LATENT DEPTH RECURSION REVEALS POST-TRAINING DYNAMICS IN MULTIMODAL LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent work has shown that frozen large language models (LLMs) can correct their own predictions by re-executing a block of their own layers. Prior methods exploit this recursive structure at inference time, either by searching for the right block to repeat or by training a router to select one. All of these evaluations target pretrained or instruction-tuned LLMs. Every deployed model in practice, however, undergoes a longer post-training pipeline that adds multimodal adaptation and reinforcement learning (RL). In this work, we investigate whether the recursive structure that these methods rely on survives this pipeline. To this end, we reframe the recursive capability as a measurable budget: the fraction of instances that the default forward pass gets wrong but that some recursion route gets right. We compute this budget by exhaustively evaluating every route in a small, closed recursion family. Tracking it across the training stages of several LLM and multimodal LLM (MLLM) families, we find that the budget is large in pretrained models and shrinks consistently as post-training proceeds. Both supervised fine-tuning (SFT) and RL absorb the recursive structure into the default forward pass, and their updates align with where recursion was previously helpful. As a result, little budget remains for test-time routing to exploit, which explains why existing methods fail on fully post-trained models. Instead of recovering recursion at test time, we propose recursion-aware post-training, which deliberately spends the remaining budget by adding a fixed per-category recursion route to the training forward pass. At the same parameter count, our scheme outperforms standard GSPO post-training.

1 INTRODUCTION

Re-executing a contiguous block of layers in a frozen large language model (LLM) can turn an incorrect answer into a correct one, without requiring new parameters or extra output tokens (Li et al., 2026; Lys et al., 2026; Chen et al., 2026a). Several methods exploit this latent recursive structure at inference time, either by searching for the right block to repeat (Chen et al., 2026a) or by training a router to select one (Heakl et al., 2026; Li et al., 2026), and report large accuracy gains on LLMs.

These evaluations target pretrained or instruction-tuned LLMs, whereas models in practice go through a longer post-training pipeline of multimodal adaptation, supervised fine-tuning (SFT), and reinforcement learning (RL). This gap matters because recursion and post-training may act on the same instances. Concretely, recursion reaches correct answers by changing the computation with the weights fixed, while post-training changes the weights so that the default forward pass answers better. If post-training already fixes the same instances that recursion fixes, it is likely *internalize* the recursive structure, and answers that once needed a repeated block become default answers. We therefore ask whether the recursive structure survives the full post-training pipeline or has already been absorbed into the default forward pass.

We measure this structure as the *recursion budget*, the fraction of instances that the default forward pass gets wrong but at least one route of a small, closed recursion family gets right. Each route repeats one contiguous block once, so we can run every route on every instance and track the budget exactly across checkpoints. We study multimodal LLMs (MLLMs) because they sit at the end of the post-training pipeline and their benchmarks span perception, reasoning, and knowledge.

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.

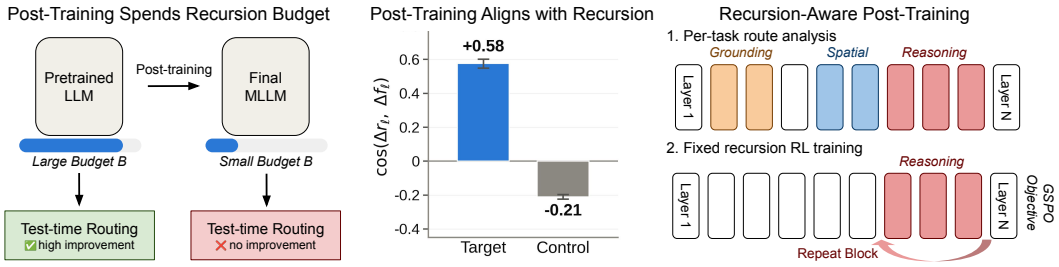


Figure 1: **Left:** Post-training spends the recursion budget. A pretrained LLM holds a large budget B that test-time routing can exploit, while the fully post-trained MLLM holds only a small budget, so routing has nothing left to gain. **Middle:** The post-training logit update $\Delta \boldsymbol{f}_\ell$ aligns with the update from a good recursion route $\Delta \boldsymbol{r}_\ell$. This alignment is strong for the target set (instances post-training newly learns) and negative for the control set (instances already answered correctly). **Right:** We propose recursion-aware post-training. Per-task route analysis finds the most useful route for each task, and the selected route is applied as a fixed loop during GSPO training with no extra parameter.

We find that post-training steadily spends this budget. Along released training checkpoints, the budget drops at each post-training stage, most sharply during instruction tuning and multimodal adaptation (Figure 1, Left). Although the budget shrinks, it does not vanish. Across 21 checkpoints from 8 MLLM families, the instance-level oracle, which picks the best route per instance, still gains $+23.5$ to $+60.2$ points over the default path. However, no fixed route gains more than $+2.7$ points, even when chosen with the test labels, so the remaining budget is overwhelmingly instance-specific. In addition, we call an instance *out of reach* when neither the default forward pass nor any route of the recursion family answers it correctly. In the RL stages we measure, RL never turns an out-of-reach instance into a correct one, whereas SFT does. This agrees with studies finding that RL mainly sharpens what a model can already solve (Chen et al., 2026b; Matsutani et al., 2026). While other studies argue that prolonged or task-targeted RL can go further (Liu et al., 2026; Shen et al., 2026), our claim is limited to standard RL stages on general benchmarks.

We then compare where and in which direction post-training and recursion change the model, contrasting a target set of instances that a transition newly answers correctly with a control set that both checkpoints already answer correctly. On the target set, the logit shift from post-training aligns with the shift from the best recursion route at the earlier checkpoint (Figure 1, Middle), and hidden-state shifts behave the same way. On the control set, this alignment is much weaker. Thus, post-training moves the model toward what recursion provided specifically on the instances it newly learns.

These results predict that test-time routing has little left to exploit on a fully post-trained model, and three prior methods (Chen et al., 2026a; Li et al., 2026; Heakl et al., 2026) indeed give at most marginal gains in the MLLM setting (§6). Test-time methods must find a helpful route for each instance without changing the model weights, which is hard when such routes are rare, but training can instead directly teach the model to benefit from a fixed route. We therefore propose *recursion-aware post-training*, which adds a fixed per-category recursion route, chosen from a utility map on a small training subset, to the training forward pass (Figure 1, Right). It uses the same trainable parameters as standard fine-tuning at 1.06 to $1.23\times$ the step time, and on InternVL3.5-2B it outperforms standard GSPO on five of six task categories, by up to $+3.1$ points.

Our contributions are as follows. (1) We define a measurable recursion budget by running every route of a small, closed family on every instance. (2) We find that post-training consumes this budget and that the released RL stages never solve an out-of-reach instance. (3) We demonstrate that post-training updates align with recursion utility on newly learned instances. (4) We confirm that test-time routers fail on fully post-trained MLLMs and propose recursion-aware post-training as a training-time alternative.

2 Preliminaries

Routes and the recursion family. Let $f_{\boldsymbol{\theta}}$ be a frozen MLLM with a vision encoder, an MLP connector, and an LLM backbone of L layers. We index layers from 0, so the backbone layers are

$0, \ldots, L{-}1$. This 0-based convention is used for every route, table, and figure in the paper. A *route* is an ordered sequence of LLM layer indices that defines a forward pass, and $r_0 = (0, \ldots, L{-}1)$ is the default forward pass. For $0 \le s \le e \le L{-}1$, the *recursion route*

$$r_{s,e} \;:=\; \big(0, \ldots, e,\; \underbrace{s, \ldots, e}_{\text{repeated block}},\; e{+}1, \ldots, L{-}1\big) \tag{1}$$

re-executes layers $[s, e]$ once before continuing to layer $e{+}1$. The full *recursion family* is

$$\mathcal{R} \;:=\; \{r_0\} \cup \big\{r_{s,e} : 0 \le s \le e \le L{-}1\big\}, \qquad |\mathcal{R}| = \tfrac{L(L+1)}{2} + 1. \tag{2}$$

This extends the block-level recursion (BLR) family of Sakalyan et al. (2026). Their family requires $s < e$, while ours also includes single-layer routes ($s = e$). Any non-default route leaves the weights unchanged and gives a forward pass with $L{+}(e{-}s{+}1)$ effective layers. Because $|\mathcal{R}| = O(L^2)$, we run every route on every evaluation instance exactly, with no search or sampling that prior methods require (Li et al., 2026; Heakl et al., 2026).

Instance categories and the recursion budget. Fix a checkpoint $\boldsymbol{\theta}$ and dataset $\mathcal{D}$, and let $\mathrm{acc}(\boldsymbol{x}, r) \in \{0, 1\}$ record whether route r answers instance $\boldsymbol{x}$ correctly. Every instance falls into exactly one of three categories. It is *always-correct* if $\mathrm{acc}(\boldsymbol{x}, r_0) = 1$, *in budget* if $\mathrm{acc}(\boldsymbol{x}, r_0) = 0$ and some $r \neq r_0$ in $\mathcal{R}$ has $\mathrm{acc}(\boldsymbol{x}, r) = 1$, and *out of reach* if $\mathrm{acc}(\boldsymbol{x}, r) = 0$ for every $r \in \mathcal{R}$. The *recursion budget* is the fraction of in-budget instances,

$$B \;:=\; \frac{\big|\{\boldsymbol{x} \in \mathcal{D} : \boldsymbol{x} \text{ is in budget}\}\big|}{|\mathcal{D}|}. \tag{3}$$

B is the main quantity we track across training stages (§4). It is the largest accuracy gain any route-selection method could reach with perfect per-instance knowledge. We also track how instances move between the three categories across consecutive checkpoints, with a focus on whether the out-of-reach $\rightarrow$ always-correct transition ever happens at a given post-training stage.

An in-budget instance may have several correct routes. We define its *best recursion route* as the correct route in $\mathcal{R}$ with the highest average log-likelihood of the ground-truth answer tokens. We use this route in the analyses of §5.

Recursion fix density. B counts how many instances some route can rescue. It does not say how easily a method without ground-truth labels can find such a route. We measure that accessibility with the *recursion fix density*,

$$\rho_{\mathrm{fix}} \;:=\; \mathop{\mathbb{E}}_{\boldsymbol{x}\,:\,\mathrm{acc}(\boldsymbol{x}, r_0)=0} \frac{\big|\{r \neq r_0 : \mathrm{acc}(\boldsymbol{x}, r) = 1\}\big|}{|\mathcal{R}| - 1}, \tag{4}$$

the average fraction of non-default routes that fix an instance the default route gets wrong. A high ρ_{fix} gives a router a strong training signal. A low ρ_{fix} means helpful routes are sparse, so a non-default route is unlikely to fix the error and may instead break generation.

Instance-level and task-level oracles. Two oracles bound the accuracy reachable within $\mathcal{R}$. The *task-level oracle*

$$\mathrm{Oracle}_{\mathrm{task}} \;:=\; \max_{r \neq r_0} \frac{1}{|\mathcal{D}|} \sum_{\boldsymbol{x} \in \mathcal{D}} \mathrm{acc}(\boldsymbol{x}, r) \tag{5}$$

is the best accuracy of any single non-default route applied to all instances. We exclude r_0, so $\mathrm{Oracle}_{\mathrm{task}}$ can fall below baseline. The *instance-level oracle*

$$\mathrm{Oracle}_{\mathrm{inst}} \;:=\; \frac{1}{|\mathcal{D}|} \sum_{\boldsymbol{x} \in \mathcal{D}} \max_{r \in \mathcal{R}} \mathrm{acc}(\boldsymbol{x}, r) \tag{6}$$

is the accuracy of picking the best route per instance. It satisfies $\mathrm{Oracle}_{\mathrm{inst}} = \frac{1}{|\mathcal{D}|} \sum_{\boldsymbol{x}} \mathrm{acc}(\boldsymbol{x}, r_0) + B$, so the oracle gap is exactly the budget. The difference $\mathrm{Oracle}_{\mathrm{inst}} - \mathrm{Oracle}_{\mathrm{task}}$ is how much a single global route gives up relative to per-instance selection.

Decoding and scoring. We decode every route on every instance greedily with a 32-token budget, raised to 512 tokens for MIA-Bench (Qian et al., 2025). A rule-based gibberish detector runs before

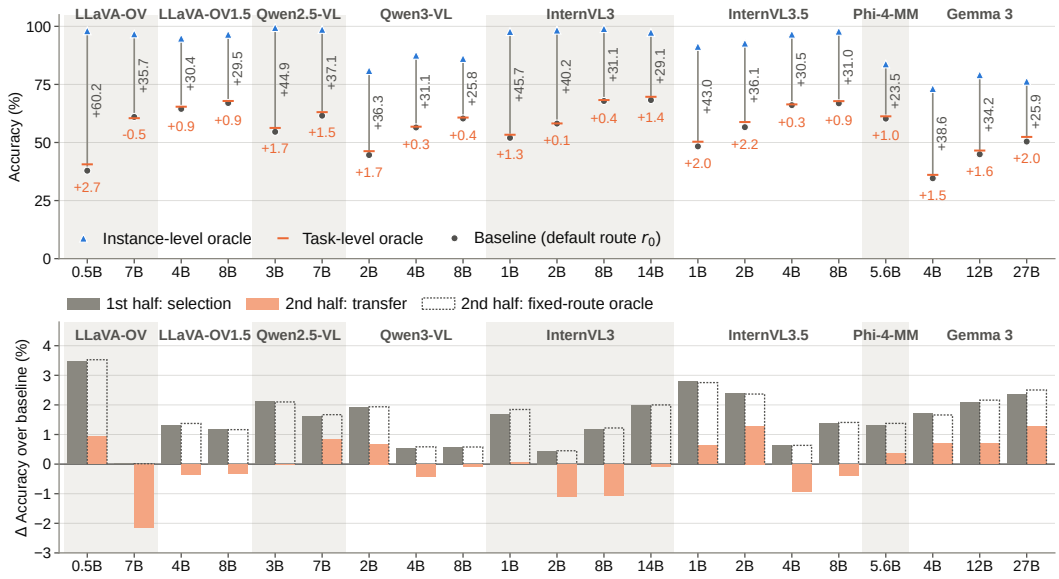


Figure 2: **The recursion budget is universal, but fixed selection cannot reach it. Top:** Instance-level and task-level oracle accuracy on MMStar across 21 checkpoints from 8 families (0.5B to 27B). Numbers above each bar are the oracle gains over baseline. **Bottom:** A split-half test of transfer. We select the best fixed route on one half of MMStar (selection) and apply it to the other half (transfer).

scoring so that malformed strings cannot accidentally match a correct answer. Full details are in Appendix A.1.

Models and benchmarks. We analyze 21 checkpoints across 8 MLLM families (LLaVA-OneVision (Li et al., 2024a), LLaVA-OneVision-1.5 (An et al., 2025), Qwen2.5-VL (Team, 2025b), Qwen3-VL (Bai et al., 2025), InternVL3 (Zhu et al., 2025a), InternVL3.5 (Wang et al., 2025a), Phi-4-Multimodal (Abouelenin et al., 2025), Gemma 3 (Team, 2025a)) at scales from 0.5B to 27B. For the post-training dynamics analysis (§4 and §5), we use the publicly released intermediate checkpoints of the Qwen3 LLM and InternVL3.5 families, which span the full pipeline from LLM pre-training through MLLM RL. Our primary benchmarks are MMStar (Chen et al., 2024) (1,500 image instances spanning perception, reasoning, and knowledge) and ARC-Challenge (Clark et al., 2018) (1,172 text-only multiple-choice science questions), both small enough for exhaustive route evaluation. We also report on Video-MME (Fu et al., 2025a), EgoSchema (Mangalam et al., 2023), and MIA-Bench (Qian et al., 2025) to confirm generalization (Appendix A.2).

3 The Recursion Budget Is Universal, but Inaccessible

The budget exists in every model we check. Figure 2 (Top) evaluates both oracles on MMStar across all 21 checkpoints. The instance-level oracle gain, which equals B by construction, ranges from $+23.5$ to $+60.2$ points. Even after several rounds of post-training, between a quarter and more than half of all instances can be answered correctly by some route in $\mathcal{R}$ while the default route fails. The ordering baseline $<$ $\text{Oracle}_{\text{task}}$ $<$ $\text{Oracle}_{\text{inst}}$ holds on every checkpoint except one (LLaVA-OV 7B). The same pattern appears on video and free-form generation benchmarks (Appendix B.1). At matched cardinality $N = |\mathcal{R}|$, best-of-N temperature sampling reaches comparable gains (Appendix B.2), so the recursion and the sampling oracle likely measure the same budget: the instances the model can get right by some process but not by default. A larger Monte Carlo route family at matched cardinality reaches a comparable oracle too, so our small family does not understate it.

No fixed route can fully capture this budget. The task-level oracle gains only -0.5 to $+2.7$ points over baseline. The 24 to 60 point gap between the two oracles means that useful routes are overwhelmingly instance-specific. To test this directly, we split MMStar into two equal halves, select the best fixed route on the first half, and apply it to the second. Figure 2 (Bottom) shows that this route either hurts or has no effect on 10 of 21 checkpoints. Because a route chosen on

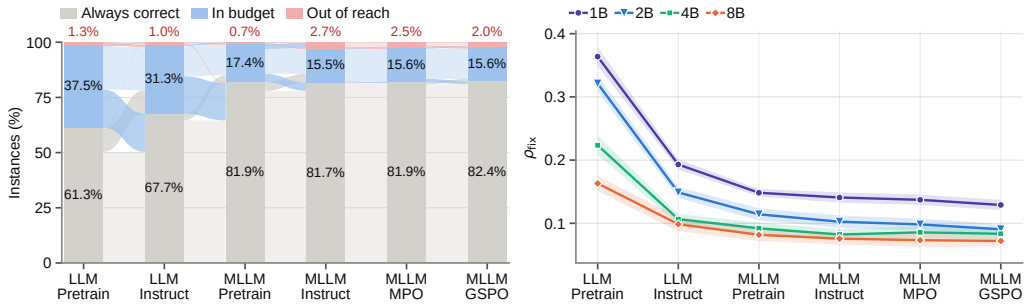


Figure 3: **Post-training draws down the recursion budget.** Results on ARC-Challenge along the Qwen3-1.7B → InternVL3.5-2B checkpoints. **Left:** The three instance categories across training stages. **Right:** ρ_{fix} across the same stages, one line per model scale.

one set of instances does not transfer to another, the budget cannot be reached by committing to a route in advance. This already suggests that test-time routing, which must choose a route without per-instance ground truth, will find little to exploit here, and it motivates the training-time method we develop in §6.

Summary. The recursion budget is universal across different models, yet it resists fixed-route selection. The next three sections ask what post-training does to this budget (§4), where in the network the interaction happens (§5), and whether any method can put what remains to practical use (§6).

4 Post-Training Draws Down the Recursion Budget

Is the recursion budget B a fixed property of an architecture, or does post-training reduce it? We answer this with the released intermediate checkpoints of Qwen3 (pretrained, instruct) and InternVL3.5 (pretrained, instruct, MPO (Wang et al., 2024b), GSPO (Zheng et al., 2025)). InternVL3.5 builds on the Qwen3 LLM backbone, so together these families span the full pipeline from LLM pretraining to the final MLLM RL stage.

The budget decreases with each training stage. We assign every instance at every checkpoint to one of the three categories from §2 and count transitions between consecutive checkpoints. Figure 3 (Left) shows the Qwen3-1.7B and InternVL3.5-2B series on ARC-Challenge. The budget B (blue) generally drops across transitions. The largest drops come at LLM instruction tuning (LLM Pretrain→LLM Instruct) and multimodal adaptation (LLM Instruct→MLLM Pretrain), where the in-budget fraction falls from 37.5% to 17.4%. Across the MLLM stages it then holds near 15.5%, most likely because those stages train on multimodal data rather than the text-only format of ARC-Challenge. The same pattern holds on MMStar across the InternVL3 and InternVL3.5 checkpoints (Appendix C.1).

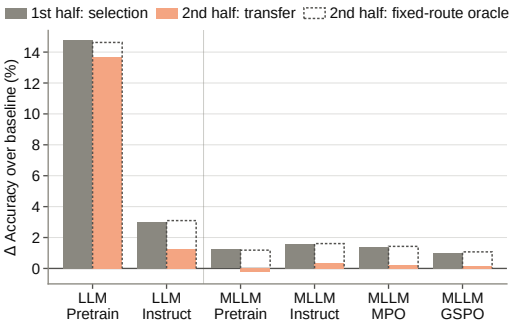


Figure 4: **Split-half transfer collapses after post-training.** We select the best fixed route on one half of ARC-Challenge and apply it to the other, along the Qwen3-1.7B → InternVL3.5-2B checkpoints.

The RL stages stay within the recursion family's reach. The full transition matrices on both benchmarks (Appendix C.1) show a striking asymmetry. Under every RL transition we measure (Instruct→MPO, MPO→GSPO), the out-of-reach → always-correct count is exactly zero. In these pipelines, RL never makes an out-of-reach instance correct. SFT (Pretrain→Instruct) faces no such limit, because it imitates teacher answers directly, and its out-of-reach → always-correct flow is nonzero. This split agrees with sampling-based studies (Chen et al., 2026b; Matsutani et al., 2026), where RL sharpens the correct outputs a model already produces without enlarging the set it can

solve. We reach the same conclusion through depth routes, where the reachable set is enumerated exactly rather than sampled.

The remaining budget becomes harder to exploit. Figure 3 (Right) tracks ρ_{fix} from LLM Pretrain to the final MLLM stage. Recall that ρ_{fix} is the average fraction of non-default routes that fix an instance the default route gets wrong, so it measures how easy the budget is to reach. Fix density falls at every scale in both families, with one exception at Instruct→MPO at 4B. Post-training therefore does two things at once, reducing how many instances are in budget and making the rest harder to reach. This is why prior test-time routers, evaluated on pretrained or lightly fine-tuned LLMs where ρ_{fix} is high, lose their edge on fully post-trained MLLMs where ρ_{fix} is 2 to $3.5\times$ lower (§6).

Figure 4 adds a complementary test, the split-half transfer of §3 repeated across all stages. At LLM pretraining the first-half route transfers almost perfectly, then degrades and disappears by the MLLM stages, in line with the falling ρ_{fix}. Appendix C.2 adds a representation-level view: mean pairwise inter-layer CKA falls with post-training, so layers grow more distinct and random routes are less likely to help.

Summary. Post-training steadily lowers both the size of the recursion budget and the accessibility of the instances left in it. This establishes *that* post-training draws down the budget, but not *how*. The budget could shrink because post-training improves the default path on its own, or because it learns to produce exactly what recursion was providing. §5 decides between these two accounts.

5 Post-Training Acts Where Recursion Was Helpful

To decide between the two accounts from §4, we compare post-training updates against recursion utility in weight space, logit space, and representation space. We define two instance sets for a given post-training transition. The *target set* holds instances that move from in-budget to always-correct, the ones the transition newly learns. The *control set* holds instances that are always-correct at both checkpoints. If the link between post-training and recursion were incidental, the two sets would behave the same way.

Weight space. For each layer ℓ, we compute r_ℓ, the fraction of instances in a set whose best recursion route repeats layer ℓ, and $\|\Delta W_\ell\|$, the Frobenius norm of the weight change across the transition. Figure 5 (Left) shows the InternVL3.5-2B MPO→GSPO transition. For the target set, r_ℓ and $\|\Delta W_\ell\|$ correlate positively (Spearman $\rho = +0.786$), so the layers that receive the largest updates are the ones most often covered by useful recursion routes. For the control set the same correlation flips to negative ($\rho = -0.551$). Useful routes cluster mid-network for both sets, so a generic mid-layer update could show up in each. The sign flip, however, shows the updates track the target set's routes specifically, not a shared mid-layer band. SFT transitions look different. Their updates concentrate on the last few layers, in line with fixing output format (Appendix D.1).

Logit space. For each instance $\boldsymbol{x}$ and layer ℓ, let $\boldsymbol{s}_\ell$, $\boldsymbol{f}_\ell$, and $\boldsymbol{r}_\ell$ be the LM-head logits under the default route of the earlier checkpoint, the default route of the later checkpoint, and the best recursion route of the earlier checkpoint (at the second pass through the repeated layer). We measure $\cos(\Delta \boldsymbol{r}_\ell, \Delta \boldsymbol{f}_\ell)$ with $\Delta \boldsymbol{r}_\ell = \boldsymbol{r}_\ell - \boldsymbol{s}_\ell$ and $\Delta \boldsymbol{f}_\ell = \boldsymbol{f}_\ell - \boldsymbol{s}_\ell$. Figure 5 (Middle) shows that for the target set, similarity peaks at 0.9 at the mid-to-late layers that most directly shape the generated output. The target set stays above the control set across this range, which confirms that post-training and recursion move the logits in the same direction for newly solved instances. The pattern holds for both RL and SFT transitions (Appendix D.2).

Representation space. For each layer ℓ, we measure the cosine similarity between two shifts. The first is the shift in the default hidden state from the earlier to the later checkpoint, $\mathbf{h}_\ell^{\text{default}}(\text{later}) - \mathbf{h}_\ell^{\text{default}}(\text{earlier})$. The second is the shift from the default hidden state to the best-recursion-route hidden state at the earlier checkpoint, $\mathbf{h}_\ell^{\text{recur}}(\text{earlier}) - \mathbf{h}_\ell^{\text{default}}(\text{earlier})$. If post-training absorbs the recursion budget, the two shifts should align for the target set. Figure 5 (Right) confirms this. The target set shows positive alignment at the mid-to-late layers, matching the logit-space result, while the control set shows no such pattern. This holds for both RL and SFT transitions (Appendix D.2).

Summary. Across all three spaces, the target-versus-control contrast supports the view that post-training absorbs the recursion budget. In logit and representation space, both SFT and RL align with

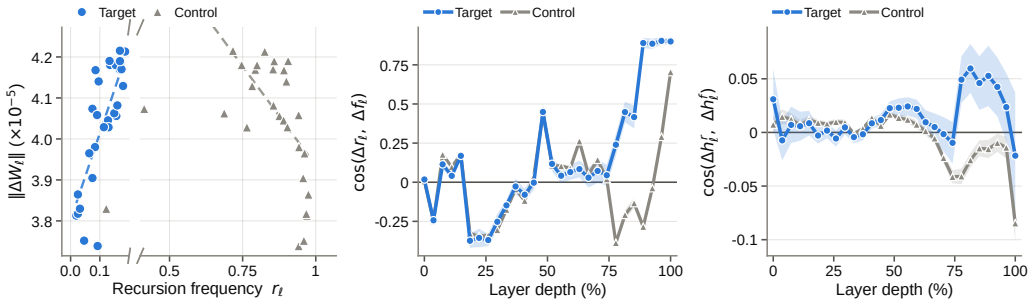


Figure 5: **Post-training acts where recursion was helpful.** InternVL3.5-2B, the MPO→GSPO transition, on MMStar. Each panel contrasts the target set (instances the transition newly learns) with the control set (instances already correct at both checkpoints). **Left:** Per-layer weight update norm $\|\Delta W_\ell\|$ against recursion layer frequency r_ℓ. **Middle:** Layer-wise cosine similarity between the recursion logit shift $\Delta \boldsymbol{r}_\ell$ and the post-training logit shift $\Delta \boldsymbol{f}_\ell$. **Right:** The same for hidden-state shifts. The target set aligns with recursion at the mid-to-late layers compared to the control set.

recursion for newly acquired instances. In weight space the two stages differ. RL updates the mid-layers where recursion was most useful, while SFT updates the late layers. These findings line up with prior work showing that RL updates concentrate in a sparse mid-layer subnetwork (Mukherjee et al., 2026) and that a single middle layer recovers most of RL's gain (Zhang et al., 2026). The practical consequence is direct. By the time a model is fully post-trained, its exploitable recursive structure has largely been absorbed into the default forward pass.

6 Recursion-Aware Post-Training

Test-time routing is not effective on fully post-trained MLLMs. The analysis in §4 and §5 predicts that test-time routing should find little benefit on a fully post-trained model. We verify this by implementing POLAR (Li et al., 2026), Dr.LLM (Heakl et al., 2026), and TFLT (Chen et al., 2026a) in the MLLM setting. For the two router methods, we sample 8k instances from the STEM split of Vero (Sarch et al., 2026), a multimodal instruction corpus whose STEM split covers mathematical and scientific reasoning, and generate MCTS-guided training paths in each method's own action space, following its original protocol. Table 1 reports the results. The router methods POLAR and Dr.LLM do not beat the default route, and the training-free block-selection method TFLT gains only $+0.4$ points. This is what the low ρ_{fix} predicts, since sparse helpful routes leave too little signal for any of these methods. Per-benchmark numbers and details are in Appendix F.

Table 1: **Test-time routing does not help on a fully post-trained MLLM** (InternVL3.5-2B, average over the four STEM benchmarks of Appendix G.3). POLAR and Dr.LLM are means over 3 seeds. TFLT is training-free and deterministic.

Method	Default	POLAR	Dr.LLM	TFLT
STEM	38.23	37.71	19.86	38.64

Method: spending the budget at training time. Since the remaining budget cannot be recovered at inference, we spend it during training. We take the recursion route most useful for a given task type and add it to the training forward pass. The model runs layers up to e, re-executes block $[s, e]$ once, then continues to layer $L-1$. Where standard fine-tuning only adjusts the weights behind the default forward pass, recursion-aware fine-tuning changes the pass itself. Block $[s, e]$ runs twice, and because both passes share weights, the optimizer can lower the loss only by making those layers produce representations that stay useful after re-entry. This is a stronger constraint than standard fine-tuning, applied at the band where §5 found post-training already absorbs recursion on its own. Fixing the band in advance makes that behavior explicit and installs re-entry-stable representations there, an inductive bias distinct from the standard RL stages which mainly sharpen outputs the model can already reach. Trainable parameters match standard fine-tuning, and each step costs 1.06 to 1.23× the wall-clock time depending on block size.

Route selection. We enumerate all routes on a 2k-instance subset of Vero-600k and select the top-1 route per task category by accuracy. No test instance takes part. We use the six Vero categories: STEM, Chart & OCR, Spatial & Action, Knowledge & Recognition, Grounding, Counting

Table 2: **Recursion-aware post-training on InternVL3.5-2B.** *Route* is the top-1 per-category route selected on the 2k subset. *Rec. Baseline* applies that route at inference without training. *Rec. GSPO* is our recursion-aware post-training, reported as mean and standard error over 3 seeds. Each cell is the average over that category's held-out benchmarks. Best accuracy per row in bold.

Category	Route	Baseline	Rec. Baseline	GSPO	Rec. GSPO
STEM	(19,19)	38.23	37.94	$39.01_{\pm 0.53}$	$\mathbf{40.89}_{\pm 0.15}$
Chart & OCR	(17,18)	55.57	55.13	$56.23_{\pm 0.20}$	$\mathbf{56.92}_{\pm 0.24}$
Spatial & Action	(13,15)	47.48	49.64	$49.18_{\pm 0.65}$	$\mathbf{51.57}_{\pm 0.62}$
Knowledge & Recognition	(14,15)	59.01	58.09	$\mathbf{59.22}_{\pm 0.08}$	$58.55_{\pm 0.11}$
Grounding, Counting & Search	(2,2)	51.19	50.55	$53.40_{\pm 0.27}$	$\mathbf{54.52}_{\pm 0.34}$
Captioning & IF	(2,3)	60.09	59.31	$63.58_{\pm 0.37}$	$\mathbf{66.70}_{\pm 0.28}$

& Search, and Captioning & IF. The selected routes follow the functional hierarchy from prior layer-role analyses (Shi et al., 2026; Yu & Lee, 2025), where perception favors early-layer routes and reasoning favors mid-to-late routes.

Setup. We fine-tune InternVL3.5-2B (L=28) on each of the six Vero-600k category splits independently (100k instances each) using that category's top-1 route. Each resulting model is evaluated on a separate benchmark suite that does not overlap with the route-selection subset, listed in Appendix G.3. We compare standard GSPO (Zheng et al., 2025) trained with default route r_0 against recursion-aware GSPO trained with per-category top-1 route under identical data, schedule, and trainable parameters, with a grid search over learning rates {1e-5, 1e-6}. SFT is not a baseline, because the model has already been through SFT and RL, and Sarch et al. (2026) report that further SFT does not help on their data. Following the short-answer setting of Sarch et al. (2026), we suppress chain-of-thought generation during training for both regimes, so that any difference between them comes from the recursion route rather than from longer generation. The reward is therefore the task-level outcome from the Vero scorer alone. Details can be found in Appendix G.1.

Results. Table 2 reports per-category accuracy, alongside the accuracy of applying the selected route at inference without training (Rec. Baseline), which isolates the training effect from inference-time depth. Recursion-aware GSPO outperforms standard GSPO on five of the six categories, with the largest gains on Captioning & IF (+3.1 points) and Spatial & Action (+2.4 points). Rec. Baseline stays at or below the baseline in five of the six categories and never reaches Rec. GSPO. In other words, applying the same route at inference without training does not improve accuracy, so the gain comes from the re-entry constraint, not the added depth. It also confirms that training reshapes the shared recursion weights so the chosen band represents the task better, a band the 2k subset already marked as helpful. It falls behind only on Knowledge & Recognition, matching Zhu et al. (2025b), who find looped models add no knowledge capacity, though for models looped from scratch. The overall pattern is consistent: knowledge-heavy categories stay flat or degrade slightly (Knowledge & Recognition and Chart & OCR), while perception (Captioning & IF and Grounding, Counting & Search) and reasoning (Spatial & Action and STEM) improve. Appendix G.6 gives per-benchmark numbers, Appendix G.5 applies the same protocol to three more backbones, and Appendix G.7 reports ablations controlling for compute, route selection, and category match.

7 Related Work

Layer reuse in frozen models. POLAR (Li et al., 2026) and Dr.LLM (Heakl et al., 2026) search per-instance chains of skipped and repeated layers on frozen text LLMs and train routers over that space. Inner Loop Inference (Lys et al., 2026) and Training-Free Looped Transformers (Chen et al., 2026a) re-apply a contiguous mid-stack block at inference. Our contribution is the measurement regime. We enumerate a small, closed route family exhaustively, which yields a budget that is comparable across checkpoints.

Training with reused depth. Think-at-Hard (Fu et al., 2025b) iterates latent depth at hard tokens with a depth-aware LoRA. RecursiveVLM (Xu et al., 2026) builds recursion into multimodal training from initialization. Universal Transformers (Dehghani et al., 2019), looped models (Zhu et al., 2025b; Saunshi et al., 2025; Geiping et al., 2025; McLeish et al., 2025), and Relaxed Recursive Transformers (Bae et al., 2024) tie weights by construction and train from scratch. Our contribution

is which route to use and why, chosen from the model's own exhaustively measured per-category utility map and applied to an already post-trained model.

What post-training does to model capabilities. Chen et al. (2026b) show that RL sharpens sampling toward solutions the model can already produce, and Matsutani et al. (2026) show that RL compresses incorrect trajectories while SFT expands correct ones. Our analysis gives independent support from a different measurement space. In contrast, prolonged RL (Liu et al., 2026) and RL on tasks designed to require new capabilities (Shen et al., 2026) can solve problems that the base model fails at any sampling budget. Our measurements cover only the standard RL stages of released MLLM pipelines on general benchmarks, so we make no claim about these settings. Zhang et al. (2026) show that a single middle layer recovers most of RL's gain, and Mukherjee et al. (2026) show that RL updates a sparse subnetwork. We claim only the correspondence with recursion utility. Layer-role analyses (Shi et al., 2026; Yu & Lee, 2025) place perception early and OCR deepest, and our per-category route ordering recovers this independently.

8 Discussion, Limitations, and Conclusion

Modality asymmetry. We also apply the route enumeration to vision-encoder (ViT) blocks instead of LLM decoder layers. Recursion helps in both places but less reliably in the vision encoder. The gap is about location rather than count. On Qwen2.5-VL-7B the ViT family is larger than the LLM decoder family (529 versus 407 routes), yet its instance-level headroom is 16 points smaller. Recursion capacity in a multimodal model is therefore better spent on the decoder. Appendix E.1 gives the full comparison.

Task-wise depth structure. The layers the best route covers per task type match known layer-role structure in MLLMs (Shi et al., 2026; Yu & Lee, 2025): perception errors are fixed early, reasoning errors mid-stack, and OCR errors late. This is what makes the per-category route selection in §6 principled rather than arbitrary, and why no single route helps in aggregate while per-category routes do. Appendix E.2 gives the full analysis.

Can post-training fully replace recursion? Our results show that standard post-training pipelines consume the recursion budget in a pretrained backbone, absorbing recursive structure into the default forward pass. Several considerations still suggest that recursive architectures keep structural advantages that post-training alone cannot recover. First, recursion-aware post-training beats standard fine-tuning even on a fully post-trained model. Had post-training fully replaced recursion, it would not. Second, models trained to be recursive from initialization (Dehghani et al., 2019; Zhu et al., 2025b; Geiping et al., 2025) show different convergence and generalization on depth-sensitive tasks than models adapted for recursion after post-trainings. Third, our analysis covers only a single re-execution of a fixed route. A model designed for variable-depth recursion has a different inductive bias, one where depth adapts to instance difficulty instead of being fixed at training time. Our analysis therefore supports only the narrower claim that post-training can make a pretrained MLLM recursive up to the budget left after the standard pipeline. Whether a from-scratch recursive architecture avoids spending its own budget during SFT and RL is left for future work.

Limitations. Our recursion family repeats one contiguous block once. Wider families that allow multiple repetitions or layer skipping exist, but Appendix B.2 shows that ours reaches comparable oracle coverage at matched cardinality, so the budget we measure is representative. Our cross-stage analysis relies on InternVL3 and InternVL3.5 because these are, to our knowledge, the only MLLM families that release intermediate checkpoints after both SFT and RL. Our RL findings therefore cover the standard MPO and GSPO stages of these releases on general benchmarks. Finally, we do not study thinking models that generate chain-of-thought tokens at inference or during training. How latent depth recursion acts with token-level chain-of-thought reasoning is a natural next step.

Conclusion. In released multimodal LLMs, post-training has already spent most of the budget for latent depth reuse, leaving little for recursion at inference to recover. This spending is not coincidental. Post-training logit shifts and representation changes align with where recursion was previously helpful. As a result, test-time routing methods that work on earlier-stage models fail on fully post-trained ones. Recursion-aware post-training offers a principled alternative that spends the remaining budget deliberately during training instead of searching for it at inference.

Reproducibility statement

All checkpoints used in this paper are publicly available on HuggingFace under their respective licenses, as listed in Appendix A.2. The recursion enumeration requires no training and is fully deterministic given the decoding procedure described in Appendix A.1. Recursion-aware post-training uses the same data, schedule, and codebase as the standard fine-tuning baselines, with the only modification being the addition of the fixed recursion route to the training forward pass.

Ethics statement

This work analyses and fine-tunes publicly released models on publicly released benchmarks. Recursion fine-tuning inherits the biases of the underlying checkpoints and of Vero-600k. It neither adds nor removes safety training, and we did not evaluate its effect on refusal behaviour, which is a gap for any deployment.

AI use statement

Large language models were used for grammar checking and sentence-level rephrasing during paper writing. All experimental results, analysis, and scientific claims are the authors' own.

References

Abdelrahman Abouelenin, Atabak Ashfaq, Adam Atkinson, Hany Awadalla, Nguyen Bach, Jianmin Bao, Alon Benhaim, Martin Cai, Vishrav Chaudhary, Congcong Chen, et al. Phi-4-mini technical report: Compact yet powerful multimodal language models via mixture-of-loras. *arXiv preprint arXiv:2503.01743*, 2025.

Pravesh Agrawal, Szymon Antoniak, Emma Bou Hanna, Baptiste Bout, Devendra Chaplot, Jessica Chudnovsky, Diogo Costa, Baudouin De Monicault, Saurabh Garg, Theophile Gervet, et al. Pixtral 12b. *arXiv preprint arXiv:2410.07073*, 2024.

Xiang An, Yin Xie, Kaicheng Yang, Wenkang Zhang, Xiuwei Zhao, Zheng Cheng, Yirui Wang, Songcen Xu, Changrui Chen, Didi Zhu, et al. Llava-onevision-1.5: Fully open framework for democratized multimodal training. *arXiv preprint arXiv:2509.23661*, 2025.

Sangmin Bae, Adam Fisch, Hrayr Harutyunyan, Ziwei Ji, Seungyeon Kim, and Tal Schuster. Relaxed recursive transformers: Effective parameter sharing with layer-wise lora. *arXiv preprint arXiv:2410.20672*, 2024.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Lucas Beyer, Andreas Steiner, André Susano Pinto, Alexander Kolesnikov, Xiao Wang, Daniel Salz, Maxim Neumann, Ibrahim Alabdulmohsin, Michael Tschannen, Emanuele Bugliarello, et al. Paligemma: A versatile 3b vlm for transfer. *arXiv preprint arXiv:2407.07726*, 2024.

Lin Chen, Jinsong Li, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Zehui Chen, Haodong Duan, Jiaqi Wang, Yu Qiao, Dahua Lin, et al. Are we on the right way for evaluating large vision-language models? *Advances in Neural Information Processing Systems*, 37:27056–27087, 2024.

Lizhang Chen, Jonathan Li, Chen Liang, Ni Lao, and Qiang Liu. Training-free looped transformers. *arXiv preprint arXiv:2605.23872*, 2026a.

Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? *Advances in Neural Information Processing Systems*, 38:57654–57689, 2026b.

Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Li YanTao, Jianbing Zhang, and Zhiyong Wu. Seeclick: Harnessing gui grounding for advanced visual gui agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9313–9332, 2024.

Xianfu Cheng, Wei Zhang, Shiwei Zhang, Jian Yang, Xiangyuan Guan, Xianjie Wu, Xiang Li, Ge Zhang, Jiaheng Liu, Yuying Mai, et al. Simplevqa: Multimodal factuality evaluation for multimodal large language models. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4637–4646. IEEE, 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.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. Universal transformers. In *International Conference on Learning Representations*, 2019.

Shengyuan Ding, Shenxi Wu, Xiangyu Zhao, Yuhang Zang, Haodong Duan, Xiaoyi Dong, Pan Zhang, Yuhang Cao, Dahua Lin, and Jiaqi Wang. Mm-ifengine: Towards multimodal instruction following. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 1099–1109. IEEE, 2025.

Mengfei Du, Binhao Wu, Zejun Li, Xuan-Jing Huang, and Zhongyu Wei. Embspatial-bench: Benchmarking spatial understanding for embodied tasks with large vision-language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 346–355, 2024.

Chaoyou Fu, Yuhan Dai, Yongdong Luo, Lei Li, Shuhuai Ren, Renrui Zhang, Zihan Wang, Chenyu Zhou, Yunhang Shen, Mengdan Zhang, et al. Video-mme: The first-ever comprehensive evaluation benchmark of multi-modal llms in video analysis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 24108–24118, 2025a.

Tianyu Fu, Yichen You, Zekai Chen, Guohao Dai, Huazhong Yang, and Yu Wang. Think-at-hard: Selective latent iterations to improve reasoning language models. *arXiv preprint arXiv:2511.08577*, 2025b.

Xingyu Fu, Yushi Hu, Bangzheng Li, Yu Feng, Haoyu Wang, Xudong Lin, Dan Roth, Noah A Smith, Wei-Chiu Ma, and Ranjay Krishna. Blink: Multimodal large language models can see but not perceive. In *European Conference on Computer Vision*, pp. 148–166. Springer, 2024.

Jonas Geiping, Sean Michael McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Ahmed Heakl, Martin Gubri, Salman Khan, Sangdoo Yun, and Seong Joon Oh. Dr.LLM: Dynamic layer routing in LLMs. In *The Fourteenth International Conference on Learning Representations*, 2026.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International conference on machine learning*, pp. 3519–3529. PMLR, 2019.

Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, et al. Llava-onevision: Easy visual task transfer. *arXiv preprint arXiv:2408.03326*, 2024a.

Bohao Li, Yuying Ge, Yixiao Ge, Guangzhi Wang, Rui Wang, Ruimao Zhang, and Ying Shan. Seed-bench: Benchmarking multimodal large language models. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13299–13308. IEEE, 2024b.

Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Xin Zhao, and Ji-Rong Wen. Evaluating object hallucination in large vision-language models. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 292–305, 2023.

Ziyue Li, Yang Li, and Tianyi Zhou. Skip a layer or loop it? learning program-of-layers in llms. *arXiv preprint arXiv:2606.06574*, 2026.

Mingjie Liu, Shizhe Diao, Ximing Lu, Jian Hu, Xin Dong, Yejin Choi, Jan Kautz, and Yi Dong. Prorl: Prolonged reinforcement learning expands reasoning boundaries in large language models. *Advances in Neural Information Processing Systems*, 38:17998–18031, 2026.

Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, et al. Mmbench: Is your multi-modal model an all-around player? In *European conference on computer vision*, pp. 216–233. Springer, 2024a.

Yuliang Liu, Zhang Li, Mingxin Huang, Biao Yang, Wenwen Yu, Chunyuan Li, Xu-Cheng Yin, Cheng-Lin Liu, Lianwen Jin, and Xiang Bai. Ocrbench: on the hidden mystery of ocr in large multimodal models. *Science China Information Sciences*, 67(12):220102, 2024b.

Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. In *International Conference on Learning Representations*, 2024.

Jonathan Lys, Vincent Gripon, Bastien Pasdeloup, Axel Marmoret, Lukas Mauch, Fabien Cardinaux, and Ghouthi Boukli Hacene. Inner loop inference for pretrained transformers: Unlocking latent capabilities without training. *arXiv preprint arXiv:2602.14759*, 2026.

Karttikeya Mangalam, Raiymbek Akshulakov, and Jitendra Malik. Egoschema: A diagnostic benchmark for very long-form video language understanding. *Advances in Neural Information Processing Systems*, 36:46212–46244, 2023.

Ahmed Masry, Jia Qing Tan, Shafiq Joty, Enamul Hoque, et al. Chartqa: A benchmark for question answering about charts with visual and logical reasoning. In *Findings of the association for computational linguistics: ACL 2022*, pp. 2263–2279, 2022.

Minesh Mathew, Dimosthenis Karatzas, and CV Jawahar. Docvqa: A dataset for vqa on document images. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pp. 2200–2209, 2021.

Minesh Mathew, Viraj Bagal, Rubèn Tito, Dimosthenis Karatzas, Ernest Valveny, and CV Jawahar. Infographicvqa. In *2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 2582–2591. IEEE, 2022.

Kohsei Matsutani, Shota Takashiro, Gouki Minegishi, Takeshi Kojima, Yusuke Iwasawa, and Yutaka Matsuo. Rl squeezes, sft expands: A comparative study of reasoning llms. In *International Conference on Learning Representations*, volume 2026, pp. 141482–141526, 2026.

Sean McLeish, Ang Li, John Kirchenbauer, Dayal Singh Kalra, Brian R Bartoldson, Bhavya Kailkhura, Avi Schwarzschild, Jonas Geiping, Tom Goldstein, and Micah Goldblum. Teaching pretrained language models to think deeper with retrofitted recurrence. *arXiv preprint arXiv:2511.07384*, 2025. doi: 10.48550/arXiv.2511.07384.

Sagnik Mukherjee, Lifan Yuan, Dilek Hakkani-Tur, and Hao Peng. Reinforcement learning finetunes small subnetworks in large language models. *Advances in Neural Information Processing Systems*, 38:132119–132138, 2026.

Yusu Qian, Hanrong Ye, Jean-Philippe Fauconnier, Peter Grasch, Yinfei Yang, and Zhe Gan. Miabench: Towards better instruction following evaluation of multimodal llms. In *International Conference on Learning Representations*, volume 2025, pp. 35145–35165, 2025.

Pooyan Rahmanzadehgervi, Logan Bolton, Mohammad Reza Taesiri, and Anh Totti Nguyen. Vision language models are blind. In *Asian Conference on Computer Vision*, pp. 293–309. Springer, 2024.

Kristiyan Sakalyan, Sanghwan Kim, Leo Schwinn, Quentin Bouniot, and Zeynep Akata. Block-level recursion: Adaptive test-time routing in large language models. In *AdaptFM: Resource-Adaptive Foundation Model Inference*, 2026.

Gabriel Sarch, Linrong Cai, Qunzhong Wang, Haoyang Wu, Danqi Chen, and Zhuang Liu. Vero: An open rl recipe for general visual reasoning. *arXiv preprint arXiv:2604.04917*, 2026.

Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J. Reddi. Reasoning with latent thoughts: On the power of looped transformers. In *The Thirteenth International Conference on Learning Representations*, 2025.

Minghe Shen, Zhuo Zhi, Chonghan Liu, Shuo Xing, Zhengzhong Tu, and Che Liu. Does rlvr extend reasoning boundaries? investigating capability expansion in vision-language models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 45317–45331, 2026.

Cheng Shi, Yizhou Yu, and Sibei Yang. Vision function layer in multimodal llms. *Advances in neural information processing systems*, 38:27510–27537, 2026.

Yueqi Song, Tianyue Ou, Yibo Kong, Zecheng Li, Graham Neubig, and Xiang Yue. Visualpuzzles: Decoupling multimodal reasoning evaluation from domain knowledge. *arXiv preprint arXiv:2504.10342*, 2025.

Jayant Sravan Tamarapalli, Rynaa Grover, Nilay Pande, and Sahiti Yerramilli. Countqa: How well do mllms count in the wild? *arXiv preprint arXiv:2508.06585*, 2025.

Liyan Tang, Grace Kim, Xinyu Zhao, Thom Lake, Wenxuan Ding, Fangcong Yin, Prasann Singhal, Manya Wadhwa, Zeyu Liu, Zayne Sprague, et al. Chartmuseum: Testing visual reasoning capabilities of large vision-language models. *Advances in Neural Information Processing Systems*, 38, 2026.

Gemma Team. Gemma 3 technical report, 2025a. URL https://arxiv.org/abs/2503.19786.

Qwen Team. Qwen2.5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025b. URL https://arxiv.org/abs/2502.13923.

Shengbang Tong, Ellis Brown, Penghao Wu, Sanghyun Woo, Manoj Middepogu, Sai C Akula, Jihan Yang, Shusheng Yang, Adithya Iyer, Xichen Pan, et al. Cambrian-1: A fully open, vision-centric exploration of multimodal llms. *Advances in Neural Information Processing Systems*, 37:87310–87356, 2024.

Ke Wang, Junting Pan, Weikang Shi, Zimu Lu, Houxing Ren, Aojun Zhou, Mingjie Zhan, and Hongsheng Li. Measuring multimodal mathematical reasoning with math-vision dataset. In *Advances in Neural Information Processing Systems*, 2024a.

Weiyun Wang, Zhe Chen, Wenhai Wang, Yue Cao, Yangzhou Liu, Zhangwei Gao, Jinguo Zhu, Xizhou Zhu, Lewei Lu, Yu Qiao, et al. Enhancing the reasoning ability of multimodal large language models via mixed preference optimization. *arXiv preprint arXiv:2411.10442*, 2024b.

Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, Xingguang Wei, Zhaoyang Liu, Linglin Jing, Shenglong Ye, Jie Shao, et al. Internvl3.5: Advancing open-source multimodal models in versatility, reasoning, and efficiency. *arXiv preprint arXiv:2508.18265*, 2025a.

Wenbin Wang, Liang Ding, Minyan Zeng, Xiabin Zhou, Li Shen, Yong Luo, Wei Yu, and Dacheng Tao. Divide, conquer and combine: A training-free framework for high-resolution image perception in multimodal large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2025b.

Zirui Wang, Mengzhou Xia, Luxi He, Howard Chen, Yitao Liu, Richard Zhu, Kaiqu Liang, Xindi Wu, Haotian Liu, Sadhika Malladi, et al. Charxiv: Charting gaps in realistic chart understanding in multimodal llms. *Advances in Neural Information Processing Systems*, 37:113569–113697, 2024c.

Jinming Wu, Zihao Deng, Wei Li, Yiding Liu, Bo You, Bo Li, Zejun Ma, and Ziwei Liu. Mmsearch-r1: Incentivizing lmms to search. *arXiv preprint arXiv:2506.20670*, 2(4), 2025.

Penghao Wu and Saining Xie. V?: Guided visual search as a core mechanism in multimodal llms. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13084–13094, 2024.

Ruihan Xu, Yuting Gao, Lan Wang, Jianing Li, Weihao Chen, Qingpei Guo, Ming Yang, and Shiliang Zhang. Looping back to move forward: Recursive transformers for efficient and flexible large multimodal models. *arXiv preprint arXiv:2602.09080*, 2026.

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.

Sihan Yang, Runsen Xu, Yiman Xie, Sizhe Yang, Mo Li, Jingli Lin, Chenming Zhu, Xiaochen Chen, Haodong Duan, Xiangyu Yue, et al. Mmsi-bench: A benchmark for multi-image spatial intelligence. In *International Conference on Learning Representations*, volume 2026, pp. 157051–157088, 2026.

Weihao Yu, Zhengyuan Yang, Lingfeng Ren, Linjie Li, Jianfeng Wang, Kevin Lin, Chung-Ching Lin, Zicheng Liu, Lijuan Wang, and Xinchao Wang. Mm-vet v2: A challenging benchmark to evaluate large multimodal models for integrated capabilities. *arXiv preprint arXiv:2408.00765*, 2024.

Zhuoran Yu and Yong Jae Lee. How multimodal llms solve image tasks: A lens on visual grounding, task reasoning, and answer decoding. *arXiv preprint arXiv:2508.20279*, 2025.

Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, et al. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9556–9567, 2024.

Zijian Zhang, Rizhen Hu, Athanasios Glentis, Dawei Li, Chung-Yiu Yau, Hongzhou Lin, and Mingyi Hong. Is one layer enough? training a single transformer layer can match full-parameter rl training. *arXiv preprint arXiv:2607.01232*, 2026.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, et al. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 2025.

Jinguo Zhu, Weiyun Wang, Zhe Chen, Zhaoyang Liu, Shenglong Ye, Lixin Gu, Hao Tian, Yuchen Duan, Weijie Su, Jie Shao, et al. Internvl3: Exploring advanced training and test-time recipes for open-source multimodal models. *arXiv preprint arXiv:2504.10479*, 2025a.

Rui-Jie Zhu, Zixuan Wang, Kai Hua, Tianyu Zhang, Ziniu Li, Haoran Que, Boyi Wei, Zixin Wen, Fan Yin, He Xing, et al. Scaling latent reasoning via looped language models. *arXiv preprint arXiv:2510.25741*, 2025b. doi: 10.48550/arXiv.2510.25741.

APPENDIX OVERVIEW

This appendix provides supplementary details, extended results, and ablations for the main paper.

A. Analysis Setup

A.1. Decoding and Scoring: Gibberish detection, answer extraction rules, and LLM-as-judge.
A.2. Models and Benchmarks: Full checkpoint list and benchmark descriptions.

B. Extended Oracle and Budget Analysis

B.1. Oracle on Video and Free-Form Generation: Two-oracle gap beyond MMStar.
B.2. Matched-Cardinality Oracle Controls: Comparison with MCTS route families and temperature sampling.

C. Post-Training Dynamics

C.1. Bucket Transition Matrices: Full instance-category flows on MMStar and ARC-Challenge.
C.2. Inter-Layer Similarity: CKA analysis across scale and training stages.

D. Alignment Analysis

D.1. Weight-Space Correlation: Extended weight-update vs. recursion frequency plots.
D.2. Logit and Representation Alignment: Full transition set for policy and hidden-state alignment.

E. Additional Analysis

E.1. Modality Asymmetry: LLM decoder vs. ViT encoder recursion.
E.2. Task-Wise Depth Structure: Per-category optimal route distributions.

F. Test-Time Routing Reproduction

F.1. Methods: TFLT, Dr.LLM, and POLAR descriptions.
F.2. Adaptation to the Multimodal Setting: Search corpus, KV cache, and router input.
F.3. Per-Benchmark Results: Breakdown of Table 1.

G. Recursion-Aware Post-Training

G.1. Implementation Details: Vero categories, reward, GSPO objective.
G.2. Training Hyperparameters: GSPO configuration for recursion-aware post-training.
G.3. Held-Out Evaluation Benchmarks: Per-category benchmark suites.
G.4. Route Utility Heatmap: Per-category route selection on Vero subset.
G.5. Generalization to Other Models: Results on additional backbones.
G.6. Extra Results: Per-category, average-route, and cross-category evaluation.
G.7. Ablations: FLOP-matched, random route, and wrong-category controls.

A Analysis Setup

A.1 Decoding and Scoring

During analysis, every route on every instance is decoded greedily with a fixed token budget of 32, extended to 512 for MIA-Bench (Qian et al., 2025). The same setting is used for every route on a given benchmark, so differences between routes come solely from the route itself.

Our scoring pipeline has two stages: a gibberish detector and a rule-based answer extractor. Because models have not been trained with recursion routes, non-default routes occasionally produce degenerate outputs. The gibberish detector filters these before scoring to prevent broken strings from accidentally matching a correct answer. Table 3 lists the failure patterns it catches.

For lettered ground truths, we look for explicit cues such as "the answer is B" or a bracketed letter rather than the first bare capital letter, which is easily confused with an article or unrelated token. We also read the question's own option list so that a route merely listing several options is not credited with choosing one, and so that a lettered prediction can be matched against a worded ground truth. Table 4 shows examples of indirect answers that our extractor resolves correctly despite not being plain string matches.

For long-form responses (MIA-Bench), we use Qwen3-32B (Yang et al., 2025) as an LLM judge, which scores each component and returns a total out of 10. To keep this consistent with the binary $\mathrm{acc}(\boldsymbol{x}, r)$ used elsewhere, we count a route as correct only when it earns a perfect score.

Table 3: **Patterns caught by the gibberish detector.**

#	Pattern	Example
1	Empty output	`(blank)`
2	Character-level repetition	`0000000000000000000`
3	Word-level repetition	`-nessesque-nessesque-nessesque`
4	Phrase-level repetition	`The answer is 18. The answer is 18. The answer is 18.`
5	Language switching	Mixed use of English and CJK
6	Markup format	`{"content": "80"}{"content": "80"}`
7	Punctuation repetition	`e: e: e: e(e; e:`
8	Camel Case	`TheAnswerIsDefinitelyOptionB`

Table 4: **Indirect answers correctly matched to ground truth.** These outputs pass the gibberish gate and are resolved to the correct answer.

#	Model output	GT
1	`The answer is A.`	A
2	`I think the answer is A, since the chart clearly trends upward toward the end.`	A
3	`(C) is correct.`	C
4	`Based on the image, option D seems to best match the description.`	D
5	`The answer is (B) Blue.`	Blue

A.2 Models and Benchmarks

Table 5 lists all checkpoints used in this paper, all publicly available on HuggingFace. For InternVL3 and InternVL3.5, we include every released intermediate post-training stage in addition to the final checkpoint. Table 6 lists the benchmarks used in our analysis.

Table 5: **Model checkpoints.**

#	Family	Checkpoint
1	Qwen3	`Qwen3-{0.6B,1.7B,4B,8B}` `Qwen3-{0.6B,1.7B,4B,8B}-Base`
2	LLaVA-OneVision	`llava-onevision-qwen2-{0.5b,7b}-ov`
3	LLaVA-OneVision-1.5	`LLaVA-OneVision-1.5-{4B,8B}-Instruct`
4	Qwen2.5-VL	`Qwen2.5-VL-{3B,7B}-Instruct`
5	Qwen3-VL	`Qwen3-VL-{2B,4B,8B}-Instruct`
6	Phi-4-Multimodal	`Phi-4-multimodal-instruct`
7	Gemma 3	`gemma-3-{4b,12b,27b}-it`
8	InternVL3	`InternVL3-{1B,2B,8B,14B}` `InternVL3-{1B,2B,8B,14B}-{Pretrained,Instruct}`
9	InternVL3.5	`InternVL3_5-{1B,2B,4B,8B}` `InternVL3_5-{1B,2B,4B,8B}-{Pretrained,Instruct,MPO}`

Table 6: **Analysis benchmarks.**

Benchmark	Size	Answer format
ARC-Challenge (Clark et al., 2018)	1,172	Multiple choice
MMStar (Chen et al., 2024)	1,500	Multiple choice
ChartQA-lite (Masry et al., 2022)	500	Short free-form
Video-MME-sub (Fu et al., 2025a)	600	Multiple choice
EgoSchema-sub (Mangalam et al., 2023)	500	Multiple choice
MIA-Bench (Qian et al., 2025)	400	Free-form, LLM judge

B Extended Oracle and Budget Analysis

B.1 Oracle Generalizes to Video and Free-Form Generation

Table 7 extends both oracles to video (Video-MME-sub, EgoSchema-sub), short free-form answers (ChartQA-lite), and long-form generation (MIA-Bench) for the two models used throughout the paper. The pattern of §3 holds in every cell: the instance oracle gains $+12.6$ to $+37.8$ points while the best single fixed non-default route gains at most $+5.2$. Long-form generation scored by the judge behaves identically, with $+26.6$ and $+31.5$ instance-level gains against $+1.3$ and $+2.8$ task-level gains. The budget is therefore not specific to multiple-choice perception.

The two columns should be read together. On Qwen2.5-VL-7B/ChartQA-lite, the best fixed route $(26, 26)$ ties the baseline exactly ($+0.00$) while the instance oracle sits $+12.6$ above it: useful routes are entirely instance-specific. Note that r_0 is excluded from the task-level search by construction, so that column may fall below baseline. The winning routes share no common location across models, consistent with the non-transfer observed in Figure 2.

Table 7: **Task-level vs. instance-level oracle beyond MMStar.** $\mathrm{Oracle}_{\mathrm{task}}$ is the best single non-default route applied uniformly (r_0 excluded, winning route shown). $\mathrm{Oracle}_{\mathrm{inst}}$ is the per-instance best over all routes. Δ is relative to that row's baseline. MIA-Bench reports mean Qwen3-32B judge score.

				$\mathrm{Oracle}_{\mathrm{task}}$			$\mathrm{Oracle}_{\mathrm{inst}}$	
Model	**Task**	n	base	score	Δ	route	score	Δ
InternVL3.5-8B	MMStar	1500	66.87	67.80	+0.93	(16, 23)	97.87	+31.00
($L = 36$,	ChartQA-lite	500	79.20	81.00	+1.80	(13, 17)	95.00	+15.80
$\lvert\mathcal{R}\rvert = 667$)	MIA-Bench	400	62.69	63.95	+1.26	(28, 28)	89.25	+26.56
	Video-MME-sub	600	67.50	69.00	+1.50	(15, 16)	97.00	+29.50
	EgoSchema-sub	500	64.40	69.60	+5.20	(3, 3)	94.00	+29.60
Qwen2.5-VL-7B	MMStar	1500	61.87	62.87	+1.00	(19, 19)	98.33	+36.47
($L = 28$,	ChartQA-lite	500	83.00	83.00	+0.00	(26, 26)	95.60	+12.60
$\lvert\mathcal{R}\rvert = 407$)	MIA-Bench	400	57.34	60.11	+2.77	(18, 21)	88.89	+31.54
	Video-MME-sub	600	63.67	64.83	+1.17	(12, 15)	98.00	+34.33
	EgoSchema-sub	500	59.60	64.00	+4.40	(0, 0)	97.40	+37.80

B.2 Matched-Cardinality Oracle Controls and Route Family Comparison

An instance oracle over $|\mathcal{R}|$ actions is a maximum over $|\mathcal{R}|$ chances. Before interpreting its magnitude we must ask what any other action set of the same size would deliver. We run two comparisons: (1) $N = |\mathcal{R}|$ i.i.d. draws at temperature 1.0 (top-p = 1.0, top-k disabled, no repetition penalty), which is best-of-N sampling at exactly the recursion budget, and (2) a Monte Carlo route family of the same cardinality, where each route is a randomly sampled chain of layer-level skip and repeat operations from the space used by POLAR and Dr.LLM, truncated to N unique routes. Prompts, image tiling, decode length, and scorer are shared with the route sweep. Table 8 reports results for our family and temperature sampling. Table 9 compares all three families.

The oracle magnitude reflects model headroom, not the specific route family. At matched cardinality, all action sets land on similar oracles. Over the six (model $\times$ task) cells, gains differ by at most 2.0 points between our family and temperature sampling, and by at most 2.0 points between our family and the Monte Carlo route family. The sign of the difference is inconsistent. Thus, a large instance oracle can be the generic behavior of $\sim 10^2$ actions on these benchmarks, reflecting headroom above greedy decoding rather than the only property of latent depth.

In other words, both the recursion oracle and the best-of-N oracle may reflect the same underlying budget: instances the model can answer correctly by some process but does not by default. Post-training consumes this budget from both directions simultaneously. RL has been reported to sharpen the default path toward correct outputs the model could already sample (Chen et al., 2026b), and our analysis shows that the RL stages we measure also absorb the recursion budget.

Routes produce more answer variation than sampling. A route changes the answer on 37.4–42.3% of instances, compared to 13.4–25.4% for temperature sampling: recursion moves the answer 2–3$\times$ as often. Normalizing the oracle gain by this rate, sampling returns 1.7–3.0$\times$ more oracle per unit of answer change. On MIA-Bench both sets change essentially every answer (98.1–98.8%) and the normalized numbers coincide. Re-executing a route is thus a high-variance intervention, and what recursion offers is not a well-behaved neighborhood of the default answer.

Comparable coverage to larger search families. A natural concern is that our small route family understates the true budget. Table 9 addresses this directly: we search the larger MCTS space until it yields $N = |\mathcal{R}|$ distinct layer programs per instance. All three families land within 3.5 points of one another, and ours is within 2.0 points of the searched one. At $k = N$ every family sits at 96–100%, near saturation. Since the MCTS search is reward-guided, its oracle overestimates what a random draw of N programs would give, making this a conservative comparison for our family. We therefore use the oracle gap as an upper bound on what any selector of similar budget can achieve.

Table 8: **Matched-cardinality control: recursion routes vs. temperature sampling at $N = |\mathcal{R}|$.** Both action sets hold exactly N members. The k columns are the expected maximum over a random k-subset, so $k{=}N$ is the instance oracle. Δ is oracle minus that row's base, and "chg" is the fraction of actions that change the answer. MIA-Bench entries are mean Qwen3-32B judge score.

				$\text{Oracle}_{\text{inst}}$					
Model / task	**action set**	N	base	$k{=}8$	$k{=}32$	$k{=}128$	$k{=}N$	Δ	chg
InternVL3.5-8B ($L = 36$)									
MMStar	routes	667	66.87	75.90	83.59	91.13	**97.87**	+31.00	39.1%
	sampling	667	66.87	81.25	88.04	92.97	96.33	+29.47	13.4%
ChartQA-lite	routes	667	79.20	85.86	90.17	92.81	95.00	+15.80	42.3%
	sampling	667	79.20	87.71	91.62	94.07	**95.80**	+16.60	14.8%
MIA-Bench	routes	667	62.69	74.93	82.19	86.44	89.25	+26.56	98.3%
	sampling	667	62.55	78.38	84.49	88.19	**90.68**	+28.12	98.5%
Qwen2.5-VL-7B ($L = 28$)									
MMStar	routes	407	61.87	76.60	88.12	95.72	98.33	+36.47	42.1%
	sampling	407	61.87	85.15	94.05	97.79	**98.93**	+37.07	25.4%
ChartQA-lite	routes	407	83.00	88.10	91.50	93.77	95.60	+12.60	37.4%
	sampling	407	83.00	90.36	93.72	96.09	**97.60**	+14.60	18.7%
MIA-Bench	routes	407	57.34	72.32	80.73	86.11	**88.89**	+31.54	98.1%
	sampling	407	57.34	73.54	81.12	85.96	88.55	+31.21	98.8%

Table 9: **Oracle comparison across route families at matched cardinality $N = |\mathcal{R}|$.** All three families hold exactly N actions. "MCTS family" denotes N routes sampled from the layer-skip-and-repeat space used by POLAR and Dr.LLM with no ground-truth guidance. Δ is oracle minus baseline. Results on MMStar.

Model	**Action family**	N	$\text{Oracle}_{\text{inst}}$	Δ
InternVL3.5-8B	Our recursion family	667	97.87	+31.00
	MCTS route family	667	99.87	+33.00
	Temperature sampling	667	96.33	+29.47
Qwen2.5-VL-7B	Our recursion family	407	98.33	+36.47
	MCTS route family	407	98.87	+37.00
	Temperature sampling	407	98.93	+37.07

C Post-Training Dynamics

C.1 Bucket Transition Matrices

Figure 6 extends the budget analysis of §4 to MMStar, showing bucket population flows and ρ_{fix} across training stages for InternVL3.5 and InternVL3 at the 8B scale. The recursion budget (in-budget fraction) decreases with each post-training stage, and ρ_{fix} falls consistently, weakening the training signal available to any test-time router.

Figures 7 and 8 report the complete per-cell transition matrices. Each matrix shows how instances move between the three categories (always-correct, in-budget, out-of-reach) across consecutive checkpoints. Rows are the bucket at the earlier checkpoint, columns at the later one, and each row sums to 100%.

The key observation is the boxed cell: the out-of-reach $\rightarrow$ always-correct flow. Under every RL transition (Instruct$\rightarrow$MPO, MPO$\rightarrow$GSPO), this count is an exact zero at every scale in both families. In these transitions, RL never produces a correct default-path answer for an instance that no recursion route could solve at the prior checkpoint. Under SFT, the same entry is nonzero: 12.0% for InternVL3.5 and 8.7% for InternVL3. This dissociation indicates that, in these pipelines, RL sharpens within the model's existing reach while SFT can expand it.

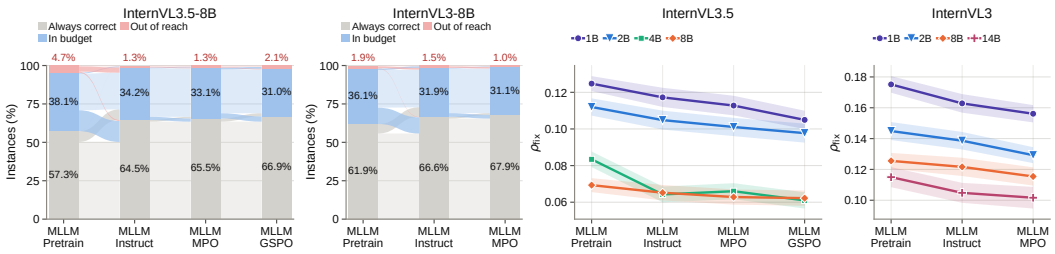


Figure 6: **Post-training spends the recursion budget on MMStar.** All results pooled over InternVL3.5 (Pretrained, Instruct, MPO, GSPO) and InternVL3 (Pretrained, Instruct, MPO). **Left pair:** Bucket population flow at 8B scale. The recursion budget (blue) decreases with each training stage. **Right pair:** ρ_{fix} across training stages, one line per scale. Recursion fix density decreases consistently, weakening the signal available to test-time routers.

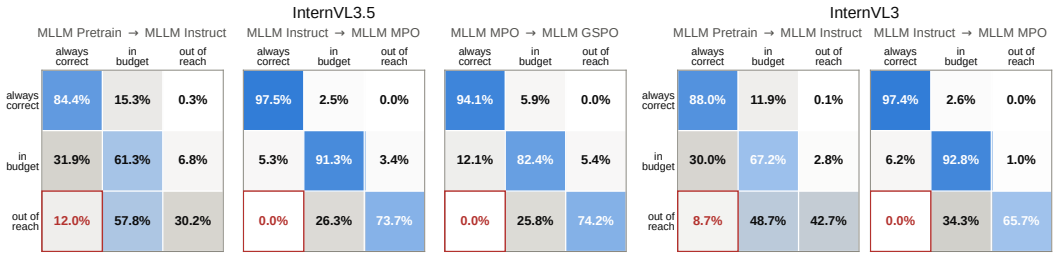


Figure 7: **Instance-bucket transitions on MMStar. Left**: InternVL3.5, pooled over 1B, 2B, 4B, 8B scales. **Right**: InternVL3, pooled over 1B, 2B, 8B, 14B scales. Rows are the bucket at the earlier checkpoint, columns at the later one. The boxed cell is the out-of-reach → always-correct flow, which is an exact zero at every RL step and nonzero only under SFT.

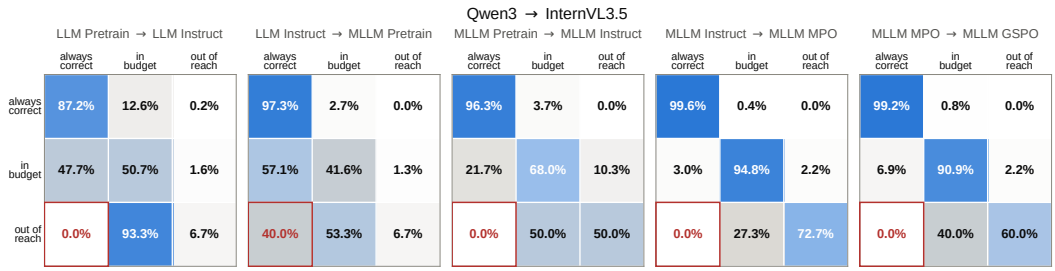


Figure 8: **Instance-bucket transitions on ARC-Challenge along the text-LLM → MLLM checkpoints.** Qwen3-{0.6B,1.7B,4B,8B} → InternVL3.5-{1B,2B,4B,8B}. Every checkpoint shares the Qwen3 decoder, so the route family is identical along the checkpoints and all five panels are directly comparable. The boxed cell is the out-of-reach → always-correct flow.

The ARC-Challenge transitions (Figure 8) reveal a finer distinction within SFT stages. The multimodal adaptation step (LLM Instruct→MLLM Pretrain) shows a 40.0% out-of-reach → always-correct flow, because this stage mixes text and multimodal data, allowing the model to expand its text-task capabilities. In contrast, the subsequent MLLM SFT step (MLLM Pretrain→MLLM Instruct) shows an exact zero in the same cell. This is expected: the MLLM instruction tuning stage trains exclusively on multimodal data and does not update the model's text-only task boundary, so no previously out-of-reach ARC-Challenge instance becomes solvable. The RL stages (MLLM Instruct→MPO, MPO→GSPO) remain at zero, consistent with the MMStar results.

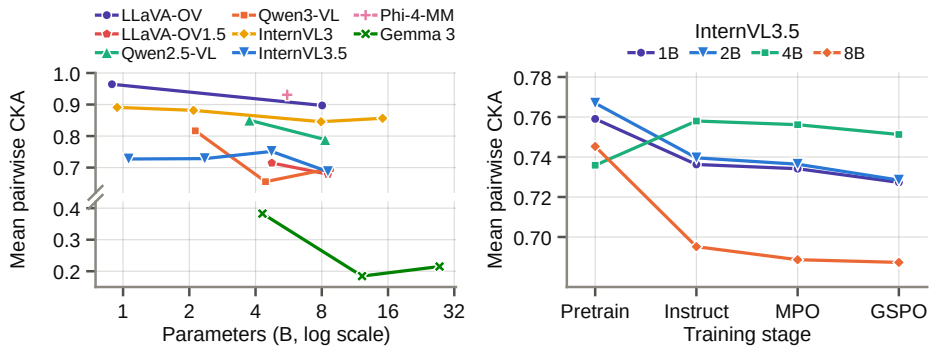


Figure 9: **Layers become less interchangeable with scale and with post-training.** Mean inter-layer linear CKA on MMStar, mean-pooled prefill states, no recursion applied. **Left:** All 21 released checkpoints against parameter count. $\overline{\text{CKA}}$ falls with scale in general, with model family as the primary separating factor. **Right:** InternVL3.5 across its four post-training stages. $\overline{\text{CKA}}$ decreases with more post-training, consistent with the model absorbing recursive depth structure into its default forward pass.

C.2 Inter-Layer Similarity

Block-layer recursion re-executes a route $[s, e]$ and feeds the result back into layer $e{+}1$. Whether this is a small perturbation or a large one depends on how alike the layers inside the route are to the surrounding layers. We measure this directly using linear CKA (Kornblith et al., 2019).

For every checkpoint, we run one prefill forward pass per MMStar instance with no recursion applied, mean-pool each decoder layer's output over the prefill tokens, and compute the $L \times L$ CKA matrix between layers, centering features over the 1,500 instances. The reported scalar is the mean of the strict upper triangle: $\overline{\text{CKA}} = \text{mean}_{i<j}\,\text{CKA}(h_i, h_j)$. High values indicate redundancy along the depth axis, while low values indicate that every layer computes something distinct.

Figure 9 reports two views. Across model scale (Left), $\overline{\text{CKA}}$ falls with parameter count in general, with model family being the primary separating factor. This is consistent with Figure 2 (Bottom): larger models are harder to help with a blindly chosen recursion route. The CKA measurement adds something the density measurement cannot: it is taken on the unmodified forward pass, so it is a property of the representation itself and cannot be explained by the growth of $|\mathcal{R}|$ with L.

Across post-training stages (Right), $\overline{\text{CKA}}$ decreases with more post-training at all scales of InternVL3.5. Layer representations become more distinct as training proceeds, consistent with the model internalizing recursive function into individual layers rather than relying on repeated computation across similar layers.

D Alignment Analysis

D.1 Weight-Space Correlation

Figure 10 extends the weight-space analysis of Figure 5 (Left) to all available transition pairs across InternVL3.5. For completeness, we report both recursion frequency and $\|\Delta W_\ell\|$ (Frobenius norm of the weight change) per layer . This distinguishes our finding from the sparsity result of Mukherjee et al. (2026), who show that RL updates a sparse subnetwork. Our claim concerns *which layers* receive the largest weight changes, not sparsity within those layers. Nonetheless, the two observations are compatible: RL can update a sparse but large-magnitude subnetwork concentrated at mid-layers.

The SFT (Pretrain→Instruct) weight update profiles are shown alongside for comparison. SFT exhibits a qualitatively different pattern: updates concentrate on the last few layers rather than mid-layers, consistent with SFT's role in fixing output format and improving instruction following. This contrast between SFT and RL update profiles provides additional support for the different behaviors observed in the bucket transition matrices (Appendix C.1).



Figure 10: **Weight-space correlation between post-training and recursion, all transitions.** Each panel shows the per-layer weight update norm $\|\Delta W_\ell\|$ alongside the target-set recursion layer frequency r_ℓ for a given transition. The SFT row shows qualitatively different last-layer concentration, contrasting with RL mid-layer concentration.

D.2 Logit and Representation Alignment

Figure 11 extends the logit-space cosine similarity analysis of Figure 5 (Middle) to all available post-training transition pairs across InternVL3.5. For target-set instances, the recursion policy update $\Delta \boldsymbol{r}_\ell$ and the RL policy update $\Delta \boldsymbol{f}_\ell$ show positive cosine similarity at mid-to-late layers, the depth range most directly related to generated output. The control set shows weaker or negative alignment across the same range. This pattern replicates across all SFT and RL transition pairs in model families.

Figure 12 reports the representation-level alignment as described in §5. For each layer ℓ, we compute the cosine similarity between (a) the shift in the default hidden state from the earlier to the later checkpoint and (b) the shift from the default hidden state to the best-recursion-route hidden state at the earlier checkpoint. The target set shows positive alignment at mid-to-late layers, consistent with the logit-space and weight-space results. The control set shows no such pattern.

Both figures confirm across the full set of transitions that post-training absorbs the recursion budget in a directionally specific way: the updates point where recursion was already pointing, and this correspondence is specific to newly acquired instances.

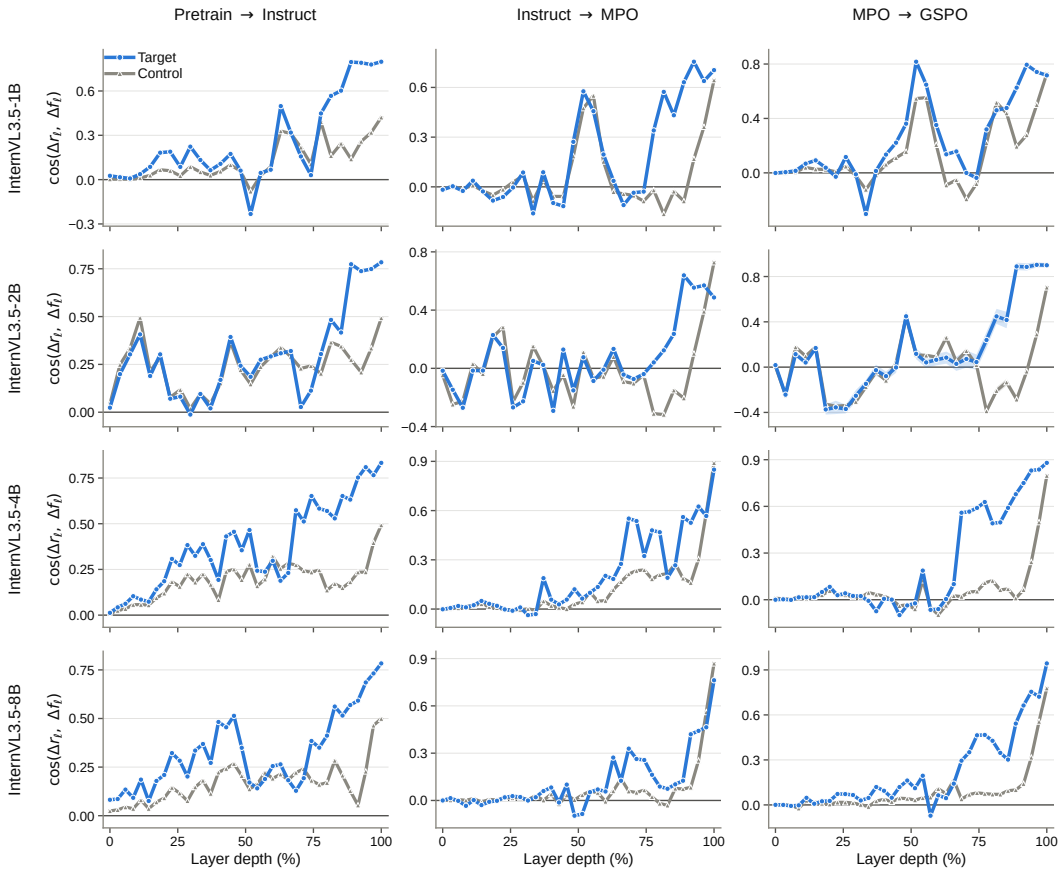


Figure 11: **Logit-space policy alignment between post-training and recursion, all transitions.** Layer-wise cosine similarity between the recursion policy update $\Delta \boldsymbol{r}_\ell$ and the post-training policy update $\Delta \boldsymbol{f}_\ell$, shown separately for target-set and control-set instances. Positive mid-to-late layer alignment for the target set, and weaker or negative alignment for the control set, replicate across all post-training transition pairs in InternVL3.5 model families.

E Additional Analysis

E.1 Modality Asymmetry: LLM vs. ViT Recursion

We apply the same route enumeration to vision-encoder (ViT) blocks instead of LLM decoder layers, using the same MMStar instances. For each model, we construct a ViT recursion family of identical structure (one contiguous route re-executed once over ViT blocks) and compute both oracle measures.

Table 10 reports the results. The LLM decoder consistently provides more recursion headroom than the ViT encoder. On Qwen2.5-VL-7B, the ViT family is *larger* than the decoder family ($529 > 407$), yet its instance-level headroom is 16 points smaller. The asymmetry is therefore a property of where recursion is useful, not of how many actions are available.

Neither model benefits from a fixed ViT-encoder route: the task-level oracle gains are $+0.60$ and $+1.27$, comparable to the decoder but paired with much lower instance-level headroom. Recursion capacity in a multimodal model appears better spent on the decoder, which is a concrete constraint for architectures designed to be recursive by construction.

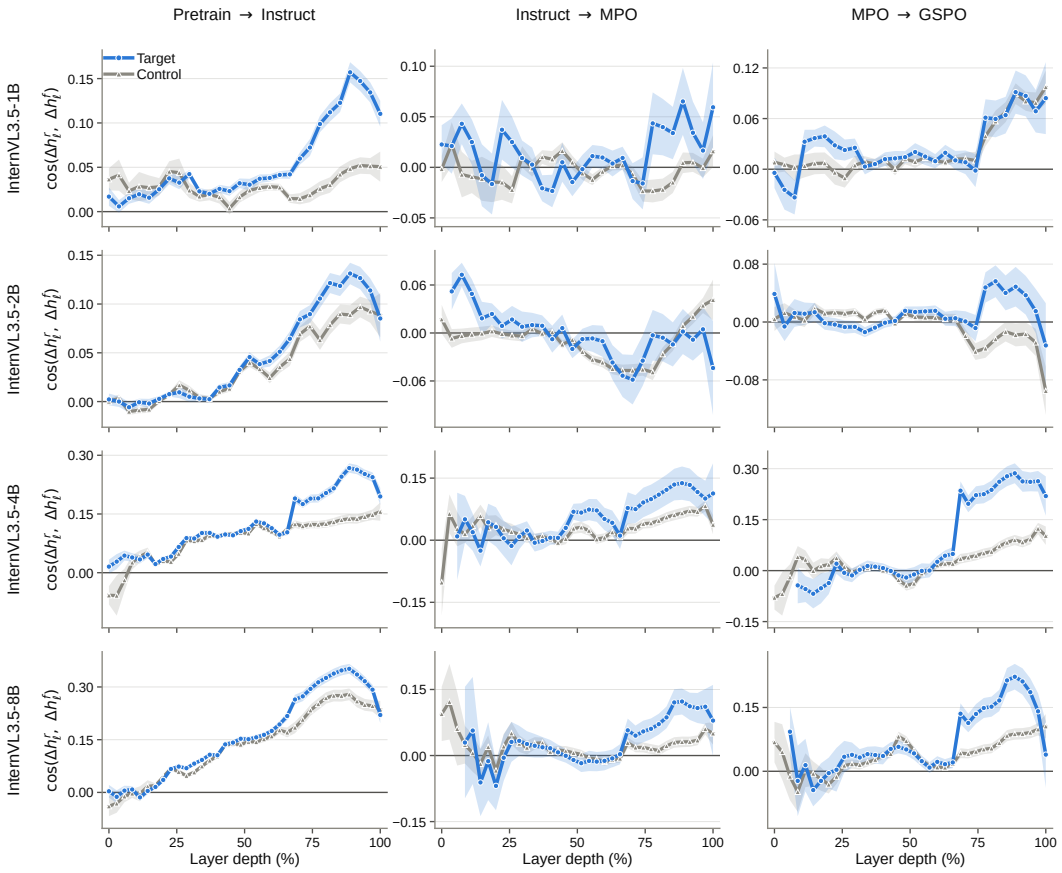


Figure 12: **Representation-level alignment between post-training and recursion, all transitions.** Layer-wise cosine similarity between the hidden-state shift under post-training and the hidden-state shift under the best recursion route at the earlier checkpoint, shown separately for target-set and control-set instances. Positive alignment at mid-to-late layers for the target set supports that post-training absorbs the recursive property and is consistent with the logit-space and weight-space results.

Table 10: **LLM decoder vs. ViT encoder recursion oracles on MMStar**. Both families use the same enumeration strategy: one contiguous route re-executed once, giving $\frac{L(L+1)}{2} + 1$ routes for a stack of depth L. $\text{Oracle}_{\text{task}}$ is the best single fixed non-default route applied uniformly. $\text{Oracle}_{\text{inst}}$ is the per-instance best. Baseline (r_0) is identical for both families since neither changes weights.

		LLM decoder			ViT encoder		
Model	base	$\lvert\mathcal{R}\rvert$	$\text{Oracle}_{\text{task}}$	$\text{Oracle}_{\text{inst}}$	$\lvert\mathcal{R}\rvert$	$\text{Oracle}_{\text{task}}$	$\text{Oracle}_{\text{inst}}$
InternVL3.5-8B	66.87	667	67.80	97.87	301	67.47	88.07
Qwen2.5-VL-7B	61.87	407	62.87	98.33	529	63.14	81.87

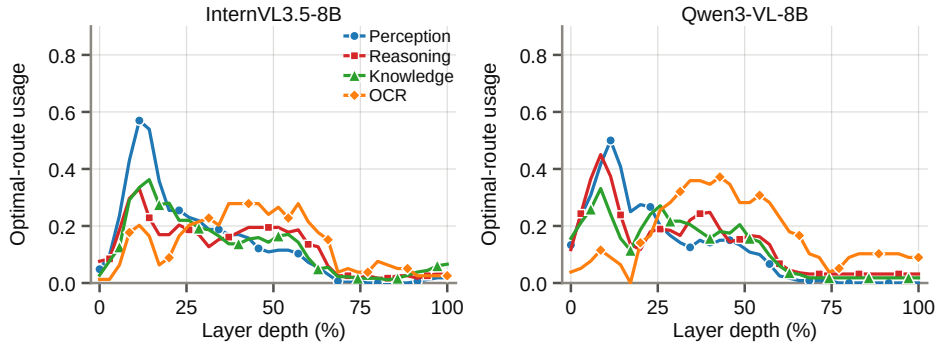


Figure 13: **Recoverable errors favor early-to-middle recursion routes.** For each in-budget instance, the curve marks the fraction of instances whose optimal route covers each layer. Depth is normalized by L so the two panels are directly comparable. **Left:** InternVL3.5-8B. **Right:** Qwen3-VL-8B. Perception, reasoning, and knowledge are MMStar sub-category pairs and OCR is ChartQA-lite.

E.2 Task-Wise Depth Structure

Figure 13 shows, for each MMStar category (perception, knowledge, reasoning) and ChartQA-lite (OCR), the distribution over which layers the best recursion route covers for in-budget instances. For every instance that the default route answers incorrectly but some route in $\mathcal{R}$ answers correctly, we take the correct route with the highest average log-likelihood and mark the layers its route $[s, e]$ spans. The curve is the fraction of such instances whose optimal route covers each layer.

Both models place their usage mass early-to-middle, with peak at 10–20% depth and near-zero mass past 75% layer depth. OCR is the one category whose mass sits consistently deeper (center of mass 45% for OCR vs. 25% for perception on InternVL3.5-8B), matching the per-category routes used for recursion-aware post-training.

This depth ordering aligns with known layer-role structure in MLLMs (Shi et al., 2026; Yu & Lee, 2025): perception errors are most often fixed by early-layer routes, reasoning errors by mid-layer routes, and OCR errors by late-layer routes. This structure is what makes the per-category route selection in §6 principled rather than arbitrary, and it explains why no single route helps in all task categories while per-category routes can.

F Test-Time Routing Reproduction

This section describes the three test-time routing methods evaluated in Table 1, their adaptation to the multimodal setting, and per-benchmark results. We build on the official code of each method rather than re-implementing it. For Dr.LLM, we use the released router, MCTS generator, and training code, and for POLAR, the released predictor and training code. For TFLT, we use the released looped forward pass. Before moving to MLLMs, we reproduce each method in the text-only LLM setting of its original paper and confirm the improvement reported in each paper.

F.1 Methods

TFLT (Chen et al., 2026a). TFLT re-executes a contiguous block $[s, e]$ as a K-stage Runge–Kutta integrator. With their Butcher tableau the update reduces to

$$x_1 \;=\; \beta\, g(x_0) + (1-\beta)\, F^K(x_0), \qquad F(x) \;=\; x + \tfrac{1}{K}\big(g(x) - x\big), \tag{7}$$

where K *damped* Euler steps anchored to the single pass $g(x_0)$ with weight β. Damping is essential: the authors report that undamped repetition degrades performance almost universally. There is no training and no per-instance decision. We keep their recipe with $K = 3$, $\alpha = 1/K$, and window width 4, giving $(K-1)W/L = +29\%$ decoder compute. Their window placement heuristic and β value must be chosen via grid-search. We sweep 13 four-wide windows at stride 2 across the full

stack $\times$ $\beta \in \{0, 0.25, 0.5\}$ (39 configurations) on the same 8k Vero STEM corpus used for the other two methods, and apply the winner fixed to the benchmarks.

Dr.LLM (Heakl et al., 2026). Dr.LLM attaches a small MLP router to every layer that chooses $\{\text{SKIP}, \text{EXECUTE}, \text{REPEAT}\}$ from the pooled hidden state entering that layer. Routers are trained by supervised learning on layer programs found by offline MCTS search, with the backbone frozen. We use their architecture (Linear(d, 128)–GELU–Linear(128, 3) per layer, 8 pooled windows, 7.35M parameters) and their teacher-forced training signal: router sees the state produced by the ground-truth program prefix.

POLAR (Li et al., 2026). POLAR predicts a whole layer program in one shot. A frozen text encoder embeds the question, L learned layer queries cross-attend over the question tokens, and two heads emit a segmentation over layers with a per-segment operation in $\{\text{SKIP}, \text{EXECUTE}, \text{REPEAT}\}$ (segments of width at most 4). The predictor never sees a decoder hidden state, avoiding the train/test prefix mismatch of Dr.LLM's teacher forcing. We use their published hyperparameters (20 epochs, batch 16, lr 10^{-4}, $d_{\text{model}} = 256$, length preference $\beta = 0.05$) with optional biasing terms left at their defaults.

F.2 Adaptation to the Multimodal Setting

Search corpus. In order to train the routers from POLAR and Dr.LLM, we run MCTS over layer programs on 8k instances of Vero's STEM split with 200 simulations per instance, exploration constant 1.8, no early stopping, and memoization of every evaluated program. The reward is the benchmark's own rule-based scorer. The search is run separately in each method's own action space: per-layer $\{\text{skip}, \text{execute}, \text{repeat}\}$ for Dr.LLM and segment-level operations of width ≤ 4 for POLAR. Those two action spaces have some shared corpus but cannot be fully covered by each other. As a result, 72.8% of the programs found in POLAR's space are not expressible in Dr.LLM's, confirming the two searches are not interchangeable.

Recursion under a KV cache. A layer program that skips or repeats layers changes the number of executed layers, so the recursion pass writes into virtual KV-cache slots indexed by position in the program rather than by physical layer. The program is fixed at prefill and replayed at every decode step. We directly adopt their settings.

Router input. Dr.LLM's router pools the hidden state into 8 windows over the sequence. In an MLLM the prefill is mostly image tokens, so equal-width windows place roughly seven windows inside the image and one on the question. We keep their 8 windows to faithfully reproduce their results.

POLAR conditions on question text only and we kept this as published. As an ablation, we also trained POLAR variants that append 1 or 64 pooled ViT tokens to the predictor's key/value sequence. Both routed more dynamically but scored worse on STEM benchmarks (-6.1 and -10.5 points vs. -0.5 for the text-only run), so the published text-only run is reported.

F.3 Per-Benchmark Results

Table 11 breaks Table 1 out by benchmark on two backbones: InternVL3.5-2B and Qwen3VL-2B. The same failure patterns appear on both models.

Dr.LLM degrades substantially on every benchmark, averaging 19.86 and 19.88 on the two backbones. Inspecting its routing decisions reveals the cause: Dr.LLM never selects the default route on any instance (0% on both models) and aggressively skips layers, executing only 25.5 of 28 layers on InternVL3.5-2B and 24.3 of 28 on Qwen3VL-2B. This aggressive deviation from the default path increases the risk of generating degraded outputs, consistent with the low ρ_{fix} leaving too little signal for a per-layer router.

POLAR stays close to the baseline but does not improve it, averaging 37.71 and 31.25. Unlike Dr.LLM, POLAR selects the default route on the majority of instances (86% on InternVL3.5-2B, 82% on Qwen3VL-2B) and its programs are slightly longer than the default (28.6 of 28 layers) because it favors repeats over skips. POLAR therefore fails conservatively by rarely departing from the default, while Dr.LLM fails aggressively by always departing and losing layers.

Table 11: **Per-benchmark results for test-time routing**. POLAR and Dr.LLM are means over 3 seeds. TFLT is deterministic. Bold marks the best result per row.

Benchmark	Default	POLAR	Dr.LLM	TFLT
InternVL3.5-2B				
MathVision	19.37	**21.71**	2.52	20.37
MathVista	60.00	57.87	27.10	**60.70**
MMMU-val	47.18	45.07	25.00	**47.33**
VisualPuzzles	**26.37**	26.17	24.83	26.14
Average	38.23	37.71	19.86	**38.64**
Qwen3VL-2B				
MathVision	**7.24**	6.58	0.77	6.58
MathVista	**59.90**	53.10	29.10	56.70
MMMU-val	39.22	36.08	24.67	**40.22**
VisualPuzzles	29.28	29.25	24.97	**30.57**
Average	**33.91**	31.25	19.88	33.52

TFLT applies a single fixed configuration to all instances without per-instance routing decisions, which prevents the kind of collapse seen in Dr.LLM. It produces small gains on individual benchmarks (e.g. MathVista on InternVL3.5-2B, MMMU-val and VisualPuzzles on Qwen3VL-2B) but the aggregate accuracy remains at or slightly below the baseline on both models.

G Recursion-Aware Post-Training

G.1 Implementation Details

Vero categories. Vero-600k (Sarch et al., 2026) labels every instance with one of six abilities, which we use as the task categories throughout: STEM, Chart & OCR, Spatial & Action, Knowledge & Recognition, Grounding, Counting & Search, and Captioning & IF. The abilities map one-to-one onto the evaluation categories of Appendix G.3. Each category split contributes 100k instances to the per-category runs and the single-route run of Table 16 trains on the full corpus. Every instance carries a question, an image, a ground-truth answer, and a *reward type* that selects its scorer. The five non-captioning abilities are fully rule-scored while Captioning & IF is the only ability that needs a model judge.

Reward. The reward is the outcome of the instance's own scorer in $[0, 1]$ following Sarch et al. (2026), with no format or length shaping term. Rule-scored types cover exact and normalized string match (after Unicode, article, and number-word normalization), multiple choice with option-letter extraction, numeric match with a relative tolerance, unordered lists of strings or numbers, grounding (box IoU above 0.5), clicking (predicted point inside the target box), and web actions (action type plus its arguments). The Captioning & IF ability uses three types: rule-based instruction following, scored as the fraction of the instance's constraints that the response satisfies; caption quality, scored by an external Qwen3-32B judge that sees the question, the response, and the reference answer and returns a 1–10 score rescaled to $[0, 1]$ and their weighted combination with equal weight on the two parts. The same scorers produce the held-out validation reward used for checkpoint selection.

GSPO objective. We follow GSPO (Zheng et al., 2025) without modification. For each prompt x we draw a group of G rollouts $\{y_i\}$ from the current policy and centre their rewards within the group,

$$A_i \;=\; \frac{R(x, y_i) - \operatorname{mean}_j R(x, y_j)}{\operatorname{std}_j R(x, y_j)}, \tag{8}$$

so the group itself is the baseline and no value network is needed. The importance ratio is taken at the *sequence* level and length-normalized,

$$s_i \;=\; \Big(\frac{\pi_\theta(y_i \mid x)}{\pi_{\theta_{\text{old}}}(y_i \mid x)}\Big)^{1/|y_i|}, \qquad \mathcal{L} \;=\; -\,\mathbb{E}\Big[\tfrac{1}{G}\textstyle\sum_i \min\big(s_i A_i,\; \text{clip}(s_i, 1-\varepsilon_{\text{low}}, 1+\varepsilon_{\text{high}})\, A_i\big)\Big], \tag{9}$$

which keeps one ratio per response rather than one per token and is why the clipping range is narrow (Table 12). There is no reference model, no KL penalty, and no entropy bonus. Recursion-aware training changes only the forward pass: rollouts and the policy update both run through the fixed route $r_{s,e}$, so π_θ is the recursive policy on both sides of the ratio and the update stays on-policy. The baseline run is the same loop with the default route r_0.

G.2 Training Hyperparameters

Table 12 lists the GSPO training hyperparameters used for both standard and recursion-aware post-training. All settings are shared between the two regimes. The only difference is the addition of a fixed recursion route to the training forward pass.

Table 12: **GSPO training hyperparameters.**

Hyperparameter	**Value**
Trainable parameters	Full-parameter in LLM (ViT frozen)
Optimizer	AdamW ($\beta = 0.9, 0.999$)
Weight decay	0
Learning rate	$\{1 \times 10^{-6}, 1 \times 10^{-5}\}$
LR schedule	Constant
Clip lower (ε_{low})	0.0003
Clip upper ($\varepsilon_{\text{high}}$)	0.0004
KL coefficient	0
Rollouts per prompt (G)	8
PPO mini-batch size	128
Max new tokens	64 (short answer) / 512 (captioning & IF, average)
Context length	4096
Epochs	1
Reward	Vero rule scorer + Qwen3-32B judge for captioning & IF

G.3 Held-Out Evaluation Benchmarks

Table 13 lists the benchmark suite for each category. No instance from any benchmark below participates in the 2k Vero-600k subset used for route selection, and all benchmarks are drawn from sources disjoint from the Vero-600k training splits. Each category is scored as the average over its own benchmarks.

G.4 Route Utility Heatmap

Figure 14 shows the per-category route utility heatmaps on a 2k-instance, validation subset of Vero-600k with InternVL3.5-2B. Each cell (s, e) shows the accuracy of recursion route $r_{s,e}$ for instances of a given category.

The per-category panels reveal clear depth structure: perception and grounding tasks favor early-layer routes, spatial and knowledge tasks favor mid-layer routes, and STEM tasks favor a narrow band around layer 19. The averaged heatmap has no improving route, confirming that per-category structure is invisible in aggregate. These heatmaps are the basis for the top-1 per-category route selection used in recursion-aware post-training.

G.5 Generalization to Other Models

Table 14 applies the protocol of Table 2 to three more backbones: InternVL3.5-4B, Qwen2.5VL-3B-Instruct, and Qwen3VL-2B-Instruct. For each backbone, we build its own route utility map on the same 2k Vero-600k subset, take the top-1 route per category, train GSPO on 100k instances per category, and evaluate on the held-out benchmarks of Appendix G.3. The selected routes differ across backbones, so what transfers is the selection procedure rather than a fixed set of routes.

Recursion-aware GSPO outperforms standard GSPO in 14 of the 18 backbone–category cells. It wins five of six categories on InternVL3.5-4B and Qwen2.5VL-3B, as on InternVL3.5-2B, and four

Table 13: **Evaluation benchmarks per Vero category.**

Category	#	Benchmarks
STEM	4	MathVision (Wang et al., 2024a), MathVista (Lu et al., 2024), MMMU-val (Yue et al., 2024), VisualPuzzles (Song et al., 2025)
Chart & OCR	5	CharXiv-reasoning (Wang et al., 2024c), ChartMuseum (Tang et al., 2026), InfographicVQA (Mathew et al., 2022), OCRBench (Liu et al., 2024b), DocVQA-val (Mathew et al., 2021)
Spatial & Action	5	BLINK (Fu et al., 2024), EmbSpatial-Bench (Du et al., 2024), CV-Bench (Tong et al., 2024), MMSI-Bench (Yang et al., 2026), VLM-sAreBlind (Rahmanzadehgervi et al., 2024)
Knowledge & Recognition	6	SimpleVQA-en (Cheng et al., 2025), FVQA (Wu et al., 2025), MM-Vet v2 (Yu et al., 2024), MMBench-en-dev (Liu et al., 2024a), SEED-Bench (Li et al., 2024b), POPE (Li et al., 2023)
Grounding, Counting & Search	5	CountBenchQA (Beyer et al., 2024), CountQA (Tamarapalli et al., 2025), V*Bench (Wu & Xie, 2024), ScreenSpot (Cheng et al., 2024), HR-Bench-4k (Wang et al., 2025b)
Captioning & IF	3	MM-MT-Bench (Agrawal et al., 2024), MIA-Bench (Qian et al., 2025), MM-IFEval (Ding et al., 2025)
Total	28	

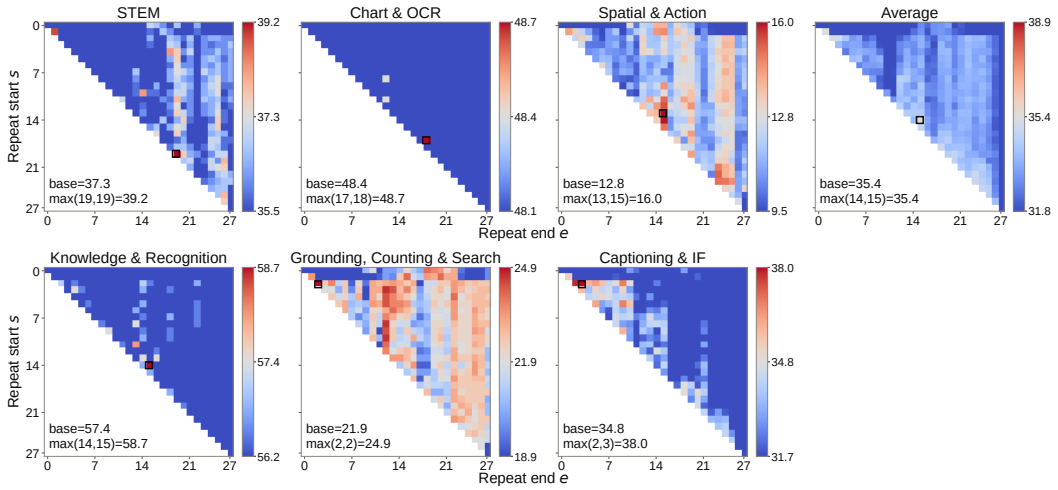


Figure 14: **Per-category route utility heatmaps on Vero-600k subset** (InternVL3.5-2B, 2k instances). Each cell (s, e) shows the accuracy of route $r_{s,e}$. The baseline accuracy and the top-1 route are annotated per panel. Per-category structure is clearly visible, while no single route improves accuracy in Average.

of six on Qwen3VL-2B. The largest gains are on Captioning & IF for InternVL3.5-4B ($+3.27$) and Qwen3VL-2B ($+4.62$), and on STEM for Qwen2.5VL-3B ($+1.52$). Knowledge & Recognition is again the exception, with standard GSPO ahead on all three backbones by 0.24 to 1.63 points, consistent with the main result. The only other loss is Chart & OCR on Qwen3VL-2B (-0.89), and the Chart & OCR gain on Qwen2.5VL-3B ($+0.08$) is within one standard error. As on InternVL3.5-2B, applying the selected route without training falls below the untrained baseline in 15 of the 18 cells, so the gains come from training through the route rather than from the added depth.

G.6 Extra Results

Table 15 expands the per-category results of Table 2 to every individual benchmark, Table 16 reports an additional average-route experiment, and Figure 15 examines cross-category transfer. In all

Table 14: **Recursion-aware post-training on additional backbones.** Same protocol as Table 2. *Route*: top-1 per-category route on 2k training subset. *Rec. Baseline*: recursion route applied at inference without training. *Rec. GSPO*: recursion-aware post-training (mean ± standard error over 3 seeds). Each cell averages over that category's held-out benchmarks. Best accuracy per row in bold.

Category	Route	Baseline	Rec. Baseline	GSPO	Rec. GSPO
InternVL3.5-4B					
STEM	(15,30)	47.59	46.02	$47.71_{\pm 0.23}$	**48.30**$_{\pm 0.17}$
Chart & OCR	(11,14)	**61.20**	60.54	$60.53_{\pm 0.24}$	$61.06_{\pm 0.29}$
Spatial & Action	(10,15)	53.70	51.55	$54.08_{\pm 0.55}$	**55.91**$_{\pm 0.61}$
Knowledge & Recognition	(17,20)	62.09	62.23	**62.32**$_{\pm 0.10}$	$62.08_{\pm 0.12}$
Grounding, Counting & Search	(13,19)	60.11	59.09	$60.58_{\pm 0.09}$	**60.92**$_{\pm 0.13}$
Captioning & IF	(11,12)	66.93	65.82	$70.57_{\pm 0.39}$	**73.84**$_{\pm 0.45}$
Qwen2.5VL-3B-Instruct					
STEM	(16,24)	40.83	41.04	$40.77_{\pm 0.19}$	**42.29**$_{\pm 0.21}$
Chart & OCR	(20,24)	59.88	59.36	$60.10_{\pm 0.32}$	**60.18**$_{\pm 0.35}$
Spatial & Action	(12,14)	48.31	46.32	$49.08_{\pm 0.53}$	**49.89**$_{\pm 0.49}$
Knowledge & Recognition	(13,13)	62.42	61.38	**62.99**$_{\pm 0.17}$	$62.69_{\pm 0.21}$
Grounding, Counting & Search	(20,26)	47.91	47.27	$49.60_{\pm 0.19}$	**50.37**$_{\pm 0.15}$
Captioning & IF	(8,11)	59.83	58.10	$60.35_{\pm 0.37}$	**61.84**$_{\pm 0.38}$
Qwen3VL-2B-Instruct					
STEM	(2,15)	35.43	33.87	$35.29_{\pm 0.23}$	**36.12**$_{\pm 0.29}$
Chart & OCR	(19,19)	58.09	57.71	**58.40**$_{\pm 0.36}$	$57.51_{\pm 0.29}$
Spatial & Action	(12,15)	52.13	53.39	$53.57_{\pm 0.41}$	**54.60**$_{\pm 0.37}$
Knowledge & Recognition	(14,16)	60.98	59.95	**61.92**$_{\pm 0.19}$	$60.29_{\pm 0.13}$
Grounding, Counting & Search	(13,13)	50.38	48.61	$50.71_{\pm 0.08}$	**52.36**$_{\pm 0.13}$
Captioning & IF	(5,9)	63.53	56.71	$64.27_{\pm 0.29}$	**68.89**$_{\pm 0.35}$

results, *Baseline* is the released InternVL3.5-2B on its default route r_0, *Rec. Baseline* applies the recursion route at inference without training, *GSPO* is standard GSPO on identical data and schedule, and *Rec. GSPO* is our recursion-aware GSPO.

Per-category run. Table 15 reports the full benchmark breakdown for the six independent per-category checkpoints. Each checkpoint is trained on one 100k Vero-600k category split through that category's top-1 route and evaluated only on its own category's held-out benchmarks. The route in each category header is the recursion block baked into that particular checkpoint's training forward pass, and no single model produces the whole column. Rec. GSPO outperforms Standard GSPO on 5 of 6 categories, with the largest gains on Captioning & IF ($+3.12$), Spatial & Action ($+2.39$), and STEM ($+1.88$). The one exception is Knowledge & Recognition, where Standard GSPO leads by $+0.67$.

Average-route run. Table 16 reports results for a single checkpoint trained on the full 600k corpus through the average top-1 route $r_{14,15}$ and evaluated on all 28 benchmarks. This is the directly comparable setting to a one-model-for-everything recipe. Standard GSPO outperforms Rec. GSPO on average (54.07 vs. 53.33), with the gap concentrated in Grounding, Counting & Search (-4.25) and Captioning & IF (-1.51). This is consistent with the finding that no single route improves all categories (Figure 14): the average top-1 route $r_{14,15}$ benefits mid-layer tasks (Spatial & Action: $+1.86$, Chart & OCR: $+0.48$) but hurts categories whose optimal routes lie at different depths.

Cross-category transfer. Figure 15 evaluates each of the six per-category checkpoints of Table 15 on all six category suites and reports the change relative to the untrained baseline. Each Rec. GSPO checkpoint keeps its training route at evaluation, since the route is part of the trained model. The two methods trade in-category gains against transfer. Standard GSPO transfers broadly: 25 of its 30 off-diagonal cells are positive, and their mean of $+0.8$ points is about half its mean in-category gain of $+1.5$. Rec. GSPO instead concentrates its gains on the training category. Its in-category gain averages $+2.9$ points, and in five of the six evaluation categories the checkpoint trained on that

Table 15: **Per-benchmark results, per-category run** (InternVL3.5-2B, L=28). Six independent checkpoints, one per Vero category. Each is trained on that category's 100k split through the category top-1 route and evaluated only on that category's held-out benchmarks. Best accuracy per row in bold.

	No post-training		Post-trained	
Benchmark	**Baseline**	**Rec. Baseline**	**GSPO**	**Rec. GSPO**
STEM (route $r_{19,19}$)				
MathVision	19.37	18.71	20.39	**26.64**
MathVista	60.00	59.00	**60.90**	**60.90**
MMMU-val	47.18	47.84	47.67	**48.44**
VisualPuzzles	26.37	26.20	27.08	**27.57**
Category Avg	38.23	37.94	39.01	**40.89**
Chart & OCR (route $r_{17,18}$)				
CharXiv$_{Reason}$	28.80	28.00	31.50	**33.90**
ChartMuseum	17.80	17.60	18.30	**20.30**
InfographicVQA	**63.49**	62.60	63.13	62.98
OCRBench	79.60	79.20	**79.80**	79.00
DocVQA-val	88.18	88.26	**88.43**	**88.43**
Category Avg	55.57	55.13	56.23	**56.92**
Spatial & Action (route $r_{13,15}$)				
BLINK	49.04	51.66	48.53	**51.88**
EmbSpatial	59.42	62.01	60.71	**62.88**
CV-Bench	71.53	72.55	72.86	**75.44**
MMSI-Bench	25.00	**25.90**	25.20	25.60
VLMsAreBlind	32.40	36.07	38.54	**42.07**
Category Avg	47.48	49.64	49.18	**51.57**
Knowledge & Recognition (route $r_{14,15}$)				
SimpleVQA$_{En}$	36.82	35.74	**37.51**	36.33
FVQA	14.94	13.67	**15.06**	13.78
MM-Vet v2	65.34	64.31	**66.09**	64.76
MMBench-en-dev	77.75	77.66	77.75	**77.84**
SEED-Bench	71.50	70.39	**70.09**	69.86
POPE	87.70	86.77	**88.80**	88.74
Category Avg	59.01	58.09	**59.22**	58.55
Grounding, Counting & Search (route $r_{2,2}$)				
CountBenchQA	68.02	69.25	72.10	**73.73**
CountQA	15.39	15.19	15.12	**16.23**
VStarBench	68.57	66.16	**70.32**	69.89
ScreenSpot	55.87	53.29	61.56	**63.99**
HR-Bench 4k	48.12	48.88	47.88	**48.75**
Category Avg	51.19	50.55	53.40	**54.52**
Captioning & IF (route $r_{2,3}$)				
MM-MT-Bench	44.63	44.05	46.36	**48.59**
MIA-Bench	81.02	80.30	81.75	**87.80**
MM-IFEval	54.63	53.58	62.63	**63.71**
Category Avg	60.09	59.31	63.58	**66.70**

category is the best of the six. Off the diagonal, Rec. GSPO transfers little. Its off-diagonal cells average close to zero and 14 of the 30 are negative, with the largest losses on Captioning & IF and Spatial & Action. STEM is the exception and improves under every Rec. GSPO checkpoint, by $+0.5$ to $+2.7$ points. Recursion-aware post-training therefore specializes the model to the category whose route it is trained through, which is consistent with the weaker result of the single average-route model in Table 16.

G.7 Ablations

Table 17 reports ablations for recursion-aware post-training. All runs use InternVL3.5-2B on the STEM category (100k instances) with identical trainable parameters.

Table 16: **Per-benchmark results, average-route run** (InternVL3.5-2B, L=28). A single checkpoint trained on the full Vero-600k corpus through route $r_{14,15}$ and evaluated on all 28 held-out benchmarks. Standard GSPO is also trained on full Vero-600k corpus with default route r_0. Column definitions follow Table 15.

	No post-training		Post-trained	
Benchmark	**Baseline**	**Rec. Baseline**	**GSPO**	**Rec. GSPO**
STEM	38.23	39.66	**40.58**	40.22
Chart & OCR	55.57	55.03	55.54	**56.02**
Spatial & Action	47.48	48.20	49.04	**50.90**
Knowledge & Recognition	59.01	58.09	**60.57**	59.94
Grounding, Counting & Search	51.19	48.06	**53.63**	49.38
Captioning & IF	60.09	59.94	**65.04**	63.53
Overall Avg (28)	52.09	51.55	**54.07**	53.33



Figure 15: **Cross-category transfer of the per-category checkpoints** (InternVL3.5-2B). Each column is a checkpoint trained on one Vero category, and each row is a held-out evaluation category. Each cell is the category accuracy minus that of the untrained baseline and the diagonal is the in-category result of Table 15. **Left:** Standard GSPO. **Right:** Recursion-aware GSPO, with each checkpoint evaluated through its own training route.

(a) Compute and hyperparameter controls. Standard GSPO is robust to learning rate: the gap between LR=1×10^{-5} and 1×10^{-6} is only 0.22 points. Rec. GSPO shows a larger sensitivity (0.79 points), likely because the recursion block requires stronger updates to adapt to the recursive forward route. Doubling the training time of Standard GSPO (2 epochs) yields only $+0.21$ over the 1-epoch baseline and still falls 1.67 points below Rec. GSPO at $1.06\times$ wall-clock cost. The gain from Rec. GSPO is therefore not a compute effect but an inductive bias provided by the explicit recursion block.

(b) Route selection within STEM. The improvement depends strongly on which route is chosen. The Top-1 route ($r_{19,19}$) gains $+1.88$ over Standard GSPO. Top-3 ($r_{10,14}$) remains positive at $+0.58$, while Top-2 ($r_{1,1}$) falls slightly below Standard GSPO (-0.30). The underperformance of Top-2 is likely due to the small sample size used for route selection: only 333 instances from the Vero utility map cover the STEM category, so the ranking among nearby routes is noisy. A larger selection set would more reliably identify the useful depth band. Nonetheless, both Top-2 and Top-3 outperform the untrained baseline, indicating that these routes are not harmful to training. In contrast, a random route ($r_{6,12}$) degrades accuracy by -7.56 points, falling well below the untrained baseline. This confirms that the depth placement of the recursion block is critical rather than the additional compu-

Table 17: **Recursion-aware post-training ablations.** InternVL3.5-2B trained on the STEM category of Vero-600k (100k instances), evaluated as the average over STEM benchmarks. Δ and *Train time* are both relative to Standard GSPO (LR=1×10^{-5}, 1 epoch). Best result in bold.

Method	Route	STEM	Δ	Train time
(a) Compute and hyperparameter controls				
Baseline (no training)	r_0	38.23	—	—
Standard GSPO (LR=1×10^{-5})	r_0	39.01	0.00	$1.00\times$
Standard GSPO (LR=1×10^{-6})	r_0	38.79	-0.22	$1.00\times$
Standard GSPO (LR=1×10^{-5}, 2 epochs)	r_0	39.22	$+0.21$	$2.00\times$
Rec. GSPO (LR=1×10^{-5}, ours)	$r_{19,19}$	**40.89**	$+1.88$	$1.06\times$
Rec. GSPO (LR=1×10^{-6})	$r_{19,19}$	40.10	$+1.09$	$1.06\times$
(b) Route selection within STEM				
Rec. GSPO, STEM Top-2	$r_{1,1}$	38.71	-0.30	$1.06\times$
Rec. GSPO, STEM Top-3	$r_{10,14}$	39.59	$+0.58$	$1.23\times$
Rec. GSPO, Random route	$r_{6,12}$	31.45	-7.56	$1.31\times$
(c) Cross-category route transfer to STEM				
Rec. GSPO, Chart & OCR Top-1	$r_{17,18}$	38.46	-0.55	$1.09\times$
Rec. GSPO, Spatial & Action Top-1	$r_{13,15}$	38.60	-0.41	$1.14\times$

tation alone. Training time scales with block size: $1.06\times$ for single-layer routes, $1.23\times$ for a 5-layer block, and $1.31\times$ for the 7-layer random route.

(c) Cross-category route transfer. We apply the Top-1 routes from Chart & OCR ($r_{17,18}$) and Spatial & Action ($r_{13,15}$) to STEM training. Neither outperforms Standard GSPO (-0.55 and -0.41), though both remain above the untrained baseline. These routes are not harmful to training but lack the specialization of the STEM Top-1 route, which targets the depth band most relevant to mathematical and scientific reasoning. This reinforces the per-ability depth structure observed in Figure 14 and the finding that no single route improves all task categories (Table 16).