TA-IBAR: TASK-ADAPTIVE INFERENCE BLOCK ATTENTION RESIDUALS

Anonymous authors
Paper under double-blind review

ABSTRACT

Attention residuals (AttnRes) improve information integration across network depth, but Full AttnRes incurs increasing memory and communication costs as historical representations accumulate. Block AttnRes reduces these costs by compressing consecutive layers into block-level sources, yet uses a fixed depth partition across tasks. We propose Task-Adaptive Inference Block Attention Residuals (TA-IBAR), which derives a task-specific contiguous partition from residual representations using centred Linear CKA and structurally constrained dynamic programming. The partition remains fixed during post-training, while the backbone and lightweight task-specific routing parameters are jointly optimised. At inference time, a lightweight probe selects the corresponding task-adapted configuration. Experiments span training-from-scratch and post-training settings, model sizes from 220M to 14B, and network depths from 28 to 40. Across Qwen3 models from 1.7B to 14B, TA-IBAR achieves the lowest validation loss in all eight Math and Multi-hop settings, including reductions from 0.4810 to 0.4629 on 14B Math and from 9.7513 to 8.9869 on 14B Multi-hop over fixed Block AttnRes. Across five additional model families, TA-IBAR outperforms fixed Block AttnRes in all 10 model–task settings and Full AttnRes in 8 of 10 while retaining the memory efficiency of blockwise residual access.

1 INTRODUCTION

Residual connections are a fundamental component of modern deep networks, enabling stable optimisation by propagating information and gradients across depth (He et al., 2016). In transformers, the residual stream integrates representations across depth through fixed additive accumulation, requiring each layer to operate on a single state that already merges all preceding transformations (Zhang, 2026; Zheng et al., 2026). Recent work has consequently explored different mechanisms for depth-wise information flow, ranging from weighted cross-layer aggregations (Pagliardini et al., 2024), multi-stream residual representations (Zhu et al., 2025; Xie et al., 2026) to attention residuals (AttnRes) (Kimi Team et al., 2026), which explicitly formulates residual aggregation as attention over network depth.

Full AttnRes provides each transformation with selective access to all preceding residual sources, offering a fine-grained mechanism for retrieving information across depth (Jia et al., 2026). However, this flexibility requires maintaining an increasing number of representations as the network grows (Team et al., 2026; Su, 2026). Block AttnRes addresses this issue by compressing consecutive transformations into block-level representations and performing depth-wise attention over a smaller set of sources (Cheng et al., 2026). The resulting reduction in memory and communication makes AttnRes substantially more practical at scale (Kimi Team et al., 2026). Yet this efficiency assumes a fixed partition of network depth across all inputs and downstream tasks.

This one-size-fits-all assumption about partitioning network depth is restrictive. Downstream tasks can rely on different subsets and combinations of representations across depth (Wu et al., 2024). A partition suited to one task may be too coarse for another, merging intermediate representations that remain task-relevant, while a uniformly fine partition preserves more sources at higher memory and communication cost. Existing Block AttnRes (Wang, 2026; Fan et al., 2026) therefore couples two design decisions: *how depth should be compressed* and *how the resulting depth sources should be routed*. The routing is learned, while the partition remains a global architectural hyperparameter.

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.

We introduce **Task-Adaptive Inference Block AttnRes (TA-IBAR)**, which turns depth partitioning from a fixed architectural choice into a task-specific post-training decision. We first train with Full AttnRes to expose individual residual sources across depth. For each downstream task, we analyse residual representations from the transferred backbone using centred Linear CKA (Kornblith et al., 2019) and apply structurally constrained dynamic programming to group consecutive Transformer layers into a task-specific partition. The partition remains fixed during post-training, while the backbone and task-specific routing parameters are jointly optimised from a common pretrained checkpoint.

At inference time, an early probe selects the task-specific configuration, and the input is recomputed from layer 0 using the corresponding partition and routing parameters. TA-IBAR therefore adapts the granularity of depth-wise access together with routing over the resulting residual sources.

Our contributions are threefold:

- We identify the *fixed-partition assumption* of Block AttnRes and formulate depth compression as a task-dependent structural adaptation problem.
- We introduce TA-IBAR, which discovers task-specific contiguous depth partitions from residual representations and learns lightweight routing over the resulting structures from a common pretrained backbone checkpoint.
- We evaluate TA-IBAR across training-from-scratch and post-training settings, multiple model scales and depths, five additional backbone families, and three task domains.

2 Related Work

Residual connections and depth-wise information flow. Residual connections are fundamental to the optimisation of deep neural networks, providing an identity path that stabilises gradient propagation across depth (He et al., 2016). In modern Transformers, PreNorm further improves optimisation stability by placing normalisation before each transformation (Xiong et al., 2020), but the residual stream still aggregates preceding transformations through fixed additive updates. A number of approaches have therefore revisited how information should propagate across depth. ReZero (Bachlechner et al., 2020), LayerScale (Touvron et al., 2021), and DeepNorm (Wang et al., 2022) introduce learned or carefully scaled residual paths, while Hyper-Connections (Zhu et al., 2025) and its stabilised variants (Xie et al., 2026) expand the residual state into multiple interacting streams. These methods improve the conditioning or capacity of depth-wise recurrence; each layer still operates primarily on a recurrently accumulated state and does not explicitly select individual earlier representations.

Cross-layer aggregation and Attention Residuals. A complementary line of work provides direct access to representations from multiple depths. DenseNet (Huang et al., 2017) establishes dense cross-layer connectivity through feature concatenation, while ELMo (Peters et al., 2018) and DenseFormer (Pagliardini et al., 2024) learn weighted combinations of representations from different layers. More recent methods introduce input-dependent cross-layer interactions, including MRLA (Fang et al., 2023), MUDDFormer (Xiao et al., 2025), LAuReL (Menghani et al., 2025), and related depth-routing mechanisms. Attention Residuals (AttnRes) (Kimi Team et al., 2026) provides a particularly direct formulation by interpreting residual aggregation as attention over the depth dimension: Full AttnRes applies softmax attention over all preceding residual sources, whereas Block AttnRes compresses consecutive layers into block-level representations to reduce memory and communication costs. However, the block structure in Block AttnRes is prescribed globally and remains fixed across inputs and downstream tasks. This implicitly assumes that a single depth partition provides an appropriate abstraction granularity for all task distributions.

Task-specific structural adaptation. TA-IBAR extends this line of work by adapting which residual sources remain explicitly available for each task. Starting from a fine-grained Full AttnRes backbone, it estimates task-specific similarities between layer residuals using representation similarity analysis (Kornblith et al., 2019), discovers a contiguous depth partition, and learns routing over the resulting structure. This separates Transformer computation, the granularity of depth compression, and routing over the compressed residual sources.

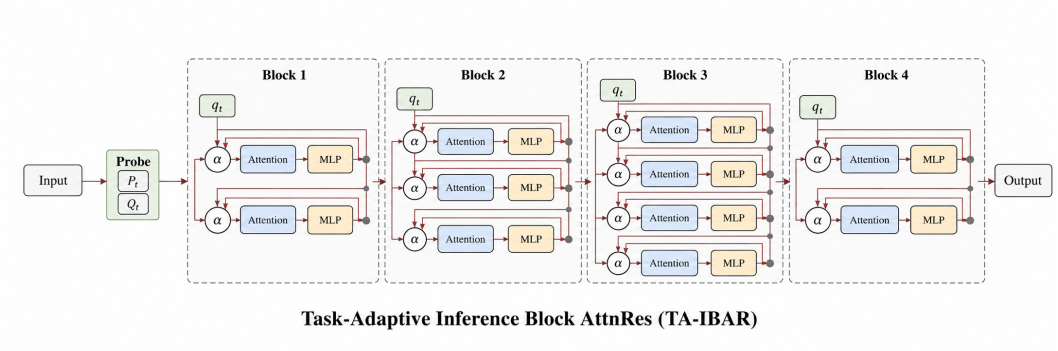

Figure 1: Overview of Task-Adaptive Inference Block AttnRes (TA-IBAR). A lightweight probe selects a task-specific configuration comprising a depth partition and routing parameters, enabling task-dependent residual routing over compact block-level representations.

3 METHODOLOGY

3.1 OVERVIEW

We propose **Task-Adaptive Inference Block AttnRes (TA-IBAR)**, which separates fine-grained depth modelling from task-specific structural adaptation. We first use Full Attention Residuals (Full AttnRes) (Kimi Team et al., 2026) to expose individual residual sources across network depth. For each downstream task, we then discover a contiguous partition of Transformer blocks from the residual representations produced by the transferred backbone. The discovered partition determines which depth regions are compressed into block-level sources, while task-specific routing parameters determine how those sources are combined during subsequent computation. Figure 1 summarises this workflow.

For task t, we denote the resulting routing bank by $\mathcal{B}_t = (P_t, Q_t)$, where P_t is the discovered depth partition and Q_t contains the task-specific pseudo-queries and key-normalisation parameters used for depth routing. Each task starts from the same pretrained backbone checkpoint. During post-training, P_t remains fixed, while the backbone and Q_t are optimised jointly.

3.2 FINE-GRAINED FULL ATTNRES PRE-TRAINING

Consider a Transformer with hidden dimension d. Let $v_0 = e_0 \in \mathbb{R}^d$ denote the token embedding and $v_i \in \mathbb{R}^d$ the residual contribution produced by preceding transformation i. At routing site l, Full AttnRes keeps the source set $\mathcal{V}_l = \{v_0, \ldots, v_{l-1}\}$ individually accessible. Following Kimi Team et al. (2026), each source is mapped to a normalised key $k_i = \mathrm{RMSNorm}(v_i)$ (Zhang & Sennrich, 2019), and a learned pseudo-query $q_l \in \mathbb{R}^d$ assigns attention across depth:

$$a_{l,i} = \frac{\exp(q_l^\top k_i)}{\sum_{j=0}^{l-1} \exp(q_l^\top k_j)}, \quad h_l = \sum_{i=0}^{l-1} a_{l,i} v_i, \tag{1}$$

where $a_{l,i}$ is the depth-routing weight assigned to source v_i, and h_l is the aggregated representation supplied to the current transformation.

The model is trained with the standard causal language-modelling objective over its target-token positions. After this stage, we write the learned parameters as $\Theta^*_{\mathrm{Full}} = (\theta^{(0)}_{\mathrm{backbone}}, \Psi^*_{\mathrm{Full}})$, where $\theta^{(0)}_{\mathrm{backbone}}$ denotes the transferred Transformer parameters and Ψ^*_{Full} the Full-AttnRes-specific routing state. We retain $\theta^{(0)}_{\mathrm{backbone}}$, discard Ψ^*_{Full}, and initialise the task-specific routing parameters independently.

3.3 TASK-SPECIFIC STRUCTURAL DISCOVERY

Structural discovery determines how consecutive Transformer blocks should be compressed for a downstream task. Let the transferred Transformer contain L blocks, and let $h_x^{(l)}$ denote the hidden representation after block l for discovery example x. For block $l \in \{1, \dots, L\}$, its residual contribution is $v_{x,l} = h_x^{(l)} - h_x^{(l-1)}$. For task t, we collect the valid-token residual vectors from all discovery examples into $V_{t,l} \in \mathbb{R}^{T_t \times d}$, where T_t is the total number of valid tokens used for discovery. We centre each matrix as $\widetilde{V}_{t,l} = V_{t,l} - \mathbf{1}\mu_{t,l}^\top$, where $\mathbf{1} \in \mathbb{R}^{T_t}$ is the all-ones vector and $\mu_{t,l} = T_t^{-1} \sum_{r=1}^{T_t} V_{t,l,r}$ is the mean residual vector for block l.

We measure the task-specific similarity between blocks i and j using centred Linear CKA (Kornblith et al., 2019):

$$S_t(i,j) = \frac{\left\| \widetilde{V}_{t,i}^\top \widetilde{V}_{t,j} \right\|_F^2}{\left\| \widetilde{V}_{t,i}^\top \widetilde{V}_{t,i} \right\|_F \left\| \widetilde{V}_{t,j}^\top \widetilde{V}_{t,j} \right\|_F}, \tag{2}$$

where $\|\cdot\|_F$ denotes the Frobenius norm. High $S_t(i,j)$ indicates that the two blocks produce similar residual structure on task t.

For a contiguous interval $B(s,e) = \{s, \dots, e\}$, we define its coherence by the least similar pair of blocks it contains:

$$C_t(s,e) = \begin{cases} 1, & s = e, \\ \min\limits_{s \le i < j \le e} S_t(i,j), & s < e. \end{cases} \tag{3}$$

An interval is admissible when $C_t(s,e) \ge \tau_t$, where τ_t is the coherence threshold for task t; singleton intervals are always admissible. Let $\mathcal{P}_t$ be the set of admissible contiguous partitions that cover all L Transformer blocks exactly once. For a candidate partition $P_t = (B(s_1,e_1), \dots, B(s_{N_t}, e_{N_t}))$, N_t denotes its number of blocks. We select the task-specific partition by

$$P_t^* = \arg\min_{P_t \in \mathcal{P}_t} \left(N_t, -\sum_{n=1}^{N_t} C_t(s_n, e_n) \right), \tag{4}$$

where the pair is minimised lexicographically. The first criterion favours the most compact admissible partition, and the second favours higher total coherence among partitions with the same block count. We solve Eq. equation 4 with dynamic programming (Bellman, 1957), so both the block boundaries and N_t are determined by the task.

3.4 TASK-ADAPTIVE BLOCK ATTNRES

After structural discovery, the selected partition P_t^* is fixed. Let e_0 denote the embedding source, $c_{t,n}$ the residual representation committed for completed discovered block n, and p_l the partial residual representation of the current discovered block before Transformer block l. If block l lies in discovered block $B_{t,n} = B(s_n, e_n)$, the completed history is $\mathcal{H}_{t,l} = (e_0, c_{t,1}, \dots, c_{t,n-1})$.

At any routing site r, let $\mathcal{V} = (v_1, \dots, v_M)$ be the M currently available depth sources. The corresponding parameters in Q_t provide a pseudo-query $q_{t,r}$ and task-specific key normalisation. TA-IBAR computes

$$\begin{aligned} k_{t,r,j} &= \mathrm{RMSNorm}_{t,r}(v_j), \\ a_{t,r,j} &= \frac{\exp(q_{t,r}^\top k_{t,r,j})}{\sum_{m=1}^{M} \exp(q_{t,r}^\top k_{t,r,m})}, \\ R_{t,r}(\mathcal{V}) &= \sum_{j=1}^{M} a_{t,r,j} v_j, \end{aligned} \tag{5}$$

where $a_{t,r,j}$ is the depth-routing weight and $R_{t,r}(\mathcal{V})$ is the routed representation. TA-IBAR uses the No-Alpha formulation, supplying the routed representation directly to the corresponding transformation without an additional interpolation gate.

Within Transformer block l, the Attention site uses $\mathcal{V}_{t,l}^{\text{attn}} = \mathcal{H}_{t,l} \oplus (p_l)$ when a partial state exists and $\mathcal{V}_{t,l}^{\text{attn}} = \mathcal{H}_{t,l}$ at the start of a discovered block. The resulting Attention output is accumulated into both the native residual stream h_l and the current block residual. The MLP site then routes over $\mathcal{V}_{t,l}^{\text{mlp}} = \mathcal{H}_{t,l} \oplus (\widetilde{p}_l)$, where $\oplus$ denotes concatenation along the depth-source dimension and $\widetilde{p}_l$ is the partial block residual after the Attention transformation. Using Eq. equation 5 at both sites gives

$$\begin{aligned} z_{t,l}^{\text{attn}} &= R_{t,(l,\text{attn})}(\mathcal{V}_{t,l}^{\text{attn}}), & r_l^{\text{attn}} &= \text{Attention}_l\big(\text{LN}_{\text{in}}(z_{t,l}^{\text{attn}})\big), \\ \widetilde{p}_l &= \begin{cases} p_l + r_l^{\text{attn}}, & p_l \text{ exists}, \\ r_l^{\text{attn}}, & \text{otherwise}, \end{cases} & h_l^{\text{attn}} &= h_l + r_l^{\text{attn}}, \\ z_{t,l}^{\text{mlp}} &= R_{t,(l,\text{mlp})}(\mathcal{V}_{t,l}^{\text{mlp}}), & r_l^{\text{mlp}} &= \text{MLP}_l\Big(\text{LN}_{\text{post}}(z_{t,l}^{\text{mlp}})\Big), \\ p_{l+1} &= \widetilde{p}_l + r_l^{\text{mlp}}, & h_{l+1} &= h_l^{\text{attn}} + r_l^{\text{mlp}}. \end{aligned} \tag{6}$$

Here r_l^{attn} and r_l^{mlp} are the residual contributions produced by the Attention and MLP transformations, respectively. At the end of discovered block $B_{t,n}$, we commit $c_{t,n} = p_{l+1}$ and reset the partial state before entering the next discovered block. Transformer computation therefore remains fine-grained within each block, while cross-depth routing operates over the embedding, completed block representations, and the current partial block representation.

After the final Transformer block, the available sources are $\mathcal{V}_t^{\text{final}} = (e_0, c_{t,1}, \ldots, c_{t,N_t})$. Applying Eq. equation 5 once more yields the final routed state $u_t = \text{LN}(R_{t,\text{final}}(\mathcal{V}_t^{\text{final}}))$, which is passed to the language-model head.

3.5 Task-Specific Post-training and Inference

For task t, the routing bank is $\mathcal{B}_t = (P_t^*, Q_t)$. The discovered partition P_t^* remains fixed throughout post-training. Starting from the common backbone checkpoint $\theta_{\text{backbone}}^{(0)}$, we jointly optimise the backbone parameters θ_t and the task-specific routing parameters Q_t. For input sequence x, let $\mathcal{Y}_t(x)$ denote the positions whose target tokens contribute to the task-specific training loss. The post-training objective is

$$\mathcal{L}_t(\theta_t, Q_t) = -\frac{1}{|\mathcal{Y}_t(x)|} \sum_{k \in \mathcal{Y}_t(x)} \log p_{\theta_t, P_t^*, Q_t}\left(y_k \mid x_{<k}\right), \tag{7}$$

where y_k is the target token at position k. The backbone is updated more conservatively than the routing parameters, as specified by the separate learning rates in Appendix A.

At inference time, a lightweight probe (Alain & Bengio, 2016) predicts the task identity as $\hat{t} = g(x)$ and activates the corresponding task-adapted configuration. The probe is used only for configuration selection; its intermediate computation is discarded, and the original input is recomputed from layer 0 using the selected partition and routing parameters. The selected configuration remains fixed throughout generation.

4 Experiments

We evaluate TA-IBAR in both training-from-scratch and post-training settings, examining optimisation and memory efficiency, task-specific depth structures, generalisation across model scales, backbone families, and task domains, and the contributions of structural discovery and pseudo-query adaptation. Experiments span Qwen3 models from 1.7B to 14B on mathematical and multi-hop reasoning, five additional backbone families, code generation, and controlled ablations.

4.1 Experimental Setup

Training from scratch. We consider two model scales, 100M and 220M. Models are initialised from scratch and trained on FineWeb under matched data, optimisation, and compute budgets. We compare standard residual connections, Full AttnRes, fixed Block AttnRes, and TA-IBAR. Complete architecture and optimisation settings are reported in Appendix A.

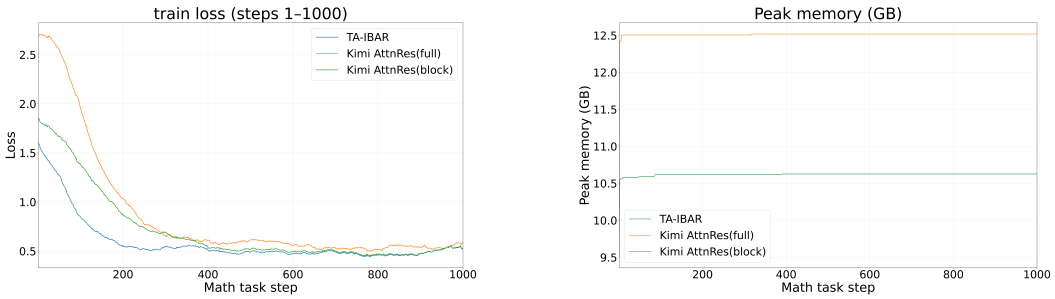

Figure 2: Optimisation and memory efficiency on Qwen3-1.7B. **Left:** training-loss trajectories under the same post-training protocol. **Right:** corresponding peak GPU memory consumption.

Table 1: Task-specific depth partitions discovered for mathematical and multi-hop reasoning. τ denotes the discovery threshold and N_t the resulting number of blocks.

Model	τ_m	P_{math}	N_m	τ_{mh}	P_{multi}	N_{mh}
Qwen3-1.7B	0.45	[3,4,4,3,1,4,6,1,1,1]	10	0.40	[1,2,2,5,4,1,5,4,2,1,1]	11
Qwen3-4B	0.50	[1,2,1,1,2,5,4,4,3,1,7,3,2]	13	0.43	[1,3,2,3,3,4,1,5,2,1,4,3,4]	13
Qwen3-8B	0.55	[3,2,2,5,4,1,1,1,2,1,2,9,1,2]	14	0.65	[1,2,1,1,2,5,4,1,1,1,4,10,1,2]	14
Qwen3-14B	0.50	[2,2,4,1,1,1,2,6,1,2,5,1,9,2,1]	15	0.55	[3,1,2,2,1,7,3,1,1,1,5,1,10,2]	14

Post-training. We evaluate pretrained Qwen3 models with 1.7B, 4B, 8B, and 14B parameters. Structural discovery is performed independently for each model–task pair, producing a task-specific partition P_t that is fixed during subsequent adaptation. Mathematical and multi-hop reasoning are evaluated at all four scales, while code generation is additionally evaluated at 14B. To test architectural generality, we further evaluate Granite-3.3-2B, SmolLM3-3B, Phi-4-mini-3.8B, Llama-3.2-3B, and Falcon3-3B. Unless otherwise stated, methods within the same model–task pair share the backbone checkpoint, training data, sample ordering, optimisation budget, and evaluation protocol.

4.2 Optimisation and Memory Efficiency

We begin with Qwen3-1.7B, where all residual formulations can be compared under a tightly controlled post-training protocol. Full AttnRes retains fine-grained access to historical representations but must maintain a larger set of depth-wise sources, whereas blockwise methods compress consecutive depth regions. Figure 2 jointly reports optimisation behaviour and peak GPU memory. TA-IBAR retains effective cross-layer optimisation while avoiding the full residual-state footprint of Full AttnRes. The key distinction from fixed Block AttnRes lies in the location of the compression boundaries. TA-IBAR makes these boundaries task dependent.

4.3 Training from Scratch

We train TA-IBAR from random initialisation at both model scales under matched data and optimisation conditions across residual formulations. TA-IBAR remains stable in this setting, showing compatibility with end-to-end training from scratch.

4.4 Post-Training Results

The central premise of TA-IBAR is that redundancy along depth depends on the target task. We therefore discover Math and Multi-hop partitions independently. Table 1 shows that the resulting structures differ substantially even when derived from the same backbone. Figure 3 provides a direct visualisation on Qwen3-1.7B: the two tasks induce visibly different segmentations, while the corresponding Linear-CKA matrices show that TA-IBAR merges locally coherent depth regions and preserves finer resolution around representation transitions. The discovered partitions are strongly non-uniform, with large blocks compressing locally redundant regions and task-relevant transitions remaining finely separated. Appendix C provides complementary evidence through direct CKA comparisons (Figure 6) and boundary-removal analysis (Figure 10).

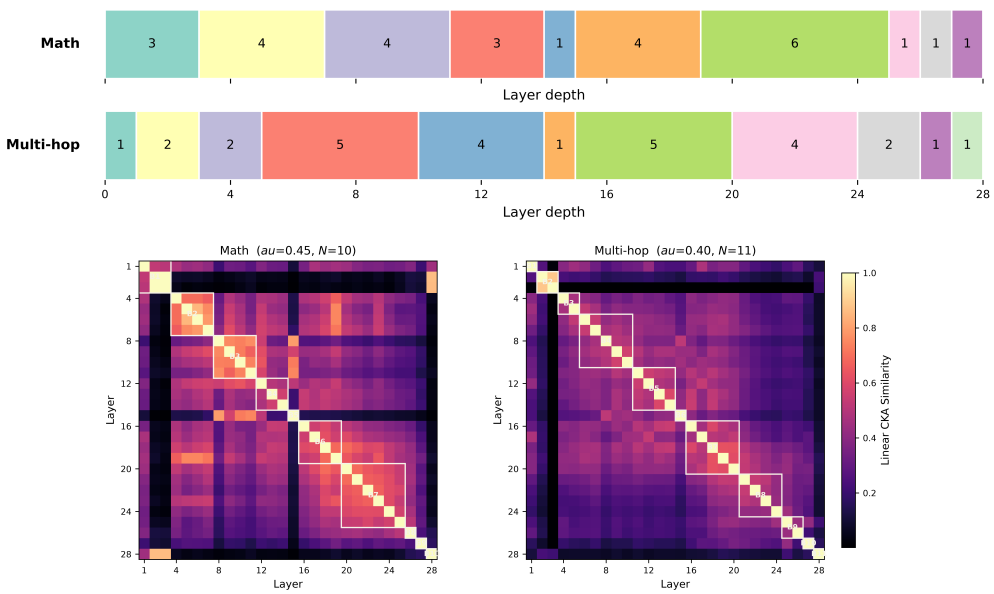

Figure 3: Task-specific structural discovery on Qwen3-1.7B. **Top:** discovered partitions for Math and Multi-hop. **Bottom:** Linear-CKA similarity matrices with the corresponding boundaries overlaid.

Table 2: Post-training results on mathematical reasoning. Mem. denotes peak GPU memory consumption.

Model	Method	Train Loss ↓	Valid Loss ↓	Mem. ↓
Qwen3-1.7B	Native	4.1505	0.4326	**10.08**
	Full AttnRes	4.0195	0.4762	12.52
	Block AttnRes	4.2797	0.4182	12.52
	TA-IBAR	**4.0187**	**0.4162**	10.51
Qwen3-4B	Native	0.6534	0.4347	**18.88**
	Full AttnRes	**0.6115**	0.4412	20.72
	Block AttnRes	0.7265	0.3915	20.72
	TA-IBAR	0.6122	**0.3697**	20.83
Qwen3-8B	Native	1.1268	1.0781	22.25
	Full AttnRes	0.9140	0.8743	20.01
	Block AttnRes	0.9319	0.9051	19.21
	TA-IBAR	**0.8962**	**0.8710**	**19.05**
Qwen3-14B	Native	0.9704	0.5982	**36.2**
	Full AttnRes	0.7530	0.4723	37.1
	Block AttnRes	0.7981	0.4810	36.6
	TA-IBAR	**0.7409**	**0.4629**	36.4

Table 3: Post-training results on multi-hop reasoning. Mem. denotes peak GPU memory consumption.

Model	Method	Train Loss ↓	Valid Loss ↓	Mem. ↓
Qwen3-1.7B	Native	8.8987	0.6926	**10.08**
	Full AttnRes	**4.3198**	0.2853	12.40
	Block AttnRes	6.2839	0.4271	12.40
	TA-IBAR	4.4262	**0.2445**	10.51
Qwen3-4B	Native	3.7486	3.1532	21.67
	Full AttnRes	0.2568	0.3643	24.94
	Block AttnRes	0.2759	0.3245	24.94
	TA-IBAR	**0.2371**	**0.3022**	**20.83**
Qwen3-8B	Native	11.0360	12.6283	19.50
	Full AttnRes	9.4783	9.8410	18.91
	Block AttnRes	9.5599	10.9455	18.85
	TA-IBAR	**9.4180**	**9.3487**	**18.78**
Qwen3-14B	Native	1.9140	10.2445	**41.4**
	Full AttnRes	1.5365	9.7270	42.3
	Block AttnRes	1.5397	9.7513	42.1
	TA-IBAR	**1.4976**	**8.9869**	41.6

We compare Native, Full AttnRes, fixed Block AttnRes, and TA-IBAR under matched post-training conditions. Tables 2 and 3 report the results across four Qwen3 scales. Across all four scales, TA-IBAR obtains the lowest validation loss on both tasks. At 14B, Math validation loss decreases from 0.4810 with fixed Block AttnRes and 0.4723 with Full AttnRes to 0.4629; Multi-hop decreases from 9.7513 and 9.7270 to 8.9869. These results show that task-adaptive compression can match or improve Full AttnRes while operating over a compressed residual history.

We evaluate TA-IBAR on Granite-3.3, SmolLM3, Phi-4-mini, Llama-3.2, and Falcon3 using a common protocol, with structural discovery performed independently for Math and Multi-hop. Complete results are reported in Appendix B, Table 11. Across the ten model–task settings, TA-IBAR achieves lower validation loss than fixed Block AttnRes in all ten and improves over Full AttnRes in eight. On Multi-hop, it obtains the lowest validation loss for all five backbones, showing that task-adaptive depth compression generalises beyond Qwen3.

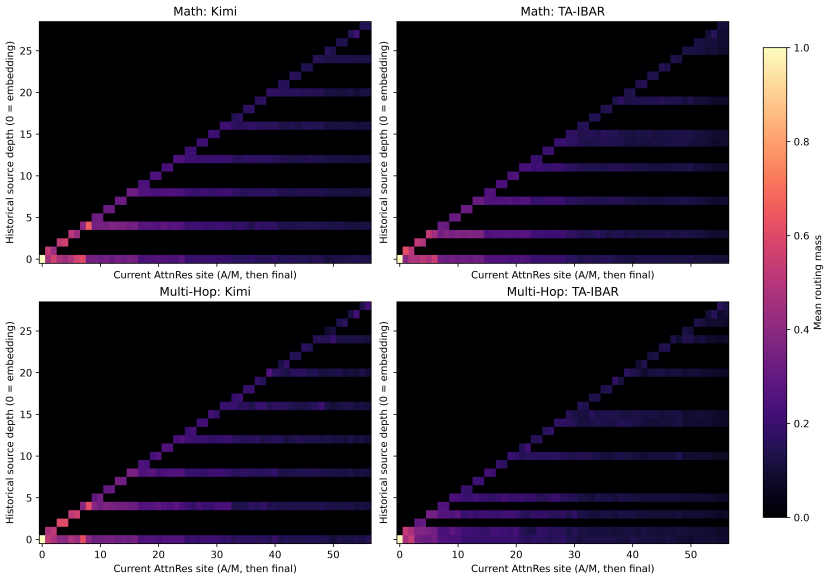

Figure 4: Mean depth-routing patterns for Math and Multi-hop reasoning. Rows denote historical residual sources and columns denote current AttnRes sites. TA-IBAR preserves strong local propagation while learning selective non-local retrieval over its task-specific block structure.

Table 4: Post-training results for code generation on Qwen3-14B. Mem. denotes peak GPU memory consumption.

Scale	Method	Train Loss ↓	Valid Loss ↓	Mem. (GB) ↓
14B	Native	1.1820	0.8420	**41.8**
	Full AttnRes	0.9250	0.6810	42.8
	Block AttnRes	0.9510	0.7040	42.5
	TA-IBAR (Ours)	**0.9010**	**0.6420**	42.0

Structural discovery determines which historical states remain accessible, while pseudo-query adaptation determines how those states are used. Figure 4 visualises the resulting routing patterns. The distributions retain strong diagonal structure while concentrating non-local retrieval on a smaller set of historical sources, and Math and Multi-hop exhibit distinct patterns that mirror their discovered partitions. Appendix C provides a more detailed view. Figures 8 and 9 visualise the resulting sparse long-range flows, Figure 7 shows high fidelity to Full AttnRes, and Figure 11 shows a more stable layer-wise representation trajectory than the full formulation.

We additionally evaluate code generation on Qwen3-14B using the same discovery–adaptation pipeline, without introducing a task-specific routing architecture. Table 4 reports the results. TA-IBAR achieves the lowest training loss (0.9010) and validation loss (0.6420), improving over Full AttnRes ($0.9250/0.6810$) and fixed Block AttnRes ($0.9510/0.7040$). Its 42.0 GB peak memory also remains below both AttnRes baselines.

4.5 Ablation Study

We conduct controlled ablations under the Qwen3-1.7B Math post-training setting. Unless otherwise specified, all variants share the same pre-adaptation checkpoint, examples, sample ordering, random seed, optimiser, learning-rate schedule, validation set, and 500K-token optimisation budget.

The discovery threshold τ controls structural granularity: lower values allow more aggressive compression, whereas higher values retain more explicit depth states. We sweep $\tau \in \{0.30, 0.40, 0.45, 0.50, 0.60\}$ while keeping the CKA matrix and optimisation protocol fixed. Tables 5 and 6 report the resulting partitions and post-training performance, respectively. Validation loss remains comparatively stable as the partition varies from 6 to 16 blocks, indicating robustness

Table 5: Depth partitions under different discovery thresholds for Qwen3-1.7B Math. N denotes the number of blocks.

τ	Partition	N	Max Block	Singletons
0.30	[3,11,1,11,1,1]	6	11	3
0.40	[3,4,4,3,1,2,9,1,1]	9	9	3
0.45	[3,4,4,3,1,4,6,1,1,1]	10	6	4
0.50	[3,4,5,2,1,4,6,1,1,1]	10	6	4
0.60	[1,2,4,4,1,1,1,1,1,1,3,4,1,1,1,1]	16	4	11

Sensitivity to the Discovery Threshold. The discovery threshold τ controls the granularity of the retained depth structure. Lower values permit more aggressive compression, whereas higher values retain more explicit depth states. We sweep $\tau \in \{0.30, 0.40, 0.45, 0.50, 0.60\}$ while keeping the CKA matrix and optimization protocol fixed. Increasing τ changes the number of blocks from 6 to 16, yet the validation loss remains relatively stable. This indicates that TA-IBAR does not rely on a narrowly tuned partition and remains effective across a broad range of structural granularities.

Table 6: Post-training behaviour under different structural discovery thresholds. Train Loss is averaged over the final 100 optimisation steps.

τ	Train Loss ↓	Valid Loss ↓	Mem. ↓
0.30	0.4968	0.4437	10.63
0.40	0.4711	0.4328	10.63
0.45	0.4583	0.4292	10.63
0.50	0.4351	0.4313	10.63
0.60	0.4629	0.4135	10.64

across substantially different structural granularities. This is consistent with the boundary analysis in Figure 10, where structural importance is concentrated at a subset of task-relevant boundaries.

We next isolate structural discovery from task-specific routing adaptation. In the *w/o Query Adaptation* variant, the discovered partition is unchanged but the pseudo-query remains frozen. Table 7 shows that freezing the pseudo-query substantially increases validation loss without changing memory consumption. Structural discovery therefore determines which depth states are retained, while pseudo-query adaptation is required to learn how those states should be combined.

Table 7: Effect of pseudo-query adaptation under the Qwen3-1.7B Math post-training setting.

Variant	Train Loss ↓	Valid Loss ↓	Mem. (GB) ↓
TA-IBAR	0.4292	0.8836	10.63
w/o Query Adaptation	0.4328	1.4781	10.63

Figure 5 compares Alpha and No-Alpha variants under identical backbone, partition, data, and optimisation settings. No-Alpha converges substantially faster during the early stage of adaptation and reaches a comparable final optimisation regime without introducing an additional interpolation pathway. We therefore adopt No-Alpha in the final TA-IBAR formulation, leaving structural discovery and pseudo-query optimisation as the only task-specific adaptation components. This also keeps the residual mechanism simpler while avoiding additional parameters and routing operations.

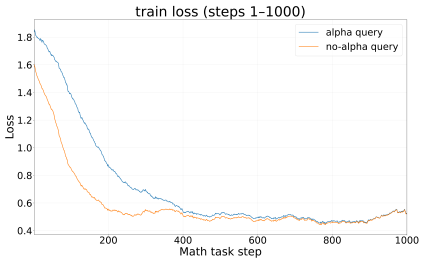

Figure 5: Alpha vs. No-Alpha adaptation on Qwen3-1.7B Math.

5 Conclusion

We introduced **TA-IBAR**, a task-adaptive formulation of Block AttnRes that discovers task-specific depth partitions and learns lightweight routing over the resulting residual structure. Across Qwen3 models from 1.7B to 14B, TA-IBAR achieves lower post-training validation loss than fixed Block AttnRes and Full AttnRes on mathematical and multi-hop reasoning while retaining the memory efficiency of blockwise residual access. The gains also extend across five additional backbone families and to code generation. Structural analyses show that TA-IBAR preserves task-relevant depth boundaries and supports selective long-range routing with high fidelity to Full AttnRes. These results demonstrate the effectiveness of adapting residual structure to the target task for efficient depth-wise information flow.

AI USE STATEMENT

Large language models and other AI tools were used for language editing, writing refinement, and retrieval of relevant literature during the preparation of this manuscript. Their use was limited to improving clarity, presentation, and supporting literature discovery. The research ideas, methodology, experimental design, implementation, analysis, and scientific conclusions were developed and verified by the authors.

REFERENCES

Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes, 2016.

Thomas Bachlechner, Bodhisattwa Prasad Majumder, Huanru Henry Mao, Garrison W. Cottrell, and Julian McAuley. ReZero is all you need: Fast convergence at large depth. *arXiv preprint arXiv:2003.04887*, 2020.

Richard Bellman. *Dynamic Programming*. Princeton University Press, 1957.

Wenzhuo Cheng, Shipeng Nie, Qixin Guo, Xuefeng Sun, Jianguo Lou, and Zhengwei Zheng. Deres: Decoupling residual stability and adaptivity for scalable ctr prediction. *arXiv preprint arXiv:2606.07980*, 2026.

Gucongcong Fan, Chaoyue Niu, Chengfei Lyu, Fan Wu, and Guihai Chen. Core: Reducing ui exposure in mobile agents via collaboration between cloud and local llms. *Advances in Neural Information Processing Systems*, 38:68690–68726, 2026.

Yanwen Fang, Yuxi Cai, Jintai Chen, Jingyu Zhao, Guangjian Tian, and Guodong Li. Cross-layer retrospective retrieving via layer attention. In *International Conference on Learning Representations*, 2023.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4700–4708, 2017.

Yanhao Jia, Jiepeng Wang, Haibin Huang, Chi Zhang, Erik Cambria, and Xuelong Li. Mmoe: Modernizing diffusion transformers with efficient expert design. *arXiv preprint arXiv:2607.24665*, 2026.

Kimi Team, Guangyu Chen, Yu Zhang, Jianlin Su, Weixin Xu, Siyuan Pan, Yaoyu Wang, Yucheng Wang, Guanduo Chen, Bohong Yin, Yutian Chen, Junjie Yan, Ming Wei, Y. Zhang, Fanqing Meng, Chao Hong, Xiaotong Xie, Shaowei Liu, Enzhe Lu, Yunpeng Tai, Yanru Chen, Xin Men, Haiqing Guo, Y. Charles, Haoyu Lu, Lin Sui, Jinguo Zhu, Zaida Zhou, Weiran He, Weixiao Huang, Xinran Xu, Yuzhi Wang, Guokun Lai, Yulun Du, Yuxin Wu, Zhilin Yang, and Xinyu Zhou. Attention residuals, 2026.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 3519–3529, 2019.

Gaurav Menghani, Ravi Kumar, and Sanjiv Kumar. LAuReL: Learned augmented residual layer. In *Proceedings of the 42nd International Conference on Machine Learning*, pp. 43826–43836, 2025.

Matteo Pagliardini, Amirkeivan Mohtashami, Francois Fleuret, and Martin Jaggi. DenseFormer: Enhancing information flow in transformers via depth weighted averaging. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 2227–2237, 2018.

Jonathan Su. Low-rank attention residuals. *arXiv preprint arXiv:2607.09694*, 2026.

TabLDM Team, Penghui Wang, Wei Liu, Hong Wang, Chengyue Huang, Yuxi Sun, Zirui Wang, Hongming Huang, Quan Wang, Chunxiao Liu, et al. Xiaomi-tabldm: A tabular foundation model technical report. *arXiv preprint arXiv:2609.03880*, 2026.

Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Herve Jegou. Going deeper with image transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 32–42, 2021.

Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Dongdong Zhang, and Furu Wei. DeepNet: Scaling transformers to 1,000 layers. *arXiv preprint arXiv:2203.00555*, 2022.

Xiao Wang. How much cache does reasoning need? depth-cache tradeoffs in kv-compressed transformers. *arXiv preprint arXiv:2604.17935*, 2026.

Zongwei Wu, Jilai Zheng, Xiangxuan Ren, Florin-Alexandru Vasluianu, Chao Ma, Danda Pani Paudel, Luc Van Gool, and Radu Timofte. Single-model and any-modality for video object tracking. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19156–19166. IEEE, 2024.

Da Xiao, Qingye Meng, Shengping Li, and Xingyuan Yuan. MUDDFormer: Breaking residual bottlenecks in transformers via multiway dynamic dense connections. In *Proceedings of the 42nd International Conference on Machine Learning*, pp. 68440–68458, 2025.

Zhenda Xie, Yixuan Wei, Huanqi Cao, Chenggang Zhao, Chengqi Deng, Jiashi Li, Damai Dai, Huazuo Gao, Jiang Chang, Kuai Yu, Liang Zhao, Shangyan Zhou, Zhean Xu, Zhengyan Zhang, Wangding Zeng, Shengding Hu, Yuqing Wang, Jingyang Yuan, Lean Wang, and Wenfeng Liang. mHC: Manifold-constrained hyper-connections, 2026.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. On layer normalization in the transformer architecture. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 10524–10533, 2020.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Yifan Zhang. Residual stream duality in modern transformer architectures. *arXiv preprint arXiv:2603.16039*, 2026.

Zhizhan Zheng, Feiyun Zhang, Shuchun Liu, Tian Xia, Xi Liu, Dasheng Hu, and Hongquan Zhou. Multi-gate residuals. *arXiv preprint arXiv:2605.23259*, 2026.

Defa Zhu, Hongzhi Huang, Zihao Huang, Yutao Zeng, Yunyao Mao, Banggu Wu, Qiyang Min, and Xun Zhou. Hyper-connections. In *International Conference on Learning Representations*, 2025.

Appendix

TABLE OF CONTENTS

A Training and Discovery Details **12**

A.1 Model and Optimisation Configurations 12

A.2 Per-Task Training Volume 12

A.3 Structural Discovery Configuration 13

A.4 Training and Adaptation Protocol 14

A.5 Controlled Ablation Protocol 14

A.6 Memory Measurement 15

B Cross-Family Generalisation **15**

B.1 Cross-Family Results 15

C Additional Structural Analysis **15**

C.1 Representation Structure 15

C.2 Residual Fidelity 15

C.3 Task-Adaptive Routing Flow 15

C.4 Boundary Importance 16

C.5 Layer-Wise Representation Drift 16

A TRAINING AND DISCOVERY DETAILS

This appendix reports the optimisation, data exposure, and structural discovery settings used in the experiments. Comparisons within each model–task pair use the same backbone checkpoint, training examples, sample ordering, optimisation budget, and evaluation protocol unless otherwise stated.

A.1 MODEL AND OPTIMISATION CONFIGURATIONS

Table 8 summarises the final model-specific optimisation settings used in the reported experiments. For post-training, we distinguish the learning rate applied to the pretrained backbone (BB LR) from that applied to the task-specific pseudo-query and key-normalisation parameters (Q/Norm LR). This separation allows the routing mechanism to adapt more rapidly while limiting perturbations to the pretrained representation space. Large-model runs use AdamW with $(\beta_1, \beta_2) = (0.9, 0.95)$ and $\epsilon = 10^{-8}$, cosine learning-rate decay with a 3% warmup ratio, and global gradient clipping at 1.0. Where separate parameter groups are used, backbone parameters receive weight decay 0.1, whereas pseudo-query and key-normalisation parameters use zero weight decay. Backbone computation is performed in BF16, while task-specific routing parameters are maintained in FP32. Large-model runs use FSDP `FULL_SHARD` with activation checkpointing. The 14B and cross-family configurations do not use CPU parameter offloading. The final TA-IBAR formulation uses the **No-Alpha** design, so no additional interpolation gate is inserted between the native and task-adapted residual pathways: structural discovery determines which historical depth states remain available, while the pseudo-query directly learns how to combine them.

A.2 PER-TASK TRAINING VOLUME

A fixed example pool and an optimisation budget are distinct quantities: examples may be revisited until the specified number of effective non-padding input tokens has been consumed. We therefore

Table 8: Model and optimisation configurations used in the paper. L denotes the number of Transformer blocks and d the hidden dimension. Tok./step denotes the target number of global non-padding input tokens per optimiser step. BB LR and Q/Norm LR denote the learning rates of the pretrained backbone and task-specific routing parameters, respectively.

Regime	Model	Params.	L	d	Max Len.	Tok./step	BB LR	Q/Norm LR	Precision	Hardware
Scratch	Scratch-Small	103.40M	12	448	2048	4096	3×10^{-4}	3×10^{-4}	BF16 + FP32 master	4×RTX 3090
Scratch	Scratch-Large	218.46M	18	672	2048	4096	3×10^{-4}	3×10^{-4}	BF16 + FP32 master	4×RTX 3090
Post-training	Qwen3	1.7B	28	2048	1024	4096	5×10^{-6}	3×10^{-5}	BF16 / FP32	2×A100
Post-training	Qwen3	4B	36	2560	1024	4096	3×10^{-6}	2×10^{-5}	BF16 / FP32	2×A100
Post-training	Qwen3	8B	36	4096	1024	4096	1×10^{-6}	3×10^{-5}	BF16 / FP32	4×RTX 3090
Post-training	Qwen3	14B	40	5120	2048	8192	1×10^{-6}	3×10^{-5}	BF16 / FP32	4×RTX PRO 6000
Cross-family	Granite-3.3	2B	40	2048	1024	8192	1×10^{-6}	3×10^{-5}	BF16 / FP32	4×RTX 3090
Cross-family	SmolLM3	3B	36	2048	1024	8192	1×10^{-6}	3×10^{-5}	BF16 / FP32	4×RTX 3090
Cross-family	Phi-4-mini	3.8B	32	3072	1024	8192	1×10^{-6}	3×10^{-5}	BF16 / FP32	4×RTX 3090
Cross-family	Llama-3.2	3B	28	3072	1024	8192	1×10^{-6}	3×10^{-5}	BF16 / FP32	4×RTX 3090
Cross-family	Falcon3	3B	22	3072	1024	8192	1×10^{-6}	3×10^{-5}	BF16 / FP32	4×RTX 3090

report post-training exposure directly in tokens. Table 9 gives the training volume for every model–task pair. Qwen3-1.7B uses 350K tokens per reasoning task, Qwen3-4B and Qwen3-14B use 1M, and Qwen3-8B and all five cross-family models use 500K. The Qwen3-14B Code experiment uses 500K tokens.

Table 9: Task-specific post-training data and effective training volume. Training Amount denotes the total number of global non-padding input tokens consumed by the corresponding run.

Model	Task	Training Data	Training Amount
Qwen3-1.7B	Math	SVAMP	350K tokens
Qwen3-1.7B	Multi-hop	CLUTRR	350K tokens
Qwen3-4B	Math	GSM8K + SVAMP	1M tokens
Qwen3-4B	Multi-hop	CLUTRR	1M tokens
Qwen3-8B	Math	GSM8K + SVAMP	500K tokens
Qwen3-8B	Multi-hop	CLUTRR	500K tokens
Qwen3-14B	Math	GSM8K + SVAMP	1M tokens
Qwen3-14B	Multi-hop	CLUTRR	1M tokens
Qwen3-14B	Code	MBPP + xCoder-80K	500k tokens
Granite-3.3-2B	Math	SVAMP	500K tokens
Granite-3.3-2B	Multi-hop	CLUTRR	500K tokens
SmolLM3-3B	Math	GSM8K + SVAMP	500K tokens
SmolLM3-3B	Multi-hop	CLUTRR	500K tokens
Phi-4-mini-3.8B	Math	GSM8K + SVAMP	500K tokens
Phi-4-mini-3.8B	Multi-hop	CLUTRR	500K tokens
Llama-3.2-3B	Math	GSM8K + SVAMP	500K tokens
Llama-3.2-3B	Multi-hop	CLUTRR	500K tokens
Falcon3-3B	Math	GSM8K + SVAMP	500K tokens
Falcon3-3B	Multi-hop	CLUTRR	500K tokens

A.3 Structural Discovery Configuration

TA-IBAR performs structural discovery before task-specific post-training. Discovery is executed on representations produced by the native residual backbone, without updating model parameters. Centred Linear CKA is computed in FP32 to characterise similarity across depth, after which a contiguous partition is derived from the saved similarity matrix. Table 10 reports the discovery configuration for every model and task for which TA-IBAR is applied. The threshold τ is model- and task-specific for the Qwen3 experiments. In the cross-family study, τ is fixed per backbone and shared between Math and Multi-hop to avoid task-specific hyperparameter tuning.

For Qwen3-14B Code, the 300-example discovery set is deterministically derived from the 500-example Code discovery pool while preserving its source composition: 224 examples are drawn from MBPP and 76 from xCoder-80K. Four selected xCoder prompts exceed the 2048-token con-

Table 10: Structural-discovery configurations used across model scales and families. Cases denotes the number of fixed examples used to estimate the Linear-CKA matrix. τ is the threshold used by the corresponding run, and N denotes the number of blocks in the resulting task-specific partition.

Model	Task	Cases / Data	Max Len.	τ	N	Metric	Partition
Qwen3-1.7B	Math	200 / Math pool	2048	0.45	10	Linear CKA	[3,4,4,3,1,4,6,1,1,1]
Qwen3-1.7B	Multi-hop	200 / CLUTRR	2048	0.40	11	Linear CKA	[1,2,2,5,4,1,5,4,2,1,1]
Qwen3-4B	Math	200 / GSM8K+SVAMP+MATH+OpenMath	4096	0.50	13	Linear CKA	[1,2,1,1,2,5,4,4,3,1,7,3,2]
Qwen3-4B	Multi-hop	CLUTRR	4096	0.43	13	Linear CKA	[1,3,2,3,3,4,1,5,2,1,4,3,4]
Qwen3-8B	Math	500 / GSM8K 250 + SVAMP 250	1024	0.55	14	Linear CKA	[3,2,2,5,4,1,1,1,2,1,2,9,1,2]
Qwen3-8B	Multi-hop	500 / CLUTRR	1024	0.65	14	Linear CKA	[1,2,1,1,2,5,4,1,1,1,4,10,1,2]
Qwen3-14B	Math	300 / GSM8K 150 + SVAMP 150	2048	0.50	15	Linear CKA	[2,2,4,1,1,1,2,6,1,2,5,1,9,2,1]
Qwen3-14B	Multi-hop	300 / CLUTRR	2048	0.55	14	Linear CKA	[3,1,2,2,1,7,3,1,1,1,5,1,10,2]
Qwen3-14B	Code	300 / MBPP 224 + xCoder-80K 76	4096	0.65	18	Linear CKA	[1,1,2,1,2,3,1,4,2,1,5,1,4,2,3,1,4,2]
Granite-3.3-2B	Math	300 / GSM8K + SVAMP	1024	0.60	18	Linear CKA	[1,1,1,1,3,2,2,1,7,1,6,4,1,2,1,1,1,4]
Granite-3.3-2B	Multi-hop	300 / CLUTRR	1024	0.60	10	Linear CKA	[2,1,1,3,2,8,16,1,2,4]
SmolLM3-3B	Math	300 / GSM8K + SVAMP	1024	0.60	10	Linear CKA	[1,1,3,13,7,6,1,1,2,1]
SmolLM3-3B	Multi-hop	300 / CLUTRR	1024	0.60	9	Linear CKA	[1,1,3,19,7,1,1,2,1]
Phi-4-mini-3.8B	Math	300 / GSM8K + SVAMP	1024	0.55	10	Linear CKA	[1,1,1,3,1,3,8,1,10,3]
Phi-4-mini-3.8B	Multi-hop	300 / CLUTRR	1024	0.55	11	Linear CKA	[1,1,1,1,3,3,8,1,10,2,1]
Llama-3.2-3B	Math	300 / GSM8K + SVAMP	1024	0.55	6	Linear CKA	[1,1,1,12,10,3]
Llama-3.2-3B	Multi-hop	300 / CLUTRR	1024	0.55	5	Linear CKA	[1,1,1,22,3]
Falcon3-3B	Math	300 / GSM8K + SVAMP	1024	0.50	6	Linear CKA	[1,7,5,7,1,1]
Falcon3-3B	Multi-hop	300 / CLUTRR	1024	0.50	6	Linear CKA	[1,1,8,10,1,1]

text used by the reasoning tasks, so Code discovery uses a 4096-token limit. No examples are replaced or resampled to satisfy the longer context requirement. The discovery pass computes the Linear-CKA similarity matrix, and threshold selection with dynamic-programming partitioning is performed offline from the saved matrix, so alternative structural choices do not require another forward pass.

For the cross-family experiments, discovery uses 300 fixed examples per task and a maximum sequence length of 1024. A single model-specific threshold is shared by both tasks: $\tau = 0.60$ for Granite-3.3-2B and SmolLM3-3B, $\tau = 0.55$ for Phi-4-mini-3.8B and Llama-3.2-3B, and $\tau = 0.50$ for Falcon3-3B. This shared-threshold design limits task-specific tuning in the cross-family evaluation.

A.4 Training and Adaptation Protocol

Structural discovery fixes the task-specific partition before post-training. The backbone and task-specific routing parameters are then jointly optimised using the separate learning rates reported in Table 8. Keeping the partition fixed prevents the discovered structure from changing during adaptation and supports controlled comparisons across routing configurations. Within each model–task pair, Native, Kimi Full, Kimi Block, and TA-IBAR use matched task data, data exposure, and evaluation protocols.

A.5 Controlled Ablation Protocol

All controlled ablations are conducted on Qwen3-1.7B under the Math post-training setting. Unless otherwise stated, the variants share the same pre-adaptation checkpoint, training examples, sample ordering, random seed, optimiser, learning-rate schedule, validation split, and 500K-token optimisation budget. The discovery-threshold ablation changes only the threshold used to convert a fixed Linear-CKA similarity matrix into a contiguous partition. The query-adaptation ablation keeps the discovered partition unchanged and varies only whether the task-specific pseudo-query is optimised. The Alpha versus No-Alpha study similarly holds the backbone, partition, data, and optimisation protocol fixed and modifies only the residual interpolation mechanism.

A.6 Memory Measurement

Peak GPU memory includes model parameters, gradients, optimiser states, activations, and materialised depth-routing state under the corresponding training protocol. The reported values are intended for comparisons among methods evaluated at the same model scale and on the same hardware; they do not represent hardware-independent complexity estimates.

B Cross-Family Generalisation

The main experiments use Qwen3 to provide controlled scaling comparisons from 1.7B to 14B. To test whether TA-IBAR generalises beyond this backbone family, we additionally evaluate five pretrained architectures with different model designs and depths: Granite-3.3-2B, SmolLM3-3B, Phi-4-mini-3.8B, Llama-3.2-3B, and Falcon3-3B. Math and Multi-hop are treated as independent adaptation tasks for each backbone. Structural discovery is performed separately for the two tasks, and each resulting partition is fixed during post-training under the common 500K-token protocol described above. The shared optimisation and discovery recipe limits architecture-specific tuning.

B.1 Cross-Family Results

C Additional Structural Analysis

This section provides additional analyses of TA-IBAR's representation structure, residual fidelity, routing flow, boundary importance, and layer-wise drift.

C.1 Representation Structure

Figure 6 compares the layer-wise representation structure of Kimi Block and TA-IBAR on Math. TA-IBAR forms block boundaries that better follow locally coherent representation regions, producing a more task-adaptive depth decomposition than the fixed block structure.

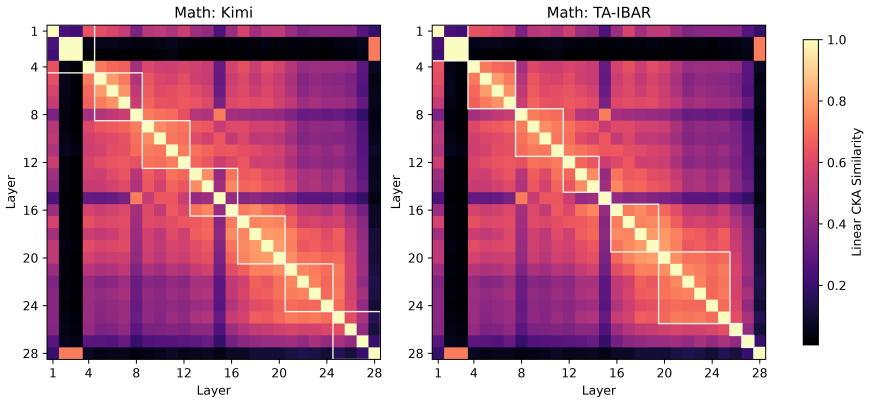

Figure 6: Layer-wise Linear-CKA similarity on Math. TA-IBAR produces a task-adaptive partition that more closely follows locally coherent representation regions than the fixed Kimi Block structure.

C.2 Residual Fidelity

Figure 7 compares Kimi Block and TA-IBAR against Full AttnRes on held-out Math and Multi-hop examples. TA-IBAR maintains high representation fidelity while operating over a more compact set of historical residual sources.

C.3 Task-Adaptive Routing Flow

Figures 8 and 9 visualise depth-wise residual routing. TA-IBAR preserves strong local information propagation while concentrating long-range routing on a smaller set of task-relevant historical

Table 11: Cross-family post-training results on mathematical and multi-hop reasoning. Train Loss denotes the mean training loss over the recorded optimisation trajectory. Valid Loss denotes validation loss, and Mem. denotes peak GPU memory consumption.

Model	Params.	Method	Train Loss ↓	Valid Loss ↓	Mem. (GB) ↓
Mathematical Reasoning					
Granite-3.3	2B	Native	0.6430	0.5974	**6.65**
		Kimi Full	0.5815	0.4130	6.95
		Kimi Block	0.6360	0.4586	6.95
		TA-IBAR (Ours)	**0.5991**	**0.3830**	6.70
SmolLM3	3B	Native	0.9458	0.7145	**8.98**
		Kimi Full	0.8643	0.6279	9.22
		Kimi Block	0.8701	0.6345	9.20
		TA-IBAR (Ours)	**0.8595**	**0.6192**	9.17
Phi-4-mini	3.8B	Native	0.9756	0.7105	**15.02**
		Kimi Full	0.8689	0.6294	15.25
		Kimi Block	0.8967	0.6813	15.25
		TA-IBAR (Ours)	**0.8463**	**0.5951**	15.18
Llama-3.2	3B	Native	2.0430	1.8978	**11.20**
		Kimi Full	1.2185	**0.9265**	11.66
		Kimi Block	1.2842	1.2146	11.65
		TA-IBAR (Ours)	**1.1431**	0.9682	11.14
Falcon3	3B	Native	1.2350	0.6383	10.64
		Kimi Full	0.9308	**0.4986**	10.75
		Kimi Block	1.0215	0.7619	10.73
		TA-IBAR (Ours)	**0.9224**	0.5157	**10.53**
Multi-Hop Reasoning					
Granite-3.3	2B	Native	0.5391	0.4962	6.48
		Kimi Full	**0.4278**	0.4794	6.97
		Kimi Block	0.4536	0.4851	6.31
		TA-IBAR (Ours)	0.4472	**0.3412**	**6.05**
SmolLM3	3B	Native	1.2038	0.5083	**8.98**
		Kimi Full	**0.9576**	0.4280	9.21
		Kimi Block	1.1962	0.4337	9.20
		TA-IBAR (Ours)	1.0345	**0.4226**	9.17
Phi-4-mini	3.8B	Native	0.9886	0.6773	**15.02**
		Kimi Full	0.9334	0.6391	15.25
		Kimi Block	0.9162	0.5937	15.25
		TA-IBAR (Ours)	**0.8570**	**0.5428**	15.18
Llama-3.2	3B	Native	1.6184	0.9525	11.20
		Kimi Full	**1.1473**	0.6412	11.67
		Kimi Block	1.2450	0.6302	11.65
		TA-IBAR (Ours)	1.2852	**0.6129**	**11.14**
Falcon3	3B	Native	**2.8283**	3.1761	10.64
		Kimi Full	3.2351	1.5816	10.77
		Kimi Block	3.5646	1.6239	1.76
		TA-IBAR (Ours)	2.8640	**1.5652**	**10.53**

sources. Compared with fixed Kimi Block, its cross-layer connectivity is more selective and task-dependent.

C.4 Boundary Importance

Figure 10 evaluates the contribution of individual TA-IBAR boundaries. The effects are strongly concentrated at a small subset of depth locations, showing that the discovered partition preserves a limited number of important structural separations while compressing less important regions.

C.5 Layer-Wise Representation Drift

Figure 11 compares representation drift across residual formulations. Full AttnRes produces substantially larger intermediate-layer drift, whereas TA-IBAR remains much closer to the compact Block AttnRes regime while introducing selective task-specific changes at key depths.

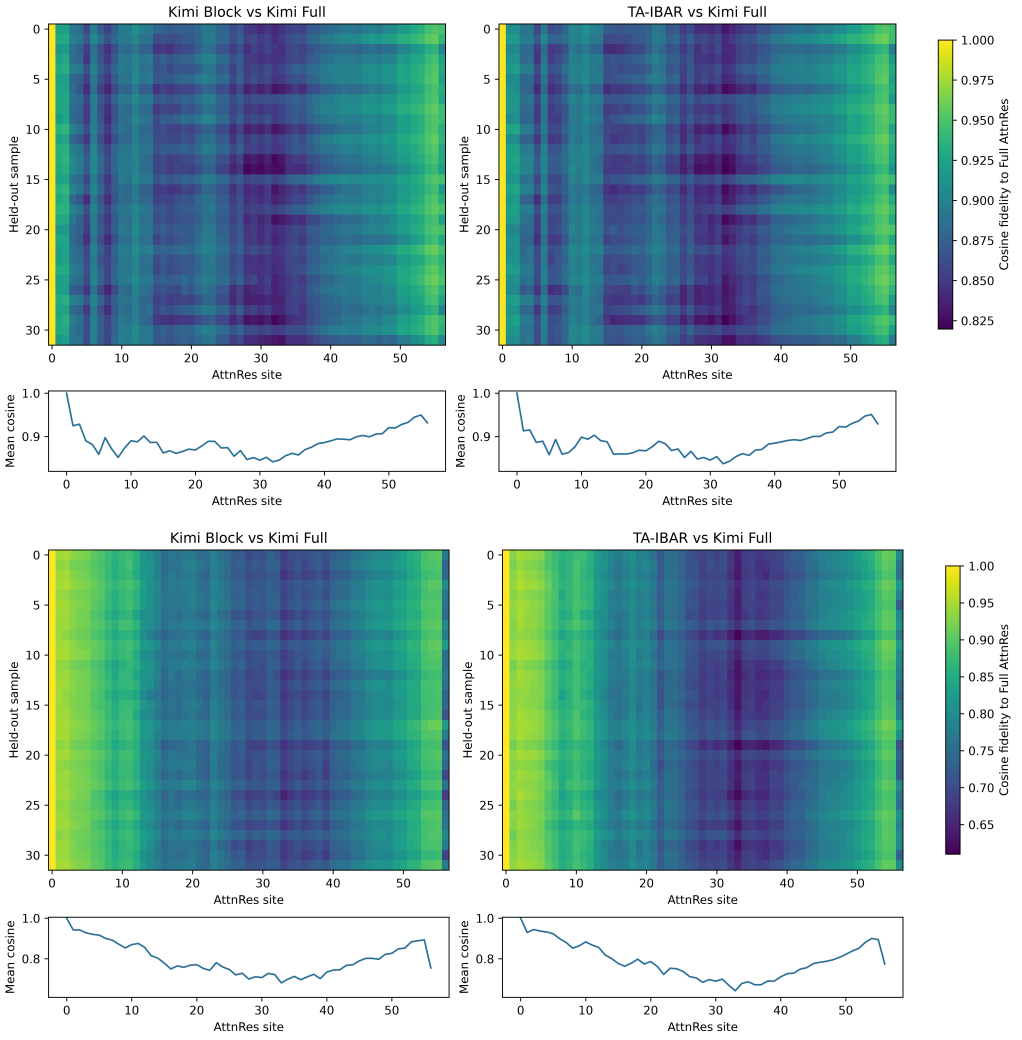

Figure 7: Residual representation fidelity to Full AttnRes. **Top:** Math. **Bottom:** Multi-hop. TA-IBAR preserves high fidelity while using a more compact task-specific residual structure.

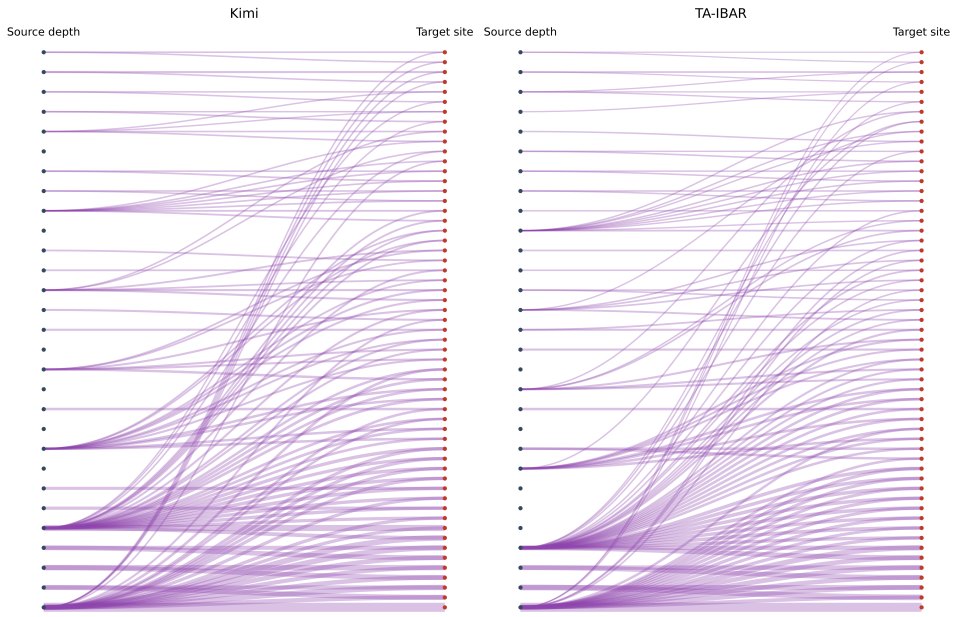

Figure 8: Depth-wise residual routing on Math. TA-IBAR exhibits more selective long-range routing over its task-specific block structure.

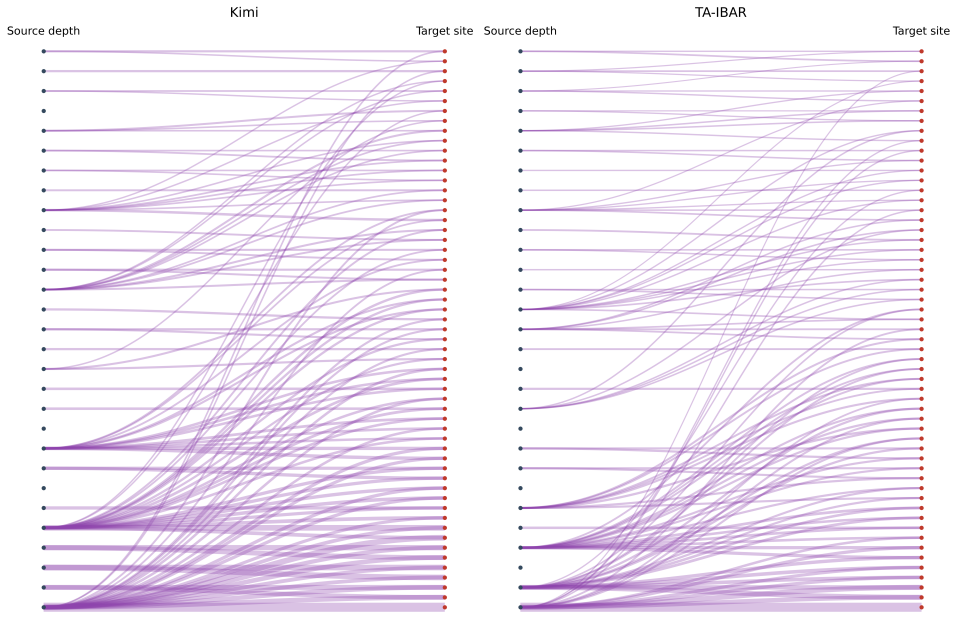

Figure 9: Depth-wise residual routing on Multi-hop. TA-IBAR forms a distinct task-specific routing pattern while retaining sparse long-range access to informative historical depth states.

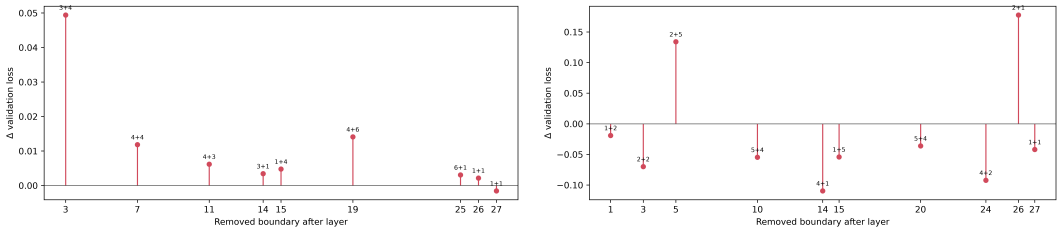

Figure 10: Boundary-removal analysis on **Math** (left) and **Multi-hop** (right). TA-IBAR concentrates structural importance on a small number of task-dependent boundaries, enabling aggressive compression elsewhere.

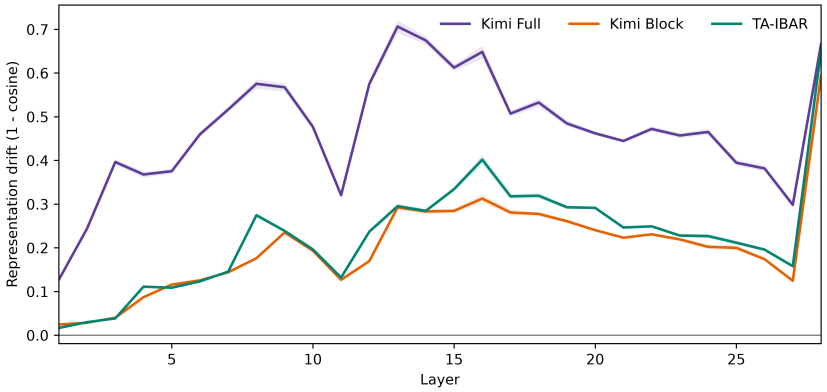

Figure 11: Layer-wise representation drift on Math. TA-IBAR maintains a more stable representation trajectory than Full AttnRes while allowing selective task-specific adaptation at intermediate depths.