell)\big]^\top \\ &= x_\ell(\widehat{\Delta W}_\ell - \Delta W_\ell)^\top. \end{aligned} \tag{10}$$

Let $\mathcal{J}(\hat{\tau};\mathcal{D})$ denote the total output reconstruction error summed over calibrated projections, and stack the retained activations as the rows of $X_\ell \in \mathbb{R}^{n_\ell \times d_{\text{in}}}$. Then

$$\begin{aligned} \mathcal{J}(\hat{\tau};\mathcal{D}) &= \sum_\ell \mathbb{E}_\ell \big\| x_\ell(\widehat{\Delta W}_\ell - \Delta W_\ell)^\top \big\|_2^2 \\ &= \sum_\ell \frac{1}{n_\ell} \big\| X_\ell(\widehat{\Delta W}_\ell - \Delta W_\ell)^\top \big\|_F^2. \end{aligned} \tag{11}$$

The second equality follows because the squared Frobenius norm sums the squared Euclidean norms of all matrix rows.

Substituting the representation $\widehat{\Delta W}_\ell = \text{diag}(\boldsymbol{\lambda}_\ell) S_\ell(\alpha)$ into Equation 11 yields

$$\mathcal{J}(\hat{\tau};\mathcal{D}) = \sum_\ell \underbrace{\mathbb{E}_\ell \big\| x_\ell(\text{diag}(\boldsymbol{\lambda}_\ell) S_\ell(\alpha) - \Delta W_\ell)^\top \big\|_2^2}_{J_\ell(\alpha, \boldsymbol{\lambda}_\ell)}. \tag{12}$$

A.2 Fitting the Scales

Fix a candidate α. As in Section 3.4, write $u_{\ell,r}(x_\ell;\alpha) = [x_\ell S_\ell(\alpha)^\top]_r$ and $v_{\ell,r}(x_\ell) = [x_\ell \Delta W_\ell^\top]_r$ for the output-channel contributions of the code and original weight change. Because $\text{diag}(\boldsymbol{\lambda}_\ell)$ rescales rows of the weight matrix, it rescales the corresponding output channels:

$$\big[x_\ell(\text{diag}(\boldsymbol{\lambda}_\ell) S_\ell(\alpha) - \Delta W_\ell)^\top\big]_r = \lambda_{\ell,r} u_{\ell,r}(x_\ell;\alpha) - v_{\ell,r}(x_\ell). \tag{13}$$

Suppressing the arguments of $u_{\ell,r}$ and $v_{\ell,r}$, expanding the squared Euclidean norm and using linearity of expectation separates the objective:

$$\begin{aligned} J_\ell(\alpha, \boldsymbol{\lambda}_\ell) &= \sum_{r=1}^{d_{\text{out}}} q_{\ell,r}(\lambda_{\ell,r};\alpha), \\ q_{\ell,r}(\lambda;\alpha) &= \mathbb{E}_\ell[(\lambda u_{\ell,r} - v_{\ell,r})^2]. \end{aligned} \tag{14}$$

Define the scalar coefficients

$$\begin{aligned} a_{\ell,r}(\alpha) &= \mathbb{E}_\ell[u_{\ell,r}^2], \\ b_{\ell,r}(\alpha) &= \mathbb{E}_\ell[u_{\ell,r} v_{\ell,r}], \\ e_{\ell,r} &= \mathbb{E}_\ell[v_{\ell,r}^2]. \end{aligned} \tag{15}$$

Here $a_{\ell,r}(\alpha) \geq 0$ is the coded-output energy, $b_{\ell,r}(\alpha)$ measures its alignment with the original adaptation output, and $e_{\ell,r}$ is the original adaptation-output energy. Since only the first two depend on α, expanding the squared norm gives

$$q_{\ell,r}(\lambda;\alpha) = a_{\ell,r}(\alpha)\lambda^2 - 2b_{\ell,r}(\alpha)\lambda + e_{\ell,r}. \tag{16}$$

If $a_{\ell,r}(\alpha) > 0$, this quadratic is strictly convex, with derivatives

$$\frac{\partial q_{\ell,r}}{\partial \lambda} = 2a_{\ell,r}(\alpha)\lambda - 2b_{\ell,r}(\alpha), \qquad \frac{\partial^2 q_{\ell,r}}{\partial \lambda^2} = 2a_{\ell,r}(\alpha) > 0.$$

Its unconstrained minimizer is $b_{\ell,r}(\alpha)/a_{\ell,r}(\alpha)$. When this value is nonnegative, it is also the constrained minimizer. Otherwise $b_{\ell,r}(\alpha) < 0$, so the derivative is positive throughout $\lambda \geq 0$ and the constrained minimum occurs at $\lambda = 0$.

If $a_{\ell,r}(\alpha) = 0$, nonnegativity of $u_{\ell,r}^2$ implies $u_{\ell,r} = 0$ with probability one under $\mathcal{P}_\ell$. Consequently, $b_{\ell,r}(\alpha) = 0$. The objective is then the constant $e_{\ell,r}$: all nonnegative scales minimize it, and we select zero. Together these cases give the complete form of Equation 7:

$$\lambda^*_{\ell,r}(\alpha) = \begin{cases} \max\left(0, \dfrac{b_{\ell,r}(\alpha)}{a_{\ell,r}(\alpha)}\right), & a_{\ell,r}(\alpha) > 0, \\ 0, & a_{\ell,r}(\alpha) = 0. \end{cases} \tag{17}$$

The nonnegative constraint prevents a scale from reversing the signs in $S_\ell(\alpha)$; the separate zero-energy case avoids evaluating an undefined ratio. Because the constraints and quadratics are independent across output channels and projections, these solutions jointly minimize $\sum_\ell J_\ell$ for the fixed α.

The fit also retains input-channel correlations. To see this, define the uncentered second-moment matrix as $\Gamma_\ell = \mathbb{E}_\ell[x_\ell^\top x_\ell]$, and let $s_{\ell,r}(\alpha)$ and $\Delta W_{\ell,r}$ be row r of $S_\ell(\alpha)$ and ΔW_ℓ. Then

$$q_{\ell,r}(\lambda;\alpha) = (\lambda s_{\ell,r}(\alpha) - \Delta W_{\ell,r})\Gamma_\ell(\lambda s_{\ell,r}(\alpha) - \Delta W_{\ell,r})^\top. \tag{18}$$

This identity follows by substituting the definition of Γ_ℓ and using $(x_\ell z^\top)^2 = z x_\ell^\top x_\ell z^\top$ for each row vector z. Computing the coefficients directly from $u_{\ell,r}$ and $v_{\ell,r}$ therefore accounts for the off-diagonal correlations without forming or inverting Γ_ℓ. No full-rank assumption on the activation matrices is needed.

A.3 Selecting the Pruning Level

Substituting Equation 17 into the scalar quadratic gives its minimum value:

$$q^*_{\ell,r}(\alpha) := \min_{\lambda \geq 0} q_{\ell,r}(\lambda;\alpha) = \begin{cases} e_{\ell,r} - \dfrac{\max(0, b_{\ell,r}(\alpha))^2}{a_{\ell,r}(\alpha)}, & a_{\ell,r}(\alpha) > 0, \\ e_{\ell,r}, & a_{\ell,r}(\alpha) = 0. \end{cases} \tag{19}$$

For positive $b_{\ell,r}(\alpha)$, this follows from $a(b/a)^2 - 2b(b/a) + e = e - b^2/a$; for nonpositive $b_{\ell,r}(\alpha)$, the minimizing scale is zero and the loss is $e_{\ell,r}$. Consequently, the summed projection error after scale fitting is

$$\sum_\ell J_\ell(\alpha, \boldsymbol{\lambda}^*_\ell(\alpha)) = \sum_\ell \sum_{r=1}^{d_{\text{out}}} q^*_{\ell,r}(\alpha). \tag{20}$$

The denominator of Equation 8, $\sum_\ell \mathbb{E}_\ell \|x_\ell \Delta W_\ell^\top\|_2^2$, can be written as

$$Z := \sum_\ell \mathbb{E}_\ell \|x_\ell \Delta W_\ell^\top\|_2^2 = \sum_\ell \sum_{r=1}^{d_{\text{out}}} e_{\ell,r}. \tag{21}$$

Setting $\hat{\tau} = 0$ in Equation 11 gives $Z = \mathcal{J}(0;\mathcal{D})$. Thus, Z measures the output difference between the AR and diffusion projections on the same recorded diffusion-endpoint activations. Both the recorded activations and the original linear-weight changes are fixed, so $v_{\ell,r}$ and Z are independent

of α and the fitted scales. Equation 8 therefore gives $E(\alpha) = \sum_\ell J_\ell(\alpha, \boldsymbol{\lambda}_\ell^*(\alpha))/Z$. Under the assumption $Z > 0$, division by Z preserves every comparison:

$$\arg\min_{\alpha \in \mathcal{A}} E(\alpha) = \arg\min_{\alpha \in \mathcal{A}} \sum_\ell J_\ell(\alpha, \boldsymbol{\lambda}_\ell^*(\alpha)) \,. \tag{22}$$

Thus, fitting the scales for every candidate and comparing their summed errors solves the remaining finite minimization. A minimizing candidate exists because $\mathcal{A}$ is nonempty and finite and each candidate's minimum error is attained by the scales above. The selected α is shared across all calibrated projections; selecting a different value for each projection would solve a different optimization problem from the global-pruning problem in Section 3.4.

If $Z = 0$, all target outputs vanish with probability one under their corresponding $\mathcal{P}_\ell$, and zero scales attain zero unnormalized loss. The normalized score is undefined in that case, which is why Proposition 1 assumes positive target-output energy.

A.4 Proof of Proposition 1

Proposition 1 (Optimality of finite-candidate calibration)**.** *Fix a nonempty finite candidate set $\mathcal{A}$, its ternary masks, and finite empirical activation distributions with fixed sampling probabilities. Assume positive total adaptation-output energy. In exact arithmetic, fitting the nonnegative scales in Equation 7 for every candidate, with zero scale for a zero denominator, and selecting a minimizer of Equation 8 solves*

$$\min_{\substack{\alpha \in \mathcal{A} \\ \boldsymbol{\lambda}_\ell \geq 0 \ \forall \ell}} \sum_\ell J_\ell(\alpha, \boldsymbol{\lambda}_\ell).$$

Proof. For each $\alpha \in \mathcal{A}$, Section A.2 constructs nonnegative scales $\boldsymbol{\lambda}_\ell^*(\alpha)$ that attain

$$ZE(\alpha) = \sum_\ell J_\ell(\alpha, \boldsymbol{\lambda}_\ell^*(\alpha)) = \min_{\{\boldsymbol{\lambda}_\ell \geq 0\}_\ell} \sum_\ell J_\ell(\alpha, \boldsymbol{\lambda}_\ell).$$

This includes channels with zero coded-output energy. By Section A.3, the normalized selector chooses $\alpha^* \in \arg\min_{\alpha \in \mathcal{A}} E(\alpha)$. The selected pruning level and its nonnegative scales satisfy the constraints of the joint calibration problem.

Now take any candidate $\widetilde{\alpha} \in \mathcal{A}$ and any nonnegative scales $\widetilde{\boldsymbol{\lambda}}_\ell$ for all projections. Since $Z > 0$, selection of α^* and per-candidate scale optimality give

$$\begin{aligned} \sum_\ell J_\ell(\alpha^*, \boldsymbol{\lambda}_\ell^*(\alpha^*)) &= ZE(\alpha^*) \\ &\leq ZE(\widetilde{\alpha}) \\ &\leq \sum_\ell J_\ell(\widetilde{\alpha}, \widetilde{\boldsymbol{\lambda}}_\ell). \end{aligned} \tag{23}$$

The first inequality follows from minimizing over $\mathcal{A}$; the second follows from minimizing the scales for the fixed $\widetilde{\alpha}$. Since the alternative candidate and scales were arbitrary and the selected ones are feasible, this proves

$$\sum_\ell J_\ell(\alpha^*, \boldsymbol{\lambda}_\ell^*(\alpha^*)) = \min_{\substack{\alpha \in \mathcal{A} \\ \boldsymbol{\lambda}_\ell \geq 0 \ \forall \ell}} \sum_\ell J_\ell(\alpha, \boldsymbol{\lambda}_\ell).$$

Duplicate ternary masks, tied pruning-level scores, or nonunique scales in zero-energy channels do not affect the argument. □

B Baseline Details

B.1 Common comparison setup

All baselines use the model pairs and evaluation protocol in Section 4.1 and Appendix C. We write $\Delta W_\ell = W_{D,\ell} - W_{A,\ell}$ for an aligned parameter matrix. Using the AR checkpoint as the anchor leaves one nonzero task vector. Data-dependent methods share the 128 Nemotron fitting prompts described in Section 4.1; method-specific inputs and fitting procedures are given below. All fitted parameters are frozen across benchmarks.

B.2 PARAMETER-COMBINATION METHODS

Task Arithmetic. Task Arithmetic (Ilharco et al., 2022) adds scaled task vectors to an anchor. In our setting, it reduces to

$$W_{\mathrm{TA},\ell} = W_{A,\ell} + \beta \Delta W_\ell. \tag{24}$$

We use a fixed $\beta = 0.6$ for standalone Task Arithmetic across all models and benchmarks. Appendix D.2 examines how varying β affects AR- and diffusion-mode accuracy on GSM8K across the four model pairs.

TIES-Merging. TIES (Yadav et al., 2023) trims small task-vector entries, elects a magnitude-weighted sign, and averages sign-consistent entries. We retain the top 20% of absolute entries in the concatenated task vector, with scale 1 and disjoint averaging. Because our AR anchor leaves only one nonzero task vector, this becomes $\theta_A + \mathrm{Trim}_{0.2}(\tau)$, without inter-task sign conflicts.

Layer-wise AdaMerging and AdaMerging++. AdaMerging (Yang et al., 2024b) optimizes merge coefficients by minimizing prediction entropy while keeping endpoint weights fixed:

$$W_{\mathrm{Ada},\ell} = W_{A,\ell} + \beta_\ell \Delta W_\ell. \tag{25}$$

We assign one coefficient per aligned named parameter, sharing coefficients across tied aliases. AdaMerging++ uses the same procedure on the top-20% TIES-preprocessed task vector. For language-model calibration, we minimize mean token-prediction entropy at unresolved positions in the first denoising step of the native decoder. Coefficients start at 0.3 and are clamped to $[0, 1]$ in the forward pass. We use Adam with learning rate 10^{-3} for 500 updates, with an effective batch of four prompts, one from each domain. We select the coefficient state with the lowest entropy on 32 disjoint validation prompts, evaluated initially and every 25 updates; ties retain the earlier state. Benchmark inputs are not used for adaptation or selection.

DARE on Task Arithmetic. DARE (Yu et al., 2024) randomly removes weight changes and rescales the survivors before merging. With independent $B_{\ell,rj} \sim \mathrm{Bernoulli}(1-p)$, we use

$$W_{\mathrm{DARE},\ell} = W_{A,\ell} + \beta \frac{B_\ell \odot \Delta W_\ell}{1-p}, \qquad p = 0.5, \quad \beta = 0.5. \tag{26}$$

These values follow the `supermario_v2` configuration in Appendix A.5 of Yu et al. (2024). We sample the mask once and reuse it across generation modes and benchmarks.

B.3 ACTIVATION-BASED METHODS

RegMean. RegMean (Jin et al., 2022) combines linear projections using full activation second moments. For endpoint $t \in \{A, D\}$, let $G_{t,\ell} = \mathbb{E}[x_{t,\ell}^\top x_{t,\ell}]$ denote the empirical moment of activations entering projection ℓ. We collect activations at every token position computed during generation from the calibration prompts and average their second moments over all collected rows. AR collection uses greedy causal generation with a 512-new-token cap; diffusion collection follows the endpoint's native decoder. We use equal endpoint weights and off-diagonal shrinkage $\kappa = 0.9$:

$$\begin{aligned} \widetilde{G}_{t,\ell} &= \kappa G_{t,\ell} + (1-\kappa)\,\mathrm{diag}(G_{t,\ell}) + \rho_\ell I, \\ W_{\mathrm{R},\ell} &= (W_{A,\ell}\widetilde{G}_{A,\ell} + W_{D,\ell}\widetilde{G}_{D,\ell})(\widetilde{G}_{A,\ell} + \widetilde{G}_{D,\ell})^{-1}. \end{aligned} \tag{27}$$

Here $\mathrm{diag}(G)$ retains only the diagonal of G. We solve the linear system directly, with $\rho_\ell = 0$ except for SDAR, where symmetric diagonal regularization uses $\rho_\ell = 10^{-4}\,\mathrm{tr}(G_{A,\ell} + G_{D,\ell})/(2d_\ell)$ and d_ℓ is the input width. The loading is applied to both endpoint moments in both the system matrix and the weighted sum of endpoint weights. Parameters outside the regression scope, including embeddings and biases, are averaged equally.

For LLaDA2.0-mini and its AR counterpart, we record the activations entering each MoE layer and separately pass them through every expert, regardless of the router's selection. Each expert's down projection receives that expert's own intermediate activations. These additional computations are used only to collect statistics and do not alter native routing or model outputs. We solve on input channels with nonzero second-moment energy and use the equal endpoint average for zero-energy channels, whose weights are unconstrained by the reconstruction objective.

Representation Surgery (RS). Representation Surgery (Yang et al., 2024a) learns a lightweight module that aligns a merged model's final-layer activations with those of the target endpoint. We use a fixed Task Arithmetic carrier with $\beta = 0.4$ and a rank-16, bias-free ReLU bottleneck immediately before the output head, applying the subtractive correction $z - \Phi(z)$. The module minimizes the L_1 difference from the diffusion endpoint's activations when both models process identical inputs at the first denoising step. Only the module is trained, using Adam with learning rate 10^{-3}, batch size 16, and 1,000 updates; the final module is used for evaluation. This is an adaptation of RS to language-model representations rather than its original vision-task calibration protocol.

AIM on Task Arithmetic. AIM (Heyrani Nobari et al., 2025) attenuates a merge according to activation importance at the AR anchor. For input channel j, let $a_{\ell,j}$ be the mean absolute activation, first averaged over positions within each prompt and then equally across prompts. We set $I_{\ell,j} = a_{\ell,j} / \max_k a_{\ell,k}$, with $I_\ell = 0$ if all channels have zero importance, and apply

$$W_{\mathrm{AIM},\ell,rj} = W_{A,\ell,rj} + \big[1 - (1-\omega) I_{\ell,j}\big]\big(W_{\mathrm{TA},\ell,rj} - W_{A,\ell,rj}\big). \tag{28}$$

We use $\omega = 0.4$ and a Task Arithmetic carrier with $\beta = 0.4$. Statistics come from one causal forward pass of each calibration prompt through the AR anchor, without masking or added noise. For Ling-mini-2.0, the recorded activations entering each MoE layer are separately passed through every expert, as described above for RegMean. Importance weighting covers non-embedding modules, including normalization layers and the output head; token embeddings retain their Task Arithmetic values.

B.4 Task-specific representations

T-Switch and Mask-On retain the AR anchor and reconstruct diffusion weights by adding a compressed update. Both retain the diffusion endpoint's token embeddings and output head without compression. Reported additional storage includes codes, scales, and separately retained native tensors.

T-Switch. T-Switch (Qi et al., 2025) stores support and sign codes with a scale. We discard a nominal $\alpha = 0.5$ fraction of small-magnitude changes within each sign group and use one retained-update RMS scale per named tensor:

$$W_{\mathrm{TS},\ell} = W_{A,\ell} + \lambda_\ell S_\ell(0.5), \qquad \lambda_\ell = \frac{\|\Delta W_\ell \odot |S_\ell(0.5)|\|_F}{\|S_\ell(0.5)\|_F}. \tag{29}$$

Here $|S_\ell(0.5)|$ identifies retained entries; empty support gives scale zero. The rule applies to all aligned non-vocabulary tensors, including normalization weights and biases, with no activation-based fitting. Embeddings, the output head, and unmatched native tensors are retained exactly.

C Experimental Details

C.1 Generation Configuration

All evaluations use NVIDIA B200 GPUs. We use zero-shot prompts and a maximum of 1,024 newly generated tokens, with native stopping conditions allowing earlier termination. Within each model and generation mode, methods share the same benchmark items, rendered prompts, decoding settings, precision, and scoring rules. Diffusion evaluation uses batch size one and each endpoint's native inference implementation with deterministic token selection.

Fast dLLM v2. We use BF16, temperature 0, top-$p = 1$, confidence threshold 1, 32-token blocks, and eight-token sub-blocks, with block caching disabled. The native chat template includes the system instruction, *You are a helpful assistant.*

WeDLM-8B-Instruct. We use the native engine in BF16 with temperature 0, top-$p = 1$, no top-k filtering, entropy threshold 0.4, and positional penalty 0.02. Generation stops at the native end-of-text or end-of-turn token.

SDAR-8B-Chat. We use the official static block decoder in FP16, with block length 4, four denoising steps, temperature 1, top-$k = 1$, top-$p = 1$, and low-confidence remasking with threshold 1. Restricting sampling to top-$k = 1$ makes token selection deterministic.

LLaDA2.0-mini. We use the native generation interface in BF16 with block length 32, 32 denoising steps, temperature 0, and early stopping at EOS.

AR generation. We use vLLM 0.11.0 without quantization, greedy decoding, and up to eight requests per batch. Qwen3 evaluation disables thinking through its native template option. AR evaluation uses the same rendered prompts, output budget, precision, and scorers as the corresponding diffusion comparison, retokenizing the text with the AR tokenizer and retaining native AR stopping conditions. WeDLM and SDAR therefore have separate Qwen3 reference scores because their prompt templates and BF16/FP16 precision differ. Directly merged checkpoints are evaluated under causal generation; dagger-marked scores in Table 8 reuse the unchanged AR endpoint.

C.2 Benchmarks and Scoring

Table 7 summarizes the evaluation sets and primary metrics. Every method uses the same items and scoring rules within each comparison. Invalid predictions count as incorrect and remain in the denominator.

Table 7: Evaluation benchmarks, sample counts, and scoring criteria.

Benchmark	Items	Task and metric
GSM8K	1,319	Grade-school mathematical reasoning; final numerical-answer accuracy.
MBPP	500	Python code generation; pass@1 on all provided tests.
ARC-Challenge	1,172	Multiple-choice science reasoning; generated answer-label accuracy.
IFEval	541	Verifiable instruction following; strict prompt-level accuracy.

GSM8K. Prompts request step-by-step reasoning followed by `#### <answer>`. The extractor removes commas, considers text before any `Question:` delimiter, and compares the last numerical answer with the reference.

MBPP. Prompts contain the problem description and provided tests, without reference programs or few-shot examples. We generate one completion per item and execute the extracted program in a `Bubblewrap` sandbox. A completion passes only if it satisfies all provided tests.

ARC-Challenge. Prompts list the answer choices and request a label. The common extractor prioritizes a punctuated leading choice, followed by explicit-answer patterns and corroborated choice evidence; incidental letters in prose or echoed instructions are not accepted as answers. The extraction rule was refined using saved outputs and applied uniformly across methods, without access to reference answers.

IFEval. We retain the original instruction prompts without adding an answer-format request. Strict prompt-level accuracy requires every instruction in a prompt to be satisfied. The 541 public prompts are used only for evaluation. We use the official checks with one shared correction: when a character-frequency instruction explicitly requests punctuation, the checker counts that literal symbol rather than substituting a random letter. All other instruction checks are unchanged, and the correction is applied to every endpoint and method.

C.3 Calibration Data and Activation Collection

We sample 128 fitting prompts and 32 held-out validation prompts from Nemotron post-training v2: 32 fitting and eight validation prompts from each of code, math, STEM, and chat. No evaluation item is used for calibration. Pruning selection and scale fitting use fitting data only; validation prompts assess reconstruction without refitting or selecting parameters.

Calibration activations. Following Section 3.3, the frozen diffusion endpoint generates a response to each prompt and then processes the completed prompt–response sequence in one additional forward pass, with caching disabled. We reuse the generated token IDs and retain positions through the first native terminal token, without adding masks, noise, or padding positions. This additional forward pass uses 32-token block-causal attention for Fast dLLM v2 and LLaDA2.0-mini, four-token block-causal attention for SDAR, and causal attention for WeDLM. Calibration response generation is separate from benchmark evaluation; its native generation configurations are provided with the experiment code.

Activations are extracted immediately before each calibrated linear projection, after its preceding normalization, attention computation, or MLP nonlinearity and gating. We sample uniformly without replacement from prompt and response positions, retaining 384 fitting and 192 validation rows per projection and domain: 1,536 fitting and 768 validation rows in total. Domains receive equal weight. For LLaDA2.0-mini, we also record the activations entering each MoE layer and separately pass them through every expert, regardless of the router's selection. Each expert's down projection receives its own intermediate activations. These additional computations provide calibration activations for every expert without altering native routing or model outputs.

Fitting and response-source comparison. We search $\mathcal{A} = \{0.05, 0.10, \ldots, 0.95\}$ and fit nonnegative output-channel scales using the objective in Section 3.4. The selected pruning levels are 0.40, 0.35, 0.30, and 0.45 for Fast dLLM v2, WeDLM, SDAR, and LLaDA2.0-mini, respectively. For the matched-source comparison, we repeat this procedure with the dataset-provided assistant responses while retaining the same prompts, activation budgets, and domain weights. Both sources select the same pruning levels within each model. Appendix D.3 compares this completed-sequence collection with intermediate denoising activations and examines the benefit of endpoint-generated rather than dataset-provided responses.

C.4 Decoding and Draft-and-Review Settings

Section 5.2 evaluates all 1,319 GSM8K test items with Fast dLLM v2 and a fixed Mask-On checkpoint, anchored at Qwen2.5-7B-Instruct with native embeddings and output head. We use BF16, batch size one, greedy token selection, 32-token blocks, eight-token sub-blocks, and a 1,024-new-token limit. All generations in this section use Hugging Face transformers rather than the vLLM backend used for the AR results in Section 4.1: diffusion uses Fast dLLM v2's native block decoder, and the standalone AR reference and AR review use eager dispatch with SDPA attention and a KV cache, without compilation or CUDA graphs.

Decoding configurations. We evaluate six configurations from the five strategies below, varying which unresolved positions are revealed and how many are revealed per step. Selected positions receive their highest-probability tokens, and already revealed tokens remain fixed. KLASS and SlowFast are adapted to Fast dLLM v2's native sub-block boundaries and prefix-cache interface.

Top-1 confidence. Following confidence-guided token selection (Chang et al., 2022), we score each unresolved position by its maximum predicted vocabulary probability. Our configuration reveals only the highest-confidence position in the active sub-block at each denoising step. The order therefore follows model confidence rather than a fixed left-to-right traversal, while revealing one position at a time allows subsequent predictions to incorporate each committed token.

Top-2 margin. Probability-margin selection (Kim et al., 2025a) ranks positions by $p_{(1)} - p_{(2)}$, where $p_{(1)}$ and $p_{(2)}$ are the two largest predicted vocabulary probabilities at a position. A larger margin indicates stronger separation between the preferred token and its closest alternative, rather than high top-1 probability alone. We reveal up to two unresolved positions with the largest margins per step. The two-token probability comparison defines the ranking score; the two-position reveal budget is a separate setting in our evaluation.

Confidence threshold. Confidence-aware parallel decoding (Wu et al., 2026b) determines the number of revealed positions from their predictive confidence instead of a fixed quota. We reveal all unresolved positions in the active sub-block whose maximum token probability exceeds the threshold, evaluating thresholds 0.9 and 0.8 as separate configurations. If no position qualifies,

we reveal the highest-confidence position to ensure progress. Lowering the threshold permits more simultaneous predictions before their mutual dependencies are resolved.

KLASS. KLASS (Kim et al., 2025b) combines confidence with temporal stability, measured by the KL divergence between a position's current and previous predicted vocabulary distributions. This criterion distinguishes a confident prediction from one that also remains stable as the surrounding context is resolved. We reveal a position when its confidence exceeds 0.6 and both of its two most recent consecutive-step KL divergences are below 0.015. The divergence is computed from the current distribution to the previous one, and the history is reset at each sub-block boundary. If no position satisfies both criteria, including while the history is incomplete, we reveal the highest-confidence unresolved position.

SlowFast. SlowFast (Wei et al., 2026) alternates between exploratory decoding that identifies a suitable interval and accelerated decoding within that interval. Our configuration uses six forward passes per exploratory stage. It estimates the interval's right boundary from the rightmost unresolved position with confidence at least 0.3, declaring the estimate stable when the population standard deviation of the last two boundary estimates is below 1. If stability is not reached, the final window's mean determines the boundary. During accelerated decoding, positions within the interval with confidence at least 0.9 are revealed in parallel, with a highest-confidence fallback to ensure progress; the same confidence rule is used during exploration. The first accelerated forward retains the full native block and cached prefix; subsequent forwards omit positions to the right of the interval. After the interval is completed, exploration resumes for the remaining positions. Optional feature caching is disabled, while Fast dLLM v2's native prefix cache is retained.

Draft-and-review. We generate one diffusion draft per item at threshold 0.9 or 0.8. The reviewer receives the original question and complete saved draft, and is instructed to confirm a correct solution or identify the first incorrect step, repair it, and return a corrected answer. AR review uses the unchanged Qwen2.5-7B-Instruct weights with a fresh causal cache. Diffusion review uses the reconstructed weights that generated the draft, with confidence threshold 1. Both reviewers receive identical prompts and drafts, use greedy generation with a 1,024-token review budget, and run through native eager dispatch with SDPA. Every draft is reviewed once, with no additional retry.

Review scoring. We apply the GSM8K final-number extractor to the review response. When no number is present, a bare `CORRECT` or a digit-free explanation ending in `#### CORRECT` retains the draft answer, provided there is no explicit `INCORRECT` verdict. Other no-number responses are invalid. An explicit numerical answer takes precedence over the verdict.

Throughput measurement. We measure generation throughput as $\text{TPS} = \sum_i n_i / \sum_i t_i$, where n_i counts generated content tokens and t_i is synchronized generation time, including prefill, token selection, and cache operations. For draft-and-review, we count both stages: $\text{TPS} = (N_D + N_R)/(T_D + T_R)$, where N_D, N_R and T_D, T_R are their token totals and generation times. Each checkpoint–configuration pair is evaluated once on B200 hardware. The two-stage totals combine separately measured generation times and exclude loading, switching, tokenization, and persistence; they are not end-to-end answer latencies. Table 6 reports the corresponding per-question stage times and token counts.

D Additional Results

D.1 Full Results

Table 8 reports per-benchmark accuracy in both AR and diffusion modes for all four model pairs, complementing the diffusion-only results and mode-wise averages in the main text. Each cell reports AR / Diffusion, and Avg. is the unweighted mean across the four benchmarks. Dagger-marked AR scores are reused from the unchanged AR endpoint.

Table 8: **Accuracy (%) on four benchmarks.** Each cell reports AR / Diffusion. TA and RS denote Task Arithmetic and Representation Surgery, respectively; standalone TA uses $\beta = 0.6$. Bold and underlined values indicate the **best** and second-best scores for each model and mode. † denotes AR scores reused from the corresponding AR endpoint.

Method	Fast dLLM v2 (Qwen2.5-7B Instruct) GSM8K	MBPP	ARC-C	IFEval	Avg.	WeDLM-8B-Instruct (Qwen3-8B) GSM8K	MBPP	ARC-C	IFEval	Avg.
Endpoint	91.13 / 83.24	65.20 / 50.20	90.02 / 89.33	73.20 / 63.59	79.89 / 71.59	93.25 / 85.37	65.80 / 63.40	91.64 / 92.41	83.18 / 69.32	83.47 / 77.63
Parameter-combination										
Task Arithmetic	89.23 / 67.78	64.80 / 43.80	89.93 / 88.65	73.57 / 59.33	79.38 / 64.89	91.96 / 82.71	**70.00** / 39.80	92.66 / 86.95	78.56 / 70.24	83.30 / 69.92
TIES	88.32 / 78.85	62.40 / 44.60	89.93 / **89.85**	72.83 / 56.01	78.37 / 67.33	92.80 / 0.08	69.00 / 0.00	92.66 / 8.62	77.08 / 6.47	82.89 / 3.79
AdaMerging	90.45 / 13.87	64.20 / 7.40	90.44 / 83.11	74.12 / 38.63	79.80 / 35.75	93.10 / 0.00	68.60 / 0.00	91.98 / 0.00	82.07 / 0.00	83.94 / 0.00
AdaMerging++	89.76 / 0.08	63.80 / 1.40	90.61 / 77.22	72.09 / 18.85	79.07 / 24.39	94.01 / 0.00	68.60 / 0.00	92.58 / 0.00	82.99 / 0.00	**84.55** / 0.00
DARE + TA	90.30 / 46.55	64.20 / 35.40	**90.78** / 87.88	74.49 / 55.64	79.94 / 56.37	94.16 / 0.00	69.00 / 0.00	92.66 / 0.00	80.22 / 0.18	84.01 / 0.05
Activation-based methods										
RegMean	89.31 / 46.32	65.00 / 40.20	90.36 / 86.69	73.94 / 56.19	79.65 / 57.35	94.16 / 75.51	68.00 / 19.40	91.81 / 76.28	81.15 / 49.35	83.78 / 55.14
RS	**91.13**†/ 19.41	**65.20**†/ 30.40	90.02†/ 87.29	73.20†/ 50.46	79.89†/ 46.89	93.25†/ 0.99	65.80†/ 0.00	91.64†/ 77.73	**83.18**†/ 20.15	83.47†/ 24.72
AIM + TA	89.39 / 23.81	65.00 / 25.00	90.70 / 86.77	**75.42** / 48.80	**80.13** / 46.10	**94.31** / 0.00	68.60 / 0.00	**92.75** / 0.00	82.26 / 0.00	84.48 / 0.00
Task-specific representation										
T-Switch	**91.13**†/ 75.89	**65.20**†/ 48.60	90.02†/ 88.99	73.20†/ 60.44	79.89†/ 68.48	93.25†/ 89.08	65.80†/ **65.20**	91.64†/ 92.41	**83.18**†/ 72.64	83.47†/ 79.83
MASK-ON	**91.13**†/ **84.99**	**65.20**†/ **51.80**	90.02†/ 89.16	73.20†/ **64.14**	79.89†/ **72.52**	93.25†/ **90.52**	65.80†/ 64.40	91.64†/ **92.83**	**83.18**†/ **74.86**	83.47†/ **80.65**
Method	**SDAR-8B-Chat** (Qwen3-8B) GSM8K	MBPP	ARC-C	IFEval	Avg.	**LLaDA2.0-mini** (Ling-mini-2.0) GSM8K	MBPP	ARC-C	IFEval	Avg.
Endpoint	93.40 / 91.28	66.80 / 65.60	90.70 / 91.89	79.85 / 61.92	82.69 / 77.67	92.87 / 91.81	67.80 / 66.80	91.21 / 89.76	72.27 / 80.59	81.04 / 82.24
Parameter-combination										
Task Arithmetic	93.10 / 86.73	70.80 / 64.00	91.89 / **91.89**	68.39 / **68.58**	81.05 / 77.80	76.35 / 52.54	65.40 / 5.80	**91.38** / 89.76	49.35 / 45.10	70.62 / 48.30
TIES	80.82 / 2.05	67.20 / 34.60	**92.32** / 84.39	52.50 / 29.57	73.21 / 37.65	89.46 / 0.76	60.00 / 0.00	88.14 / 63.23	53.23 / 21.26	72.71 / 21.31
AdaMerging	93.71 / 0.00	70.60 / 0.00	91.98 / 0.00	81.33 / 0.00	**84.41** / 0.00	82.71 / 0.08	66.00 / 0.00	91.21 / 25.60	54.71 / 11.28	73.66 / 9.24
AdaMerging++	**93.93** / 0.00	69.40 / 0.00	91.81 / 0.00	**82.26** / 0.00	84.35 / 0.00	**92.95** / 0.00	**69.60** / 0.00	91.04 / 0.17	**76.71** / 2.77	**82.58** / 0.74
DARE + TA	93.25 / 27.52	70.20 / 29.80	91.72 / 87.03	75.05 / 57.49	82.56 / 50.46	82.11 / 3.94	62.00 / 0.20	88.23 / 80.63	47.69 / 25.32	70.01 / 27.52
Activation-based methods										
RegMean	92.95 / 84.61	**71.40** / 52.20	91.47 / 87.46	78.93 / 67.10	83.69 / 72.84	90.14 / 1.06	**69.60** / 0.00	91.13 / 88.23	48.98 / 32.35	74.96 / 30.41
RS	93.40†/ 1.67	66.80†/ 4.40	90.70†/ 82.00	79.85†/ 36.78	82.69†/ 31.21	92.87†/ 0.99	67.80†/ 0.00	91.21†/ 18.26	72.27†/ 24.95	81.04†/ 11.05
AIM + TA	93.63 / 1.14	71.20 / 0.40	92.15 / 2.39	79.48 / 19.04	84.12 / 5.74	92.49 / 1.74	69.00 / 0.00	91.30 / 74.83	61.74 / 23.48	78.63 / 25.01
Task-specific representation										
T-Switch	93.40†/ 72.86	66.80†/ 55.80	90.70†/ 91.72	79.85†/ 44.36	82.69†/ 66.19	92.87†/ 70.05	67.80†/ 32.20	91.21†/ 87.46	72.27†/ 39.37	81.04†/ 57.27
MASK-ON	93.40†/ **90.45**	66.80†/ **67.40**	90.70†/ **91.89**	79.85†/ 65.06	82.69†/ **78.70**	92.87†/ **85.37**	67.80†/ **41.60**	91.21†/ **89.93**	72.27†/ **55.82**	81.04†/ **68.18**

D.2 TASK ARITHMETIC COEFFICIENT SENSITIVITY UNDER GSM8K

We evaluate $W_\beta = W_A + \beta(W_D - W_A)$ at $\beta \in \{0.2, 0.4, 0.6, 0.8\}$ on all 1,319 GSM8K test items, using the same mode-specific evaluation protocol as the main results. Figure 3 compares both generation modes with their native endpoints. The main tables report the common $\beta = 0.6$ setting; this sweep does not establish it as an optimal coefficient across models or benchmarks.

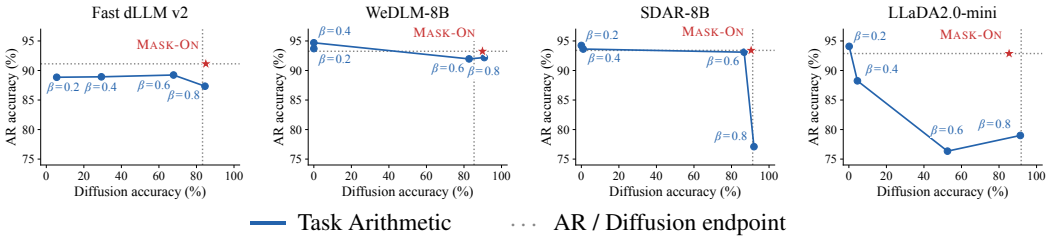

Figure 3: **GSM8K accuracy tradeoffs under Task Arithmetic.** Blue points connect the tested coefficients; gray dotted lines mark native endpoint accuracies. Red stars combine measured MASK-ON diffusion scores with reused AR endpoint scores, not AR evaluation of its reconstructed diffusion weights. The AR axes are cropped to 74–97% for readability.

Across these four models, no tested coefficient matches both endpoint accuracies simultaneously. Increasing β generally improves diffusion recovery but can reduce AR accuracy: for SDAR, moving from 0.6 to 0.8 increases diffusion accuracy from 86.73% to 91.96% while lowering AR accuracy from 93.10% to 77.10%. Some coefficients exceed a native endpoint in one mode, so the sweep supports a coefficient-dependent tradeoff rather than uniform degradation or universal dominance by MASK-ON.

D.3 Activation Collection Ablation

We examine whether collecting activations throughout denoising improves calibration over a single forward pass on completed prompt–response sequences. Within each model, the six variants in Table 9 share the same 128 calibration prompts, 1,536 fitting rows per projection, equal domain weights, and projection-output least-squares objective. We fix $\alpha = 0.40/0.40/0.30$ for Fast dLLM v2, WeDLM, and SDAR, respectively, retaining the same support/sign codes and native embeddings and output heads while refitting only output-channel scales. All variants use matched benchmark items, generation settings, and scorers.

Collection variants. During native generation, we compare four sampling policies over observations of token positions under distinct input contexts. The first pools these observations without balancing their context or generation progress. The second samples only from initial prompt processing and fully resolved generated context; the third samples only positions that currently contain a mask. The fourth assigns equal weight to prompt processing, the region currently being generated, and fully resolved generated context, and further balances three progress intervals within active generation.

For Fast and SDAR, fully resolved context is observed when a generated block is processed without any remaining masks. For WeDLM, it consists of the contiguous generated prefix about to be committed to the cache, excluding the remaining active window even if some positions in that window have already been filled. Fast and SDAR divide generation progress by the fraction of masks remaining relative to the first observed masked input of the active block, using boundaries at 0.75 and 0.25. WeDLM instead divides progress by the fraction of the completed response already committed to the prefix, using boundaries at 0.25 and 0.75. Positions read only from the cache supply no new activation observations.

The two completed-sequence variants instead process either dataset-provided or endpoint-generated responses together with their prompts in one unmasked forward pass, sampling both prompt and response positions as described in Appendix C.3.

Table 9: **Activation collection at fixed pruning levels.** Values are mean diffusion-mode accuracy (%) over GSM8K, MBPP, ARC-C, and IFEval. The pruning levels are 0.40, 0.40, and 0.30 for Fast, WeDLM, and SDAR. The highlighted row uses the adopted collection procedure; WeDLM uses $\alpha = 0.35$ in the final main results.

Activation collection	Fast dLLM v2	WeDLM-8B	SDAR-8B
Generation: pooled observations	72.26	78.81	78.88
Generation: prompt and resolved context	72.54	80.05	78.91
Generation: masked positions only	72.25	2.94	77.70
Generation: balanced contexts and progress	72.76	79.72	78.71
Completed dataset responses	72.20	80.29	78.38
Completed endpoint responses	72.52	80.09	78.70

A single completed-sequence pass provides competitive calibration. Completed endpoint responses achieve averages within 0.37 percentage points of collection balanced across contexts and generation progress on all three models, with no consistent advantage from sampling intermediate denoising steps (Table 9). We therefore adopt the simpler completed-sequence procedure, without collecting intermediate denoising activations. This does not make the sampled population irrelevant: restricting collection to masked positions severely degrades WeDLM performance.

Endpoint-generated responses improve averages under the final recipe. The fixed-pruning comparison above shows mixed effects from replacing dataset responses, including a small decrease on WeDLM. Under the final recipe, however, the matched-source comparison in Table 5 improves every model's four-benchmark average, from 74.21% to 75.01% overall. Prompts, row budgets, domain weights, and fitting objectives are matched; independent searches select the same pruning levels for both sources ($0.40/0.35/0.30/0.45$ for Fast/WeDLM/SDAR/LLaDA2.0-mini). These gains support using the endpoint's own responses as calibration content, rather than attributing the

Table 10: **Model-wise reconstruction ablation.** Each model column reports its unweighted four-benchmark diffusion accuracy (%); Avg. averages the four models. Rows marked + add and ↔ replace one component, at $\alpha = 0.5$ until pruning selection. Remaining tensors exclude calibrated linear projections and vocabulary tensors.

Reconstruction	Fast	WeDLM	SDAR	LLaDA	Avg.
T-Switch	68.48	79.83	66.19	57.27	67.94
+ weight-LS for remaining tensors	69.65	79.70	66.30	57.03	68.17
+ channel-wise RMS rescaling	72.68	79.68	70.97	68.24	72.89
↔ channel-wise weight-LS	72.70	79.42	66.76	60.77	69.91
↔ output-LS, dataset responses	72.29	79.56	77.78	66.23	73.97
↔ endpoint-generated responses	72.81	80.45	77.74	65.95	74.24
+ pruning selection (MASK-ON)	72.52	80.65	78.70	68.18	75.01

improvement to different pruning choices or coverage of intermediate denoising steps. They do not establish improvement on every task or across independently sampled calibration sets.

D.4 RECONSTRUCTION BREAKDOWN

Table 10 complements the benchmark-wise ablation in Section 5.1 with model-wise averages. Both tables report the same configurations and scored predictions. The aggregate gain from dataset-response output-LS over channel-wise RMS is concentrated in SDAR; Fast dLLM v2, WeDLM, and LLaDA2.0-mini do not improve in this contrast. The effects of reconstruction choices therefore depend on the model and preceding configuration, even when their benchmark-wise averages improve.

The channel-wise weight-LS control retains $\alpha = 0.5$, the same support/sign codes, native vocabulary tensors, and weight-LS treatment of the remaining backbone tensors. For each output channel r, it sets

$$\lambda_{\ell,r} = \max\left(0, \frac{\langle \Delta W_{\ell,r:}, S_{\ell,r:} \rangle}{\|S_{\ell,r:}\|_2^2}\right),$$

with zero scale for empty support. This changes the RMS fitting rule to least squares without using activations or changing scale granularity, allowing the subsequent output-LS comparison to isolate activation weighting at fixed support.

D.5 STORAGE ACCOUNTING

Measurement scope. Table 3 reports the bytes required for the additional diffusion representation, divided by the AR checkpoint's weight-file bytes. We sum measured serialized component lengths and native-tensor byte counts; these values are not the measured sizes of newly repacked standalone checkpoints. For both T-Switch and MASK-ON, we count the required backbone codes and scales, the uncompressed diffusion token embeddings and output head, and metadata. Unused embedding/head code and scale ranges are excluded for both methods, since the native vocabulary tensors are retained instead. Metadata is counted in full, including descriptions of these excluded ranges. Dense evaluation checkpoints, calibration caches, tokenizer files, and model configuration files are not included. The denominator is the sum of the AR safetensors shard file sizes, including their headers; native-tensor contributions are measured from safetensors data offsets in their stored dtype. An independently exported compact bundle would require its own container metadata and size measurement.

Packed codes and scales. Both methods use the same grouped bit-packing format, with groups of four flattened entries. Each group stores one presence bit; only nonempty groups store four within-group support bits, and only retained entries store sign bits. The final group is zero-padded if necessary, and each of the three bitstreams is padded separately to a byte boundary for each tensor. No entropy compression is applied. For projection ℓ, let $n_\ell = d_{\mathrm{out},\ell} d_{\mathrm{in},\ell}$, $G_\ell = \lceil n_\ell/4 \rceil$, H_ℓ be the

Table 11: **Component-wise storage accounting.** All sizes are absolute bytes, except the final percentage rows. Components sum to the required additional representation size; the shared AR denominator is reported separately. Non-projection tensors include their codes and scales.

Component	Fast dLLM v2	WeDLM-8B	SDAR-8B	LLaDA2.0-mini
AR weight files	15,231,271,888	16,381,516,776	16,381,516,776	32,513,136,008
T-Switch				
Projection group bits	203,915,264	217,055,232	217,055,232	487,866,368
Projection support bits	752,363,987	802,023,583	803,274,825	1,822,641,374
Projection sign bits	403,957,896	432,825,269	432,940,352	972,405,779
Projection scales	784	1,008	1,008	58,844
Non-projection tensors	28,345	56,367	71,932	20,967
Native embeddings	1,089,994,752	1,244,659,712	1,244,659,712	643,825,664
Native output head	1,089,994,752	1,244,659,712	1,244,659,712	643,825,664
Metadata	246,280	291,496	291,763	11,022,963
Total additional bytes	3,540,502,060	3,941,572,379	3,942,954,536	4,581,667,623
Additional storage (%)	23.24	24.06	24.07	14.09
MASK-ON				
Projection group bits	203,915,264	217,055,232	217,055,232	487,866,368
Projection support bits	783,762,514	847,847,177	856,692,347	1,865,480,064
Projection sign bits	478,538,388	561,094,260	604,767,691	1,069,620,197
Projection scales	5,562,368	5,603,328	5,603,328	60,480,512
Non-projection tensors	31,063	58,560	78,912	21,644
Native embeddings	1,089,994,752	1,244,659,712	1,244,659,712	643,825,664
Native output head	1,089,994,752	1,244,659,712	1,244,659,712	643,825,664
Metadata	286,504	339,756	340,214	13,101,895
Total additional bytes	3,652,085,605	4,121,317,737	4,173,857,148	4,784,222,008
Additional storage (%)	23.98	25.16	25.48	14.71

number of nonempty groups, and K_ℓ the number of retained entries. Its code size in bytes is

$$B_{\mathrm{code},\ell} = \left\lceil \frac{G_\ell}{8} \right\rceil + \left\lceil \frac{4H_\ell}{8} \right\rceil + \left\lceil \frac{K_\ell}{8} \right\rceil . \tag{30}$$

A fixed-width two-bit encoding would instead use $\lceil n_\ell/4 \rceil$ bytes independently of pruning; the implemented format varies with H_ℓ and K_ℓ. Scales are stored in FP32, requiring four bytes each. Thus, the total additional representation size is

$$B_{\mathrm{add}} = \sum_\ell B_{\mathrm{code},\ell} + 4N_\lambda + B_{\mathrm{other}} + B_{\mathrm{native}} + B_{\mathrm{meta}}, \qquad p_{\mathrm{add}} = 100\,\frac{B_{\mathrm{add}}}{B_{\mathrm{AR}}}. \tag{31}$$

Here $N_\lambda = \sum_\ell d_{\mathrm{out},\ell}$ for MASK-ON, whereas T-Switch uses one scale per projection. B_{other} includes the packed codes and tensor-wise FP32 scales for non-projection backbone tensors; B_{native} comprises the retained embeddings and output head; and B_{meta} is the metadata size. B_{AR} is the AR weight-file denominator, and p_{add} is the percentage reported in Table 3.

Sources of the storage difference. Table 11 provides the absolute-byte decomposition for both methods under the same accounting scope. T-Switch uses projection pruning level 0.50 for all four models, while MASK-ON uses $0.40/0.35/0.30/0.45$ for Fast/WeDLM/SDAR/LLaDA2.0-mini. The additional projection scales account for only 0.034–0.186 percentage points of AR storage. Most of the remaining difference comes from the longer support and sign streams at the selected pruning levels, rather than scale granularity. The comparison therefore measures the representations used for the reported results, not performance at an identical storage budget.

D.6 ANSWER TRANSITIONS IN DRAFT-AND-REVIEW

We examine how review changes final-answer correctness in the Fast dLLM v2/GSM8K experiment from Section 5.2. At each draft threshold, the unchanged AR checkpoint and the reconstructed

Table 12: **Final-answer transitions after draft-and-review on GSM8K.** Each transition cell reports the item count and the percentage within its initially correct or incorrect draft group. Thresholds refer to draft generation; the diffusion reviewer uses threshold 1 at both draft thresholds. Accuracy changes are percentage points relative to the same drafts.

	Draft threshold 0.9		**Draft threshold** 0.8	
Draft → reviewed answer	**AR review**	**Diffusion review**	**AR review**	**Diffusion review**
Correct → correct	1,105 (98.93%)	1,111 (99.46%)	1,088 (98.82%)	1,098 (99.73%)
Correct → incorrect	12 (1.07%)	6 (0.54%)	13 (1.18%)	3 (0.27%)
Incorrect → correct	50 (24.75%)	19 (9.41%)	56 (25.69%)	15 (6.88%)
Incorrect → incorrect	152 (75.25%)	183 (90.59%)	162 (74.31%)	203 (93.12%)
Final accuracy (%)	87.57	85.67	86.73	84.38
Accuracy change (pp)	$+2.88$	$+0.99$	$+3.26$	$+0.91$

diffusion model review the same 1,319 saved drafts with identical instructions. The initial drafts contain 1,117 correct and 202 incorrect answers at threshold 0.9, and 1,101 correct and 218 incorrect answers at threshold 0.8. Table 12 partitions each panel into four correctness transitions. Transition percentages are conditional on the draft's initial correctness; final accuracy and its change use all 1,319 items. Correctness refers to the extracted final answer under the scoring rule in Appendix C.4.

AR review corrects approximately 25–26% of initially incorrect drafts while changing about 1.1–1.2% of correct drafts into incorrect answers. Its larger net gains reflect more successful corrections of wrong drafts, despite slightly more incorrect revisions of correct ones than diffusion review.

As reported in Table 6, AR reviews generate shorter responses on average and require less generation time than diffusion reviews in this comparison. The measurements include each stage's prefill, token selection, and cache operations but exclude model loading, mode switching, and persistence.

D.7 A Qualitative Example of Draft-and-Review

Figure 4 shows a selected wrong-to-correct example from the draft-and-review evaluation in Section 5.2. MASK-ON generates the diffusion draft at confidence threshold 0.9, and the unchanged Qwen2.5-7B-Instruct checkpoint reviews the question and complete draft with a fresh causal cache. The example illustrates a wrong-to-correct transition counted in Table 12.

Mathematical Reasoning: Diffusion Draft and AR Review

User: A football team has 105 members. There are twice as many players on the offense as there is on the defense. There is half the number of players on the special teams as there is on the defense. How many players are on the defense?

Diffusion: Let's use the number of players on the defense as "x".

We know that there are twice as many players on the offense as there is on the defense, so there are 2x players on the offense.

We also know that there is half the number of players on the special teams as there is on the defense, so there are x/2 players on the special teams.

The total number of players on the team is 105, so we can write an equation:

$$x + 2x + x/2 = 105$$

Combining like terms, we get:

$$3x + x/2 = 105$$

To solve for x, we can multiply both sides by 2:

$$66x + x = 210$$

$$67x = 210$$

$$x = 210/67$$

$$x = 3$$

So there are 3 players on the defense.

Final answer: $\boxed{3}$

User: Check the proposed solution to the mathematics problem below. If both its reasoning and final answer are correct, output exactly CORRECT. Otherwise output INCORRECT on the first line. Then identify the first incorrect step, repair it, and work through the remaining solution step by step. Check the arithmetic and answer the original question. Finish with exactly one final line in the format `#### <number>`. Do not merely guess a replacement answer. Do not include another verdict or another `####` marker in the explanation.

AR review:

INCORRECT

The equation should be combined correctly and solved properly. The equation $x + 2x + \frac{x}{2} = 105$ simplifies to $3x + \frac{x}{2} = 105$. To eliminate the fraction, multiply every term by 2:

$$2(3x) + 2\left(\frac{x}{2}\right) = 2(105)$$

$$6x + x = 210$$

$$7x = 210$$

$$x = \frac{210}{7}$$

$$x = 30$$

So there are 30 players on the defense.

30

Figure 4: Selected correction on GSM8K test item `gsm8k-314`: Fast dLLM v2 with MASK-ON drafts 3 after expanding $2(3x) + x$ as $66x + x$; original Qwen2.5-7B-Instruct repairs this step and answers 30, matching the reference. Red emphasis is added. The review instruction is shown separately for readability; the actual reviewer input also contains the complete question and saved draft.

E LIMITATION AND FUTURE WORK

Dependence on paired checkpoints. MASK-ON requires an existing diffusion endpoint and a corresponding AR anchor with aligned parameters. It compresses the weight changes between these checkpoints rather than equipping an AR model with diffusion generation on its own. Developing masks or compact updates that enable diffusion generation from an arbitrary AR checkpoint, without requiring an already adapted diffusion endpoint, is a promising direction for future work.

Model-dependent reconstruction. The calibration objective is a fixed-activation reconstruction proxy, not a guarantee of downstream generation quality. The contribution of each calibration choice varies across models, and completed-sequence activations do not explicitly cover intermediate denoising states. Although MASK-ON remains below the LLaDA2.0-mini endpoint on MBPP (41.60% versus 66.80%) and IFEval (55.82% versus 80.59%), it leads the compared merging methods and improves four-benchmark average accuracy over T-Switch from 57.27% to 68.18%. Improving expert-specific calibration and understanding which parameter changes must be retained are promising directions; our experiments do not isolate the cause of this model's losses.

Scope of hybrid cooperation. We focus on a simple, training-free *draft-and-review* composition in which diffusion generates a candidate response and AR reviews it. This demonstrates a use of the two modes' complementary behaviors, but does not explore more tightly integrated forms of cooperation. Future work could investigate speculative decoding and other approaches that exploit diffusion's parallelism, such as exploring multiple candidate solutions before AR-based assessment and refinement. Identifying which tasks benefit more from AR or diffusion generation and developing request-level routing policies based on task characteristics and quality–latency requirements are also promising directions for future work.

Checkpoint storage versus serving efficiency. Our storage results account for the required additional representation bytes relative to the AR checkpoint's weight files. The evaluated implementation materializes dense weights, and the draft-and-review implementation retains two dense states rather than demonstrating compact-resident inference. Reported generation throughput and summed stage times exclude model loading and mode switching and do not establish an end-to-end speedup. Efficient execution directly from the compact representation, low-overhead mode switching, and end-to-end evaluation of latency and peak memory are promising directions for future work.